



ASTOS[®] 9

Model Reference

©2006-2020

Astos Solutions GmbH
Meitnerstraße 8
70563 Stuttgart
Germany

This document is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in other ways, and storage in data banks. For any kind of use permission of the copyright owners must be obtained.

GESOP and ASTOS are registered trademarks of Astos Solutions GmbH.

Printed in Germany.

While every precaution has been taken in the preparation of this document, the publisher and authors assume no responsibility for errors or omissions, or for damages resulting from the use of the information contained herein.

December 2020

Preface

The *ASTOS* Software Project has been funded by the European Space Research and Technology Center (ESA/ESTEC), Noordwijk, Netherlands, under Contract No. 10136/92/NL/JG(SC) from 1989 - 2000. The contract monitor has been Klaus Mehlem.

Originally named *ALTOS* (Advanced Launcher Trajectory Optimization Software) the software evolved through various stages and with the participation of many individuals. In particular the following persons were directly involved in designing, implementing, testing and documenting the software:

Wolfgang Buhl, Klaus Ebert, Heinrich Herbst (Astrium GmbH, formerly DASA). Christian Jänsch, Zack Crues, Klaus Schnepfer, Eugen Berger (Deutsches Zentrum für Luft- und Raumfahrt, DLR). Denis Fischer, Peter Gath, Alexander Hauer, Jörg Kindler, Albert Markl, Michael Paus, Axel Roenneke, and Klaus Well (Institut für Flugmechanik und Flugregelung, Universität Stuttgart, IFR).

In addition, several people contributed indirectly by providing software modules: Philip Gill (University of Southern California, USC), Kathy Horn, Dieter Kraft (DLR), Michael Rahn, Ullrich Schöttle (Institut für Raumfahrtsysteme, Universität Stuttgart, IRS).

The contributions of all team members are highly appreciated. Without their commitment and dedication, the goal could not have been achieved. Most of all, the continuous support and feedback from Klaus Mehlem at ESTEC in these years has given stability to the development team such that new software engineering concepts could be implemented successfully.

Since the summer of 2000 *ALTOS* has been renamed to *ASTOS* (Aero- Space Trajectory Optimization Software) due to a trademark conflict. Until 2006, *ASTOS* has been a commercial software package sold by the TTI GmbH, Department of Optimization, Guidance & Control.

Since October 2006 the former TTI department "ASTOS Solutions (Optimization, Guidance & Control)" has been spun-off, such that now development, sales and support for *ASTOS* have been taken over by Astos Solutions GmbH.

Since 2015 - in line with the extended functionality of the software - the meaning of the *ASTOS* acronym has been changed to "Analysis, Simulation and Trajectory Optimization Software for Space Scenarios".

PREFACE

Table of Contents

1	Introduction	22
1.1	Scope of this Manual	23
2	Environment Models	25
2.1	Celestial Body (Shape, Gravity, Spin and Ephemeris Models)	25
2.1.1	Configuration	26
2.1.2	Background	32
2.2	Third Bodies Gravity Model	36
2.3	Atmosphere Models	37
2.3.1	Standard Atmosphere Models	38
2.3.2	U.S. Standard Atmosphere Models	38
2.3.3	Fitted U.S. Standard Atmosphere Model	39
2.3.4	No Atmosphere Model	40
2.3.5	Exponential Atmospheres	40
2.3.6	MSIS Atmospheres	41
2.3.7	User Atmosphere	42
2.3.8	Jacchia-Bowman Atmosphere Models	43
2.3.8.1	Jacchia-Bowman 2006 Atmosphere Model	44
2.3.8.2	Jacchia-Bowman 2008 Atmosphere Model	44
2.3.9	EMCD Atmosphere	45
2.3.10	GRAM-99 Atmosphere	46
2.4	Hydrosphere Model	48
2.5	Wind Models	50
2.5.1	Standard Wind Model	51
2.5.2	GRAM-99 Wind	53
2.6	Magnetic Field Models	54
2.7	Solar Radiation Pressure	55
2.8	Relativistic Effects	57
2.9	Gravity Gradient	58
3	Vehicle Parts and Properties Models	59
3.1	Vehicle Parts Common Characteristics	60
3.1.1	Common - Individual	60
3.1.2	Mass Distribution and Inertia Computation	62
3.1.3	Electrical Power	63
3.1.4	Thermal Characteristics	64

TABLE OF CONTENTS

3.1.5	Irradiance	66
3.2	Actuator Models	68
3.2.1	Clamp Band	68
3.2.2	Propulsion Models	69
3.2.2.1	Profile Rocket Engine	72
3.2.2.2	Throttleable Rocket	76
3.2.2.3	Linear Design Rocket	78
3.2.2.4	Basic Design Rocket	79
3.2.2.5	Throat Design Rocket	80
3.2.2.6	Typical Rocket with Variable Mixture Ratio	89
3.2.2.7	Solid Design Rocket	90
3.2.2.8	Airbreathing Engine	95
3.2.2.9	User-Defined Propulsion Model	96
3.2.3	Spring Damper Models	97
3.2.3.1	Translational Spring Damper	97
3.2.3.2	Rotational Spring Damper	98
3.2.4	Angular Momentum and Torque Actuator Models	100
3.2.4.1	Momentum Wheel	100
3.2.4.2	Magnetorquer	101
3.2.5	Generic Actuator	103
3.3	Component Models	104
3.3.1	Common Characteristics	105
3.3.1.1	Flexibility	105
3.3.1.2	Sloshing	107
3.3.1.3	Mass Computation	114
3.3.2	Vehicle Stage	115
3.3.2.1	Basic Vehicle Stage	117
3.3.2.2	Basic Vehicle Design Stage	119
3.3.2.3	Linear Vehicle Design Stage	121
3.3.2.4	Dual Propellant Stage	123
3.3.2.5	Dual Propellant Design Stage	124
3.3.2.6	Detailed Vehicle Design Stage	127
3.3.2.6.1	Mono Propellant	128
3.3.2.6.2	Dual Propellant	133
3.3.2.7	Detailed Vehicle Design Booster	141
3.3.2.7.1	Mono Propellant	142
3.3.2.7.2	Dual Propellant	147
3.3.3	Auxiliary, Payload and Fairing	154
3.3.3.1	Auxiliary	154

TABLE OF CONTENTS

3.3.3.2	Payload	155
3.3.3.3	Basic Design Payload Fairing	156
3.3.3.4	Detailed Design Payload Fairing	157
3.3.4	Reentry Vehicle Models	160
3.3.4.1	Sphere Cone	164
3.3.4.2	Capsule	164
3.3.4.3	Probe	165
3.3.4.4	Biconic	166
3.3.4.5	Ellipsled	167
3.3.5	User-Defined Component	168
3.3.6	Deployable Boom	169
3.4	Sensor Models	170
3.4.1	Camera Model	172
3.4.2	Generic Sensor Model	174
3.4.3	Inertial Measurement Unit Model	175
3.4.4	Magnetometer Model	175
3.4.5	MBS Sensor Model	177
3.4.6	Sun Sensor Model	178
3.4.7	Transmitter, Receiver and Transceiver Models	179
3.4.7.1	Simple	184
3.4.7.2	Isotropic	184
3.4.7.3	Complex	185
3.4.7.4	GNSS	188
3.5	Aerodynamics and Aerothermodynamics Models	195
3.5.1	Aerodynamics models	195
3.5.1.1	Tabular Aerodynamics	196
3.5.1.2	Hermes Reentry Vehicle Aerodynamics	204
3.5.1.2.1	Trimmed 3-DOF Hermes Aerodynamics	205
3.5.1.2.2	Trimmed 6-DOF Hermes Aerodynamics	207
3.5.1.3	Misdat Aerodynamics	210
3.5.1.4	SOSE Aerodynamics	213
3.5.1.5	User Aerodynamics	213
3.5.1.6	Bridging Functions	214
3.5.1.6.1	Error function	215
3.5.1.6.2	Sinusoidal function	215
3.5.1.7	Distributed Aerodynamics	216
3.5.2	Aerothermodynamics models	220
3.5.2.1	Convective heat flux	221
3.5.2.2	Radiative heat flux	223

TABLE OF CONTENTS

3.6	Electrical Power System Modelling	224
3.6.1	Battery Model	225
3.6.2	Power Control and Distribution Unit Model	228
3.6.3	Solar Generator Model	230
3.7	Data System Modelling	234
3.7.1	Data Buffer Model	236
3.8	Thermal System Models	236
3.8.1	Surface Element	236
4	Dynamics Configuration	238
4.1	Epoch, Time Scales and Time Formats	238
4.2	Initial State Definition	241
4.2.1	Constellations	242
4.2.2	Ground Stations	244
4.2.3	Vehicles	244
4.2.3.1	Position and Velocity State	244
4.2.3.2	Orbital Elements State	246
4.2.3.2.1	Keplerian Elliptic Orbit	247
4.2.3.2.2	Keplerian Hyperbolic Orbit	248
4.2.3.2.3	Keplerian Parabolic Orbit	250
4.2.3.2.4	Equinoctial Elements	251
4.2.3.2.5	Stationary Orbit	251
4.2.3.2.6	Sun-Synchronous Orbit	252
4.2.3.2.7	Sun-Synchronous Repeat Orbit	252
4.2.3.2.8	Ground Repeat Orbit	253
4.2.3.2.9	Frozen Orbit	254
4.2.3.2.10	Halo/Lissajous Orbit	254
4.2.3.2.11	Molniya Orbit	256
4.2.3.2.12	Two Line Elements (TLE)	257
4.2.3.2.13	Initial Deorbit	258
4.2.3.3	Relative Motion	261
4.2.3.4	Barycentric Rotating	261
4.3	Attitude Definition	262
4.3.1	Aerodynamic Angles	263
4.3.2	Aerodynamic Angles Controller 3-DOF	265
4.3.3	Docked	267
4.3.4	Euler Angles	267
4.3.5	Quaternion	270
4.3.6	Target Pointing	271
4.3.7	Unit Vector	272

TABLE OF CONTENTS

4.3.8	From File	272
4.4	Control Laws	273
4.4.1	Bilinear Tangent	275
4.4.2	Constant Law	277
4.4.3	Dynamic Pressure	277
4.4.4	Gravity Turn	280
4.4.5	Indirect	280
4.4.6	Inertial Tangential	280
4.4.7	Linear Law	281
4.4.8	Maximum Lift to Drag Ratio	282
4.4.9	Orbit Transfer	282
4.4.10	Pitch Over	283
4.4.11	Pointing Maneuver	285
4.4.12	Pollard	286
4.4.13	Profile Law	287
4.4.14	Radial	287
4.4.15	Required Velocity	288
4.4.16	Slip Free	289
4.4.17	Tangential	289
4.4.18	Target Orbit	289
4.4.19	Target Pointing	290
4.4.20	Target Velocity Pointing	291
4.4.21	Thrust Arc Schwer	292
4.4.22	Trimmed Aero	292
4.4.23	Vertical	293
4.4.24	Vertical Geodetic	293
4.4.25	Vertical Take Off	294
4.5	Equations of Motion	294
4.5.1	Translational Dynamics	295
4.5.1.1	Accelerate Flight Path Velocity States	296
4.5.1.2	Circular Restricted Three Body Problem	297
4.5.1.3	Circular Restricted Three Body Problem Analytical	298
4.5.1.4	Docked	299
4.5.1.5	Equinoctial Orbital Elements	299
4.5.1.6	Flight Path Velocity States	301
4.5.1.7	Inertial Cartesian Components	302
4.5.1.8	Inertial Velocity Components	303
4.5.1.9	Keplerian Orbital Elements	304
4.5.1.10	Launch Pad	305

TABLE OF CONTENTS

4.5.1.11	(Flexible) Multibody Dynamics	306
4.5.1.12	Perturbed Keplerian Orbital Elements	306
4.5.1.13	SGP4	307
4.5.1.14	State from File	308
4.5.2	Rotational Dynamics	308
4.5.2.1	Aerodynamic Angles	309
4.5.2.2	Euler Angles	310
4.5.2.3	Quaternion	311
4.5.3	Auxiliary States	311
4.5.3.1	Horizontal Range	312
4.5.3.2	Velocity Increment (Delta V)	312
4.5.3.3	Gravity Loss	312
4.5.3.4	Steering Loss	313
4.5.3.5	Heat Load	313
4.5.3.6	Average Drag Deviation	314
4.5.3.7	Trajectory Smoothness	315
4.5.3.8	Drag Loss	316
4.5.3.9	Total Impulse	316
4.5.3.10	Ozone Load	316
4.6	Maneuvers at the Beginning of a Phase	317
4.6.1	Impulse Maneuvers	318
4.6.2	Spin Impulse Maneuvers	320
4.7	Linearised Flexible Dynamics	320
5	Analyses	322
5.1	Chemical Plume Dispersion Analysis	323
5.1.1	Configuration	324
5.1.2	Background	325
5.1.3	Output	326
5.2	Data Timeliness Analysis	326
5.2.1	Configuration	326
5.2.2	Background	327
5.3	Destructive Reentry Analysis	327
5.3.1	DARS	328
5.3.1.1	Setup of DARS	328
5.3.1.1.1	DARS Explosion Model	329
5.3.1.1.2	Monte Carlo Analysis with DARS	330
5.3.1.1.3	DARS Material Table	331
5.3.1.1.4	DARS Trigger Table	331
5.3.1.1.5	DARS Fragment Table	332

TABLE OF CONTENTS

5.3.1.2	Simulation and Results Plotting	333
5.3.1.3	DARS Output Files	335
5.3.2	Ballistic Coefficient Analysis	335
5.3.2.1	Configuration	336
5.3.3	Risk Analysis	337
5.3.3.1	Data sources and calculation comments for the risk assessment module	337
5.3.3.2	Set up the risk assessment	339
5.3.3.3	Simulation and Results	341
5.4	DrNUM Aerodynamics Analysis	342
5.5	Earth Observation Mission Analysis	342
5.5.1	Configuration	343
5.5.2	Background	344
5.5.3	Output	344
5.6	Eclipse Analysis	346
5.6.1	Configuration	346
5.6.2	Background	347
5.6.3	Output	348
5.7	GNC Analysis	349
5.7.1	Configuration	349
5.7.2	Background	349
5.7.3	Output	350
5.8	Interplanetary Transfer Analysis	350
5.8.1	Configuration	351
5.8.2	Background	352
5.8.3	Output	353
5.9	Launch Safety Analysis	353
5.9.1	Configuration	354
5.9.2	Output	356
5.10	Link Budget Analysis	358
5.10.1	Configuration	359
5.10.2	Background	360
5.10.3	Output	366
5.11	Load Case	368
5.11.1	Configuration	369
5.11.2	Background	369
5.12	Navigation Analysis	372
5.12.1	Configuration	373
5.12.2	Background	375
5.12.3	Output	378

TABLE OF CONTENTS

5.13	Operational Lifetime Prediction Analysis	379
5.13.1	Configuration	380
5.13.2	Background	383
5.13.3	Output	386
5.14	Requirements Analysis	387
5.14.1	Configuration	388
5.14.2	Output	388
5.15	Rocket Motor Analysis	388
5.15.1	Configuration	389
5.15.2	Output	390
5.16	Tracking Analysis	392
5.16.1	Configuration	392
5.16.2	Output	392
5.17	Visibility Analysis	393
5.17.1	Configuration	393
5.17.2	Output	394
6	Phase-End Conditions and Optimization Constraints	395
6.1	Automatic Model Constraints	398
6.2	Position Related Conditions/Constraints	400
6.3	Velocity Related Conditions/Constraints	403
6.4	Attitude Control Related Conditions/Constraints	406
6.5	Forces Related Conditions/Constraints	408
6.6	Orbit Related Conditions/Constraints	408
6.7	Deorbit Related Conditions/Constraints	411
6.8	Integrated Quantity Conditions/Constraints	413
6.9	Phase Duration and Model Conditions/Constraints	414
6.10	Indirect Phase Conditions/Constraints	418
6.11	Inter-Satellite Conditions/Constraints	419
7	Cost Functions	422
7.1	Chebychev	424
7.2	Inter-Satellite Cost Functions	426
7.3	Max Ascending Node Delta	426
7.4	Max Crossrange, Max Downrange, Max Great Circle Range	427
7.5	Max DeltaV	428
7.6	Max Initial Value, Max Final Value, Max Initial Delta, Max Final Delta	428
7.7	Max Final Mass	430
7.8	Max Inclination Delta	431
7.9	Min Initial Propellant	432
7.10	Max Payload	432

TABLE OF CONTENTS

7.11	Max Safety	432
7.12	Max Splash Down	433
7.13	Min Heatload	433
7.14	Min Loss	434
7.15	None	434
7.16	Max Smoothness	435
7.17	User Model	436
8	Error Modelling	437
9	Coordinate Systems and Transformations	441
9.1	Definition of terms	442
9.2	Notation	444
9.3	Coordinate systems	445
9.3.1	Orbit-defining axis systems	452
9.3.1.1	Local orbit (LO)	452
9.3.1.2	Perifocal (PF)	454
9.3.1.3	Tangential orbit (TO)	455
9.3.2	Planetocentric axis systems	456
9.3.2.1	Mean Equator and Equinox of Epoch at J2000.0 (J2000)	456
9.3.2.2	Mean Ecliptic and Equinox of Epoch at J2000.0 (J2000Ec)	457
9.3.2.3	True Equator and Equinox of Date (TOD)	458
9.3.2.4	Planet-centered planet-fixed (PCPF)	460
9.3.3	Body-axis system	461
9.3.3.1	Body-axes (B)	462
9.3.3.2	Total angle of attack (TA)	463
9.3.3.3	Total force (TF)	464
9.3.4	Vehicle-carried axis systems	465
9.3.4.1	Local planetocentric (L)	466
9.3.4.2	Local planetodetic (LD)	467
9.3.4.3	Local horizontal (H)	468
9.3.4.4	Local vertical (V)	468
9.3.4.5	Sun-pointing (S)	469
9.3.5	Velocity-carried axis systems	470
9.3.5.1	Trajectory (T)	470
9.3.5.2	Vertical air path (AV)	471
9.3.5.3	Air path (A)	472
9.3.6	Other axis systems	473
9.3.6.1	Euler angles (EA)	473

TABLE OF CONTENTS

9.3.6.2	Launcher (FL)	474
9.3.6.3	R-bar/v-bar	475
9.3.6.4	Navigation (N)	476
9.4	Conversions and transformations	478
9.4.1	Orthogonal transformation	478
9.4.2	Conversion of velocities	479
9.4.3	Coordinate transformations	482
10	Physical Properties and Constants	484
11	User-Defined Models	485
11.1	Common Model Routines	487
11.1.1	Model Initialization Routine	488
11.1.2	Model Finalization Routine	489
11.1.3	Model Auxiliary States Routines	490
11.1.4	Model Control Variables Routines	494
11.1.5	Model Constraints Routines	498
11.1.6	Model Cost Functions Routines	503
11.1.7	Model Algebraic Functions Routines	505
11.1.8	Model Info Routines	508
11.2	User-Defined Actuator Model Routines	510
11.3	User-Defined Aerodynamics Model Routines	513
11.4	User-Defined Atmosphere Model Routines	516
11.5	User-Defined Component Model Routines	520

List of Figures

Fig. 2.1:	Scheme of the two-body problem	26
Fig. 2.2:	Section along the meridian of an oblate celestial body: definitions of latitude, declination, the local radii, and altitude	33
Fig. 2.3:	Wind direction and Cartesian components: definition with respect to a local axis coordinate system	53
Fig. 3.1:	Schematic of a surface element with four nodes and the considered cross sections for heat transfer	66
Fig. 3.2:	Example of a <i>Sun Sensor</i> with reflected sunlight from earth's surface. Reflected rays of sunlight are not considered in the irradiance if they are outside the field of view.	67
Fig. 3.3:	Correlation of the engine thrust and the engine chamber pressure	72
Fig. 3.4:	Unburned propellant as <i>Fraction of total propellant</i> : if <i>Mean</i> fraction is selected, the mass flow of unburned propellant is constant (left); if <i>Current</i> fraction is specified, the mass flow of unburned propellant is a fraction of the current total mass flow (right).	76
Fig. 3.5:	Engine geometry parameters [110]	85
Fig. 3.6:	Mass flow rate of the <i>Star</i> profile	92
Fig. 3.7:	Mass flow rate of the star profile with optimizable low mark	93
Fig. 3.8:	Mass flow rate of the <i>Star Advanced</i> profile	94
Fig. 3.9:	Mass flow rate of the <i>Tubular</i> profile	94
Fig. 3.10:	Hysteresis curve (blue) with the required inputs for the Ferromagnetic core (red)	103
Fig. 3.11:	Beam boundary conditions	107
Fig. 3.12:	Beam <i>CoM</i>	107
Fig. 3.13:	Sloshing model for cylindrical tanks	110
Fig. 3.14:	Viscous damping for a cylindrical tank with a spherical dome at the bottom	111
Fig. 3.15:	Sloshing model for spherical tanks	112
Fig. 3.16:	Height of the sloshing mass of the first natural frequency with respect to the free liquid surface in the spherical tank	113
Fig. 3.17:	Sloshing frequency parameter for a spherical tank	113
Fig. 3.18:	Viscous damping for a spherical tank	114
Fig. 3.19:	Definition of mono propellant stage geometry parameters	128
Fig. 3.20:	Centre of mass position of the payload adapter (and of a payload mass without specified dimensions)	130
Fig. 3.21:	Length definitions for thrust frame	132

LIST OF FIGURES

Fig. 3.22:	<i>Common Bulkhead</i> configuration (upper tank defined as complete tank) ...	135
Fig. 3.23:	<i>Separate Tanks</i> configuration and general design parameters	136
Fig. 3.24:	<i>Enclosed Tank</i> configuration	136
Fig. 3.25:	<i>Suspended Tank</i> configuration.	137
Fig. 3.26:	Center of mass position of the payload adapter (and of a payload mass without specified dimensions)	138
Fig. 3.27:	Length definitions for thrust frame	139
Fig. 3.28:	Definition of mono propellant booster geometry parameters	142
Fig. 3.29:	Length definitions for thrust frame	144
Fig. 3.30:	Length definitions for thrust frame	150
Fig. 3.31:	Booster tanks in <i>Common Bulkhead</i> configuration	151
Fig. 3.32:	<i>Separate Tanks</i> configuration and general booster design parameters	152
Fig. 3.33:	Enclosed tank booster configuration with the fuel tank in the forward position and the oxidizer tank as enclosed tank	152
Fig. 3.34:	Dimensions and shape of a <i>Detailed Design Payload Fairing</i> model	158
Fig. 3.35:	Dependency of the nose radius on the exponent n	159
Fig. 3.36:	TPS profile with heat flux and temperature definitions	162
Fig. 3.37:	<i>Sphere Cone</i> geometry	164
Fig. 3.38:	<i>Capsule</i> geometry	165
Fig. 3.39:	<i>Probe</i> geometry	166
Fig. 3.40:	<i>Biconic</i> geometry	167
Fig. 3.41:	<i>Ellipsled</i> geometry (side and front view)	168
Fig. 3.42:	<i>Camera</i> field of view	173
Fig. 3.43:	Signal modulation and polarization	181
Fig. 3.44:	<i>Astroview</i> screenshot showing a sensor cone angle (in blue) pointed to the Earth	182
Fig. 3.45:	Transmitter components diagram	182
Fig. 3.46:	Receiver components diagram	183
Fig. 3.47:	Antenna, transmission line and receiver for system temperature computation	183
Fig. 3.48:	Isotropic vs. directional antenna gain	185
Fig. 3.49:	Nominal angle of attack vs. Mach profile and angle of attack corridor	205
Fig. 3.50:	Lift coefficient versus Mach number and angle of attack	206
Fig. 3.51:	Drag coefficient versus Mach number and angle of attack	207
Fig. 3.52:	Lift coefficient vs. Mach number for constant angles of attack	208
Fig. 3.53:	Drag coefficient vs. Mach number for constant angles of attack	208
Fig. 3.54:	Lift-to-drag ratio vs. Mach number for constant angles of attack	209
Fig. 3.55:	Side force derivative $\partial C_f / \partial \beta$ vs. Mach number for constant angles of attack	209

LIST OF FIGURES

Fig. 3.56:	Roll moment derivative $\partial C_l / \partial \beta$ vs. Mach number for constant angles of attack	210
Fig. 3.57:	Yaw moment derivative $\partial C_n / \partial \beta$ vs. Mach number for constant angles of attack	210
Fig. 3.58:	Aerodynamic coefficients distribution	217
Fig. 3.59:	Cumulative coefficients distribution	217
Fig. 3.60:	Cumulative aerodynamic coefficients w.r.t. the vehicle length for total AoA = 5	219
Fig. 3.61:	Local force and moment vectors applied to the nodes along the vehicle length	220
Fig. 3.62:	Example of an electrical system using 2 solar generators as power sources, a battery as power source and power drain and 4 electrical consumers organized in two circuits with different voltages.	225
Fig. 3.63:	Typical discharge characteristics of a Panasonic NiMH HHR650D battery with fully charged voltage, exponential voltage (V exp.), exponential capacity (Q exp), nominal voltage (V nom.), nominal capacity (Q nom.) and cell capacity as well as discharge current.	227
Fig. 3.64:	Equivalent circuit of a solar cell with photo current I_{PH} , diode current I_D , shunt current I_{SH} , shunt resistance R_{SH} , series resistance R_S , terminal current I and terminal voltage V	232
Fig. 4.1:	Relationship between different time scales	241
Fig. 4.2:	Specification of a transfer orbit	259
Fig. 4.3:	Block diagram schematic of the attitude control system	266
Fig. 4.4:	Effect of the curvature parameter ξ for $a=100$. The boundary values are $\tan\Theta_0=0$, and $\tan\Theta_f=1$, respectively.	277
Fig. 4.5:	Dynamic pressure controller	279
Fig. 4.6:	Air path inclination angle γ and pitch angle Θ and their difference $\Delta\Theta$ over time during a pitch over maneuver	284
Fig. 4.7:	Inertial (XYZ) and co-rotating (xyz) reference frames.	298
Fig. 5.1:	Materials settings for the scenario	329
Fig. 5.2:	<i>Fragments</i> settings for the scenario	329
Fig. 5.3:	DARS scenario: 3D view	334
Fig. 5.4:	Impact swath as calculated by <i>RAM</i>	337
Fig. 5.5:	Eclipse geometry: partial, total, annular	347
Fig. 5.6:	Umbra/penumbra conditions	347
Fig. 5.7:	Flight corridor	357
Fig. 5.8:	Link budget functional blocks	359
Fig. 5.9:	Loss sources during propagation	361
Fig. 5.10:	External data required for the propagation losses	362
Fig. 5.11:	Atmosphere discretization	362
Fig. 5.12:	SNR computation diagram	366
Fig. 5.13:	Scheme for load case determination.	371

LIST OF FIGURES

Fig. 5.14:	Range measurement: signal transit time and clock synchronization error ...	375
Fig. 5.15:	Good (A) and poor (B) geometry	376
Fig. 5.16:	Example of a rocket motor contour obtained with <i>ASTOS</i>	389
Fig. 5.17:	Throat Design engine geometry [110]	391
Fig. 6.1:	Splash Down line constraint at the west coast of Africa: defining an upper limit of 0° forbids the area east of the lower blue line. A polygon as shown can only be defined using several constraints. But they would influence each other.	413
Fig. 6.2:	Illustration of the plume impingement constraint with a beamwidth of 20 degree	420
Fig. 6.3:	<i>R-bar Distance</i> and <i>V-bar Distance</i> between two vehicles	421
Fig. 9.1:	Definition of heading and elevation angles in polar vector representation ..	444
Fig. 9.2:	Orbit-defining coordinate systems: (1) Perifocal coordinate system (<i>index_p</i>) and (2) Local orbital coordinate system (<i>index_{LO}</i>)	447
Fig. 9.3:	Relative motion coordinate system	447
Fig. 9.4:	Definition of the Keplerian elements in an inertial coordinate system (here: true of date) and the perifocal coordinate system	448
Fig. 9.5:	Planet-centered and local horizontal coordinate systems: (1) True of date coordinate system (<i>index_{TOD}</i>), (2) Planet-centered planet-fixed coordinate system (<i>index_E</i>), (3) Local planetocentric coordinate system (<i>index_L</i>), (4) Local horizontal coordinate system (<i>index_H</i>), (5) Local vertical coordinate system (<i>index_V</i>)	448
Fig. 9.6:	The trajectory coordinate system (<i>index_T</i>)	449
Fig. 9.7:	Definition of the vertical air path coordinate system (<i>index_{AV}</i>)	449
Fig. 9.8:	Definition of the air path coordinate system (<i>index_A</i>)	450
Fig. 9.9:	Definition of the body-fixed coordinate system (<i>index_B</i>) and aerodynamic angles	450
Fig. 9.10:	Definition of the Euler angles in the local planetocentric coordinate systems	451
Fig. 9.11:	Definition of the aerodynamics related coordinate systems and their angles: (1) air path coordinate system (<i>index_A</i>), (2) body-fixed coordinate system (<i>index_B</i>), (3) total angle of attack coordinate system (<i>index_{TA}</i>), (4) total force coordinate system (<i>index_{TF}</i>)	451
Fig. 9.12:	Definition of the pitch $_{LO}$ angle in the x-y-plane (radial-transverse plane) ...	453
Fig. 9.13:	Definition of the out-of-orbit plane angle yaw_{LO}	453
Fig. 9.14:	Illustration of the R-bar/V-bar reference frame	476
Fig. 9.15:	Relationships between the inertial and Earth-fixed velocity vectors	481

List of Tables

Table 2.1:	Polynomial coefficients of the fitted atmospheric quantities	40
Table 2.2:	The PPN parameters and their interpretation	57
Table 3.1:	Available propulsion system models	70
Table 3.2:	Typical correlation coefficients for the engine model	80
Table 3.3:	Possible combinations for automatic engine mass correlation (S - Staged Combustion, G - Gas Generator, E - Expander Cycle P - Pressure Fed, A - Any)	83
Table 3.4:	Possible combinations for automatic Isp efficiency definition (S - Staged Combustion, G - Gas Generator, E - Expander Cycle P - Pressure Fed, A - Any)	84
Table 3.5:	Typical mass correlation coefficients for the engine model	87
Table 3.6:	Throat design engine output	88
Table 3.7:	Kinematic viscosity values of typical liquid propellant	108
Table 3.8:	Comparison of vehicle stage models	116
Table 3.9:	Typical correlation coefficients for the Basic Design Stage model	121
Table 3.10:	Typical correlation coefficients for the model	157
Table 3.11:	TPS material properties	163
Table 3.12:	Parameters for the Arrhenius equation	163
Table 3.13:	Geometry parameter definition	164
Table 3.14:	Geometry parameter definition	165
Table 3.15:	Geometry parameter definition	166
Table 3.16:	Geometry parameter definition	167
Table 3.17:	Geometry parameter definition	168
Table 3.18:	GNSS signal definition	190
Table 3.19:	GNSS transmitter antenna models	191
Table 3.20:	Cumulative aerodynamic coefficients in total AoA frame	218
Table 4.1:	WGS gravity models	258
Table 4.2:	Definition of flight path velocity state variables	305
Table 5.1:	Launcher classes	354
Table 5.2:	Level of damage expected at different overpressure values	356
Table 5.3:	Flight corridor line segment lengths [km]	357
Table 5.4:	Polarization mismatch	367
Table 5.5:	Error sources	377
Table 5.6:	Typical DOP values	379
Table 9.1:	Vehicle position as a set of Keplerian elements	442

LIST OF TABLES

Table 9.2:	Vehicle position as a set of modified equinoctial elements	442
Table 9.3:	Vehicle position w.r.t. a central body	442
Table 9.4:	Components of the flight path velocity and of the air path velocity	442
Table 9.5:	Vehicle orientation w.r.t. vehicle-carried coordinate systems	443
Table 9.6:	Vehicle orientation w.r.t. air path velocity	443
Table 9.7:	Aerodynamic forces and moments	443
Table 9.8:	Notations	445
Table 9.9:	Coordinate systems overview	445
Table 9.10:	Aeronautical standard terms and symbols for relative velocities	480
Table 9.11:	Transformations between orbital systems and planet-centered systems ..	482
Table 9.12:	Transformations between planet-centered systems and vehicle-carried systems	482
Table 9.13:	Transformations between velocity-carried and vehicle-carried systems	482
Table 9.14:	Transformations between body-fixed system and vehicle-carried systems	482
Table 9.15:	Transformations between aerodynamic systems and body-fixed systems	483
Table 10.1:	Physical constants	484
Table 10.2:	Physical properties	484
Table 11.1:	Index values of auxiliary variables	486

1 Introduction

The **Model Reference** book gives a complete and detailed explanation of all models and their options available in *ASTOS*. The library that provides these models is the heart of the *ASTOS* software. It is a modular software component suitable for mission analysis and trajectory simulation. Written in Ada 95 [1], its several hundred classes and several 100,000 lines of code provide the *ASTOS* user with a powerful, high-level programming and user data interface for aerospace applications. These model library components comprise

- translational and rotational equations of motion
- gravity, atmosphere, and wind
- aerodynamic forces and moments
- rocket and air-breathing propulsion systems
- solutions to orbital mechanics problems
- general purpose coordinate vectors and transformations
- a variety of utility functions

All terms and coordinate systems used throughout the library comply with Aerospace Standard ISO 1511 [2] and the ANSI Recommended Practice [3], [7].

When designing this library the goal was to provide the user with a collection of self-contained software modules that are easy to use as well as easy to extend. The design philosophy was to define modules which construct and manipulate encapsulated, yet extensible data types and relieve users of being familiar with unnecessary details. This concepts especially streamlines manipulating large amounts of user-specified data that are, for example, required by aerodynamics and propulsion models. We also organized different modeling features into separate software modules so that the library can be extended without the need to modify existing components.

As a consequence, *ASTOS* application programs are provided with elaborate data structures and powerful operations to translate complex mathematical operations easily into code.

For example, objects representing flight mechanics vector quantities and general consistent transformation methods make it easy to set up dynamic systems or manipulate forces defined in various different coordinate systems. Other packages simplify the access to complex user input data by providing objects to store and specific methods to evaluate the information. These packages also facilitate data analysis and are useful in other stand-alone applications.

INTRODUCTION

From *ASTOS 7* on the user input is saved in XML files instead of Viola files (all previous versions employed the I/O language system *VIOLIN*). It is no longer required for the user to manually program these input files: all input is now provided to *ASTOS* through the GUI, *ASTOS'* input graphical interface. This results in a user-friendly interface between user and program, as well as in a faster and more robust way to save user input.

1.1 Scope of this Manual

This Reference Manual is meant to give a detailed description of all *ASTOS* models. It provides important information for both users of *ASTOS* applications and software engineers designing a new *ASTOS* application. Users of *ASTOS* applications find a technical description of the modeling features and a detailed description of the input data options. Additionally, a description of the programming interface is provided for software engineers.

Information intended for users of *ASTOS* applications does not require knowledge of a programming language since the user is assisted by the GUI to create a model of a scenario.

The *ASTOS* Model Library and its documentation provides several main categories of model areas each containing several specific model packages:

- **Mission Definition** defining the mission to be accomplished - like cost functions and constraints.
- **Phase Configuration** describing the sequence of phases and which model parts are used therein.
- **Model Definition** giving a collection of model objects and initializing them.
- **Vehicle Component Models** models of vehicle stages, tanks, and for stage design defined by user data.
- **Propulsion Models** models to approximate user-specified propulsion system data and models of particular vehicles.
- **Aerodynamics Models** models to approximate user-specified aerodynamic data and models of particular vehicles.
- **Environment Models** models describing the planet environment, its atmosphere, wind, shape, and acceleration of gravity, based on user data.
- **Cost Functions** describing in a detailed way the available cost functions.
- **Constraints Data** defining the constraint packages that can be selected by the user.
- **Controlled Forces** attitude control estimates and transformation of external forces and moments into principal coordinate systems.
- **Dynamics** equations of motion and other differential equations and quadratures.
- **Analytical Trajectory Propagation** methods to solve orbital mechanics problems
- **Coordinate Vectors and Transformations** methods to define coordinate vectors, coordinate transformation, and change of reference frames.

INTRODUCTION

- **User-Defined Models** providing information on the user-defined models (propulsion, component, aerodynamics or atmosphere).
- **Miscellaneous** physical constants, heat-flux computation, interpolation data format and data import.
- **Output** describing the data output possibility of *ASTOS*
- **Aerospace Terms and Coordinate Systems** contains a summary of the definitions of terms and coordinate systems that are used throughout this document.

2 Environment Models

The vehicle environment in *ASTOS* consists of the following models which can be combined within a scenario:

- **Celestial Body** models the shape, gravity, spin state and ephemerides of a celestial body, see Section 2.1
- **Third Body Gravity** models the effect on the spacecraft motion due to the presence of a third body gravitational field, see Section 2.2
- **Atmosphere Models** compute the atmospheric properties like density, pressure, temperature, speed of sound etc. for a given time and location, see Section 2.3
- **Hydrosphere** considers the presence of water in the atmosphere within a link budget analysis, see Section 2.4
- **Wind Models** consider the wind velocity of the atmosphere, see Section 2.5
- **Magnetic Fields** model the magnetic field of the planet Earth and compute the magnetic flux density for a given time and location, see Section 2.6
- **Solar Radiation Pressure** models forces and the resulting accelerations due to the interaction of the solar radiation with the spacecraft surface, see Section 2.7
- **Solar Radiation Torque** considers the torques due to the solar radiation pressure in the spacecraft dynamics, see Section 2.7
- **Relativistic Effects** model the additional acceleration on the spacecraft due to its motion in the gravitational field of a massy celestial body, see Section 2.8
- **Gravity Gradient** models the torques due to the different values of the gravity acceleration for different parts of a spacecraft, see Section 2.9

2.1 Celestial Body (Shape, Gravity, Spin and Ephemeris Models)

This section contains the description of the *Celestial Body* model, its configuration and underlying mathematical models. The celestial body model allows the user to introduce a celestial body into a scenario and configure in detail its shape, gravity field, spin state and ephemeris. While the model can be quickly configured by selecting a solar system planet from a list, the user may also set it up according to his individual needs.

ENVIRONMENT MODELS

Except for few exceptions, most of the equations of motion available in *ASTOS* (e.g. *Circular restricted three body problem*, *Docked*, *State from file*, etc) deal with the classical two-body problem (perturbed or unperturbed), which can be reduced to the central-force problem. In classical mechanics, the central-force problem is to determine the motion of a point mass under the influence of a single central force, that points from the point mass towards a fixed point in space, the force center, and whose magnitude only depends on the distance of the point mass to the center.

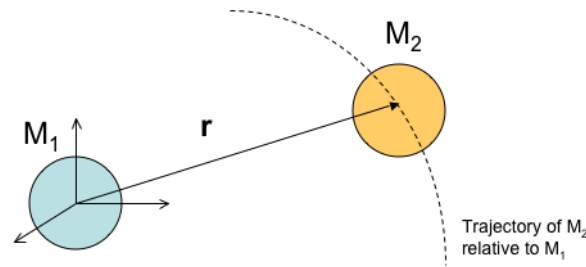


Fig. 2.1: Scheme of the two-body problem

The gravity acceleration due to the spherical gravity field of a point mass celestial body can be expressed by the following equation:

$$\vec{a}_{CB} = -\mu \frac{\vec{r}}{r^3} \quad (2.1)$$

where a_{CB} is the acceleration that M_2 (vehicle) experiences due to the gravity field of M_1 (central body), μ is the gravity parameter of the central body and r is the position vector of the vehicle. In case of a non-spherical gravity field, the induced acceleration can be computed as explained in Section 2.1.2.

In case additional perturbations (e.g. third-body, etc) need to be considered, their accelerations a_{Pert} are to be added to the gravity acceleration as follows:

$$\ddot{\mathbf{r}} = \mathbf{a}_{CB} + \mathbf{a}_{Pert} \quad (2.2)$$

2.1.1 Configuration

The *Celestial Bodies* node in the *Modelling* tree accommodates all instances of the celestial body model within the scenario. A new celestial body instance can be created and added to this list by using the corresponding button in the ribbon *Add*→*Environment Models*→*Celestial Bodies*. In order to configure a celestial body instance, please click on one of them in the list of all created instances and the corresponding configuration panel will be shown on the right side of the GUI. The modelling aspects behind the celestial body model settings are explained in Section 2.1.2. The configuration panel contains the following properties:

- The *IAU ID* list contains the pre-defined celestial bodies in *ASTOS*: the planets of the solar system and the Moon. By selecting one of these celestial bodies the corresponding values

ENVIRONMENT MODELS

for its shape and gravity parameter are set. By selecting the option *Other*, the user can define his own celestial body.

- The *Shape* panel allows the definition of the shape model for the celestial body. One of the following models can be selected from the respective drop-down list:

- *Sphere* requires only the *Radius* of the celestial body to be specified.
- *Spheroid* supports different parameters depending on how the oblateness of the body is defined using the respective drop-down list:
 - *Equatorial radius* defines the radius of the big circle of zero latitude.
 - *Polar radius* defines the distance from the body's center to one of its poles.
 - *Flattening factor* can be provided as an alternative to polar radius. The dimensionless flattening factor of the ellipsoid is defined as:

$$f = \frac{r_{equ} - r_{pol}}{r_{equ}} \quad (2.3)$$

Tip: The default values in the *Shape* panel are updated when a different item in the *IAU ID* list is selected.

- The *Gravity* panel allows the configuration of the gravity model. Its type is selected from the *Type* drop-down list. Three different options are available:

Tip: A default *Gravity parameter* value is set when an item in the *IAU ID* list is selected.

- *Spherical* requires only the *Gravity parameter* since the celestial body is treated as a point mass in terms of the gravity field it generates.
- *Ellipsoid* requires the definition of the *Gravity parameter* and allows the specification of values for the dimensionless coefficients *J2* to *J6* (for details see Section 2.1.2), which take into consideration the effect of the first and higher order body oblateness and the longitudinal variations in the body shape when computing the gravity force.

Tip: The selection of the central body gravity field as well as the definition of the perturbations to be accounted for is essential in order to precisely propagate the trajectory of a vehicle. A general rule of thumb is that the perturbation resulting from the *J2* term is the dominant perturbation in Low-Earth Orbit (LEO) and Medium-Earth Orbit (MEO), followed by atmospheric drag in LEO. In the upper MEO region and higher Earth orbits, third-body gravitational effects are of the same order of magnitude as those resulting from the *J2* term so that gravitational perturbations due to the Moon and the Sun should be considered in this case.

- The *Spherical Harmonics* setting gives the user the possibility to model the gravity field with a very high accuracy. The available input settings are:

- *Filename* specifies the path and name of the file containing the table of *Cos and Sin* coefficients (also called *C S Table*) [62] required to configure the spherical harmonics model. A description of the *C S Table* text file format is given below:

C S Table Data File Format

- First line (space-separated data):
 - Maximum degree
 - Maximum order
 - Reference radius
 - Gravitational parameter of the body (i.e. body mass times gravitational constant)
- Following lines (space-separated data) for each data set:
 - Degree
 - Order
 - Related C coefficient
 - Related S coefficient

C S Table Data files are provided with the *ASTOS* installation. Files according to different standards can be found in the folder `/lib/gravity_fields`.

- *Gravity param. input* allows the user to select which value for the gravity parameter is used. In case of *File* the value is read from the file specified in the *Filename* field. In case of *User* the value provided in the GUI is used.
- The *Used degree/order* setting defines up to which degree and order the gravity coefficients are read from the file specified above.

Note: The maximum supported degree and order is 150.

- The *Normalized Coefficients* setting specifies if the gravity coefficients read from file are treated as normalized coefficients.
- The *Tides* panel allows the user to consider the effects of the ocean and solid tides on the Earth gravity. These effects are modelled by adjusting the spherical harmonics coefficient values of the gravity model. In order to compute the tides effect the scenario needs to contain celestial bodies with both IAU identifier *Sun* as well as IAU identifier *Moon*. Otherwise a warning is displayed in the *Execution log* and the tides computation is deactivated! Both the ocean tides and the solid tides require a *Doodson corrections file* for the computation of the tides frequency-dependent corrections (see [136]). The file format is explained below. The *Tides* panel contains the following settings:

- In case the *Compute solid tides* option is *enabled* then the effects of the Earth solid tides on the gravity field of the planet are considered. A *Doodson corrections file* has to be provided by the user.

Doodson corrections data file format

The values for the coefficient data text file have to be arranged as follows (space-separated):

- First line:
 - Degree of the Love number coefficient to be corrected (0, 1 or 2)
- Following lines for each Love number coefficient:
 - Frequency [deg/hr]
 - Doodson number
 - Multipliers of the Doodson arguments (τ , s , h , p , N' , p_s)
 - Multipliers of the Delaunay variables (l , l' , F' , D , Ω)
 - Real and imaginary parts values of the amplitude corrections which will be internally multiplied by a factor of $10E-5$
 - In-phase and out-of-phase amplitude correction values which will be internally multiplied by a factor of $10E-12$
- In case the *Compute ocean tides* option is *enabled* then the effects of the Earth ocean tides on the gravity field of the planet will be considered. This option requires that both a *Doodson corrections file* and a *Ocean tides file* be provided by the user (see [136]).

Ocean tides data file format

The coefficient values in this data text file have to be arranged as follows (space-separated):

- First two lines:
 - Maximum degree of tides corrections minus one (i.e. n)
 - Water density [kg/m³]
- The following n lines contain the following space-separated values:
 - Order
 - Load deformation coefficient
- Following lines (list of tide constituents):

ENVIRONMENT MODELS

- Constituent format 'oap' (ocean tides coefficients) or 'ocs' (ocean tides amplitude and phase according to Schwiderski notation)
 - Order
 - Degree
 - 'pro' or 'retro' string for prograde or retrograde wave constituent
 - Doodson number / 1000.0
 - Constituent amplitude [m] ('oap') or C_{snm} coefficient [m] ('ocs')
 - Constituent phase [deg] ('oap') or S_{snm} coefficient [m] ('ocs').
- *Polyhedron* allows the user to set up a gravitational model for the celestial body which is computed as if the body shape is a constant-density polyhedron [153].

Note: The constant-density polyhedron is used solely for the computation of the gravitational field and has no effect on other settings (e.g. celestial body shape).

A polyhedron might include depressions such as craters, overhangs and even holes. This model is well suited for irregular-shaped bodies such as asteroids or comets. Since the polyhedron model is a valid and exact solution up to the surface of the body, it is also well suited for simulations of landing scenarios and close approaches of a spacecraft w.r.t. such an irregular-shaped body.

The shape of the body is given through the specified *File format*:

- *OBJ* is a widely known and used 3D geometry definition file format. This open and simple data-format contains the position of each vertex and the faces that make each polygon.

Note: Only triangular shaped faces are supported by the polyhedron gravity field model.

These settings are required:

- *Filename* is the path and name of the OBJ file
- *Unit of vertex coordinates* is required since the OBJ coordinates have no units. The provided unit scales the dimensions of the body, its volume and thus its gravitational parameter. In case of celestial bodies, *Kilo-Meter* is typically used as unit of the vertex coordinates in an OBJ file.
- *Density* specifies the constant density of the body

Tip: Since the computational effort for this model is high and both initialization and evaluation may take a non-negligible amount of time, it is suggested to use polyhedrons consisting of not more than few 100,000 faces.

- The *Spin State* setting allows the configuration of the celestial body spin state (for details on the spin state see Section 2.1.2). If enabled the spin state model *Type* can be chosen from the respective drop-down list:
 - The *IAU* spin state model requires the following parameters to be configured:
 - *Rotation* defines the location of the prime meridian according to [76]
 - *Right ascension* is the first setting defining the direction of the north pole according to [76]
 - *Declination* is the second setting defining the direction of the north pole according to [76]

Each of these items requires the following settings to be made:

- *Offset angle*: has to be specified for the epoch J2000.0 (Julian date = 2451545.0)
- *Rate* is the angular rate
- *Acceleration* is the respective angular acceleration
- *Sine Terms*: optional

For additional information about the inputs and underlying definitions refer to [76]. In case this model is used, the reference frames TOD, TEME and MOD are considered identical to each other [76].

- The *Earth IAU 76* precise spin state model utilizes the 1980 IAU theory of nutation and the IAU 1976 precession model. No further user settings are required.

Note: If *Spin State* is disabled, a predefined IAU model of the Earth with values taken from [76] is used.

- *Ephemeris* data is required if at least one of the following settings is used:
 - Gravitational perturbations due to third bodies
 - Relativistic effects
 - The Jacchia-Bowman 2006 (*JB2006*) model is used as atmospheric model
 - Solar radiation pressure is used in at least one vehicle

When the *IAU ID* is one of *Mars*, *Jupiter*, *Saturn*, *Uranus*, *Neptune* or *Pluto*, the corresponding system barycenter is used to retrieve the ephemeris relative to the solar system barycenter (e.g. for Mars by default the *Body ID* is internally set to 'Mars Barycenter', etc.).

ENVIRONMENT MODELS

- The *Default* ephemeris setting is *Spice* (see below) with the *de430.bsp* spice ephemeris kernel [146].
- In case *Custom* is selected, there are three types of celestial body ephemeris models available:
 - The *SPICE* toolkit (*Spice* setting) can be used as an ephemeris library. A *Kernel filename* has to be specified by the user. Optionally a different *Body ID* can be specified by the user which will be used for the specified kernel file. This is required in case the *SPICE* kernel file contains body IDs, which are different from those available in the *IAU ID* list. For more information about the *SPICE* toolkit refer to [146]. Additional information about the *SPICE* kernels can be found in the available documents of [147].
 - The *Keplerian* setting allows the user to provide the position represented by Keplerian elements at a reference epoch (see Mission Start Date in User Manual) in the J2000 frame.
 - The *Polynomial* setting causes that an analytical ephemeris model for the eight planets of the solar system plus Pluto and the Moon is used by *ASTOS*.
- The *Solar F10.7 index* setting is available only if the *IAU ID* is set to *Sun*. Here, the observed solar F10.7 index is expected as input and not the value adjusted for 1 astronomical unit. The F10.7 index is a measure of the solar radio flux per unit frequency at a wavelength of 10.7 cm near the peak of the observed solar radio emission. It is an indicator of the overall solar activity levels and correlates well with solar UV emissions. It can be *Defined* by a constant *Value* or as a *Profile* as function of the independent variables *Time* or *Julian_Date*. For a general description about Profiles see Profile Data Tables in User Manual.

For additional information about this setting check the paragraph **Celestial Body Total Solar Irradiance** in Section 2.1.2. Its connection with the solar radiation pressure model is explained in Section 2.7.

The F10.7 index is measured by the Canadian National Research Council Dominion Radio Astrophysical Observatory at Penticton, BC, Canada.

2.1.2 Background

This section provides explanations of the celestial body modelling in *ASTOS*.

Celestial Body Shape

The planet's shape is modelled as an oblate spheroid. Sections parallel to the equator are assumed to be perfect circles while meridians are ellipses with constant equatorial and polar radii R_E and R_P , respectively, see Fig. 2.2.

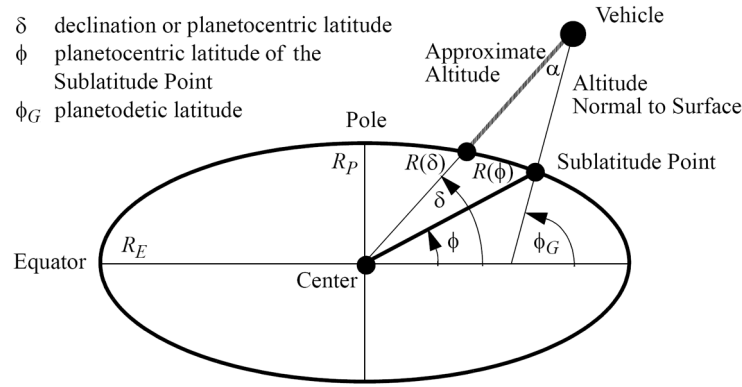


Fig. 2.2: Section along the meridian of an oblate celestial body: definitions of latitude, declination, the local radii, and altitude

The point on the celestial body's surface directly below the (vehicle) is called sublatitude point. For a spherical central body this point is the intersection of the satellite radius vector (the vector from the center of the central body to the satellite) and the central body spherical surface.

Planetodetic Latitude

The planetodetic latitude ϕ_G of a point on the elliptical meridian (e.g. the sublatitude point of a vehicle) is the angle between a) the line going through this point and being orthogonal to the elliptical meridian and b) the celestial body equator (see Fig. 2.2).

Planetocentric Latitude

The planetocentric latitude ϕ of a point on the elliptical meridian, the sublatitude point, is equal to its eccentric anomaly while the local radius of the spheroid (planetocentric radius of the sub-latitude point) is the distance to the centre given by the equation for an ellipse [14]:

$$R_{\text{Spheroid}}(\phi) = \frac{R_E(1-f)}{\sqrt{1-f(2-f)\cos^2\phi}} \quad ; \quad f = 1 - \left(\frac{R_P}{R_E}\right) \quad (2.4)$$

The parameter f is the spheroid's flattening which is related to the eccentricity e of the meridian by

$$e^2 = 2f - f^2 \quad (2.5)$$

Declination

The declination δ (also known as planetocentric latitude) is the angle between the equator of the central body and the line connecting the center of the central body and the vehicle. The declination δ is used in the dynamic equations. "Latitude" always describes the planetodetic latitude ϕ_G (also known as planetographic latitude), which is the angle between the equatorial plane and the normal to the surface of the ellipsoid.

Conversion Between the Declination and the Planetodetic Latitude

The conversion formula from declination δ to planetodetic latitude ϕ_G (see Fig. 2.2) is given by Eq. 2.6. This is an iterative process which is started by approximating ϕ_G with δ :

$$\begin{aligned}\varphi_i &= \delta \\ N &= \frac{R_E}{\sqrt{1-e^2\sin^2\varphi_i}} \\ \varphi_G &= \tan^{-1}\left[\tan\delta + \frac{Ne^2\sin\varphi_i}{R\cos\delta}\right]\end{aligned}\tag{2.6}$$

where N is the radius of curvature of the meridian, and R is the radius from the centre of the spheroid to the vehicle. The iteration process (loop over the index i) continues until $|\phi_G - \phi_i|$ is within the desired tolerance.

■ Altitude

The exact altitude h of a vehicle is defined as the shortest distance from it to the central body's surface (see Fig. 2.2). This distance can be approximated recursively, if the vehicle's position in orbit is specified by radius and declination δ and the planetocentric latitude ϕ of the corresponding sublatitude point on the central body is unknown [16]. In Fig. 2.2, the approximate altitude is marked by a thicker grey line.

In *ASTOS* the altitude normal to the central body's surface is computed according to the formula:

$$h = R \frac{\cos\delta}{\cos\varphi_G} - N\tag{2.7}$$

Celestial Body Gravity Field

The gravitational potential of an oblate spheroid can be approximated as a function of radius R and declination δ , and expanded into spherical harmonics of up to 6th order depending on the non-zero coefficients specified by the user. [31]

$$\begin{aligned}U(R, \delta) = & -\frac{\mu}{R} + \frac{\mu}{R} \cdot \frac{J_2}{2} \left(\frac{R_E}{R}\right)^2 (3\sin^2\delta - 1) \\ & + \frac{\mu}{R} \cdot \frac{J_3}{2} \left(\frac{R_E}{R}\right)^3 (5\sin^2\delta - 3)\sin\delta \\ & + \frac{\mu}{R} \cdot \frac{J_4}{8} \left(\frac{R_E}{R}\right)^4 (35\sin^4\delta - 30\sin^2\delta + 3) \\ & + \frac{\mu}{R} \cdot \frac{J_5}{8} \left(\frac{R_E}{R}\right)^5 (\sin\delta (63\sin^4\delta - 70\sin^2\delta + 15)) \\ & + \frac{\mu}{R} \cdot \frac{J_6}{16} \left(\frac{R_E}{R}\right)^6 (231\sin^6\delta - 315\sin^4\delta + 105\sin^2\delta - 5)\end{aligned}\tag{2.8}$$

where μ denotes the standard gravitational parameter of the central body, R_E the equatorial radius, and J_i are the non-dimensional zonal harmonics of the central body due to the meridian ellipticity. Tesseral harmonics are neglected, since the central body's parallels are assumed to be perfect circles. The standard gravitational parameter of the central body is the product of its mass M and the universal gravitational constant G (see Chapter 10 for the value of G implemented in the software):

$$\mu = GM\tag{2.9}$$

The local acceleration due to gravity is equal to the gradient vector of the potential function along the local axis system:

$$\hat{G} = \nabla U = \begin{bmatrix} \frac{\partial U}{\partial R} \\ \frac{1}{R \cos \delta} \left(\frac{\partial U}{\partial \lambda} \right) \\ \frac{1}{R} \left(\frac{\partial U}{\partial \delta} \right) \end{bmatrix}_L = \left(\frac{-\mu}{R^2} \right) \begin{bmatrix} G_{\text{radial}} \\ 0 \\ G_{\text{north}} \end{bmatrix}_L \quad (2.10)$$

$$\begin{aligned} G_{\text{radial}} = & 1 + \frac{3}{4} J_2 \cdot \left(\frac{R_E}{R} \right)^2 (3 \cos(2\delta) - 1) \\ & + \frac{1}{8} J_3 \cdot \left(\frac{R_E}{R} \right)^3 (20 \sin(3\delta) - 12 \sin \delta) \\ & + \frac{5}{64} J_4 \cdot \left(\frac{R_E}{R} \right)^4 (-35 \cos(4\delta) + 20 \cos(2\delta) - 9) \\ & + \frac{3}{128} J_5 \cdot \left(\frac{R_E}{R} \right)^5 (2(-63 \sin(5\delta) + 35 \sin(3\delta) - 30 \sin \delta)) \\ & + \frac{7}{512} J_6 \cdot \left(\frac{R_E}{R} \right)^6 (231 \cos(6\delta) - 126 \cos(4\delta) + 105 \cos(2\delta) - 50) \end{aligned} \quad (2.11)$$

$$\begin{aligned} G_{\text{North}} = & \frac{3}{4} J_2 \cdot \left(\frac{R_E}{R} \right)^2 2 \sin(2\delta) \\ & - \frac{1}{8} J_3 \cdot \left(\frac{R_E}{R} \right)^3 (15 \cos(3\delta) - 3 \cos \delta) \\ & - \frac{5}{64} J_4 \cdot \left(\frac{R_E}{R} \right)^4 (28 \sin(4\delta) - 8 \sin(2\delta)) \\ & + \frac{3}{128} J_5 \cdot \left(\frac{R_E}{R} \right)^5 (105 \cos(5\delta) - 35 \cos(3\delta) + 10 \cos \delta) \\ & + \frac{7}{512} J_6 \cdot \left(\frac{R_E}{R} \right)^6 (198 \sin(6\delta) - 72 \sin(4\delta) + 30 \sin(2\delta)) \end{aligned} \quad (2.12)$$

Note: The magnitude of the above vector is not strictly proportional to the inverse square of the radius which is in contrast to Newton's law of gravitation.

Celestial Body Spin State

The spin state of a celestial body defines the direction of the north pole and the location of the prime meridian following the definition of the IAU [76]. The direction of the north pole is specified by the value of its right ascension α and declination δ . The location of the prime meridian is specified by the angle measured along the celestial bodies' equator in an easterly direction starting from the ascending node of the celestial bodies' equator on the ICRF equator plane (i.e. the Earth's mean equator at epoch J2000.0). All angles are measured with respect to the inertial ICRF/J2000 coordinate system (see Section 9.3.2.1) and define the coordinate transformation from the ICRF/J2000 to the planet-centered planet-fixed (PCPF, refer to Section 9.3.2.4) coordinate system.

The Earth IAU-76/FK5 model takes into account the Earth's precession and nutation as explained in [67].

Celestial Body Total Solar Irradiance

The total solar irradiance (TSI) is a measure of the mean solar radiative power per unit area normal to the rays, incident on the Earth's upper atmosphere, adjusted for one astronomical unit from the Sun. It is a function of the solar cycle and the cross-cycle changes.

Due to the high level of dependency between solar activity indices and the solar F10.7 radio index, it is possible to derive the total solar irradiance from it. A second order polynomial regression is used to compute the total solar irradiance from the observed solar F10.7 radio index [130]:

$$TSI = 1364.4 + 0.017 \cdot F_{10.7} - 3.7E^{-5} \cdot F_{10.7}^2 \quad (2.13)$$

2.2 Third Bodies Gravity Model

Description

Gravitational forces from third bodies are relevant in case of a high-precision propagation of the vehicle trajectory.

Configuration

In the *Gravitational perturbation* panel of the *Default Environment* node in the *Modelling* tree the user can specify from a list of available celestial bodies which of them shall have a gravitational effect on the vehicle.

Background

Outside the third body sphere of influence the effect of the third body gravity field asymmetries on a trajectory are negligible. In this case, a spherical point-mass gravity model is sufficient to account for the gravity effects of the third body. In *ASTOS* the gravitational forces by third bodies are therefore modeled based on a gravitational reduced n-body problem. The acceleration due to n third bodies a_{3B} can be expressed as:

$$\begin{aligned} \vec{a}_{3B} &= \sum_{j=1}^n -\frac{\mu_j}{r_j^3} (1 - 3\cos S) \vec{u}_r \\ \cos S &= \vec{u}_j \cdot \vec{u}_r \\ \vec{u}_r &= \frac{\vec{r}}{r} \\ \vec{u}_j &= \frac{\vec{r}_j}{r_j} \end{aligned} \quad (2.14)$$

where μ_j is the gravity parameter of the j -th third body, r is the position of the spacecraft w.r.t. the central body, r_j is the position of the j -th body w.r.t. the central body, u_r is the unit vector from the central body pointing at the spacecraft, u_j is the unit vector pointing to the j -th body and S is the angle between the last two vectors.

2.3 Atmosphere Models

An atmospheric model computes density, temperature, viscosity and speed of sound given the inputs altitude, longitude, geodetic latitude and the date. The following atmospheric models are available:

- *Exponential* models provide a simple exponential atmosphere (in terms of density, pressure and speed of sound) with constant temperature, see Section 2.3.5
- *Standard* atmospheres use tabular data to specify their properties, see Section 2.3.1
- *US Standard* atmospheres are idealized, steady-state representations for the Earth's atmosphere, see Section 2.3.2
- **Fitted U.S. Standard Atmosphere** applies polynomials fitted to the data table of the US Standard 1976 model, see Section 2.3.3
- *MSIS* models describe the neutral temperature and densities in Earth's atmosphere from ground to thermospheric heights and are based on extensive data compilation from measurements of rockets, satellites and incoherent scatter radars, see Section 2.3.6
- *GRAM 99* is a NASA/MSFC Global Reference Atmospheric Model (GRAM) model for Earth with user-definable perturbations, see Section 2.3.10
- *Jacchia Bowman* represents the Jacchia-Bowman models of the Earth's atmosphere, see Section 2.3.8
- *User* provides an external interface to an atmospheric model as dynamic library, see Section 2.3.7
- *None* is the default setting in any scenario if no other model is used, see Section 2.3.4
- *EMCD* represents an advanced model of the Martian atmosphere, the Mars Climate Database, see Section 2.3.9

Common atmosphere models settings

Several atmosphere models require the following common setting:

- *Ratio of specific heats* is common for several atmosphere models and is explained in Section 2.3, subsection *Common atmosphere models settings*
- *Viscosity* can be used to define the parameters *Reference viscosity* η_B and *Reference temperature* T_S to compute the dynamic viscosity η using the law of Sutherland [77]:

$$\eta = \eta_B \frac{(T_0 + T_S)}{(T + T_S)} \left(\frac{T}{T_0} \right)^{1.5}, \quad (2.15)$$

with the current temperature T and the zero temperature $T_0=273.16$ K.

- *Density scaling factor* allows to modify the computed atmospheric density with an optimizable scaling factor. By default this is *Disabled* and the scaling factor value is set to 1.0. This scaling factor can be used in Monte Carlo simulations using the batch mode.

2.3.1 Standard Atmosphere Models

Description

This model creates an atmosphere from tabular data. Atmospheric properties have to be provided as function of the local altitude, longitude and geodetic latitude. This model can generally be used for any celestial body, i.e. it is not specially related to the Earth.

Configuration

The *Configuration Panel* of the Standard atmosphere lists the following options:

- *Ratio of specific heats* is common for several atmosphere models and is explained in Section 2.3, subsection *Common atmosphere models settings*
- *Density*: density interpolation data
- *Pressure*: pressure interpolation data
- *Speed of Sound*: speed of sound interpolation data
- *Viscosity* is common for several atmosphere models and is explained in Section 2.3, subsection *Common atmosphere models settings*
- *Mole weight* is the mole weight M_{SL} of air at sea level. Used in the computation of the temperature T . Default value is 28.97 kg/kmol.

$$T = c^2 M_{SL} / 1.4 / \mathfrak{R} \quad (2.16)$$

with the sonic velocity c and the gas constant R

- *Ozone concentration* is optional interpolation data with an additional *Seasonal scaling factor* with which the interpolated ozone concentration is multiplied
- *Density scaling factor* is common for several atmosphere models and is explained in Section 2.3, subsection *Common atmosphere models settings*

Note: For a general description about the interpolation settings see Profile Data Tables

2.3.2 U.S. Standard Atmosphere Models

Description

This package computes the U.S. Standard Atmospheres 1962 [40] and 1976 [41] that are given as a function of the local altitude. The U.S. Standard Atmosphere is an idealized, steady-state representation of the Earth's atmosphere from 5 km below sea level up to 1000 km. A built-in ozone model is implemented.

Configuration

The following model setting is available:

- *Density scaling factor* is common for several atmosphere models and is explained in Section 2.3, subsection *Common atmosphere models settings*

Background

The *US Standard* models compute the viscosity according to the following Eq. 2.17.

$$\eta = \eta_B \frac{T^{1.5}}{T + T_S} \quad (2.17)$$

Where η_B is $1.458 \cdot 10^{-6}$ and T_S is 110.4 K (Sutherland constant).

Since the documentation does not supply ozone data, the U.S. Standard Atmosphere 1962 uses the ozone model of the U.S. Standard Atmosphere 1976. The hard coded tabular data are approximated either by linear interpolation, by a natural cubic spline, or by a shape preserving Akima spline. Above an altitude of 74 km the partial density of ozone is assumed to be constant.

2.3.3 Fitted U.S. Standard Atmosphere Model

Description

This model provides properties of the Earth's atmosphere computed from smooth polynomial functions fitting U.S. Standard Atmosphere 1976 data [16]. Atmosphere data returned by this package are customized for solving numerical trajectory optimization problems. The functions are differentiable and fast to compute. Relative deviations from Standard Atmosphere 1976 data are on the order of 10-20 percent.

Configuration

The following model setting is available:

- *Density scaling factor* is common for several atmosphere models and is explained in Section 2.3, subsection *Common atmosphere models settings*

Background

The Fitted U.S. Standard Atmosphere model determines the local air density, ozone density, ambient pressure and sonic velocity at the given altitude. Between 0 and 130 km altitude data are computed using exponential functions while above 130 km data are extrapolated using an exponential function.

The density and pressure are fitted similar to that of [16] with an exponential function

$$\rho(a) = e^{\sum_{i=1}^m (c_i a^{i-1}) + c_{m+1} e^{\sum_{i=m+2}^1 c_i a^i}} \quad (2.18)$$

The sonic velocity and ozone density is fitted with a polynomial of the following type

$$c(a) = \sum_{i=1}^6 c_i a^{i-1} \quad (2.19)$$

ENVIRONMENT MODELS

See Table 2.1 for all polynomial coefficients c_i .

Table 2.1: Polynomial coefficients of the fitted atmospheric quantities

No	Density m=7	Pressure m=5	Sonic Velocity
1	0.463998	11.9228	359.478
2	- 0.0933987	- 0.187684	- 9.59161
3	- 0.0025143	0.00111257	0.451224
4	1.68219e-6	- 1.73032e-7	- 0.00798204
5	- 3.0846e-8	9.84117e-10	5.67085e-5
6	2.09694e-10	8.05029e-6	- 1.31987e-7
7	- 4.96531e-13	1.36976	
8	- 0.152699	- 0.0603357	
9	0.000736823	0.00102073	
10	- 0.0096224	- 7.70472e-6	
11	0.00025306	2.13586e-8	
12	- 3.03699e-6		
13	1.6559e-8		
14	- 3.30113e-11		

2.3.4 No Atmosphere Model

Description

This model computes an atmosphere, where density and pressure is constant zero and the sonic velocity is constant 400 m/s. If the scenario does not use any other atmosphere, an atmosphere of type *None* is internally created.

Configuration

No user input data are required.

2.3.5 Exponential Atmospheres

Description

This model computes a simple exponential atmosphere with constant temperature.

Configuration

The panel present the following entries:

- *Reference density*: reference density at sea level

ENVIRONMENT MODELS

- *Reference pressure*: reference pressure at sea level
- *Scale height*: atmospheric pressure scale height
- *Mean temperature*: constant mean temperature
- *Ratio of specific heats* is common for several atmosphere models and is explained in Section 2.3, subsection *Common atmosphere models settings*
- *Viscosity* is common for several atmosphere models and is explained in Section 2.3, subsection *Common atmosphere models settings*
- *Mole weight*: mole weight M_{SL} of air at sea level. Used in the computation of the temperature T . Default value is 28.97 kg/kmol.

$$T = c^2 M_{SL} / 1.4 / \mathcal{R} \quad (2.20)$$

with the sonic velocity c and the gas constant R

- *Density scaling factor* is common for several atmosphere models and is explained in Section 2.3, subsection *Common atmosphere models settings*

Background

$$\delta = \delta_{\text{ref}} e^{-h/H_s}, \quad (2.21)$$

$$p = p_{\text{ref}} e^{-h/H_s}, \quad (2.22)$$

$$c = \sqrt{K \mathcal{R} T / M_{SL}} \quad (2.23)$$

where h is the altitude, H_s is the constant scale height, T is the mean temperature, K is the adiabatic coefficient, $\mathcal{R}$ is the gas constant and M_{SL} the mole weight at sea level. p_{ref} and ρ_{ref} are reference values at sea level.

2.3.6 MSIS Atmospheres

Description

This group of models compute the atmospheric variables according to the MSIS models: the MSIS 86 [24], the MSISE 90 [25] and the NRLMSISE_00 [26].

Configuration

The *Configuration Panel* of the MSIS atmospheres offers the following options:

- *Ratio of specific heats* is common for several atmosphere models and is explained in Section 2.3, subsection *Common atmosphere models settings*
- *Viscosity* is common for several atmosphere models and is explained in Section 2.3, subsection *Common atmosphere models settings*
- *Solar geomagnetic activity*: adapt the atmospheric model to solar activity and fluctuations in the magnetic field:
 - *Solar flux*: $F_{10.7}$ solar flux

ENVIRONMENT MODELS

- *Solar flux 81 days average*: $F_{10.7}$ values averaged over a period of 81 days
- *geomagnetic index*: geomagnetic A_p index
- *Scaling factors*: factors for scaling computed atmospheric values:
 - *Density scaling factor*: scale atmospheric density
 - *Temperature scaling factor*: scale atmospheric temperature
- *Density scaling factor* is common for several atmosphere models and is explained in Section 2.3, subsection *Common atmosphere models settings*

Background

- **MSIS 86**: The Mass-Spectrometer-Incoherent-Scatter (MSIS) model describes the neutral temperature and densities in the upper atmosphere. MSIS-86 constitutes the upper part of the COSPAR International Reference Atmosphere (CIRA) 1986. The MSIS model is based on the extensive data compilation and analysis work of A.E.Hedin and his colleagues. Data sources include measurements from several rockets, satellites and incoherent scatter radars.

Note: The MSIS 86 model can only be used above an altitude of 85.0km

- **MSISE 90**: The MSISE 90 model describes the neutral temperature and densities in Earth's atmosphere from ground to thermospheric heights. Below 72.5 km the model is primarily based on the MAP Handbook (Labitzke et al., 1985) tabulation of zonal average temperature and pressure by Barnett and Corney, which was also used for the CIRA-86. Below 20 km these data were supplemented with averages from the National Meteorological Center (NMC). In addition, pitot tube, falling sphere, and grenade sounder rocket measurements from 1947 to 1972 were taken into consideration. Above 72.5 km MSISE-90 is essentially a revised MSIS-86 model taking into account data derived from space shuttle flights and newer incoherent scatter results. MSISE is not the model of preference for specialized tropospheric work. It is rather for studies that reach across several atmospheric boundaries.
- **NRLMSISE-00 2001**: The NRLMSISE-00 empirical atmosphere model was developed by Mike Picone, Alan Hedin, and Doug Drob based on the MSISE90 model. The main differences to MSISE90 are noted in the comments at the top of the computer code. They involve the extensive use of drag and accelerometer data on total mass density, the addition of a component to the total mass density that accounts for possibly significant contributions of ion oxygen and hot oxygen at altitudes above 500 km, and the inclusion of the SMM UV occultation data on molecular oxygen.

2.3.7 User Atmosphere

Description

ENVIRONMENT MODELS

This model links a user-defined atmosphere as described in Section 11.4. The user-defined atmosphere must provide ambient pressure, density and speed of sound.

Configuration

The *Configuration Panel* of the user atmosphere offers the following options:

- *Library file*: defines a string with the path and file name to the user library. The maximum length is 2048 characters.
- *User parameters*: defines a string with a maximum length of 1024 characters. The initialization routine of the library interface gets this string as input parameter. It can be used in any way, e.g. a filename for loading data, data values itself or a combination of both.
- *Ratio of specific heats* is common for several atmosphere models and is explained in Section 2.3, subsection *Common atmosphere models settings*
- *Viscosity* is common for several atmosphere models and is explained in Section 2.3, subsection *Common atmosphere models settings*
- *Mole weight*: mole weight M_{SL} of air at sea level. Used in the computation of the temperature T . Default value is 28.97 kg/kmol.

$$T = c^2 M_{SL} / 1.4 / \mathfrak{R} \quad (2.24)$$

with the sonic velocity c and the gas constant R .

- *Density scaling factor* is common for several atmosphere models and is explained in Section 2.3, subsection *Common atmosphere models settings*

2.3.8 Jacchia-Bowman Atmosphere Models

Description

Two versions of the Jacchia-Bowman Atmosphere Model are available in ASTOS.

- *JB2006*, see Section 2.3.8.1
- *JB2008*, see Section 2.3.8.2

Note: During simulation or optimization, an error is raised if the altitude is lower than 90 km, since both versions of the Jacchia-Bowman Atmosphere Model are not valid below that altitude.

Configuration

The *Configuration Panel* of both atmosphere models contains the following options:

- *Earth model*: select an earth to be considered from the drop-down menu. Only celestial bodies with an IAU identifier Earth are available.
- *Sun model*: select a sun to be considered from the drop-down menu. Only celestial bodies with an IAU identifier Sun are available.
- *Solar Flux*: $F_{10.7}$ index

ENVIRONMENT MODELS

- *Solar Flux 81 day average*: $F_{10.7}$ index averaged over a period of 81 days
- *EUV index*: extreme ultra-violet (euv) index scaled to $F_{10.7}$ ($= S_{10.7}$)
- *EUV 81 day average*: $S_{10.7}$ index averaged over 81 days
- *MG2 index*: Mg II index scaled to $F_{10.7}$ ($= M_{10.7}$)
- *MG2 81 day average*: $M_{10.7}$ index averaged over 81 days
- *Ratio of specific heats* is common for several atmosphere models and is explained in Section 2.3, subsection *Common atmosphere models settings*
- *Viscosity* is common for several atmosphere models and is explained in Section 2.3, subsection *Common atmosphere models settings*
- *Density scaling factor* allows to modify the computed atmospheric density with an optimizable scaling factor. By default this is *Disabled*, so internally this value is set to 1.0.

2.3.8.1 Jacchia-Bowman 2006 Atmosphere Model

Description

The Jacchia-Bowman 2006 (*JB2006*) model was developed using the CIRA72 (Jacchia 71) model as the basis for the diffusion equations. New solar indices have been used for the solar irradiances in the extreme and far ultraviolet wavelengths. New exospheric temperature and semiannual density equations were created to represent the major thermospheric density variations. Temperature correction equations were also developed for diurnal and latitudinal effects, and finally density correction factors have been included for model corrections required at high altitudes (1500- 4000 km). This model has been validated in an altitude range between 90 km and 5000 km through comparisons of accurate daily density drag data previously computed for numerous satellites. For 400 km altitude the standard deviation of 16% for the Jacchia 1971 model has been reduced to 10% in the JB2006 model during periods of low geomagnetic storm activity [72].

Configuration

In addition to the options listed in Section 2.3.8 the *Configuration Panel* of *JB2006* contains the following option:

- *Geomagnetic index* is the geomagnetic A_p index

Background

Details about this model can be found in [72].

2.3.8.2 Jacchia-Bowman 2008 Atmosphere Model

Description

ENVIRONMENT MODELS

The Jacchia-Bowman 2008 (*JB2008*) model is an improved revision to the version *JB2006*. This model takes into account the semiannual density variations as a result of solar activity and also models thermospheric heating from solar ultraviolet radiation.

Configuration

In addition to the options listed in Section 2.3.8 the *Configuration Panel* of *JB2008* contains the following option:

- *Solar x-ray index* is solar x-ray and Lya index scaled to F10 (= Y_{10})
- *Solar x-ray index 81 days average* is the Y_{10} averaged over 81 days.
- *Temperature change from DST index* is the dTc value computed from the Disturbance Storm Time (Dst) index.

Background

Documentation about this model can be found in [149].

2.3.9 EMCD Atmosphere

Description

The European Mars Climate Database [79] (EMCD) V5.2 is a statistical database describing the climate and environment of the Martian atmosphere. The interface to this atmosphere model is implemented in *ASTOS* and can be used as long as the user provides the model data to the path specified in the *EMCD data path* model setting (see below).

Configuration

The EMCD atmosphere model has the following settings:

- *Viscosity* is common for several atmosphere models and is explained in Section 2.3, subsection *Common atmosphere models settings*
- *Ratio of specific heats* is common for several atmosphere models and is explained in Section 2.3, subsection *Common atmosphere models settings*
- *Resolution*: to select between the GCM resolution or the high resolution topography.
- *Scenario*: several scenarios are available to be selected, Martian Year 24 (MY24), dust storm, warm, cold plus various value for tau; additional information can be found in [79].
- *EMCD data path* allows the insertion of a user specific path to the Mars Climate Database files. The default entry is %ASTOS%\lib\atmospheres\emcd\mcd_data where %ASTOS% is the installation path of *ASTOS*. This data is required for the model to run.
- *Density scaling factor* is common for several atmosphere models and is explained in Section 2.3, subsection *Common atmosphere models settings*

2.3.10 GRAM-99 Atmosphere

Description

The *GRAM 99* library allows the user to perform initialization, simulations and optimizations with the atmospheric model provided by NASA/MSFC [80],[81]. Moreover, the user can choose to perform the mission with a perturbed atmosphere, i.e. to perform a dispersion analysis at the boundary sigma values. The library returns the complete set of the atmospheric variables according to the user choice: the mean values in case of no-perturbation selection or the perturbed values in case of standard deviation or random analysis.

Note: This model is not part of the standard *ASTOS* installation due to license restrictions. In order to use the model, the user has to manually copy the model library to the *ASTOS* installation as described further below.

This NASA/MSFC Global Reference Atmospheric Model (GRAM) was developed in response to the need for a design reference atmosphere that provides complete global geographical variability, and complete altitude coverage (surface to orbital altitudes) as well as complete seasonal and monthly variability of the thermodynamic variables, of the wind components and selected atmospheric constituent concentrations.

GRAM is an amalgamation of three empirically based models that represent different altitude ranges. It uses either Global Upper Air Climatic Atlas (GUACA) or Global Gridded Upper Air Statistics (GGUAS) data sets, for 0-27 km altitudes. It uses a specially developed data set, based on Middle Atmosphere Program (MAP) data, for 20-120 km altitudes. Above 90 km, GRAM-99 uses the 1999 version of the Marshall Engineering Thermosphere (MET) model (also known as the Jacchia model). Fairing techniques assure smooth transition in overlap altitude ranges (20-27 km and 90-120 km). The model includes water vapor and 11 other atmospheric constituents (O₃, N₂O, CO, CH₄, CO₂, N₂, O₂, O, A, He and H). A variable scale perturbation model provides both large-scale (wave) and small-scale (stochastic) deviations from mean values for thermodynamic variables and horizontal and vertical wind components. There is an option to substitute Range Reference Atmosphere (RRA) data for conventional GRAM climatology when a trajectory passes sufficiently near any RRA site (launch stations).

A complete user's guide for GRAM-99 is available in the reference document [27]([28] for GRAM-95). The user's guide includes samples of the specific output files and plots which compare the atmospheric variables calculated by GRAM-99 with other reference values: the U.S. Standard Atmosphere and the data observed during several STS missions.

The GRAM-99 source code is written in Fortran and can be retrieved from the official GRAM-99 web site [29].

Note: The GRAM model compiled library file (named either `gram99.dll` or `libgram99.so`) shall be placed in the ".lib" folder of the *ASTOS* installation. An automatic script can be requested from service@astos.de to support users to compile the library in case they are eligible to use the source code.

ENVIRONMENT MODELS

The library receives the required input data from *ASTOS* which consists of the geographic position and the Julian date and calculates the requested output which consists of the density, temperature, pressure and wind velocity (optional). The reference data files and the solar and geomagnetic index, necessary to perform the GRAM computations, are read during the initialization process by the *ASTOS* GUI.

Configuration

The common atmosphere entries have been already presented. The following entries determine the additional data:

- *Upper air climate database*: a drop-down list defining which atmospheric model data set has to be used in the low range altitude (till 27 km) between *Average*, *GUACA* or *GGUAS*. In case of *Average* the atmospheric values are calculated from an average between the *GUACA* and the *GGUAS* values.
- *GUACA, GGUAS path*: the main directory path where the *GUACA* (or *GGUAS*) sub-directories and files are located. If this entry is not specified or leave empty, the default value is used.
- *Range reference data path*: the directory path where the Range Reference Atmosphere (RRA) files are located. The RRA tables contain the atmospheric data measured in twelve RRA sites provided (launch stations). If this entry is not specified or leave empty, the default value is used.
- *Atmosdat file path*: path name for atmospheric database (*atmosdat . txt*). If this entry is not specified or left empty, the default value is used.
- *GRAM textfile output*: the directory path where the specific GRAM-99 output files are saved. If this field is left empty or *GRAM textfile output* is *disabled*, the output file is not created. The GRAM specific output files are:
 - *output . txt*: the standard GRAM output file. For any calculated position, it contains the coordinates, the atmospheric data and wind velocity in all the configurations: mean values, standard deviation and random perturbation for the small scale and for the large scale.
 - *species . txt*: for any calculated position, it contains the concentration and the number density of the twelve considered species.
 - *special . txt*: it contains only the most important values in a "compact" version: elapsed time, coordinates, atmospheric (and wind) mean values and standard deviations. All the data relative to a single position are in the same row, it would be easy to import this file to perform further analysis.
- *Ratio of specific heats* is common for several atmosphere models and is explained in Section 2.3, subsection *Common atmosphere models settings*
- *Viscosity* is common for several atmosphere models and is explained in Section 2.3, subsection *Common atmosphere models settings*
- *Perturbation*: a drop-down list defining which *Type* of perturbation has to be computed along the entire mission:
 - *None*: only the mean values for the atmospheric variables are calculated and reported.

ENVIRONMENT MODELS

- *Sigma Density and Temperature*: the GRAM library provides the perturbed values for the atmospheric variables. The perturbed values are calculated adding to the mean values the standard deviation data (sigma) multiplied for the two float number inserted by the user, respectively for the *Sigma density* and *Sigma temperature*.

Note: A value equal to 3.0 identifies a dispersion analysis at the three sigma boundary, i.e. the Gaussian curve includes the 99.73% of the possible values.

- *Sigma Pressure and Temperature*: similar to *Sigma Density and Temperature* with pressure as independent variable instead of density.
- *Random*: The GRAM-99 library can provide the pseudo randomly generated perturbed data for the three main atmospheric variables, using the *Random scale* float number as a weight index.

Note: The gas-law equation links the three relevant atmospheric variables: temperature, pressure and density; for this reason the user can define only two sigma number. Once two variables are calculated according to the user requested sigma indices, the third variable (the pressure in case of *Sigma Density and Temperature*) has to respect the gas law and can not be modified by the user.

■ *Solar geomagnetic activity*

- *Solar flux*: the daily value of solar 10.7-cm radio noise flux in units of $10^{-22} \text{ W m}^{-2} \text{ Hz}^{-1}$ Bandwidth (the normal units for this parameter). This parameter is used only in the Jacchia model altitude range (above 90km). Historically, a value of 230 has been used for both design steady-state, as well as maximum solar activity conditions. For "quiet sun" conditions, a value of 150 may be used.
- *Solar flux 162 day average*: the ± 81 -day mean solar 10.7-cm radio flux, centred at the time for which the atmospheric values are to be computed. The considerations and the values are similar to the Solar_Flux entry.
- *Geomagnetic index*: the 3-hourly value geomagnetic index, taken 6 hours prior to the time for which the atmospheric values are to be computed. As for the solar flux, this index is used only in the Jacchia model (altitude above 90km). A value of 20.3 has been usually used for design steady-state conditions and a value of 400 for maximum conditions.

- *Density scaling factor* is common for several atmosphere models and is explained in Section 2.3, subsection *Common atmosphere models settings*

2.4 Hydrosphere Model

Description

ENVIRONMENT MODELS

The hydrosphere refers to the natural phenomenon of the presence of water in the atmosphere (e.g. rain, clouds, fog and water vapour). The effect of the hydrosphere is considered only in the link budget analysis (see Section 5.10 for more details), since the presence of water in the atmosphere obstructs the signal propagation. The hydrosphere model has no effect on the trajectory propagation of a vehicle.

The model currently implements the following sub-models:

- Rain
- Clouds and Fog
- Water Vapour

which is done according to the ITU recommendations [87]. The model configuration is given in the following sub-section and the sub-models are briefly explained in the **Background** sub-section further below.

Configuration

The hydrosphere configuration panel contains the following settings, which are explained in the **Background** subchapter below:

- *Average rainfall rate exceedance*
- *Average cloud liquid water exceedance*
- *Average sea level water vapour density exceedance*

Background

In the following a brief explanation of the hydrosphere sub-models is provided.

■ Rain

An important climatic effect on a satellite link is the rainfall. Rain results in attenuation of electromagnetic waves due to scattering and energy absorption. Rain attenuation increases with the frequency, being worse for Ku-band than for C-band (10-15 GHz). Enough extra power must be transmitted to overcome the additional attenuation induced by rain to provide adequate link availability.

The procedure to compute the rain loss is provided by [87] for frequencies up to 55 GHz.

The rain sub-model calculates the rainfall rate exceeded for a given percentage of the average year, i.e. R_p , where *Average rainfall rate exceedance* $\leq P_o(Lat, Lon)$ is the percentage value and is provided by the user and where $P_o(Lat, Lon)$ is the percentage probability of rain in an average year (see [90] for details).

The ESARAIN_PR6_v5.TXT, ESARAIN_MT_v5.TXT and ESARAIN_BETA_v5.TXT files (see [90] for details) used in the rain model contain the tabular data of the coefficients required for computation of $P_o(Lat, Lon)$. These coefficients are derived by the European Centre of Medium-range Weather Forecast from data gathered over the past 40 years.

The coefficient values at a certain position are obtained by a bi-linear interpolation along latitude and longitude.

■ Clouds and Fog

Clouds and fog as well as rain cause scattering of electromagnetic waves. For clouds or fog consisting entirely of small droplets, the Rayleigh scattering approximation (see [91] for details) is valid for frequencies below 200 GHz and it is possible to express the attenuation in terms of the total water content per unit volume.

The annual values of total columnar content of cloud liquid water, exceeded for a certain time percentage p (between 0.1% and 99%) of the average year, i.e. $L(Lon, Lat, p)$, are available in the form of digital maps (see [91] for details). The value for the percentage of the average year is specified by the user in the *Average cloud liquid water exceedance* field. The data files containing the maps are named *ESAWRED_xx.TXT*, where xx stands for the time percentage p . The total columnar content of cloud liquid water are derived by interpolating along a linear L vs. $\log(p)$ scale (where p is the time percentage).

■ Water Vapour

Losses in the electromagnetic signal transmission can also occur due to absorption by atmospheric gases such as oxygen and water vapor. This phenomenon depends on the frequency, altitude above sea level, and absolute humidity. At frequencies below 10 GHz, the effect of atmospheric absorption is negligible.

The atmospheric absorption depends on the density of the gases in the atmosphere. While the concentration of oxygen is relatively constant, the concentration of water vapour varies both geographically and in time. Therefore, a model able to return the current value of the water vapour density is required. The ITU water vapour model implemented in *ASTOS* is explained in [89]. The water vapour density $\rho_{ww}(\rho'_{ww}, h, alt, vsch)$ depends on ρ'_{ww} being the water vapour density at sea level, on h being the current altitude, on alt being the topographic altitude and on $vsch$ being the water vapour scale height.

The water vapour scale height data is extracted from the *VSCH_xx.TXT* file, the topographic altitudes from the *TOPO_1DOT5.TXT* files and the annual values of surface water vapour density from the *SURF_WVxx.TXT* files (see [89] for details), where xx stands for the time percentage of exceedance of the average year and has to be provided by the user in the *Average sea level water vapour density exceedance* field.

2.5 Wind Models

Description

ENVIRONMENT MODELS

Wind models are used to simulate the horizontal or three-dimensional wind velocity components of the atmosphere as functions of altitude, latitude and/or longitude. The following wind models are available:

- **None** is the default setting in any scenario if no wind model is explicitly specified. In this case, no wind is present in the scenario.
- *Standard* uses tabular wind velocity data, see Section 2.5.1
- *Gram99* is the wind model of the NASA/MSFC Global Reference Atmospheric Model (GRAM) for Earth, see Section 2.5.2

2.5.1 Standard Wind Model

Description

This model is used to approximate tabular wind velocity data as a function of altitude, longitude and latitude. Wind velocities (relative to the planet) vs. altitude can be provided either using Cartesian or cylindrical wind coordinates in the local planetocentric coordinate system (see Section 9.3.4.1). Wind gust and turbulence models are optionally available.

Configuration

There are two ways to define the *Wind coordinates*:

- *Cartesian*: the *Wind profile* data can be inserted as:
 - *Velocity east* is the wind velocity component to east, either specified as constant *Value* or as *Profile*. In the latter case, the *Reference altitude* is the altitude offset w.r.t. the reference ellipsoid of the central body. Such an offset could specify the altitude of a measurement station in case the profile data is given with respect to that station.
 - *Velocity north* is the wind velocity component to north, either specified as constant *Value* or as *Profile*. In the latter case, the *Reference altitude* is the altitude offset w.r.t. the reference ellipsoid of the central body. Such an offset could specify the altitude of a measurement station in case the profile data is given with respect to that station.
- *Cylindrical*: The wind profile can be inserted as:
 - *Wind direction* defines the horizontal wind coordinate system (see Fig. 2.3):
 - *Physical* is the direction in which the wind is blowing and the wind direction angle specifies the planet-relative azimuth of the wind velocity vector.
 - *Meteorological* is the direction from which the wind originates.

For example, a wind coming from the south has a physical azimuth of zero, while the meteorological azimuth angle is 180 degrees.

- *Velocity horizontal* is the horizontal magnitude of the wind velocity vector either specified as constant *Value* or as *Profile*. In the latter case, the *Reference altitude* is the altitude

ENVIRONMENT MODELS

offset w.r.t. the reference ellipsoid of the central body. Such an offset could specify the altitude of a measurement station in case the profile data is given with respect to that station.

- *Velocity direction* specifies the physical or meteorological azimuth of the wind velocity either as constant *Value* or as *Profile*. In the latter case, the *Reference altitude* is the altitude offset w.r.t. the reference ellipsoid of the central body. Such an offset could specify the altitude of a measurement station in case the profile data is given with respect to that station.

Independent of the selected wind coordinates the remaining options are:

- *Velocity vertical* is the vertical magnitude of the wind velocity vector either provided as constant *Value* or as *Profile*. In the latter case, the *Reference altitude* is the altitude offset w.r.t. the reference ellipsoid of the central body. Such an offset could specify the altitude of a measurement station in case the profile data is given with respect to that station.
- *Windshear* is optional and specifies the wind shear magnitude either as constant *Value* or as *Profile*. In the latter case, the *Reference altitude* is the altitude offset w.r.t. the reference ellipsoid of the central body. For example, it could specify the altitude of a measurement station in case the wind shear profile data is given with respect to that station.
- *Turbulences*: to specify wind gusts the following options are available:
 - *Severity level*: select either *Light*, *moderate* or *Severe*
 - *Time step*: sampling time step in seconds
 - *Noise seeds*: natural numbers for the four random number generators implemented in software. Default seeds are 23341, 23342, 23343 and 23344.
 - *Wingspan*: wingspan of the vehicle. Default value is 30.0 meters.

Each wind component may depend on up to three independent variables: altitude, longitude and latitude.

Tip: The value range for the longitude independent variable is in the range of -180° to +180°.

For a general description about Profile Data Tables see chapter Profile Data Tables in the User Manual book.

Background

The local wind velocity vector is provided as a local axis vector with Cartesian components in the upward, Eastern, and Northern direction:

$$\hat{V}_w(H) = \begin{bmatrix} V_{Up}(H) \\ V_{East}(H) \\ V_{North}(H) \end{bmatrix}_L \quad (2.25)$$

Each Cartesian component is interpolated individually as a function of altitude.

ENVIRONMENT MODELS

If the horizontal wind data are provided by cylindrical components, horizontal magnitude and direction angle (see Fig. 2.3) are converted to Easterly and Northerly wind components by the following transformation before the data are approximated:

$$\begin{bmatrix} V_{Up} \\ V_{East} \\ V_{North}_L \end{bmatrix} = \begin{bmatrix} V_{Vertical} \\ V_{Horizontal} \sin \chi_w \\ V_{Horizontal} \cos \chi_w \end{bmatrix}_L ; \quad \chi_w = \chi_m - \pi \quad (2.26)$$

The meteorological azimuth χ_m is converted into the physical azimuth χ_w if required. The vertical wind component is positive upward, regardless of the specified direction angle.

ASTOS applications require wind data to specify a planet-relative velocity according to ISO 1511.

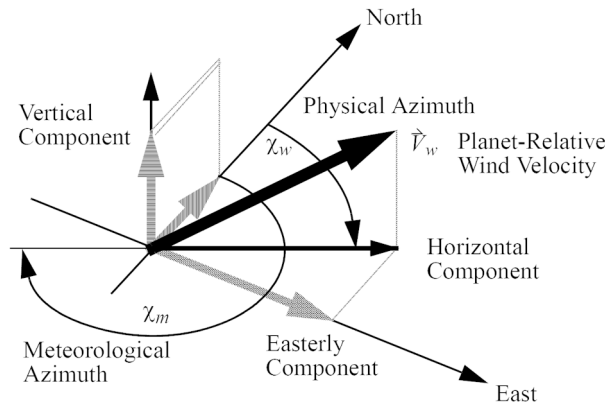


Fig. 2.3: Wind direction and Cartesian components: definition with respect to a local axis coordinate system

Turbulences

The turbulences are based on the Dryden wind turbulence model with an implementation according to [139].

2.5.2 GRAM-99 Wind

Description

The *GRAM 99* wind model is part of the Global Reference Atmospheric Model (GRAM) model for Earth by NASA/MSFC [80], [81] (see Section 2.3.10 for details on the *GRAM 99* atmosphere model). The *GRAM 99* library computes the three components of the wind velocity according to Fig. 2.3. Moreover, the user can choose to perform the mission with a perturbed wind condition, i.e. to perform a dispersion analysis at the boundary sigma values. The wind velocity components are reported according to the user choice: the mean values in case of no-perturbation selection; and the perturbed values in case of standard deviation or random analysis.

Note: This model is not part of the standard *ASTOS* installation due to license restrictions. In order to use the model, the user has to manually copy the model library to the *ASTOS* installation as described in the Background sub-section of Section 2.3.10.

Configuration

The following settings are available:

- *Perturbation*: select the kind of perturbation from a drop-down menu:
 - *None*: only the mean values for the wind velocity components are computed and reported.
 - *Sigma*: the perturbed values for the three wind velocity components are computed and reported. The perturbed values are calculated adding to each of the mean values the correspondent standard deviation data (sigma) multiplied for the respective float number inserted by the user:
 - *Sigma east*: this index identifies the requested deviation for the wind velocity in the east direction (positive from west to east). A value equal to 3.0 identifies a dispersion analysis at the three sigma boundary, i.e. the Gaussian curve includes the 99.73% of the possible values.
 - *Sigma north*: this index identifies the requested deviation for the wind velocity in the North direction (positive from south to north).
 - *Sigma radial*: this index identifies the requested deviation for the wind velocity in the Radial direction (positive from centre of the Earth to external).
 - *Random*: GRAM can provide the pseudo randomly generated perturbed data for the three wind velocity components.

2.6 Magnetic Field Models

Description

ASTOS implements an International Geomagnetic Reference Field (IGRF) model [138]. It can be used to estimate any resulting forces and torques on a vehicle containing a magnetic dipole and how these affect the vehicle attitude. The geomagnetic model can be used in case a magnetometer sensor model (refer to Section 3.4.4) or a magnetorquer actuator model (refer to Section 3.2.4.2) is present in the scenario.

The following date ranges are supported depending on the IGRF model version:

- Version 10: from beginning of the year 1900 to the end of the year 2014
- Version 11: from beginning of the year 1900 to the end of the year 2019

ENVIRONMENT MODELS

- Version 12: from beginning of the year 1900 to the end of the year 2024
- Version 13: from beginning of the year 1900 to the end of the year 2029

Note: The last 5 years of the supported date ranges are computed with reduced accuracy [138]. Outside the date ranges the models cannot be used and any ASTOS computation is terminated with an error message, which in case of the IGRF model version 10 for example reads:

```
Version 10 of the IGRF magnetic field model is intended for use up to the year 2010.
Values up to the end of the year 2014 will be computed but may be of reduced
accuracy.
***** Error in IGRF DLL *****
DATE = 2019.999 - out of range
Process terminated with status 0
```

Configuration

The model has the following setting:

- *Version* specifies the version of the IGRF model (see information in the previous section)

Background

For details on the IGRF models please refer to [138].

2.7 Solar Radiation Pressure

Solar radiation pressure is exerted by solar radiation on objects within the solar system. All spacecraft experience the effect of the pressure and this effect is modelled as a function of the solar irradiance and of the characteristics of the spacecraft: mass and projected area in the direction of the Sun, which usually are combined to a parameter similar to the ballistic coefficient of entry vehicles (i.e. mass to the projected area ratio). For higher values of this coefficient the influence of the solar radiation pressure is lower and vice versa.

Solar Radiation Force

The force acting on the spacecraft due to the solar radiation pressure is modelled to vary according to an inverse-square law with the distance from the Sun and to always acts along the line between the Sun and the spacecraft. The solar radiation acceleration is given by:

$$\vec{a}_{SPR} = \frac{K_I F_s A_C}{cm^2} \vec{u}_{Sun} \quad (2.27)$$

ENVIRONMENT MODELS

where F_s is the total solar irradiance at 1 AU, C_R is the reflectivity coefficient, c is the speed of light, m is the mass and A is the projected area of the vehicle, r is the heliocentric distance, u_{sun} is the unit vector along the Sun-spacecraft direction and K_i is the illumination coefficient. The projected area A and the reflectivity coefficient C_R can be defined in the *Solar radiation pressure properties* panel of the vehicle phase specific settings (*Dynamics Configuration*→*Vehicle & POIs Dynamics*), while the total solar irradiance is computed as explained in Section 2.1.2.

The reflectivity coefficient is a dimensionless constant in the range $0 \leq C_R \leq 2$ [129]:

- $C_R < 1$ for translucent material;
- $C_R = 1$ for perfectly absorbent material (black body);
- $C_R = 2$ for material reflecting all light directly back toward the Sun.

The illumination coefficient is a dimensionless constant in the range $0 \leq K_i \leq 1$, which depends on the illumination condition of the spacecraft:

- $K_i = 0$ in case of umbra condition (eclipse);
- $0 \leq K_i \leq 1$ in case of penumbra condition (partial or annular eclipse);
- $K_i = 1$ in case of full light condition.

In case of penumbra, an intermediate value between 0.0 and 1.0 is computed, indicating the percentage of visible Sun surface. The method to determine the correct value is illustrated in Section 5.6.2. The K_i coefficient can be computed from the eclipse function evaluated in the *Eclipse* analysis (see Section 5.6.3) as 1.0 - Eclipse.

Solar Radiation Torque

Another possible effect of the solar radiation pressure on a spacecraft is the torque. In case that the spacecraft is not modelled as a point mass and has irregular surface geometry along with heterogenous reflectance properties, some parts of the spacecraft experience a different solar radiation force than others. This may result in a torque acting on the spacecraft. Depending on the spacecraft configuration, such a torque might cause an attitude destabilisation.

By default, the solar radiation torque model is disabled within a scenario and can be activated in the *Default Environment* node of the *Modelling* tree by switching the *Solar radiation torque* setting to *Enabled*. When *Enabled*, the scaled *Torque* about the three principal axes of the spacecraft can be defined in the *Solar radiation pressure properties* panel of the vehicle phase specific settings (*Dynamics Configuration*→*Vehicle & POIs Dynamics*) as function of azimuth and elevation. These angles are the spherical representation of the Sun direction in the body fixed coordinate frame of the vehicle. The torque scaling is done using the solar irradiance, so that the provided data is independent of the solar radiation pressure. When the torque on the spacecraft is computed, the scaled torque is multiplied by the actual solar irradiance in the location of the spacecraft.

2.8 Relativistic Effects

Modelling the relativistic effects is an important aspect for long-term orbit propagation as a gravitational field produced by a massy body (e.g. planet) varies with velocity. The *Relativistic effects* model can be set up in the *Default Environment* node of the *Modelling* tree. The relativistic effect is modelled using the Parameterized Post-Newtonian formalism (PPN). This PPN formalism can also be used to simulate any other planetary system where general relativity effects are more important than in the solar system, but not systems where general relativity is dominating (i.e. where the PPN approximation does not apply anymore). According to [127], the additional acceleration due to relativistic effects for the i-th body (e.g. spacecraft), assuming that all gravitating masses are point masses, is:

$$\begin{aligned}
\vec{a}_{\text{Rel}} = & \sum_{j \neq i} \mu_j \vec{r}_{ji} \left[1 - \frac{2}{c^2} (\beta + \gamma) \sum_{k \neq i} \frac{\mu_k}{r_{ik}} \right. \\
& + \frac{1}{c^2} (1 - 2\beta + \zeta_2) \sum_{k \neq j} \frac{\mu_k}{r_{jk}} + \frac{\gamma}{c^2} \left| \frac{d\vec{r}_i}{dt} \right|^2 \frac{1}{c^2} \left(1 + \gamma + \frac{1}{2}\alpha_2 + \frac{1}{2}\alpha_3 \right) \left| \frac{d\vec{r}_j}{dt} \right|^2 \\
& - \frac{2}{c^2} \left(1 + \gamma + \frac{1}{4}\alpha_1 \right) \frac{d\vec{r}_i}{dt} \cdot \frac{d\vec{r}_j}{dt} - \frac{3}{2c^2} \left(\left(\frac{\vec{r}_{ji}}{r_{ij}} \cdot \frac{d\vec{r}_j}{dt} \right)^2 + \alpha_2 \left(\frac{\vec{r}_{ji}}{r_{ij}} \cdot \left(\frac{d\vec{r}_j}{dt} + \vec{w} \right) \right)^2 \right) \\
& + \frac{1}{2c^2} (1 + \alpha_2 - \zeta_1) \vec{r}_{ji} \frac{d^2 \vec{r}_j}{dt^2} + \frac{1}{2c^2} \vec{w} \left(\alpha_1 \left(\frac{d\vec{r}_j}{dt} - \frac{d\vec{r}_i}{dt} \right) \right. \\
& \left. \left. - (\alpha_1 - \alpha_2 - \alpha_3) \left(\frac{d\vec{r}_j}{dt} + \left(\frac{d\vec{r}_j}{dt} + \vec{w} \right) \right) \right) \right] \\
& - \frac{1}{c^2} \sum_{j \neq i} \frac{\mu_j}{r_{ij}^3} \left[\left((1 + 2\gamma) \frac{d\vec{r}_j}{dt} - 2(1 + 2\gamma) \frac{d\vec{r}_i}{dt} \right) \left(\frac{d\vec{r}_j}{dt} - \frac{d\vec{r}_i}{dt} \right) \right. \\
& \left. + \left(\frac{\alpha_1}{2} \left(\frac{d\vec{r}_j}{dt} - \frac{d\vec{r}_i}{dt} \right) - \alpha_2 \left(\frac{d\vec{r}_j}{dt} - \vec{w} \right) \right) \left(\frac{d\vec{r}_j}{dt} + \vec{w} \right) \right] \\
& \left. + \frac{3}{2c^2} \left(1 + \frac{3}{4}\gamma + \frac{1}{3}\alpha_1 - \frac{1}{3}\alpha_2 + \frac{1}{3}\zeta_1 \right) \sum_{j \neq i} \frac{\mu_j}{r_{ij}} \frac{d^2 \vec{r}_j}{dt^2} \right]
\end{aligned} \tag{2.28}$$

where the seven PPN parameters β , γ , α_1 , α_2 , α_3 , ζ_1 , ζ_2 , as well as the parameter w (speed of chosen frame relative to universe) are considered known. The internal energy density of all the celestial bodies is considered to be zero. The r_{ij} symbol stands for the position vector of the i-th celestial body w.r.t. the j-th body. The meaning of the PPN parameters is explained in the table below [127].

Table 2.2: The PPN parameters and their interpretation

Parameter	What it measures, relative to General Relativity	Value in General Relativity
β	How much space-curvature is produced by unit rest mass?	1
γ	How much nonlinearity is there in the superposition law for gravity?	1
α_1	To what extent and in what manner does the theory single out a preferred universal rest-frame?	0
α_2	To what extent and in what manner does the theory single out a preferred universal rest-frame?	0

Parameter	What it measures, relative to General Relativity	Value in General Relativity
α_3	To what extent and in what manner does the theory single out a preferred universal rest-frame?	0
ζ_1	How much and what kind of violation of conservation of total momentum does the theory predict?	0
ζ_2	How much and what kind of violation of conservation of total momentum does the theory predict?	0

The values of the PPN parameters vary from metric theory to metric theory. Typical metric theories are general relativity, scalar-tensor-theories, vector-metric theory, and stratified theories. For the case where $\beta = \gamma = 1$ and all other parameters are zero the equation reduces to the one of *General relativity*.

2.9 Gravity Gradient

Description

The gravity acceleration of an object in orbit around a celestial body depends on the radial distance from the center of gravity among other things. Since a spacecraft is not a point mass, those parts of it which are closer to the central body experience a different gravity acceleration than the spacecraft parts which are further away from it. This results in a torque acting on the spacecraft. Depending on the spacecraft configuration, such a torque can be used for attitude stabilization, e.g. to achieve a nadir pointing. If no gravity gradient torque effects are desired, they can be compensated by attitude control mechanisms.

Configuration

The gravity gradient model is disabled by default within a scenario and can be activated in the *Default Environment* node of the *Modelling* tree by setting the *Gravity gradient* setting to *enabled*.

Background

The gravity gradient torque is computed according to the following formula:

$$\vec{M} = \frac{3}{|r|^3} \vec{g} \times \left(I_{COM} \cdot \frac{\vec{g}}{|\vec{g}|} \right) \quad (2.29)$$

with $|r|$ being the radial distance from the center of gravity to the spacecraft center of mass, $\vec{g}$ being the gravity acceleration vector of the spacecraft center of mass in its body frame and I_{COM} being the inertia tensor of the spacecraft with the center of mass as a reference point.

3 Vehicle Parts and Properties Models

The vehicle parts and properties models which can be used to define a vehicle are described in this chapter. The main model categories are:

- **Vehicle Parts Common Characteristics** are settings common to all vehicle parts models, see Section 3.1
- **Actuator Models** comprise the *propulsion*, *spring-damper*, *angular momentum and torque actuator* as well as *generic actuator* models which are used to model induced forces and moments acting on the vehicle, see Section 3.2
- **Component Models** comprise the vehicle propellant mass and structure models, see Section 3.3
- **Sensor Models** provide among others a *camera*, *sun sensor* and *transmitter and receiver models* models which allow the user to perform additional analyses on the existing scenario, see Section 3.4
- **Aerodynamics and aerothermodynamics Models** allow the user to consider the interaction between the atmosphere and the vehicle on a more detailed level, see Section 3.5
- **Electrical Power Models** define how power is generated and used on board the vehicle, see Section 3.6
- **Data System Models** allow data generation and handling to be considered within a scenario, see Section 3.7

Vehicle equipment generic output

For the vehicle equipment types *Actuators*, *Components*, *Power_Components* and *Thermal_Models* in the *Simulation Data* tree a generic set of *Auxiliary functions* are computed during simulation and are then present as output in the corresponding equipment type node:

- The *Orientation* node contains the following *Auxiliary functions*:
 - *angle<number>~<part_ID>B~<vehicle_ID>* where <number> is 1, 2 or 3 denotes the Euler angle of the dynamic equipment body frame w.r.t. vehicle reference frame.
 - *angle<number>~<part_ID>J2000* where <number> is 1, 2 or 3 denotes the Euler angle of the dynamic equipment body frame w.r.t. J2000 frame.
 - *angle<number>~<part_ID>EA~<equipment_name>* where <number> is 1, 2 or 3 denotes the Euler angle of the dynamic equipment body frame w.r.t. initial equipment attitude frame. This *Auxiliary function* is available only if the *Orientation offset* setting is *Enabled* in the corresponding component in the **Vehicle Builder**.

VEHICLE PARTS AND PROPERTIES MODELS

- *quaternion_<axis>~<part_ID>B~<vehicle_ID>* where <axis> is x, y or z denotes the quaternion component describing the equipment attitude w.r.t. vehicle reference frame.
- The *Position* node accommodates the following *Auxiliary function*:
 - *<axis>_offset~<component_name>:<vehicle_name>* where <axis> is x, y or z denotes the offset of the bottom part of each component along the x-, y- and z-axis of the vehicle w.r.t. to the **Global Node**. The explanation of the latter is provided in the Configuration of assemblies and elements sub-section in the User Manual book.

3.1 Vehicle Parts Common Characteristics

This section describes the generic model properties present in some of the *ASTOS* vehicle parts models.

A list of these properties and a short explanation is given below:

- The *Common / Individual* property specifies if a certain model characteristic is common for several model instances or if it can change independently for different instances of the model, see Section 3.1.1
- The *Thermal Characteristics* property gives an explanation of the common thermal properties of vehicle part models, see Section 3.1.4
- The chapter *Electrical power properties* contains a description of the electrical power modelling principles and of all available electrical power models in *ASTOS*, see Section 3.1.3
- Section 3.1.2 *Mass Distribution and Inertia Computation* provides a description of the component and vehicle mass modelling and the algorithm to compute the vehicle's moment of inertia tensors based on the vehicle's structure as defined in the Vehicle Builder.

3.1.1 Common - Individual

Description

Common - Individual is a specific setting for certain physical properties provided by some of the vehicle part models (e.g. actuators and components). Each of these physical properties can be defined either as *Common* or as *Individual*:

- *Common* means this physical quantity is a generic property of the vehicle part model. All instances of this model share the same information/setting, which is specified only once.

VEHICLE PARTS AND PROPERTIES MODELS

- *Individual* means the physical quantity is a specific property of each instance of the vehicle part model. In this case, the instances do not share the same information/setting values and each single instance requires a separate configuration.

Configuration

The required configurations in case of *Common* and in case of *Individual* are carried out in the following panels:

- *Common* settings have to be provided in the same panel of the vehicle part model where the property is set to *Common*
- *Individual* settings have to be provided in the corresponding panels in the *Modelling* tree → *Vehicles POIs Dynamics* to which the model instances are associated

Background

Multiple instances of a vehicle part model may be present within a scenario at least in one of the following cases:

- the vehicle part model is mounted more than once in the same vehicle, e.g. an attitude thruster of a satellite (two thrusters in opposite directions)
- more than one vehicle uses the same vehicle part model, for example an engine used for satellite #1 and satellite #2 (refer to section The Vehicle Builder of the User Manual book for details)
- more than one phase exists in the scenario (refer to section Phases of the User Manual book for details)

The following scenario is an example for the last two items in the list above:

- The scenario has at least two phases and contains two satellites (#1 and #2) where satellite #1 has two engines mounted and satellite #2 has only one engine mounted
- The engine which both satellites have in common is a *Throttleable Rocket Engine* (see Section 3.2.2.2) with the property *Throttle setting* specified as *Common* or *Individual*

In this case, a *Common* setting would provide the same throttle setting (specified by the user only once in the actuator model panel) for all instances of this throttleable engine, i.e. both engines of satellite #1 as well as the engine of satellite #2 would share the same throttle setting throughout all phases. In case the property is specified as optimizable parameter or control, then only one single optimizable quantity would be present which is common for all engine instances.

On the other hand, when the throttle setting is specified as *Individual*, this property needs to be specified for each model instance separately. It means the throttle setting can be different for both engines of satellite #1 and also for both satellites. Besides the throttle setting can change from one phase to the next. In case of optimization the throttle setting of each instance can be optimized separately while all other properties of the engine like maximum thrust magnitude or specific impulse are identical. Each instance has its own optimizable parameter or control.

3.1.2 Mass Distribution and Inertia Computation

Description

The *Mass distribution* setting is available for nearly all sensor, actuator, component, power and data models. It is also available for entire vehicles in the dynamics configuration. The *Mass distribution* setting is preset to *Default* so that the center of mass and the inertia tensor are computed automatically based on simple geometric shapes. When the setting is set to *Custom*, no automatic computation is performed and the user can specify values (e.g. more accurate ones) for the center of mass and the inertia tensor to be used instead.

Configuration

The *Mass distribution* is disabled by default. If enabled the following options are displayed:

- *Common - Individual* is the choice whether the mass distribution is the same for all instances of this template, e.g. multiple vehicles use the same defined thruster or whether the mass distribution is specific for each instance. More information about this generic option is available in Section 3.1.1
- *Reference node* is the vehicle part node (e.g. Top, Bottom, Left or Right) which shall be used as a reference for the definition of the center of mass. The position of the center of mass is described as offsets from this node. This setting is available for all vehicle parts and is not available for the mass distribution setting of the complete vehicle.
- *Structure and Consumables* is a drop down menu to choose whether the center of mass and inertia tensor are given together for both liquid properties as well as structural masses (*Combined*) or as separate options, i.e. one CoM and inertia tensor for the structure and another one for liquid consumables or propellant (*Separate*). This drop-down menu is only available if the model supports both consumables and structural properties (e.g. vehicle stages). Otherwise only a *Combined* panel is visible.
- *Center of Mass* defines the CoM position with *X offset*, *Y offset* and *Z offset*. The axes are the ones of the local component coordinate system. Each offset can be *Defined* by constant *Value* or as *Profile* over time.
- *Moments of inertia* defines the moments of inertia I_{xx} (# XX), I_{yy} (# YY) and I_{zz} (# ZZ) and the products of inertia I_{xy} (# XY), I_{xz} (# XZ) and I_{yz} (# YZ) w.r.t. the center of mass. Each of them can be specified as constant *Value* or as *Profile* with the independent variable time.

Note: There exist at least two different definitions of the products of inertia I_{xy} , I_{xz} and I_{yz} . In ASTOS the definition is as follows: $I_{xy} = - \int xy \, dm$, $I_{xz} = - \int xz \, dm$ and $I_{yz} = - \int yz \, dm$. For more details see section **Dynamics of Rotation Vector** in Section 4.5.2. It is essential that the user provides the correct values for the products of inertia.

For additional information about the profile option see chapter Profile Data Tables in the User Manual.

Background

VEHICLE PARTS AND PROPERTIES MODELS

The center of mass and moments of inertia values computation for the entire vehicle can automatically be done by *ASTOS* or these values can be provided by the user. The center of mass and the moments of inertia of each vehicle part can also be computed automatically by *ASTOS* based on the assumption of simple geometric shapes, e.g. cylinders, cones, boxes and frustums, etc or can be provided by the user. Once these quantities are provided for all vehicle parts, *ASTOS* automatically computes the equivalent center of mass and inertia tensor for the vehicle. If a vehicle part, e.g. a sensor has no dimensions but a mass is specified then a point mass is assumed.

In case of a vehicle component model with a tank the moments of inertia of the propellant are added by default. In case of detailed design vehicle stages and detailed design boosters, where the type of propellant is known, it is assumed that the liquid propellant of a tank is not always rotating with the tank. In particular, when the axis of symmetry of the tank coincides with the angular velocity vector of the vehicle, the liquid propellant of this tank is not rotating with the tank structure and therefore the propellant does not add any contribution to the moments of inertia.

3.1.3 Electrical Power

Description

Every sensor, actuator, component or data component model in *ASTOS* can be assigned electrical consumption and can be added to electrical circuits. The electrical power consumption model is generic and can be applied to any of the afore mentioned components as long as the *Power system* setting is enabled in the *Active features* (see chapter Scenario General Settings in the User Manual book for details) of the scenario.

Note: Computations involving this model require the corresponding feature to be present in the *ASTOS* license.

This section gives an explanation of the electrical power consumption set up.

Configuration

This is a *Common - Individual* setting. More information about this generic option is given in Section 3.1.1. The following options are available:

- *Active* defines the electrical power (constant value) consumed by a component in active state
- *Inactive* defines the electrical power (constant value) consumed by a component in inactive state
- *Power consumption* defines the constant electrical power consumption for vehicle parts which do not support the *Active/Inactive* modes (e.g. the *Auxiliary* component, see Section 3.3.3.1 for details)

Background

VEHICLE PARTS AND PROPERTIES MODELS

Whether a vehicle part is considered active or inactive is defined in a different way depending on the type of component. A brief description of how this is done for each component type is provided below. For more detailed instructions please refer to the chapter explaining the corresponding vehicle part.

- *Actuators* need to be selected in the *Active Actuators* panel in the *Dynamics Configuration* setting of the vehicle. Otherwise they are inactive.
- *Components* with a tank are active when propellant is drawn from the tank. Otherwise they are considered inactive. If the component does not have a tank it is always active.
- *Sensors* are always active by default. They can be deactivated based on conditions like inside an eclipse, state of charge of batteries is too low or no *Link Partner* is in view.
- *Data buffers* are considered to be always active

3.1.4 Thermal Characteristics

Description

The thermal model in *ASTOS* is generic and can be applied to any component, actuator or sensor as long as the *Thermal characteristics* setting is enabled in the *Active features* (see chapter Scenario General Settings in the User Manual book for details) of the scenario.

Note: Computations involving this model require the corresponding feature to be present in the *ASTOS* license.

In case the thermal model is activated for a certain vehicle component a thermal node is assigned to the component and the node temperature is integrated over time. Sensors and actuators can also generate thermal power which is added to the assigned thermal node. The vehicle component model *Surface Element* and the *Solar Generator* consider also radiated power and irradiated power due to sunlight. All thermal nodes within one vehicle can be connected with a thermal connection in the Vehicle Builder in the *ASTOS GUI*.

Note: The thermal node model does not consider heating due to friction.

Configuration

The *Thermal characteristics* setting can be configured as follows:

- *Temperature type*, which can be an *Initial Temperature* or a *Temperature profile*. In case of *Initial Temperature* an *Initial Temperature* for the thermal node has to be specified. In case of *Temperature Profile* either a constant value or a temperature profile (for details see chapter Profile Data Tables in the User Manual) has to be provided.
- *Specific heat capacity* is the specific heat capacity assigned to the component thermal node.
- *Emissivity* is a numeric value between 0.0 (non-radiator body) and 1.0 (ideal black-body radiator). This option is only available for *Surface Element* and *Solar Generator* models.

VEHICLE PARTS AND PROPERTIES MODELS

- *Generated power* allows to specify generated thermal power for both the *Active* state and the *Inactive* state (only available for sensor and actuator models). This is a *Common - Individual* setting (see Section 3.1.1 for details).
- *Specific heat resistance* of the material. This option is only available for *Surface Element* models.

Note: If the *Thermal characteristics* setting is *Enabled*, a nonzero *Structural mass* for this component must be provided.

Background

If the *Temperature type* of a thermal node is selected as *Initial Temperature* the temperature change is computed according to the following equations:

$$\dot{Q} = \frac{\Delta T}{R_{th}} \quad (3.1)$$

$$\dot{T} = \frac{\sum_{i=1}^n \dot{Q}_i}{m \cdot c_p} \quad (3.2)$$

with

- ΔT being the temperature difference between two connected nodes
- R_{th} the absolute thermal resistance of the connection
- $Q\_dot = dQ/dt$ the heat transfer rate
- m the mass of the node
- c_p the specific heat capacity of the node
- $T\_dot = dT/dt$ the temperature derivative

Additionally irradiated and radiated thermal power are considered in *Surface Element* and *Solar Generator* models according to the following equations:

$$\dot{Q}_{rad} = A \cdot \varepsilon \cdot k \cdot T^4 \quad (3.3)$$

$$\dot{Q}_{irr} = \begin{cases} 0.0 & \text{if } \alpha \geq 90.0^\circ \\ \cos(\alpha) \cdot S \cdot A \cdot \varepsilon & \text{if } \alpha < 90.0^\circ \end{cases} \quad (3.4)$$

with

- A being the node surface
- ε emissivity of the surface
- k the Stefan-Boltzmann constant
- T the temperature of the node
- S the solar irradiance (see Section 3.1.5 for details). The *Surface Element* only considers direct sunlight.
- α the angle between surface normal and incident sunlight beam

VEHICLE PARTS AND PROPERTIES MODELS

The *Surface Element* model can consist of more than one thermal node. If more than one node is selected by the user the nodes are distributed evenly along the selected axis. An example with two nodes along the y-axis and two nodes along the z-axis is shown in Fig. 3.1.

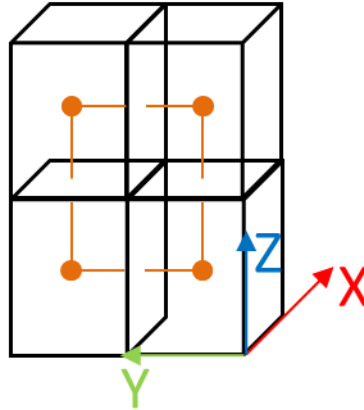


Fig. 3.1: Schematic of a surface element with four nodes and the considered cross sections for heat transfer

For each internal connection the contribution to the temperature derivative is calculated as follows:

$$\dot{Q} = \frac{A \cdot \Delta T}{d \cdot R_{\lambda}} \quad (3.5)$$

with

- A being the cross section between two nodes
- d the distance between both nodes
- R_{λ} the specific heat resistance of the surface material

3.1.5 Irradiance

Description

This chapter describes the calculation of the irradiance. Apart from direct sunlight the albedo of the central body can also be considered. The irradiance described here is used only by the *Sun Sensor*, *Surface Element* and *Solar Generator* models. The albedo model is only valid if the central body is Earth.

Background

Direct sunlight

First the Total Solar Irradiance (TSI) at Earth is calculated according to Eq. 2.13. This value is then scaled with the inverse square of the actual distance of Earth to the Sun (if the Earth is defined in the scenario, otherwise a distance of 1 AU is assumed) and with the inverse square of the distance of the vehicle to the sun (all in astronomical units). This

yields the irradiance E_e at the position of the *Sun Sensor* or *Solar Generator*. This value is then multiplied with the eclipse value (see Section 5.6.2 for a description of the eclipse model) of the central body and the incident angle of the sunlight. The limit for the incident angle is different depending on the vehicle part:

- *Solar Generator*: In case the incident angle is greater than 90 degree the irradiance due to direct sunlight is assumed to be 0.
- *Surface Element* is similar to the *Solar Generator*. The limit for the incident angle is also 90 degree.
- *Sun Sensor*: In case the incident angle is greater than the specified half-cone angle the irradiance due to direct sunlight is assumed to be 0.

Albedo

The albedo model used by *ASTOS* discretizes the Earth surface in areas of $30^\circ \times 30^\circ$ in longitudinal and latitudinal direction respectively. The reflectivity coefficients for each cell are read as monthly averaged values from files located in `./Lib/albedo` within the *ASTOS* installation folder. The source of the original daily averaged reflectivity coefficients is [144]. In order to consider also the attitude of a *Solar Generator* or *Sun Sensor* the incident angle of the line of sight from each surface cell to the observer is calculated. The limit on the incident angle is the same as stated in the direct sunlight paragraph above.

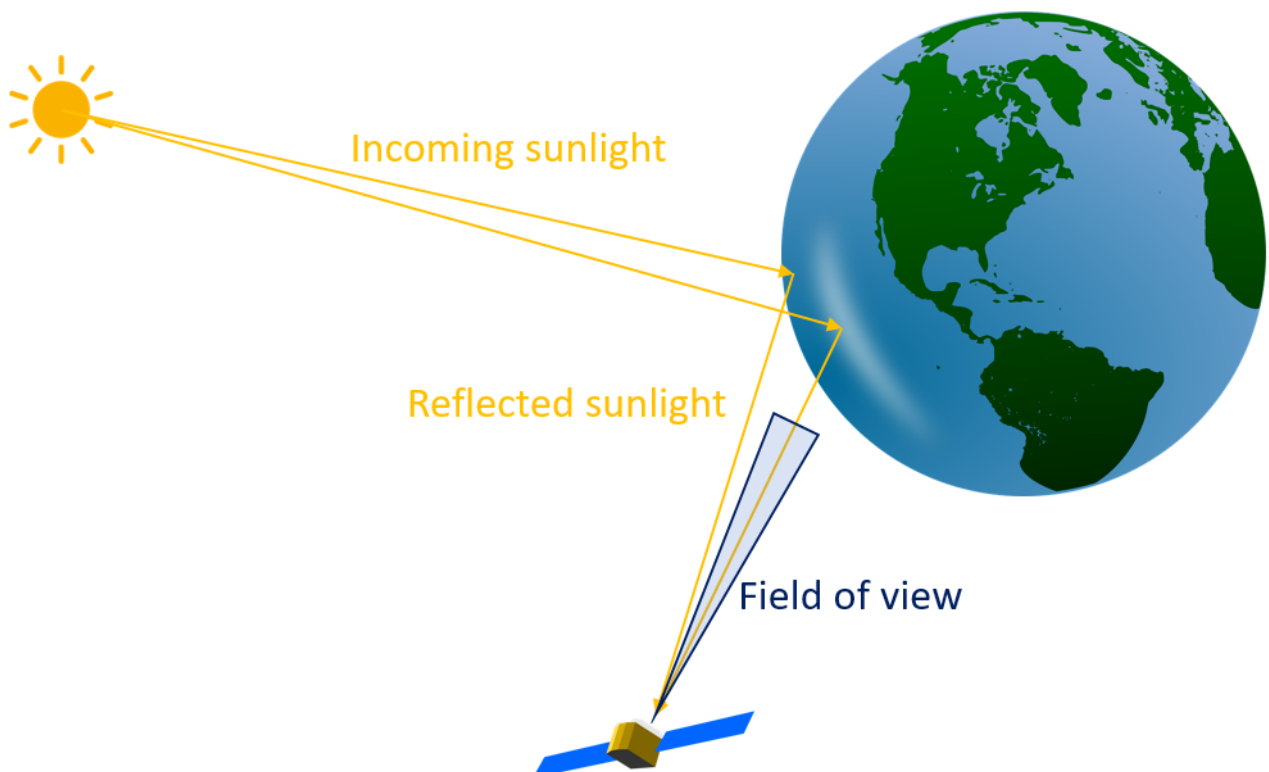


Fig. 3.2: Example of a Sun Sensor with reflected sunlight from earth's surface. Reflected rays of sunlight are not considered in the irradiance if they are outside the field of view.

The shape of the Earth is assumed to be spherical with a radius equal to the equator radius.

3.2 Actuator Models

Actuators onboard a vehicle are used to induce forces or moments, or to conserve or change the vehicle momentum and thus modify the trajectory and/or the attitude of a vehicle. The available actuator models in ASTOS are grouped as follows:

- *Clamp Band Model* as separation device between two vehicle components
- *Propulsion Models* of engines, which can be used to thrust launchers or satellites or to carry out attitude correction maneuvers
- *Spring Damper Models* of damped spring forces between two vehicle components
- *Angular Momentum and Torque Models* of attitude control or stabilization actuators
- *Generic Actuator Model* of forces and moments vectors whose components are independently modifiable

In the subsections below a detailed explanation of each actuator model is provided.

3.2.1 Clamp Band

Description

The clamp band model is an actuator working as separation device between two stages (or vehicle elements in general). It models a user-specified number of spring dampers with a compression load. All spring dampers are evenly distributed along the circumference of a circle. The clamp band has two attachment points which are defined in the vehicle builder by adjustable mounting points on the corresponding linked vehicle elements. Both attachment points define the direction of the load of the clamp band.

Note: This model does not influence the propagation in ASTOS! It is only used for multibody systems.

Configuration

The following parameters are available:

- *Diameter* defines the diameter of the clamp band device
- *Generic spring* means the clamp band has a certain number of spring dampers all with the same properties. The following settings are required:
 - *Number of springs* indicates the number of spring dampers of the clamp band
 - *Spring stiffness* is a measure of the resistance to deformation of a single spring

VEHICLE PARTS AND PROPERTIES MODELS

- *Damping constant* is the damping ratio of a single damper and describes how oscillations decay
- *Relaxed length* is the length of a single spring when the resulting force is zero
- The generic settings *Mass distribution*, *Electrical power* and *Thermal characteristics* are explained in Section 3.1.2, Section 3.1.3 and Section 3.1.4, respectively. The latter two settings are available if the corresponding features in the ASTOS software have been installed and then enabled in the scenario *Active features*.

Background

The force of each spring damper is calculated as linear function of its actual length:

$$F = -K \cdot (d - d_0) - C \cdot d' \quad (3.6)$$

where

- F is the single spring damper force (load) acting along the line defined by the mounting points
- d is the distance between the mounting points, the actual length of a spring
- d_0 is the *Relaxed length*, the length of a spring (distance between mounting points) at which the spring force is zero
- K is the *Spring stiffness* constant
- C is the *Damping constant*

When the actual length of the spring is greater than the relaxed length, the load is set to zero.

3.2.2 Propulsion Models

Description

The propulsion models described in the following sections are divided into two main categories for rocket engines and air-breathing engines. Specific engine models are available within each of these categories. *Rocket engines* comprise the following models:

- *Profile Rocket* models a rocket engine where the nozzle area and two of the three quantities mass flow, vacuum thrust and specific impulse can be a given function of up to 7 independent variables. This engine does not have a control input and can be restarted.
- *Throttleable Rocket* models a *Profile Rocket* engine whose mass flow is throttled
- *Linear Design Rocket* models a throttleable rocket engine whose mass flow is scaled by an optimizable design parameter and its engine mass is a linear function of the maximum vacuum thrust
- *Basic Design Rocket* models a throttleable rocket engine whose mass flow is scaled by an optimizable design parameter and its engine mass is a function of correlation coefficients and the maximum vacuum thrust.

VEHICLE PARTS AND PROPERTIES MODELS

- *Throat Design Rocket* models a throttleable rocket engine using exhaust velocity and characteristic velocity at equilibrium or data provided by RPA [110] which is used by ASTOS to compute thrust and mass flow of fuel and oxidizer. It is a design rocket model providing also engine mass estimations and I_{sp} efficiency estimations (depending both on the selection of Stage, Engine Cycle, Oxidizer and Fuel).
- *Typical Rocket with Variable Mixture Ratio (VMR)* models a throttleable rocket engine whose fuel-to-oxidizer ratio is controlled.
- *Solid Design Rocket* models a solid propulsion engine whose mass flow rate and therefore the thrust can be selected from predefined progression models. Depending what shape is used several parameters of the mass flow rate can be optimized.
- The *Airbreathing Engine* model features interpolated thrust and mass flows with up to 8 independent variables.
- The *User Propulsion* offers the user the possibility to program and integrate in ASTOS his own engine model where he can code all the functions of an interface. It supports optimizable controls and design parameters and has an engine mass model.

Note: This engine model cannot be inserted several times in the vehicle structure. In this case, the user has to create multiple engine models and include each of them in the vehicle structure.

None of the propulsion models has a mass model for the propellant mass and must be associated with a vehicle stage.

Most propulsion models provide auxiliary output functions of the vacuum thrust, mass flow, specific impulse and the nozzle exit area. Used independent variables and other characteristics are printed in the propulsion properties of the *Result Summary*.

Background

The compatibility criteria whether an actuator model can be connected to a vehicle stage is the number of mass flows which the actuator model can handle. Table 3.1 lists available actuator models and the type of stage that can be connected to them.

Table 3.1: Available propulsion system models

Engine Model	Associated Stage Model
Profile Rocket	Basic Vehicle Stage Basic Design Stage Detailed Design Stage Mono Propellant
Throttleable Rocket	Basic Vehicle Stage Basic Design Stage Detailed Design Stage Mono Propellant
Linear/Basic Design Rocket	Basic Vehicle Stage Basic Design Stage

VEHICLE PARTS AND PROPERTIES MODELS

Engine Model	Associated Stage Model
	Detailed Design Stage Mono Propellant
Throat Design Rocket	all vehicle stages since the <i>Propellant Type</i> option includes mono and dual propellant
Variable Mixture Ratio Rocket	Dual Tank Stage
Solid Design Rocket	Basic Vehicle Stage Basic Design Stage Detailed Design Stage Mono Propellant
Airbreathing Engine	Basic Vehicle Stage Basic Design Stage Detailed Design Stage Mono Propellant

Visualization

The visualization of the vehicle requires the knowledge of the vehicle engines' length among others. The latter can be computed using an expression containing two of the following three quantities: throat area, exit area and expansion ratio.

In case of a *Throat Design Engine* the engine throat area and the nozzle exit area are computed by the model. For other engine models the user specified nozzle exit area is used to compute the expansion ratio. An expansion ratio of 1.0 is assumed if the nozzle exit area is not specified.

The throat area A_t is given by

$$A_t = \frac{C^*}{p_c} \dot{m} \quad (3.7)$$

with the chamber pressure p_c , an average of the characteristic velocity C^* for hydrazine (1,333 m/s) and cryogenic oxygen/hydrogen (2,360 m/s) as well as the user specified mass flow rate $\dot{m}$. In Fig. 3.3 the relationship between the chamber pressure of a rocket engine and its thrust is shown. Rocket engines' data of typical launchers has been used for the figure. For a user-specified thrust value the ASTOS software computes a corresponding chamber pressure according to the formula shown next to the fitting line in Fig. 3.3.

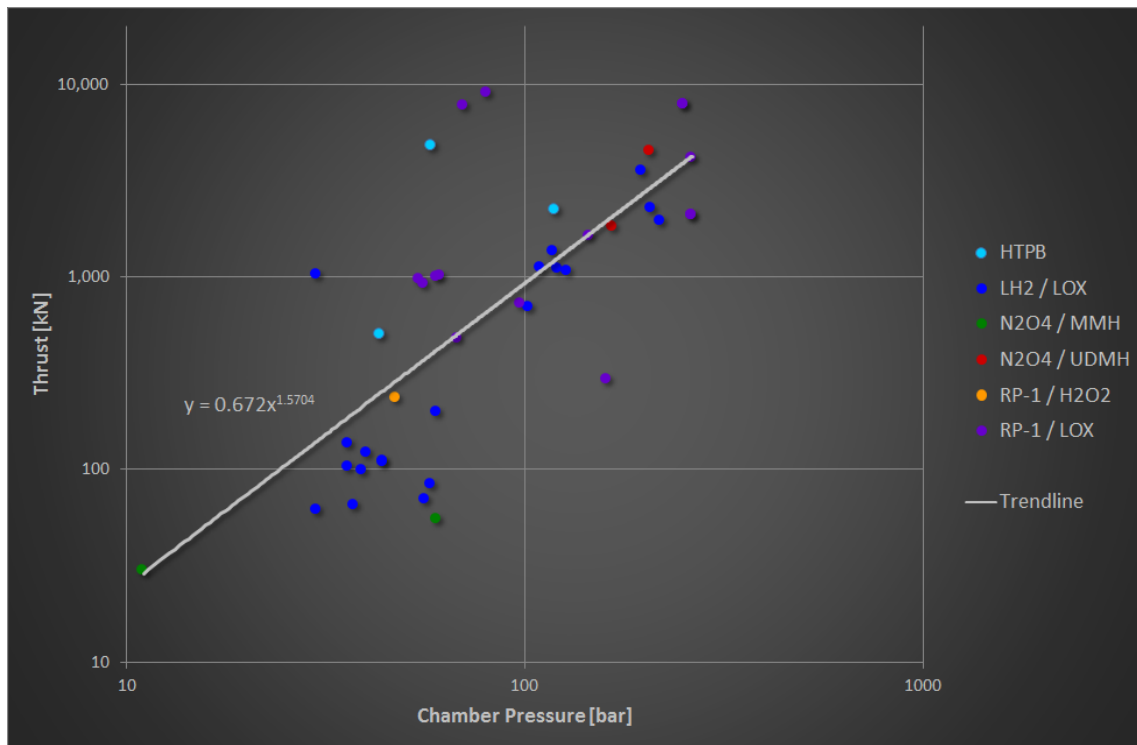


Fig. 3.3: Correlation of the engine thrust and the engine chamber pressure

In case that the engine throat area is larger than the user-specified nozzle exit area, the expansion ratio is assumed to be 1.0 and the throat area value is set to the nozzle exit area value.

3.2.2.1 Profile Rocket Engine

Description

The profile rocket engine is the basic model for any rocket engine. All the models presented in the next paragraphs expand the functionalities provided by this model.

Two of the following input parameters have to be defined: vacuum thrust, mass flow and vacuum Isp or exhaust velocity. They can be provided as constant values or as functions of a set of independent variables. This model can be used to reproduce almost any uncontrolled rocket systems, especially typical solid boosters and liquid rocket motors with a fixed mass flow profile and a fixed burn duration. Moreover, it allows to consider that a fraction of the propellant mass has to be ejected without being burnt. No mass model is associated with this engine. The structural and propellant mass budget of the associated vehicle stage can be modelled by one of the *Components* described in Section 3.3.2.

A *Profile Rocket* engine is restartable, i.e. it can be switched off during an intermediate phase, e.g. a coast arc. Therefore the virtual ignition time parameter is defined. The burn time is always the difference of flight time and virtual ignition time.

VEHICLE PARTS AND PROPERTIES MODELS

The user can decide to set the mass flow-time integral equal to a certain total propellant mass, regardless of the chosen interpolation method. This is only possible, if the mass flow is function only of the burned time.

Configuration

The engine configuration requires the following settings: The performance of the engine is defined by setting the following input:

- *Thrust scaling factor*. The user can provide a positive values used to multiply the current thrust magnitude. The scaling factor may be useful to consider cases when the propulsion force vector does not lay exactly in the axial axis, e.g. $c = \cos \varepsilon$, where ε is the angle of the thrust vector.
- *Nozzle Ae* is the total nozzle exit area. If zero, the back pressure term of the thrust force is neglected.
- *Performance defined by* allows the engine performance parameters to be provided by choosing two of the following four engine parameters *Vacuum thrust*, *Vacuum Isp* or *Exhaust velocity* and *Mass flow*. Four possible combinations are available:
 - vacuum thrust T_{vac} and total mass flow $\dot{m}$ (see explanations below): *Vacuum thrust and Mass flow*
 - vacuum thrust T_{vac} and effective exhaust velocity c_e (see explanations below): *Vacuum thrust and Exhaust velocity*
 - vacuum thrust T_{vac} and specific impulse I_{SP} (see explanations below): *Vacuum thrust and Vacuum Isp*
 - total mass flow $\dot{m}$ and specific impulse I_{SP} (see explanations below): *Mass flow and Vacuum Isp*

Depending on the selected combination, the following values need to be specified:

- *Vacuum thrust* is the engine thrust that is generated in vacuum.
- *Vacuum Isp* is the normalized specific vacuum impulse. This quantity is normalized by the standard gravitational acceleration value of $g_0 = 9.80665 \text{ m/s}^2$.
- *Exhaust velocity* is the effective exhaust velocity (sometimes also called specific impulse). This quantity corresponds to the denormalized specific vacuum impulse and can only be used only as alternative to the vacuum Isp and in combination with vacuum thrust, but not with the mass flow.
- *Mass flow* defines the propellant mass flow. One additional setting may be provided:
 - *Scale to fit* allows to set the time integral of the mass flow exactly to the specified total propellant mass. If this setting is not activated, the mass flow follows the provided *Profile* or *Value*. *Scale to fit* is only possible if the mass flow depends only on burn time, see Eq. 3.12.

VEHICLE PARTS AND PROPERTIES MODELS

- The *Unburned propellant* panel contains information on the propellant mass flow, that does not contribute to the engine thrust. Leaving this option *disabled* sets the amount of unburned propellant to zero. The unburned propellant can be specified as:
 - *Fraction of Total Propellant*, which defines a fraction of the total propellant mass that is not converted into thrust and can be provided as *Mean* (Eq. 3.14) or *Current* value (Eq. 3.15).
 - *Given Amount*, which defines a fixed (*Value*) or variable (*Profile*) amount of propellant mass that is not converted into thrust (Eq. 3.16).
- The *Consider jet damping effect* is caused by the flame of the engine, which creates a small resistance to any change of attitude. This effect can become quite important in case of 6-DOF scenario of small launchers (e.g. sounding rockets). It is function of the distance between the center of gravity and the nozzle plane. The computation does not require any additional user input and follows the mathematical model presented in [68]. This effect is quite negligible if applied to big launchers.
- The *Actuation damping* model allows the user to consider damping when simulating thrust vector control (TVC). When controls are applied to the thrust vector, a second order transfer function can be defined in order to model the reaction of the dynamic system to the provided controls. The commanded nozzle deflection angle $\Delta\delta_C$ differs from the realised one $\Delta\delta_R$ depending on the *Damping* ζ , *Natural frequency* ω and *Gain* K selected. The relationship is defined as follows:

$$\Delta\delta_R(t) = \frac{K\omega_a^2}{s^2 + 2\zeta\omega_a s + \omega_a^2} \Delta\delta_C \quad (3.8)$$

- The *Structural mass* is a constant mass of the engine structure and is considered in the total mass of the engine.
- The generic settings *Mass distribution*, *Electrical power* and *Thermal characteristics* are explained in Section 3.1.2, Section 3.1.3 and Section 3.1.4 respectively. The latter two settings are available if the corresponding features in the ASTOS software have been installed and then enabled in the scenario *Active features*.

Model background

The given tabular data are either processed by *Linear* interpolation or approximated by a natural cubic *Spline* or shape-preserving *Akima* spline or a *General B-Spline*. Each data set may depend on any of the independent variables:

- Burn time t
- Altitude h
- Pressure p
- Density ρ
- Mach number M
- Angle of attack α
- Dynamic pressure q
- State of charge SoC

VEHICLE PARTS AND PROPERTIES MODELS

- Propellant mass m_p
- Longitude λ
- Latitude Φ

The vector of independent variables is $x = [t, h, p, \rho, M, \alpha, q, SoC, m_p, \lambda, \Phi]^T$.

Note: The possible interpolation methods depend on the number of specified dimensions (see Profile Data Tables in User Manual). Instead of a data table also a constant value can be specified.

The thrust force vector is computed in the body-fixed reference frame as

$$\tau = C \begin{bmatrix} T_{\text{axial}}(x) \\ 0 \\ 0 \end{bmatrix}_B \quad (3.9)$$

$$T_{\text{axial}}(x) = g_0 I_{\text{sp}}(x) \dot{m}_b(x) - A_e(\hat{x}) p = T_{\text{vac}}(x) - A_e(x) p \quad (3.10)$$

where $I_{\text{sp}}(x)$ is the specific **vacuum** impulse, $\dot{m}_b(x)$ is the mass flow of burnt propellant, $T_{\text{vac}}(x)$ is the vacuum thrust, $c_e(x)$ is the effective exhaust velocity, $A_e(x)$ the engines nozzle exit area and p the ambient atmospheric pressure. Moreover, $T_{\text{vac}}(x) = \dot{m}_b(x) \cdot c_e + p_e A_e(x)$, where p_e is the nozzle exit area pressure.

Note: If the nozzle exit area value $A_e(x)$ showed in the *ASTOS GUI* is different from the value of the nozzle exit area used for the calculation of $T_{\text{vac}}(x)$ the resulting engine thrust may become negative. In this case, *ASTOS* sets the value to zero.

The total mass flow ejected by the engine is computed from

$$\dot{m}(x) = \dot{m}_b(x) + \dot{m}_M(x) \quad (3.11)$$

In most cases a profile rocket is defined by one of the data sets above and given as a function of the burn time interval between zero and burn-out $0 \leq t_k \leq \Delta t$.

If the interpolation method is not linear, the total ejected mass may differ from the propellant mass in the tank and the burn duration may differ from the nominal burn duration. This can be bypassed using a scale to fit option. It scales the mass flow, so that the burn duration and propellant mass are equal to the nominal values. The scaled mass flow can be computed by

$$\dot{\bar{m}}(t) = \frac{\dot{m}(t)}{c} \quad \text{with: } c = \int_{t_f}^{t_i} \frac{\dot{m}(t)}{m_p} dt \quad (3.12)$$

where m_p is the propellant mass and c the scaling factor. The scaling factor is set to unity, if the *Scale to fit* flag is *disabled*. Using the scale-to-fit option is recommended, since the integral of the mass flow function with respect to time varies with the chosen interpolation method. These variations may accumulate to several tons of propellant for typical rocket booster data. The value of the scaling factor depends highly on the usability of the mass flow data for spline interpolation. So it is also a measurement for the accuracy of the spline approximation. The value is printed in the propulsion properties of the *Result Summary*.

The mass flow of the burned propellant is given by

$$\dot{m}_b(x) = \dot{m}(x) - \dot{m}_u(x) \quad (3.13)$$

where $\dot{m}_u(x)$ is the mass flow of unburned propellant, which depending on the settings specified in the *Unburned propellant* panel can be defined in one of the following ways (see Fig. 3.4):

- If *Mean* fraction is selected, the unburned propellant mass flow is a constant fraction of the average mass flow Φ :

$$\dot{m}_u = u\Phi \quad \text{with: } \Phi = \int_{t_f}^{t_i} \frac{\dot{m}(t)}{t_f - t_i} dt \quad (3.14)$$

If the mass flow depends not only on the burn time, the mean mass flow cannot be computed and the *Current* fraction approach is used instead.

- If *Current* fraction is selected, the unburned propellant mass flow is a fraction of the current total mass flow and varies as a function of burn time:

$$\dot{m}_u(x) = \dot{m}(x) u \quad (3.15)$$

- If *Given amount* is selected, the unburned propellant mass flow is given as tabular data (*Profile*) or as a constant value (*Value*):

$$\dot{m}_u(x) = \dot{m}_u(x) \quad (3.16)$$

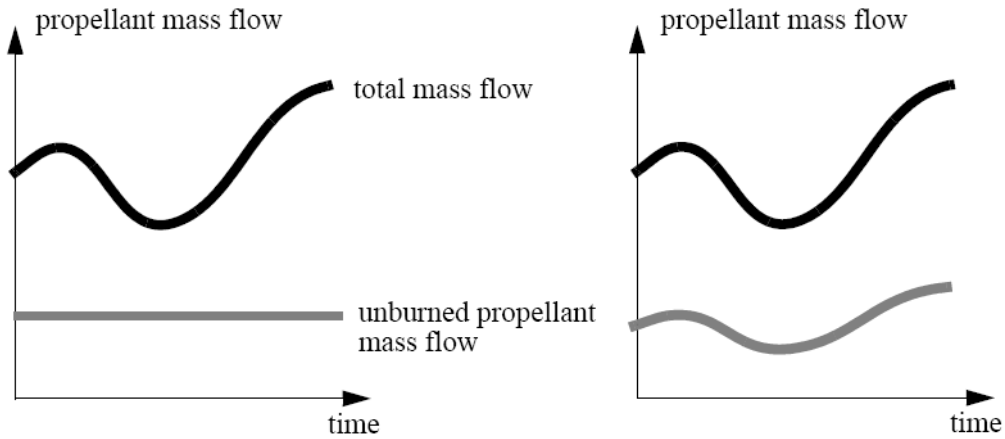


Fig. 3.4: Unburned propellant as Fraction of total propellant: if *Mean* fraction is selected, the mass flow of unburned propellant is constant (left); if *Current* fraction is specified, the mass flow of unburned propellant is a fraction of the current total mass flow (right).

3.2.2.2 Throttleable Rocket

Description

This model expands the functionalities of a profile rocket engine by allowing the mass flow to be function of the controllable throttle setting. It applies mostly to restartable and throttleable

VEHICLE PARTS AND PROPERTIES MODELS

liquid rocket motors. The *Throttleable Rocket* engine is equal to the *Profile Rocket* engine in every aspect, except that it is throttleable.

This paragraph explains only the differences to the *Profile Rocket* propulsion which is described in detail in Section 3.2.2.1.

The axial thrust of Eq. 3.10 depends now on the optimizable throttle setting σ as follows:

$$T_{\text{axial}} = g_0 I_{\text{sp}}(x) \dot{m}_b(x) \sigma - A_e(x) p = T_{\text{vac}}(x) \sigma - A_e(x) p, \quad (3.17)$$

$$\text{where } 0 < \sigma \leq 1 \text{ and} \quad (3.18)$$

$$x = [t, h, p, \rho, q, M, \alpha, \sigma]^T \quad (3.19)$$

The vector of independent variables x includes the throttle setting σ . If the tabular data of the mass flow is given as a function of the throttle setting σ , then the mass flow is not multiplied by σ as shown in Eq. 3.17, since the dependency from the throttle is already part of the mass flow itself. This feature allows to define an upper and lower profile for the mass flow, while by using the throttle control the optimizer can directly modify the mass flow within the user-defined bounds.

Configuration

All settings common to *Profile Rocket* engines are showed in detail in Section 3.2.2.1. The specific model input is described here:

- *Throttle setting* is optional and can be defined as *Common* or *Individual* (see Section 3.1.1). It specifies the initial value and the bounds of the throttle setting. A *Constant Law* or a *Profile* can be specified. In this second case the following independent variables can be selected: burn time, altitude, air path velocity, Mach and dynamic pressure. This setting may be overwritten by settings in the *Phases* configuration (see *Phases* and *Common Settings* in the User Manual book).

Note: If *Throttle setting* is disabled, a default value of 1.0 is used with no bounds (from 1.0 to 1.0): the optimizer cannot change the throttle.

- The generic settings *Mass distribution*, *Electrical power* and *Thermal characteristics* are explained in Section 3.1.2, Section 3.1.3 and Section 3.1.4 respectively. The latter two settings are available if the corresponding features in the *ASTOS* software have been installed and then enabled in the scenario *Active features*.

The throttle setting of a controllable engine can be modelled as a continuous control. To edit it, the *Grids* panel in the *Optimization* tree should be used. In this way the throttle setting can also be fixed to a constant value. The name of the throttle setting control contains the ID string of the corresponding engine.

3.2.2.3 Linear Design Rocket

Description

Linear design rocket engines can be used to model the mass flow as a linear function of the controllable throttle setting. The mass flow may be scaled by an optimizable engine sizing factor. Additionally, the model provides the "dry" mass of the engine which is also a linear function of the engine size. This model is especially useful for early design studies in order to optimize the available number of engines.

This model should be used for simultaneous trajectory and engine size design in the neighbourhood of nominal engine performance data. The engine size is modified by scaling the maximum mass flow and nozzle exit area; a higher mass flow is penalized by increasing the engine dry mass and by decreasing the pressure thrust due to the increased nozzle exit area. A relationship between the engine's dry mass and the maximum vacuum thrust must be supplied when using this model.

The *Linear Design Rocket* engine is equal to the *Throttleable Rocket* engine in every aspect, except that it provides an optimizable engine sizing factor. This paragraph explains only the differences with the *Throttleable Rocket* and thus also to the *Profile Rocket* propulsion, which are described in detail in Section 3.2.2.2 and Section 3.2.2.1.

The computation of the axial thrust in general is given in (Eq. 3.17). With this engine, the mass flow is computed from the nominal mass flow $\dot{m}_0(x)$ and the engine sizing factor z :

$$\dot{m}(x) = \dot{m}_0(x)z \quad (3.20)$$

$$\text{where } x = [t, h, p, \rho, q, M, \alpha, \sigma]^T \quad (3.21)$$

Since the engine nozzle expansion ratio is assumed to be fixed, the nozzle exit area is also a function of the engine sizing factor

$$A_e(x) = A_0(x)z \quad (3.22)$$

where A_0 is the nominal exit area.

In this model, a linear relationship between the engine dry mass m_e and the maximum vacuum thrust is assumed. The user is requested to define the ratio r between the engine dry mass and the maximum vacuum thrust:

$$m_e = rT_{\max} \quad r = \frac{m_{e,0}}{T_{\max,0}} \quad (3.23)$$

with the maximum vacuum thrust $T_{\max}$ given by

$$T_{\max} = g_0 I_{sp} \dot{m}_{\max} z \quad (3.24)$$

Configuration

The *Linear Design Rocket* requires the same input data as the *Throttleable* and *Profile Rocket* engines, which are described in Section 3.2.2.2 and Section 3.2.2.1, except for:

- *Engine mass to thrust ratio*, which specifies the constant engine mass to thrust ratio.
- The *Engine sizing factor* is an optional input which provides the initial value and (bounds) of the engine sizing factor.
- The generic settings *Mass distribution*, *Electrical power* and *Thermal characteristics* are explained in Section 3.1.2, Section 3.1.3 and Section 3.1.4 respectively. The latter two settings are available if the corresponding features in the ASTOS software have been installed and then enabled in the scenario *Active features*.

3.2.2.4 Basic Design Rocket

Description

This engine model can be used to model a liquid rocket engine whose mass flow is a linear function of the controllable throttle setting. The mass flow may be scaled by an optimizable engine sizing factor. Additionally, the model provides the "dry" mass of the engine which is also a function of the engine size [17]. This model applies to a family of rocket motors for a given fuel mixture ratio, constant chamber pressure, and nozzle expansion ratio.

As for linear design rocket engines Section 3.2.2.3, the engine size is changed by scaling the maximum mass flow and nozzle exit area. A relationship between the engine dry mass and the maximum vacuum thrust must be provided; two different parametrized correlation functions are available.

This model should be used for simultaneous trajectory and engine size design in the neighborhood of nominal engine performance data. In essence, the engine size is changed by scaling the maximum mass flow and nozzle exit area; a higher mass flow is penalized by an increase in engine dry mass and a decrease in pressure thrust due to the growing nozzle exit area. A relationship between the engine dry mass and the maximum vacuum thrust must be supplied when using this model. Two different parameterized correlation functions are available. The fitting coefficients are to be specified.

Configuration

The *Basic Design Rocket* engine is equal to the *Linear Design Rocket* engine (Section 3.2.2.3) in every aspect, except that it provides advanced models to estimate the engine mass.

In the *Engine mass correlation* panel, the user can specify the relationship between engine performance and required mass to be used in the engine sizing design. It requires the following data:

- The *Formulation* selector allows to choose different types of analytical relationships to compute the corresponding mass vs. performance correlation formula.

VEHICLE PARTS AND PROPERTIES MODELS

- A coefficient, *B coefficient* and *C coefficient* allow to fully define the engine mass correlation relationship, as explained below.
- The generic settings *Mass distribution*, *Electrical power* and *Thermal characteristics* are explained in Section 3.1.2, Section 3.1.3 and Section 3.1.4 respectively. The latter two settings are available if the corresponding features in the ASTOS software have been installed and then enabled in the scenario *Active features*.

Model background

In this model, an analytical relationship between the engine dry mass m_e and the maximum vacuum thrust is postulated. The user is requested to choose one of the following analytical correlation formulae and supply the corresponding fitting coefficients:

- For an *Exponential* relationship of the form

$$m_e = aT_{\max}^b \quad (3.25)$$

- For a *Logarithmic* relationship of the form

$$m_e = a + b \ln T_{\max} \quad (3.26)$$

- For a *Reciprocal Logarithmic* relationship of the form

$$m_e = \frac{T_{\max}}{a + b \ln T_{\max}} \quad (3.27)$$

- For an *Quadratic* relationship of the form

$$m_e = aT_{\max}^2 + bT_{\max} + c \quad (3.28)$$

where $T_{\max}$ is the maximum vacuum thrust given by

$$T_{\max} = g_0 I_{sp} \dot{m}_{\max} \quad (3.29)$$

The constant correlation coefficients a , b , c are to be specified for $T_{\max}$ in units of Kilo-Newton, and m_e in units of Kilogram. Typical numeric values of the correlation coefficients are listed in Table 3.2 .

Table 3.2: Typical correlation coefficients for the engine model

Engine Type	Correlation Coefficients (Exponential Formulation)	
	a	b
LOX / LH ₂ rocket	3.6234	0.8731
Storable propellant rocket	0.8889	1.0254

3.2.2.5 Throat Design Rocket

Description

VEHICLE PARTS AND PROPERTIES MODELS

This model is suitable for simultaneous trajectory and engine size design. It is used to setup either a dual propellant or a mono propellant rocket engine defined by optimizable parameters and a throttle control. The four basic parameters to characterize the engine are *Chamber pressure*, *Expansion ratio*, *Mixture ratio* and an additional **sizing parameter** to define the (chamber) engine size.

Furthermore, the model provides geometry settings in order to put requirements on the nozzle shape and mass correlation settings which defines how the structural mass is computed.

Tip: This model is suitable for simultaneous trajectory and engine size design.

Note: A mono propellant rocket engine can only be linked to a mono propellant stage, while a dual propellant rocket can only be linked to a dual propellant stage. The *Propellant type* selector (see below) allows to use this engine as mono or dual propellant rocket engine.

Model configuration

Design parameters define performance and dimension of the engine as already mentioned above, i.e. *Chamber pressure*, *Mixture ratio*, and *Expansion ratio*. *Size defined by* determines which sizing parameter is used amongst:

- *Throat Area*
- *Mass Flow Rate*
- *Vacuum Thrust*

Depending on the selection, an input field for the respective value is available.

The *Equilibrium calculation* panel defines the technique used to compute the engine performance (*Characteristic velocity* and *Exhaust velocity*):

- *Table*: a single *Value* or tabular set of values (*Profile*) is provided. In the latter case, the current value is computed by means of interpolation with chamber pressure, expansion ratio and mixture ratio as independent variables.
- *RPA Configuration File*: a configuration file created with *RPA* [110] is used for the computations performed by *RPA.dll*.
- *RPA*: a valid *RPA* configuration file is created by *ASTOS* using the additional input as described below. The computation is performed using the *RPA.dll*.

Note: The usage of the *RPA.dll* requires *Microsoft Visual C++ Redistributable 2010* or later to be installed on the system.

The *RPA* option first requires to specify an equilibrium *Condition*. If *Shifting* is selected, then the shifting chemical equilibrium is assumed in the whole nozzle. If *Frozen* is selected, the nozzle section (via *Area Ratio* or *Pressure Ratio*) needs to be defined where the model should be applied.

VEHICLE PARTS AND PROPERTIES MODELS

In both cases the *Nozzle inlet condition* has to be provided via *Contraction Area Ratio* (chamber to throat), *Mass Flux* or *Infinity Area* (which assumes an infinite cross-section area).

Note: *RPA configuration file* or *RPA Equilibrium calculation* must be selected in order to use *RPA* also for further computations (*Engine mass estimation*, *Isp efficiency* or *Geometry definition*).

Isp efficiency definition settings determine how the specific impulse efficiency of the engine is computed:

- *RPA*: *RPA* is used to compute the efficiency.
- *User*: This option allows to explicitly define the engine efficiency (between 0.0 and 1.0).
- *Automatic*: The efficiency is computed depending on the given *Engine characteristics* (see below).

The *Engine mass correlation* settings specify the relationship between performance and mass of the engine. The following *Formulation* types are available (see below the **Model background** paragraph for formulas with related coefficients):

- *Exponential*: via *Coefficient A,B*
- *Quadratic*: via *Coefficient A,B,C*
- *Logarithmic*: via *Coefficient A,B*
- *Reciprocal Logarithmic*: via *Coefficient A,B*
- *RPA*: the mass is calculated directly by *RPA*. Furthermore, the user can define the *Mass of other elements*, i.e. of fixed components as pipes, wires, etc.
- *Automatic*: the mass is computed depending on given *Engine characteristics* (see below). No coefficients are required to select the corresponding mass-performance correlation formula.

Engine characteristics define the main aspects of the engine. Not all of the following settings are always present in the configuration panel. Most of them are only required if *Automatic* or *RPA* was selected to estimate the mass or the specific impulse efficiency. All settings are used to choose values from an internal database for typical engines based on DLR-SART results.

- *Stage* defines the kind of stage the engine is mounted on. Available options are:
 - *First stage*
 - *Upper stage*
 - *Any*: used when the stage type for the engine is unknown or irrelevant.
- *Engine design* further classifies the engine as:
 - *Low cost*
 - *Baseline*

VEHICLE PARTS AND PROPERTIES MODELS

- *High performance*
- The *Engine cycle* defines the type of combustion:
 - *Staged combustion*
 - *Gas generator*
 - *Expander cycle*
 - *Pressure fed*
 - *Any*: used when the type of combustion is unknown or should not be restricted to a certain type
- The *Propellant type* defines the type of rocket engine and thus the number of mass flows:
 - *Mono propellant*
 - *Dual propellant*
- The *Oxidizer* and *Fuel* lists allow to choose oxidizer and fuel also in case of mono propellant rocket (in that case they are assumed to be bonded). Depending on the selected oxidizer, a restricted list of fuel types (in brackets below) is available:
 - *LOX liquid oxygen* (allows *LH2 Liquid Hydrogen*, *RP1 Kerosene*, *RG1 Kerosene* and *CH4 Methane*)
 - *N2O4 dinitrogen tetroxide* (allows *UDMH Dimethylhydrazine*, *CH6N2 Monomethylhydrazine*, *UH25 A50 Aerozine* and *N2H4 Hydrazine*)
 - *N2O Nitrous Oxide 184K* (allows *C32H66 Dotriacontane*)
 - *NH4CLO4* (allows solid *Al*)
 - *Unspecified*: in case no restriction on the propellant is desired. This option cannot be used for RPA equilibrium calculation and does not allow a selection of the fuel.

Tip: In case a different combination of oxidizer/fuel is desired or a species is not available in the lists, the option *RPA Configuration File* file can be used to initialize the engine with the desired combination.

- The following options are supported in case of *Automatic Engine mass correlation*:

Note: *Stage type Any* is not supported in this case.

Table 3.3: Possible combinations for automatic engine mass correlation (S - Staged Combustion, G - Gas Generator, E - Expander Cycle P - Pressure Fed, A - Any)

Oxidizer / Fuel	First Stage	Upper Stage
LOX / LH ₂	S, G, A	G, E, A
LOX / RP1	S, G, A	S, A
N ₂ O ₄ / UDMH	S, A	G, P, A
N ₂ O ₄ / UH25	S, A	G, P, A

VEHICLE PARTS AND PROPERTIES MODELS

N_2O_4 / A50	S	G, P
N_2O_4 / N_2H_4	S	G, P
Unspecified	S, G, A	A

- The following options are supported in case of *Automatic Isp efficiency definition*

Note: Stage type Any is not supported in this case.

Table 3.4: Possible combinations for automatic Isp efficiency definition (S - Staged Combustion, G - Gas Generator, E - Expander Cycle P - Pressure Fed, A - Any)

Oxidizer / Fuel	First Stage			Upper Stage		
	Baseline	Low Cost	High Perf.	Baseline	Low Cost	High Perf.
LOX / LH_2	S, G	S, G	S, G	G, E	G, E	G, E
LOX / RP1	S, G	G	S, G	S	S	S
N_2O_4 / UDMH	S, G	S	S	S, P	S, P	S, P
N_2O_4 / UH25	G			S, P	S, P	S, P
N_2O_4 / A50	G			S, P	S, P	S, P
N_2O_4 / N_2H_4				S, P	S, P	S, P
Unspecified						

Geometry settings define the dimensions and shape of engine and nozzle. This panel is only available if *RPA* is selected for the *Equilibrium calculation* and is *disabled* by default. However, it has to be *enabled* if also the *Engine mass correlation* is computed with *RPA* or if the thermal cycle within a rocket motor analysis shall be computed (see Section 5.15). Then, the following settings are required (see Fig. 3.5 for the physical meaning of the parameters):

- Contraction angle b
- $R1/Rt$
- Rn/Rt
- $R2/R2 \max$
- Nozzle shape provides several options and depending on the selected shape additional parameters are needed:
 - Conical shape defined by its *Half cone angle*
 - Parabolic shape defined by an *Initial parabola angle* (Tn), a *Final parabola angle* (Te) and a relative *Nozzle length* (usually between 60% and 80%); by default 30°, 10° and 80% are used.
 - TIC (truncated ideal contour) requires a *Surface temperature* ratio ($Tw/T0$) between 0.1 and 0.9 (0.2 by default). The upper value corresponds to an adiabatic nozzle.

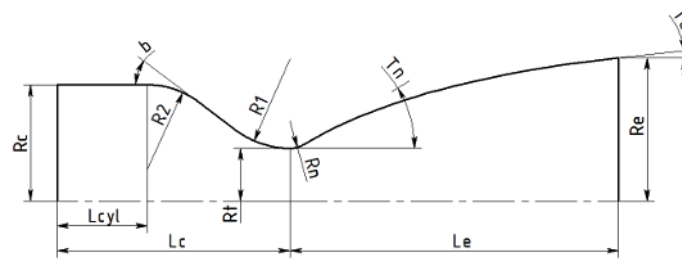


Fig. 3.5: Engine geometry parameters [110]

Throttle setting is an optional input which allows to define the throttle behaviour of the engine. It can be defined as *Constant law*, *Linear law* or *Profile law*.

It can be defined as *Common* or *Individual* (see Section 3.1.1). If *Common* the *Linear law* option is not available, since this would imply that the provided final value is set at the end of each phase (which basically results in an unrealistic profile). For a "common" linear law extended to all the phases, the profile law shall be used instead.

Note: If the *Throttle setting* is disabled, a fixed default value of 1.0 is used.

Assembly diameter settings specify the computation of the overall diameter of an engine assembly (see also **Model background** below):

- *Assembly diameter* defines the nominal diameter.
- *Free space factor* defines the additional free space between two nozzles as fraction of the nozzle exit diameter.
- *Diameter scaling factor* which is specified as a profile data table is used as multiplier of the nozzle exit diameter to obtain the assembly diameter. The available independent variables for the profile data table are *Number_Engines* and *Free_Space_Factor*.
- *Consider jet damping effect*, *Unburned propellant* and *Actuation damping* fields are explained in Section 3.2.2.1 *Profile Rocket*
- The generic settings *Mass distribution*, *Electrical power* and *Thermal characteristics* are explained in Section 3.1.2, Section 3.1.3 and Section 3.1.4 respectively. The latter two settings are available if the corresponding features in the ASTOS software have been installed and then enabled in the scenario *Active features*.

Background

The **engine performance** is computed from *Exhaust velocity* and *Characteristic velocity*. These values depend on the propellant type and chamber characteristics. Exhaust and characteristic velocity at equilibrium conditions can be computed by means of tabular data or internally by RPA [110]. In the first case the effective exhaust velocity c_e is interpolated from table data as a function of chamber pressure p_0 , mixture ratio f and expansion ratio r_A . The characteristic velocity c^* is interpolated from table data as a function of chamber pressure p_0 and mixture ratio f .

The **total burned mass flow** is determined by the definition of c^* :

VEHICLE PARTS AND PROPERTIES MODELS

$$\dot{m}_b(x) = \frac{\delta p_0 A_{th}}{c^*(x)} Z \quad (3.30)$$

where $x = [p_0, f]^T$ and δ the throttle setting, A_{th} the throat area and the engine sizing factor z . The **total mass flow** can be computed with the unburned mass flow fraction $u = \dot{m} / \dot{m}_u$:

$$\dot{m} = \frac{\dot{m}_b}{(1-u)} \quad (3.31)$$

The **thrust** is computed similarly to the other propulsion systems out of the vacuum thrust and the back pressure term:

$$T = \delta \dot{m}_b c_e \eta - p_a A_{th} r_A Z \quad (3.32)$$

where p_a is the ambient pressure and η the efficiency of the exhaust velocity.

In reality, c^* , c_e and the reaction efficiency (part of the total efficiency, see Section 5.15) are functions of the throttle. *RPA* considers this effect while computing the engine performance [111]. However, when the *Throttle setting* is selected as *optimizable*, *ASTOS* neglects its effect on the performance and uses the values (c^* , c_e and the reaction efficiency) computed for throttle equal to 1.0. This restriction has practical reasons: changing the throttle value at each time step implies continuous calling of the *RPA.dll* which significantly slows down the whole optimization process. The neglected effect is normally very small though (about 1% at throttle 0.5).

The *nozzle area* of a single engine is a function of the optimizable throat area and expansion ratio:

$$A_e = A_{th} r_A \quad (3.33)$$

An *Assembly diameter* of all nozzles is computed together with a *Free space factor* s , which indicates the required assembly space between two nozzles as a function of the nozzle diameter: $s = d_{free\ space} / d_{nozzle}$. The assembly diameter d_a needs to be provided as tabular data as a function of the engine sizing factor z and s . See Chapter 6 for information of how to put constraints on assembly and nozzle diameter. The assembly diameter is available as design parameter to allow a connection (see e.g. *Model Params Delta* constraint on Section 6.9) with other geometric design parameters of vehicle components or aerodynamics, e.g. a reference diameter.

The **length of the engine** can be computed by *RPA* or via the following formula [71], that relates the length to the vacuum thrust:

$$l_{engine} = 0.2259 T_{vac}^{0.194} \quad (3.34)$$

The engine model postulates an analytical relationship between the engine's dry mass m_e and the maximum vacuum thrust. In case of an *Automatic Formulation* this mass m_e computed automatically from database values according to the provided *Engine characteristics*. Alternatively, it is also possible to use one of the following analytical **mass correlation** formulas defined by the corresponding fitting coefficients:

- For an *Exponential* relationship of the form

VEHICLE PARTS AND PROPERTIES MODELS

$$m_e = aT_{\max}^b \quad (3.35)$$

- For a *Logarithmic* relationship of the form

$$m_e = a + b \ln T_{\max} \quad (3.36)$$

- For a *Reciprocal logarithmic* relationship of the form

$$m_e = \frac{T_{\max}}{a + b \ln T_{\max}} \quad (3.37)$$

- For a *Quadratic* relationship of the form

$$m_e = aT_{\max}^2 + bT_{\max} + c \quad (3.38)$$

where $T_{\max}$ is the **maximum vacuum thrust** given by

$$T_{\max} = c_e \dot{m}_b|_{\delta=1} \quad (3.39)$$

The constant mass correlation coefficients a , b , c are to be specified for $T_{\max}$ in units of Kilo-Newton, and engine dry mass m_e in units of Kilogram. Typical numeric values of coefficients are listed in Table 3.5 .

Table 3.5: Typical mass correlation coefficients for the engine model

Engine Type	Coefficients (Exponential Formulation)	
	a	b
LOX / LH ₂ rocket	3.6234	0.8731
Storable propellant rocket	0.8889	1.0254

The specific vacuum impulse I_{sp} is always computed using the following equation:

$$I_{sp} = \frac{c_e \eta}{g_0} \quad (3.40)$$

where c_e is the vacuum exit velocity and η the *Isp efficiency*.

The *geometry* section allows the user to dimension the whole engine. The combustion chamber can be sized:

- For required thrust level at a specific ambient condition;
- At a specific propellant mass flow rate;
- For a specific throat diameter.

The whole engine geometry (and mass) is computed during simulation as well as during optimization, but it is not provided as output in the `struct` file, as done in the thermal analysis. The geometry computation affects the performance by computing all the efficiencies, otherwise not considered (e.g. divergence efficiency and nozzle efficiency).

Model output

VEHICLE PARTS AND PROPERTIES MODELS

The following output is produced by each throat design engine and is added to the auxiliary functions in *Curve Plots* and *Condition Plots* (*Results* tree).

Table 3.6: Throat design engine output

Type	Name	Description	Availability
Function	engine_mass	mass of the engine	always
	mass_flow_oxidizer	oxidizer mass-flow	dual propellant engine
	mass_flow_fuel	fuel mass-flow	dual propellant engine
	v_ex	exhaust velocity	RPA used for equilibrium calculation
	c_star	characteristic velocity	RPA used for equilibrium calculation
Scalar	isp_reaction_efficiency	Isp reaction efficiency	RPA used for Isp efficiency calculation
	isp_nozzle_efficiency	Isp nozzle efficiency	RPA used for Isp efficiency calculation

Optimization

Following entries specify the optimizable design parameters with initial value and bounds:

- *Chamber pressure*: a bounded float record specifying the initial value and its bounds of the chamber pressure in units of pressure. The chamber pressure is required for interpolation of the exhaust and characteristic velocity.
- *Mixture ratio*: a bounded float record specifying the initial value and its bounds of the mixture ratio of oxidizer to fuel. The mixture ratio may be required for interpolation of the exhaust and characteristic velocity.
It is also possible to have the mixture ratio as control.
- *Expansion ratio*: a bounded float record specifying the initial value and its bounds of the expansion ratio of the nozzle exit to nozzle throat area. The expansion ratio is required for interpolation of the exhaust velocity and computation of the exit area.
It is also possible to have the expansion ratio as control.
- *Throat area, vacuum thrust or mass flow*: a bounded float record specifying the initial value and its bounds of the forth engine parameter.
- *Throttle setting*: a bounded float record specifying the initial value and its bounds of the throttle settings between 0.0 and 1.0.
It is although possible to have the throttle setting as control.
- *Assembly diameter*: a bounded float record specifying the initial value and its bounds of the assembly diameter.

Note: If RPA is used for mass and geometry computation during optimization, tables for the engine mass, chamber length, nozzle length and nozzle exit radius are computed, which have the four design parameters (chamber pressure, mixture ratio, expansion ratio and sizing parameter) as independent variables. Tables are only created if at least one of these parameters is set as *optimizable*. For each optimizable parameter, the interval defined by the bounds is discretized using four equidistant values.

Cost functions

This model supports a cost function of type *User Propulsion* (see Section 7.17):

- *Initial_Min_Thrust_Vacuum*, allows to minimize the vacuum thrust at the beginning of the specified phase, the force is computed in Kilo-Newton and it is not considering the throttle setting.

Constraints

This model supports constraints of type *Equipment Design* (see Section 6.9) with the following *Parameters*:

- *Delta_Assembly_Diameter* ensures, that the internal design parameter *Assembly diameter* d corresponds to the computed assembly diameter $d_a = d$. It computes the difference between both values and the limit is always zero in units of length (meter). This constraint should be always added, if the assembly diameter is of interest.
- *Nozzle_Diameter* allows to constraint the nozzle exit diameter d_{nozzle} of a single engine. It computes the nozzle exit diameter in units of length (Meter).
- *Nozzle_Length* allows to constraint the nozzle length l_{nozzle} of a single engine. It computes the nozzle length in units of length (Meter).
- *Isp_Vacuum* allows to constraint the vacuum specific impulse of a single engine computed in units of time (Second).
- *Isp* allows to constraint the specific impulse of a single engine computed in units of time (Second).
- *Thrust_Vacuum* allows to constraint the vacuum thrust of a single engine computed in units of force (Newton).
- *Thrust* allows to constraint the effective thrust of a single engine computed in units of force (Newton).

3.2.2.6 Typical Rocket with Variable Mixture Ratio

Description

This engine can be used to model a liquid rocket engine whose mass flow is a function of the controllable throttle setting and fuel mixture ratio. This model applies to a family of rocket motors with dual mass flows.

The Typical Variable Mixture Ratio Rocket (*Typical VMR Rocket*) is equal to the *Throttleable Rocket* (Section 3.2.2.2) in every aspect, except that it provides an additional control for the fuel mixture ratio. The new aspect in this model regards the total propellant mass flow, which can be defined as non-linear function of the fuel mixture ratio. The same applies to specific impulse and vacuum thrust.

Configuration

VEHICLE PARTS AND PROPERTIES MODELS

When setting up the model, the only differences between a *Typical VMR Rocket* and a *Throttleable* concern the mixture ratio, as explained here:

- *Mixture ratio*, which can be defined as a constant value (*Constant Law*) as a function linear in time *Linear Law* or *Profile Law*.
- *Mixture ratio efficiency*, which can be used to reduce the mass flow depending on the current mixture ratio value.
- The generic settings *Mass distribution*, *Electrical power* and *Thermal characteristics* are explained in Section 3.1.2, Section 3.1.3 and Section 3.1.4 respectively. The latter two settings are available if the corresponding features in the ASTOS software have been installed and then enabled in the scenario *Active features*.

Note: The usage of the independent variables is free. But to define a correct *Typical VMR* rocket, at least the mass flow data and the mass flow ratio w should be provided as a function of the fuel mixture ratio f .

Model background

The fuel mixture ratio f is defined as the ratio of oxidizer and fuel mass flow, the total mass flow is the sum of the two:

$$f = \frac{\dot{m}_{O_2}}{\dot{m}_{H_2}} \quad ; \quad \dot{m} = \dot{m}_{O_2} + \dot{m}_{H_2} \quad (3.41)$$

The fuel mixture ratio can be defined as optimizable control like the throttle setting. Given the total mass flow and fuel mixture ratio, oxidizer and fuel mass flow components are computed as

$$\dot{m}_{O_2} = \frac{\dot{m}}{1+f} \quad ; \quad \dot{m}_{H_2} = \frac{\dot{m}f}{1+f} \quad (3.42)$$

The computation of the axial thrust is derived from (9.9). The vector of independent variables is extended by the mixture ratio $x = [t, h, p, \rho, q, M, \alpha, \sigma, f]^T$.

$$T_{\text{axial}} = g_0 I_{sp}(x) \dot{m}_b(x) w(x) \sigma - A_e(x) p = T_{\text{vac}}(x) w(x) \sigma - A_e(x) p \quad (3.43)$$

where $w(x)$ is the mixture ratio efficiency.

3.2.2.7 Solid Design Rocket

Description

This model allows the design of a solid propulsion whereas the nozzle exit area and several parameters of the selected mass flow rate model (and therefore the thrust) are optimizable.

Configuration

In this paragraph only settings differing from those presented in the previous chapters are discussed:

VEHICLE PARTS AND PROPERTIES MODELS

- The *Nozzle exit area* can be defined as optimizable parameter.

Due to the physical fact that the maximum mass flow rate is directly linked to the nozzle exit area, it is necessary to introduce a constraint that assures

$$\dot{m}_{\max} = \rho A_e I_{sp} g_0 (1 - u) \quad (3.44)$$

ρ , I_{sp} , g_0 and u are fixed values (ρ is the density of the exhaust gases). Therefore the user has to provide a *Model Params Ratio* constraint (see Section 6.9) that ensures a constant ratio between $\dot{m}_{\max}$ and A_e :

$$\frac{\dot{m}_{\max}}{A_e} = \text{const.} \quad (3.45)$$

Internally (for aerodynamic reasons) the code guarantees that the nozzle exit area never exceeds the area of its related component (with component diameter d):

$$A_e \leq \frac{d^2 \pi}{4} \quad (3.46)$$

- The *Unburned prop. mass fraction* defines the fraction of the mass flow not contributing to the engine thrust.
- *Vacuum Isp* is the normalized specific vacuum impulse. This quantity is normalized by the standard gravitational acceleration value of $g_0 = 9.80665 \text{ m/s}^2$.
- *Isp efficiency* is a coefficient which multiplies the vacuum Isp to compute the final Isp value. It allows the user to take into account eventual losses in the engine.
- The *Mass flow* panel allows the user to specify the requested mass flow profile. The available models differ from those presented in the previous chapters due to the specifics of solid rocket engines. All the models are presented below.
- *Thrust scaling factor*. The user can provide a positive values used to multiply the current thrust magnitude. The scaling factor may be useful to consider cases when the propulsion force vector does not lay exactly in the axial axis, e.g. $c = \cos \varepsilon$, where ε is the angle of the thrust vector.
- The *Consider jet damping effect* is caused by the flame of the engine, which creates a small resistance to any change of attitude. This effect can become quite important in case of 6-DOF scenario of small launchers (e.g. sounding rockets). It is function of the distance between the center of gravity and the nozzle plane. The computation does not require any additional user input and follows the mathematical model presented in [68]. This effect is quite negligible if applied to big launchers.
- The *Actuation damping* model allows the user to consider damping when simulating thrust vector control (TVC). When controls are applied to the thrust vector, a second order transfer function can be defined in order to model the reaction of the dynamic system to the provided controls. The commanded nozzle deflection angle $\Delta\delta_C$ differs from the realised one $\Delta\delta_R$ depending on the *Damping* ζ , *Natural frequency* ω and *Gain* K selected. The relationship is defined as follows:

$$\Delta\delta_R(t) = \frac{K\omega_a^2}{s^2 + 2\zeta\omega_a s + \omega_a^2} \Delta\delta_C \quad (3.47)$$

- The generic settings *Mass distribution*, *Electrical power* and *Thermal characteristics* are explained in Section 3.1.2, Section 3.1.3 and Section 3.1.4 respectively. The latter two

settings are available if the corresponding features in the *ASTOS* software have been installed and then enabled in the scenario *Active features*.

Model background

In general the computation of the mass flow rate is

$$\dot{m} = \frac{\bar{\dot{m}}_x \dot{m}_{\max}}{1 - u} \quad (3.48)$$

where $\bar{\dot{m}}_x$ is the normalized mass flow rate (normalized in time and mass flow). $\dot{m}_{\max}$ is the maximum mass flow rate of the burnt propellant and u is the unburned propellant mass fraction. The normalized mass flow rate $\bar{\dot{m}}_x$ is calculated depending on the users choice of mass flow profile:

- *Normalized Profile*. Within the Normalized Profile the user has the possibility to define normalized points (normalized with respect to time and mass flow rate). Between these points the mass flow rate is approximated. The user can select two different approximation methods:

- Interpolation
- Alpha Polynomial

For these approximation methods several settings can be made (e.g. selection of the interpolation method).

- The *Star* profile has the following shape (the name "star profile" is related to the geometry of the inner grain of the booster propellant):

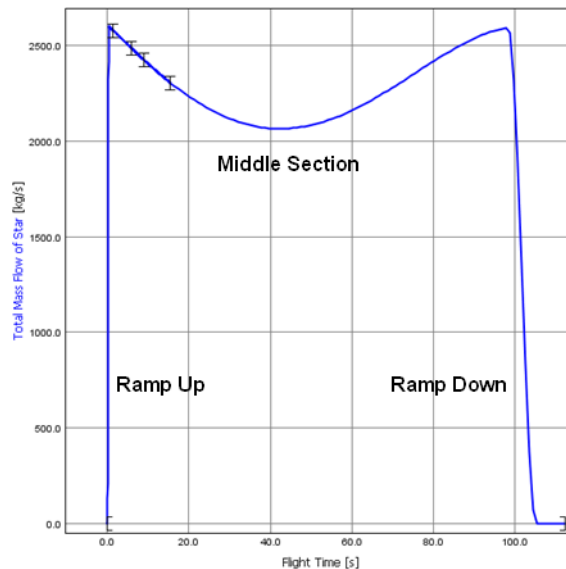


Fig. 3.6: Mass flow rate of the Star profile

The mass flow rate consists of a predefined ramp up profile with a duration of 0.4 seconds. The duration of the middle section varies depending on the propellant mass that is defined

VEHICLE PARTS AND PROPERTIES MODELS

in the related component. The ramp down is also predefined and has a duration of 7 seconds.

- The *Star Optimizable Low Mark* has the same shape like the *Star* profile. The difference is that the user can define a burn time and mass flow rate value (and bounds) for the lowest mark of the middle section. Therefore this point becomes optimizable.

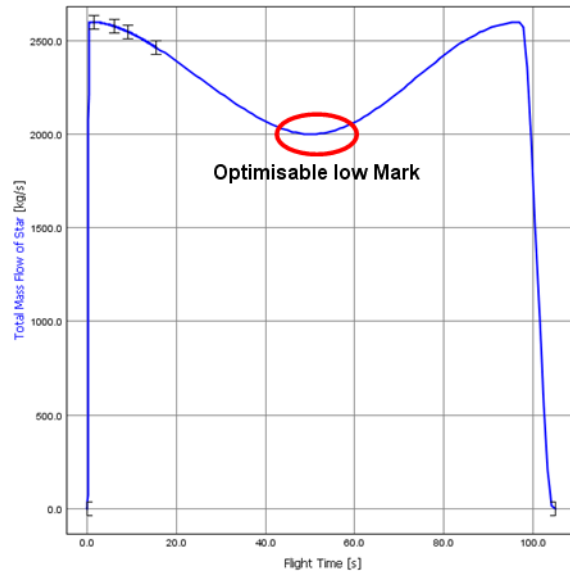


Fig. 3.7: Mass flow rate of the star profile with optimizable low mark

- *Star Advanced*. Within the *Star Advanced* profile it is possible to select a predefined *Ramp up / Ramp down* profile or to define normalized points with approximation in between the points. Furthermore the durations of the ramps are definable. The middle section consists of a constant section (with maximum mass flow rate and optimizable duration) followed by a polynomial progression with low mark (optimizable low mark point). Furthermore the maximum mass flow value at the beginning of the ramp down profile can be defined by the user (*Ramp down fraction*). The following picture illustrates the whole mass flow rate progression of this model.

VEHICLE PARTS AND PROPERTIES MODELS

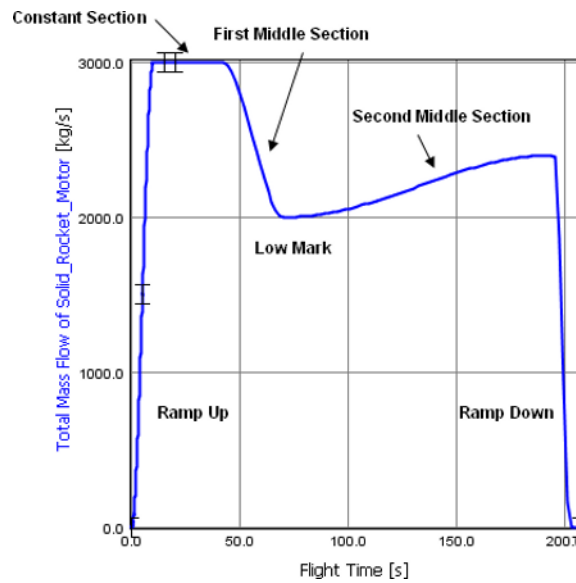


Fig. 3.8: Mass flow rate of the Star Advanced profile

- The *Tubular* profile (this name is related to the geometry of the inner grain of the booster propellant) consists of a predefined ramp up profile with a duration of 0.4 seconds. The mass flow rate at the end of ramp up is about $0.4 \cdot \dot{m}_{\max}$ whereas the maximum mass flow rate $\dot{m}_{\max}$ is user provided input data. The Middle Section consists of a progression at which the mass flow rate increases to $1.0 \cdot \dot{m}_{\max}$. The duration of this progression depends on the propellant mass that is defined in the related component. The end of this model consists of a ramp down profile (duration 7 seconds) which simulates the burn out of the booster.

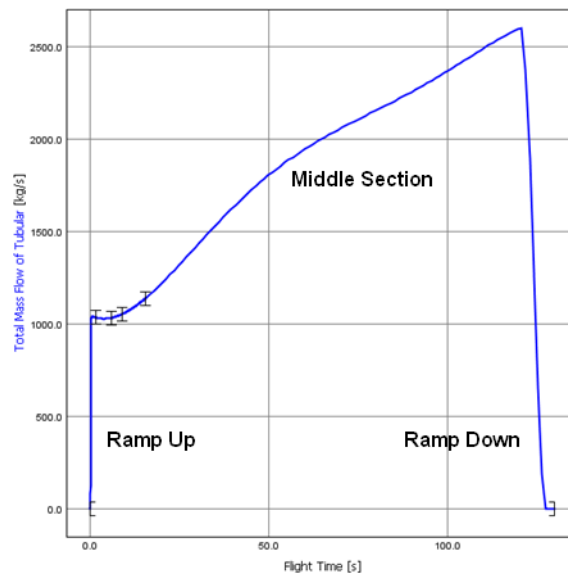


Fig. 3.9: Mass flow rate of the Tubular profile

- *Tubular Advanced* profile. The shape of the *Tubular Advanced* profile is the same as the *Tubular* profile but it is possible to choose between a predefined ramp up / ramp down

VEHICLE PARTS AND PROPERTIES MODELS

profile or to define normalized points with approximation in between the points. Furthermore the durations of the ramps as well as the ratio between the mass flow rate at the end of ramp up and the beginning of ramp down are definable.

The thrust model of the *Solid Design Rocket* is:

$$T = C (T_{vac} - A_e p) = C (\eta I_{sp} \bar{m}_x \dot{m}_{max} g_0 - A_e p) \quad (3.49)$$

where I_{sp} is the specific vacuum impulse (which can be defined as fixed value or profile) and η is the impulse efficiency. g_0 stands for the standard acceleration of gravity, A_e is the nozzle exit area, p the ambient atmospheric pressure, T_{vac} the thrust in vacuum condition and C a constant thrust scaling factor.

Constraints

This model supports constraints of type *Equipment Design* (see Section 6.9) with the following *Parameters*:

- *Nozzle_Diameter* allows to constraint the nozzle exit diameter d_{nozzle} of a single engine. It computes the nozzle exit diameter in units of length (Meter).
- *Nozzle_Length* allows to constraint the nozzle length l_{nozzle} of a single engine. It computes the nozzle length in units of length (Meter).
- *Thrust_Vacuum* allows to constraint the vacuum thrust of a single engine computed in units of force (Newton).
- *Thrust* allows to constraint the effective thrust of a single engine computed in units of force (Newton).

3.2.2.8 Airbreathing Engine

Description

This model allows the design of an air-breathing engine whose thrust and fuel flow depend on up to eight independent variables. No fuel budgeting is included in this model. The fuel mass of the associated vehicle stage can be modelled by the vehicle component models described in Section 3.3.

Configuration

- *Axial thrust* is a model specific setting and defines the thrust components in the x-axis of the actuator's body-fixed frame.
- *Normal thrust* is a model specific setting and defines the thrust component in z-axis of the actuator's body-fixed frame.
- The *Mass flow* and *Structural mass* settings are described in Section 3.2.2.1
- The *Throttle setting* is described in detail in Section 3.2.2.2
- The generic settings *Mass distribution*, *Electrical power* and *Thermal characteristics* are explained in Section 3.1.2, Section 3.1.3 and Section 3.1.4 respectively. The latter two

VEHICLE PARTS AND PROPERTIES MODELS

settings are available if the corresponding features in the *ASTOS* software have been installed and then enabled in the scenario *Active features*.

Model background

The nominal thrust components and the mass flow rate are interpolated from tabular input data that may be functions of the independent variables: burn time t , altitude h , ambient pressure p , ambient density ρ , dynamic pressure q Mach number M , angle of attack α , and the controllable equivalence ratio R .

$$x = [t, h, p, \rho, q, M, \alpha, R]^T \quad (3.50)$$

In case of a Ramjet engine the equivalence ratio R is defined as the ratio between the current fuel-to-air flow rate and a reference value for stoichiometric combustion.

$$R \equiv \frac{\left(\frac{\dot{m}_{\text{Fuel}}}{\dot{m}_{\text{Air}}}\right)}{\left(\frac{\dot{m}_{\text{Fuel}}}{\dot{m}_{\text{Air}}}\right)_{\text{ref}}} \quad (3.51)$$

3.2.2.9 User-Defined Propulsion Model

Description

This model allows the use of a propulsion system library implemented by the user. Such a library may support optimizable controls, design parameters and a mass estimation formula.

Configuration

The following settings are required to setup the model:

- *Library file (DLL)* defines a string with the path and filename to the user library. Both an absolute path or a path relative to the current scenario folder can be provided. The maximum length is 2048 characters.
- The *User parameters* field can accommodate a string with a maximum length of 1028 characters. The initialization routine of the library receives this string as an input parameter. The parameter string can contain e.g. file names, numerical value, etc.
- The *Structural mass* setting is described in Section 3.2.2.1
- The generic settings *Mass distribution*, *Electrical power* and *Thermal characteristics* are explained in Section 3.1.2, Section 3.1.3 and Section 3.1.4 respectively. The latter two settings are available if the corresponding features in the *ASTOS* software have been installed and then enabled in the scenario *Active features*.

The model interface for the user-defined propulsion model is described in detail in chapter Section 11.2.

3.2.3 Spring Damper Models

Description

This actuator models damped spring forces acting between two vehicle components. There are two different subtypes of spring dampers:

- *Translational* is a damped spring where the load is a tension or compression, see Section 3.2.3.1
- *Rotational* is a damped torsion spring where the load is either a torque or twisting force, see Section 3.2.3.2

Note: These models do not influence the propagation in ASTOS! They are only used for multibody systems.

3.2.3.1 Translational Spring Damper

Description

A *Translational* spring damper operates with a tension or compression load. It has two attachment points which are defined in the vehicle builder by adjustable mounting points on the corresponding linked components.

Configuration

The following parameters are available:

- *Load type* defines how the load of the spring is calculated. These options are available:
 - *Linear* means the load follows a linear function of the spring length
 - *Quadratic* means the load follows a quadratic function of the spring length
 - *Cubic* means the load follows a cubic function of the spring length
- *Hardstop* is only available for *Linear Load types*. It defines if the spring has a minimum or maximum displacement. These options are available:
 - *None* means there is no minimum or maximum displacement of the spring
 - *Lower_Bound* defines a minimum displacement and the load is set to zero when the actual length of the spring is smaller than the provided value
 - *Upper_Bound* defines a maximum displacement and the load is set to zero when the actual length of the spring is greater than the provided value
- *X axis*, *Y axis* and *Z axis* are optional and specify the axis/axes along which the spring damper is acting. If *enabled*, the following settings are required:
 - *Spring stiffness* is a measure of the resistance to deformation of the spring

VEHICLE PARTS AND PROPERTIES MODELS

- *Damping constant* is the damping ratio and describes how oscillations decay
- *Relaxed length/Minimum displacement/Maximum displacement* is the length of the spring when the resulting force is zero, or the minimum/maximum displacement depending on the specified *Hardstop*.
- The generic settings *Mass distribution*, *Electrical power* and *Thermal characteristics* are explained in Section 3.1.2, Section 3.1.3 and Section 3.1.4, respectively. The latter two settings are available if the corresponding features in the *ASTOS* software have been installed and then enabled in the scenario *Active features*.

The *Spring stiffness*, *Damping constant* and *Relaxed length/Minimum displacement/Maximum displacement* can either be defined by a constant *Value* or by providing a *Profile* as function of the time (refer to Profile Data Tables in User Manual for more information about profiles).

Background

The force of the spring damper model is calculated depending on the specified *Load type*. For *Linear*, *Upper_Hardstop* and *Lower_Hardstop* the following equation is used:

$$F = -K \cdot (d - d_0) - C \cdot d' \quad (3.52)$$

where

- *F* is the spring damper force (load) acting along the line defined by the mounting points
- *d* is the distance between the mounting points, the actual length of the spring
- *d₀* is the *Relaxed length*, the length of the spring (distance between mounting points) at which the spring force is zero
- *K* is the *Spring stiffness* constant
- *C* is the *Damping constant*

In case of *Quadratic* the force is according to

$$F = -K \cdot (d - d_0) \cdot |d - d_0| - C \cdot d' \cdot |d'| \quad (3.53)$$

and in case of *Cubic* the force equals

$$F = -K \cdot (d - d_0)^3 - C \cdot (d')^3 \quad (3.54)$$

3.2.3.2 Rotational Spring Damper

Description

A *Rotational* spring damper works by torsion or twisting. It has two attachment points which are defined in the vehicle builder by adjustable mounting points on the corresponding linked components.

Configuration

VEHICLE PARTS AND PROPERTIES MODELS

The following parameters are available:

- *Load type* defines how the load of the spring is calculated. These options are available:
 - *Linear* means the load follows a linear function of the spring length
 - *Quadratic* means the load follows a quadratic function of the spring length
 - *Cubic* means the load follows a cubic function of the spring length
- *Hardstop* is only available for *Linear Load types*. It defines if the spring has a minimum or maximum displacement. These options are available:
 - *None* means there is no minimum or maximum displacement of the spring
 - *Lower_Bound* defines a minimum displacement and the load is set to zero when the actual angle of twist of the spring is smaller than the provided value
 - *Upper_Bound* defines a maximum displacement and the load is set to zero when the actual angle of twist of the spring is greater than the provided value
- *X axis*, *Y axis* and *Z axis* are optional and specify the axis/axes about which the spring damper is acting. If *enabled*, the following settings are required:
 - *Spring stiffness* is the torsion coefficient, a measure of the resistance to deformation of the spring
 - *Damping constant* is the damping ratio and describes how oscillations decay
 - *Relaxed angle/Minimum displacement/Maximum displacement* is the angle of twist of the spring when the resulting torque is zero, or the minimum/maximum displacement depending on the specified *Hardstop*
- The generic settings *Mass distribution*, *Electrical power* and *Thermal characteristics* are explained in Section 3.1.2, Section 3.1.3 and Section 3.1.4, respectively. The latter two settings are available if the corresponding features in the ASTOS software have been installed and then enabled in the scenario *Active features*.

The *Spring stiffness*, *Damping constant* and *Relaxed angle/Minimum displacement/Maximum displacement* can either be *defined by* a constant *Value* or by providing a *Profile* as function of the time (refer to Profile Data Tables in User Manual for more information about profiles).

Background

The force of the spring damper model is calculated depending on the specified *Load type*. For *Linear*, *Upper_Hardstop* and *Lower_Hardstop* the following equation is used:

$$F = -K \cdot (a - a_0) - C \cdot a' \quad (3.55)$$

where

- F is the spring damper torque (load) acting about the specified axis
- a is the actual angle of twist
- a_0 is the *Relaxed angle*, the angle of twist at which the spring torque is zero

VEHICLE PARTS AND PROPERTIES MODELS

- K is the torsion coefficient, the *Spring stiffness*
- C is the *Damping constant*

In case of *Quadratic* the force is according to

$$F = -K \cdot (a - a_0) \cdot |a - a_0| - C \cdot a' \cdot |a'| \quad (3.56)$$

and in case of *Cubic* the force equals

$$F = -K \cdot (a - a_0)^3 - C \cdot (a')^3 \quad (3.57)$$

3.2.4 Angular Momentum and Torque Actuator Models

Description

Angular momentum and torque actuators can be used for torque generation, attitude control or stabilization of satellites and other types of vehicles. They do not generate thrust and do not require an associated tank. The following models are available:

- *Flywheel* is a rotating mechanical device with a constant rotational speed. See Section 3.2.4.1.
- *Magnetorquer* is an electromagnetic coil which generates torque in an external magnetic field. See Section 3.2.4.2.

3.2.4.1 Momentum Wheel

Description

This type of actuator is spun at a constant angular speed to generate a stabilizing action around its rotational axis. This can be used for example to stabilize the attitude of a satellite. The momentum wheel spins about the x axis of the local actuator coordinate system.

Configuration

The momentum wheel is configured with the following parameters:

- *Mass* specifies the spinning mass of the wheel
- *Moment of inertia* of the spinning wheel.
- *Initial speed* is the constant angular speed at which the wheel is rotating.
- The generic settings *Mass distribution*, *Electrical power* and *Thermal characteristics* are explained in Section 3.1.2, Section 3.1.3 and Section 3.1.4 respectively. The latter two settings are available if the corresponding features in the *ASTOS* software have been installed and then enabled in the scenario *Active features*.

The position and orientation of the wheel is set in the vehicle builder.

3.2.4.2 Magnetorquer

Description

The magnetorquer model is an electromagnetic coil also referred to as solenoid. By running a direct current through the solenoid it generates a magnetic field. This field interacts with the earth's magnetic field and generates a torque for attitude control.

Configuration

The magnetorquer is configured with the following parameters:

- *Definition* is a drop down menu offering to directly specify a *Magnetic Moment* or *Coil Parameters* from which the magnetic dipole moment vector is calculated.
- *Magnetic moment* can either be *Defined by* a constant *Value* or by providing a *Profile* as function of the applied current (refer to Profile Data Tables in User Manual for more information about profiles).
- *Coil Parameters* specifies physical quantities of the solenoid. The following inputs are available:
 - *Cross section* of the solenoid
 - *Number of windings* of the solenoid
 - *Ferromagnetic core* adds a core made of a ferromagnetic metal to the solenoid thereby significantly increasing the flux density of the created field. The core requires some magnetic property inputs:
 - *Length* of the core which is assumed to be equal to the length of the solenoid.
 - *Coercivity* is the strength of a constant external magnetic field required to reduce the magnetization of the core to zero after the core was fully saturated.
 - *Coercive magnetic flux density derivative* is the first derivative of the internal magnetic flux density at the point of Coercivity.
 - *Saturation* describes the external *Magnetic field strength* necessary to saturate the internal *Magnetic flux density*.
 - *Disable remanent moment* is a run-time variable control to disable the hysteresis inside the core. By default the ferromagnetic core is subject to hysteresis effects. By setting the control value to 1.0 the magnetic dipole moment of the magnetorquer is zero if the current in the solenoid is zero.
- *Inductance* is the relationship between the change in electric current and the voltage applied to the solenoid.
- *Impedance* is the electrical resistance of the coil. It can be specified as constant by setting the *Defined by* drop down menu to *Value* or it can be a *Profile* over time or temperature. Refer to Profile Data Tables in User Manual for more information about profiles. If the *Thermal characteristics* are disabled the temperature of the magnetorquer is assumed to be constant at 300 Kelvin.

VEHICLE PARTS AND PROPERTIES MODELS

- *Supply voltage* is the voltage applied to the magnetorquer. The *Control Law* offers the possibilities to use a *Constant Law* (constant voltage), a *Linear Law* (linearly changing voltage) or a *Profile Law* (Voltage value specified over time). Refer to Profile Data Tables in User Manual for more information about profiles. It can also be defined as *Common* or *Individual* (see Section 3.1.1).
- *Structural mass* of the magnetorquer
- The generic settings *Mass distribution*, *Electrical power* and *Thermal characteristics* are explained in Section 3.1.2, Section 3.1.3 and Section 3.1.4 respectively. The latter two settings are available if the corresponding features in the ASTOS software have been installed and then enabled in the scenario *Active features*.

Background

If a voltage is applied to the magnetorquer the current changes according to

$$\dot{I} = \frac{1}{L} \cdot (V - R \cdot I) \quad (3.58)$$

with

- $\dot{I}$ being the current derivative,
- L the inductance of the solenoid,
- V the voltage applied,
- R the impedance,
- I the current in the solenoid.

In case *Definition* is set to *Coil Parameters* the magnetic dipole is calculated as

$$m_{total} = m_{solenoid} + m_{core} \quad (3.59)$$

$$m_{solenoid} = n \cdot I \cdot A \quad (3.60)$$

$$m_{core} = I \cdot A \cdot \left(\frac{B_{internal}}{\mu_0} - H_{external} \right) \quad (3.61)$$

with

- m_i being the dipole moment of i ,
- n the number of windings in the solenoid,
- A the cross section of the solenoid which is assumed to be equal to the cross section of the core
- l being the length of the core which is assumed to be equal to the length of the solenoid
- $B_{internal}$ the magnetic flux density inside the core
- μ_0 the vacuum permeability,
- $H_{external}$ the magnetic field strength of the solenoid

$B_{internal}$ does not depend linearly on the $H_{external}$ but also on the history of $H_{external}$ (Hysteresis). An example of the hysteresis effects in ferromagnetic materials is shown in Fig. 3.10.

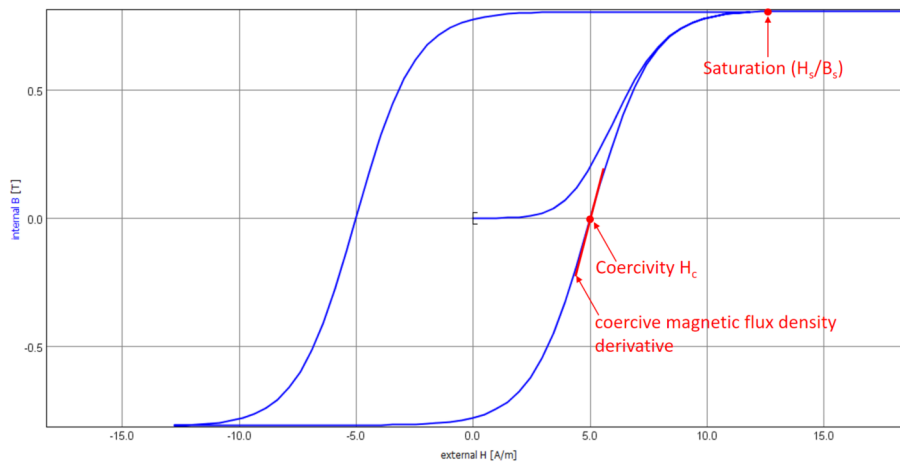


Fig. 3.10: Hysteresis curve (blue) with the required inputs for the Ferromagnetic core (red)

3.2.5 Generic Actuator

Description

A generic actuator model provides forces and moments but does not consider a model of a specific physical device. All force and moment vectors can be either directly defined, for example as profile, or specified as optimizable controls. Force and moment are applied in the actual attitude frame of the actuator. This model is suitable for the ASTOS/Simulink environment.

Configuration

The generic actuator model has the following specifications:

- *Forces* is the force vector in Cartesian representation (X, Y, Z)
- *Moments* is the moment vector in Cartesian representation (X, Y, Z)
- *Scaling Factors* is optional and specifies scaling factors for:
 - *Forces* is the force vector in Cartesian representation (X, Y, Z)
 - *Moments* is the moment vector in Cartesian representation (X, Y, Z)
- *Mass flow* is optional and specifies the mass flow rate of the actuator model

Forces, *Moments* and *Mass flow* can be either defined as *Common* or *Individual* (refer to Section 3.1.1). For these setting the following *Control laws* are available:

- *Constant law* describes a constant value
- *Linear law* describes a linear behaviour
- *Profile law* describes the value as a function of one of the following independent variables:

- *Flight_Time* is the duration elapsed since initial time of phase one
- *Mission_Time* is the duration elapsed since the mission start time
- *Phase_Time* is the duration elapsed since the beginning of each phase

Background

Defining the *Scaling Factors* as optimizable controls can be used to switch the actuator between ON (value equal to 1.0) and OFF (value equal to 0.0), for example. In this case, it is not required to specify the force and moment components as optimizable quantities as well. As an example they can be as profile but not optimizable. Having two optimizable quantities for the force x-component and its scaling factor is possible but typically not meaningful.

3.3 Component Models

The models in this chapter provide different options to define structural masses and propellant masses if the model contains a tank:

- *Auxiliary* defines a constant structural mass with user-definable geometric shape which can be used to model structural masses which are not included in other component models (e.g. mass of fuel pipes etc.), see Section 3.3.3.1
- *Basic Design Payload Fairing* defines a fairing model with optimizable mass scaled by the payload mass, see Section 3.3.3.3
- *Basic Vehicle Design Stage* defines a stage model with optimizable propellant mass and a structural mass which scales according to the propellant volume using a fixed coefficient, see Section 3.3.2.2
- *Basic Vehicle Stage* defines a simple stage model with a structural mass and an optimizable propellant mass, see Section 3.3.2.1
- *Deployable Boom*, see Section 3.3.6
- *Detailed Design Payload Fairing* defines a detailed fairing design model with optimizable diameter, optimizable length and masses for different sections of the fairing, see Section 3.3.3.4
- *Detailed Vehicle Design Booster* defines a detailed booster design model with optimizable propellant mass and individual structural masses for skirts, thrust frames etc., but also calculates forces acting on the connecting bolt to the vehicle stage, see Section 3.3.2.7
- *Detailed Vehicle Design Stage* defines a detailed stage design model with optimizable propellant mass and individual structural masses for skirts, interstages, thrust frames etc., see Section 3.3.2.6
- *Dual Propellant Design Stage* defines a stage model with optimizable propellant mass and optimizable mixture ratio with a fixed propellant to structural mass ratio, see Section 3.3.2.5

VEHICLE PARTS AND PROPERTIES MODELS

- *Dual Propellant Stage* defines a stage model with a structural mass and two optimizable propellant masses, i.e. fuel and oxidizer, see Section 3.3.2.4
- *Linear Vehicle Design Stage* defines a stage model with optimizable propellant mass and structural mass as linear function of the total propellant mass, see Section 3.3.2.3
- *Payload* defines a structural component with nominal mass and optimizable scaling factor, see Section 3.3.3.2
- *Reentry Vehicle* provides a detailed re-entry vehicle design model with individual structural mass calculation and for the definition of a thermal protection system, see Section 3.3.4
- *User* provides a programming interface, where the user can code his own components including real parameters, optimization constraints and algebraic output functions, see Section 3.3.5

3.3.1 Common Characteristics

Description

This section describes the configuration of the characteristics available for the vehicle components.

The sloshing feature described in Section 3.3.1.2 can be enabled only for vehicle stage components (see Section 3.3.2 Vehicle stage).

3.3.1.1 Flexibility

Description

In case of scenarios where the vehicle structural elasticity plays an important role in the system dynamics, it may be of interest to consider the effect of the bending deformation of the vehicle's mechanical system on the vehicle dynamics. The modelling of the vehicle as an elastic body allows the computations of the bending natural frequencies of the system.

Typically only the vehicle's low frequency oscillations affect the system dynamics. During the simulation, the structural elastic deformations are superimposed to the rigid displacements in order to reproduce a more accurate dynamics.

The *Flexibility* setting is available for all component models.

Note: The *Flexibility* setting is available only if the **Multibody Dynamics** feature is *Enabled*.

Configuration

The component's *Flexibility* setting is disabled by default. If enabled the following options are displayed:

VEHICLE PARTS AND PROPERTIES MODELS

- The *Model* specifies the flexibility model which shall be used for the flexibility computation. Currently only the flexible analytical beam (*Beam* setting in the drop-down menu) is available.
 - *Beam* contains all beam-related settings:
 - *Young modulus* : is a measure of the ability of the material to withstand deformation under tension or compression and it represents the stiffness of a solid material. This is the Young modulus assigned to the component and can be set by considering the main component material.
 - *Damping ratio* : is a dimensionless measure describing oscillations decay. This parameter considers the processes that dissipate the energy stored in the oscillations and depends on the material and structure geometry. The damping ratio here defined is considered constant for any mode shapes and frequencies. Typical values for spacecraft and launchers vary between 1% and 7%.
 - *Length* : defines the length of the flexible beam. The *Length* option is preset to *Default* so that the beam length is computed automatically based on the component geometrical properties.
 - *Cross section* : defines the area of the perpendicular section of the beam. The *Cross section* option is preset to *Default* so that it is computed automatically based on the component geometrical properties.
 - *Area moment of inertia* : defines the moment of inertia of the plane area of the beam. The *Area moment of inertia* option is preset to *Default* so that it is computed automatically based on the component geometrical properties. When the option is set to *Custom*, the specified user value represents the area moment of inertia when the component tank is full of propellant. The area moment of inertia value changes proportionally with the amount of the propellant left.
- Tip:** The product of the Young modulus and the area moment of inertia is proportional to the stiffness of the beam. The user can thus tune the component flexible behavior by adjusting these parameters.
- *Boundary conditions* : are the constraints limiting the beam degrees of freedom. The *Boundary conditions* option is preset to *Default* so that the *Fixed-Free* (corresponds to the "Locked-Free" condition in Fig. 3.11) condition is used. When the option is set to *Custom*, the user can specify the boundary conditions from the drop down menu. Fig. 3.11 shows the different boundary conditions available.

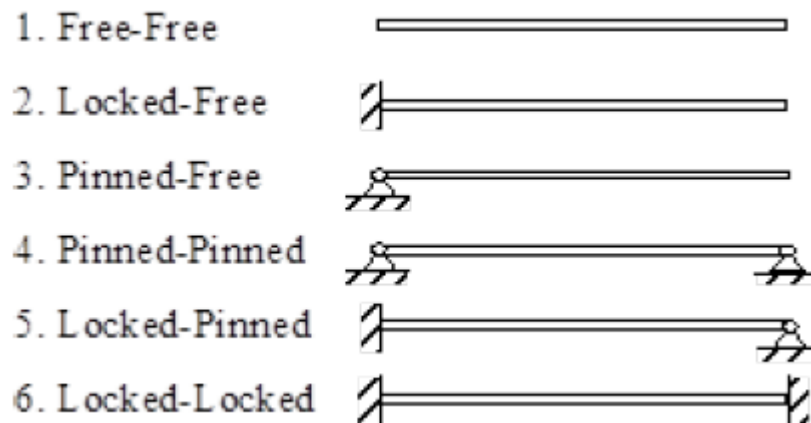


Fig. 3.11: Beam boundary conditions

- **Number of modes** : is the number of flexible modes related to the beam model. A minimum of two modes have to be chosen, which is usually enough to describe the component deformations.

Tip: Since the beam is a rotationally symmetrical structure, all modes in one plane containing the beam center line are identical to the modes in all other planes containing the beam center line. Having this in mind the user shall always specify an even number of modes since each frequency is indeed related to two modes with identical shapes but acting on two perpendicular planes containing the beam center line.

Background

The flexible beam is based on an analytic homogeneous 3D Euler-Bernoulli beam model where shear, axial and torsional deformations are neglected and only bending modes are considered. Fig. 3.12 shows the CoM always in the middle of the beam.

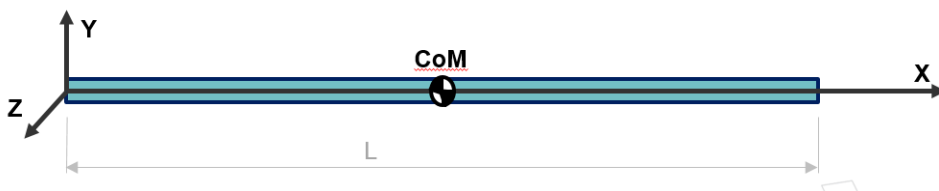


Fig. 3.12: Beam CoM

3.3.1.2 Sloshing

Description

VEHICLE PARTS AND PROPERTIES MODELS

The Sloshing setting is available for all component models with a tank (stage, booster and user component type).

Propellant may move considerably when the fuel tank is only partially filled. The lateral fuel movement in the tank is known as the propellant sloshing. There is a strong coupling between the dynamic force resulting from sloshing, the vehicle mechanical structure and the vehicle control system.

The Sloshing setting is preset to *Disabled*. When the setting is set to *Enabled*, the propellant sloshing dynamic effect is taken into account during the simulation.

Note: The Sloshing setting is available only if the *Multibody Dynamics* feature is *Enabled* (see Scenario General Settings in the User Manual book). The computation time of a model when sloshing is considered is slightly increased.

Configuration

The *Sloshing* feature is disabled by default. If enabled, the following options are displayed:

- *Tank shape*: two different sloshing models are available depending on the tank shape. The user can select *Cylindrical* or *Spherical Tank shape* from the drop down menu.

Note: The spherical tank model only supports one sloshing mass.

- *Number of sloshing Mass*: is the number of moving masses. Each sloshing mass introduces two frequencies along the two directions perpendicular to the tank axis.
- *Kinematic Viscosity*: is a property of the liquid propellant. Table 3.7 reports kinematic viscosity values for common liquid fuels.

Table 3.7: Kinematic viscosity values of typical liquid propellant

Propellant	Acronym	Kinematic Viscosity [m ² /s]
Hydrazine	NH4	0.962e-6
Hydrogen-peroxide	HTP	0.937e-6
Nitrogen Tetroxide (Dinitrogen tetroxide)	NTO	0.292e-6
Liquid Oxygen	LOX	0.167e-6
Liquid Hydrogen	LH2	0.325e-6
Rocket Propellant-11	RP-1	0.970e-6
Unsymmetrical DiMethyl Hydrazine	UDMH	2.171e-6
Mono-Methyl Hydrazine	MMH	0.978e-6
Aerozine 503	A50	0.911e-6
Liquid Methane	LCH4	0.260e-6

VEHICLE PARTS AND PROPERTIES MODELS

- *Minimum Stiffness*: is the minimum stiffness value allowed by the user, which the sloshing model shall handle for a specific tank. If the stiffness computed by the sloshing model goes below the Minimum Stiffness value, the sloshing model will be deactivated and no relative movement will be computed.
- *Minimum Mass*: is the minimum mass value allowed by the user, which the sloshing model shall handle for a specific tank. If the mass computed by the sloshing model goes below the Minimum Mass value, the sloshing model will be deactivated and no relative movement will be computed.
- *Output*: if enabled, all the sloshing parameters and the sloshing relative displacements are saved as auxiliary functions.

Background

The sloshing dynamics can be reproduced by an equivalent spring mass mechanical model [154][155].

The model consists of one static mass rigidly attached to the tank and one or more sloshing masses moving in a plane perpendicular to the tank longitudinal axis.

Each sloshing mass has:

- a mass m_i
- a height h_i , measured from the centre of mass of the static liquid ($h/2$ from the bottom of the tank)
- a spring stiffness k_i
- a constant damping c_i
- a moment of inertia I_i

Cylindrical tank sloshing parameters

The total liquid mass m_{liq} in a cylindrical tank is defined as:

$$m_{liq} = \frac{\pi}{4} \rho d^2 h \quad (3.62)$$

where ρ is the material density of the propellant, d is the tank diameter and h is the propellant height.

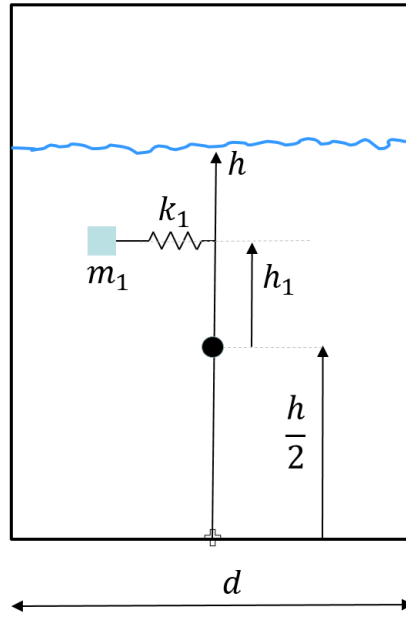


Fig. 3.13: Sloshing model for cylindrical tanks

The moving sloshing masses are computed using the following equation:

$$m_i = m_{liq} \left[\frac{d \tanh(2\varepsilon_i h / d)}{\varepsilon_i (\varepsilon_i^2 - 1) h} \right] \quad (3.63)$$

Where the constant parameters ε_i are defined as:

$$\varepsilon_1 = 1.841, \varepsilon_2 = 5.329, \varepsilon_3 = 8.531, \varepsilon_i \approx \varepsilon_{i-1} + \pi; \quad (3.64)$$

The sloshing mass distances with respect to the liquid centre of mass are defined by the following equation:

$$h_i = \frac{h}{2} - \frac{d}{\varepsilon_i} \tanh\left(\varepsilon_i \frac{h}{d}\right) \quad (3.65)$$

The moment of inertia of the sloshing mass is defined as:

$$I_i = m_{liq} d^2 \left[2 \sum_i \left(\frac{1 - \frac{d}{h} \cdot \frac{\tanh(\varepsilon_i \frac{h}{d})}{\varepsilon_i}}{\varepsilon_i^2 (\varepsilon_i^2 - 1)} \right) \right] \quad (3.66)$$

The spring stiffness values are computed using the following equation:

$$k_i = \frac{m_i g}{d} 2\varepsilon_i \tanh\left(\frac{2\varepsilon_i h}{d}\right) \quad (3.67)$$

where g is the gravity acceleration or an equivalent thrust-induced acceleration acting along the main axis of the tank. The damping values are obtained as follows:

$$c_i = \gamma_i 2m_i \omega_i \quad (3.68)$$

The constant γ_i is the damping ratio for each sloshing mode and it is shown in Fig. 3.14. The Reynolds number Re for environment with gravity or an equivalent thrust-induced acceleration, can be computed as:

$$\Re = \frac{v}{\sqrt{g(\frac{d}{2})^3}} \quad (3.69)$$

where v is the kinematic viscosity.

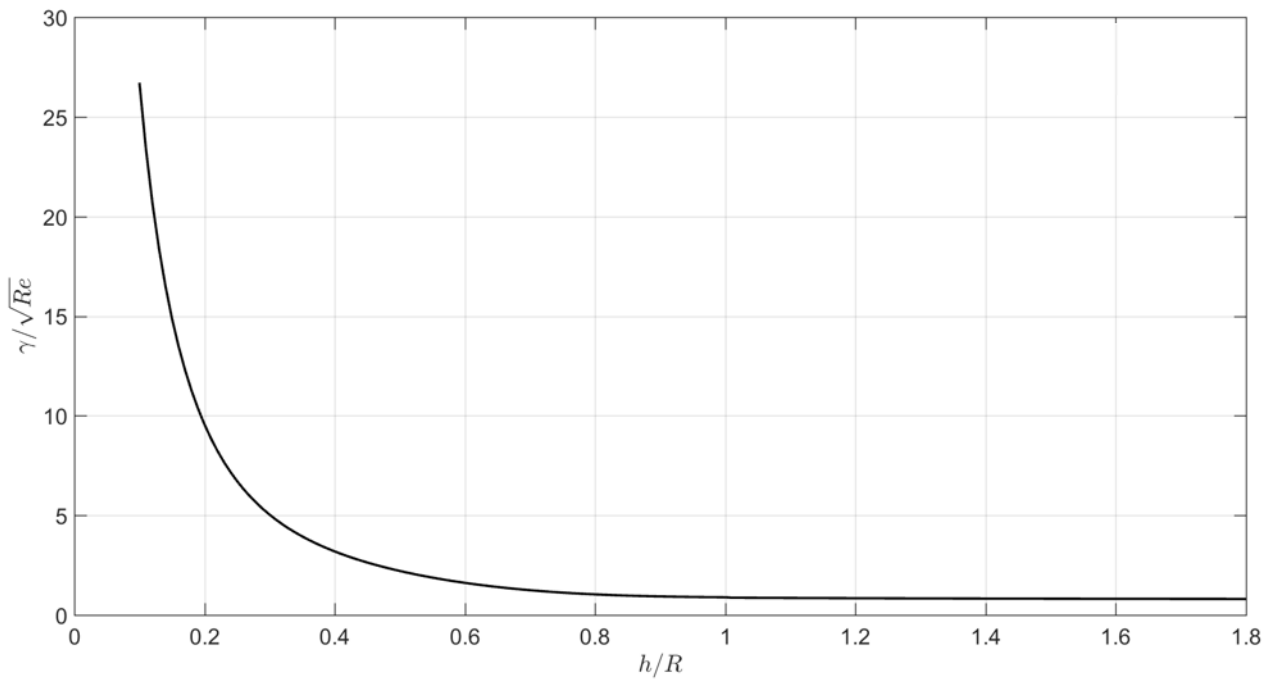


Fig. 3.14: Viscous damping for a cylindrical tank with a spherical dome at the bottom

Finally the natural frequencies of the sloshing model ω_i are defined as:

$$\omega_i = \sqrt{\frac{k_i}{m_i}} \quad (3.70)$$

Spherical tank sloshing parameters

The total liquid mass m_{liq} in a spherical tank with a diameter d and filled up to the height h is defined as:

$$m_{liq} = \pi h^2 \left(\frac{d}{2} - \frac{h}{3} \right) \rho \quad (3.71)$$

where ρ is the material density of the propellant.

VEHICLE PARTS AND PROPERTIES MODELS

The sloshing model for a spherical tank results in a model with zero moment of inertia. Furthermore only information for the first natural frequency is available.

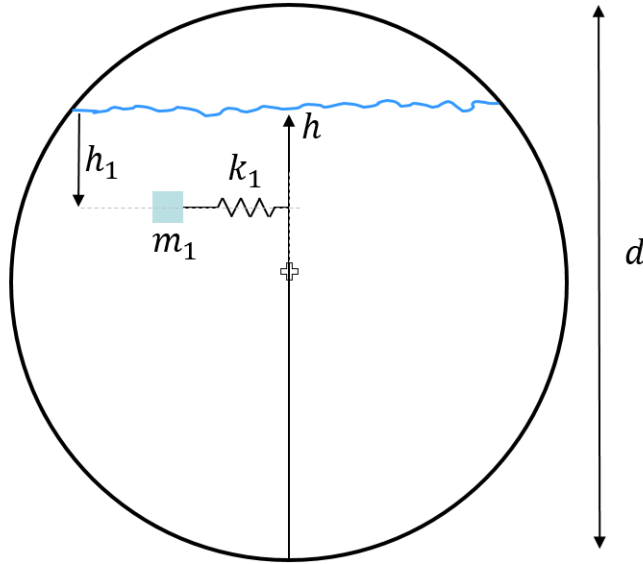


Fig. 3.15: Sloshing model for spherical tanks

The formula for the sloshing mass of the first mode is defined as follows:

$$m_1 = m_{liq} \left[-0.0833 \left(\frac{h}{R} \right)^2 - 0.3441 \left(\frac{h}{R} \right) + 0.9953 \right] \quad (3.72)$$

Fig. 3.16 shows the height of the sloshing mass with respect to the free liquid surface in a spherical tank.

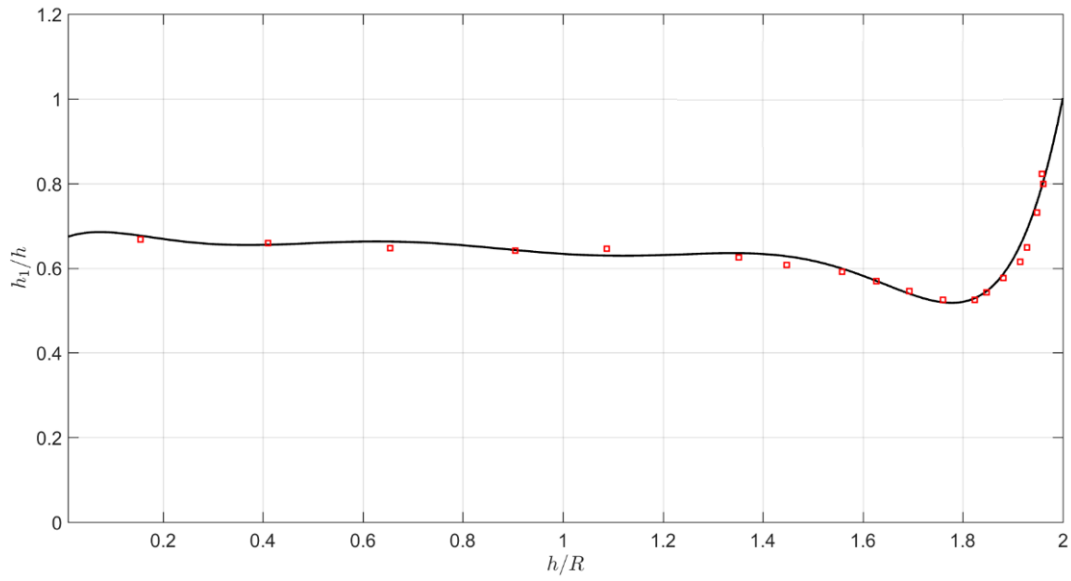


Fig. 3.16: Height of the sloshing mass of the first natural frequency with respect to the free liquid surface in the spherical tank

The spring stiffness value is obtained using the following formula:

$$k_1 = m_1 \frac{g}{R} \lambda_1^2 \quad (3.73)$$

where g is the gravity acceleration or an equivalent thrust-induced acceleration acting along the main axis of the tank and λ is the frequency parameter shown in Fig. 3.17.

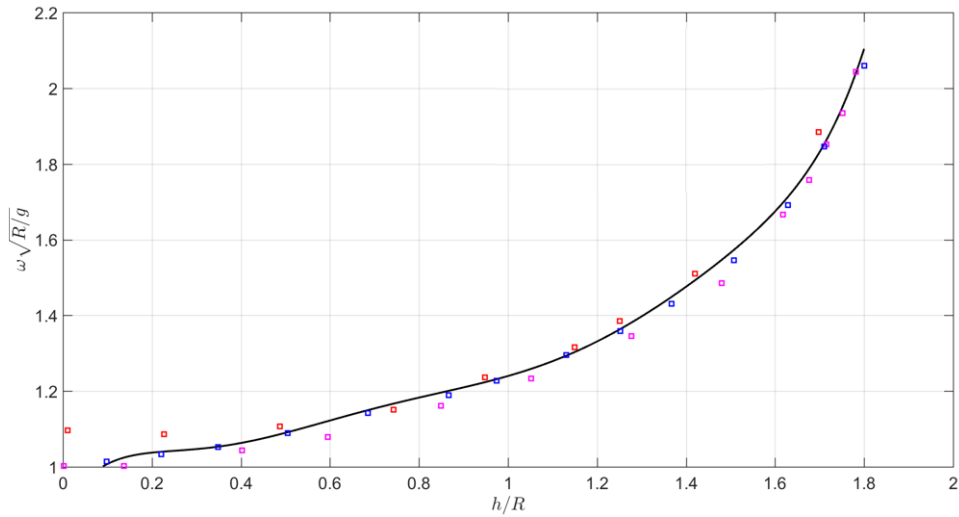


Fig. 3.17: Sloshing frequency parameter for a spherical tank

The natural frequency of the sloshing model is defined as:

$$\omega_1 = \sqrt{\frac{k_1}{m_1}} \quad (3.74)$$

VEHICLE PARTS AND PROPERTIES MODELS

The damping value is obtained with the following equation:

$$c_1 = \gamma 2m_1 \omega_1 \quad (3.75)$$

The damping ratio γ for spherical tanks is shown in Fig. 3.18. The Reynolds number Re for environment with gravity or an equivalent thrust-induced acceleration, can be computed as:

$$\Re = \frac{v}{\sqrt{g\left(\frac{d}{2}\right)^3}} \quad (3.76)$$

where v is the kinematic viscosity.

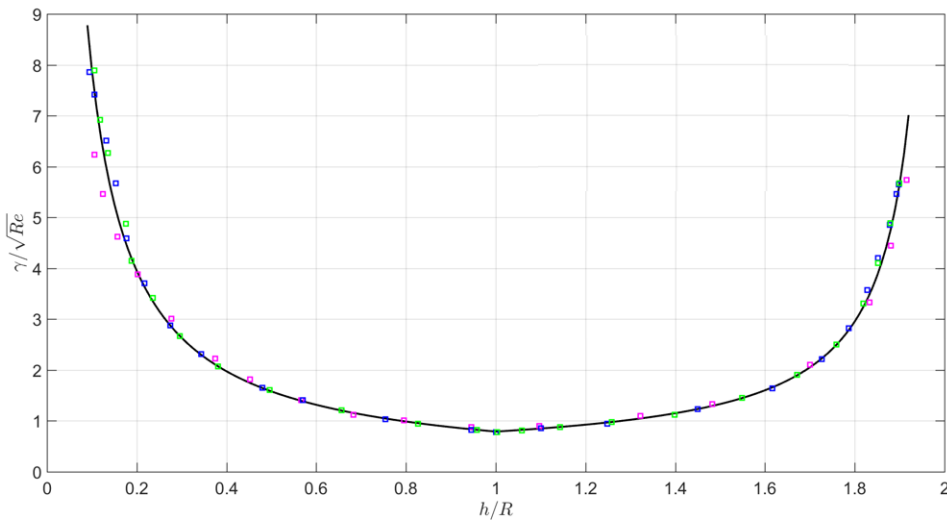


Fig. 3.18: Viscous damping for a spherical tank

3.3.1.3 Mass Computation

ODIN

The detailed vehicle design components (Section 3.3.2.6 Detailed vehicle design stage, Section 3.3.2.7 Detailed vehicle design booster and Section 3.3.3.4 Detailed design payload fairing) are built up of ODIN mass estimation relationship (ODINMER) components. ASTOS uses the following type of ODINMER components:

- Cylinder
- Bulkhead
- Cone
- Struts

Note: The kind of ODINMER component used does not depend on user selection. Each detailed design component has a predefined combination of ODINMER components.

VEHICLE PARTS AND PROPERTIES MODELS

The user has to specify a *stiffening* method for predefined component parts and to define an associated *material* for each part via the respective drop-down lists. Available options for the *stiffening* are:

- *Isotropic*
- *Orthogrid*
- *Isotropic Metallic Hexagonal Sandwich*

The material drop-down menu is empty at first. After typing the material name once, e.g. *AL2195* this material is added to all other material drop-down menus. The mass of each component part is interpolated from the ODINMER tables as function of the dimensions, material, stiffening and load cases.

Tip: If one of the independent variables for interpolation is outside the table ranges, then a warning is printed in the execution log; this warning is printed only once per action, so it could refer to the previous generated loads.

The ODINMER tables are stored in files: each file represents one combination of a material, a shape and a stiffening concept. These files are located in the `lib\ODIN` sub-folder of the ASTOS installation path. As an alternative these files can be located in the `XXX.gtp\model\ODIN` sub-folder of the working scenario. The filename follows the following scheme:

- Material name
- double underscore '__'
- Type of Component
- double underscore '__'
- Stiffening concept (replace blanks with a single underscore)
- .txt

An example would be *CFK__Cone__orthogrid.txt*. If this file is present in `XXX.gtp\model\ODIN` this file is loaded. If it is not present the `lib\ODIN` sub-folder of the ASTOS installation path is searched for the file. If it is not found an error giving the missing filename is displayed in the *Execution Log*.

3.3.2 Vehicle Stage

Description

These models are used to configure structural and propellant masses of a vehicle stage that is connected to one or several engines. Nearly all engine models require a vehicle stage to account for the structural and propellant mass budget. Models exist for both mono propellant stages and stages where fuel and oxidizer flow is accounted for separately (dual propellant). Dual-tank stages apply to engines with a variable fuel-to-oxidizer ratio and are only required if this parameter needs to be varied.

VEHICLE PARTS AND PROPERTIES MODELS

Depending on the selected stage model different additional design parameters are available. One of these additional model feature allows to vary the total propellant mass using a filling ratio parameter fr . In case of design stages a sizing factor (z) is present as parameter.

Table 3.8: Comparison of vehicle stage models

Model	No. of Tanks/ Mass Flows	No. of Tank Filling/ Sizing Factors	Additional Design Features
Basic Vehicle Stage	1	1	-
Basic Vehicle Design Stage	1	1	Structural mass as logarithmic function of propellant mass using correlation factors
Linear Vehicle Design Stage	1	2	Structural mass as linear function of propellant mass
Dual Propellant Stage	2	1	-
Dual Propellant Design Stage	2	1 or 2	Structural mass as linear function of propellant mass
Detailed Vehicle Design Stage	1 or 2	1 or 2	Detailed stage design and individual propellant and structural mass calculation capabilities
Detailed Vehicle Design Booster	1 or 2	1 or 2	Detailed booster design and individual propellant and structural mass calculation capabilities

Optimization Constraints

For each defined stage tank a mass state is internally defined in all phases where this tank is active (i.e. an active engine is associated with it). The following constraints are related to all vehicle stage models, i.e. a vehicle component with a tank:

- *Initial_PROP_MASS*
- *Initial_Propellant_Volume*
- *Residual_PROP_MASS*

For more details about these constraints see Section 6.1.

States

For each defined tank a mass state is defined in all phases where this tank is active, i.e. an active engine is associated with it.

Cost Functions

At least one vehicle stage component is required, if the cost function is set to *Min. Initial Propellant* (see Section 7.9). In this case, the optimization program is allowed to reduce the initial propellant mass of each stage component by multiplying the specified propellant mass with a constant filling ratio parameter. If the tank is modeled as a *Basic Vehicle Design Stage* component, the optimization varies the associated tank sizing factor which affects both structural and propellant mass simultaneously.

Real Parameters

For each active vehicle stage component, the corresponding filling ratio parameter or tank sizing factor, respectively, is added automatically to the real design parameter array of the dynamic system. The label of each of these parameters contains the ID string of the stage to which it corresponds. The default box constraints on the filling ratio allow a reduction of the initial propellant mass of up to 30%.

For each design stage, the default range limit of its tank sizing factor is $\pm 15\%$.

Moments Of Inertia

Many vehicle stage models do not provide a detailed geometry and only consider masses. These models all use a common model for the moments of inertia which is explained in this paragraph.

- The moments of inertia of the structural mass is computed as hollow cylinder with a wall thickness of 5% of the stage radius.
- The propellant is assumed to form a solid cylinder filling the entire stage, i.e. a cylinder radius of 95% of the stage radius.
- The center of mass is located at the center node of this stage, i.e. the geometric center of its bounding box.

Tip: Any vehicle stage offers the possibility to use user input for the center of mass position and inertia tensor. This setting is called *Mass distribution* and is explained in detail in Section 3.1.2.

3.3.2.1 Basic Vehicle Stage

Description

This chapter describes a simple vehicle stage model with a user-definable structural mass and a propellant mass which can be made optimizable. It can be used as rocket stage but can also be used as propellant tank in satellites etc.

Configuration

The configuration panel provides various settings to customize a basic vehicle stage component:

- *Propellant type*: defines the state of aggregation of the propellant. Possible options are *Solid* and *Liquid*.
- *Filling ratio*: represents a positive dimensionless scaling factor. It is multiplied with the propellant mass and allows an optimization of the propellant mass in case this parameter is set as *optimizable*. In any case, it can be used to define a filling ratio not equal to 1. It can be defined as *Common* or *Individual* (see Section 3.1.1).
- *Dimensions*: defines the stage geometry. When set to *custom*, *X* defines the height of a cylindrical body, *Y* corresponds to its diameter. When set to *default* the dimensions are calculated according to the same power law explained in the background paragraph of Section 3.3.3.1. These dimensions are used to define the inner diameter and the inner height of the tank.
- *Structural mass*: the "dry" mass of the stage
- *Propellant mass*: the maximum propellant mass the stage holds
- *Residual Propellant Mass* can be used to set the amount of propellant which has to be inside the tank by the time it is jettisoned. The default value is 0.0 kg. If selected the available options are:
 - *Defined by* selects the kind of limit to be used. If *Range* is selected a *Lower limit* and an *Upper limit* have to be specified, otherwise only one value has to be provided.
 - Depending on the kind of limit selected above either a *Reference* value, a *Lower limit*, an *Upper limit* or a combination of those has to be inserted as residual propellant mass
- The generic settings *Mass distribution*, *Electrical power* and *Thermal characteristics* are explained in Section 3.1.2, Section 3.1.3 and Section 3.1.4 respectively. The latter two settings are available if the corresponding features in the ASTOS software have been installed and then enabled in the scenario *Active features*.

Model Background

The *filling ratio* parameter *fr* is defined by

$$m_p = m_{p,0} \cdot fr \quad 0 < fr \quad (3.77)$$

where $m_{p,0}$ is the nominal *propellant mass* specified by the user.

Moments Of Inertia

The moments of inertia of the structure are computed as follows:

- The radius of the tank is computed as half of *Y* dimension. The length of the tank is equal to the *X* dimension.
- If there is a cylindrical part, i.e. $I_{cyl} = X - Y > 0$ (two hemispheres with radius $0.5 * Y$) then the following equations are used.

VEHICLE PARTS AND PROPERTIES MODELS

$$l_{cyl} = l_{tank} - 2 \cdot r \quad (3.78)$$

$$m_{sphere} = \frac{200 \cdot M \cdot r}{l_{cyl} + 200r} \quad (3.79)$$

$$m_{cyl} = M - m_{sphere} \quad (3.80)$$

with

- l_{cyl} as length of the cylindrical part of the tank
- l_{tank} being the X dimension of the stage
- r the radius of the stage, i.e. half of the Y dimension
- M the user defined structural mass
- m_{sphere} the mass of the spherical bulkheads
- m_{cyl} the mass of the cylindrical part.

Based on these masses and lengths a hollow cylinder with two hollow hemispheres at both ends is used to compute the moments of inertia. The wall thickness of the cylindrical part is assumed to be 1% of r and the wall thickness of the bulkheads is set to 5% of r .

- If it is a flat tank, i.e. $X < Y$ dimension a hollow cylinder with a wall thickness of 5% of r is assumed.

The moments of inertia of the propellant are calculated as hollow cylinder where the inner radius is computed based on the fill level of the tank with a filling of 1.0 being a solid cylinder and as propellant becomes less the inner radius gets larger.

Optimization Constraints

This model only creates the constraints mentioned in Section 3.3.2.

3.3.2.2 Basic Vehicle Design Stage

Description

This model is used to define the structural and propellant masses of a vehicle stage whose tank size can be adjusted by a sizing factor. It should be used for simultaneous trajectory and vehicle stage size design in the neighbourhood of a nominal vehicle design. The tank size is varied by scaling the maximum propellant mass the stage holds. An increase in propellant mass is penalized by an increase in structural mass and stage length assuming a constant stage diameter [21]. A nominal stage design and a relationship between the stage total propellant mass and the structural mass must be supplied.

Configuration

The configuration panel provides various settings to customize a basic vehicle design stage component:

VEHICLE PARTS AND PROPERTIES MODELS

- *Propellant type*: defines the state of aggregation of the propellant. Possible options are *Solid* and *Liquid*.
- *Sizing factor* : represents a positive scaling factor for the tank size. This factor can be set as *optimizable* to obtain an optimized propellant mass. It is unity for nominal conditions and user-defined values should not deviate significantly from a range between 0.85 and 1.15.
- *Structure properties of the stage*:
 - *Nominal mass* is the nominal "dry" mass of the stage
 - *Nominal length* is the nominal length (height) of the cylindrical stage
 - *Nominal diameter* is the constant nominal diameter of the cylindrical stage
- *Propellant properties*:
 - *Nominal mass* of the propellant the stage holds
 - *Mean density* of the propellant which is used to determine the nominal propellant volume
- *Residual Propellant Mass* can be used to set the amount of propellant which has to be inside the tank by the time it is jettisoned. The default value is 0.0 kg. If selected the available options are:
 - *Defined by* selects the kind of limit to be used. If *Range* is selected a *Lower limit* and an *Upper limit* have to be specified, otherwise only one value has to be provided.
 - Depending on the kind of limit selected above either a *Reference* value, a *Lower limit*, an *Upper limit* or a combination of those has to be inserted as residual propellant mass
- *Mass correlation coefficient* represents positive dimensionless coefficient *b* as defined in the mass correlation equation .
- The generic settings *Mass distribution*, *Electrical power* and *Thermal characteristics* are explained in Section 3.1.2, Section 3.1.3 and Section 3.1.4 respectively. The latter two settings are available if the corresponding features in the ASTOS software have been installed and then enabled in the scenario *Active features*.

Model Background

The propellant mass of the stage is determined by the tank *sizing factor* *z*

$$m_p = m_{p,0} \cdot fr \quad 0 < fr \quad (3.81)$$

where $m_{p,0}$ is the *nominal mass* of propellant specified by the user. The sizing factor is unity at nominal conditions. The corresponding change in tank volume is given by

$$\Delta V = \frac{m_{p,0}(z-1)}{\rho_p} \quad (3.82)$$

where ρ_p is the propellant *mean density* specified by the user. The change in stage length and the stage diameter-to-length ratio *r* are approximated as

$$\Delta L = \frac{4\Delta V}{\pi d^2} \quad ; \quad r = \frac{d}{L+\Delta L} \quad (3.83)$$

VEHICLE PARTS AND PROPERTIES MODELS

where d is the constant *nominal diameter* and L the *nominal length* of the stage as specified by the user.

In this model, an analytical relationship between the structural mass m_S and the propellant mass m_P is postulated of the form

$$m_S(m_P) = a \cdot m_P^b ; \quad a = \frac{m_{S,0}}{m_{P,0}^b} \quad (3.84)$$

The user specifies the design data for the *nominal mass* of the structure and propellant ($m_{S,0}$ and $m_{P,0}$ respectively) as well as the *mass correlation coefficient* b . The correlation coefficient applies to masses given in SI units (kg). Typical numeric values are listed in Table 3.9 .

Table 3.9: Typical correlation coefficients for the Basic Design Stage model

Stage Type	Correlation Coefficients	
	a	b
LOX / LH ₂ (1st stage)	0.2498	0.9171
LOX / LH ₂ (2nd & 3rd stage)	1.4515	0.7622
Storable prop. (1st stage)	0.03517	1.0404
Storable prop. (2nd & 3rd stage)	1.7864	0.7223
Liquid propellant booster	0.1434	0.9786
Solid propellant booster	0.2814	0.9542

Moments Of Inertia

The assumptions used for the moments of inertia and center of mass computation are explained .

Optimization Constraints

This model only creates the constraints mentioned in Section 3.3.2.

3.3.2.3 Linear Vehicle Design Stage

Description

This model is used to configure the structural and propellant masses of a vehicle stage whose tank size can be adjusted by a sizing factor. The structural mass of the tank can be specified by a fixed ratio to the total propellant mass which is optimizable.

Configuration

The configuration panel provides various settings to customize a linear vehicle design stage:

VEHICLE PARTS AND PROPERTIES MODELS

- *Propellant type*: defines the state of aggregation of the propellant. Possible options are *Solid* and *Liquid*.
- *Sizing factor* represents a positive dimensionless scaling factor for the tank size. It can be set as *optimizable* to obtain an optimized propellant mass.
- *Filling ratio* represents a positive dimensionless scaling factor. It is multiplied by the total propellant mass and allows an optimization of the propellant mass in case this parameter is set as *optimizable*. In any case it can be used to define a filling ratio not equal to 1. It can be defined as *Common* or *Individual* (see Section 3.1.1).
- *Residual Propellant Mass* can be used to set the amount of propellant which has to be inside the tank by the time it is jettisoned. The default value is 0.0 kg. If selected the available options are:
 - *Defined by* selects the kind of limit to be used. If *Range* is selected a *Lower limit* and an *Upper limit* have to be specified, otherwise only one value has to be provided.
 - Depending on the kind of limit selected above either a *Reference* value, a *Lower limit*, an *Upper limit* or a combination of those has to be inserted as residual propellant mass
- *Structure to propellant mass ratio* is a dimensionless value for the ratio of structural mass to the total propellant mass (not necessary equivalent to the nominal propellant mass!)
- *Nominal propellant mass* of the stage
- The generic settings *Mass distribution*, *Electrical power* and *Thermal characteristics* are explained in Section 3.1.2, Section 3.1.3 and Section 3.1.4 respectively. The latter two settings are available if the corresponding features in the ASTOS software have been installed and then enabled in the scenario *Active features*.

Model Background

The maximum propellant mass of the stage m_P is determined by the optimizable tank *sizing factor* z

$$m_P = m_{P,0} \cdot z \quad 0 < z \quad (3.85)$$

while the effective loaded propellant mass m_{Pe} is determined by the optimizable *filling ratio* fr

$$m_{Pe} = m_P \cdot fr \quad 0 < fr \quad (3.86)$$

where $m_{P,0}$ is the *nominal propellant mass* specified by the user. The relationship between the structural mass m_S and the total propellant mass m_P is postulated of the form:

$$m_S(m_P) = r \cdot m_P \quad ; \quad r = \frac{m_S}{m_{P,0}} \quad (3.87)$$

The user has to supply the *structure to propellant mass ratio* r related to the total propellant mass m_P .

Moments Of Inertia

The assumptions used for the moments of inertia and center of mass computation are explained in .

Optimization Constraints

This model only creates the constraints mentioned in Section 3.3.2.

3.3.2.4 Dual Propellant Stage

Description

This model offers a user-definable structural mass and separate masses for fuel and oxidizer which can be made optimizable by means of the filling ratio parameter. The oxidizer tank is located in the lower half of the stage and the fuel tank in the upper half.

Note: Any rocket engine connected to a dual Propellant stage has to be able to handle two mass flows, e.g. the subtype *Typical VMR*.

Configuration

The configuration panel provides various settings to customize a dual propellant stage:

- *Propellant type*: defines the state of aggregation of the propellant. Possible options are *Solid* and *Liquid*.
- *Filling ratio* represents a positive dimensionless scaling factor. It is multiplied by the total propellant mass and allows an optimization of the propellant mass in case this parameter is set as *optimizable*. In any case it can be used to define a filling ratio not equal to 1. It can be defined as *Common* or *Individual* (see Section 3.1.1).
- *Dimensions* define the stage geometry. When set to *custom*, *X* defines the height of a cylindrical body and *Y* corresponds to its diameter. When set to *default* *X*, *Y* and *Z* are all set to 1.0 meter.
- *Propellant mass* offers a separate definition of the maximum *Oxidizer* and *Fuel* masses
- *Residual propellant mass* is the propellant mass at the end of the last phase where this stage is active. The default value is 0.0. This value is used by the *Residual_PROP_MASS* constraint which is automatically created for all stages.
- *Structural mass* is the "dry" mass of the stage
- The generic settings *Mass distribution*, *Electrical power* and *Thermal characteristics* are explained in Section 3.1.2, Section 3.1.3 and Section 3.1.4 respectively. The latter two settings are available if the corresponding features in the *ASTOS* software have been installed and then enabled in the scenario *Active features*.

Model Background

For a dual-tank stage, total oxidizer and fuel masses are given by

$$\begin{aligned} m_{ox} &= m_{ox,0} \cdot fr \\ m_{fuel} &= m_{fuel,0} \cdot fr \end{aligned} \quad 0 < fr \quad (3.88)$$

VEHICLE PARTS AND PROPERTIES MODELS

where the maximum *oxidizer* and *fuel* masses $m_{ox,0}$ and $m_{fuel,0}$ are specified by the user. fr represents the *filling ratio* parameter.

Moments Of Inertia

Moments of inertia for the structure and propellant masses are computed separately. The computation for the structure is the same as for the *Basic Vehicle Stage*. Refer to the moments of inertia paragraph in Section 3.3.2.1 for details. For the moments of inertia of the fuel and of the oxidizer the following assumptions are used.

- If the length of the tank (half of X dimension) is smaller than the diameter a flat tank is approximated as solid cylinder with a radius of 95% of the stage diameter and a length equal to the tank length multiplied with the filling ratio.
- If the length of the tank is greater than the diameter the fill level of the tank determines the moments of inertia as follows:
 - If only the lower bulkhead is filled with propellant then the propellant forms a solid hemisphere.
 - If the lower bulkhead and the cylindrical part are filled a solid cylinder is added to the hemisphere with a length equal to the length of the cylindrical part multiplied with the filling ratio of the cylindrical part.
 - If lower bulkhead, cylindrical part and upper bulkhead are filled then a solid hemisphere is added to the lower parts.

Note: Liquid propellant doesn't necessarily rotate with the tank. Whether moments of inertia of liquid masses are considered is explained in detail in Section 3.1.2.

Optimization Constraints

This model only creates the constraints mentioned in Section 3.3.2.

3.3.2.5 Dual Propellant Design Stage

Description

This model is used to configure the structural and propellant masses of a dual tank vehicle stage whose tank mass is a linear function of the propellant mass. This model is suitable for simultaneous trajectory and vehicle design during an early design phase. The structural mass of the tank can be specified by a fixed ratio to the total propellant mass. Several properties are optimizable.

Configuration

The configuration panel provides various settings to customize a dual propellant design stage:

VEHICLE PARTS AND PROPERTIES MODELS

- *Propellant type*: defines the state of aggregation of the propellant. Possible options are *Solid* and *Liquid*.
- *Ratio type* of the structure to propellant mass ratio with two options are available:
 - *Single Ratio*
 - *Split Ratio*

While the first option relates the structure mass to the total propellant, the second option allows an individual ratio for oxidizer and fuel.

- *Structure to propellant mass ratio* defines a dimensionless ratio which is only present in case *Single Ratio* is selected)
- *Structure to oxidizer/fuel mass ratio* are two dimensionless ratios which are only present in case *Split Ratio* is selected.
- *Mixture ratio* is the ratio of oxidizer and fuel mass (the default value corresponds to a typical LOX/LH2 ratio). Optionally this parameter can be set as *optimizable*.
- *Filling ratio* represents a positive dimensionless scaling factor. It is multiplied by the total propellant mass and allows an optimization of the propellant mass in case this parameter is set as *optimizable*. In any case it can be used to define a filling ratio not equal to 1. It can be defined as *Common* or *Individual* (see Section 3.1.1).
- *Volume computation* affects how the volume computation is realized. Either *default* or *custom* settings can be used. Detailed settings are provided .
- *Propellant sizing factor*: a linear scaling factor on the nominal total propellant mass which affects also the structural mass. Optionally this parameter can be set as *optimizable*.
- *Nominal propellant mass* is the total propellant mass the stage holds
- *Residual propellant mass* is the propellant mass at the end of the last phase where this stage is active. The default value is 0.0. This value is used by the *Residual_PROP_MASS* constraint which is automatically created for all stages.
- The generic settings *Mass distribution*, *Electrical power* and *Thermal characteristics* are explained in Section 3.1.2, Section 3.1.3 and Section 3.1.4 respectively. The latter two settings are available if the corresponding features in the ASTOS software have been installed and then enabled in the scenario *Active features*.

Volume computation settings

The default volume computation is that the volume is not fixed but the mass is. This is why a *Initial_Propellant* constraint is automatically created. If the *Volume computation* is set to *custom* the following options become available:

- *Total volume fixed* offers the possibility to automatically create an *Initial_Propellant_Volume* instead of *Initial_Propellant* constraint by choosing *enabled*.
- *Density oxidizer* is the density of the oxidizer used in the *Initial_Propellant_Volume* constraint. The default values displayed is the density of liquid oxygen.
- *Density fuel* is the density of the fuel used in the *Initial_Propellant_Volume* constraint. The default values displayed is the density of liquid hydrogen.

Note: For further information about the mentioned constraints please refer to Section 6.1.

Model Background

The propellant mass of the stage is determined by the *propellant sizing factor* z

$$m_p = m_{p,0} \cdot z = (m_{LOX} + m_{LH_2}) \cdot z \quad (3.89)$$

where $m_{p,0}$ is the *nominal propellant mass* specified by the user. The mass of the tanks (e.g. LOX and LH₂) are computed by the *mixture ratio* f :

$$f = \frac{m_{LOX}}{m_{LH_2}} \quad (3.90)$$

$$m_{LOX} = \frac{m}{1+f} \quad ; \quad m_{LH_2} = \frac{m}{1+f} \quad (3.91)$$

The fuel mixture ratio and the mixture ratio specified for the propulsion system do not have to be equal. An optimizable mixture ratio between both initial propellant masses together with an unconstrained burn time ensures that both tanks will be empty at burn out, even if the mixture ratio of the engine is not constant.

Note: The type of propellant is not fixed for each tank. Only the index is important. ASTOS propulsion systems with dual propellant mass flow treat the oxidizer as first mass flow (which is connected to the first tank). In case of a user-defined propulsion system, this order can be chosen freely.

The relationship between the structural mass m_s and the total propellant mass m_p is postulated of the form

$$m_s = r \cdot m_p \quad (3.92)$$

or considering tank depended mass ratios

$$m_s = r_{LOX} m_{LOX} + r_{LH_2} m_{LH_2} \quad (3.93)$$

where r is the user-defined *structure to propellant mass ratio* and r_{LOX} and r_{LH_2} are the individual ratios in case *Split Ratio* is selected.

This model supports the *volume computation* of the variable tank size and/or to fix the initial total volume. In this way a variable tank bottom can be optimized. As additional input parameters the density of both propellant components have to be provided.

Optimization Constraints

The following constraints are typically related to this model:

- Constraints in common with other vehicle stage models can be found in the Optimization Constraints paragraph of Section 3.3.2.
- If the initial tank volume is fixed an *Initial Tank Volume* constraint is automatically added.

3.3.2.6 Detailed Vehicle Design Stage

Description

This model is used to configure both structural and propellant mass and to design the shape of the stage (e.g. the length and diameter of each tank). For accurate computation the basic architecture of the stage can be defined in detail. The user can choose between a mono (solid or liquid) or dual (liquid) propellant stage model. These subtypes can be found in Section 3.3.2.6.1 Mono and Section 3.3.2.6.2 Dual sub-chapters.

Optimization Constraints

Some optimization constraints are typically related to the design stage definition. As they apply to both mono and dual propellant models they are shortly outlined together below:

- The *volumetric loading fraction* u relates the tank volume to the volume of the propellant and it has to be ensured that the utilizable tank volume provides enough space to carry all the propellant, i.e.:

$$V_{\text{tank}}u \geq V_p \quad (3.94)$$

for mono propellant and

$$V_{\text{tank-f}}u \geq V_f \quad (3.95)$$

$$V_{\text{tank-ox}}u \geq V_{ox} \quad (3.96)$$

for dual propellant models respectively.

For the mono propellant model, this can be realized by a *Equipment Design* constraint subtype set as initial boundary or parameter constraint. The *Parameter* should be set to *Initial_Tank_Volume_Ratio* which is defined as the dimensionless ratio of the usable tank volume to the propellant volume. A *lower limit* of 1.0 assures that the propellant volume does not exceed the available tank volume.

For the dual propellant model, two separate *Equipment Design* constraints have to be generated equivalently (i.e. *Parameter* = *Initial_Fuel_Tank_Volume_Ratio* and *Parameter* = *Initial_Ox_Tank_Volume_Ratio*).

- To ensure that the engine diameter does not exceed the diameter of the stage an *Equipment Design* constraint with *Parameter* = *Engine_to_Stage_Diameter_Ratio* is suitable. It computes the dimensionless ratio of the engine nozzle diameter to the stage diameter. Choose *upper limit* as *limit type* with a value of 1.0 to set the engine diameter upper bound to the stage diameter.
- To ensure that the engine length of the stage above does not exceed the length of the interstage an *Equipment Design* constraint with *Parameter* = *Engine_to_Interstage_Length_Ratio* is suitable. It computes the dimensionless ratio of the engine length (from stage above) to the interstage length. The engine length is calculated assuming an expansion ratio of 58.2 (Vulcain 2 engine) and a characteristic length of 0.9m (mean value for all propellant/oxidizer combinations); only the *Throat Design Rocket* models with RPA and *Geometry* enabled is providing a more precise length calculation.

- The proper length of the engine with respect to the interstage is automatically set in the interstage section of the stages, no constraint is required.

Placement in the Vehicle Builder

The stage can be inserted in the vehicle via the Vehicle Builder. In particular the placement is an important aspect during the design of a vehicle. It is good practice in build up a vehicle starting from the bottom. In this case, of the stage placement is performed with the lower stage as *Reference element* with *Top (x)* as *Reference node* and *Bottom (-x)* as *Anchor node*. Without defining any position offset the outer skin of the lower and the upper stage will be connected. The anchor nodes for a *Detailed Vehicle Design Stage* are defined as:

- *Top (x)*: the topmost point of the interstage or, in case the interstage is not present, the topmost point of the forward skirt. In case an upper stage is present, the topmost point is where the bottom node of the upper stage is located. This is required by the fact that the lower and upper stage are overlapping e.g. the interstage covers parts of the thrust frame.
- *Bottom (-x)*: the bottommost point of the thrust frame or, in case the thrust frame is not present, the bottommost point of the aft skirt.

3.3.2.6.1 Mono Propellant

Description

This model subtype allows a detailed definition of mono propellant stage with either liquid or solid propellant. An overview of the geometry parameters used for the configuration is shown below.

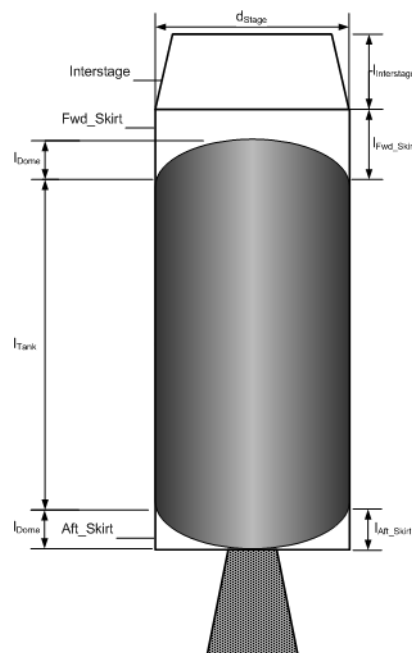


Fig. 3.19: Definition of mono propellant stage geometry parameters

Configuration

The configuration panel provides various settings to customize a detailed mono propellant vehicle stage:

- *Diameter*: define the maximum outer diameter of the stage This parameter can be set as *optimizable*.
- *Propellant type*: defines the state of aggregation of the propellant. Possible options are *Solid* and *Liquid*. The latter requires additional settings for the thrust frame.
- *Propellant sizing factor*: a linear scaling factor on the nominal propellant mass. Optionally this parameter can be set as *optimizable*. It can be defined as *Common* or *Individual* (see Section 3.1.1).
- *Nominal propellant mass* of the stage
- *Residual propellant mass* is the propellant mass at the end of the last phase where this stage is active. The default value is 0.0. This value is used by the *Residual_PROP_MASS* constraint which is automatically created for all stages.
- *Structural mass safety factor* is a dimensionless value ≥ 1.0 multiplied with the computed structural mass of the stage.
- *Interstage*: defines the properties of an interstage between the forward skirt and the next stage above. Its optional configuration is *disabled* by default meaning no interstage is present. The detailed settings for an *enabled* interstage are described .
- *Payload adapter*: defines the properties of a configurable payload adapter. By default, its configuration is *disabled* meaning no payload adapter is used. The detailed settings for an *enabled* payload adapter are described .
- *Forward skirt* defines the properties of the forward skirt of the stage. Detailed settings are described .
- *Tank*: defines the properties of the stage tank. Detailed settings are described .
- *Aft skirt* defines the properties of the aft skirt of the stage. Detailed settings are described .
- *Thrust frame*: defines the properties of the thrust frame that connects the engine with the tank of its related stage. Detailed settings are described . This panel is only present when *liquid* propellant has been selected.
- The generic settings *Mass distribution*, *Electrical power* and *Thermal characteristics* are explained in Section 3.1.2, Section 3.1.3 and Section 3.1.4 respectively. The latter two settings are available if the corresponding features in the ASTOS software have been installed and then enabled in the scenario *Active features*.

Interstage Settings

The interstage lower diameter is equal to the current stage diameter; the upper diameter is equal to the diameter of the stage above. The interstage configuration requires the following user inputs:

- *Minimum Length* is used to define the interstage length (see $l_{Interstage}$ in Fig. 3.19) together with the *Gap between bulkhead and nozzle*.
- *Gap between bulkhead and nozzle* is the distance between the topmost point of the forward bulkhead of the current stage and the end of the nozzle (i.e. lowest point) of the engine in

the stage above. The $I_{Interstage}$ is computed considering this gap together with the length of the forward dome and skirt as well as the length of the aft skirt, thrust frame and nozzle of the stage above. The resulting length is compared with the *Minimum Length* and the maximum is used to define $I_{Interstage}$.

Tip: The common way to define a fixed interstage length is to set the *Gap between bulkhead and nozzle* to a small negative number (e.g. -20 meters), in this way the *Minimum Length* is used.

- *Mass Computation* defines the *stiffening* and *material* of the interstage cone

Payload Adapter Settings

For the payload adapter, only its *mass* has to be specified. This mass (and also the mass of a payload component without specified dimensions) is defined as a point mass located as shown in Fig. 3.20.

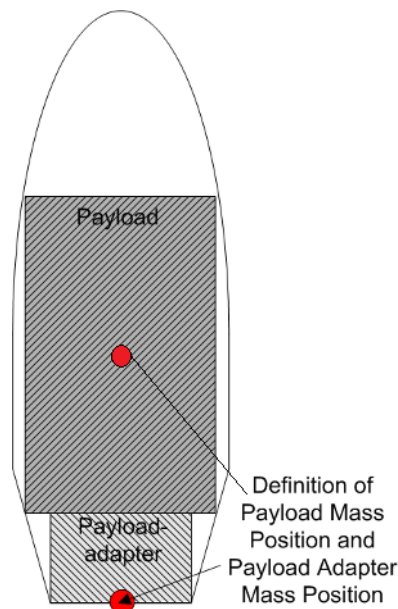


Fig. 3.20: Centre of mass position of the payload adapter (and of a payload mass without specified dimensions)

Tip: If the center of mass position of the payload adapter is of major importance, the adapter can also be defined separately as an auxiliary component with customizable dimensions.

Forward and Aft Skirt Settings

The options for configuring the forward and aft skirt of the stage are equivalent and comprise:

- *Ratio to tank dome length*: the ratio (>1) of the skirt length I_{Fwd_Skirt} (or I_{Aft_Skirt} respectively) to the dome length I_{Dome}
- *Mass Computation* defines the *stiffening* and *material* of the *cylinder* part of the skirts

Tank Settings

VEHICLE PARTS AND PROPERTIES MODELS

The configuration of the propellant tank requires several user inputs:

- *Length without domes* is the *optimizable* length l_{Tank} of the cylindrical part of the propellant tank.
- *Cylinder segment length* is optional and specifies the length of a single segment of the cylindrical part of the tank. If not specified, the upper bound of the tank length without domes is used, and if this is also not specified its value is used. The specified cylinder segment length and the length without domes (upper bound or value) yield the number of cylindrical segments. If *default* the cylindrical tank part will consist of one segment.
- *Volumetric loading fractions* specifies the ratio of the initial volume filled with propellant to the total tank volume
- *Propellant density*: the average density of the propellant. Default value is 1550.0 kg/m³.
- *Thermal protection system* defines properties of a TPS layer inside the tank. The available settings depend on the selected *Type* and are listed below:
 - *Automatic* is only available if the *Propellant type* is solid. This computes the mass of the TPS according to equation (6.16) of [143].
 - *Fixed mass* defines the TPS mass as user input.
 - *Density and thickness* specifies the TPS layer with thickness of the layer and material density. This option computes the TPS mass but also influences the usable tank volume.
- *Tank pressure* is the pressure in the propellant tank (chamber pressure in case of *solid* propellant). The tank pressure can either be *defined by* a constant *Value* or by providing a *Profile* as function of the fill ratio (refer to Profile Data Tables in User Manual for more information about profiles).
- *Domes*: define the shape of the tank domes which is either *Spherical* or *Ellipsoidal*. In the latter case, the user has to specify an *Ellipse ratio*, i.e. the ratio of tank radius to tank dome height.
- *Mass Computation* defines the *stiffening* and *material* of the *forward bulkhead*, *cylindrical part* and *aft bulkhead*

Thrust frame Settings

The options for configuring the thrust frame are only available when *liquid* propellant is selected. In this case they comprise:

- *Length ratio of thrust frame to tank dome* defines the geometry of the thrust frame using the ratio of the dome length l_{dome} to the length of the thrust frame l_{thrust_frame} (see Fig. 3.21). It is used for a proper calculation of the center of gravity.

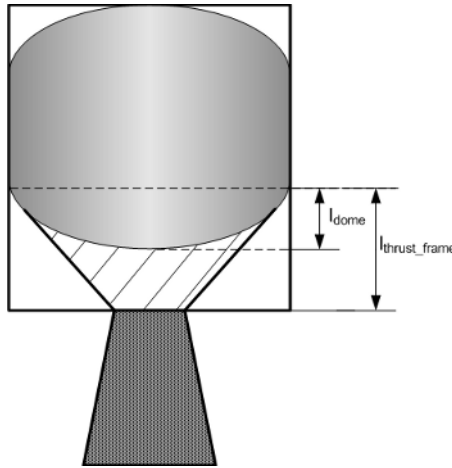


Fig. 3.21: Length definitions for thrust frame

- *Mass Computation* defines the *stiffening* and *material* of the upper and lower *ring stiffener* and the *struts* of the thrust frame.

Model Background

The propellant mass of the stage is determined using the *propellant sizing factor* z :

$$m_p = m_{p0} z \quad (3.97)$$

where m_{p0} is the *nominal propellant mass* specified by the user. The required volume for this mass is a function of the *propellant density* ρ_p and the volume of the propellant tank is a function of the tank *diameter* and tank length:

$$V_p = \frac{m_p}{\rho_p} \quad (3.98)$$

$$V_{\text{tank}} = l d \quad (3.99)$$

Moments of Inertia

The moments of inertia of this model are computed separately for propellant and structure. For the structure a hollow cylinder is assumed using a wall thickness of a steel tank computed as $Wall = 2.5 \cdot 10^3 \cdot d$ with d being the diameter of the stage. The center of mass is assumed to be the center node of this component. The center of mass and moments of inertia of the propellant are computed based on the following assumptions:

- If the propellant is solid:
 - The propellant forms a hollow cylinder with an outer radius of $r_{prop} = 0.5 \cdot d - Wall$.
 - The inner radius of the hollow cylinder is computed based on the filling ratio of the tank with a full tank having an inner radius of 0.0
 - The center of mass is located in the geometric center of the tank (see Fig. 3.28).
- If the propellant is liquid:

VEHICLE PARTS AND PROPERTIES MODELS

- If the length of the tank is smaller than the diameter a flat tank is approximated as solid cylinder with a radius of 95% of the stage diameter and a length equal to the tank length multiplied with the filling ratio.
- If the length of the tank is greater than the diameter the fill level of the tank determines the moments of inertia as follows:
 - If only the lower bulkhead is filled with propellant then the propellant forms a solid hemisphere.
 - If the lower bulkhead and the cylindrical part are filled a solid cylinder is added to the hemisphere with a length equal to the length of the cylindrical part multiplied with the filling ratio of the cylindrical part.
 - If lower bulkhead, cylindrical part and upper bulkhead are filled then a solid hemisphere is added to the lower parts.

Optimization Constraints

Some constraints which are typically related to a detailed stage definition. As they are identical for mono and dual propellant models they are outlined together in Section 3.3.2.6

3.3.2.6.2 Dual Propellant

Description

This model subtype allows a detailed definition of dual propellant stage with liquid propellant.

Configuration

The configuration panel provides various settings to customize a detailed dual propellant vehicle stage:

- *Diameter*: define the maximum outer diameter of the stage This parameter can be set as *optimizable*.
- *Propellant sizing factor*: a linear scaling factor on the nominal propellant mass. Optionally this parameter can be set as *optimizable*. It can be defined as *Common* or *Individual* (see Section 3.1.1).
- *Nominal propellant mass* of the stage
- *Residual propellant mass* is the propellant mass at the end of the last phase where this stage is active. The default value is 0.0. This value is used by the *Residual_PROP_MASS* constraint which is automatically created for all stages.
- *Structural mass safety factor* is a dimensionless value ≥ 1.0 multiplied with the computed structural mass of the stage.
- *Mixture ratio* represents the ratio of oxidizer and fuel mass. Optionally this parameter can be set as *optimizable*. It can be defined as *Common* or *Individual* (see Section 3.1.1).

VEHICLE PARTS AND PROPERTIES MODELS

- *Tank configuration*: define how the oxidizer and fuel tank are arranged in the stage. The different options available for this purpose are described further .
- *Interstage*: defines the properties of an interstage between the forward skirt and the next stage above. Its optional configuration is *disabled* by default meaning no interstage is present. The detailed settings for an *enabled* interstage are described .
- *Payload adapter*: defines the properties of a configurable payload adapter. By default, its configuration is *disabled* meaning no payload adapter is used. The detailed settings for an *enabled* payload adapter are described .
- *Forward skirt* defines the properties of the forward skirt of the stage. Detailed settings are described .
- *Oxidizer tank*: define the properties of the oxidizer tank. Detailed settings are described .
- *Fuel tank*: define the properties of the fuel tank. Detailed settings are described .
- *Aft skirt* defines the properties of the aft skirt of the stage. Detailed settings are described .
- *Thrust frame*: define the properties of the thrust frame that connects the engine with the first (lower) tanks of its related stage. Detailed settings are described .
- The generic settings *Mass distribution*, *Electrical power* and *Thermal characteristics* are explained in Section 3.1.2, Section 3.1.3 and Section 3.1.4 respectively. The latter two settings are available if the corresponding features in the *ASTOS* software have been installed and then enabled in the scenario *Active features*.

Tank Configuration Settings

Four basic tank configurations are available for dual propellant design stage:

- *Common Bulkhead*: oxidizer and fuel tank share a common bulkhead. The user has to specify which of the tanks defines the *complete tank* (forward bulkhead, cylindrical part and aft bulkhead) (see Fig. 3.22).

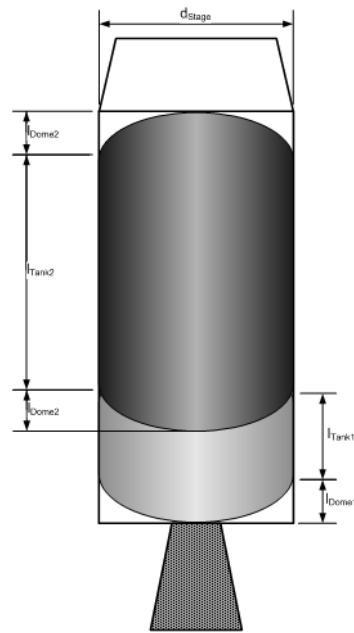


Fig. 3.22: Common Bulkhead configuration (upper tank defined as complete tank)

- **Separate Tanks:** oxidizer and fuel tank are completely separated (see Fig. 3.23). The user has to configure an intertank describing the connection between the two tanks. The configuration comprises:
 - *Ratio to tank dome length:* the ratio of the intertank length to the average dome length of both tanks.
 - *Mass Computation* defines the *Stiffening* and *material* of the *cylinder part* of the intertank.

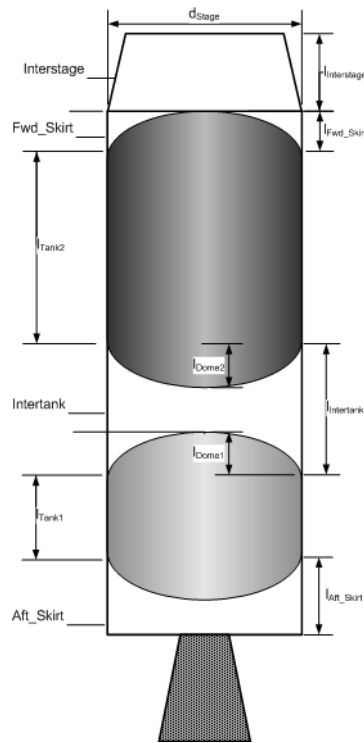


Fig. 3.23: Separate Tanks configuration and general design parameters

- **Enclosed Tank:** the aft tank is enclosed in the forward tank (e.g. the fuel tank encloses the oxidizer tank), see Fig. 3.24. The user has to specify the *tank diameter ratio*: the ratio between the enclosed tank diameter and the enclosing tank diameter (i.e. less than 1) and the *forward tank*, i.e. the enclosing one.

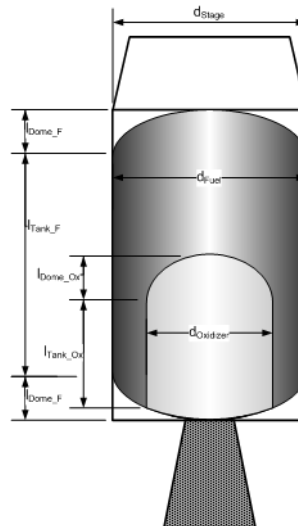


Fig. 3.24: Enclosed Tank configuration

- **Suspended Tank:** oxidizer and fuel tank are completely separated (see Fig. 3.25). The forward and aft skirts are attached to the forward tank. The user has to specify the *tank diameter ratio*: the ratio between the suspended tank diameter and the forward tank

diameter (i.e. less than 1). The user has to configure an intertank defined by a cylinder and struts that connect the two tanks. The configuration comprises:

- *Gap between bulkheads*: the distance between the lowest point of the aft dome of the forward tank and the highest point of the forward dome of the aft tank.
- *Cylinder segment length*: the length of the cylinder required to avoid that the struts intersect the aft dome of the forward tank.
- *Mass Computation* settings depend on the *mass computation method* (see Section 3.3.1.3) selected for the entire stage:
 - *Mass Estimation Relationship*: define the coefficients *A* and *B* of a *linear* law for the *surface area* as *parameter*.
 - *Structural Analysis*: define the *wall thickness* of the intertank and several *material* properties.
 - *ODIN*: define the *Stiffening* and *material* of the *cylinder part* and *Struts* of the intertank.

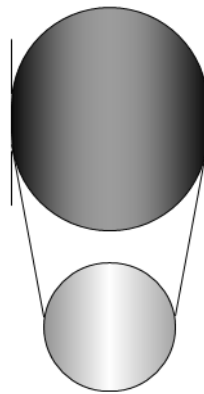


Fig. 3.25: Suspended Tank configuration.

Interstage Settings

The interstage lower diameter is equal to the current stage diameter; the upper diameter is equal to the diameter of the stage above. The interstage configuration requires the following user inputs:

- *Minimum Length* is used to define the interstage length (see $l_{Interstage}$ in Fig. 3.23) together with the *Gap between bulkhead and nozzle*.
- *Gap between bulkhead and nozzle* is the distance between the topmost point of the forward bulkhead of the current stage and the end of the nozzle (i.e. lowest point) of the engine in the stage above. The $l_{Interstage}$ is computed considering this gap together with the length of the forward dome and skirt as well as the length of the aft skirt, thrust frame and nozzle of the stage above. The resulting length is compared with the *Minimum Length* and the maximum is used to define $l_{Interstage}$.

Tip: The common way to define a fixed interstage length is to set the *Gap between bulkhead and nozzle* to a small negative number (e.g. -20 meters), in this way the *Minimum Length* is used.

- *Mass Computation* defines the *Stiffening* and *material* of the interstage cone.

Payload Adapter Settings

For the payload adapter, only its *mass* has to be specified. This mass (and also the mass of a payload component without specified dimensions) is defined as a point mass located as shown in Fig. 3.26.

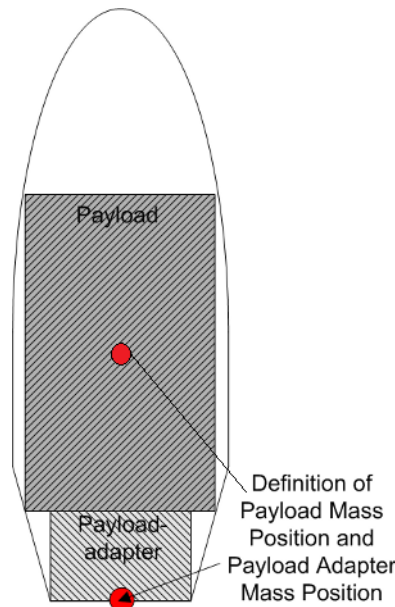


Fig. 3.26: Center of mass position of the payload adapter (and of a payload mass without specified dimensions)

Tip: If the center of mass position of the payload adapter is of major importance, the adapter can also be defined separately as an auxiliary component with customizable dimensions.

Forward and Aft Skirt Settings

The options for configuring the forward and aft skirt of the stage are equivalent and comprise:

- *Ratio to tank dome length*: the ratio (>1) of the skirt length l_{Fwd_Skirt} (or l_{Aft_Skirt} respectively) to the dome length l_{Dome}
- *Mass Computation* defines the *stiffening* and *material* of the *cylinder* part of the skirts

Oxidizer and Fuel Tank Settings

The configuration of the propellant tanks require several user inputs:

- *Length without domes* is the *optimizable* length l_{Tank} of the cylindrical part of the propellant tank.
- *Cylinder segment length* is optional and specifies the length of a single segment of the cylindrical part of the tank. If not specified, the upper bound of the tank length without domes is used, and if this is also not specified its value is used. The specified cylinder segment length and the length without domes (upper bound or value) yield the number of cylindrical segments. If *default* the cylindrical tank part will consist of one segment.
- *Volumetric loading fraction* is the ratio of the initial volume filled with propellant to the total tank volume.
- *Propellant density*: the average density of the propellant (default value of 1550.0 kg/m³)
- *Tank pressure* is the pressure in the propellant tank. It can be defined as *Common* or *Individual* (see Section 3.1.1). The tank pressure can either be *defined* by a constant *Value* or by providing a *Profile* as function of the fill ratio (refer to Profile Data Tables in User Manual for more information about profiles).
- *Domes*: define the shape of the tank domes which are either *Spherical* or *Ellipsoidal*. In the latter case, the user has to specify an *Ellipse ratio*, i.e. the ratio of tank radius to tank dome height.
- *Mass Computation* defines the *stiffening* and *material* of the *forward bulkhead*, *cylindrical part* and *aft bulkhead*. Depending on the selected *Tank configuration* not all parts are visible. See the paragraph about for further information.

Thrust Frame Settings

The options for configuring the thrust frame are:

- *Length ratio of thrust frame to tank dome*: define the geometry of the thrust frame using the ratio of the dome length l_{dome} to the length of the thrust frame l_{thrust_frame} (see Fig. 3.27).

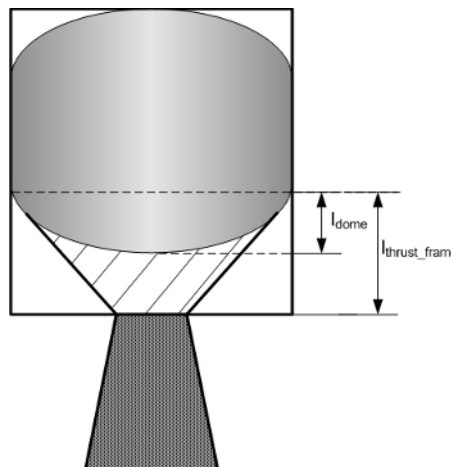


Fig. 3.27: Length definitions for thrust frame

- *Mass Computation* defines the *stiffening* and *material* of the *Cylinder* and the *struts* of the thrust frame.

Model Background

The propellant mass of the stage is determined by the *propellant sizing factor* z :

$$m_p = m_{p0} z \quad (3.100)$$

where m_{p0} is the *nominal propellant mass* specified by the user. The oxidizer (subscript ox) and fuel (subscript f) mass is a function of the *mixture ratio* f of oxidizer to fuel:

$$f = \frac{m_{ox}}{m_f} \quad (3.101)$$

$$m_{ox} = \frac{m_p}{z \left(1 + \frac{1}{f} \right)} \quad (3.102)$$

$$m_f = \frac{m_p}{z (1 + f)} \quad (3.103)$$

The required volume for these masses is a function of the *propellant density* ρ :

$$V_{ox} = \frac{m_{ox}}{\rho_{ox}} \quad (3.104)$$

$$V_f = \frac{m_f}{\rho_f} \quad (3.105)$$

The volumes of oxidizer and fuel tank are functions of the tank lengths and diameters. The latter is equivalent for both tanks in case of *separate tanks* and *common bulkhead* configuration:

$$V_{\text{tank-ox}} = l_{ox} d \quad (3.106)$$

$$V_{\text{tank-f}} = l_f d \quad (3.107)$$

Moments of Inertia

The moments of inertia of this model are computed separately for propellant and structure. For the structure a hollow cylinder is assumed using a wall thickness of a steel tank computed as $Wall = 2.5 \cdot 10^3 \cdot d$ with d being the diameter of the stage. The center of mass is assumed to be the center node of this component. For the moments of inertia of the fuel and of the oxidizer the following assumptions are used.

- If the length of the tank is smaller than the diameter a flat tank is approximated as solid cylinder with a radius of 95% of the stage diameter and a length equal to the tank length multiplied with the filling ratio.
- If the length of the tank is greater than the diameter the fill level of the tank determines the moments of inertia as follows:
 - If only the lower bulkhead is filled with propellant then the propellant forms a solid hemisphere.
 - If the lower bulkhead and the cylindrical part are filled a solid cylinder is added to the hemisphere with a length equal to the length of the cylindrical part multiplied with the filling ratio of the cylindrical part.

- If lower bulkhead, cylindrical part and upper bulkhead are filled then a solid hemisphere is added to the lower parts.

Optimization Constraints

Some constraints which are typically related to a detailed stage definition. As they are identical for mono and dual propellant models they are outlined together in Section 3.3.2.6

3.3.2.7 Detailed Vehicle Design Booster

Description

This model is used to configure both structural and propellant mass and to design the shape of the booster (e.g. the length and diameter of each tank). For accurate computation the basic architecture of the booster can be defined in detail. The user can choose between a mono (solid or liquid) or dual (liquid) propellant booster. The configuration panel as well as detailed description for mono propellant can be found in Section 3.3.2.7.1, for dual propellant see Section 3.3.2.7.2.

Optimization constraints

Some optimization constraints are typically related to the design booster definition. As they apply to both mono and dual propellant models they are shortly outlined together below:

- The *volumetric loading fraction* u relates the tank volume to the volume of the propellant and it has to be ensured that the utilizable tank volume provides enough space to carry all the propellant, i.e.:

$$V_{\text{tank}}u \geq V_p \quad (3.108)$$

for mono propellant and

$$V_{\text{tank-f}}u \geq V_f \quad (3.109)$$

$$V_{\text{tank-ox}}u \geq V_{ox} \quad (3.110)$$

for dual propellant models respectively.

For the mono propellant model, this can be realized by a *Equipment Design* constraint subtype set as initial boundary or parameter constraint. The *Parameter* should be set to *Initial_Tank_Volume_Ratio* which is defined as the dimensionless ratio of the usable tank volume to the propellant volume. A *lower limit* of 1.0 assures that the propellant volume does not exceed the available tank volume.

For the dual propellant model, two separate *Equipment Design* constraints have to be generated equivalently (i.e. *Parameter* = *Initial_Fuel_Tank_Volume_Ratio* and *Parameter* = *Initial_Ox_Tank_Volume_Ratio*).

- To ensure that the engine diameter does not exceed the diameter of the stage an *Equipment Design* constraint with *Parameter* = *Engine_to_Stage_Diameter_Ratio* is suitable. It computes the dimensionless ratio of the engine nozzle diameter to the stage

diameter. Choose *upper limit* as *limit type* with a value of *1.0* to set the engine diameter upper bound to the stage diameter.

- To ensure that the engine length of the stage above does not exceed the length of the interstage an *Equipment Design* constraint with *Parameter = Engine_to_Interstage_Length_Ratio* is suitable. It computes the dimensionless ratio of the engine length (from stage above) to the interstage length. The engine length is calculated assuming an expansion ratio of 58.2 (Vulcain 2 engine) and a characteristic length of 0.9m (mean value for all propellant/oxidizer combinations); only the *Throat Design Rocket* models with RPA and *Geometry* enabled is providing a more precise length calculation.
- The proper length of the engine with respect to the interstage is automatically set in the interstage section of the stages, no constraint is required.

3.3.2.7.1 Mono Propellant

Description

This model subtype allows a detailed definition of mono propellant booster with either liquid or solid propellant. An overview of the geometry parameters used for the configuration are shown below.

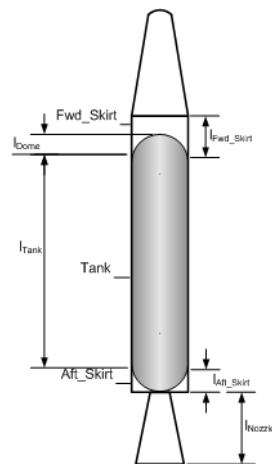


Fig. 3.28: Definition of mono propellant booster geometry parameters

Configuration

The configuration panel provides various settings to customize a detailed mono propellant vehicle booster:

- *Lower attachment point* of the booster at the related stage. It is always centered at the *Stage 1 Aft Skirt*.

Note: The booster aft end is always matched to the level of the vehicles lowest stage aft end.

VEHICLE PARTS AND PROPERTIES MODELS

- *Upper attachment point* of the booster at the related stage. Possible locations are the centers of the *Stage 1 Forward Skirt* or the *Stage 1 Intertank*.
- *Axial force introduction point* specifies whether the lower or the upper attachment point is used to introduce forces generated by the booster to the center stage of the vehicle
- *Propellant sizing factor* is a linear scaling factor on the nominal propellant mass. Optionally this parameter can be set as *optimizable*. It can be defined as *Common* or *Individual* (see Section 3.1.1).
- *Diameter* defines the maximum outer diameter of the stage This parameter can be set as *optimizable*.
- *Propellant type* defines the state of aggregation of the propellant. Possible options are *Solid* and *Liquid*. The latter requires additional settings for the thrust frame.
- *Nose*: defines the geometry and mass computation properties of the booster nose. See for more details.
- *Forward skirt* defines the properties of the forward skirt of the booster. Detailed settings are described .
- *Tank* defines the properties of the booster tank. Detailed settings are described .
- *Aft skirt* defines the properties of the aft skirt of the booster. Detailed settings are described .
- *Thrust frame*: defines the properties of the thrust frame that connects the engine with the tank of its related booster. Detailed settings are described . This panel is only present when *liquid* propellant has been selected.
- *Nominal propellant mass* of the booster
- *Structural mass safety factor* is a dimensionless multiplier used in the computation of the minimum wall thickness required to withstand the dimensioning load cases.
- *Residual Propellant Mass* can be used to set the amount of propellant which has to be inside the tank by the time it is jettisoned. The default value is 0.0 kg. If selected the available options are:
 - *Defined by* selects the kind of limit to be used. If *Range* is selected a *Lower limit* and an *Upper limit* have to be specified, otherwise only one value has to be provided.
 - Depending on the kind of limit selected above either a *Reference* value, a *Lower limit*, an *Upper limit* or a combination of those has to be inserted as residual propellant mass
- The generic settings *Mass distribution*, *Electrical power* and *Thermal characteristics* are explained in Section 3.1.2, Section 3.1.3 and Section 3.1.4 respectively. The latter two settings are available if the corresponding features in the *ASTOS* software have been installed and then enabled in the scenario *Active features*.

Nose settings

The configuration of the booster nose section requires the following user inputs:

- *Ellipse ratio* defines the shape of the ellipsoidal booster nose defined by the ratio of booster radius to nose length
- *Bulkhead* defines the bulkhead used as nose of the booster:

- *Stiffening* concept for this bulkhead
- *Material* is the name of the material for this bulkhead (e.g. AL2195 or D6AC)
- *Fixed mass* prevents the mass of this bulkhead from changing even if the geometric dimensions or the defining load case changes

Thrust frame settings

The options for configuring the thrust frame are only available when *liquid* propellant is selected. In this case they comprise:

- *Length ratio of thrust frame to tank dome* defines the geometry of the thrust frame using the ratio of the dome length l_{dome} to the length of the thrust frame l_{thrust_frame} (see Fig. 3.29). This ratio has to be ≥ 1.0 since a lower value would mean that the thrust frame intersects the tank dome!

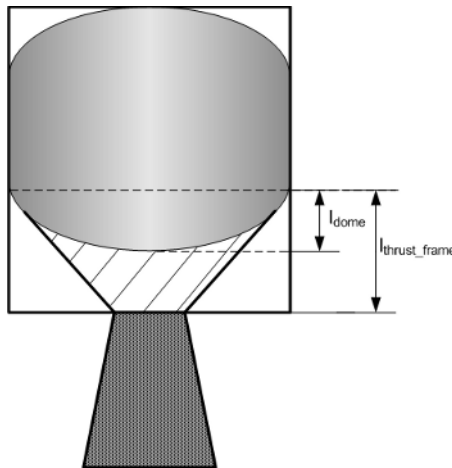


Fig. 3.29: Length definitions for thrust frame

- *Mass computation* defines the settings for computing the mass of a thrust frame:
 - *Cylinder* defines the cylindrical part of the thrust frame:
 - *Stiffening* is the stiffening concept and is selected from the drop-down menu
 - *Material* is the name of the material used in this structure
 - *Fixed mass* prevents the mass of this structure to be calculated from geometric dimensions and load case values and uses the provided mass instead
 - *Struts* defines the struts cone of the thrust frame:
 - *Stiffening* is the stiffening concept and is selected from the drop-down menu
 - *Material* is the name of the material used in this structure

VEHICLE PARTS AND PROPERTIES MODELS

- *Fixed mass* prevents the mass of this structure to be calculated from geometric dimensions and load case values and uses the provided mass instead

Tank settings

The configuration of the propellant tank requires several user inputs:

- *Length without domes* is the *optimizable* length I_{Tank} of the cylindrical part of the propellant tank.
- *Cylinder segment length* is optional and specifies the length of a single segment of the cylindrical part of the tank. If not specified, the upper bound of the tank length without domes is used, and if this is also not specified its value is used. The specified cylinder segment length and the length without domes (upper bound or value) yield the number of cylindrical segments. If *default* the cylindrical tank part will consist of one segment.
- *Volumetric loading fraction* is the ratio of the initial volume filled with propellant to the total tank volume
- *Propellant density* is the average density of the propellant
- *Ullage pressure* is the pressure in the propellant tank (chamber pressure in case of *solid* propellant). It can be defined as *Common* or *Individual* (see Section 3.1.1). The ullage pressure can either be *defined by* a constant *Value* or by providing a *Profile* as function of the fill ratio (refer to Profile Data Tables in User Manual for more information about profiles).
- *Domes* define the shape of the tank domes which is either *Spherical* or *Ellipsoidal*. In the latter case, the user has to specify an *Ellipse ratio*, i.e. the ratio of tank radius to tank dome height.
- *Mass computation* defines the settings for computing the structural mass of a tank:
 - *Forward bulkhead* defines the forward bulkhead of the tank:
 - *Stiffening* is the stiffening concept and is selected from the drop-down menu
 - *Material* is the name of the material used in this structure
 - *Fixed mass* prevents the mass of this structure to be calculated from geometric dimensions and load case values and uses the provided mass instead
 - *Cylindrical Part* defines the cylindrical part of the tank:
 - *Stiffening* is the stiffening concept and is selected from the drop-down menu
 - *Material* is the name of the material used in this structure
 - *Fixed mass* prevents the mass of this structure to be calculated from geometric dimensions and load case values and uses the provided mass instead
 - *Aft bulkhead* defines the aft bulkhead of the tank:
 - *Stiffening* is the stiffening concept and is selected from the drop-down menu
 - *Material* is the name of the material used in this structure

VEHICLE PARTS AND PROPERTIES MODELS

- *Fixed mass* prevents the mass of this structure to be calculated from geometric dimensions and load case values and uses the provided mass instead

Forward and aft skirt settings

The options for configuring the forward and aft skirt of the booster are equivalent and comprise:

- *Ratio to tank dome length* is the ratio (>1) of the skirt length l_{Fwd_Skirt} (or l_{Aft_Skirt} respectively) to the dome length l_{Dome}
- *Mass computation* defines the settings for computing the structural mass of skirts:
 - *Cylindrical Part* defines the cylindrical part of the skirt:
 - *Stiffening* is the stiffening concept and is selected from the drop-down menu
 - *Material* is the name of the material used in this structure
 - *Fixed mass* prevents the mass of this structure to be calculated from geometric dimensions and load case values and uses the provided mass instead

Background

The propellant mass of the booster is determined using the *propellant sizing factor* z :

$$m_p = m_{p0} z \quad (3.111)$$

where m_{p0} is the *nominal propellant mass* specified by the user. The required volume for this mass is a function of the *propellant density* ρ_p and the volume of the propellant tank is a function of the tank *diameter* and tank length:

$$V_p = \frac{m_p}{\rho_p} \quad (3.112)$$

The volume of the propellant tank is a function of the optimizable tank diameter and tank length:

$$V_{\text{tank}} = l d \quad (3.113)$$

Moments of Inertia

The moments of inertia of this model are computed separately for propellant and structure. For the structure a hollow cylinder is assumed using a wall thickness of a steel tank computed as $Wall = 2.5 \cdot 10^3 \cdot d$ with d being the diameter of the stage. The center of mass is assumed to be the center node of this component. The center of mass and moments of inertia of the propellant are computed based on the following assumptions:

- If the propellant is solid:
 - The propellant forms a hollow cylinder with an outer radius of $r_{prop} = 0.5 \cdot d - Wall$.
 - The inner radius of the hollow cylinder is computed based on the filling ratio of the tank with a full tank having an inner radius of 0.0

VEHICLE PARTS AND PROPERTIES MODELS

- The center of mass is located in the geometric center of the tank (see Fig. 3.28).
- If the propellant is liquid:
 - If the length of the tank is smaller than the diameter a flat tank is approximated as solid cylinder with a radius of 95% of the stage diameter and a length equal to the tank length multiplied with the filling ratio.
 - If the length of the tank is greater than the diameter the fill level of the tank determines the moments of inertia as follows:
 - If only the lower bulkhead is filled with propellant then the propellant forms a solid hemisphere.
 - If the lower bulkhead and the cylindrical part are filled a solid cylinder is added to the hemisphere with a length equal to the length of the cylindrical part multiplied with the filling ratio of the cylindrical part.
 - If lower bulkhead, cylindrical part and upper bulkhead are filled then a solid hemisphere is added to the lower parts.

Optimization Constraints

Some constraints which are typically related to a detailed booster definition. As they are identical for mono and dual propellant models they are outlined together in Section 3.3.2.7

3.3.2.7.2 Dual Propellant

Description

This model subtype allows a detailed definition of dual propellant booster with liquid propellant.

Configuration

The configuration panel provides various settings to customize a detailed dual propellant vehicle booster:

- *Nominal propellant mass* of the booster
- *Structural mass safety factor* is a dimensionless multiplier used in the computation of the minimum wall thickness required to withstand the dimensioning load cases.
- *Lower attachment point* of the booster at the related stage. It is always centered at the *Stage 1 Aft Skirt*.

Note: The booster aft end is always matched to the level of the vehicles lowest stage aft end.

VEHICLE PARTS AND PROPERTIES MODELS

- *Upper attachment point* of the booster at the related stage. Possible locations are the centers of the *Stage 1 Forward Skirt* or the *Stage 1 Intertank*.
- *Axial force introduction point* specifies whether the lower or the upper attachment point is used to introduce forces generated by the booster to the center stage of the vehicle
- *Residual Propellant Mass* can be used to set the amount of propellant which has to be inside the tank by the time it is jettisoned. The default value is 0.0 kg. If selected the available options are:
 - *Defined by* selects the kind of limit to be used. If *Range* is selected a *Lower limit* and an *Upper limit* have to be specified, otherwise only one value has to be provided.
 - Depending on the kind of limit selected above either a *Reference* value, a *Lower limit*, an *Upper limit* or a combination of those to be inserted as residual propellant mass
- *Propellant sizing factor* is a linear scaling factor on the nominal propellant mass. Optionally this parameter can be set as *optimizable*. It can be defined as *Common* or *Individual* (see Section 3.1.1).
- *Diameter* defines the maximum outer diameter of the stage This parameter can be set as *optimizable*.
- *Mixture ratio* represents the ratio of oxidizer and fuel mass. Optionally this parameter can be set as *optimizable*.
- *Nose* defines the geometry and mass computation properties of the booster nose. See for more details.
- *Forward skirt* defines the properties of the forward skirt of the booster. Detailed settings are described .
- *Tank configuration* defines how the oxidizer and fuel tank are arranged in the booster. The different options available for this purpose are described in the .
- *Oxidizer tank* defines the properties of the oxidizer tank. Detailed settings are described below in the .
- *Fuel tank* defines the properties of the fuel tank. Detailed settings are described below in the .
- *Aft skirt* defines the properties of the aft skirt of the booster. Detailed settings are described below in the .
- *Thrust frame* defines the properties of the thrust frame of the booster. Detailed settings are described below in the .
- The generic settings *Mass distribution*, *Electrical power* and *Thermal characteristics* are explained in Section 3.1.2, Section 3.1.3 and Section 3.1.4 respectively. The latter two settings are available if the corresponding features in the *ASTOS* software have been installed and then enabled in the scenario *Active features*.

Nose settings

The configuration of the booster nose section requires the following user inputs:

- *Ellipse ratio* defines the shape of the ellipsoidal booster nose defined by the ratio of booster radius to nose length

VEHICLE PARTS AND PROPERTIES MODELS

- *Bulkhead* defines the bulkhead used as nose of the booster:
 - *Stiffening* concept for this bulkhead
 - *Material* is the name of the material for this bulkhead (e.g. AL2195 or D6AC)
 - *Fixed mass* prevents the mass of this bulkhead from changing even if the geometric dimensions or the defining load case changes

Oxidizer and fuel tank settings

The configuration of the propellant tanks require several user inputs:

- *Length without domes* is the *optimizable* length l_{Tank} of the cylindrical part of the propellant tank.
- *Cylinder segment length* is optional and specifies the length of a single segment of the cylindrical part of the tank. If not specified, the upper bound of the tank length without domes is used, and if this is also not specified its value is used. The specified cylinder segment length and the length without domes (upper bound or value) yield the number of cylindrical segments. If *default* the cylindrical tank part will consist of one segment.
- *Volumetric loading fraction* is the ratio of the initial volume filled with propellant to the total tank volume
- *Propellant density* is the average density of the propellant
- *Ullage pressure* is the pressure in the propellant tank (chamber pressure in case of *solid* propellant). It can be defined as *Common* or *Individual* (see Section 3.1.1). The ullage pressure can either be *defined by* a constant *Value* or by providing a *Profile* as function of the fill ratio (refer to Profile Data Tables in User Manual for more information about profiles).
- *Domes* define the shape of the tank domes which is either *Spherical* or *Ellipsoidal*. In the latter case, the user has to specify an *Ellipse ratio*, i.e. the ratio of tank radius to tank dome height.
- *Mass computation* defines the settings for computing the structural mass of a tank:
 - *Forward bulkhead* defines the forward bulkhead of the tank:
 - *Stiffening* is the stiffening concept and is selected from the drop-down menu
 - *Material* is the name of the material used in this structure
 - *Fixed mass* prevents the mass of this structure to be calculated from geometric dimensions and load case values and uses the provided mass instead
 - *Cylindrical Part* defines the cylindrical part of the tank:
 - *Stiffening* is the stiffening concept and is selected from the drop-down menu
 - *Material* is the name of the material used in this structure
 - *Fixed mass* prevents the mass of this structure to be calculated from geometric dimensions and load case values and uses the provided mass instead
 - *Aft bulkhead* defines the aft bulkhead of the tank:

- *Stiffening* is the stiffening concept and is selected from the drop-down menu
- *Material* is the name of the material used in this structure
- *Fixed mass* prevents the mass of this structure to be calculated from geometric dimensions and load case values and uses the provided mass instead

Thrust frame settings

The available options are:

- *Length ratio of thrust frame to tank dome* defines the geometry of the thrust frame using the ratio of the dome length l_{dome} to the length of the thrust frame l_{thrust_frame} (see Fig. 3.30). This ratio has to be ≥ 1.0 since a lower value would mean that the thrust frame intersects the tank dome!

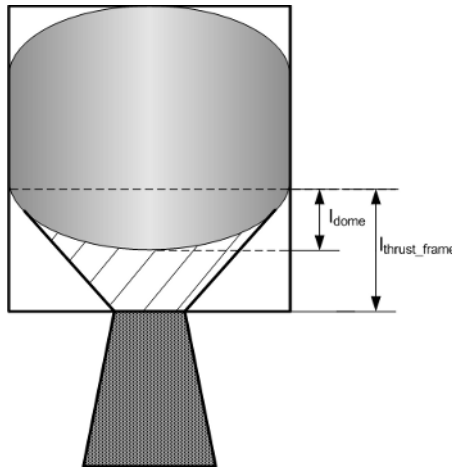


Fig. 3.30: Length definitions for thrust frame

- *Mass computation* defines the settings for computing the mass of a thrust frame:
 - *Cylinder* defines the cylindrical part of the thrust frame:
 - *Stiffening* is the stiffening concept and is selected from the drop-down menu
 - *Material* is the name of the material used in this structure
 - *Fixed mass* prevents the mass of this structure to be calculated from geometric dimensions and load case values and uses the provided mass instead
 - *Struts* defines the struts cone of the thrust frame:
 - *Stiffening* is the stiffening concept and is selected from the drop-down menu
 - *Material* is the name of the material used in this structure

- *Fixed mass* prevents the mass of this structure to be calculated from geometric dimensions and load case values and uses the provided mass instead

Tank configuration setting

Three basic tank configurations are available for dual propellant design booster:

- *Common Bulkhead*: oxidizer and fuel tank share a common bulkhead. The user has to specify which of the tanks defines the *complete tank* (see Fig. 3.31).

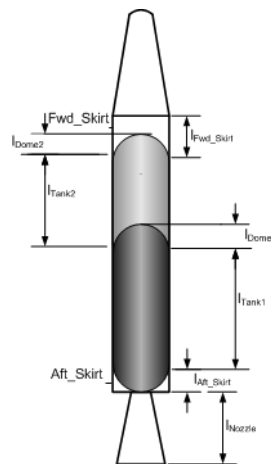


Fig. 3.31: Booster tanks in Common Bulkhead configuration

- *Separate Tanks*: oxidizer and fuel tank are completely separated (see Fig. 3.23). The user has to configure an intertank describing the section between the two tanks. The configuration comprises:
 - *Ratio to tank dome length* is the ratio of the intertank length to the average dome length of both tanks.
 - *Mass Computation* settings depend on the *mass computation method* (see Section 3.3.1.3) selected for the entire booster:
 - *Mass Estimation Relationship* defines the coefficients *A* and *B* of a *linear law* for the *surface area* as *parameter*.
 - *Structural Analysis* defines the *wall thickness* of the intertank and several *material* properties.
 - *ODIN* defines the *Stiffening* and *material* of the *cylinder part* of the intertank.

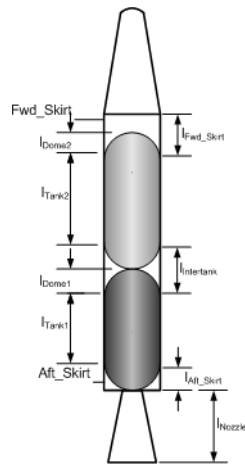


Fig. 3.32: Separate Tanks configuration and general booster design parameters

- **Enclosed Tank:** one tank encloses the other. The following settings are available:
 - **Tank diameter ratio** is the ratio of the diameter of the inner tank to the diameter of the outer tank
 - **Forward tank** is the larger, outer tank. The enclosed tank will be at the aft end of this tank.

An example for this configuration can be seen in Fig. 3.33.

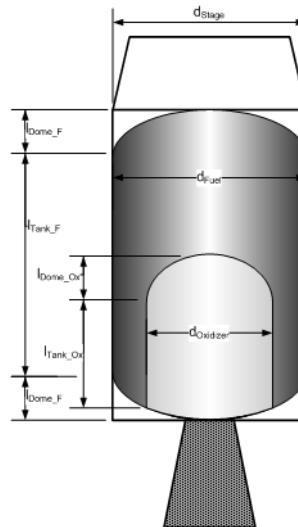


Fig. 3.33: Enclosed tank booster configuration with the fuel tank in the forward position and the oxidizer tank as enclosed tank

Forward and aft skirt settings

The options for configuring the forward and aft skirt of the booster are equivalent and comprise:

- **Ratio to tank dome length** is the ratio (>1) of the skirt length l_{Fwd_Skirt} (or l_{Aft_Skirt} respectively) to the dome length l_{Dome}

VEHICLE PARTS AND PROPERTIES MODELS

■ *Mass computation* defines the settings for computing the structural mass of skirts:

■ *Cylindrical Part* defines the cylindrical part of the skirt:

- *Stiffening* is the stiffening concept and is selected from the drop-down menu
- *Material* is the name of the material used in this structure
- *Fixed mass* prevents the mass of this structure to be calculated from geometric dimensions and load case values and uses the provided mass instead

Background

The propellant mass of the stage is determined by the *propellant sizing factor* z :

$$m_p = m_{p0} z \quad (3.114)$$

where m_{p0} is the *nominal propellant mass* specified by the user. The oxidizer (subscript ox) and fuel (subscript f) mass is a function of the *mixture ratio* f between oxidizer and fuel:

$$f = \frac{m_{ox}}{m_f} \quad (3.115)$$

$$m_{ox} = \frac{m_p}{z \left(1 + \frac{1}{f} \right)} \quad (3.116)$$

$$m_f = \frac{m_p}{z (1 + f)} \quad (3.117)$$

The required volume for these masses is a function of the respective *propellant density* ρ :

$$V_{ox} = \frac{m_{ox}}{\rho_{ox}} \quad (3.118)$$

$$V_f = \frac{m_f}{\rho_f} \quad (3.119)$$

The volume of oxidizer and fuel tank are functions of the tank lengths and diameters. The latter is equivalent for both tanks in case of *separate tanks* and *common bulkhead* configuration:

$$V_{\text{tank-ox}} = l_{ox} d \quad (3.120)$$

$$V_{\text{tank-f}} = l_f d \quad (3.121)$$

Moments of inertia

The moments of inertia of this model are computed separately for propellant and structure. For the structure a hollow cylinder is assumed using a wall thickness of a steel tank computed as $Wall = 2.5 \cdot 10^3 * d$ with d being the diameter of the stage. The center of mass is assumed to be the center node of this component. For the moments of inertia of the fuel and of the oxidizer the following assumptions are used.

- If the length of the tank is smaller than the diameter a flat tank is approximated as solid cylinder with a radius of 95% of the stage diameter and a length equal to the tank length multiplied with the filling ratio.
- If the length of the tank is greater than the diameter the fill level of the tank determines the moments of inertia as follows:
 - If only the lower bulkhead is filled with propellant then the propellant forms a solid hemisphere.
 - If the lower bulkhead and the cylindrical part are filled a solid cylinder is added to the hemisphere with a length equal to the length of the cylindrical part multiplied with the filling ratio of the cylindrical part.
 - If lower bulkhead, cylindrical part and upper bulkhead are filled then a solid hemisphere is added to the lower parts.

Optimization constraints

Some constraints which are typically related to a detailed booster definition. As they are identical for mono and dual propellant models they are outlined together in Section 3.3.2.7

3.3.3 Auxiliary, Payload and Fairing

Description

These models are used to define both payload components and other auxiliary vehicle components which do not contain propellant and are not linked to an engine.

3.3.3.1 Auxiliary

Description

An auxiliary component can be used to model any constant structural mass. This can be a fairing, the core structure of a satellite or an electronics box for example.

Configuration

The configuration of an auxiliary structure component requires the following user inputs:

- *Dimensions* define the dimensions and shape of the component. When set to *custom* the following options are available:
 - X, Y, Z define the dimensions
 - *Shape* defines the geometric shape of this auxiliary component:

VEHICLE PARTS AND PROPERTIES MODELS

- *Box* is a rectangular box defined by its *X*, *Y* and *Z* components
- *Cylinder* is defined with the *Cylinder axis*. This axis defines the length of the cylinder, the other two dimensions have to be equal and define the diameter.
- *Ellipsoid* is defined as the full axis length in all dimensions of the ellipsoid

When set to *default* the dimensions are automatically calculated. The equation can be found below in the background paragraph.

- *Total mass* is the structural total mass of the component
- The generic settings *Mass distribution*, *Electrical power* and *Thermal characteristics* are explained in Section 3.1.2, Section 3.1.3 and Section 3.1.4 respectively. The latter two settings are available if the corresponding features in the *ASTOS* software have been installed and then enabled in the scenario *Active features*.

For a general description about Profiles see Profile Data Tables in User Manual.

Background

Dimensions

The default dimensions are calculated according to the following power law:

$$s = \frac{m}{m_{ref}} \quad (3.122)$$

$$x = s^{\frac{1}{3}} \cdot x_{ref} \quad (3.123)$$

$$y = s^{\frac{1}{3}} \cdot y_{ref} \quad (3.124)$$

$$z = s^{\frac{1}{3}} \cdot z_{ref} \quad (3.125)$$

with m being the user-defined mass, m_{ref} a reference mass of 505.0 kg, s the sizing factor and x_{ref} a length of 3.7 m, and y_{ref} and z_{ref} with a value of 1.0 m.

Moments of Inertia

If the dimensions are set to *custom*, the user specifies the geometric shape in addition to the three dimensions. Together with the mass the moments of inertia are computed based on the assumption of solid shapes, i.e. solid cuboid, solid, cylinder or solid ellipsoid. If the dimensions are set to *default* a box is assumed as shape. The center of mass is assumed to be the center node of the component.

3.3.3.2 Payload

Description

VEHICLE PARTS AND PROPERTIES MODELS

A payload component can be used to model the payload mass carried by launchers. It has a structural mass which can be optimized by means of a scaling factor.

Configuration

The configuration panel provides various settings to customize a payload component:

- *Payload scaling factor* defines a positive scaling factor to be multiplied with the nominal payload mass. This factor can be set as *optimizable* and lower and upper bounds can be set for the scaling factor.
- *Dimensions* of the payload. When set to *custom*, the payload dimension can be specified by its X, Y and Z components. The shape is assumed as solid box. When set to *default* the dimensions are set according to the same power law as explained in the background paragraph of Section 3.3.3.1.
- *Nominal mass* is the structural total mass of the component
- The generic settings *Mass distribution*, *Electrical power* and *Thermal characteristics* are explained in Section 3.1.2, Section 3.1.3 and Section 3.1.4 respectively. The latter two settings are available if the corresponding features in the *ASTOS* software have been installed and then enabled in the scenario *Active features*.

Background

If the dimensions are set to *custom*, the user specifies the three dimensions otherwise the power law explained in the background paragraph of Section 3.3.3.1 is used. Together with the mass the moments of inertia are computed based on the assumption of a solid cuboid. The center of mass is assumed to be the center node of the component.

3.3.3.3 Basic Design Payload Fairing

Description

This model is used to configure a fairing whose mass varies with the payload it confines [21]. This model is suitable for simultaneous trajectory and vehicle stage size design in the neighborhood of a nominal vehicle design. An increase in payload mass is penalized by an increase in mass of the fairing. A nominal payload volume and a relationship between the fairing mass and the payload mass must be supplied.

Configuration

The configuration panel provides various settings to customize a basic design payload fairing component:

- *Mean payload density* is used to determine the nominal payload volume from the nominal payload mass
- *Nominal fairing mass* should represent the mass of a fairing sufficiently sized to confine the nominal payload.

VEHICLE PARTS AND PROPERTIES MODELS

- *Mass correlation coefficient* is a positive dimensionless coefficient b describing the correlation between payload and fairing mass (see).
- *Nominal payload mass* is the payload mass from which the payload volume and fairing mass changes are derived. This parameter can be set as *Optimizable* to obtain an optimized payload mass.
- The generic settings *Mass distribution*, *Electrical power* and *Thermal characteristics* are explained in Section 3.1.2, Section 3.1.3 and Section 3.1.4 respectively. The latter two settings are available if the corresponding features in the ASTOS software have been installed and then enabled in the scenario *Active features*.

Background

In this model, an analytical relationship between the mass of the fairing m_F and the payload mass m_P is postulated of the form

$$m_F(m_P) = a \left(\frac{m_P}{\rho_P} \right)^b ; \quad a = m_{F,0} \left(\frac{m_{P,0}}{\rho_P} \right)^{-b} \quad (3.126)$$

where $m_{P,0}$, $m_{F,0}$ and ρ_P are the *nominal payload mass*, *nominal fairing mass* and *mean payload density* respectively as specified by the user. The *mass correlation coefficient* b has to be provided in relation to SI units (m^3 and Kg). Example numeric values are listed in Table 3.10 .

Table 3.10: Typical correlation coefficients for the model

Fairing Type	Correlation Coefficients	
	a	b
Single payload fairing	8.6956	1.0686
Dual payload fairing	9.3594	1.0986

Optimization Constraints

A basic design payload fairing component induces the definition of an (optimizable) real parameter (*Nominal_Payload_Mass*) which is equal to the unscaled payload mass. To link this parameter with the actual payload, please manually add a *Model Params Ratio* constraint as initial boundary in the first phase. The two parameters to be linked are the *Nominal_Payload_Mass* of the basic design fairing and the *payload_scale_factor* of the payload component. The reference value to be defined is the ratio of the nominal mass to the scaling factor of the payload.

Tip: Usually with a payload scaling factor of 1, the reference value is equal to the nominal payload mass.

3.3.3.4 Detailed Design Payload Fairing

Description

This model is used to design the shape and mass of a fairing. The following picture shows the basic architecture and geometry parameter definitions of a detailed design payload fairing.

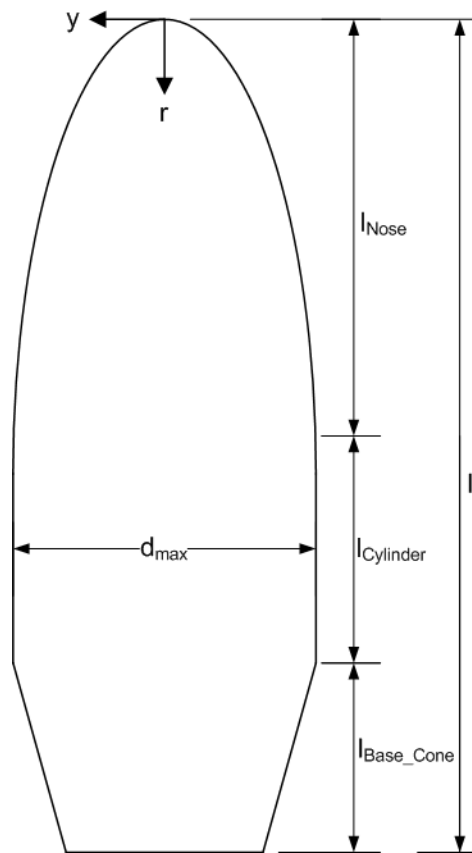


Fig. 3.34: Dimensions and shape of a Detailed Design Payload Fairing model

Configuration

The configuration panel provides various settings to customize a detailed design payload fairing component:

- *Nose shape exponent* is an exponent n for the power law used to describe the nose shape of the fairing (see)
- *Cylinder length ratio* is a dimensionless value ratio describing the ratio of the total cylinder length $l_{Cylinder}$ to the overall length of the fairing l
- *Structural mass safety factor* is a dimensionless multiplier for the loads affecting the vehicle. This influences the calculation of the minimum wall thickness required to withstand the dimensioning loads.
- *Length* defines the nominal overall length l of the fairing. If this parameter is set as *optimizable*, the size of the fairing can be optimized. It can be defined as *Common* or *Individual* (see Section 3.1.1).
- *Maximum diameter* provides optional settings for the geometry of the fairing. In case this setting is *enabled*, the user has to define:

- *Base frustum length ratio*: the ratio of the length l_{Base_Cone} of the base frustum to the overall fairing length l
- *Max. diameter*: the maximum diameter d_{max} of the fairing which can optionally be defined as *optimizable* parameter.

If set to *disabled* the fairing will not contain a base frustum and the diameter (d_{max} in the picture above) of the fairing will be taken from the rocket stage this fairing is connected to. If set to *enabled* the diameter at the lower end of the base frustum will match the stage this fairing is connected to.

- *Mass computation* defines the *stiffening* and *material* of the *Elliptical part*, *Cylindrical part* and the *Frustum part*. The last part is only visible if *Maximum diameter* is enabled.
- The generic settings *Mass distribution*, *Electrical power* and *Thermal characteristics* are explained in Section 3.1.2, Section 3.1.3 and Section 3.1.4 respectively. The latter two settings are available if the corresponding features in the ASTOS software have been installed and then enabled in the scenario *Active features*.

Background

The nose shape is computed using the user inputs for the *max. diameter* d_{max} of the fairing and the *exponent* n of the power law:

$$\frac{2y}{d_{max}} = \left(\frac{x}{l_{Nose}} \right)^n \quad (3.127)$$

The required nose length l_{Nose} is derived from the user-defined overall length and the various length ratios.

Fig. 3.35 below shows the dependency of the nose radius on the exponent n in Eq. 3.127.

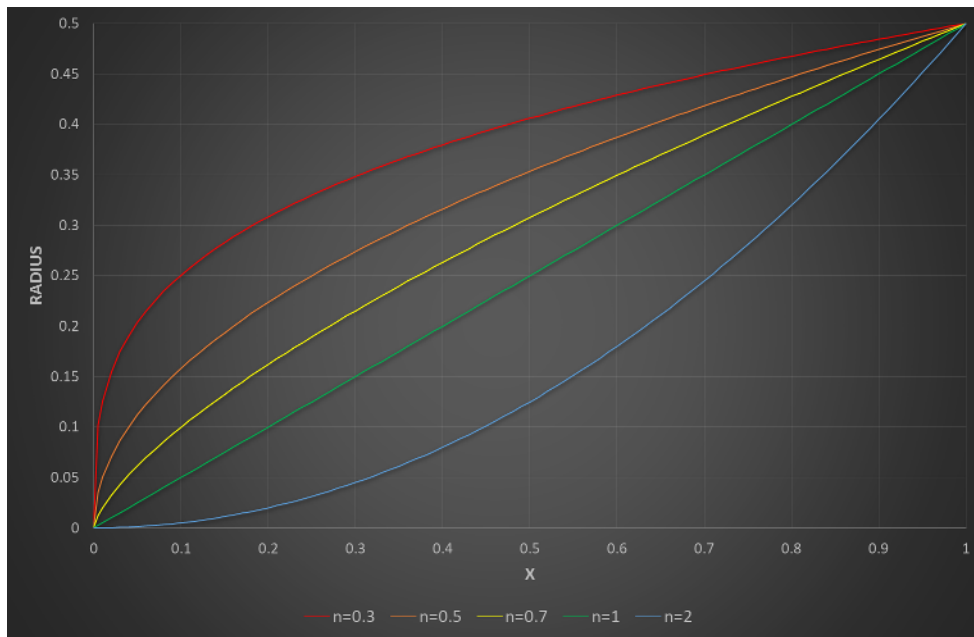


Fig. 3.35: Dependency of the nose radius on the exponent n

Moments of Inertia

This model has a detailed geometric shape. The mass of this model is equally distributed over several point masses. For each point mass the precise position is known so that the inertia matrix is built by summing up the point masses with the equation $I_{point} = m_{point} * r^2$ with r being the distance from the axis.

3.3.4 Reentry Vehicle Models

Description

This section describes the configuration of the re-entry vehicle models that are implemented in ASTOS. They can be used for both trajectory and vehicle design optimization. For that purpose, all shape parameters of various vehicle geometries are generally *optimizable* between user-defined bounds.

Different methods exist for the *mass computation* of the re-entry vehicle and optionally a *Thermal Protection System (TPS)* can be configured in detail.

Geometry settings

Depending on the selected re-entry vehicle subtype, the user has to specify a set of related geometry parameters. The individual settings for the different subtypes are detailed in Section 3.3.4.1 to Section 3.3.4.5

Mass computation settings

The *mass computation* options available for re-entry vehicles are:

- *Constant Mass:*
Use this setting to directly apply a constant mass to the re-entry vehicle, independent of its dimensions.
- *Mass Estimation Relationship:*
Use this option to apply a *linear* mass estimation using the *surface area* as *parameter*. For more information on mass estimation methods and typical parameter values refer to Section 3.3.1.3.

Thermal protection system settings

The *Thermal Protection System (TPS)* for a re-entry vehicle is an optional settings which is *disabled* by default. If *enabled*, the user can configure a thermal protection coating which is composed of one or multiple layers. The thickness of the individual layers is *optimizable*.

Configuration

The configuration of the TPS (identical for all re-entry vehicle subtypes) requires several user inputs:

VEHICLE PARTS AND PROPERTIES MODELS

- *TPS covered area*: specifies the percentage of the forebody area (i.e. the vehicles body area from the front to the largest diameter) that is covered with TPS
 - *Heat load ratio*: specifies the ratio of the average heat load to the maximum heat load on the TPS
 - *Initial temperature*: specifies the uniform initial TPS temperature.
 - *Maximum bondline temperature*: specifies the the acceptable maximum bondline temperature of the vehicles surface (i.e. without the TPS layers)
 - *Environment*: specifies the environment (celestial body) for the re-entry to ensure that the proper atmospheric model is used
 - *Layer*: specifies the properties of the thermal protection coating. It is possible to **Add** or **Remove** layers using the respective buttons. If more than one layer is defined, thermally perfect contact between the layers is assumed (the first layer is considered as outermost). User inputs for each layer are:
 - *Thickness*: of the layer which can optionally be defined as *optimizable* parameter.
 - *Type*: either choose from selected existing heat shield materials (see Table 3.11) or create a *user-defined ablative* or *user-defined reusable* material.
The thickness of reusable types remains constant throughout the scenario while the thickness of ablative types decreases depending on the environmental conditions. Steady-state ablation is assumed, i.e. the complete net incoming heat flux is ablated.
- Note:** If more than one layer is defined, only the first (i.e. outer) layer can be *user-defined ablative*.
- *Discretization*: if a *custom*, configuration is selected, the user can specify the *number of temperature nodes* in the model grid. The *default* grid uses 6 nodes.

Model Background

Fig. 3.36 illustrates a TPS profile with n layers with the respective heat flux and temperature definitions.

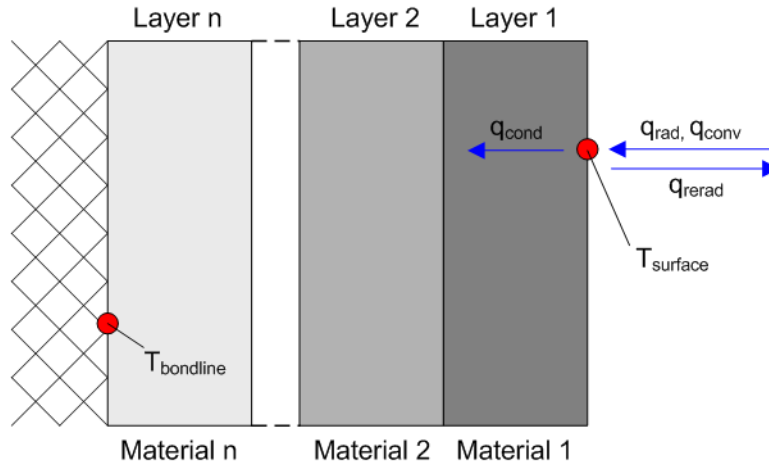


Fig. 3.36: TPS profile with heat flux and temperature definitions

where:

- q_{conv} is the convective fraction of the heat flux on the vehicles TPS surface area
- q_{rad} is the radiative fraction of the heat flux on the vehicles TPS surface area
- q_{rerad} is a the heat flux reradiation originated by the vehicles surface
- q_{cond} is the heat flux through the TPS material
- $T_{surface}$ is the temperature of the TPS surface area
- $T_{bondline}$ is the temperature of the surface area of the vehicle (i.e. the bondline where the TPS material is connected with the vehicle).

The mass of the TPS is determined from the following sum of all n layer masses:

$$m_{TPS} = \frac{Q_{ave}}{Q_{max}} A_{TPS} \sum_{i=1}^n t_{TPS,i} \rho_{TPS,i} \quad (3.128)$$

where $t_{TPS,i}$ and $\rho_{TPS,i}$ are the thickness and density of layer i . A_{TPS} is the surface area of the outer TPS layer and the quotient of Q_{ave} and Q_{max} represent the heat load ratio specified by the user.

For ablative materials, the declination of the material density ρ over time t is derived from the Arrhenius equation of the form:

$$\frac{\delta \rho}{\delta t} = -k \left(\frac{\rho - \rho_c}{\rho_v - \rho_c} \right)^n e^{-\frac{B}{T}} \quad (3.129)$$

Here ρ_c and ρ_v are the *Density Char* and *Density Virgin* of the material respectively. The *density exponent* n , the *activation energy coefficient* B and the *prefactor* k are further parameters which have to specified by the user. T is the current TPS temperature which is internally computed. The tables below summarize the material properties of predefined materials and typical values to be used in the Arrhenius equation.

Table 3.11: TPS material properties

VEHICLE PARTS AND PROPERTIES MODELS

Property	LI 900	AVCOAT 5026	PICA	SLA 561
Type	Reusable	Ablative	Ablative	Ablative
Thermal conductivity [W/m/K]	0.0476	0.297	0.402	0.0592
Specific heat [J/kg/K]	628.0	1700.0	984.0	1170.0
Absorptivity [-]	1.0	1.0	1.0	1.0
Emissivity [-]	0.85	0.667	0.8	0.9
Density [kg/m ³]	144.2	N/A	N/A	N/A
Max. temperature [K]	1590.0	N/A	N/A	N/A
Density virgin [kg/m ³]	N/A	529.0	266.0	264.0
Density char [kg/m ³]	N/A	264.5	210.0	145.0
Heat of ablation [MJ/kg]	N/A	13.8	13.8	54.1
Ablation temperature [K]	N/A	922.0	922.0	922.0

Note: It is not possible to define temperature-dependent TPS material data.

Table 3.12: Parameters for the Arrhenius equation

Property	LI 900	AVCOAT 5026	PICA	SLA 561
Prefactor [lb/ft ³ /h]	N/A	7.22E8	7.22E8	1.8126E11
Density exponent [-]	N/A	2.5	2.5	3.0
Activation energy coefficient [K]	N/A	12111.0	12111.0	19000.0

Optimization Constraints

Some constraints which are typically related to the definition of a re-entry vehicle are shortly outlined below:

- To ensure that the vehicle dimensions do not exceed or fall below a certain limit a *Vehicle_Length* constraint (type *Equipment Design*) is suitable. Choose *lower limit*, *upper limit* or *range* as *limit type* and specify the desired bounds. Link the constraint to the re-entry vehicle by specifying its respective *Component ID*.
- The same holds if the vehicle volume has to be kept between certain limits. Here a *Vehicle_Volume* constraint (type *Equipment Design*) is suitable. Choose *lower limit*, *upper*

limit or *range* as *limit* type and specify the desired bounds. Link the constraint to the re-entry vehicle by specifying its respective *Component ID*).

3.3.4.1 Sphere Cone

All required quantities to define the vehicle geometry are shown in Fig. 3.40 (the assumed flight direction is from the left to the right). The link to the labels in the configuration panel is provided in the subsequent list together with the default values.

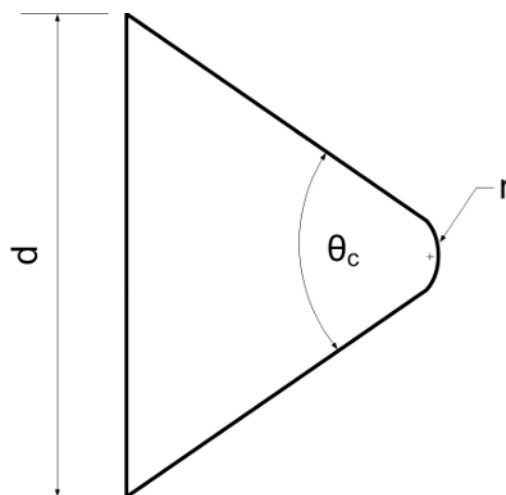


Fig. 3.37: Sphere Cone geometry

Table 3.13: Geometry parameter definition

Label	Symbol
Cone base diameter	d
Nose radius	r
Cone angle	θ_c

Mass computation and Thermal protection system (TPS) settings are described in Section 3.3.4 as they are equivalent for all geometry subtypes.

3.3.4.2 Capsule

All required quantities to define the vehicle geometry are shown in Fig. 3.38 (the assumed flight direction is from the left to the right). The link to the labels in the configuration panel is provided in the subsequent list together with the default values.

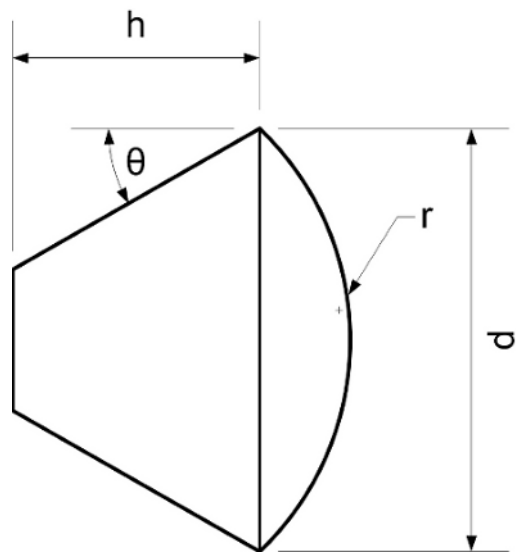


Fig. 3.38: Capsule geometry

Table 3.14: Geometry parameter definition

Label	Symbol	
Nose radius	r	
Diameter	d	3.0 Meter
Frustum height	h	2.7 Meter
Frustum angle	θ	30.0 Degree

Mass computation and Thermal protection system (TPS) settings are described in Section 3.3.4 as they are equivalent for all geometry subtypes.

3.3.4.3 Probe

All required quantities to define the vehicle geometry are shown in Fig. 3.39 (the assumed flight direction is from the left to the right). The link to the labels in the configuration panel is provided in the subsequent list together with the default values.

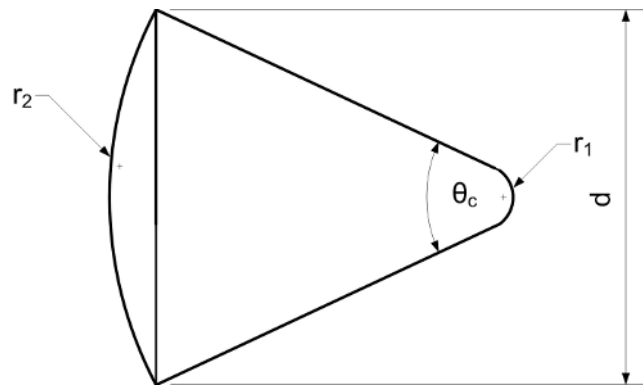


Fig. 3.39: Probe geometry

Table 3.15: Geometry parameter definition

Label	Symbol
Cone base diameter	d
Cone angle	θ_c
Nose radius	r_1
Backshell radius	r_2

Mass computation and Thermal protection system (TPS) settings are described in Section 3.3.4 as they are equivalent for all geometry subtypes.

3.3.4.4 Biconic

All required quantities to define the vehicle geometry are shown in Fig. 3.40 (the assumed flight direction is from the left to the right). The link to the labels in the configuration panel is provided in the subsequent list together with the default values.

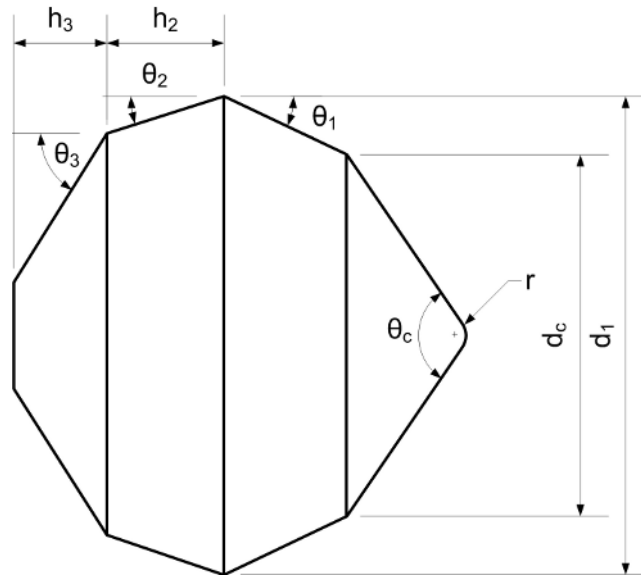


Fig. 3.40: Biconic geometry

Table 3.16: Geometry parameter definition

Label	Symbol
Nose radius	r
Cone angle	θ_c
Cone base diameter	d_c
Frustum 1 angle	θ_1
Frustum 1 base diameter	d_1
Frustum 2 angle	θ_2
Frustum 2 height	h_2
Frustum 3 angle	θ_3
Frustum 3 height	h_3

Mass computation and Thermal protection system (TPS) settings are described in Section 3.3.4 as they are equivalent for all geometry subtypes.

3.3.4.5 Ellipsled

All required quantities to define the vehicle geometry are shown in Fig. 3.41 (the assumed flight direction is from the left to the right). The link to the labels in the configuration panel is provided in the subsequent list together with the default values.

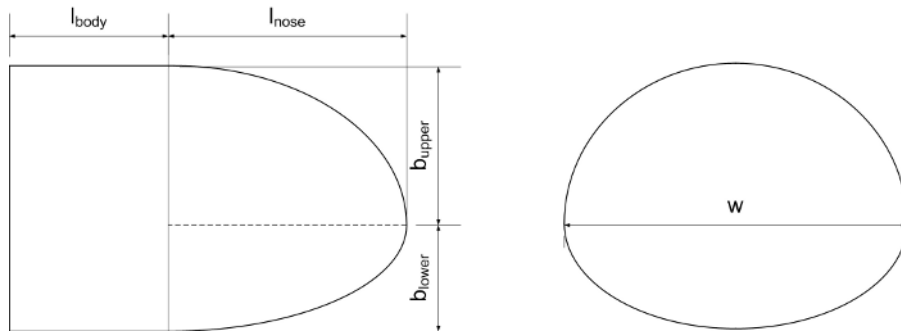


Fig. 3.41: Ellipsed geometry (side and front view)

Table 3.17: Geometry parameter definition

Label	Symbol
Width	w_c
Lower semi axis	b_{lower}
Upper semi axis	b_{upper}
Nose length	l_{nose}
Body length	l_{body}

Mass computation and Thermal protection system (TPS) settings are described in Section 3.3.4 as they are equivalent for all geometry subtypes.

3.3.5 User-Defined Component

Description

With a user-defined component the user is free in defining his own component with design parameters best suitable for simulation and optimization, for example design optimization. This component has a structural mass and a user-defined number of tanks and propellant masses. The computation of all corresponding properties is programmed in the user library. For more details see Chapter 11.

Note: A user-defined component cannot be inserted several times in the vehicle builder, for example in case of two boosters. In this case, please create one user-defined component for each booster.

Configuration

The following settings are required to interface with a user-defined aerodynamics:

- *Library file (DLL)* is the path and filename of the user-defined DLL. The maximum length is 2048 characters. The path may be relative to your problem directory (GTP folder).

- *User parameters* defines a string with a maximum length of 1028 characters. This string is provided to the initialization routine of the DLL as input parameter. It can be used in any way, e.g. a filename for loading data, data values itself or a combination of both.
- The generic settings *Mass distribution*, *Electrical power* and *Thermal characteristics* are explained in Section 3.1.2, Section 3.1.3 and Section 3.1.4 respectively. The latter two settings are available if the corresponding features in the *ASTOS* software have been installed and then enabled in the scenario *Active features*.

Background

For a detailed description of the interface please refer to chapter Chapter 11.

3.3.6 Deployable Boom

Description

A *Deployable Boom* component can be used to model any boom with a constant total mass and with varying length and mass distribution respectively over time. This can be an antenna or a protruding beam for example.

The deployable boom model consists of two parts:

- a fixed structure part with constant dimensions and varying mass
- a moving part with varying mass and length

Boom mass computation

The total mass of the *Deployable Boom* component is constant during the deployment and only the mass distribution between the fixed structure part and the moving boom part and the length of the moving part are changing. The mass of the fixed structure part is computed as the total component mass minus the mass of the moving part.

Boom length computation

The length of the fixed structure part is constant and is equal to the length of the moving part specified at initial time.

Center of mass and inertia computation

The center of mass (CoM) and the inertia of the total component is computed by considering the center of the mass and the inertia of the two model parts (fixed structure part and moving part).

Configuration

The configuration of the *Deployable Boom* component requires the following user settings:

- *Total mass* is the mass of the entire component and consists of the fixed structure part mass and of the moving part mass

VEHICLE PARTS AND PROPERTIES MODELS

- *Deployed mass* defines a profile of the moving part mass over time. For this purpose the *Defined by* setting must be set to *Profile* (for details see Profile Data Tables in the User Manual book). The mass of the fixed structure part is equal to the *Deployed mass* value at a certain time minus the *Total mass*.

Note: Each mass value in the profile table shall not exceed the *Total mass* value.

Note: The time entries specified in the *Deployed mass* and *Boom length* profile definitions must be identical.

- *Boom length* defines the profile of the moving part length over time. For this purpose the *Defined by* setting must be set to *Profile* (for details see Profile Data Tables in the User Manual book). The initial value for the length in the profile table (i.e. in the first row) is the length of the fixed structure part.

Note: The time entries specified in the *Deployed mass* and *Boom length* profile definitions must be identical.

- *Cross-sectional dimensions* define the lateral dimensions of the component. Both the fixed structure part and the moving part are assumed to have a cross section with a rectangular shape of the same dimensions
- The generic setting *Mass distribution* is explained in Section 3.1.2.
- The generic setting *Flexibility* is explained in Section 3.3.1.1 and is available if the *Multibody dynamics* feature is enabled in the list of *Active features* in the root node of the *Modelling* tree.

3.4 Sensor Models

Sensor models can be used to evaluate the visibility between vehicles, to observe targets, to transmit/receive data or to measure physical quantities. Depending on the anticipated application, different models are available providing basic or more advanced models. The sensor models described in the following sections are divided into following main types:

- *Camera* is an image sensor with a rectangular field of view (see Section 3.4.1)
- *Generic Sensor* is a sensor without any specific parameters (see Section 3.4.2)
- *Inertial Measurement Unit* is a sensor measuring linear acceleration and rotational rate (see Section 3.4.3)
- *Magnetometer* is a sensor which measures the magnetic flux density of an external magnetic field (see Section 3.4.4)
- *MBS Sensor* is used for multibody systems in order to generate additional output functions, see Section 3.4.5

VEHICLE PARTS AND PROPERTIES MODELS

- *Sun Sensor* defines a coarse sun sensor which detects the irradiation of sunlight (see Section 3.4.6)
- *Transmitter* models the generation and transmission of an electromagnetic signal containing encoded data (see Section 3.4.7)
- *Receiver* models a receiver and decoder of incoming electromagnetic signal (see Section 3.4.7)
- *Transceiver* models a sensor transmitting and receiving electromagnetic signals using the same antenna model. The user needs to specify all the settings only once (see Section 3.4.7)

Transmitters, *Receivers* and *Transceivers* are the only sensor types which can be used in a link budget analysis (see Section 5.10), since they are the only ones able to transmit and/or receive signals.

All sensor models are described in the following chapters. The common settings of the sensor models are explained in the following two paragraphs.

Configuration

The set up of a sensor is done in two different configuration panels:

- in the *Vehicle Parts & Properties* in the *Sensors & Transmitters* configuration panel the data (specific for each sensor and type-specific) has to be provided and is described in the next paragraphs
- in the *Dynamics Configuration* in the *Vehicles & POIs Dynamics* configuration panel, which requires the following settings (identical for any sensor type):
 - *Viewing Window* can be specified to limit the visibility of the sensor as *Value* or as *Profile* (minimum elevation angle as function of the azimuth angle).
 - *Orientation Offset* defines how the orientation of the sensor changes in time. For each axis the *Control law* must be defined here. The following options are available:
 - *Constant Law*, the orientation is fixed and does not change in time. The orientation angle can be provided or adopted from the previous phase.
 - *Linear Law*, the orientation changes linearly in time. The initial orientation angle and the final value or the angular rate must be provided.
 - *Profile Law*, the orientation angle control law can be provided by using a profile data table object (see Profile Data Tables in User Manual).
 - *Target Pointing*, the sensor algorithm minimizes the alignment deviation between the *Pointing axis* and the selected *Target object*. Depending on the control laws defined for the other axis, the requested target pointing might not be achieved completely.

Background

VEHICLE PARTS AND PROPERTIES MODELS

In the *Viewing Window* section of the *Dynamics Configuration -> Vehicles & POIs Dynamics* configuration panel, a *Viewing Window* can be specified in order to limit the visibility of the sensor. If the *Viewing Window* is enabled and specified as *Value*, the minimum elevation angle (which is the supplementary angle of the declination) of the field of view is set. Any object which is below the minimum elevation angle it is not seen by the sensor. Sometimes the field of view of a sensor might have an obstacle (e.g. building, tree, mountain, etc.) in a certain direction (azimuth angle in the y-z plane). In this case, the user can implement this in the scenario by selecting *Profile* in the *Defined by* selector in order to specify a profile data table (see Profile Data Tables in User Manual) corresponding to the obstacle.

The difference between the *Viewing Window* and the *Cone Angle* (available for *Transmitters*, *Receivers* and *Transceivers*, see description in Section 3.4.7) or the *Horizontal and Vertical Field of View* (available for *Cameras*) is that the first is fixed w.r.t. the sensor, while the others are moving together with the sensor (they are fixed in the sensor reference frame). Therefore, the *Viewing Window* should be used for "natural" obstacles, while the others should be used to reduce the field of view due to limitations of the device or due to the presence of other components.

This difference can be explained also by defining two different frames:

- the nominal frame, is the one defined in *Vehicles & POIs Definition* and it is fixed w.r.t. vehicle;
- the sensor frame, is the one defined in *Dynamics Configuration -> Vehicles & POIs Dynamics* and it is moving with the sensor (the x-axis is along the boresight vector).

The *Viewing Window* is defined in the nominal frame, while the *Cone Angle* is defined in the sensor frame. The two frames are exactly the same frame in case no *Orientation Offset* is defined.

Note: Since the *Viewing Window* is defined in the nominal frame, this frame should be properly defined in *Vehicles & POIs Definition*. The azimuth angle is always defined in the y-z plane, while the elevation angle (viewing window angle) is the out-of-plane angle. In case of ground stations, the nominal frame should be kept as default (all angles set to 0.0°, which makes the antenna point to zenith/x-axis). If a different pointing direction is needed, the use of a pointing law is recommended, since it modifies only the boresight orientation and not the frame.

3.4.1 Camera Model

Description

The *Camera* model gives the user the possibility to model an image sensor with a rectangular field of view.

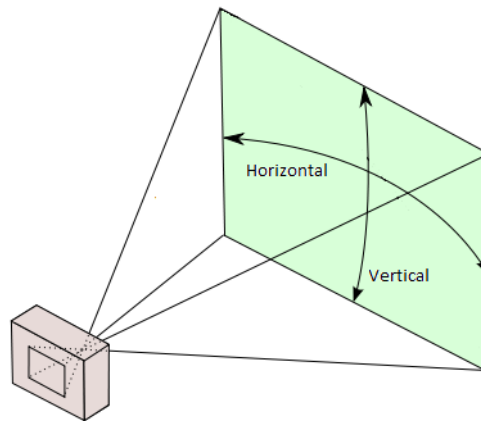


Fig. 3.42: Camera field of view

Configuration

The camera model configuration panel provides the following settings:

- *Horizontal field of view* sets the sensor's viewing angle in the x-y plane of the sensor's local reference frame.
- *Vertical field of view* sets the sensor's viewing angle in the x-z plane of the sensor's local reference frame.
- *Horizontal resolution* defines the resolution in the x-y plane of the sensor's local reference frame. The resolution affects the data size of each picture/frame.
- *Vertical resolution* defines the resolution in the x-z plane of the sensor's local reference frame. It affects the data size of each picture/frame.
- The *Mode* of a camera determines its operating condition, which can be one of the following:
 - In the *Video stream* mode the camera is continuously active and recording images
 - *Compression rate* sets the compression rate of the camera
 - *Frames per second* sets the camera frame per second rate
 - *Bits per pixel* sets the bit per pixel used by the camera
 - In the *Individual images* mode the taking of pictures is simulated at specific time instances. The following sub-settings are available:
 - *Compression rate* sets the compression rate of the camera
 - *Bits per pixel* sets the bit per pixel used by the camera
 - The *Time interval trigger* pane configures when the camera shall take a picture:
 - The *Common - Individual* setting is explained in detail in Section 3.1.1.

VEHICLE PARTS AND PROPERTIES MODELS

- *Trigger time interval* determines the frequency at which the camera is operated. For example, if one hour is specified, one picture per hour will be taken.

The coverage analysis calculates a visibility value for the camera only when the camera is active (e.g. it is recording or it is taking a picture). In the *Video stream* mode the visibility is computed for all time steps which overlap with the time interval during which the camera is recording a video. In *Individual images* mode the visibility is computed only for the time steps during which the camera takes a picture.

- The *Mass* defines an optional structural mass for this sensor. By default a sensor has no mass.
- The generic settings *Mass distribution*, *Electrical power* and *Thermal characteristics* are explained in Section 3.1.2, Section 3.1.3 and Section 3.1.4 respectively. The latter two settings are available if the corresponding features in the *ASTOS* software have been installed and then enabled in the scenario *Active features*.

Configuration

The *Horizontal resolution* and the *Vertical resolution*, together with the *Compression rate*, *Frames per second* and *Bits per pixel* are used to compute the camera data transfer rate.

3.4.2 Generic Sensor Model

Description

The *Generic Sensor* model does not have any specific parameters. It allows the user to rapidly create a sensor to evaluate the visibility of a target without having to provide a detailed model.

Configuration

The configuration panel provides the following settings:

- The *Mass* defines an optional structural mass for this sensor. By default a sensor has no mass.
- The generic settings *Mass distribution*, *Electrical power* and *Thermal characteristics* are explained in Section 3.1.2, Section 3.1.3 and Section 3.1.4 respectively. The latter two settings are available if the corresponding features in the *ASTOS* software have been installed and then enabled in the scenario *Active features*.

This type of sensor features an unlimited field of view, which can be limited by defining a *Viewing Window*.

3.4.3 Inertial Measurement Unit Model

Description

The *Inertial Measurement Unit* (IMU) model allows the user to rapidly create a sensor model that measures its linear acceleration, rotational rate and angles as done by accelerometers and gyroscopes. This model does not have any specific parameters and provides the measured quantities along each principal axis of the sensor as output functions.

Configuration

The configuration panel provides the following settings:

- The *Mass* defines an optional structural mass for this sensor. By default a sensor has no mass.
- The generic settings *Mass distribution*, *Electrical power* and *Thermal characteristics* are explained in Section 3.1.2, Section 3.1.3 and Section 3.1.4 respectively. The latter two settings are available if the corresponding features in the *ASTOS* software have been installed and then enabled in the scenario *Active features*.

Output

An *Inertial Measurement Unit* model provides the following specific output functions:

- Accelerations as measured by accelerometers along the principal axes of the sensor, represented in the sensor frame.
- The angular velocity vector components of the sensor w.r.t. inertial space represented in the sensor frame: they correspond to the measurements of a gyroscope.
- The quaternions of the sensor w.r.t. to its initial orientation.

3.4.4 Magnetometer Model

Description

The *Magnetometer* model provides the magnetic flux density vector at the position where magnetometer is mounted as simulation output, one *Auxiliary function* for each vector component. In case it mounted in a vehicle and there is a residual magnetic dipole of the vehicle or/and a *Magnetorquer* onboard the vehicle, the latter two are also considered in the measurement of the external magnetic field.

Configuration

The Configuration panel of a *Magnetometer* provides the following options:

- *Axes type* is a drop down menu to choose the number of axes along which the external field is measured. Two sub-settings are available:

- In case of *Uniaxial* only the magnetic flux density along the local sensor x-axis is measured. This results in one *Auxiliary function* as simulation output.
- In case of *Triaxial* the magnetic flux density along all three local sensor axes is measured. This results in three *Auxiliary functions* as simulation output.
- *Mass* defines an optional structural mass for this sensor. By default a sensor has no mass.
- The generic settings *Mass distribution*, *Electrical power* and *Thermal characteristics* are explained in Section 3.1.2, Section 3.1.3 and Section 3.1.4 respectively. The latter two settings are available if the corresponding features in the *ASTOS* software have been installed and then enabled in the scenario *Active features*.

For a general description about Profiles see Profile Data Tables in User Manual.

Background

For information about the magnetic field models used in *ASTOS*, please refer to Section 2.6. The measurements of a *Magnetometer* can differ from the values of the magnetic field of the central body in case there is interference with other magnetic dipoles within the vehicle. Two possible sources for interference are considered:

- *Magnetic Moment* of the vehicle.
- *Magnetorquer* which is active at the moment or it was previously active and it has a remnant magnetic field due to the hysteresis effect of the *Ferromagnetic Core* (see Section 3.2.4.2).

Eq. 3.130 depicts how one additional dipole moment affects the measurements of the magnetic field. In case multiple additional dipole moments superimpose on the external field, then they are added as additional terms to the formula.

$$\vec{B}_{measured} = \vec{B}_{field} + \frac{\mu_0}{4\pi|\vec{r}|^2} \cdot (3\vec{r}(\vec{m} \cdot \vec{r}) - \vec{m} \cdot |\vec{r}|^2) \cdot \frac{1}{|\vec{r}|^3} \quad (3.130)$$

with

- $B_{measured}$ being the magnetic flux density vector actually measured by the *Magnetometer* model
- B_{field} being the magnetic flux density vector of the magnetic field of the central body
- μ_0 being the vacuum permeability
- r being the position vector of the interfering magnetic dipole with respect to the *Magnetometer* and
- m being the magnetic dipole moment vector of one additional source.

3.4.5 MBS Sensor Model

The *MBS Sensor* model allows that the motion characteristics (e.g. position, translational and rotational rates and accelerations) at the location of the sensor model within a multibody vehicle can be utilized by the user. The model output can be considered as the data created through measurements. Thus this model can be used to create output of the inertial acceleration of a component, i.e. from an inertial measurement unit (IMU).

Configuration

In the configuration panel of the *MBS Sensor* model the provided output quantities and the generic sensor parameters can be set up. These are as follows:

- *Relative measurements* contains all model-related output w.r.t. a specified reference position:
 - *Reference position* defines the reference point for the measurements and can be one of the following:
 - *Inertial* means that the measurement is performed w.r.t. inertial space (i.e. center body)
 - *Parent node* means that the measurement is w.r.t. the reference node of the parent element of the sensor (i.e. the element at which the sensor is mounted in the vehicle builder, see The Vehicle Builder in the User Manual)
 - *Anchor node* means that the measurement is w.r.t. the anchor node of the parent element of the sensor
 - *Position*, if *Enabled*, causes that the position of the sensor w.r.t. the *Reference position* is provided as output
 - *Velocity*, if *Enabled*, causes that the velocity of the sensor w.r.t. the *Reference position* is provided as output
 - *Line of sight*, if *Enabled*, causes that the angle of the sensor w.r.t. the *Reference position* is provided as output
- *Inertial measurements* contains all model-related output w.r.t. inertial space:
 - *Acceleration*, if *Enabled*, causes that the acceleration of the sensor is provided as output
 - *Angular rate*, if *Enabled*, causes that the angular rate of the sensor is provided as output
 - *Angular acceleration*, if *Enabled*, causes that angular acceleration of the sensor is provided as output
- *Mass* is a setting for the mass of the sensor. The default value is *Disabled* which results in a sensor of zero mass.
- The generic settings *Mass distribution*, *Electrical power*, *Thermal characteristics* and *Error modelling* are explained in Section 3.1.2, Section 3.1.3, Section 3.1.4 and Chapter 8

respectively. The latter three settings are available if the corresponding features in the ASTOS software have been installed and then enabled in the scenario *Active features*.

Background

For each selected measurement, all three components of the vector of the corresponding quantity are provided as an output function. The coordinate frame is the one of the specified *Reference position*, or the inertial frame in case of *Inertial measurements*.

3.4.6 Sun Sensor Model

Description

The coarse sun sensor detects the intensity of incoming light and generates an output signal as either electrical current or voltage depending on the configuration. In addition to the sunlight the albedo of Earth can be considered as well.

Configuration

The configuration panel provides the following settings:

- *Half-cone angle* defines the size of the circular field of view.
- *Reference current* is the electrical current produced in a given reference condition. This quantity is available in the product data sheets of manufactures.
- *Reference irradiance* is the intensity of incoming light per unit area required to generate the *Reference current*. This value is also available in the product data sheets of manufactures.
- *Shunt resistance* defines an optional electrical resistance in the sensor and changes the output signal from current (*disabled*) to voltage (*enabled*).
- *Mass* defines an optional mass for the sensor. By default sensors have no mass.
- *Consider albedo* sets a flag whether the albedo of the central body shall be considered as light source or not.
- The generic settings *Mass distribution*, *Electrical power* and *Thermal characteristics* are explained in Section 3.1.2, Section 3.1.3 and Section 3.1.4 respectively. The latter two settings are available if the corresponding features in the ASTOS software have been installed and then enabled in the scenario *Active features*.

Background

The current produced in the sun sensor is calculated according to the following formula:

$$I = \frac{\lambda}{\lambda_{STC}} \cdot I_{STC} \quad (3.131)$$

with I being the generated current, λ the actual irradiance, λ_{STC} the reference irradiance, I_{STC} the reference current. In case the *Shunt resistance* is defined the current I is multiplied by the resistance R to obtain the voltage output signal.

The calculation of the actual irradiance is explained in Section 3.1.5.

3.4.7 Transmitter, Receiver and Transceiver Models

Description

Transmitter, receiver and transceiver models can be of the same sensor subtype. An overview of the available sensor subtypes is provided in the following and a more detailed description thereof is given in the following sections:

- *Simple* sensor is a simplified model with basic functionalities (see Section 3.4.7.1)
- *Isotropic* sensor is a model of an isotropic antenna (see Section 3.4.7.2)
- *Complex* sensors model the generation and transmission of an electromagnetic wave with encoded data (see Section 3.4.7.3)
- *GNSS* sensors are meant to be used together with the navigation analysis (see Section 5.12) in order to estimate the accuracy that can be achieved for geo-spatial positioning (see Section 3.4.7.4)

Configuration

Before explaining the differences between transmitters and receivers, the aspects regarding the electromagnetic wave, which is produced or received by the sensor, are discussed. The wave characteristics are independent of the sensor types and are relevant to any of them. These signal characteristics are:

- *Frequency* represents the central frequency of the transmitting/receiving signal bandwidth.
- *Bandwidth* defines the difference between the upper and lower frequencies in a continuous set of frequencies. The *Auto-scale* checkbox is available in the configuration panel; if it is selected for a receiver, the bandwidth of the receiver is assumed to be exactly the same as the one of each incoming signal, while if selected for a transmitter, the minimum bandwidth required for the selected data-rate and modulation is set.
- *Modulation* defines the technique used by a sensor to modulate or demodulate the signal. If the *Auto-tracking* checkbox within this setting is selected, the modulation of the receiver is assumed to be exactly the same as the one of each incoming signal.
- *Polarization* defines the polarization technique used by a sensor. The *Reference axis* has to be provided only for not circular polarizations, while the *Tilt angle* has to be provided only for *Linear* and *Elliptical* polarizations describing the plane of propagation of the wave. If the *Elliptical* type is chosen, an *Axial ratio* has to be specified in order to define the shape of the ellipse. If the X-axis is selected as *Reference axis*, positive tilt is measured from x to y in the x-y plane. If Y-axis is selected as *Reference axis*, positive tilt is measured from y to -x in the x-y plane. If Z axis is selected as *Reference axis*, positive tilt is measured from z to -y in the x-y plane.
- *Half-cone angle* defines the cone which is assumed as the sensor field of view.

The settings which are available only for transmitters are the following:

VEHICLE PARTS AND PROPERTIES MODELS

- *EIRP* or *Power* represents the transmitter power which would be required if the antenna was an ideal isotropic radiator connected to the transmitter without losses. The EIRP can be provided directly by the user (for isotropic antennas, see Section 3.4.7.2) or can be computed knowing the transmitter power and the antenna physical properties.
- *Maximum data rate* is the maximum rate, in bits per second, at which binary information can be transferred by a transmitter.
- *Post-transmit gains & losses* allow the user to consider constant gains or losses in a link budget analysis (see Section 5.10), which took place right after the signal leaves the transmitter. They are often needed to model gains and/or losses which affect performance but were not accounted for by the other models.

The components which are receivers-specific are the following:

- *Gain to noise temperature* (available only in the *Complex* receiver model) is the figure of merit of any receiver and can be either provided by the user or is computed by ASTOS as a function of the *System noise*.
- *Pre-receive gains & losses* can be used to account for gains and losses before the signal is processed by the receiver.
- *Pre-demodulation gains & losses* can be used to account for gains and losses before the signal is demodulated by the receiver.

The other settings available for all models are:

- The *Mass* defines an optional structural mass for this sensor. By default a sensor has no mass.
- The generic settings *Mass distribution*, *Electrical power* and *Thermal characteristics* are explained in Section 3.1.2, Section 3.1.3 and Section 3.1.4 respectively. The latter two settings are available if the corresponding features in the ASTOS software have been installed and enabled in the scenario *Active features*.

Background

Frequency describes the number of waves that pass a fixed place in a given amount of time and is equal to the velocity of propagation of the wave divided by its wavelength.

Bandwidth is the difference between the upper and lower frequencies in a continuous set of frequencies. A key characteristic of bandwidth is that any band of a given width can carry the same amount of information, regardless of where that band is located in the frequency spectrum.

The bandwidth of the receiver specifies the range of frequencies over which its performance does not suffer due to a poor impedance match. Therefore, the bandwidth of the receiver should overlap the bandwidth of the incoming signal. If a mismatch occurs, either a loss of data or of the quality of the link takes place. For this reason, the *Auto-scale* checkbox is available in the receiver configuration panel close to the *Bandwidth* field. If it is selected, the bandwidth of the receiver is assumed to be exactly the same as the one of each incoming signal. Even though it is not a realistic assumption, it has been added for the sake of convenience.

VEHICLE PARTS AND PROPERTIES MODELS

Depending on the data-rate and on the modulation type, the minimum bandwidth required for the data transmission might be computed. For this reason, the *Auto-scale* checkbox is available also for transmitters.

Modulation is the process of varying one or more properties of a periodic waveform, called the carrier signal, with a modulating signal which typically contains information to be transmitted. This is done in a similar fashion to a musician modulating a tone (a periodic waveform) from a musical instrument by varying its volume, timing and pitch. The three key parameters of a periodic waveform are its amplitude (volume), its phase (timing) and its frequency (pitch). Any of these properties can be modified to obtain the modulated signal. Typically a high-frequency sinusoid waveform is used as carrier signal. In telecommunications, modulation is the process of conveying a message signal inside another signal.

The BER computation performed by the link budget analysis (see Section 5.10 for details) is function of the Signal-to-Noise ratio as well as of the modulation scheme. More details about the modulation schemes and their role in the BER computation are provided in [85].

It is important to note that a receiver should support the same modulation scheme as the incoming signal in order to be able to decode and use it. The SNR and the C/N_0 in the link budget analysis are set respectively to -200.0 dB and -200.0 dB/Hz if the signal modulation scheme is not supported by the receiver. Also in this case an *Auto-tracking* checkbox is available in the receiver configuration panel close to the *Modulation* field. If it is selected, the modulation of the receiver is assumed to be exactly the same as the one of each incoming signal.

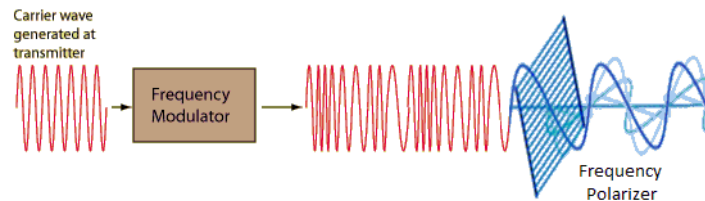


Fig. 3.43: Signal modulation and polarization

Polarization of an electromagnetic wave is the orientation of its electric field w.r.t. the Earth's surface and is determined by the combined effects of the antenna physical properties and orientation. It has nothing to do with the antenna shape. Thus, a simple straight wire antenna has one polarization when mounted vertically, and a different polarization when mounted horizontally. A reflection of the electromagnetic waves can cause a change of polarization. The most important reflector is the ionosphere: reflected signals have their polarization changed unpredictably. At the moment, changes of polarization due to atmospheric effects are not considered in ASTOS. In order to be able to use the incoming signal, the polarization of the receiver has to be compatible with the polarization of the incoming signal. If the two polarizations are of the same type, then there is no loss in the transmission. On the other hand, if the two types are different, signal degradation might occur.

If the transmitting and receiving antennas for one signal in an ASTOS scenario do not support the same type of polarization, a signal loss model is applied and the received signal quality by the receiver antenna is strongly reduced. Signal degradation due to mismatch of polarization (or of bandwidth or of modulation) is explained also in . In ASTOS the following polarization types are supported: *Linear*, *Vertical*, *Horizontal*, *Elliptical*, *Right-hand circular* and *Left-hand circular*.

VEHICLE PARTS AND PROPERTIES MODELS

As already mentioned in Section 3.4, the *Half-cone angle* defines the field of view of a sensor which is defined as the cone delimiting the area "seen" by the sensor. The value required by the *Half-cone angle* field is the semi-cone angle of the field of view (e.g. for a sensor mounted on a *Ground station* a half-cone angle of 90° allows the sensor to observe the full sky, when pointing to the zenith, while an angle of 180° allows always the full coverage of the sky). Another important aspect is that the field of view moves together with the sensor, since it is defined in its reference frame, while the *Viewing window* does not "follow" the sensor.

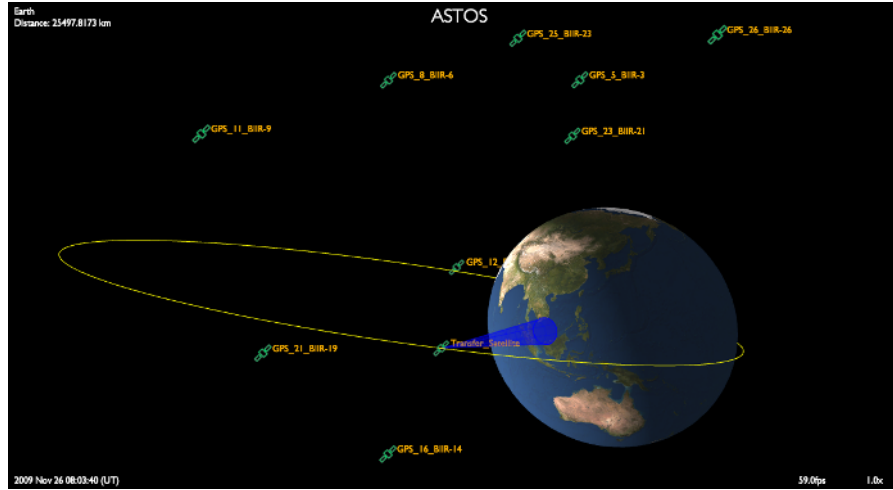


Fig. 3.44: Astroview screenshot showing a sensor cone angle (in blue) pointed to the Earth

Transmitters are electronic devices that generate and amplify a carrier wave, modulate it with a meaningful signal and radiate it from an antenna. They are complex systems which include several components (see Fig. 3.45). The carrier signal is generated by an oscillator and then goes through a modulator, which conveys a message (data) inside the signal that can be physically transmitted. Then the signal power is amplified by the amplifier and the antenna. Once the signal leaves the transmitter, predefined constant post-transmission gains/losses might be considered.

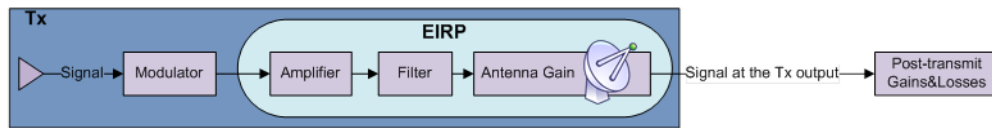


Fig. 3.45: Transmitter components diagram

Models for filters and amplifiers are currently not implemented in ASTOS.

The *EIRP* (Equivalent Isotropically Radiated Power) represents the transmitter power which would be required if the antenna was an ideal isotropic radiator connected to the transmitter without losses. The EIRP is a figure of merit of the transmitter and denotes the power density or field intensity radiated by it. The EIRP in dBW is expressed as a sum of the antenna gain G_T in dBi and the transmitted power P in W fed to the antenna [92]:

$$EIRP = 10 \log(P) + G_T \quad (3.132)$$

VEHICLE PARTS AND PROPERTIES MODELS

The EIRP can be provided directly by the user (for isotropic antennas, see Section 3.4.7.2) or can be computed by ASTOS if the transmitter power and the antenna physical properties (see below) are known.

Receivers are electronic devices that receive incoming electromagnetic signals and convert them to usable form. The antenna intercepts and converts them to alternating currents. Predefined constant pre-receive gains/losses might be considered. The receiver uses electronic filters to separate the desired frequency signal from all the other signals received by the antenna, an amplifier to increase the power of the signal for further processing, and finally recovers the desired information through demodulation. The structure of a receiver can be decomposed into separate functional blocks (see Fig. 3.46).

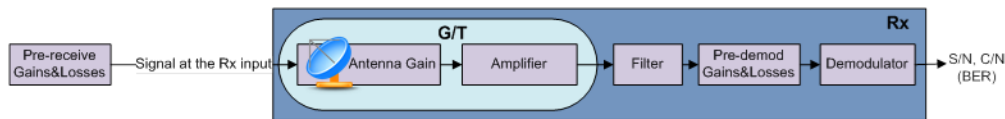


Fig. 3.46: Receiver components diagram

The *Gain to noise temperature* (G/T) is the figure of merit of any receiver and is expressed in dB/K. It is the ratio of the receiver gain and the system noise:

$$\frac{G}{T} = G_R - 10\log(T_{Sys}) \quad (3.133)$$

where G_R is the receiver antenna gain in dBi (decibels-isotropic) and T_{Sys} is the *System noise temperature* in K. The G_R can be provided directly by the user (for isotropic antennas, see Section 3.4.7.2) or can be computed from the antenna characteristics (as explained in the next paragraphs).

It has become common (see [93]) to represent the noise power received by an antenna from permanent external radiating sources by an effective *Antenna noise temperature* T_a . The noise from these permanent sources (temporary or discontinuous sources of noise, man-made noise are not considered) is similar in character to thermal noise, and can be combined with the internal receiver noise in a simple additive manner. The antenna noise temperature is a fictitious absolute temperature (as T_{Sys}).

Generally the *System noise temperature* may be considered as the sum of three components (as shown in Fig. 3.47):

- Contribution due to reception of noise from external radiating sources T_a
- Thermal noise generated due to dissipative losses in the receiving transmission-line system T_r
- Noise from sources internal to the receiver itself T_e

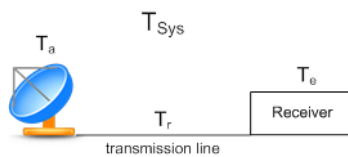


Fig. 3.47: Antenna, transmission line and receiver for system temperature computation

VEHICLE PARTS AND PROPERTIES MODELS

The transmission line power loss factor L_r of the transmission-line system plays an important role in the system noise-temperature calculation. First, this loss reduces the amount of antenna noise power reaching the receiver input terminals. Second, the magnitude of the transmission-line loss factor directly affects the transmission-line noise temperature, T_r . The power loss factor is defined as the ratio of power input to power output of the lossy device or component. The system noise temperature can be computed with the following formulas:

$$T_{Sys} = \frac{T_a}{L_r} + T_r + T_e; \quad T_r = T_t \left(1 - \frac{1}{L_r}\right); \quad T_e = \left(10^{\frac{NF}{10}} - 1\right) T_0 \quad (3.134)$$

where where NF is the noise factor in dB, T_t is the thermal temperature of the transmission line and T_0 is the reference temperature (290 K). NF , T_t , and T_0 are usually system specific and have to be provided by the user in the *System noise* section of the sensor configuration panel. The *System noise temperature* can be directly provided by the user or computed from the physical characteristics of the receiver. In the same way the computation of the *Antenna noise temperature* can be also left to *ASTOS* and depends on the position and orientation of the sensor. Due to its complexity, this computation is not discussed here, more details can be found in [85].

Transceivers implement both of the components (already explained in the previous two sections) required for transmitting and receiving electromagnetic signals. Therefore, they can be both active or passive in the link budget computation.

3.4.7.1 Simple

Description

Simple transmitters and *Simple receivers* are simplified objects with basic functionalities. Therefore, they should not be used for detailed link budget analyses, but only if the user is interested in basic information as number of links, link duration, data link windows, data volume etc.

Configuration

No details on the signal characteristics have to be provided and for this reason also the characteristics of the antenna are not required. There is only one additional setting in this model (except for the common ones explained in Section 3.4.7):

- *Maximum data rate* (only for transmitters)

3.4.7.2 Isotropic

Description

VEHICLE PARTS AND PROPERTIES MODELS

The *Isotropic* subtype allows the user to create a sensor mounting an isotropic antenna. The user needs to specify directly the EIRP and the G/T without going into further antenna model details.

Configuration

The configuration panel requires the following settings (already described in Section 3.4.7):

- *EIRP* (only for transmitters)
- *Maximum data rate* (only for transmitters)
- *Gain to noise temperature* (only for receivers)
- *Frequency*
- *Bandwidth*
- *Modulation*
- *Polarization*
- *Post-transmit gains & losses* (only for transmitters)
- *Pre-receive gains & losses* (only for receivers)
- *Pre-demodulation gains & losses* (only for receivers)

For isotropic antennas the gain is considered to be constant in all the directions (see Fig. 3.48), which is an ideal antenna model. In reality no antenna is perfectly omnidirectional.

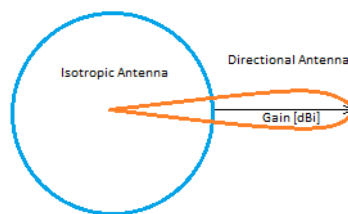


Fig. 3.48: Isotropic vs. directional antenna gain

3.4.7.3 Complex

Description

Complex transmitters and *Complex receivers* allow the user to further customize a sensor. All the components described in section Section 3.4.7 are available here. In addition several antenna models are available:

- *Parabolic*
- *Horn*
- *Dipole*
- *Axial Helix*

■ *User-Defined*

The main component of a transmitter or of a receiver is the antenna. What differentiates an antenna model from another is the different geometry and physical properties and how the antenna gain is computed. Therefore, depending on the antenna type a different set of input is required. For complex sensors the gain strongly depends on the **off-boresight angle**, which is the angle between the direction of maximum gain (usually pointing direction) and the direction where the other antenna is. The maximum gain is obtained when transmitter and receiver "look" at each other (null off-boresight angles), while the worst situation is achieved when the two antennas point in opposite directions (off-boresight angles equal to 180°).

Configuration

The following settings are required for all complex sensors:

- *Maximum data rate* (only for transmitters)
- *Frequency*
- *Bandwidth*
- *Modulation*
- *Polarization*
- *Post-transmit gains & losses* (only for transmitters)
- *Pre-receive gains & losses* (only for receivers)
- *Pre-demodulation gains & losses* (only for receivers)
- *Antenna*, whose specific settings depend on the selected *Model* and are listed and explained for each type below.

Background

Parabolic antennas are widely used for space applications due to their high directivity. Basically almost all the power is irradiated in the same direction (narrow beamwidth). In order to define the dimension/performance of a parabolic antenna three different parameters can be specified:

- *Diameter* d
- *Beamwidth* ϕ_{3dB}
- *Maximum gain* G_{Max}

Once the user chooses which of those parameters to use, the other two are automatically computed using the following relations [94]:

$$\phi_{3dB} \cong 70 \frac{\lambda}{d} \eta; \quad G_{Max} = \eta \left(\pi \frac{d}{\lambda} \right)^2 \quad (3.135)$$

The antenna (aperture) *efficiency* η accounts for all causes of the reduction of the gain. It is usually expressed as a percentage and typical values are 0.55-0.75. Several types of loss are considered [94]:

- Illumination efficiency which is the ratio of the directivity of the antenna to the directivity of a uniformly illuminated antenna with the same aperture size
- Phase error loss, cause by the fact that the aperture is not a uniform phase surface
- Spillover loss (in case of reflector antenna) which reflects the energy spilling beyond the edge of the reflector into the back lobes of the antenna
- Mismatch loss, derived from the reflection at the feed port due to impedance mismatch (especially relevant for low frequency antennas)
- Loss between the antenna and the feed port

The **antenna gain** for a parabolic antenna is defined as function of the off-boresight angle (ϕ):

$$G(\phi) = G_{Max} \left(\frac{2J_1(x)}{x} \right)^2; \quad x = \pi \frac{d}{\lambda} \sin \phi \quad (3.136)$$

where J_1 is the first order Bessel function and λ is the wavelength (speed of light divided by the frequency, as in Eq. 3.135). Additionally the user can provide the *backlobe gain* instead of computing it from the previous relation whenever the angle off-boresight is between $180^\circ \pm \phi_{3dB}/2$.

The *Horn* antenna consists of a flaring metal waveguide shaped like a horn to direct the electromagnetic waves in a beam. An advantage of this type of antennas is that they do not have resonant elements. Therefore, they can operate over a wide bandwidth. For the horn antenna, the same set of parameters as for the parabolic antenna has to be provided. The only difference is that d is not anymore a diameter but the *characteristic dimension* of the horn. For the **antenna gain** computation the following equations apply:

$$G(\phi) = G_{Max} \left(\frac{\sin x}{x} \right)^2; \quad x = \pi \frac{d}{\lambda} \sin \phi; \quad G_{Max} = 4\pi \eta \left(\frac{d}{\lambda} \right)^2 \quad (3.137)$$

The *Dipole* antenna consists of two identical conductive elements such as metal wires or rods, which are usually bilaterally symmetrical. Each element is connected to one of the conductors. This differentiates it from a common monopole antenna, which has just a single rod. The boresight orientation is assumed to be along the z-axis (along the conductive elements). Therefore, this antenna has a null gain for elevation equal to 0° . Considering a dipole antenna mounted on a ground station, the elevation of the antenna should be set to 0° (conductive elements parallel to the ground) in order to have signal emitted towards the sky. For the dipole antenna model, only two parameters have to be provided: the *characteristic length* L and the *efficiency* η . No backlobe information is required, since the antenna pattern is symmetric in the azimuth plane. For the dipole antenna, the flux density F and the directivity D are used in order to compute the gain as function of the off-boresight angle. They can be computed by using [95] and [96]:

$$F(\theta) = \left[\frac{\cos\left(\frac{k_\lambda L}{2} \cos(90^\circ - \theta)\right) - \cos\left(\frac{k_\lambda L}{2}\right)}{\sin(90^\circ - \theta)} \right]; \quad D(\theta) = \frac{2F(\theta)}{\int_0^\pi F(\theta) \sin \theta d\theta} \quad (3.138)$$

where $k_\lambda = 2\pi/\lambda$ is the wavenumber, θ is the elevation angle, η is the *efficiency* and λ is the wavelength. The integral can be discretized in 180 steps, while the **antenna gain** is computed by means of the following equation:

$$G(\theta) = \eta D(\theta) \quad (3.139)$$

An *Axial Helix* antenna is an antenna which has a conducting wire wound in the form of a helix. Helical antennas are mounted usually over a ground plane. The feed line is connected between the bottom of the helix and the ground plane. The parameters of the helix antenna are defined below ([97]):

- d is the *diameter* of a turn
- C is the circumference of a turn $C = \pi * d$
- S is the *turns separation* (vertical separation between turns)
- α is the pitch angle, which defines how far the antenna grows in the z-axis per turn $\alpha = \tan^{-1}(S/C)$;
- N is the total *number of turns*
- H is the total height $H = NS$

The normalized flux density and the directivity are given as function of the off-boresight angle ϕ by [97]:

$$\xi = 2\pi \left[\frac{S}{\lambda} (1 - \cos\phi) + \frac{1}{2N} \right]; \quad F(\phi) = \left[\sin\left(\frac{\pi}{2N}\right) \cos\phi \frac{\sin\frac{N}{2}\xi}{\sin\frac{\xi}{2}} \right]^2; \quad D(\phi) \approx 15 \frac{NC^2S}{\lambda^3} F(\phi) \quad (3.140)$$

The **antenna gain** is computed as in Eq. 3.139. These relations work well when the following conditions are fulfilled:

$$\frac{3}{4}\lambda \leq C \leq \frac{4}{3}\lambda; \quad 12^\circ \leq \alpha \leq 15^\circ; \quad 3 \leq N \leq 100 \quad (3.141)$$

The *User-Defined* antenna allows the user to fully customize the antenna gain pattern. In this way the design of an antenna, besides the standard types already discussed, is possible. The user must define the gain of the antenna by means of a profile data table object, (see Profile Data Tables in User Manual). The independent variables are the off-boresight angle components (in-plane and out-of-plane) or the total off-boresight angle. The user can decide if to provide directly the gain of the antenna (in dB) or the normalized profile (as function of the aforementioned independent variables) together with the maximum gain.

3.4.7.4 GNSS

Description

A satellite navigation system is a system of satellites providing autonomous geo-spatial positioning with global coverage. It allows receivers to determine their location (longitude, latitude, and altitude) to within a few meters using time signals transmitted along a line-of-sight from satellites.

GNSS sensors are meant to be used together with the navigation analysis (see Section 5.12) in order to provide information about the accuracy that can be achieved and the quality of the relative geometry between receiver and transmitters. For this purpose the *complex* sensors

have been used and extended in order to provide to the user a set of GNSS sensors, still customizable but with ready built-in features required by the navigation analysis.

Background

Each GNSS signal consists of two or more components:

- **Carrier** (RF sinusoidal signal with a known nominal frequency);
- **Ranging codes** (e.g. C/A, P, etc).

Ranging codes are combined with binary **navigation data** and modulated (e.g. BPSK for GPS) on the carrier. The navigation data is a binary-coded message with data on the satellite's health status, ephemeris, clock bias parameters, constellation ephemeris almanac, etc. It is important noticing that carrier, ranging codes and navigation data generators are driven by a unique clock/oscillator to assure synchronization.

3.4.7.4.1 Transmitter

Description

A *GNSS Transmitter* allows the user to set up the transmitter of a GNSS satellites with few clicks. The user needs to specify the type of GNSS constellation *system* and the *block* the satellite belongs to. Then a list of available *channels* and *signals* is shown and the user needs to select the desired one. Once the signal has been chosen the antenna model (and indeed the antenna gain profile) is automatically set in order to match the real gain distribution of such transmitter.

Configuration

The user needs to select with signal the GNSS transmitter generated. The following settings are required:

- *System* defines which GNSS constellation to be used.
- *Block* defines the generation of satellite to be used, which defines the hardware available on board.
- *Channel* defines the frequency of the signal.
- *Signal* defines the data rate and the modulation.

Background

In the same way, depending on the selected signal, the proper frequency, bandwidth, polarization, modulation and data-rate (introduced in section Section 3.4.7) are set. These default values are summarized in the next table (values taken from [98], [99], [100] and [101]):

Table 3.18: GNSS signal definition

VEHICLE PARTS AND PROPERTIES MODELS

System	Channel	Satellite Block	Signal	Frequency [MHz]	Data-Rate [Mbit/s]	Symbols/s	Modulation
GPS	L1	II-A	C/A	1575.42	1023.0	50	BPSK
GPS	L1	II-A	P	1575.42	10230.0	50	BPSK(10)
GPS	L1	II-RM	C	1575.42	1023.0	25	BOC(1,1)
GPS	L1	III	M	1575.42	5115.0	50?	BOC(10,5)
GPS	L2	II-A	P	1227.6	10230.0	50	BPSK(10)
GPS	L2	II-RM	C	1227.6	5115.0	50	BPSK(0.5)
GPS	L2	III	M	1227.6	5115.0	50?	BOC(10,5)
GPS	L5	II-F	SOL	1176.45	10230.0	100	BPSK(10)
GALILEO	E1	IOV	a	1575.42	2557.5	110	BOCcos(15,2.5)
GALILEO	E1	IOV	b/c	1575.42	1023.0	250	MBOC(1,1)
GALILEO	E5	IOV	a	1176.45	10230.0	50	QPSK(10)
GALILEO	E5	IOV	b	1207.14	10230.0	250	QPSK(10)
GALILEO	E5	IOV	a/b	1191.795	10230.0	250	AltBOC(10,10)
GALILEO	E6	IOV	a	1278.75	10230.0	110	BOCcos(10,1)
GALILEO	E5	IOV	b/c	1278.75	1023.0	1000	MBOC(1,1)
GLONASS	L1	M	OF	$1602.0 \pm n \cdot 0.5625$	5115.0	50	BPSK(0.5)
GLONASS	L1	M	SF	$1602.0 \pm n \cdot 0.5625$	51150.0	50	BPSK(5)
GLONASS	L1	K2	OC	1575.42	5115.0	50	BPSK(0.5)
GLONASS	L1	K2	SC	1575.42	51150.0	50	BPSK(5)
GLONASS	L2	M	OF	$1246.0 \pm n \cdot 0.4375$	5115.0	50	BPSK(0.5)
GLONASS	L2	M	SF	$1246.0 \pm n \cdot 0.4375$	511500.0	50	BPSK(5)
GLONASS	L2	K2	OC	1242.0	5115.0	50	BPSK(0.5)
GLONASS	L2	K2	SC	1242.0	51150.0	50	BPSK(5)
GLONASS	L3	K1	OC	1202.025	51150.0	50	BPSK(5)

The satellite *block* indicates the generation of satellites, the first implements a certain signal. All the following generations will carry all the previous signals for backwards compatibility. Many of the upcoming new signals are not fully described in the Interface Control Documents. Therefore, basic characteristics and even the name might change in the future.

The GLONASS signals L1-M and L2-M are the signals actually broadcasted by the satellites in space. They operate in a FDMA fashion. It means that "each" satellite transmits at a slightly different frequency (n goes from -7 to +6). The frequency reported in the previous table is just the nominal frequency. Since this approach requires receivers able to receive a wider bandwidth, which means higher complexity and more noise, in the GLONASS modernization process, CDMA signals will replace (or coexist with) the current FDMA signals.

VEHICLE PARTS AND PROPERTIES MODELS

The symbols/s defines the navigation data-rate (the data-rate of the navigation message) and is used in CDMA. The navigation data-rate for military signals (as GPS L1/M and L2/M) is not known. The same value of the respective civil signal has been assumed.

The notation adopted for modulations is the following:

- for BPSK and QPSK the number in brackets is the subcarrier frequency in multiples of 1.023 MHz;
- for the other modulations the first number in brackets is the same as before, while the second number is the data-rate in multiples of 1.023 Mbit/s.

In order to match the antenna radiation pattern found in literature ([102], [103] and [104]) for each GNSS transmitting antenna, the most fitting combination of antenna models has been chosen. Radiation patterns have been discretized using as independent variable the boresight angle. Along with those presented in section Section 3.4.7.3, also constant and linear patterns have been used. The models used for each signal has been summarized in the next table:

Table 3.19: GNSS transmitter antenna models

System	Channel	Satellite Block	Boresight Angle range [°]	Gain
GPS	L1	All	0.0-44.53	Axial (0.95, 10.0, 0.11 m, 0.035 m)
			44.53-60	Constant (-7.0 dB)
			60-70	Linear (-7.0 dB, -17.0 dB)
GPS	L2	II	0.0-41.4735	Axial (0.95, 10.0, 0.15 m, 0.048 m)
			41.4735-90	Constant (-17.0 dB)
GPS	L2	III	0.0-46.585	Axial (0.95, 10.0, 0.12 m, 0.064 m)
			46.585-90	Constant (-17.0 dB)
GPS	L5	II-F	0.0-46.745	Axial (0.95, 10.0, 0.15 m, 0.056 m)
			46.745-90	Constant (-17.0 dB)
GPS	L5	III	0.0-46.628	Axial (0.95, 10.0, 0.125 m, 0.0664 m)
			46.628-90	Constant (-17.0 dB)
GALILEO	E1	All	0.0-23.2	Axial (0.95, 10.0, 0.125 m, 0.04 m)
			23.2-90	Linear (-11.0 dB, -16.0 dB)
GALILEO	E5	All	0.0-44.01	Axial (0.95, 10.0, 0.14 m, 0.052 m)
			44.01-90	Linear (-17.0 dB, -24.0 dB)
GALILEO	E6	All	0.0-43.8	Axial (0.95, 10.0, 0.13 m, 0.047 m)
			43.8-90	Linear (-17.0 dB, -24.0 dB)
GLONASS	L1	All	0.0-24.15	Axial (0.95, 10.0, 0.105 m, 0.0216 m)
			24.15-90	Constant (-22.0 dB)
GLONASS	L2-L3	All	0.0-24.118	Axial (0.95, 10.0, 0.13 m, 0.028 m)
			24.118-90	Constant (-17.0 dB)

The axial antenna model has been described in Section 3.4.7.3. The values in brackets have the following meaning:

- Efficiency
- Number of turns
- Turns separation
- Turn diameter

For values of the boresight angle not covered by the previous table, the back-lobe gain (of 30 dB) is used. The half-cone angle is by default set to 180° (if not specified differently by the user), which means that theoretically the transmitter transmits in all the directions, but in certain directions the power radiated is so low that it cannot be used by any receiver.

3.4.7.4.2 Receiver

Description

A *GNSS Receiver* is a complex device which must fulfil the following tasks:

- Receive, pre-amplify and filter GNSS signals coming from satellites
- Generate pseudo-random codes for demodulation
- Extract GNSS code phase for range measuring
- Extract GNSS carrier phase for carrier phase measuring
- Demodulate the navigation message
- Solve the PVT equations in order to determine its position and velocity

The reception and handling of the incoming signal are common to all the receivers and have been tackled in the previous chapters. On the other hand, the solution of the PVT equations is performed only by GNSS receivers. This solution is described in Section 5.12, since it is related to the output of the *navigation analysis* and not so much on the definition of the receiver model. The other aspects are discussed here.

Several inputs are required to fully define the model of the GNSS receiver, which are described here.

Configuration

The configuration panel provides the following settings:

- *Signals* define which signals can be processed by the receiver. The user can specify multiple signals (e.g. multi-constellation or multi-frequency receiver) or only one.
- *Tracking* defines the performance of the receiver. The user needs to specify how the receiver performs the tracking of the incoming signals. The specific settings are explained in the next paragraph.

Background

VEHICLE PARTS AND PROPERTIES MODELS

The main functionalities of a navigation receiver rely on the capability to perfectly align the local sequence of the code generated by the receiver itself and the one coming from space. It is necessary to align the two sequences recovering the delay between them in order to minimize the range error.

Acquisition and *tracking* are the main tasks of a GNSS receiver [106]. The purpose of acquisition is to identify all satellites visible to the user. If a satellite is visible, the acquisition must determine the frequency of the incoming signal from a specific satellite (which differs from its nominal value, due to Doppler effect and atmospheric interferences) and the code phase, which denotes the point in the current data block where the code begins the fraction of the wave cycle which has elapsed relative to the origin). The code differs for each GNSS signal (in CDMA) and together with the frequency represents the ID of a signal. FDMA signals share the same code (e.g. GLONASS L1-M signal) and allow the identification of the transmitting satellite by using different frequencies. On the other hand, CDMA signals share the same frequency but implement different codes. Once the receiver has identified the signal, the frequency and code phase determination allow to "use" the incoming signal (for a description of GNSS signals see Section 3.4.7.4).

The user has to select in the GUI which *signals* can be received by the GNSS receiver. With this information the bandwidth of the receiver is set and the demodulation types are assigned. The antenna gain pattern is modelled using the Horn antenna model (see Section 3.4.7.3), assuming an efficiency of 95% and characteristics dimension of 2 m.

Then the user needs to specify how many dedicated channels the receiver has for tracking: *Trackable satellites limit*. This value allows the receiver to track at the same time more than one satellite (the maximum number depends on the hardware).

The main purpose of tracking is to refine the coarse values of code phase and frequency obtained during acquisition, especially as the signal properties change over time. Once this is achieved, the signal can be used for positioning. After a receiver has synchronized with a received signal it must continue to operate in such a way that it stays locked with its code reference, tracking the coded incoming signal. The tracking consists of two parts, **carrier phase tracking** (performed by PLL or FLL) and **code phase tracking** (performed by DLL): the first one can be achieved either by tracking the phase of the signal (PLL) or by tracking the frequency (FLL).

A carrier wave replica is generated for the data demodulation using a PLL or FLL to estimate the synchronization error. FLLs are less sensitive to frequency variations, therefore they are more robust. On the other hand since they "replicate" only the frequency of the carrier and not the phase, the accuracy achieved is in general lower. The synchronization error is then filtered and used to control a digitally controlled oscillator, that is able to generate a local replica of the carrier, with the estimated frequency. The filter is used to control the time response of the loop. The loop filter equivalent bandwidth is a critical parameter: when it is too low, the loop significantly filters the noise, but cannot react fast to high signal dynamics (which strongly affect the frequency), on the other hand when it is high, the loop does not reduce enough the noise, but can react fast to high signal changes. The sensitivity of the tracking loop represents the minimum C/N_0 the loop can bear without losing lock. Usually phase tracking is possible for signals received down to about 27 dB-Hz (depending on oscillator and application).

VEHICLE PARTS AND PROPERTIES MODELS

In order to understand if the C/N_0 is high enough to track the incoming signal, ASTOS allows to directly specify the *minimum* C/N_0 required for tracking or to provide the characteristics of the PLL/FLL. In this last case, following parameters are required for both the carrier and code loops:

- *Predetection integration time*, which usually is chosen equal to the duration of one code period
- *Maximum line of sight jerk dynamics*

If a *FLL characteristics* are used the following condition applies [105]:

$$\sigma_{FLL} = \sigma_{tFLL} + \frac{f_e}{3} \leq \frac{1}{12T} \quad (3.142)$$

where σ_{FLL} is the FLL 1-sigma error, σ_{tFLL} is the FLL 1-sigma thermal noise and F_e is the dynamic stress error. A second-order FLL is assumed. The following parameter is required for this computation (along with the common parameters already listed):

- *Carrier loop noise bandwidth*

On the other hand, if a *PLL characteristics* are used the following condition has to be matched in order to track the incoming signal [105]:

$$\sigma_{PLL} = \sqrt{\sigma_{tPLL}^2 + \sigma_v^2 + \theta_A^2} + \frac{\theta_e}{3} \leq 15^\circ \quad (3.143)$$

where σ_{PLL} is the PLL 1-sigma error, σ_{tPLL} is the PLL 1-sigma thermal noise, σ_v is 1-sigma vibration-induced oscillator jitter and θ_e is the Allan variance-induced oscillator jitter. In solving Eq. 3.143 a third-order PLL is assumed, since it involves one integrator less than a FLL tracking loop of the same order. With respect to the list of parameters required for the FLL 1-sigma error, for a PLL the *oscillator* characteristics are required too. In this case, the reference oscillator vibration and Allan deviation-induced frequency jitter are not small-order effects and therefore not negligible as for a FLL. The following oscillator properties are required [105]:

- *Short-term Allan deviation*
- *Oscillator vibration sensitivity*
- *Average random vibration power*
- *Random vibration modulation minimum frequency*
- *Random vibration modulation maximum frequency*

The Allan deviation is a common measure of oscillator stability based on the instantaneous fractional frequency deviation. For the computation of the vibration-induced oscillator jitter, the random vibration power curve is assumed to be flat between the provided minimum and maximum frequencies.

The goal of the code tracking is to generate a local replica of the GNSS code that is perfectly synchronized with the incoming one. To do so the DLL estimates the synchronization error between the local replica and the incoming code. This error is then filtered and used to control an oscillator that commands the generation of the local replica. The code tracking is most often implemented as a DLL where three local codes (replicas) are generated and correlated with the incoming signal. These three replicas are referred to as the early, prompt and late

replica, respectively. The three codes are often separated by a half chip length, which is the typical early-late correlator spacing (separation). The *DLL characteristics* are defined through the following parameters:

- Receiver front-end bandwidth
- Early-late correlator spacing
- Code loop noise bandwidth
- Noncoherent integration

Coherent integration (estimation of the phase delay) is possible only if a PLL is used, because it relies on the phase lock [105].

When there is no multi-path or other distortion of the received signal and no interference, the dominant source of range error are thermal noise range error jitter and dynamic stress error, which are computed with the previous parameters. Therefore, the rule tracking threshold for the DLL is [105]:

$$\sigma_{DLL} = \sigma_{tDLL} + \frac{R_e}{3} \leq \frac{D}{6} \quad (3.144)$$

where σ_{DLL} is the DLL 1-sigma error, σ_{tDLL} is the DLL 1-sigma thermal noise, R_e is the dynamic stress error. This last term is negligible, since it is already "mitigated" by the carrier loop. It can be proven that this tracking threshold is significantly lower than that of the PLL/FLL. Therefore, this 1-sigma error is used only to estimate the receiver noise equivalent range error.

The full mathematical description of the equations presented in this chapter (Eq. 3.143, Eq. 3.142 and Eq. 3.144) can be found in [105].

3.5 Aerodynamics and Aerothermodynamics Models

In this section the aerodynamic and aerothermodynamic models in the software are explained. Aerodynamic models allow the user to consider the aerodynamic forces and moments acting on the vehicle moving through the the atmosphere. Aerothermodynamic models consider the interaction between a moving body and the surrounding gases from a thermodynamic point of view by modelling the energy transfer from the gas to the vehicle model. Both model types are explained in the sections below:

- The **Aerodynamics** models are listed and explained in Section 3.5.1
- The **Aerothermodynamics** models are listed and explained in Section 3.5.2

3.5.1 Aerodynamics models

Description

VEHICLE PARTS AND PROPERTIES MODELS

An aerodynamics model describes the forces and moments acting on a body moving through a gas (i.e. atmosphere). The following models are supported:

- *Tabular* approximates aerodynamic force and moment coefficients by user defined tabular data, see Section 3.5.1.1
- *Misdat* computes force and moment coefficient according to *Missile DATCOM* of the USAF, see Section 3.5.1.3
- *Sose* calculates the supersonic and hypersonic aerodynamic properties of re-entry vehicles above a velocity of Mach number 2.5, see Section 3.5.1.4
- *Hermes 3dof* and *Hermes 6dof* contain ready-to-use models of a Hermes-like high-lift re-entry vehicles, see Section 3.5.1.2
- *User* links a user-defined aerodynamics dynamic library file which provides aerodynamic forces and moments, see Section 3.5.1.5
- *Distributed* computes aerodynamics force and moment coefficients acting on specific piece of the vehicle, see Section 3.5.1.7

Potential input errors

The aerodynamic reference area and length must be positive. It is suggested to use a 6-DOF aerodynamics model in case that 6-DOF equations of motion are utilized in the scenario.

3.5.1.1 Tabular Aerodynamics

Description

A tabular aerodynamics model is used to approximate tabular data of aerodynamic force and moment coefficients. It can be applied for all axial symmetric and non-axial symmetric vehicles. Force and moment coefficients can be either provided as constant value or as function of certain independent variables in tabular form.

Configuration

A tabular aerodynamics requires the following settings:

- *Aerodynamic moments definition* specifies how the aerodynamic torque is computed with one of the following options:
 - *None* means that no aerodynamic moment will be computed and the *Coefficients* table below does only accept the definition of aerodynamic force coefficients
 - *Moment Coefficients* allows the user to specify aerodynamic force and moment coefficients in the *Coefficients* table below. Furthermore, the *Moment reference centre* and *Reference length* has to be specified.
 - *Centre of Pressure Position* allows the definition of a *Center of pressure* and the aerodynamic force and damping coefficients in the *Coefficients* table. And the *Reference length* needs to be specified when at least one damping coefficient is given.

VEHICLE PARTS AND PROPERTIES MODELS

- *Reference area* is the frontal area of the vehicle and typically it is the maximum cross-sectional area. It can be defined in various ways through one of the *Type* options:
 - *Area* allows the specification of a reference area as *Value* or *Profile*.
 - *Diameter* allows the specification of a reference area through its diameter. It can be specified either as *Value* or as *Profile*.

In case of a *Profile* definition, it is a function of *Phase_Time*, *Mach* or *Altitude*. More information about profile data tables is given in chapter Profile Data Tables in User Manual. When *Optimizable* is *Enabled*, the reference area (or diameter) requires the definition of:

- *Optimizable* as defines the type of the optimizable quantity:
 - *Control* adds a control (a function of the independent variable) to the optimal control problem
 - *Parameter* adds an optimizable real parameter to the optimal control problem
- *Lower Bound* is the lower bound of the optimizable quantity
- *Upper Bound* is the upper bound of the optimizable quantity

Tip: An optimizable reference area is helpful to communicate the reference area with other model objects and to link it with e.g. a stage diameter. Such a link can be performed with a *Model Params* constraint (see chapter Section 6.9).

- *Reference length* is typically the chord length of the vehicle, but for launchers it is the maximum body diameter. It is optional in case the *Aerodynamic moments definition* is *None* but required for the moment computation in all other cases (*Moment Coefficients* or *Centre of Pressure Position*).
- *Center of pressure* defines the location of the center of pressure and is only available when the *Aerodynamic moments definition* is specified as *Centre of Pressure Position*. It is defined using the *X offset*, *Y offset* and *Z offset* along the x-, y- and z-axis of the vehicle body fixed frame (B) respectively w.r.t. the global node (and not w.r.t. the center of mass (CoM)). Each offset can be either *Defined by* as constant *Value* or *Profile*. The latter one can be a function of the phase time, mission time or Mach number.
- *Moment reference centre* defines the location of a moment reference centre in case the *Aerodynamic moments definition* is specified as *Moment Coefficients*. This reference point is defined using the *X offset*, *Y offset* and *Z offset* along the x-, y- and z-axis of the vehicle body fixed frame (B) respectively w.r.t. the global node (and not w.r.t. the center of mass (CoM)).
- *Coefficients* contains the dimensionless aerodynamic force and moment coefficient data. All defined coefficients are summarized in a table with the following columns:
 - *Type* defines the kind of the aerodynamic coefficient: *Force* or *Moment*
 - *Frame* defines the coordinate system in which the aerodynamic coefficient is defined.
 - *Axis* defines the axis of the frame in which direction the coefficient acts. One example is the *Drag direction*, the direction in which the (positive) drag acts.

VEHICLE PARTS AND PROPERTIES MODELS

- *Character* defines the kind of the coefficient, for example the value itself (*Absolute value*) or a certain slope (e.g. *Angle of attack slope*).

Next to the table these buttons are available:

- **Add force coefficient** adds a force coefficient defined by *Frame*, *Axis* and *Character*
- **Add damping coefficient** adds a damping coefficient defined by *Frame*, *Axis* and *Character*. This button is only available in case *Aerodynamic moments definition* is specified as *Centre of Pressure Position*.
- **Add moment coefficient** adds a moment coefficient defined by *Frame*, *Axis* and *Character*. This button is only available when *Aerodynamic moments definition* is specified as *Moment Coefficients*.
- **Delete selected coefficient** removes the currently selected coefficient from the table.

When adding a force or moment coefficient the following options are available:

- *Frame* specifies the coordinate frame in which the aerodynamic force or moment coefficient is defined:
 - *Air path (A)* frame is the aerodynamics frame, also known as air path system A (see Section 9.3.5.3)
 - *Body-fixed (B)* frame is attached to the vehicle's geometry (see Section 9.3.3.1)
 - *Total angle of attack (TA)* frame is an aerodynamic reference frame applied to space flight vehicles whose forward axis is an axis of rotational symmetry (see Section 9.3.3.2)
 - *Total force (TF)* frame is an aerodynamic reference frame aligned with the total aerodynamic drag and lift acting on a vehicle of rotational symmetry (see Section 9.3.3.3)
- *Axis* specifies the direction in which the coefficient acts. Depending on the coordinate frame, it might be one of the following directions for a **Force** coefficient:
 - *Axial direction* is the negative x-axis of the body fixed frame
 - *Drag direction* is the direction of the drag component, it is the negative x-axis of the air path axis system and the total force axis system
 - *Lateral direction* is the positive y-axis of the axis system
 - *Lift direction* is the direction of the lift component, it is the negative z-axis of the air path axis system
 - *Normal direction* is the negative z-axis of the body fixed frame
 - *Total lift direction* is the direction of the total lift (lift and side force). It corresponds to the negative z-axis of the total force coordinate system.
 - *Total normal direction* is the direction of the total normal force (normal and side force). It corresponds to the negative z-axis of the total angle of attack coordinate system.

A **Moment** coefficient has these options:

VEHICLE PARTS AND PROPERTIES MODELS

- *Pitch moment* defines the axis for a pitch moment component. Typically this is defined about the positive y-axis.
 - *Roll moment* defines the axis for a roll moment component. Typically this is defined about the positive x-axis.
 - *Yaw moment* defines the axis for a yaw moment component. Typically this is defined about the positive z-axis.
- *Character* specifies the kind of the coefficient, and depending on the coordinate frame and axis, one of the following options can be selected:
- *Absolute value* defines the absolute/pure component of the aerodynamic coefficient, for example the drag coefficient C_D or the pitch moment coefficient C_M
 - *Aerodynamic surface slope* is the slope/derivative of an aerodynamic surface
 - *Angle of attack slope* is the angle of attack slope component. Its dimension is one over angular, e.g. "1/Radian".
 - *Body pitch rate slope* is the normalized body pitch rate (normalized Q , normalized ω_y) slope component.
 - *Body roll rate slope* is the normalized body roll rate (normalized P , normalized ω_x) slope component.
 - *Body yaw rate slope* is the normalized body yaw rate (normalized R , normalized ω_z) slope component.
 - *Induced* is the induced drag component. It can be also specified as *Lift tan alpha* to compute the induced drag as function of lift and angle of attack: $c_{D,0} = L \tan \alpha$.
 - *Sideslip slope* is the side slip angle slope component. Its dimension is one over angular, e.g. "1/Radian".
 - *Total angle of attack slope* is the total angle of attack slope component. Its dimension is one over angular, e.g. "1/Radian".
 - *Total angle of attack rate slope* is the normalized total angle of attack rate slope component.

Each slope coefficient is multiplied by the corresponding quantity, for example *Angle of attack slope* is multiplied by the angle of attack value.

Note: Any data table provided for a slope coefficient should not depend on its slope quantity. For example, an angle of attack slope coefficient should not be a function of the angle of attack.

Each coefficient can be defined either as *Value*, *Profile* data table or *Bridged profiles* data tables. In a data table the dimension of a column is defined by its independent variables. For the bridging function setting please refer to Section 3.5.1.6, for the uncertainty definition please refer to Data Uncertainty in User Manual.

Tip: In case of *Knudsen* number dependency, please consider the bridging function.

VEHICLE PARTS AND PROPERTIES MODELS

- *Aerodynamic surfaces* can be used to specify the deflection angle of rudders, flaps or fins, for example. Further an aerodynamic coefficient is definable as function of the deflection angle. All defined surfaces are summarized in the table with the column *ID*, the identifier of an aerodynamic surface. Next to the table these buttons are available:
 - **Add aerodynamic surface** adds an aerodynamic surface defined by its *Identifier*,
 - **Delete selected aerodynamic surface** removes the currently selected aerodynamic surface from the table.

Each surface can be defined either as *Value* or *Profile* data table (see Profile Data Tables in User Manual).

Background

■ General

In case of profile data table an aerodynamics coefficient and an aerodynamics surface is a function of one or multiple of the following independent variables, also known as dimensions.

- *Mach number* M
- *Total angle of attack* α_t
- *Angle of attack* α
- *Side slip angle* β
- *Aerodynamic bank angle* μ_t
- *Aerodynamic roll angle* Φ'
- *Altitude* h
- *Pressure* p
- *Density* ρ
- *Dynamic pressure* q
- *Air path speed* V_a
- *Reynolds number* Re
- *Knudsen number* Kn
- *Angle of attack rate* $d\alpha/dt$
- *Side slip angle rate* $d\beta/dt$
- *Body roll rate* P (*omega x*) is the body-fixed axis roll rate P
- *Body pitch rate* Q (*omega y*) is the body-fixed axis pitch rate Q
- *Body yaw rate* R (*omega z*) is the body-fixed axis yaw rate R
- *Delta-x between MRC and CoG* is the distance between moment reference centre and center of gravity along the x-axis of the body-fixed frame Δx
- *Axial position* is the position along the body frame x-axis w.r.t global node.

The vector of independent variables is $x = [M, \alpha_t, \alpha, \beta, \mu_t, \Phi', h, p, \rho, q, V_a, Re, Kn, d\alpha/dt, d\beta/dt, P, Q, R, \Delta x, x_B]^T$.

Furthermore all defined aerodynamic surfaces are also available as independent variable for each aerodynamics coefficient specified as profile data table and the vector of independent variables is increased accordingly.

If the actual value(s) of the independent variable(s) is within the input data range, the data is either linearly interpolated or approximated by a natural cubic spline, a shape-preserving Akima spline or a general B-Spline. In case it is outside this range, either the data is kept constant (*Nearest Value*), extrapolated based on the chosen approximation method, or an exception is raised. More information about tabular data is given in Profile Data Tables in the User Manual.

■ *Aerodynamics forces and moments representation depending on the aerodynamics frame*

The aerodynamic force vector depends on the used coefficient set and their coordinate system representation. Fig. 9.9 shows the direction of the aerodynamic forces when decomposed along the body-fixed (B) and air path (A) coordinate systems (refer to Section 9.3.3.1 and Section 9.3.5.3, respectively). In the air path (A) coordinate system the aerodynamic force vector is given by

$$\vec{F}_{\text{aero},A} = \begin{bmatrix} -D \\ Y_a \\ -L_A \end{bmatrix} = qA_{\text{ref}} \begin{bmatrix} -C_D \\ C_Y \\ -C_{L_A} \end{bmatrix} \quad (3.145)$$

with the drag D , the lift L , the side force Y_a , the dynamic pressure q , the reference area A_{ref} , the drag coefficient C_D , the side coefficient C_Y and the lift coefficient C_L . Drag D as well as its coefficient C_D are defined positive in the negative x_A -direction (i.e. in negative airpath velocity direction), lift L as well as its coefficient C_L are pointing in the negative z_A -direction and side force Y_a as well as its coefficient C_Y are positive in the positive y_A -direction. Aerodynamic coefficients defined in the air path coordinate system are used for non-axisymmetric vehicles.

However, in the total force (TF) coordinate system (see Section 9.3.3.3) the side force is zero by definition and the aerodynamic force vector is given by

$$\vec{F}_{\text{aero},TF} = \begin{bmatrix} -D \\ 0 \\ -L_t \end{bmatrix} = qA_{\text{ref}} \begin{bmatrix} -C_D \\ 0 \\ -C_{L_t} \end{bmatrix} \quad (3.146)$$

with the total lift L_t as well as the total lift coefficient C_{L_t} pointing in the negative z_{TF} -direction. The total lift L_t is the vector sum of the lift L and side force Y_a defined in the air path coordinate system. The drag and drag coefficient are positive in the negative x_{TF} -direction (i.e. in negative airpath velocity direction), which is collinear with x_A -direction. Aerodynamic coefficients in the total force coordinate system are typically used for rotational symmetric vehicles where lift and side force cannot be distinguished and are represented as total lift force.

By analogy to drag and lift, aerospace standards define the axial force A to be positive in the negative x_B -direction, the normal force N to be positive in the negative z_B -direction, and the lateral force Y to be positive in the positive y_B -direction of the body-fixed (B) coordinate system. The aerodynamic force vector is given by

$$\vec{F}_{\text{aero},B} = \begin{bmatrix} -A \\ Y \\ -N_B \end{bmatrix} = qA_{\text{ref}} \begin{bmatrix} -C_A \\ C_Y \\ -C_{N_B} \end{bmatrix} \quad (3.147)$$

with the axial coefficient C_A , the normal coefficient C_N and the lateral coefficient C_Y . Aerodynamic coefficients defined in the body-fixed coordinate system are used for non-axisymmetric vehicles.

In the total angle of attack (TA) coordinate system (see Section 9.3.3.2) the lateral force is zero by definition and the aerodynamic force vector is given by

$$\vec{F}_{\text{aero},TA} = \begin{bmatrix} -A \\ 0 \\ -N_{t,TA} \end{bmatrix} = qA_{\text{ref}} \begin{bmatrix} -C_A \\ 0 \\ -C_{N_t,TA} \end{bmatrix} \quad (3.148)$$

with the total normal force N_t as well as the total normal coefficient C_{N_t} to be positive in the negative z_{TA} -direction. The total normal force N_t is the vector sum of the normal force N and the lateral force Y defined in the body-fixed coordinate system. The axial force and axial coefficient are positive in the negative x_{TA} -direction, which is collinear with x_B -direction. Aerodynamic coefficients defined in the total angle of attack coordinate system are typically used for rotational symmetric vehicles where normal and lateral force cannot be distinguished and are represented as total normal force.

The aerodynamic force coefficients are often composed of sub-components such as certain slope (i.e. derivative) coefficients, for example

$$\vec{F}_{\text{aero},A} = qA_{\text{ref}} \begin{bmatrix} -C_D(x) - C_{D_0}(x) \\ C_{Y_a}(x) + \left(\frac{dC_{Y_a}}{d\beta}(x) \right) \beta \\ -C_L(x) - \left(\frac{dC_L}{d\alpha}(x) \right) \alpha \end{bmatrix}_A \quad \text{or} \quad \vec{F}_{\text{aero},B} = qA_{\text{ref}} \begin{bmatrix} -C_A(x) \\ C_Y(x) + \left(\frac{dC_Y}{d\beta}(x) \right) \beta \\ -C_N(x) - \left(\frac{dC_N}{d\alpha}(x) \right) \alpha \end{bmatrix}_B \quad (3.149)$$

with the induced drag coefficient C_{D_0} .

The computation of the aerodynamic moment vector depends on the used moment coefficients set. Besides, these aerodynamic reference points are defined:

- the moment reference centre (MRC), which is the reference point where the moments (i.e. the defined moment coefficients) act on the vehicle
- the centre of pressure (CoP), which is the reference point where the sum of the aerodynamic forces act on the vehicle

By definition, if no additional moment coefficients (except damping coefficients) are defined, the moment reference centre definition is equivalent to a centre of pressure definition. The output functions of the centre of pressure coordinates w.r.t. global node depend on:

- No moment coefficient is defined: the specified *Moment reference centre* is considered to be the centre of pressure and is provided as output

VEHICLE PARTS AND PROPERTIES MODELS

- At least one non-damping moment coefficient is defined: the centre of pressure x-position is computed with pitch moment over the total normal force in the total angle of attack (TA) coordinate system, while the y- and z-position are set to zero

The aerodynamic torque for roll l , pitch m , and yaw n are defined positive along the positive axes of the body-fixed (B) coordinate system. Analogous to the body-fixed (B) coordinate system, the aerodynamic torque for roll, pitch and yaw in the total angle of attack (TA), air path (A) and total force (TF) coordinate systems are defined to be positive along the positive axes of their corresponding coordinate system. The aerodynamic torque vector in the body-fixed coordinate system is computed as

$$\vec{M}_{\text{aero}} = \begin{bmatrix} l \\ m \\ n \end{bmatrix}_B = q A_{\text{ref}} L_{\text{ref}} \begin{bmatrix} C_{LL_t}(x) \\ C_{M_t}(x) \\ C_{LN_t}(x) \end{bmatrix}_B + \begin{bmatrix} F_Z \Delta y + F_Y \Delta z \\ F_Z \Delta x - F_X \Delta z \\ -F_Y \Delta x - F_X \Delta y \end{bmatrix} \quad (3.150)$$

with the "total" moments coefficients (e.g. C_{M_t}) computed from the sum of all their components:

$$\begin{aligned} C_{LL_t}(x) &= C_{LL}(x) + \beta \left(\frac{dC_{LL}}{d\beta}(x) \right) + P \left(\frac{dC_{LL}}{dP}(x) \right) + R \left(\frac{dC_{LL}}{dR}(x) \right) + \Delta_{fin} \left(\frac{dC_{LL}}{d\Delta_{fin}}(x) \right) \\ C_{M_t}(x) &= C_M(x) + \beta \left(\frac{dC_M}{d\beta}(x) \right) + Q \left(\frac{dC_M}{dQ}(x) \right) + \alpha \left(\frac{dC_M}{d\alpha}(x) \right) + \alpha_t \left(\frac{dC_M}{d\alpha_t}(x) \right) + \dot{\alpha}_t \left(\frac{dC_M}{d\dot{\alpha}_t}(x) \right) \\ C_{LN_t}(x) &= C_{LN}(x) + \beta \left(\frac{dC_{LN}}{d\beta}(x) \right) + P \left(\frac{dC_{LN}}{dP}(x) \right) + R \left(\frac{dC_{LN}}{dR}(x) \right) \end{aligned} \quad (3.151)$$

with the reference length L_{ref} . F_X , F_Y and F_Z are the aerodynamic forces represented in the body-fixed frame; C_M is the pitch moment coefficient and C_{LL} and C_{LN} the moment coefficient of roll and yaw, respectively. P , Q , R are the normalized angular rates about the body-fixed axes x, y and z, respectively. Δ_{fin} is the fin deflection angle (also known as fin cant angle). Angle of attack, sideslip angle, total angle of attack and total angle of attack rate (normalized) follow the ASTOS nomenclature (Table 9.6).

The normalized rates (body rates P , Q , R and alpha total rate) are computed by the product of the rates itself with the normalization factor $= L_{\text{ref}} / (2 * V)$; where V is the air path speed.

The vector $(\Delta x, \Delta y, \Delta z)$ describes the difference between the point where the aerodynamic moments are computed (Moment Reference Center (MRC)) and the center of gravity (CoG) in the "A-Y-N" frame: a reference system produced by the Axial-Lateral-Normal forces (see Fig. 9.9). Typically the moment database is provided at the vehicle nose (re-entry vehicle) or at the center of pressure (CoP). For typical aerodynamic coefficient sets (re-entry vehicle and sounding rockets), MRC is at the vehicle nose and Delta_X is negative. $\Delta x_{\text{MRC\_CG}}$ can be used as independent variable for the moment coefficients CM_Q and CLN_R . $\Delta x_{\text{MRC\_CG}}$ can be used as independent variable for the aerodynamic coefficients (e.g. CM_Q).

Note: All coefficients with an underscore in the name are slopes, for example CL_{Alpha} (unit is 1/Radian) is multiplied with the angle of attack to compute the corresponding CL component. While the coefficients without any underscore are only interpolated from the user-defined profile data table (e.g. CL).

Finally, the aerodynamic force and torque vectors are applied in the set of equations of motion, represented in the coordinate system required by the dynamic equations, for example in the body-fixed or inertial J2000 coordinate system (see Section 9.3.2.1).

Optimization constraints

The tabular aerodynamic model presents a constraint interface that allows the user to access specific functions limiting them with a normal constraint: reference, upper limit, lower limit and range. The constraint name is *Equipment Design* detailed in Section 6.9; the available *Parameters* are:

- *Min_Alpha_Domain*: it sets the lower limit of the angle of attack range, the unit has to be compatible with Radian.
- *Max_Alpha_Domain*: it sets the upper limit of the angle of attack range, the unit has to be compatible with Radian.
- *Trim_Condition*: in case of a 6-DOF aerodynamics it accesses the pitch moment, the unit has to be compatible with Newton*Meter.

Potential input errors

The Mach number data must be positive and monotonically increasing. The axial force and induced drag coefficients must be positive.

3.5.1.2 Hermes Reentry Vehicle Aerodynamics

Description

These analytical models are suited for high-lift re-entry vehicles in trimmed flight. The hypersonic lift-to-drag ratio is approximately 1.2. Two subtypes are available:

- *Hermes 3dof* provides aerodynamic data for 3 degrees of freedom, see Section 3.5.1.2.1
- *Hermes 6dof* provides aerodynamic data for 6 degrees of freedom, see Section 3.5.1.2.2

Configuration

These settings are available:

- *Reference length* is typically the chord length of the vehicle, but for launchers it is the maximum body diameter. It is optional.
- *Reference area* is the frontal area of the vehicle and typically it is the maximum cross-sectional area. It can be defined in various ways through one of the *Type* options:
 - *Area* allows the specification of a reference area as *Value* or *Profile*.
 - *Diameter* allows the specification of a reference area through its diameter. It can be specified either as *Value* or as *Profile*.

VEHICLE PARTS AND PROPERTIES MODELS

In case of a *Profile* definition, it is a function of *Phase_Time*, *Mach* or *Altitude*. More information about profile data tables is given in chapter Profile Data Tables in User Manual. When *Optimizable* is *Enabled*, the reference area (or diameter) requires the definition of:

- *Optimizable as* defines the type of the optimizable quantity:
 - *Control* adds a control (a function of the independent variable) to the optimal control problem
 - *Parameter* adds an optimizable real parameter to the optimal control problem
- *Lower Bound* is the lower bound of the optimizable quantity
- *Upper Bound* is the upper bound of the optimizable quantity

Tip: An optimizable reference area is helpful to communicate the reference area with other model objects and to link it with e.g. a stage diameter. Such a link can be performed with a *Model Params* constraint (see chapter Section 6.9).

Background

The Hermes aerodynamics provides a nominal angle of attack vs. Mach profile and lower and upper bounds which may be used for re-entry trajectory optimization. For realistic results, the angle of attack control of a Hermes-type vehicle must be restricted to this corridor [10], [11].

Given the local Mach number, a nominal angle of attack value as well as lower and upper bounds along the nominal profile are computed. The complete profile is shown in Fig. 3.49. Both nominal profile and corridor schedules are smooth, piece-wise analytic functions.

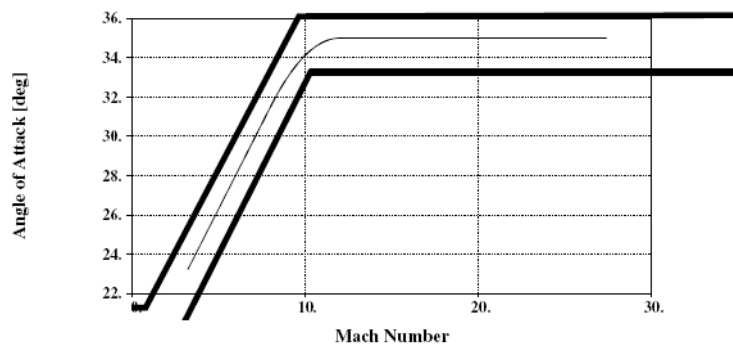


Fig. 3.49: Nominal angle of attack vs. Mach profile and angle of attack corridor

3.5.1.2.1 Trimmed 3-DOF Hermes Aerodynamics

Description

VEHICLE PARTS AND PROPERTIES MODELS

This is an analytical model of the aerodynamic forces acting on a high-lift re-entry vehicle in trimmed flight. The aerodynamic characteristics are computed from smooth analytic functions of Mach number and angle of attack fitting a realistic data base of a winged vehicle [10].

Note: The model is only valid for an angle of attack between 10° and 40° and from Mach 2.5 to 30. Outside this domain, data is extrapolated and may become absurdly erroneous.

Background

The aerodynamic force vector (i.e. drag and lift) is computed as

$$\hat{\vec{F}} = q A_{\text{ref}} \begin{bmatrix} -C_D(M, \alpha) \\ 0 \\ -C_L(M, \alpha) \end{bmatrix}_A \quad (3.152)$$

where α is the angle of attack, M is the Mach number, q the dynamic pressure, and A_{ref} the aerodynamic reference area. The side force component in the aerodynamics frame is set to zero assuming coordinated banking maneuvers. Each coefficient is computed separately by a polynomial of the form

$$\begin{aligned} C_D(M, \alpha) &= \frac{1}{e^M} \sum_{i=0}^2 r_{D,i} \alpha^i + \sum_{k=0}^3 \left(\sum_{n=0}^2 P_{D,n} \alpha^n \right) M^k \\ C_L(M, \alpha) &= \frac{1}{e^M} \sum_{i=0}^2 r_{L,i} \alpha^i + \sum_{k=0}^3 \left(\sum_{n=0}^2 P_{L,n} \alpha^n \right) M^k \end{aligned} \quad (3.153)$$

where

$$2.5 \leq M \leq 30, \quad 10 \leq \alpha \leq 40 \text{deg} \quad (3.154)$$

with constant coefficients p_i and r_i . Plots of the lift and drag coefficients are shown in Fig. 3.50 and Fig. 3.51, the original tabular data are marked with circles.

This model returns an aerodynamic reference area of 73 m². To match the original configuration, the vehicle simulated with these aerodynamics should have a mass of approximately 16,000.0 kg.

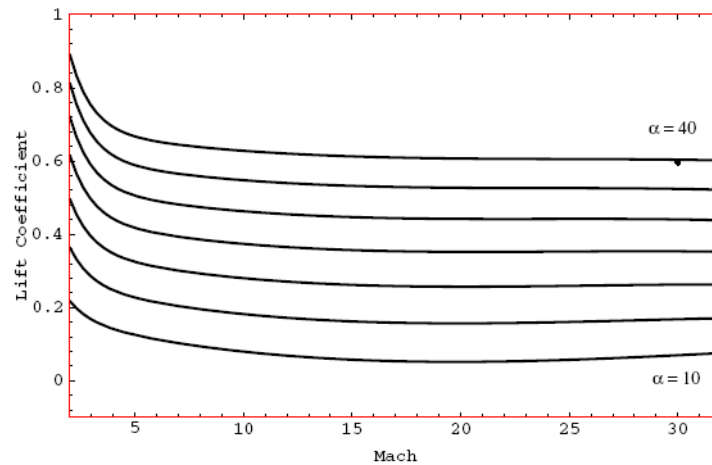


Fig. 3.50: Lift coefficient versus Mach number and angle of attack

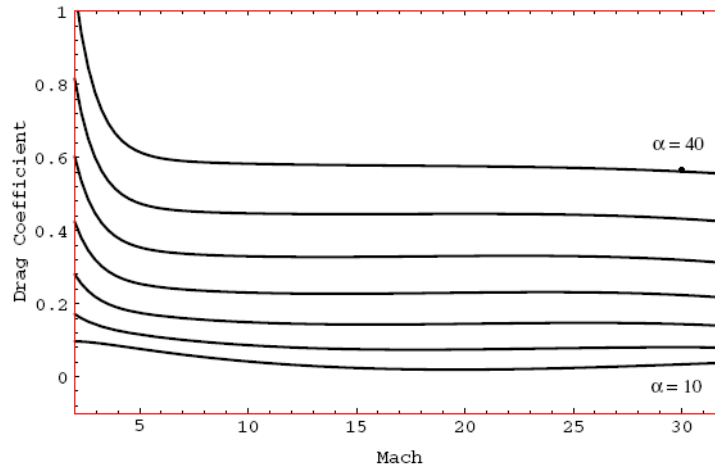


Fig. 3.51: Drag coefficient versus Mach number and angle of attack

3.5.1.2.2 Trimmed 6-DOF Hermes Aerodynamics

Description

This is an elaborate model of the dominant aerodynamic characteristics of a winged high-lift re-entry vehicle in trimmed flight [12] and is developed from an extensive aerodynamic data base [13]. Above Mach 25, viscous boundary layer effects are incorporated. The model may be used for realistic re-entry simulation and preliminary flight control system design.

Note: This model is only valid from Mach number 2 to 30, for angle of attack between 10° and 45° and a moderate sideslip angle between $\pm 5^\circ$. Outside this domain, data is extrapolated and may become absurdly erroneous.

Background

The aerodynamic force vector is computed as

$$\hat{\vec{F}} = q A_{\text{ref}} \begin{bmatrix} -C_D(M, \alpha) \\ \frac{\partial C_{Y_a}}{\partial \beta}(M, \alpha) \beta \\ -C_L(M, \alpha) \end{bmatrix}_A \quad \hat{\vec{M}} = q A_{\text{ref}} L_{\text{ref}} \begin{bmatrix} \frac{\partial C_l}{\partial \beta}(M, \alpha) \beta \\ 0 \\ \frac{\partial C_n}{\partial \beta}(M, \alpha) \beta \end{bmatrix}_B \quad (3.155)$$

where α is the angle of attack, β the sideslip angle, M the Mach number, q the dynamic pressure, and A_{ref} and L_{ref} the aerodynamic reference area and length, respectively.

All aerodynamic coefficients are nonlinear functions of the angle of attack and the local Mach number. Side force, roll and yaw moment are also linear functions of the sideslip angle. Plots of the coefficient data vs. Mach at various angles of attack are shown in Fig. 3.52 to Fig. 3.57. Above Mach 25, the lift-to-drag ratio is significantly reduced by viscous boundary layer effects (see Fig. 3.54).

VEHICLE PARTS AND PROPERTIES MODELS

This model returns an aerodynamic reference area of 73 m^2 and reference length of 15.5 m . To match the original configuration, the vehicle simulated with these aerodynamics should have a mass of $16,000.0 \text{ kg}$ and principal moments of inertia along the body-fixed axes of $I_{xx} = 36,000 \text{ kgm}^2$, I_{yy} and $I_{zz} = 256,000 \text{ kgm}^2$.

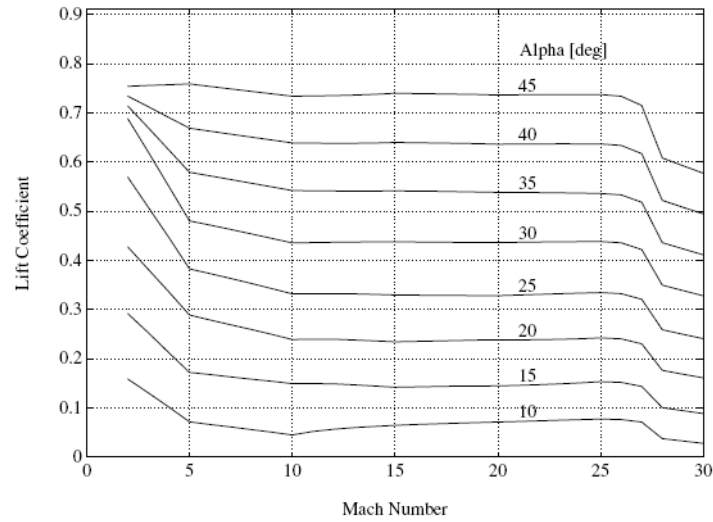


Fig. 3.52: Lift coefficient vs. Mach number for constant angles of attack

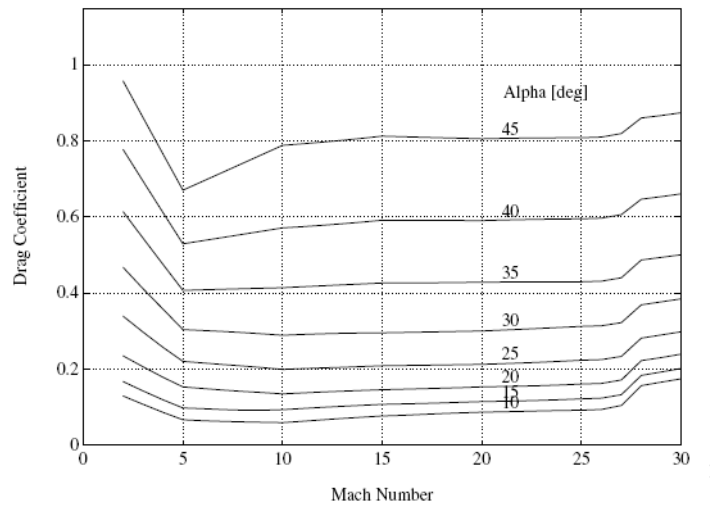


Fig. 3.53: Drag coefficient vs. Mach number for constant angles of attack

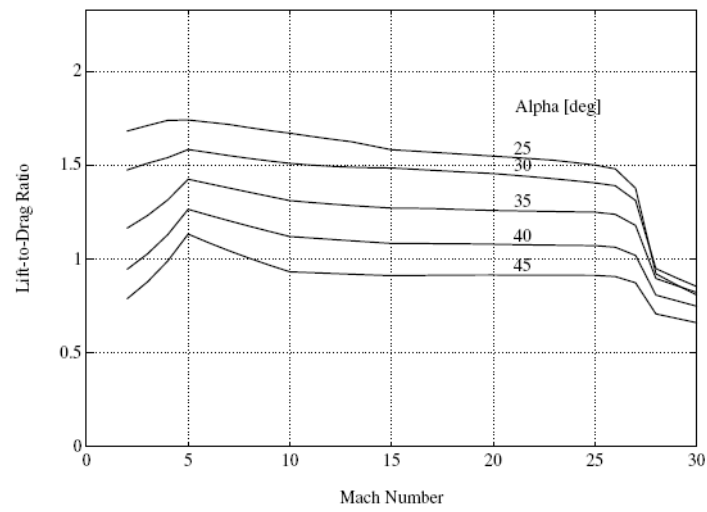


Fig. 3.54: Lift-to-drag ratio vs. Mach number for constant angles of attack

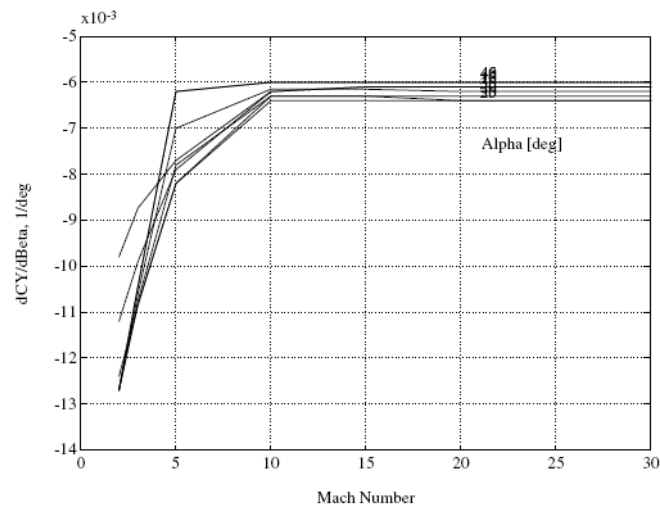


Fig. 3.55: Side force derivative $\partial C_Y/\partial \beta$ vs. Mach number for constant angles of attack

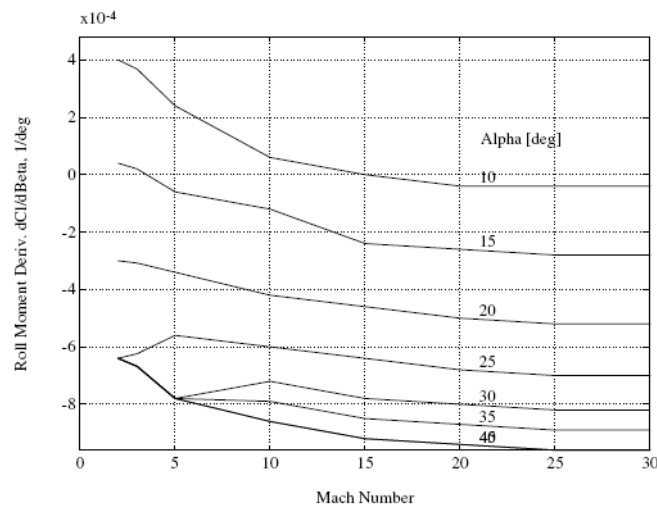


Fig. 3.56: Roll moment derivative $\partial C_l / \partial \beta$ vs. Mach number for constant angles of attack

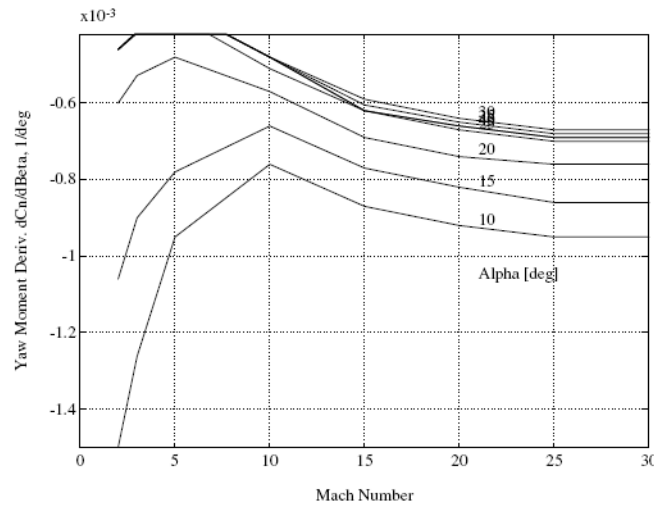


Fig. 3.57: Yaw moment derivative $\partial C_n / \partial \beta$ vs. Mach number for constant angles of attack

3.5.1.3 Misdat Aerodynamics

Description

The *Missile DATCOM* aerodynamics model computes the aerodynamic forces and moments as well as the center of pressure as function of the vehicle shape. The vehicle shape is automatically computed using the user-defined parts in the GUI. DATCOM assumes as reference area the area of a circle built on the maximum diameter of the vehicle, as reference length the total length of the vehicle and as moment reference centre the nose of the vehicle.

Note: This model should be only used for launchers with a fairing and it cannot be used after the jettison of the fairing. In this case, it is suggested to use a tabular aerodynamics (see Section 3.5.1.1).

Tip: The number of boosters has to be constant in all the phases where the same aerodynamics model is used! In case Missile DATCOM shall be used in phase before and after jettisoning boosters it is suggested to use two separate *Missile DATCOM* aerodynamics.

Tip: The shape and length of the vehicle is taken from the nominal values defined in the vehicle components. In case one of this value is an optimizable parameter, still the nominal value is used. In order to have *Missile DATCOM* using the optimized value, the user has to manually copy the real parameter value inside the component definition.

Configuration

Missile DATCOM has the following settings:

- *Use DLL* defines whether the DATCOM dynamic library is called at every time step (*enabled*) or just once during initialization (*disabled*). By default it is *disabled* and in this case Missile DATCOM executable file (`misdat.exe`) is called only once at the beginning of the computation process for the vehicle configuration at this time and the resulting aerodynamic data is stored and at each time step the pre-processed data is interpolated. In case this option is *enabled* the Missile DATCOM library (`misdat.dll`) is used and the evaluation of the aerodynamics data is done at each time step which is a non-negligible computational effort and the overall computation becomes much slower. Compared to the *disabled*-case, both the reference area and the reference length are updated at each time step.

Tip: It is suggested to keep this option *disabled* for optimization purpose, since some optimizable parameters could cause insolvable DATCOM input.

Given that `%ASTOS%` represents the installation folder of the *ASTOS* software, the Missile DATCOM binaries are expected to be in the `%ASTOS%/bin/misdat.exe` and `%ASTOS%/lib/aerodynamics/misdat.dll`.

- *Do not recalculate aerodynamics* does not call the Missile DATCOM library and keeps the already present Missile DATCOM output files if *enabled*. By default it is *disabled* so that the library is always called and the output files are overwritten and up to date.
- *Compute pressure* activates the computation of the aerodynamics contribution to the flexible dynamics and requires the feature *Multibody dynamics*.
- *Roughness* is the surface roughness of the vehicle and required by the Missile DATCOM library.

Background

All pre-processed data (e.g. aerodynamics coefficients) is available as tabular data and can be interpolated. The use of pre-processed aerodynamics data reduces the time required to compute the trajectory, but only if an accurate selection of the independent variables is performed:

- **Mach number**

VEHICLE PARTS AND PROPERTIES MODELS

It is used as independent variable and the discretization takes into account different flight regimes: more points during transonic and supersonic regime, and less points during hypersonic regime.

■ Angle of attack

A dependency on alpha can be avoided considering the computation of slope coefficients. For example, the computation of the angle of attack slope of the normal force coefficient (CNA) at zero angle of attack allows the computation of the aerodynamic normal force at small angles of attack with an acceptable precision. And this is typically true during the atmospheric flight of launchers.

■ Altitude

This dependency would be only required for the axial force coefficient, but it is quite weak. After some performance tests this dependency was removed to reduce the computation time.

In case *Use DLL* is *enabled*, all independent variables are included in the computation of the Missile DATCOM data.

Since the shape of the vehicle is generated as a radius profile function of the longitudinal dimension, the resulting surface is axial-symmetric and only a reduced set of coefficients is different from zero:

- **CA** is the axial force coefficient
- **CNA** is the normal force angle of attack (alpha) slope coefficient
- **CNQ** is the normal force normalized pitch rate (normalized body rate Q) slope coefficient
- **CYB** is the lateral force side slip angle (beta) slope coefficient
- **CYR** is the lateral force normalized yaw rate (normalized body rate R) slope coefficient
- **CMA** the pitch moment angle of attack (alpha) slope coefficient
- **CMQ** the pitch moment normalized pitch rate (normalized body rate Q) slope coefficient
- **CLNB** the yaw moment side slip angle (beta) slope coefficient
- **CLNR** the yaw moment normalized yaw rate (normalized body rate R) slope coefficient

The center of pressure (CoP) is computed analytically from CNA and CMA. Considering the moment reference center (MRC) in the vehicle nose, the distance between the CoP and the MRC is

$$\Delta_{CoP-MRC} = -\frac{CMA}{CNA} \cdot L_{ref} \quad (3.156)$$

The aerodynamic force vector in body-fixed axis frame is defined by

$$\hat{F} = qA_{ref} \begin{bmatrix} -C_A(x) \\ \left(\frac{dC_Y}{d\beta}(x) \right) \beta + \left(\frac{dC_Y}{dR}(x) \right) R \\ - \left(\frac{dC_N}{d\alpha}(x) \right) \alpha + - \left(\frac{dC_N}{dQ}(x) \right) Q \end{bmatrix}_B \quad (3.157)$$

and the aerodynamic moment vector in body-fixed axis frame is defined by

$$\hat{M} = qA_{\text{ref}}L_{\text{ref}} \begin{bmatrix} 0 \\ \left(\frac{dC_M}{d\alpha}(x) \right) \alpha + \left(\frac{dC_M}{dQ}(x) \right) Q \\ \left(\frac{dC_{LN}}{d\beta}(x) \right) \beta + \left(\frac{dC_{LN}}{dR}(x) \right) R \end{bmatrix} \begin{bmatrix} + F_Z \Delta x \\ - F_Y \Delta x \end{bmatrix} \quad (3.158)$$

Where F_Y and F_Z are the y- and z-component of the aerodynamic force vector defined in Eq. 3.157 and Δx is the distance between moment reference centre (nose of vehicle) and center of mass.

3.5.1.4 SOSE Aerodynamics

Description

SOSE, the Second Order Shock Expansion software, calculates the supersonic and hypersonic aerodynamic properties of vehicles above Mach number 2.5. It calculates the aerodynamic coefficients C_A (axial force coefficient), C_N (normal force coefficient), and C_M (pitch moment coefficient). *SOSE* is called as dynamic link library (DLL).

Note: *SOSE* should be used for re-entry vehicles only.

Note: This model can only be used in combination with *Detailed Design* vehicle components. For more information on these, refer to Section 3.3.

Configuration

No configuration settings are available for this aerodynamics model.

Background

Vehicle geometry is represented by a surface grid consisting out of arbitrary blocks, creating flat plates (panels) along the surface. Each flat plate possesses a characteristic normal-to-the-surface direction that allows *SOSE* to calculate the inclination. The pressure distribution along the surface of the vehicle is calculated based on the local surface inclination of the plates. All re-entry shapes present in the geometry module can be calculated with *SOSE*.

3.5.1.5 User Aerodynamics

Description

User-defined aerodynamics are very powerful to compute aerodynamic forces and moments. The computation of these values is completely programmed in the user library. This means it is the task of the user to provide the aerodynamics data. For more details see Chapter 11.

Configuration

VEHICLE PARTS AND PROPERTIES MODELS

The following settings are required to interface with a user-defined aerodynamics:

- *Library file (DLL)* is the path and filename of the user-defined DLL. The maximum length is 2048 characters. The path may be relative to your problem directory (GTP folder).
- *User parameters* defines a string with a maximum length of 1028 characters. This string is provided to the initialization routine of the DLL as input parameter. It can be used in any way, e.g. a filename for loading data, data values itself or a combination of both.

Background

For a detailed description of the interface please refer to chapter Chapter 11.

3.5.1.6 Bridging Functions

The aerodynamic coefficients can be defined in two different flow regimes: the free molecular and the continuum. The application of the correct regime is driven by the Knudsen number: the ratio between the mean free path and the reference length. For Knudsen number higher than a certain value ($Kn_{free\_mol\_flow}$) the coefficients are defined with this regime laws, while for Knudsen number lower than a second number ($Kn_{continuum}$) this regime define the aerodynamics. The complex part is in the region between these two limits, where the "Intermediate Flow" [82][119] is present.

In the intermediate region, the physical laws describing the two regimes loses gradually applicability, therefore a simple mathematical transition should be established. The references present in bibliography are not in accordance identifying the best function to perform the bridging task; therefore the most commons have been implemented in *ASTOS* allowing the flexibility to specify the required parameters according to experience or a reference.

The bridging is an optional settings that can be activated by selecting *Bridged profiles* from the *Defined by* drop-down list. Once activated, a *Bridged profiles* section shows up in the configuration panel. It allows the specification of the bridging function *Type* to be used. Moreover, the *Bridging variable* can be selected. The *Bridged profiles* section also contains *Profile 1* and *Profile 2* interpolation tables; the first one is for the continuum flow, the second for the free molecular regime. A detailed description of these options is presented in the following paragraphs. Available bridging function *Types* are:

- *Erf Log a b c*, see Section 3.5.1.6.1
- *Power Sin Limits*, see Section 3.5.1.6.2
- *Power Sin a b exp*, see Section 3.5.1.6.2
- *Sin*, see Section 3.5.1.6.2

Computation

The coefficient value is computed as:

$$Coeff = (1.0 - \Phi_f) \cdot Coeff\_Continuum + \Phi_f \cdot Coeff\_Free\_Molecular \quad (3.159)$$

where the continuum (*Profile 1*) free flow coefficients (*Profile 2*) are taken from the specified interpolation profiles and Φ_f is the bridging function detailed in Section 3.5.1.6.1 and Section 3.5.1.6.2.

3.5.1.6.1 Error function

The error function (also called the Gauss error function or probability integral) is a special function (non-elementary) of sigmoid shape which occurs in probability, statistics and partial differential equations. It is defined as:

$$\text{Erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt \quad (3.160)$$

An analytical solution is not available, therefore a five order series is implemented according to [120].

Inputs

- a, b, c : User-defined float number parameters (for convenience a default set is provided)
- **Saturation:**
This setting allows to limit the range of the intermediate flow between two fixed user defined Knudsen numbers. Since the error function never reaches the extreme values (-1 and +1), the default setting avoids small jumps at the limit values.

Bridging function

The bridging function is defined as:

$$\Phi_f(Kn) = a \cdot (1 + \text{Erf}(b \cdot \ln(Kn) + c)) \quad (3.161)$$

where Kn is the Knudsen number; a, b, c are the specified parameters; Erf is the error function defined in Eq. 3.160 and $\ln$ is the natural logarithm.

Tip: Since the error function result is between -1 and +1, the typical setting for the a parameter is 0.5.

3.5.1.6.2 Sinusoidal function

The sinusoidal function implemented is a power law of the trigonometric sine; the result is always between 0 and 1. It is possible to defined directly the function parameters (a, b) or the Knudsen numbers that limit the intermediate flow.

- **Kn_continuum**
Knudsen number where the flow switches from continuum to intermediate.

VEHICLE PARTS AND PROPERTIES MODELS

- `Kn_free_mol_flow`
Knudsen number where the flow switches from intermediate to free molecular.
- *Sin exp*:
Exponent of the sine function (default value 2.0)
- *a, b*:
User-defined parameters (for convenience a default set is provided)

In case of *Power sin limits*, the function parameters are computed from:

$$b := \frac{1}{2 \cdot \log_{10}(\text{Kn\_free\_mol\_flow} / \text{Kn\_continuum})} \quad (3.162)$$

$$a := -b \cdot \log_{10}(\text{Kn\_continuum}) \quad (3.163)$$

where `Kn_continuum` and `Kn_free_mol_flow` are the Knudsen number limits.

Bridging function

The bridging function is defined as:

$$\Phi_f(Kn) = \sin^{\text{Sin\_exp}}(\pi \cdot (a + b \cdot \log_{10}(Kn))) \quad (3.164)$$

where *Kn* is the Knudsen number and *a, b* are either user-defined or computed from Eq. 3.163 and Eq. 3.162.

3.5.1.7 Distributed Aerodynamics

Description

Distributed aerodynamics models evenly spread forces and moments along the length of the vehicle.

Configuration

The distributed aerodynamics algorithm is automatically enabled once the following input (in total angle of attack (TA) frame) are provided by the user while using the tabular aerodynamics:

- *Cumulative Axial coefficient*
- *Cumulative Normal coefficient*

Each of the aerodynamic coefficients above must be function of the following quantities:

- *Mach number*
- *Total angle of attack*
- *Axial position along the vehicle*

If the above mentioned conditions are fulfilled, the distributed aerodynamics algorithm in ASTOS is used to compute the aerodynamic forces and moments at discrete nodes along the structure. This results in a more realistic vehicle dynamics simulation.

Background

The purpose of this section is to summarize the input data required to enable the distributed aerodynamics algorithm. The latter models the vehicle to be divided in 30 slices with a node in the middle of each slice (see Fig. 3.58).

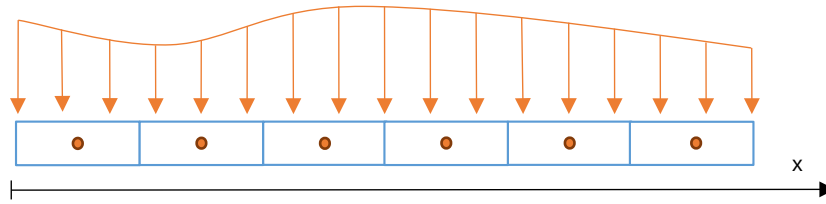


Fig. 3.58: Aerodynamic coefficients distribution

For a given Mach number and total angle of attack, there is a certain distribution of the axial **CA** and total normal **CNA** coefficients along the structure (see Fig. 3.59).

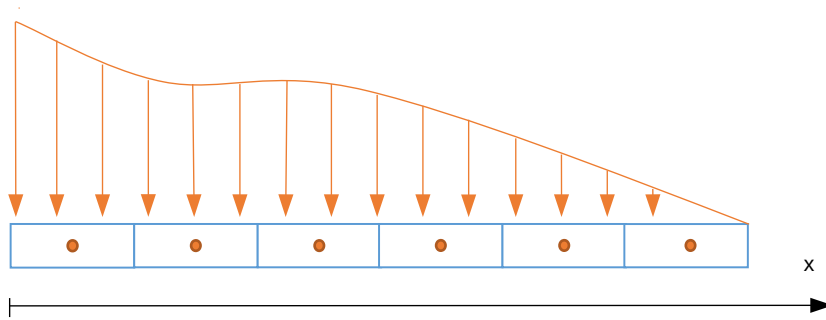


Fig. 3.59: Cumulative coefficients distribution

The required user input for the distributed aerodynamics are the cumulative values, which are computed using the following equations:

$$C_{A_{cum_j}} = C_{A_{cum_{j+1}}} + \beta_{A_j}(x_{j+1} - x_j) \quad (3.165)$$

$$C_{N_{cum_j}} = C_{N_{cum_{j+1}}} + \beta_{N_{tj}}(x_{j+1} - x_j) \quad (3.166)$$

where β_{A_j} and $\beta_{N_{tj}}$ are the loads per unit length:

$$\beta_{A_j} = \int_j^{j+1} C_A dx \quad (3.167)$$

$$\beta_{N_{tj}} = \int_j^{j+1} C_{N_t} dx \quad (3.168)$$

The cumulative pitch moment coefficient is computed by ASTOS and is not provided by the user. The coefficient is computed at the vehicle nose as follows:

VEHICLE PARTS AND PROPERTIES MODELS

$$C_{M_{cum_j}} = C_{M_{cum_{j+1}}} - \frac{\beta_{N_{t_j}}(x_{j+1} - x_j) \left((x_m - x_j) - \frac{(x_{j+1} - x_j)}{2} \right)}{L_{ref}} \quad (3.169)$$

where L_{ref} is the *reference length* and x_m is the nose node position. The $C_{M_{cum}}$ value at the nose is 0.

Table 3.20 gives an example of aerodynamic input data (see Section 3.5.1.1), where the values in the last column **CM** are computed internally by *ASTOS* and are not provided by the user.

Table 3.20: Cumulative aerodynamic coefficients in total AoA frame

Mach	Axial position	Total AoA	CA	CN	CM
None	Meter	Degree	None	None	None
0.5	0	0	0.2	0	0
0.5	0	5	0.2	0.3	-0.49735
0.5	1	0	0.18	0	0
0.5	1	5	0.18	0.27	-0.40285
0.5	2	0	0.16	0	0
0.5	2	5	0.16	0.24	-0.3183
0.5	3	0	0.14	0	0
0.5	3	5	0.14	0.21	-0.2437
0.5	4	0	0.12	0	0
0.5	4	5	0.12	0.18	-0.17905
0.5	5	0	0.1	0	0
0.5	5	5	0.1	0.15	-0.12434
0.5	6	0	0.08	0	0
0.5	6	5	0.08	0.18	-0.07958
0.5	7	0	0.06	0	0
0.5	7	5	0.06	0.09	-0.04476
0.5	8	0	0.04	0	0
0.5	8	5	0.04	0.18	-0.01989
0.5	9	0	0.02	0	0
0.5	9	5	0.02	0.03	-0.00497
0.5	10	0	0	0	0
0.5	10	5	0	0	0
5	0	0	0.2	0	0
5	0	5	0.2	0.3	-0.49735
5	1	0	0.18	0	0
5	1	5	0.18	0.27	-0.40285
5	2	0	0.16	0	0
5	2	5	0.16	0.24	-0.3183

VEHICLE PARTS AND PROPERTIES MODELS

Mach	Axial position	Total AoA	CA	CN	CM
5	3	0	0.14	0	0
5	3	5	0.14	0.21	-0.2437
5	4	0	0.12	0	0
5	4	5	0.12	0.18	-0.17905
5	5	0	0.1	0	0
5	5	5	0.1	0.15	-0.12434
5	6	0	0.08	0	0
5	6	5	0.08	0.12	-0.07958
5	7	0	0.06	0	0
5	7	5	0.06	0.09	-0.04476
5	8	0	0.04	0	0
5	8	5	0.04	0.06	-0.01989
5	9	0	0.02	0	0
5	9	5	0.02	0.03	-0.00497
5	10	0	0	0	0
5	10	5	0	0	0

Assuming the input in the previous table, the figure below shows the graphical representation of the cumulative values (axial, total normal and pitch moment) for a total angle of attack of 5°.

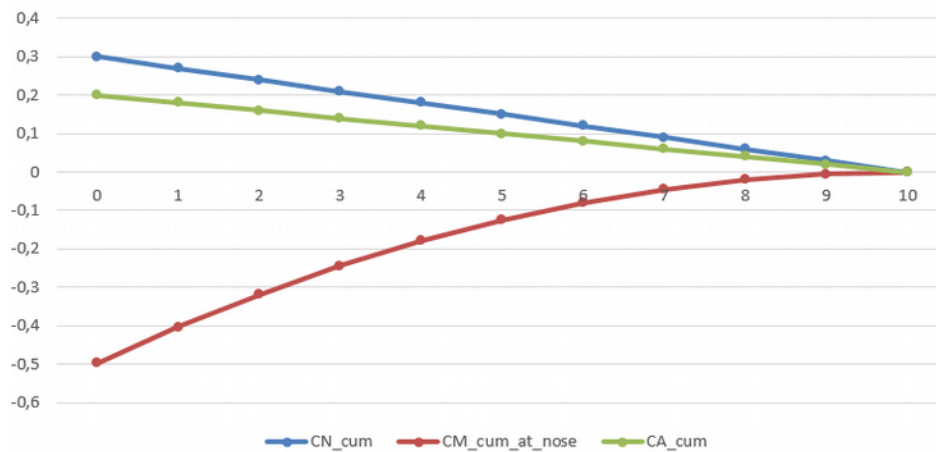


Fig. 3.60: Cumulative aerodynamic coefficients w.r.t. the vehicle length for total AoA = 5

Computation

The vehicle is divided in 30 slices with a node in the middle of each slice.

At each node, the local coefficients are computed using the following equations:

$$C_{A_i}(x, M, \alpha_t(x)) = C_{A_{cum_i}}(x, M, \alpha_t(x)) - C_{A_{cum_{i+1}}}(x, M, \alpha_t(x)) \quad (3.170)$$

VEHICLE PARTS AND PROPERTIES MODELS

$$C_{N_{t_i}}(x, M, \alpha_t(x)) = C_{N_{cum_i}}(x, M, \alpha_t(x)) - C_{N_{cum_{i+1}}}(x, M, \alpha_t(x)) \quad (3.171)$$

$$C_{M_i}(x, M, \alpha_t(x)) = C_{M_{cum_i}}(x, M, \alpha_t(x)) - C_{M_{cum_{i+1}}}(x, M, \alpha_t(x)) + \left(C_{N_{t_i}}(x, M, \alpha_t(x)) \cdot \frac{(x_m - x_i)}{L_{ref}} \right) \quad (3.172)$$

where $\alpha_t(x)$ is the total angle of attack of the node at x position x , M is the Mach number, x_m is the position of the nose node provided by the user and x_i is the position of the node at position x .

Note: The number of nodes assumed by ASTOS ($n=30$), might be different from the user input number of nodes m provided in the table.

In case of flexible components, the local total angle of attack includes also flexible contributions such as:

$$\alpha_t(x) = \alpha_{t_{RIGID}}(x) + \alpha_{t_{FLEX}}(x) \quad (3.173)$$

For further details on the flexibility see Section 3.3.1.1.

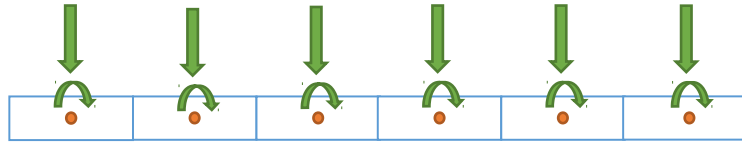


Fig. 3.61: Local force and moment vectors applied to the nodes along the vehicle length

The aerodynamic forces applied at each node (see Fig. 3.61) in the total angle of attack frame (TA) are computed as follows:

$$\begin{cases} F_{x_i} = -\frac{1}{2} \cdot \rho \cdot v^2 \cdot S_{ref} \cdot C_{A_i}(x, M, \alpha_t(x)) \\ F_{y_i} = 0 \\ F_{z_i} = -\frac{1}{2} \cdot \rho \cdot v^2 \cdot S_{ref} \cdot C_{N_{t_i}}(x, M, \alpha_t(x)) \end{cases} \quad (3.174)$$

where ρ is the air density, v is the vehicle velocity and S_{ref} is the *reference area*. Finally the aerodynamic moments applied at each node in the total angle of attack frame (TA) are defined as follows:

$$\begin{cases} T_{x_i} = 0 \\ T_{y_i} = -\frac{1}{2} \cdot \rho \cdot v^2 \cdot S_{ref} \cdot L_{ref} \cdot C_{M_i}(x, M, \alpha_t(x)) \\ T_{z_i} = 0 \end{cases} \quad (3.175)$$

3.5.2 Aerothermodynamics models

Description

VEHICLE PARTS AND PROPERTIES MODELS

The aerothermodynamics model computes the heat flux acting on a vehicle while moving through the atmosphere. An aerothermodynamics model may be considered as:

- **Vehicle configuration**
- **Active oxidation constraint**, see Section 6.3
- **Heat flux constraint**, see Section 6.3
- **Chebyshev cost function**, see Section 7.1
- **Heat flux cost function**, see Section 7.6
- Integrated **heat load**, see Section 4.5.3.5
- Part of the integrated **average drag deviation**, see Section 4.5.3.6

This model does not require the definition of an atmosphere or aerodynamics model.

Configuration

Different aerothermodynamics models are available:

- *None* means no heat flux is computed and its value is set to zero. This is useful for pure orbital missions.
- *Convective* heat is normally used for launch and re-entry vehicles or aero-assisted missions depending on the velocity. This model is explained in Section 3.5.2.1.
- *Radiative* heat may occur during aero gravity-assist maneuvers with relatively high radiative heat. A detailed description is given in Section 3.5.2.2.
- *Total* heat flux combines both previous options and considers convective as well as radiative heat flux. They are detailed in subchapter Section 3.5.2.1 and Section 3.5.2.2, respectively.
- *Emissivity* is used to compute the wall temperature. It is optional and the default value is 0.8. The input of the emissivity coefficient is only used for hypersonic heat flux and during the computation of a heat flux path constraint.

Background

The wall temperature is computed from the heat flux $\dot{Q}$ using Boltzmann's law of radiation

$$T \approx \sqrt[4]{\frac{\dot{Q}}{\sigma \varepsilon}} \quad (3.176)$$

with the Boltzmann constant σ of $5.67 \times 10^{-8} W / m^2 K^4$ and the emissivity ε .

3.5.2.1 Convective heat flux

Description

The convective heat flux is the transfer of heat from one place to another and is often the dominant form of heat transfer in gases like an atmosphere. It depends on the temperature difference between the body and the adjacent fluid (i.e. gas).

Configuration

The following settings are available for this model:

- *Free Stream Enthalpy* is normally used for launchers after a fairing is jettisoned. It uses a heat flux constant of 0.5, a density exponent of 1 and a velocity exponent of 3.
- *Supersonic* is useful for Mach regimes of 1.2 to 5.0 and requires the following input:
 - *Heatflux constant* must be given in W/m^2
 - *Density exponent* is the exponent of the density ratio
 - *Velocity exponent* is the exponent of the velocity ratio
- *Supersonic Enthalpy* is useful for Mach regimes of 1.2 to 5.0. The following input is required:
 - *Enthalpy*
 - *Nose radius* is the radius of the nose of the body
 - *Reference density* is the density at sea level
 - *Reference velocity* is a reference velocity value
 - *Density exponent* is the exponent of the density ratio
 - *Velocity exponent* is the exponent of the velocity ratio

In this model the heat flux constant is computed with given enthalpy, nose radius, reference density, reference velocity and the exponents.

- *Hypersonic* is useful for Mach numbers higher than 5
 - *Nose radius* is the radius of the nose of the body
 - *Heatflux constant* must be given in W/m^2

Convective heat flux in hypersonic and supersonic region is used in re-entry or aero-assisted missions depending on the velocity.

Background

Convective heat flux can be computed at the stagnation point in supersonic or hypersonic region. The Detra-Kemp-Riddell formula is used for the supersonic heat flux [74], where two formulations are provided:

$$\text{Supersonic Enthalpy: } \dot{Q} = \frac{h}{\sqrt{r_N / r_{dim}}} \left(\frac{\rho}{\rho_{ref}} \right)^n \left(\frac{V_a}{V_{ref}} \right)^m \quad (3.177)$$

$$\text{Supersonic: } \dot{Q} = C \rho^n V_a^m \quad (3.178)$$

where ρ is the free-stream density of the atmosphere, ρ_{ref} the density at sea level, V_a is the air path speed and V_{ref} a constant reference velocity value. r_N is the nose radius, r_{dim} a factor to make the nose radius dimensionless and h an empirical parameter of the enthalpy. C can be computed out of Eq. 3.177 or is a given model constant.

VEHICLE PARTS AND PROPERTIES MODELS

Common values for the parameters are:

- $C = 1.647 \cdot 10^{-4}$, $n = 0.5$, $m = 3.0$,
- $C = 1.155 \cdot 10^{-4}$, $n = 0.5$, $m = 3.05$,
- $C = 9.3276 \cdot 10^{-4}$, $n = 0.5$, $m = 3.15$,
- $C = 0.5$, $n = 1.0$, $m = 3.0$ (free-stream enthalpy)

A modified Fay-Riddell formula [63] is used to compute the heat flux on a body at hypersonic flow conditions:

$$\dot{Q} = C \sqrt{\left(\frac{\rho V_a^2}{r_N} \right)} \left(\frac{V_a^2}{2} + 1004 (T - T_{\text{Wall}}) \right) \quad (3.179)$$

where ρ is the free-stream density of the atmosphere, V_a is the air path speed, r_N is the nose radius and T is the free stream temperature, T_{Wall} the wall temperature, which is computed solving the fourth order polynomial to ensure an equilibrium between the convective heat transferred and the radiative heat emitted at T_{Wall} , and C is a model parameter given in $(\sqrt{\text{kg}}) / m$. A typical value of C is $3.53 \cdot 10^{-4}$.

3.5.2.2 Radiative heat flux

Description

Radiative heat transfer takes place in form of electromagnetic waves mainly in the infrared region and can be described by reference to the 'black body'. All bodies radiate energy in the form of photons moving in a random direction, with random phase and frequency. When radiated photons reach another surface, they may either be absorbed, reflected or transmitted.

Configuration

The following input is required for the radiative heat flux:

- *Heatflux constant* must be given in W/m^2
- *Density exponent* is the exponent of the density ratio
- *Nose radius* is the radius of the nose of the body
- *Nose radius exponent* is optional and defines the exponent of the nose radius. When *disabled*, equation Eq. 3.181 is used for its computation.
- *Coefficients* is the tabular data of $f(V_a)$. It is a function of the air path speed. More information about tabular data is provided in Profile Data Tables in User Manual

Background

According to [59], the radiative heat flux is

$$\dot{Q} = C r_N^a \rho^n f(V_a) \quad (3.180)$$

where C is the heat flux constant that depends on the atmosphere, r_N is the nose radius, ρ is the free-stream density of the atmosphere and $f(V_a)$ are tabulated values that are functions of the air path speed V_a and the atmospheric composition. The exponent a can be either a constant or a function of r_N and V_a :

$$a = 1.072e - 6V_a^{-1.88}\rho^{-0.325} \quad (3.181)$$

Common values for the parameters are:

- Mars: $C = 2.35$, $a = 0.526$, $n = 1.19$,
where $10 < V_a < 16 \text{ km/s}$, $6.66e-5 < \rho < 6.31e-4 \text{ kg/m}^3$, $0.3 < r_N < 3 \text{ m}$
- Earth: $C = 4.736$, $a = 1.072e-6 V_a^{-1.88} \rho^{-0.325}$, $n = 1.22$,
where $6.5 < V_a < 9 \text{ km/s}$, $1.0e-4 < \rho < 1.0e-3 \text{ kg/m}^3$, $0.3 < r_N < 3 \text{ m}$
or, if $1 \leq r_N \leq 2$, $a \leq 0.6$, if $2 \leq r_N \leq 3$, $a \leq 0.5$

3.6 Electrical Power System Modelling

Electrical Power System Modelling Approach

Description

ASTOS supports the modelling of the electrical power system on board a vehicle. Solar generator and batteries models can be connected to the models of other electrical consumers in order that the required battery capacity and the length of the battery charge and discharge cycles can be estimated for a mission. The electrical power system model in ASTOS requires the following component models to be present in a scenario:

- At least one *Solar generator*, see Section 3.6.3 for details
- At least one *Battery*, see Section 3.6.1 for details
- At least one *Power Control and Distribution Unit* (PCDU), see Section 3.6.2 for details
- At least one electrical consumer, i.e. any actuator, component, sensor or data component for which the option *Electrical power* is set to *Enabled*. The *Electrical power* setting is explained in detail in Section 3.1.3.

Building an Electrical System

The necessary components required to build an electrical system in ASTOS are outlined above. All of these models need to be configured and added to the vehicle. In the *Power Control and Distribution Unit* all circuits need to be given an identifier and the voltage. If any vehicle part except the PCDU is selected in the Vehicle Builder its configuration panel has a *Power* tab with the following options:

- If a consumer is selected the tab shows a drop-down menu *PCDU* with available PCDUs and *Power circuits* of the selected PCDU.

VEHICLE PARTS AND PROPERTIES MODELS

- If a *Solar generator* or *Battery* is selected only the *PCDU* needs to be selected.

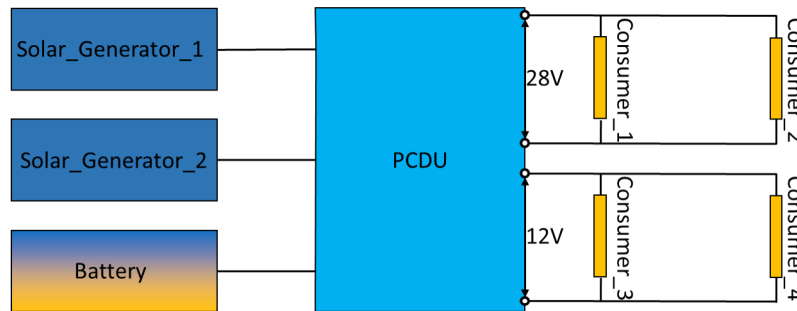


Fig. 3.62: Example of an electrical system using 2 solar generators as power sources, a battery as power source and power drain and 4 electrical consumers organized in two circuits with different voltages.

The above block diagram is only an example. The number of solar generators, batteries and circuits is defined by the user as well as the number of consumers in each circuit. The power output of solar generators varies based on the irradiation of sunlight and the operating temperature of the cells. Also the power provided by the battery is not constant. Power sources are directly connected to the PCDU and PCDU transforms the electrical power to the levels required in the different circuits. Each vehicle part can only be assigned to one PCDU. If the vehicle has more than one PCDU each of them is connected to its own (and separate) power sources and consumers.

Electrical Power System Element Models

ASTOS is capable of modelling the electrical power management on board any vehicle defined in a scenario. Any *Sensor*, *Actuator* or *Component* model for which the *Electrical power* setting in its respective *Configuration Panels* is set to *enabled* is a consumer of electrical power. In addition to the consumer models ASTOS offers models for electrical power generation, storage and management which are briefly explained below:

- The *Solar Generator* is a configurable model of a solar panel generating electrical power based on photovoltaic diodes, see Section 3.6.3 for details
- The *Battery* is a model of an electrical power storage element which simulates charge and discharge characteristics based on a selectable battery technology, see Section 3.6.1 for details
- The *Power Control and Distribution Unit* models the electrical circuits within a vehicle and manages the connections between power generation devices and power consumers, see Section 3.6.2 for details

3.6.1 Battery Model

Description

VEHICLE PARTS AND PROPERTIES MODELS

This power component models the charge and discharge behavior of different battery technologies as given in [134].

Configuration

The configuration panel of the battery model contains the following settings:

- In the *Technology* field the battery technology is selected from the drop-down menu. The available battery technologies are:
 - *Li Ion* for the Lithium-Ion technology
 - *Lead Acid* for the Lead-Acid technology
 - *NiMH* for the Nickel-Metal-Hydrid technology
 - *NiCd* for the Nickel-Cadmium technology
- *Cells parallel* is the number of battery cells in parallel configuration
- *Cells series* is the number of battery cells in serial configuration

The remaining battery model input parameters correspond to the discharge characteristics found in the battery data sheets:

- *Cell capacity* is the maximum cell capacity which is equal to *discharge_time* times the *discharge_current*
- *Fully charged voltage* is battery voltage at the beginning of a discharge cycle
- *Exponential voltage* is battery voltage at the end of the exponential zone.
- *Exponential capacity* is the capacity at the end of the exponential zone.
- *Nominal voltage* is the battery voltage at the end of the nominal zone (when the voltage starts to drop abruptly).
- *Nominal capacity* is the capacity at the end of the nominal zone.
- *Rated voltage* is the rated/nominal battery voltage.
- *Discharge current* is the discharge current used to obtain the typical discharge behavior.
- *Cell resistance* is the internal battery cell impedance and can be *Defined by Value* (constant resistance) or *Profile* to account for varying resistance due to temperature or state-of-charge. For more information about profile data tables see Profile Data Tables in the ASTOS User Manual.
- *Max allowed drain* is optional and specifies the maximum fraction of the *Cell capacity* which can be extracted from the battery
- *Structural mass* is optional and specifies the mass of the battery
- *Thermal characteristics* is a generic setting which is available if the corresponding feature in the ASTOS software has been installed and then enabled in the scenario *Active features*. Please refer to Section 3.1.4 for setup details.

Determining the battery model parameters using a battery data sheet

The values for the settings described above can be acquired by the user by reading the discharge characteristics in the battery data sheet. The model assumes that the extracted

parameters are the same for charge and discharge. Fig. 3.63 shows an example of a battery discharge plot and how the values for the battery model can be extracted from it.

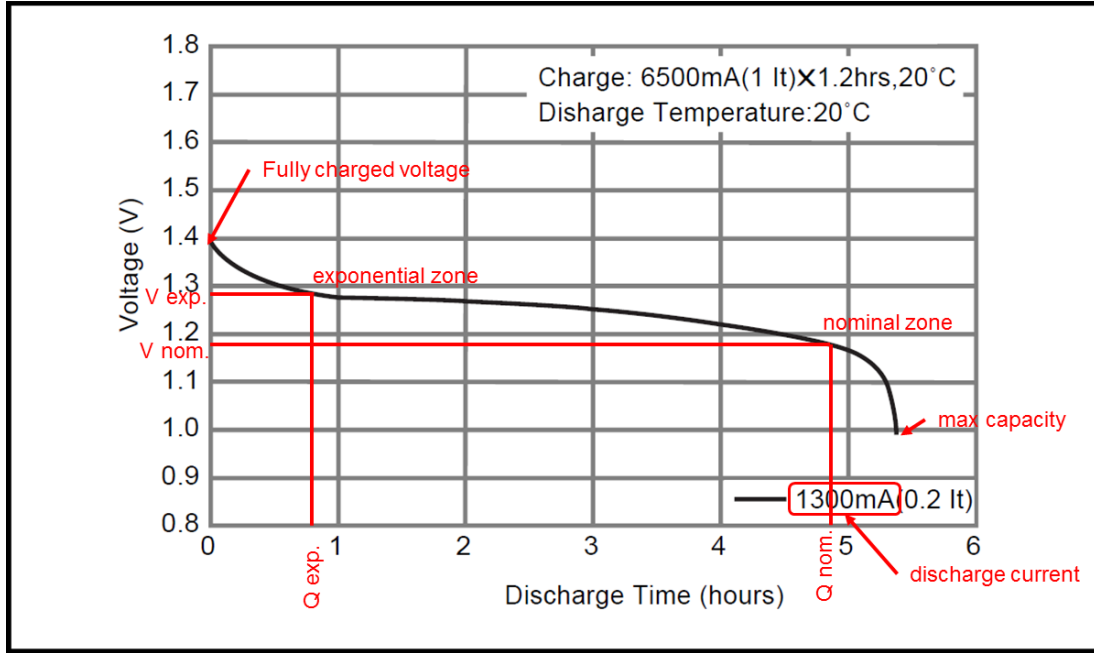


Fig. 3.63: Typical discharge characteristics of a Panasonic NiMH HHR650D battery with fully charged voltage, exponential voltage (V_{exp}), exponential capacity (Q_{exp}), nominal voltage (V_{nom}), nominal capacity (Q_{nom}) and cell capacity as well as discharge current.

To obtain the capacity values needed by this model the discharge time has to be multiplied by the discharge current, e.g. in the above figure the nominal capacity is extracted at 4.8 hours of discharge time. Together with the discharge current of 1300mA this gives a capacity of approximately 6.25Ah.

Background

The battery charging and discharging behaviour depend on the battery technology. For each supported battery technology, the corresponding equations used to model the battery voltage, state-of-charge and applied current are:

■ Li Ion

Discharge:

$$V_{batt} = E_0 - R \cdot i - K \cdot \frac{Q}{Q - it} \cdot (it + i) + Ae^{(-Bit)} \quad (3.182)$$

Charge:

$$V_{batt} = E_0 - R \cdot i - K \cdot \frac{Q}{it - 0.1Q} \cdot i - K \cdot \frac{Q}{Q - it} \cdot it + Ae^{(-Bit)} \quad (3.183)$$

■ Lead Acid

Discharge:

$$V_{batt} = E_0 - R \cdot i - K \cdot \frac{Q}{Q - it} \cdot (it + i) + Exp(t) \quad (3.184)$$

Charge:

$$V_{batt} = E_0 - R \cdot i - K \cdot \frac{Q}{it - 0.1Q} \cdot i - K \cdot \frac{Q}{Q - it} \cdot it + Exp(t) \quad (3.185)$$

■ *NiMH* and *NiCd*

Discharge:

$$V_{batt} = E_0 - R \cdot i - K \cdot \frac{Q}{Q - it} \cdot (it + i) + Exp(t) \quad (3.186)$$

Charge:

$$V_{batt} = E_0 - R \cdot i - K \cdot \frac{Q}{|it| - 0.1Q} \cdot i - K \cdot \frac{Q}{Q - it} \cdot it + Exp(t) \quad (3.187)$$

where

- V_{batt} is the battery voltage [V]
- E_0 is the battery constant voltage [V] (computed based on the given discharge characteristics)
- R is the battery internal resistance [Ohm]
- i is the battery current [A] with $i > 0$ meaning discharging the battery and $i < 0$ meaning charging the battery.
- K is the battery polarisation constant [V/(Ah)] or polarisation resistance [Ohm] (computed based on the given discharge characteristics)
- Q is the battery capacity [Ah]
- it is the extracted capacity [Ah] with $it=0$ meaning fully charged and $it=Q$ fully discharged. it is modelled as a state and integrated over time.
- A is the exponential zone amplitude [V] (computed based on the given discharge characteristics)
- B is the exponential zone time constant inverse [(Ah)⁻¹] (computed based on the given discharge characteristics)
- $Exp(t)$ is the exponential zone voltage with initial value $Exp(t_0) = A$. This models the hysteresis phenomenon between charge and discharge in the exponential zone.

More detailed information about the mathematical equations of the battery model can be found in [134].

3.6.2 Power Control and Distribution Unit Model

Description

The Power Control and Distribution Unit (PCDU) distributes, transforms and controls the voltage, current and power provided and consumed by the electrical equipment on board a vehicle. Thus all power sources (e.g. batteries and solar panels) and consumers are connected to the PCDU and can be organized in several circuits.

VEHICLE PARTS AND PROPERTIES MODELS

In the ASTOS PCDU model the number of circuits and the voltage specified in each circuit are user input. Consumers are assigned a certain electrical power consumption and are added to one of the PCDU circuits making it possible to compute the current required by the consumers. Since the solar panels' voltage and electrical current are not constant all power sources are connected directly to the PCDU in order that their output is then transformed to the required levels for each defined circuit.

Configuration

General

The PCDU model settings are as follows:

- *Own consumption* is the power consumption of the PCDU itself
- *Transformation efficiency* factor can be used to model losses between the electrical power generated in the solar panels and the actually available power inside the PCDU. Values are between 0.0 and 1.0 with 1.0 meaning no losses.
- The **Add** button is used to add a new circuit to be managed by the PCDU model. Each circuit requires a unique *Circuit ID* and the circuit *Voltage* to be defined by the user. The **Remove** button removes the corresponding circuit from the PCDU configuration.
- *Structural mass* is the mass of the PCDU hardware
- *Thermal characteristics* is a generic setting which is available if the corresponding features in the ASTOS software has been installed. Please refer to Section 3.1.4 for setup details.

Adding power sources and consumers

The PCDU model is only a template and does not assign the model to a vehicle. All connections to a PCDU are defined in the **Vehicle Builder**. See the The Vehicle Builder: Vehicles, Assemblies, Elements chapter in the User Manual book on how to realize a vehicle from defined templates.

Note: All power generation and power consumer components can only be connected to one PCDU. At least one PCDU is has to be defined within a vehicle in order to build an electrical system model.

Background

Computing the current in each circuit

The electrical current required in each circuit can be computed from the defined electrical power of a consumer and the voltage in the circuit the consumer is connected to. Summing up the current required for each consumer gives the total current in the circuit (assuming all consumers in a parallel configuration within the circuit).

Discharging and recharging connected batteries

The difference between the current from the solar panels available inside the PCDU (see *Transformation efficiency* in the Configuration paragraph) and the required current of all the defined circuits is applied to the connected batteries. If the solar panels provide more current than required in total by the consumers, batteries are recharged (if not already fully charged).

Batteries are discharged if more current is required than the current provided by the solar panels.

Coupling with thermal systems

The electrical power from the solar panels available inside the PCDU is used to supply all consumers in all defined circuits and to recharge the batteries. If all batteries are fully charged and the electrical power of the solar panels is greater than what is required then the difference is converted to thermal power and is added to the thermal node of this PCDU in case the PCDU *Thermal characteristics* are enabled.

3.6.3 Solar Generator Model

Description

A solar generator is a source for electrical power which is generated by a number of photovoltaic (PV) cells. The generated electrical power depends on the irradiated sunlight and on the temperature of the photovoltaic (solar) cells. It is designed in such a way that the user can transfer with little effort the specifications of a particular solar cell/panel into the GUI. It is standard practice that such specifications are based on the number of cells in the panel and do not contain the solar panel area as an input parameter. Therefore such a parameter is not available in the *ASTOS* solar generator model. In case that multiple solar panels (e.g. a solar array) of the same type shall be modeled, multiple instances of the solar generator model have to be inserted in the vehicle builder in the *Vehicles & POIs Definition (Modelling tree)*.

Configuration

The solar generator model settings are given below and their values are assumed to be at Standard Test Conditions (STC) with an irradiation of 1000 W/m^2 and a temperature of 25°C :

- *Cells parallel* is the number of photovoltaic (solar) cells connected in parallel
- *Cells series* is the number of photovoltaic (solar) cells connected in series
- *Short circuit current* is the maximal current in short circuit condition of a single photovoltaic cell. It is produced when the solar cell is not connected to a load but when the plus and minus wires are directly connected to each other.
- *SC current temperature coefficient* accounts for the short circuit current changes with temperature. The short circuit current slightly increases with the temperature. If the value is set to 0.0 then the temperature has no impact on the short circuit current. Please refer to the explanations of the *Thermal characteristics* setting below for additional details about thermal modelling in *ASTOS*.
- *Open circuit voltage* is the maximal voltage of a single solar cell in open circuit conditions, i.e. the voltage with no load (current is zero) on the solar cell. It is the value measured with a voltmeter across the plus and minus poles.
- *OC voltage temperature coefficient* accounts for the voltage changes with temperature. The open circuit voltage decreases with the temperature. If the value is set to 0.0 then the temperature has no impact on the open circuit voltage. Please refer to the explanations

of the *Thermal characteristics* setting below for additional details about thermal modelling in ASTOS.

- *Series resistance* is a typical parameter of a solar cell and models all combined resistances of contacts, grids and layers. It determines the slope of the characteristic I-V curve of a solar cell near the open circuit voltage when the current is zero. Typical values are below 1 Ohm. In case the solar cell is supposed to be modeled as ideal, the value should be set to zero. The default value is 0.5 Ohm.
- *Shunt resistance* is a typical parameter of a solar cell modeling the leakage current of the p-n junction. It can be determined through the slope of the characteristic I-V curve of a solar cell near the short circuit current when the voltage is zero. The shunt resistance is typically 100-1000 times larger than the series resistance and typical values are in the order between few dozen and few hundred Ohm. In case the solar cell is supposed to be modeled as ideal, the value should be set to very high number, e.g. 1e10 Ohm. The default value is 100 Ohm.
- *Ideality factor* is a typical parameter of a solar cell expressing how closely the cell follows the ideal diode equation and is a measure for the material quality. A factor value of 1.0 means an ideal match to theory. Typical values of the ideality factor are between 1 and 2.
- *Solar generator scaling factor* is an optimizable scaling factor applied simultaneously to both the short circuit current and the open circuit voltage under standard test condition. This factor is used for optimization purposes in order to estimate the total required electrical power for the solar generator.
- *Structural Mass* is the mass of the solar generator, i.e. PV module.
- *Thermal characteristics* is a generic setting. Please refer to Section 3.1.4 for setup details. If *Thermal characteristics* are *disabled* the solar generator temperature is set to the standard test condition (STC) value of 25°C. In case that the *Thermal characteristics* are disabled or the thermal model is not activated, the solar generator model will not consider the any temperature effects on its efficiency. Thus neither the *SC current temperature coefficient* nor the *OC voltage temperature coefficient* will have any effect on the generated power by the solar generator model.

Note: The temperature coefficients provided by manufactures of PV modules are sometimes normalized and given in %/°C. ASTOS requires non-normalized coefficients. In order to obtain non-normalized coefficients, the normalized coefficients have to be multiplied by the corresponding values at STC, i.e. the normalized OC temperature coefficient [%/°C] times the OC voltage [V]. More detailed information about temperature coefficients in PV modules and arrays can be found in [135]

Background

A photovoltaic cell can be modeled by an equivalent diode circuit, in particular a single diode based model considering a shunt resistance and a series resistance. The corresponding circuit is shown in Fig. 3.64.

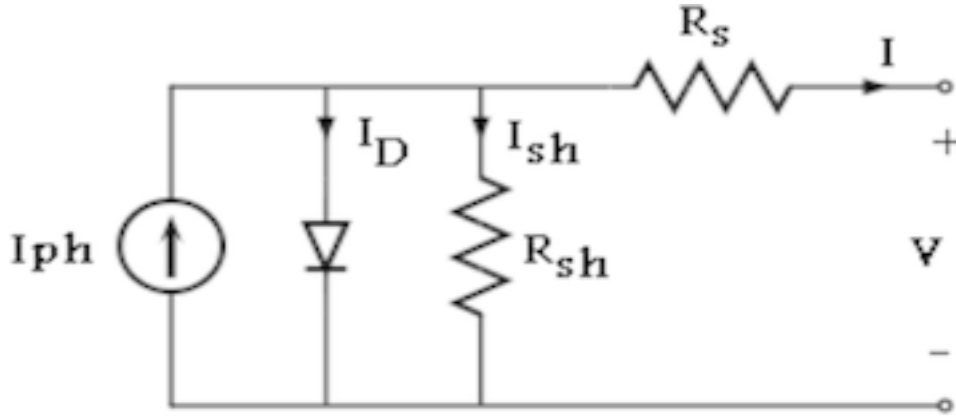


Fig. 3.64: Equivalent circuit of a solar cell with photo current I_{PH} , diode current I_D , shunt current I_{SH} , shunt resistance R_{SH} , series resistance R_S , terminal current I and terminal voltage V .

The mathematical expression of the current produced by the solar cell (i.e. operating/terminal current) is equal to the current produced by the current source minus the currents through the diode and the shunt resistor, both considered as losses. This is known as the characteristic equation of a solar cell:

$$I = I_{PH} - I_D - I_{SH} \quad (3.188)$$

with the photogenerated current I_{PH} , the diode current I_D and the shunt current I_{SH} . Both I_D and I_{SH} are considered as losses. The photogenerated current is produced by the current source and is defined by the solar cell equation under short circuit condition ($V=0$)

$$I_{PH} = (I_{SC\_STC} + K_I \cdot (T - T_{STC})) \cdot \frac{G}{G_{STC}} \quad (3.189)$$

based on the short circuit current I_{SC_STC} under standard test condition plus the temperature effect and finally scaled with the actual irradiance G over the irradiance in standard test condition G_{STC} . The temperature effect is modeled by the temperature coefficient of the current K_I and the difference between the operating temperature T and the temperature under standard test condition T_{STC} (298.15 K). For solar cells the temperature dependence on the short circuit current is positive, thus the current increases with higher temperatures.

The diode current I_D is given by the Shockley diode equation

$$I_D = I_0 \cdot \left\{ \exp\left(\frac{V_D}{n \cdot V_T}\right) - 1 \right\} \quad (3.190)$$

where I_0 is the reverse saturation current and n is the ideality factor, a measure for the material quality and ranges between 1 and 2 (1 represents an ideal solar cell). The diode voltage V_D is defined by

$$V_D = V + I \cdot R_S \quad (3.191)$$

with the operating/terminal voltage V and the loss due to the series resistance R_S . All combined resistances of contacts, layers and grids are modeled by the series resistance, a quite small value (e.g. less than 1 Ohm). The series resistance can be determined through the slope of the I-V curve near the open circuit voltage V_{OC} ($I=0$).

Besides, the thermal voltage V_T is temperature dependent and defined by

$$V_T = \frac{k \cdot T}{q} \quad (3.192)$$

where T is the operating temperature, k is the Boltzmann's constant and q is the elementary charge. The loss due to the shunt resistance R_{SH} is defined by Ohm's law

$$I_{SH} = \frac{V_D}{R_{SH}} \quad (3.193)$$

The shunt resistance models the leakage current of the p-n junction of the solar cell and is typically a large value in the order of few dozen Ohm (lower quality) up to few hundred Ohm (higher quality). It can be determined through the slope of the I-V curve near the short circuit current I_{SC} ($V=0$). The reverse saturation current I_0 , also known as dark or saturation current of the diode, is based on the solar cell equation at short circuit condition ($I=0$) and given by

$$I_0 = \frac{I_{PH} - \frac{V_{OC}}{R_{SH}}}{\exp\left(\frac{V_{OC}}{nV_T}\right) - 1} \quad (3.194)$$

with the actual open circuit voltage V_{OC} . Being an important characteristic of the diode, the dark saturation current is smaller for higher material quality, but it always increases significantly with temperature and doubles for each ~ 10 K in operating temperature. The actual voltage under open circuit condition depends on the actual operating temperature T as well and is defined as

$$V_{OC} = (V_{OC\_STC} + K_V \cdot (T - T_{STC})) \quad (3.195)$$

based on the open circuit voltage V_{OC_STC} under standard test condition plus the temperature effect modeled by the temperature coefficient of the voltage K_V and the difference between the operating temperature T and the temperature under standard test condition T_{STC} (298.15 K). There is a negative temperature dependence of a solar cell on the open circuit voltage and the voltage decreases with higher temperatures.

Finally, the characteristic solar cell equation (see Eq. 3.188) can be rewritten as

$$I = I_{PH} - I_0 \cdot \left\{ \exp\left(\frac{q \cdot (V + I \cdot R_S)}{n \cdot k \cdot T}\right) - 1 \right\} - \frac{(V + I \cdot R_S)}{R_{SH}} \quad (3.196)$$

This equation can be represented as typical I-V curve when the voltage is preset between 0 and V_{OC} . Since it is a circular equation for the current I while the voltage V is also unknown a-priori during operation, this equation is solved iteratively in such a way that the generated power, i.e. the product of I and V , is maximized at all times. This approach is known as *maximum power point tracking* and yields the current I_{MP} and voltage V_{MP} of the maximum power point. To receive in general the power output of the whole solar generator model, the actual operating current and voltage need to be multiplied with the number of solar cells in series N_S and parallel N_P , given as

$$P = N_S \cdot N_P \cdot I \cdot V \quad (3.197)$$

Note, the solar generator output voltage is always smaller than the open circuit voltage, and the output current is always smaller than the short circuit current.

All relevant quantities of the solar generator model and the solar cell equation are summarized as:

- G : actual irradiance. The computation of the actual irradiance is explained in Section 3.1.5.

VEHICLE PARTS AND PROPERTIES MODELS

- G_{STC} : irradiance of 1000 W/m² under standard test condition
- I : operating/terminal current
- I_0 : reverse saturation current
- I_D : diode current
- I_{MP} : current at maximum power point
- I_{PH} : photogenerated current
- I_{SC_STC} : short circuit current under standard test conditions
- I_{SH} : shunt current
- k : Boltzmann constant
- K_i : temperature coefficient of the current
- K_V : temperature coefficient of the voltage
- n : ideality factor, typically between 1 and 2
- N_S : number of cells in series configuration
- N_P : number of cells in parallel configuration
- q : electron charge
- R_S : series resistance, typically below 1 Ohm
- R_{SH} : shunt resistance, typically between few dozen and few hundred Ohm
- T : actual operating temperature
- T_{STC} : temperature of 298.15 K under standard test condition
- V : operating/terminal voltage
- V_D : diode voltage
- V_{MP} : voltage at maximum power point
- V_T : thermal voltage
- V_{OC} : actual open circuit voltage
- V_{OC_STC} : open circuit voltage under standard test conditions

3.7 Data System Modelling

Data System Modelling Approach

Description

Data System can be used to model a data bus system with data being written to a data storage device and stored data being transmitted to other vehicles and ground stations. The data storage capacity needed on board a satellite depends on multiple factors like the rates at which the various components generate data and the viewing windows between transmitters on the vehicle and receivers on relay satellites or ground stations. *ASTOS* can be used to model the data rates of sensors, actuators etc. and to store this data onto a

VEHICLE PARTS AND PROPERTIES MODELS

Data Buffer. A *Transmitter* or *Transceiver* (see Section 3.4.7 for details) can then be used to transmit data to free up data storage capacity.

Note: Any computations involving this model require the corresponding feature to be present in the ASTOS license.

Building a working data system

The procedure below describes how to successfully simulate the data system in ASTOS:

- Check that the *Data system* feature is *enabled*.
- Add and configure a *Data Buffer* model in the scenario and add an instance of it to the vehicle.
- Add and configure a sensor model of type *Transmitter* or *Transceiver* to the scenario and add an instance of it to the vehicle. The *Maximum data rate* needs to be *enabled*.
- Add and configure another sensor model of type *Receiver* or *Transceiver* to the scenario and add an instance of it to a ground station or relay satellite.
- At least one additional component has to be present in the vehicle except the *Data Buffer*. In case this additional component is an *Actuator* or a *Component*, the *Data rate* should be defined. Otherwise the data rate is 0 by default. If the additional component is a *Sensor* the data rate is computed from the settings specific for each sensor type.
- Select at least one of the component instances in the vehicle builder which generate data and use the data tab to select the *Data Buffer*. This will cause data being generated in this equipment to be written to the data buffer.
- Select the instance of the *Data Buffer* in the vehicle builder and use the data tab to select the previously configured *Transmitter* or *Transceiver*. The selected *Transmitter* is used to transmit data and free up on board memory.
- In the dynamics settings of the vehicle select the instance of the transmitter. Select at least one *Receiver* or *Transceiver* in the panel *Link partners*. This does not guarantee that data is actually transmitted, but only defines the receiving component of a signal transmission for which mutual visibility is checked.

The selected *Transmitter* can only transmit to any of the specified *Link partners* if there is mutual visibility between transmitter and receiver and if they are technically compatible (demodulation, channel band etc.). If both conditions are fulfilled then the maximum data rate of the transmitter is used. Adding an analysis of type *Data Timeliness* (see Section 5.2 for details) to the scenario additionally provides information about the delay between the time the data is generated on-board and the time this data is transmitted to any of the *Link partners*.

Data System Component Models

The following data component models are available in ASTOS:

- *Data Buffer Model*, see Section 3.7.1
- *Transmitter* or *Transceiver*, see Section 3.4.7

- *Receiver*, see Section 3.4.7

3.7.1 Data Buffer Model

A data buffer serves as data storage element in the vehicle. All sensors, actuators, power components and vehicle components have the option to specify a data rate at which they generate data. This data is stored in buffers until it can be transmitted to other vehicles or ground stations.

Configuration

The *Data Buffer* has the following settings:

- *Capacity* is the data storage capacity in Bit or Byte
- *Dimensions* is an optional setting specifying the size of the data buffer component which is assumed to be a rectangular solid. The default dimension for all edges is 0.1 meter.
- *Structural mass* is optional for a data buffer. By default the buffer has no mass.
- The generic settings *Electrical power* and *Thermal characteristics* are explained in Section 3.1.3 and Section 3.1.4 respectively. They are available if the corresponding features in the ASTOS software have been installed and then enabled in the scenario *Active features*.

Background

The total data stream for the data buffer is calculated by summing up the data rates of the assigned producers and subtracting the data rate of the assigned transmitter.

3.8 Thermal System Models

Besides the thermal modelling implemented within the other the vehicle components, the following model is intended to simulate solely the thermal behavior of a component:

- *Surface Element* defines a thermal surface model which considers irradiation and emission of heat, see Section 3.8.1

3.8.1 Surface Element

Description

The *Surface Element* model is part of the thermal model in ASTOS. It can be used to model radiators of satellites. It is possible to define multiple thermal nodes within the model so that a desired temperature distribution within the surface can be simulated. The model does not

VEHICLE PARTS AND PROPERTIES MODELS

generate thermal power but can be connected with thermal connections to internal components of the satellite, e.g. on-board computers or batteries.

Configuration

The model has the following settings:

- *Surface Element* → *Dimensions* depth, width and height
- *Surface Element* → *Structural mass*
- *Thermal characteristics* allow the configuration of the thermal model of the *Surface Element*. The panel offers a more extended set of thermal settings than the settings of any other vehicle part. The explanation of the settings common to the other vehicle parts as well as background information about the thermal node model in *ASTOS* are given in Section 3.1.4. The settings specific for a *Surface Element* are:
 - *Number of x nodes*, *Number of y nodes* and *Number of z nodes* define the number of nodes along each of the component axis
 - *Shape* defines whether the shape of a surface element is a *Plate* or a *Cylinder*
 - *Specific heat resistance* of the material of the Surface element

Note: The *Thermal characteristics* settings in the *Surface Element* are only available if the **Thermal characteristics** feature is activated in the *ASTOS* GUI.

4 Dynamics Configuration

Each vehicle model may include a flight dynamics model of either the vehicle itself or of one of its parts (e.g. an actuator). The settings of a flight dynamics model can consist of one or more of the following items:

- In Section 4.1 Epoch, Time Scales and Time Formats the time and epoch modelling is detailed.
- In Section 4.2 Initial State Definition the modelling of the initial translational and rotational states of the vehicle flight dynamics is explained.
- In Section 4.3 Attitude Definition the vehicle attitude modelling (3-DoF and 6-DoF) in ASTOS is described.
- In Section 4.4 Control Laws the modelling of control laws of the vehicle attitude and also of all other controllable vehicle parts (e.g. sensor pointing) is detailed.
- In Section 4.5 Equations of Motion an explanation is provided of the translational and rotational dynamics model of the vehicle.
- In Section 4.6 Maneuvers at the Beginning of a Phase it is described how the user can instantaneously change the state of the vehicle at the beginning of a phase.
- In Section 4.7 Linearised Flexible Dynamics it is described how the user can add the linearised flexible dynamics effect (without using the multibody dynamics).

4.1 Epoch, Time Scales and Time Formats

Description

This section describes the various time-scales and date formats that are supported. They are mainly required for the general settings of a scenario (see Scenario General Settings in the User Manual book) and for the initial state settings (*Dynamics Configuration* in the *Modelling* tree, see Section 4.2).

Tip: The epoch defined in the general scenario settings specifies the mission start time. This epoch can differ from the epoch of the initial state.

Configuration

The following input parameters and options are available to define an epoch:

DYNAMICS CONFIGURATION

- *Time standard* specifies the time-scale in which an epoch is defined, i.e. *TT*, *TAI*, *UTC*, or *GPS*:
 - *UTC*: Coordinated Universal Time, the basis of civil time
 - *TAI*: International Atomic Time, the official timekeeping standard
 - *TT*: Terrestrial Time, used for solar system ephemeris look-up
 - *GPS Time*: the time system of Global Positioning System
- *Date format* specifies the date format in which an epoch is defined. The following options are available:
 - *Calendar Date* allows the specification of a Gregorian date by providing values for the *Year*, *Month*, *Day*, *Hour*, *Minute* (integer numbers) and *Second* (floating number).
 - *Time Code A* allows the specification of a Gregorian date as ASCII one line element according to CCSDS 301.0-B-4 [64]. The format is: YYYY-MM-DDThh:mm:ss.dZ, with calendar-time separator T, time code separator Z and d as decimal fraction of the second e.g. 2000-01-01T12:00:00.0Z
 - *Time Code B* [64] is similar to *Time Code A*. The format in this case is: YYYY-DDDThh:mm:ss.dZ, with calendar-time separator T, time code separator Z and d as decimal fraction of second. The day of the year is given as a value between 001-365 or -366, e.g. 2000-001T12:00:00.0Z
 - *Julian Date* represents a date by providing the elapsed days with respect to a reference epoch (Jan 1, 4713 BC, 12:00 UT)
 - *Modified Julian Date 2000* as defined by ESOC represents a format which is derived from the Julian Date. Its reference is 2000-01-01 12:00 UTC = 2451544.5 JD, (note that IAU defines it in TT)
 - *Modified Julian Date 1950* as defined by CNES represents a format which is derived from the Julian Date. Its reference is 1950-01-01 12:00 UTC = 2433282.5 JD (note that Besselian year 1950 is defined at 1950-01-0.9235 TT)
 - *Modified Julian Date* as defined by SAO represents a format which is derived from the Julian Date. Its reference is 17.11.1858 00:00 UTC = 2400000.5 JD
- *(Calendar) Date* is the input field (multiple fields in case of *Calendar Date*) to set the desired value(s) for the specified date format.

Model background

The time-scales used in are defined according to the *IAU SOFA Time Scale and Calendar Tools* [66].

- *GPS Time*: GPS time matched UTC from 1980-01-01 to 1981-07-01. It is not corrected for leap seconds, thus it was 13 seconds ahead of UTC on 2000-01-01. The GPS reference epoch is 00:00 (midnight) UTC on 1980-01-06. The differences between GPS time and TAI and TT are constant at the level of some tens of nanoseconds while the difference

DYNAMICS CONFIGURATION

between GPS time and UTC changes in increments of seconds each time a leap second is added to UTC time scale.

- **TAI:** International Atomic Time (Temps Atomique International) is the official timekeeping standard. It is defined as the weighted average of the time kept by about 200 atomic clocks in over 50 national laboratories worldwide. TAI-UT1 was approximately 0 on 1958 Jan 1.
- **TCB:** Barycentric Coordinate Time is a dynamical time scale. It is the solar system barycentric coordinate time and appropriate for applications where the observer is imagined to be stationary in the solar system but with all the matter (in particular the Sun) absent. The relationship between TCB and TDB is defined as:

$$TCB = TDB + 1.55051976772 \cdot 10^{-8} \cdot (JD_{TCB} - 2443144.5003725) + 6.55 \cdot 10^{-5} s \quad (4.1)$$

- **TCG:** Geocentric Coordinate Time is a dynamical time scale and appropriate for theoretical studies of geocentric ephemerides.
- **TDB:** Barycentric Dynamical Time is a scaled form of TCB. It is used as a time scale of ephemerides referred to the barycenter of the solar system and is the independent variable of the DE405 ephemeris. TDB differs from TT by less than 2 milliseconds. The relationship between the barycentric and terrestrial time-scales is:

$$TDB = TT + 0.001657s \cdot \sin(g) + 0.000022s \cdot \sin(L - L_j) \quad (4.2)$$

$$g = 357.53^\circ + 0.98560028^\circ \cdot (JD - 2451545.0) \quad (4.3)$$

$$L - L_j = 246.11^\circ + 0.90251792^\circ \cdot (JD - 2451545.0) \quad (4.4)$$

where g is the mean anomaly of the Earth and $L - L_j$ is the difference in mean longitudes of the Earth and Jupiter. This formula is accurate to about 30 microseconds.

- **TT:** Terrestrial Time is the theoretical time scale for clocks at sea-level and used as time-scale of ephemerides for observations from the Earth's surface. This time scale was redefined in 1991 to be consistent with the SI second and the General Theory of Relativity. Originally TT was called TDT used as counterpart to TDB. For practical purposes, TT is tied to TAI through a fixed relation. *ASTOS* uses TT as time-scale for the independent variable of the ephemeris data, since the difference between TDB and TT is less than 2 milliseconds.
- **UTC:** Coordinated Universal Time is an atomic/solar time scale hybrid and the basis of civil time. UTC differs from TAI by an integer number of seconds, known as leap seconds. When required (due to the Earth's decreasing spin rate), leap seconds are introduced in UTC to keep the difference between UTC and UT less than 0.9 s. UTC was introduced in 1972. The official reference is the IERS Bulletin.
- **UT1:** Universal Time is the modern equivalent of mean solar time, and is really an angle rather than time in the physics sense. It was formerly determined by astronomical observations but today GPS satellites are used instead. This time-scale is slightly irregular and is defined by tabular data published by IERS Bulletin.

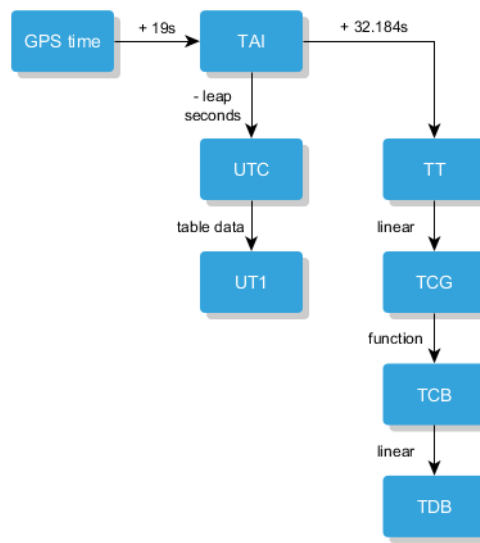


Fig. 4.1: Relationship between different time scales

4.2 Initial State Definition

Description

The initial state is a set of initial values for translational (position and velocity) and rotational (attitude and angular velocity) states. Settings are available for these object types:

- **Constellations** (see chapter Constellations in User Manual)
- **Ground Stations** (see chapter Ground Stations in User Manual)
- **Vehicles** (see chapter Vehicles in User Manual)

Configuration

The *Initial State* tab allows specifying the setup of different initial conditions:

- **Ground Stations** only require a celestial body fixed *position* defined by *altitude*, *latitude* and *longitude* (see also Section 4.2.3.1)
- **Constellations** are detailed in Section 4.2.1
- **Vehicles** have several options for the initial state definition:
 - *State type* defines the **translational** initial state with these options:
 - *Position & Velocity*, see Section 4.2.3.1
 - *Orbital Elements*, see Section 4.2.3.2
 - *Relative Motion*, see Section 4.2.3.3

DYNAMICS CONFIGURATION

- *Barycentric Rotating*, see Section 4.2.3.4
- *Attitude*, the **rotational** initial state, is only available if the *Attitude* of the first phase is specified as *State*. This input is required:
 - *Angles* describe the orientation of the vehicle body axes. Possible *Types* are:
 - *Aerodynamic Angles* are specified by *Angle of attack*, *Side slip angle* and *Airpath bank angle*
 - *Euler Angles* are specified by *Roll*, *Pitch* and *Yaw* in one of the following coordinate *Frame*: *J2000*, *L*, *LD* or *LO*.
 - *Quaternion* is specified by its elements *Qw*, *Qx*, *Qy* and *Qz* in one of the following coordinate *Frame*: *J2000* or *L*. For convenience, the quaternion elements do not need to be normalized.
- More information about attitude representation types and coordinate frames is given in Section 9.1 and Section 9.3.
- *Angle rates* describe the rotation of the spacecraft orientation. Supported *types* are:
 - *Body Angle Rates* is the rotation vector ω of the vehicle body frame with respect to the inertial frame and is expressed in components along the vehicle body axes: *Omega X*, *Omega Y* and *Omega Z*, also known as *p*, *q*, and *r*.
- *Epoch differs from mission start date* allows to define an epoch of the vehicle initial state in case it is different from the mission start date. More details about date and time formats are provided in Section 4.1.

Background

The user-defined initial state is used for the integration (propagation) executed by the initial guess generator. But it is not necessarily used during optimization or simulation, where the initial state information is read from TOPS file, since these values may be (or were) modified by the optimizer.

4.2.1 Constellations

Description

The *Initial State* configuration panel for constellations allows the user to specify the initial state of each satellite forming a constellation. It can be done by using a *TLE Catalog*, in that case the initial states are read from file, or by using *Orbital planes*, which allows a complete definition of the constellation geometry.

Configuration

DYNAMICS CONFIGURATION

A *TLE Catalog* initial state requires the same settings as described in chapter Section 4.2.3.2.12. The only difference is that only the possibility of specifying TLEs from file is provided. The vehicles identifiers defined for the constellation (in the configuration panel of the constellation top level node in *Vehicles & POI Definition*) are used.

The *Orbital planes* option provides the possibility of define directly the orbital planes.

If *Walker constellation* is selected in the *Orbital planes definition* drop-down list, a classic Walker constellation of type T/P/F can be created (see [117]), where T stands for the total number of satellites (given by *Number of planes* times *Number of satellites per plane*), P stands for the number of planes evenly spaced in node and F is the *Phasing parameter* (relative spacing between satellites in adjacent planes). Sometimes, the inclination is also added to the notation. The *Constellation type* allows to choose between a *Delta* constellation (satellites spanning the 360 degrees around the equator) and a *Star* constellation (spanning the 180 degrees around the equator).

For example, the Galileo Navigation system is a Walker 56°:27/3/1 Delta constellation. This means there are 27 satellites in 3 planes inclined at 56 degrees, across 360 degrees around the equator.

Semi-major axis, *Inclination* and *Eccentricity* are the same for all the orbital planes and user defined. The argument of perigee is set to 0.0 for all the satellites, while RAAN and true anomaly (v) are determined in the following way for Delta constellations:

$$\begin{aligned} s_{in-plane} &= \frac{2\pi}{T} P \\ s_{node} &= \frac{2\pi}{P} \\ s_{phase} &= \frac{2\pi}{T} F \\ RAAN &= s_{node} \cdot (n_{plane\#} - 1) \\ v &= s_{in-plane} \cdot n_{slot\#} + s_{phase} \cdot (n_{plane\#} - 1) \end{aligned} \quad (4.5)$$

where the spacing parameters s are auxiliary variables, $n_{plane\#}$ and $n_{slot\#}$ are user input to be provided in the *Satellites* table. In case of a Star constellation the s_{node} parameter must be divided by 2.

The **Add Satellites** button allows the user to rapidly fill the table with the satellites IDs, defined in the configuration panel of the constellation top level node in *Vehicles & POI Definition*.

The *Custom* option in the *Orbital planes definition* gives the possibility to individually define each orbital plane. In this way each plane can have different orbital elements and number of satellites. For each orbital plane the user needs to specify its orbital elements and the position of the satellite. If *Equidistant* is selected in the *Satellites distribution* drop-down list, each satellite will be placed at the same distance from the next one, starting from the true anomaly specified in *True anomaly offset* (by default 0.0, which means the first satellite has true anomaly equal to 0.0). If a *Custom* satellites distribution is selected, the user can manually insert the *True anomaly* or *Mean anomaly* value of each satellite.

Also in this case, the **Add Satellites** button allows the user to rapidly fill the table with the satellites IDs, defined in the configuration panel of the constellation top level node in *Vehicles & POI Definition*. Since in this case multiple orbital planes may be defined, a dialog pops up allowing the user to decide how many identifiers should be inserted and which one to start with.

If the **Add** button is pressed, the identifiers previously inserted are removed before inserting the new ones.

4.2.2 Ground Stations

Description

Configuration

4.2.3 Vehicles

The *State type* for vehicle objects can be defined by four main categories some of them with various sub-settings:

- *Position & Velocity*, see Section 4.2.3.1 for details
- *Orbital Elements*, see Section 4.2.3.2 for details
- *Relative Motion*, see Section 4.2.3.3 for details
- *Barycentric Rotating*, see Section 4.2.3.4 for details

4.2.3.1 Position and Velocity State

Description

Here, the translational state is defined with a position vector and a velocity vector. It is only available for vehicle objects.

Configuration

It requires two specifications:

- **Position** defines the position vector and comprises:
 - *Frame* specifies the coordinate system of the position vector:
 - *TOD* is the inertial true equator and equinox of date axis system (see Section 9.3.2.3)
 - *J2000* is the mean equator and equinox of epoch at J2000.0 axis system (see Section 9.3.2.1)
 - *PCPF* is the rotating planet-centered planet-fixed axis system (see Section 9.3.2.4)
 - *Representation* specifies the kind of vector representation:
 - *Polar* is the spherical coordinate system and requires:

DYNAMICS CONFIGURATION

- *Altitude type* allows the selection of *Altitude* or *Radius* as radial distance.
- *Latitude type* allows the selection of either *Latitude* or *Declination* as polar angle (see Section 2.1.2 for details).
- *Longitude* is the azimuth angle

Tip: The difference between the planetodetic latitude and planetocentric latitude (declination) is illustrated Fig. 2.2.

- *Cartesian* is the Cartesian coordinate system specified by three numerical coordinates *X*, *Y* and *Z* (w.r.t. selected frame).

■ **Velocity** defines the velocity vector specified by:

- *Reference frame* specifies the coordinate frame in which the velocity vector is measured:
 - *TOD* is the inertial true equator and equinox of date axis system (see Section 9.3.2.3)
 - *J2000* is the mean equator and equinox of epoch at J2000.0 axis system (see Section 9.3.2.1)
 - *PCPF* is the rotating planet-centered planet-fixed axis system (see Section 9.3.2.4)
- *Representation frame* specifies the coordinate frame in which the velocity vector is represented:
 - *TOD* is the inertial true equator and equinox of date axis system (see Section 9.3.2.3)
 - *J2000* is the mean equator and equinox of epoch at J2000.0 axis system (see Section 9.3.2.1)
 - *PCPF* is the rotating planet-centered planet-fixed axis system (see Section 9.3.2.4)
 - *L* is the planetocentric local axis system (see Section 9.3.4.1)
 - *L planetodetic* is the planetodetic local axis system (see Section 9.3.4.2)
- *Representation* specifies the kind of vector representation:
 - *Polar* is the spherical coordinate system and requires the radial distance, the polar angle and the azimuth angle
 - *Cartesian* is the Cartesian coordinate system specified by three numerical coordinates

The actual components of the velocity vector depend on the selected frame and representation:

- *Polar* representation in local horizontal frames (*L*, *L planetodetic*) requires:
 - *Speed* is the velocity magnitude (length of velocity vector)
 - *Inclination* is the flight path inclination angle (polar angle of velocity vector)

- *Heading* is the flight path heading angle (azimuth angle of velocity vector)
- *Cartesian* representation in local horizontal frames (*L*, *L planetodetic*) requires:
 - *Velocity radial* is the speed in radial direction
 - *Velocity north* is the speed in north direction
 - *Velocity east* is the speed in east direction
- *Cartesian* representation in the coordinate frames *J2000*, *TOD* or *PCPF* requires the *X*, *Y* and *Z* velocity components.

4.2.3.2 Orbital Elements State

Description

Orbital elements are a set of translational states defining the position and velocity of an object.

Configuration

Orbital elements as state type require the following specifications:

- *Coordinate system* defines the reference frame of the specified orbital elements:
 - *J2000* is the mean of equator and equinox at epoch of J2000.0 axis system (see Section 9.3.2.1)
 - *TOD* is the true equator and true equinox of date axis system (see Section 9.3.2.3)
- *Orbit type* allows a selection from supported orbit definitions:
 - *Keplerian Elliptic* is for user-defined elliptical orbits at any central body specified by set of Keplerian orbital elements, see Section 4.2.3.2.1
 - *Keplerian Hyperbolic* specifies a user-defined hyperbolic orbit at any central body using a set of Keplerian orbital elements, see Section 4.2.3.2.2
 - *Keplerian Parabolic* specifies a user-defined parabolic orbit at any central body by a set of Keplerian elements, see Section 4.2.3.2.3
 - *Equinoctial Elements* allows a user-defined orbit at any central body defined by a set of modified equinoctial elements, see Section 4.2.3.2.4
 - *Stationary Orbit* creates a stationary orbit at any central body, see Section 4.2.3.2.5
 - *Sun Synchronous Orbit* is an Earth-centered sun-synchronous orbit defined by the semi-major axis, see Section 4.2.3.2.6
 - *Sun Synchronous Repeat Orbit* is an Earth-centered sun-synchronous orbit defined by the repeat cycle, see Section 4.2.3.2.7

DYNAMICS CONFIGURATION

- *Ground Repeat Orbit* defines an orbit with a dedicated ground track repeat cycle, see Section 4.2.3.2.8
- *Frozen Orbit* creates an elliptical orbit with a minimum drift due to central body oblateness (J2) dedicated to repeated observations under constant conditions, see Section 4.2.3.2.9
- *Halo Orbit* defines a periodic orbit in the three-body problem, see Section 4.2.3.2.10
- *Lissajous Orbit* defines a periodic orbit in the three-body problem, see Section 4.2.3.2.10
- *Molniya Orbit* is an Earth-centered Molniya orbit, see Section 4.2.3.2.11
- *TLE* creates an Earth-centered orbit based on Two Line Elements, see Section 4.2.3.2.12

Furthermore it is possible to define an optional *Initial deorbit* maneuver (*disabled* by default). A detailed explanation is given in Section 4.2.3.2.13.

Tip: In case *Orbital elements* are used to define the initial state of a vehicle, it is recommended to use a set of equations of motion defined in an inertial frame, for example *Equinoctial Elements* (please refer to Section 4.5.1.5 for details). This increases the numerical stability during trajectory propagation.

4.2.3.2.1 Keplerian Elliptic Orbit

Description

Using this option allows the definition of an elliptic or circular orbit by a set of Keplerian orbital elements.

Configuration

Keplerian Elliptic orbits have these settings:

- *Anomaly type* defines the kind of anomaly, the position of the vehicle on the orbit:
 - *Mean Anomaly* is the current position of the vehicle based on the mean motion, measured from periapsis
 - *True anomaly* is the angular position between vehicle and the periapsis measured from the central body
- *Orbit shape* selects the parameters which are used to determine the shape of the elliptic orbit:
 - *Apoapsis and Periapsis Altitude*
 - *Apoapsis and Periapsis Radius*
 - *Mean Motion and Eccentricity*
 - *Orbit Energy and Eccentricity*

DYNAMICS CONFIGURATION

- *Orbital Period and Eccentricity*
- *Semi Major Axis and Eccentricity*

Depending on the selection two of these elements are required:

- *Apoapsis altitude* specifies the apoapsis altitude considering the equatorial radius of the central body
 - *Apoapsis radius* specifies the apoapsis radius of the orbit, or - in case of a circular orbit - its radius
 - *Periapsis altitude* specifies the periapsis altitude considering the equatorial radius of the central body
 - *Periapsis radius* specifies the periapsis radius of the orbit, or - in case of a circular orbit - its radius
 - *Semi-major axis* specifies the half of the major axis of the orbit, or - in case of a circular orbit - its radius
 - *Eccentricity* defines how much the orbit deviates from a circle
 - *Mean motion* is the angular velocity of the vehicle required for one orbital revolution assuming constant speed
 - *Orbit energy* is the specific energy of the orbit, the sum of the potential and kinetic energy
 - *Orbit period* specifies the orbital period
- *Inclination* specifies the inclination of the orbit
 - *RAAN* specifies the right ascension or longitude of the ascending node of the orbit

Note: The RAAN is ill-defined with an inclination of zero.

- *Periapsis argument* defines the angle between the ascending node and the periapsis of the orbit

Note: The argument of periapsis is ill-defined with an eccentricity of zero.

Optimization

For all quantities, a lower and/or upper bound can be specified when clicking the respective checkbox. If unchecked, the bounds equal the specified value. When specifying a finite range the quantity becomes optimizable. After an initial guess the parameters are available as real parameters.

4.2.3.2.2 Keplerian Hyperbolic Orbit

Description

With this option a hyperbolic orbit can be defined through a set of Keplerian elements.

Configuration

Keplerian Hyperbolic orbits have the following settings:

- *Anomaly type* defines the kind of anomaly, the position of the vehicle on the orbit:
 - *Mean Anomaly* is the current position of the vehicle based on the mean motion, measured from periapsis
 - *True anomaly* is the angular position between vehicle and the periapsis measured from the central body
- *Orbit shape* selects the parameters which are used to determine the shape of the hyperbolic orbit:
 - *Excess Velocity and Periapsis Altitude*
 - *Excess Velocity and Periapsis Radius*
 - *Semi Major Axis and Eccentricity*
 - *Semiparameter p and Eccentricity*

Depending on the selection two of these elements are required:

- *Excess velocity* specifies the excess or infinite velocity of the orbit
 - *Periapsis altitude* specifies the periapsis altitude using the equatorial radius of the central body
 - *Periapsis radius* specifies the periapsis radius of the orbit
 - *Eccentricity* specifies the eccentricity of the initial orbit (for hyperbolic orbit it has to be >1)
 - *Semi-major axis* specifies the half of the major axis of the orbit
 - *Semiparameter p* specifies the semi-parameter or semi-latus rectum of the orbit
- *Inclination* specifies the inclination of the orbit
 - *RAAN* specifies the right ascension or longitude of the ascending node of the orbit

Note: The RAAN is ill-defined with an inclination of zero.

- *Periapsis argument* defines the angle between the ascending node and the periapsis of the orbit

Optimization

For all quantities, a lower and/or upper bound can be specified when clicking the respective checkbox. If unchecked, the bounds equal the specified value. When specifying a finite range the quantity becomes optimizable. After an initial guess the parameters are available as real parameters.

4.2.3.2.3 Keplerian Parabolic Orbit

Description

This option is useful for parabolic defined as a set of Keplerian elements. A parabolic orbit always has an eccentricity of 1.

Configuration

Keplerian Parabolic orbits have the following settings:

- *Anomaly type* defines the kind of anomaly, the position of the vehicle on the orbit:
 - *Mean Anomaly* is the current position of the vehicle based on the mean motion, measured from periapsis
 - *True anomaly* is the angular position between vehicle and the periapsis measured from the central body
- *Orbit shape* selects the parameters which are used to determine the shape of the parabolic orbit:
 - *Periapsis Altitude*
 - *Periapsis Radius*
 - *Semiparameter p*

Depending on the selection two of these elements are required:

- *Periapsis altitude* specifies the periapsis altitude using the equatorial radius of the central body
- *Periapsis radius* specifies the periapsis radius of the orbit
- *Semiparameter p* specifies the semi-parameter or semi-latus rectum of the orbit
- *Inclination* specifies the inclination of the orbit
- *RAAN* specifies the right ascension or longitude of the ascending node of the orbit

Note: The RAAN is ill-defined with an inclination of zero.

- *Periapsis argument* defines the angle between the ascending node and the periapsis of the orbit

Optimization

For all quantities, a lower and/or upper bound can be specified when clicking the respective checkbox. If unchecked, the bounds equal the specified value. When specifying a finite range the quantity becomes optimizable. After an initial guess the parameters are available as real parameters.

4.2.3.2.4 Equinoctial Elements

Description

Modified equinoctial elements is a non-singular set of orbital elements without deficiencies at zero inclination and zero eccentricity (like it is with Keplerian elements). It is suitable for all kinds of orbits (circular, elliptic, parabolic, hyperbolic or degenerated).

Configuration

The following settings are available for *Equinoctial Elements*:

- *Equinoctial p* is the semiparameter p (semi-latus rectum) of the orbit, a measure for the shape of the orbit
- *Equinoctial f* is the cosine component of the equinoctial eccentricity vector
- *Equinoctial g* is the sine component of the equinoctial eccentricity vector
- *Equinoctial h* is the cosine component of the equinoctial inclination vector
- *Equinoctial k* is the sine component of the equinoctial inclination vector
- *Equinoctial l* is the true longitude of the vehicle position: the true anomaly and the longitude of the periapsis (longitude of ascending node and argument of periapsis)

4.2.3.2.5 Stationary Orbit

Description

A stationary orbit is an orbit with an orbital period according to the central body's rotational period. It is also known as synchronous orbit (e.g. geosynchronous orbit for Earth). In case of zero inclination, it is a stationary orbit and the vehicle appears motionless with respect the surface of the central body.

Configuration

Due to the inherent constraints of this orbit, most orbit elements are predefined. Only these settings are available:

- *Defined by* specifies the vehicle position with respect to the surface and has these options:
 - *Local Sidereal Time* is the angle between the vernal equinox and the vehicle position
 - *Longitude* is the relative longitude with respect prime meridian
- *Inclination* is optional and defines the orbital inclination. By default, an inclination of zero specifies a stationary orbit.

4.2.3.2.6 Sun-Synchronous Orbit

Description

A sun-synchronous orbit is an elliptical Earth orbit whose plane rotates such that the same angle w.r.t. Sun is maintained. As a consequence the satellite always passes over any given point of the central body surface at the same local solar time.

Configuration

The following settings are required:

- *Semi-major axis* specifies the half of the major axis of the orbit, or - in case of a circular orbit - its radius
- *Eccentricity* specifies the eccentricity of the initial orbit
- *Ascending node* is the right ascension of the ascending node and can be *Defined by*:
 - *Local Sidereal Time* defines the angle between vernal equinox of the coordinate system and ascending node

Tip: One sidereal day (~23h56m04s) corresponds to 360 degrees.

- *LTAN* is the local solar time which is the angle between the meridian whose solar time is 12:00 and the ascending node
- *RAAN* is the angle between vernal equinox of the coordinate system and ascending node. Two options are available:
 - *default* specifies zero as right ascension of ascending node
 - *custom* requires a value of the right ascension of ascending node

The orbit inclination is computed internally using a precession rate equal to the mean motion of the Earth about the Sun.

Note: Sun-synchronous circular orbits around the Earth are only defined for semi-major axes between 6554 km (inclination = 96°) and 12352 km (inclination = 180°). Only non-circular orbits can achieve larger semi-major axes.

Tip: Dawn-Dusk orbits have a local mean solar time of 06:00 or 18:00, and Noon-Midnight orbits have a local mean solar time of 12:00 or 24:00 (00:00).

4.2.3.2.7 Sun-Synchronous Repeat Orbit

Description

DYNAMICS CONFIGURATION

A sun-synchronous repeat orbit is a sun-synchronous orbit with a specified repeat cycle.

Configuration

The following settings are available:

- *Revolutions per day* specifies the number of revolutions per day for the repeat cycle
- *Ascending node* is the right ascension of the ascending node and can be *Defined by*:
 - *Local Sidereal Time* defines the angle between vernal equinox of the coordinate system and ascending node

Tip: One sidereal day (~23h56m04s) corresponds to 360 degrees.

- *LTAN* is the local solar time which is the angle between the meridian whose solar time is 12:00 and the ascending node
- *RAAN* is the angle between vernal equinox of the coordinate system and ascending node. Two options are available:
 - *default* specifies zero as right ascension of ascending node
 - *custom* requires a value of the right ascension of ascending node

4.2.3.2.8 Ground Repeat Orbit

Description

A ground repeat orbit is an orbit with a specified ground repeat cycle.

Configuration

This orbit requires the following specifications:

- *Inclination* is the orbit inclination
- *Revolutions per day* specifies the number of revolutions per day for the ground repeat cycle
- *Ascending node* is the right ascension of the ascending node and can be *Defined by*:
 - *Local Sidereal Time* defines the angle between vernal equinox of the coordinate system and ascending node

Tip: One sidereal day (~23h56m04s) corresponds to 360 degrees.

- *LTAN* is the local solar time which is the angle between the meridian whose solar time is 12:00 and the ascending node
- *RAAN* is the angle between vernal equinox of the coordinate system and ascending node. Two options are available:

- *default* specifies zero as right ascension of ascending node
- *custom* requires a value of the right ascension of ascending node

With the given input the eccentricity and semi-major axis are computed.

4.2.3.2.9 Frozen Orbit

Description

Frozen orbits define an orbit geometry which results in the lowest natural drift, e.g. from gravitational perturbations due to the central body oblateness. A reduction of these perturbations is realized by choosing an orbit eccentricity which results in a nearly constant altitude. Such orbit is also called minimum altitude variation orbit or frozen-eccentricity orbit.

Configuration

The following settings are available:

- *Mean altitude* is the mean orbit altitude to be maintained
- *Inclination* is the orbit inclination
- *Ascending node* is the right ascension of the ascending node and can be *Defined by*:
 - *Local Sidereal Time* defines the angle between vernal equinox of the coordinate system and ascending node

Tip: One sidereal day (~23h56m04s) corresponds to 360 degrees.

- *LTAN* is the local solar time which is the angle between the meridian whose solar time is 12:00 and the ascending node
- *RAAN* is the angle between vernal equinox of the coordinate system and ascending node. Two options are available:
 - *default* specifies zero as right ascension of ascending node
 - *custom* requires a value of the right ascension of ascending node

4.2.3.2.10 Halo/Lissajous Orbit

Description

DYNAMICS CONFIGURATION

Three-dimensional halo/Lissajous orbits are a particular kind of periodic orbits, defined in the circular restricted three-body problem. These orbits have the same frequency in the in-plane and out-plane motions and also for this reason they have been already used for several mission.

Configuration

In order to define a halo/Lissajous orbit the following input is required:

- *Second primary* is the second central body and defines the three-body problem, since the first primary can be consequently determined
- *Lagrangian point* specifies around which Lagrangian point the orbit has to be computed
- *Type* specifies whether the initial state is computed using an *analytical* or *numerical* algorithm

Note: *Analytical* orbits can be used only in combination with *Circular Restricted 3 Body Problem Analytical* equations of motion. Analytical solutions cannot be numerically integrated without diverging quite rapidly.

- *X-Amplitude* (A_x) and *Z-Amplitude* (A_z) define the preferred amplitudes of the orbit

Background

The determination of the initial state of a periodic orbit in the circular restricted three body problem is quite critical due to the complexity of the model and the typical instability of such orbits.

Several analytical approximations exist to determine periodic orbits around the Lagrangian points of a three-body system. Amongst them, one of the most widely used is of the third order and it can be used to generate analytical Halo or Lissajous orbits. The equations describing the motion around the Lagrangian point are according to [133] and [137]:

$$\begin{aligned}x &= a_{21}A_x^2 + a_{22}A_z^2 - A_x \cos(\tau_1) + (a_{23}A_x^2 - a_{24}A_z^2) \cos(2\tau_1) + (a_{31}A_x^3 - a_{32}A_x A_z^2) \cos(3\tau_1) \\y &= k A_x \sin(\tau_1) + (b_{21}A_x^2 - b_{22}A_z^2) \sin(2\tau_1) + (b_{31}A_x^3 - b_{32}A_x A_z^2) \sin(3\tau_1) \\z &= \delta_r [A_z \cos(\tau_2) + d_{21}A_x A_z (\cos(2\tau_2) - 3) + (d_{32}A_z A_x^2 - d_{31}A_z^3) \cos(3\tau_2)]\end{aligned} \quad (4.6)$$

Halo orbits are obtained from the previous formulas by setting $\tau_1 = \tau_2$.

A numerical algorithm based on the model presented in [132] allows to define Halo orbits and to assess their stability. An initial guess is required to start the numerical algorithm. In that regard, the aforementioned analytical solution can be used, computing the value at $t=0$. The monodromy matrix defined in [132] and derived from the state transition matrix allows the computation of manifold trajectories, since it relates changes in the initial state with those in the final state of a trajectory. The initial state vector of a periodic orbit has the following characteristics:

$$\vec{x}_0 = (x_0, 0, z_0, 0, \dot{y}_0, 0) \quad (4.7)$$

The differential equation to be solved is defined here:

$$\frac{d}{dt} \Phi(t, 0) = F(t) \Phi(t, 0) \quad (4.8)$$

DYNAMICS CONFIGURATION

$$F = \begin{bmatrix} 0 & I \\ U_{xx} & 2\Omega \end{bmatrix} \quad (4.9)$$

$$\Omega = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad (4.10)$$

where U_{xx} is the symmetric matrix of the second partial derivatives of the field potential and I is the identity matrix.

The circular restricted three-body problem equations of motion are integrated until y changes sign. Then the step-size is reduced and the integration goes forward again, until $|y| < 10E^{-11}$. The orbit is considered periodic if x and z velocity components are below $10E^{-8}$. Otherwise those values should be corrected adjusting the initial vector as follows:

$$\begin{pmatrix} \delta z_0 \\ \delta \dot{y}_0 \end{pmatrix} \cong \begin{pmatrix} \delta \ddot{x} \\ \delta \ddot{z} \end{pmatrix} \begin{bmatrix} \Phi_{43} \Phi_{45} \\ \Phi_{63} \Phi_{65} \end{bmatrix} - \frac{1}{\dot{y}} \begin{pmatrix} \ddot{x} \\ \ddot{z} \end{pmatrix} \begin{bmatrix} \Phi_{23} & \Phi_{25} \end{bmatrix} \Bigg]^1 \quad (4.11)$$

It is worth noticing that this iterative approach may fail or may lead to a periodic orbit, quite distant from the requested one. This is because not for any combination of inputs a periodic orbit exists.

4.2.3.2.11 Molniya Orbit

Description

This model realizes a Molniya orbit at Earth. Such highly elliptical orbits - also called Tundra orbits - are usually required to achieve long periods of visibility positions at high latitudes where stationary orbits are infeasible.

Configuration

This orbit requires only one input parameter:

- *Ascending node* is the right ascension of the ascending node and can be *Defined by*:
 - *Apoapsis longitude* is the relative longitude of the apoapsis, the angle between prime meridian and apoapsis
 - *Local Sidereal Time* defines the angle between vernal equinox of the coordinate system and ascending node

Tip: One sidereal day (~23h56m04s) corresponds to 360 degrees.

- *RAAN* is the angle between vernal equinox of the coordinate system and ascending node. Two options are available:
 - *default* specifies zero as right ascension of ascending node

- *custom* requires a value of the right ascension of ascending node

All remaining orbital elements are fixed using a semi-major axis of 26554 km, eccentricity of 0.72, inclination of 63.4° and an argument of periapsis of -90°.

4.2.3.2.12 Two Line Elements (TLE)

Description

Two Line Elements (TLE) is a set of two data lines listing orbital elements that describe the state of an Earth-orbiting object. TLE data must be provided in a special format as indicated in [108].

Configuration

Orbitat initial states specified by *TLE* require the following input parameters:

- *WGS gravity model* specifies the gravity model used to compute the orbit from TLE:
 - *WGS 2* is the World Geodetic System 2
 - *WGS 72 low* is the World Geodetic System 1972 with low precision
 - *WGS 72* is the World Geodetic System 1972
 - *WGS 84* is the World Geodetic System 1984
 - *WGS 96* is the World Geodetic System 1996
- *Data source* defines how the TLE are specified:
 - *Local* provides the input fields *Line 1* and *Line 2* for manual specification of typical TLE input
 - *From File* requires additional input for the *File name* which contains the TLE data. Two possible options *TLE Object Name* and *TLE Object Number* allow to define if the provided vehicle ID (i.e. the identifier of the vehicle in the *Modelling* tree) corresponds to either the satellite name (contained in header line) or the satellite number (in columns 3-7 of the first line) in the TLE catalog.

Background

TLE are always provided in a special inertial reference frame, namely TEME (true equinox mean equator). As TLE are usually propagated with an SGP4 propagator, they are linked to a *WGS gravity model*. All available WGS models in ASTOS and their main characteristics are listed below [67]:

Table 4.1: WGS gravity models

DYNAMICS CONFIGURATION

WGS Model	Earth Radius [km]	Earth Gravity Parameter [km^3/s^2]	J2	J3	J4
WGS 2	6378.1363	398600.4415	0.001082616	-0.00000253881	-0.00000165597
WGS 72 low	6378.135	398600.79964	0.001082616	-0.00000253881	-0.00000165597
WGS 72	6378.135	398600.8	0.001082616	-0.00000253881	-0.00000165597
WGS 84	6378.137	398600.47	0.00108262998905	-0.00000253215306	-0.00000161098761
WGS 96	6378.137	398600.4418	0.00108262998905	-0.00000253215306	-0.00000161098761

The SGP4 propagator is used to propagate the TLE from their given reference epoch to the relevant epoch (e.g. mission start date) defined for the scenario. ASTOS assumes and uses the UTC time standard for given TLE epochs. According to [148] section II.E this assumption is valid since no formal documentation exists.

4.2.3.2.13 Initial Deorbit

Description

A *Initial deorbit* is an optional feature designed for re-entry optimization tasks. It allows to start the mission with an orbit and to add the deorbit maneuver without the need to specify a separate deorbit phase.

Configuration

The *Type* of an initial deorbit maneuver can be:

- *Deorbit Impulse* specifies explicitly the components of the impulse:

- *Tangential* is the tangential delta-V component
- *Radial* is the radial delta-V component
- *Normal* is the normal delta-V component

The impulse is not defined in a dedicated coordinate frame. Activating the *Bounds* allows optimizing the selected component(s) of the impulse as real parameter.

- *Transfer Orbit* allows a fast assessment of the re-entry point or entry interface with respect to the landing site based on the initial orbit. An optimization is started at the entry interface. It requires the following specifications:

- *Periapsis Altitude* defines the periapsis altitude of the transfer orbit after the deorbit impulse. It provides a measure for the steepness of the entry and can be declared as optimizable.
- *Reentry Altitude* defines the altitude of the (atmospheric) entry, usually at the atmospheric limit. This value is also used as initial state for the state grid for purpose of optimization.

- *Target Longitude* defines the longitude of the landing site
- *Target Latitude* defines the latitude of the landing site
- *Estimated Downrange* can be used to provide an estimation of the downrange which is often known from previous computations
- *Estimated Downrange Time* can be used to provide an estimation of the flight time from (atmospheric) entry to landing site
- *Ascending Branch* specifies if the landing site is targeted on an ascending (*enabled*) or descending arc (*disabled*) with respect to the equator
- *Fixed ascending node passage* specifies if the longitude of the ascending node of the initial orbit shall be kept (*enabled*) or not (*disabled*)

Background

All altitudes are computed with respect to the equatorial radius of the central body defined in the *Environment* branch of the *Modelling* tree (see Section 2.1). As both the argument of periapsis and the latitude of any point of the transfer orbit are unknown, they can be only estimated. The *Periapsis altitude* of the transfer orbit is estimated at the *Target latitude*. As depicted in Fig. 4.2, periapsis and target point are close together (this is the only optimizable parameter, in case lower and/or upper bounds have been specified). The definition of the periapsis is equivalent to the definition of a deorbit impulse which is computed internally.

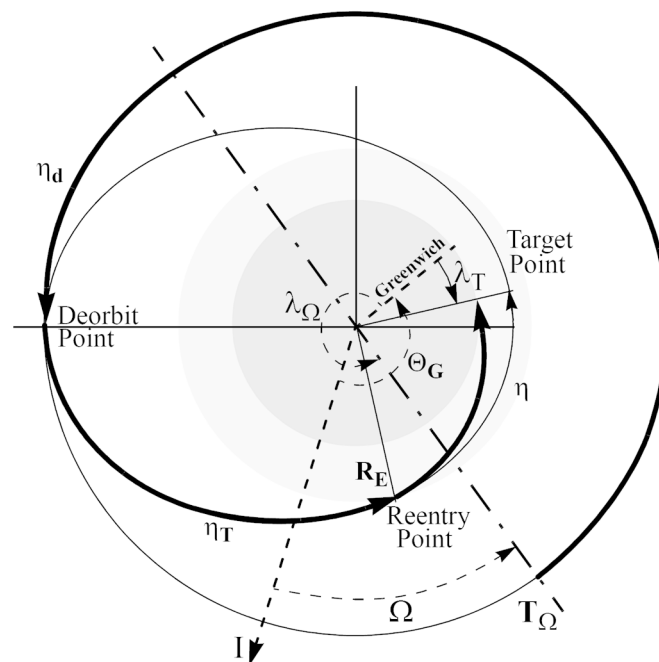


Fig. 4.2: Specification of a transfer orbit

Together with the *Target longitude* λ_T , the *Estimated downrange* η and the parameter *Ascending branch*, the transfer anomaly η_T and the transfer time TT can be evaluated using orbital mechanics. Using spherical trigonometry, the right ascension of the ascending node relative to the prime meridian λ_Ω can be computed. If the *Target latitude* is larger than the orbit inclination, the inclination value is used for the latitude.

DYNAMICS CONFIGURATION

Furthermore, the true anomaly of the initial orbit at the deorbit point η_d can be computed using geometric relations. If the deorbiting anomaly is not within the bounds of the initial orbit, it is set to the closest possible bound. Finally, the inertial ascending node Ω can be computed from the *Estimated_Downrange_Time* t , the transfer time TT and the traveling time for T_d beginning at the time of passage of the ascending node T_Ω (see Section 4.1).

The parameter *Fixed ascending node passage* determines whether the time of passage is fixed or can be modified during the computations. If the ascending node is not within the bounds of the initial state, this parameter defines the following computational steps:

- If *enabled*, the ascending node is set to the closest bound and the best matching true anomaly inside its bounds is evaluated
- If *disabled*, the ascending node is the same as defined for the initial orbit and the true anomaly as computed above. The time of ascending node passage is modified for one sidereal day until the target point is crossed very closely. This decouples the inertial and relative frame and couples it again at another time instance. This is equivalent to modifying the ascending node.

The result of this computation also depends on the boundary settings of the *RAAN* and *True anomaly* of the initial orbit and the settings of *Fixed ascending node passage*:

- **Ascending node and true anomaly are known:** Do not specify lower or upper bounds for the initial orbit parameters. If the universal time is known, too, fix the passage time. In this case, there is no possibility to adapt any parameter in order to reach the target point.
- **Only ascending node of initial orbit is known:** In most cases the ascending node is fixed, but the epoch of the deorbit maneuver is not. Then set the bounds of the true anomaly to $\pm 360^\circ$. Also the time of the ascending passage can be freed, when the target point is not reachable.
- **Orbit condition should be totally free:** Set the ascending node bounds to $\pm 180^\circ$ and the true anomaly to $\pm 360^\circ$. In this case, the time of the ascending node passage is never modified. This is useful to investigate cross range capabilities where the orbit parameters should not be bounded.

In addition to the initial orbit parameters, the computed deorbit impulse, the entry radius as well as the reentry orbital state are saved as design parameters (real parameters). If the target point cannot be reached during the optimization, the bounds of the ascending node and true anomaly can be freed, but the time of the ascending node passage cannot be modified any more.

Tip: The *Estimated downrange* and *Estimated downrange time* are only valid in case of a force-free re-entry (no drag, lift, etc.). Please change the *Estimated downrange* value to match the desired latitude and the *Estimated downrange time* to move your trajectory along the relative longitude.

4.2.3.3 Relative Motion

Description

Defining the initial state of a vehicle with respect to another one is possible with this option. It is only available for vehicles.

Note: This initial state settings requires at least two vehicles in the scenario!

Configuration

The *Relative Motion* panel has the following settings:

- *Reference vehicle* allows the selection of a reference vehicle from a drop-down list. The specified initial state is with respect to this reference vehicle.
- *Representation* specifies the coordinate system:
 - *LO* is the local orbit axis system (see Section 9.3.1.1) and has the following settings:
 - *Position* is the relative position vector defined using Cartesian components X , Y and Z w.r.t. reference vehicle
 - *Velocity* is the relative velocity vector defined using Cartesian components VX , VY and VZ w.r.t. reference vehicle
 - *R-bar/V-bar* is the R-bar/V-bar axis system (see Section 9.3.6.3) and has the following settings:
 - *Position* is the relative position defined by the components *R-bar*, *V-bar* and *H-bar* w.r.t. reference vehicle
 - *Velocity* is the relative velocity defined by the components *R-bar*, *V-bar* and *H-bar* w.r.t. reference vehicle

4.2.3.4 Barycentric Rotating

Description

With this *Initial State* option the user can specify Cartesian elements for position and velocity vectors in a two-body rotating frame. This input can be used to define any orbit in the Circular Restricted Three Body Problem (CR3BP). For more details on CR3BP, please see Section 4.5.1.2. Additional information about the two-body rotating coordinate system can be found in Section 9.3.1.1.

Configuration

DYNAMICS CONFIGURATION

The following inputs are required to fully specify the two-body rotating frame and the position and velocity vectors in this frame:

- *Secondary* is the less massive celestial body in the three-body problem. The primary is automatically determined to be the parent of the selected *Secondary*. For example if *Moon* is selected, the primary is the Earth and if *Earth* is selected, the primary is the Sun. The primary body must exist in the scenario.
- *Center* is the reference point for the Cartesian elements and can be the *Primary*, the *Secondary* or the *Barycenter* of the selected system.
- *Normalized* activates normalized Cartesian elements, i.e. the position Cartesian elements are normalized with the distance between *Primary* and *Secondary* (which is assumed to be constant) and the velocity Cartesian elements are normalized with the velocity of the *Secondary* around the *Primary*. The default is *Disabled* meaning the unit of position is a distance unit and the unit of velocity can be any distance unit divided by any time unit.
- X coordinate of the vehicle in the two-body rotating frame. The positive x-axis points from the primary to the secondary.
- Y coordinate of the vehicle in the two-body rotating frame. The y-axis completes the right handed coordinate system.
- Z coordinate of the vehicle in the two-body rotating frame. The z-axis is parallel to the angular momentum vector of the orbit of the secondary around the primary.
- VX is the x-axis component of the vehicle velocity vector.
- VY is the y-axis component of the vehicle velocity vector.
- VZ is the z-axis component of the vehicle velocity vector.

4.3 Attitude Definition

Description

The vehicle attitude is its orientation and rotation in the flight environment. It affects and is affected by the forces and torques acting on the vehicle (e.g. aerodynamic and propulsive forces and moments). Only *Vehicles* may have an attitude definition. In *ASTOS* the vehicle attitude modelling is of two main types depending on the setting of the *Control/State* option:

- *Control* means a user-defined (time-invariant or time-variant) attitude is enforced to the vehicle and any external or internal force/torque does not have any effect on the attitude. Since the latter is enforced and only the vehicle position is computed by integrating translational equations of motion, such kind of vehicle dynamics denotes a 3 degrees of freedom (3-DoF) dynamics.
- *State* means that both the vehicle attitude and translational motion are computed by integrating rotational and translational equations of motion respectively. In this case, any forces and or moments acting on the vehicle will affect both its attitude and its trajectory resulting in a 6 degrees of freedom (6-DoF) dynamics.

The attitude modelling can be configured by the user by selecting an *attitude representation/definition* using the *Defined by* selector in the *Attitude* section of each phase setup. The following attitude representations and definitions are available in ASTOS:

- The *Aerodynamic Angles* representation is used for 3-DoF and for 6-DoF scenarios, see Section 4.3.1
- The *Aerodynamic Angles Controller 3DOF* representation allows the user to enforce the aerodynamic angles of the vehicle and is used for 3-DoF scenarios, see Section 4.3.2
- The *Docked* representation shall be used for docked vehicles, see Section 4.3.3
- The *Euler Angles* representation is used for 3-DoF and for 6-DoF scenarios, see Section 4.3.4
- The *Quaternion* representation is only used for 6-DoF scenarios, see Section 4.3.5
- *Target Pointing* points the vehicle body axes to targets, see Section 4.3.6
- The *Unit Vector* representation is only used for 3-DoF scenarios, see Section 4.3.7
- *From File* reads an attitude profile from a file, see Section 4.3.8

4.3.1 Aerodynamic Angles

Description

The attitude of a vehicle can be defined by a set of the following aerodynamic angles:

- **Angle of attack** α is the angle between the vehicle's body-fixed forward axis and the projection of the air path velocity vector on the vehicle's plane of symmetry
- **Side slip angle** β is the angle between the vehicle's plane of symmetry and the air path velocity vector
- **Air path bank angle** μ_a is the angle between the aerodynamic lift vector and the local vertical plane (positive in a right turn)

Tip: In case an unstable aerodynamics model is defined (positive pitch moment for positive angle of attack) the vehicle becomes a tumbling body with increasing rotating velocity and the simulation could become very slow!

Configuration

The *Aerodynamic Angles* attitude definition takes the following settings:

- *Control/State* defines how the aerodynamic angles are considered in the vehicle dynamics:
 - *Control* defines the set of aerodynamic angles either as based on analytic attitude control laws, optimizable parameter or optimizable control. It depends on the selected control law of each aerodynamic angles as well as on further settings. The following aerodynamic angles can be controlled:

DYNAMICS CONFIGURATION

- *Angle of attack* supports the following attitude control laws:
 - *Constant Law* (see Section 4.4.2)
 - *Dynamic Pressure* (see Section 4.4.3)
 - *Linear Law* (see Section 4.4.7)
 - *Maximum Lift to Drag Ratio* (see Section 4.4.8)
 - *Profile Law* (see Section 4.4.13)
 - *Trimmed Aero* (see Section 4.4.22)
- *Side slip angle* is optional and by *default* it is set to zero. These control laws are supported:
 - *Constant Law* (see Section 4.4.2)
 - *Linear Law* (see Section 4.4.7)
 - *Profile Law* (see Section 4.4.13)

Tip: A sideslip angle of zero can be used for aerodynamics that do not supply a side force.

- *Bank angle* supports these control laws:
 - *Constant Law* (see Section 4.4.2)
 - *Dynamic Pressure* (see Section 4.4.3)
 - *Linear Law* (see Section 4.4.7)
 - *Profile Law* (see Section 4.4.13)
 - *Vertical* (see Section 4.4.23)
- *State* is intended to be used for 6-DoF dynamics and defines the set of aerodynamic angles as dynamic states. In this case, the quantities are integrated as function of the independent variable. No control is applied to the vehicle and the total forces and moments (aerodynamic, propulsive, gravitational, etc.) define the attitude of the vehicle that acts as an uncontrolled 6-DoF body. For more information please refer to Section 4.5.2.1.

Note: *Aerodynamic Angles* set as *State* requires the translational equations of motion defined as *Flightpath Velocity* (see Section 4.5.1.6).

Note: *Aerodynamic Angles* set as *State* is not accurate when a wind model is defined. Please use *Euler Angles* set as *State* or *Quaternion* instead (see Section 4.3.4 and Section 4.3.5, respectively).

4.3.2 Aerodynamic Angles Controller 3-DOF

Description

The *Aerodynamic Angles Controller 3DOF* is a controller for 6-DoF computations. It uses an inverted dynamics flight control system and comprises the aerodynamic angles **Angle of attack** α , **Side slip angle** β and **Air path bank angle** μ_a . They are modeled as states as well as the body angular rates $\mathbf{p}$, $\mathbf{q}$ and $\mathbf{r}$.

Note: These angles are considered as commands, which are input of a controller, whereas the angles taken from the rotational state serve for aerodynamics computations.

Configuration

The way to specify the aerodynamic angles *Angle of attack*, *Side slip angle* and *Bank angle* is the same as for the *Aerodynamic Angles* (see Section 4.3.1). Their setup is only required for the computation of the aerodynamic forces and moments. Besides, the following specific settings have to be provided for the controller algorithm:

- *Angle error gain* is the gain of the commanded aerodynamic angles derivatives provided as matrix
- *Rate error gain* is the gain of the commanded body angular rates derivatives provided as matrix

Both error gain matrices are required to compute the outer and inner feedback loops.

Background

The suggested control concept uses successive dynamic inversion to transform the rotational equations of motion into a system that is approximately linear [46]. Additional linear feedback control provides local tracking of the attitude command given by $[\alpha, \beta, \mu]$. The specification includes the controller gains.

The rotational dynamics are inverted successively, assuming that the closed-loop system consists of two nested systems (see Fig. 4.3): an inner loop to control the angular rates, and an outer loop to control the attitude angles. This concept requires that the response of the angular rates is sufficiently fast and does not affect the response of the attitude angles. In the controller design, linear feedback and control filtering is used to provide the desired time-scale separation.

DYNAMICS CONFIGURATION

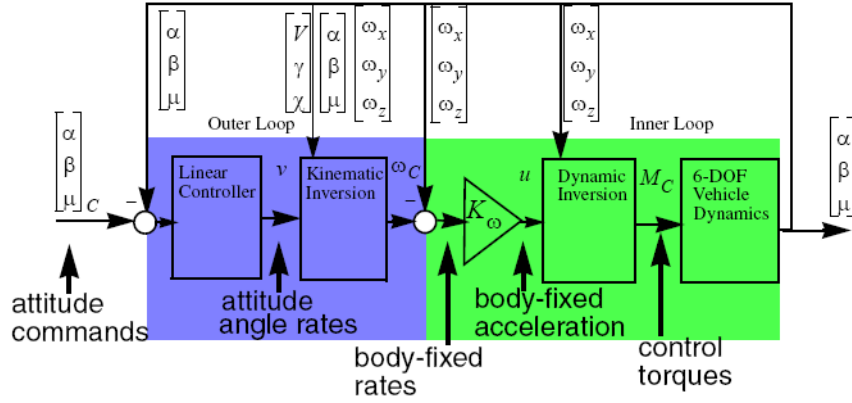


Fig. 4.3: Block diagram schematic of the attitude control system

The required angular rates along the aerodynamic angles is determined by a linear controller in the outer loop

$$\frac{d}{dt} \begin{bmatrix} \alpha_c \\ \beta_c \\ \mu_c \end{bmatrix} = -K_e \begin{bmatrix} \alpha - \alpha_c \\ \beta \\ \mu - \mu_c \end{bmatrix} \quad (4.12)$$

The corresponding angular rates in the body-fixed frame are determined by inverting the rotational kinematic equations

$$\omega_c = \begin{bmatrix} 0 & \sin\alpha & \cos\alpha\cos\beta \\ 1 & 0 & \sin\beta \\ 0 & -\cos\alpha & \sin\alpha\cos\beta \end{bmatrix} \left(\frac{d}{dt} \begin{bmatrix} \alpha_c \\ \beta_c \\ \mu_c \end{bmatrix} - \begin{bmatrix} T_\alpha \\ T_\beta \\ T_\mu \end{bmatrix} \right) \quad (4.13)$$

where

$$\begin{bmatrix} T_\alpha \\ T_\beta \\ T_\mu \end{bmatrix} = \begin{bmatrix} \frac{-1}{\cos\beta} \left(\frac{d\gamma}{dt} \cos\mu + \frac{d\chi}{dt} \cos\gamma \sin\mu \right) \\ \frac{d\chi}{dt} \cos\mu \cos\gamma - \frac{d\gamma}{dt} \sin\mu \\ \frac{d\gamma}{dt} \cos\mu \tan\beta + \frac{d\chi}{dt} (\cos\gamma \sin\mu \tan\beta + \sin\gamma) \end{bmatrix} \quad (4.14)$$

The required body-fixed acceleration is determined by a first-order linear filter in the inner loop

$$\frac{d\omega}{dt} \Big|_c = -K_\omega (\omega - \omega_c) \quad (4.15)$$

The corresponding control torque in body-fixed coordinates follows from inverting the rotational dynamics

$$M_c = \omega \times I \omega - M_{a,Model} + I \frac{d\omega}{dt} \Big|_c \quad (4.16)$$

where $M_{a,Model}$ are the trim aerodynamic moments at the current flight conditions given by model data.

Note: For proper function of the time separation the time constant in the inner loop must be at least five times faster than the one in the outer loop. More information on the control system design can be found in [46] and [47].

4.3.3 Docked

Description

A docked attitude maintains the attitude of the parent vehicle. It can be used for vehicles docked to another vehicle, the parent vehicle. Additionally, a rotation offset may be specified.

Note: A docked attitude must be used with *Docked Equations of motion* (see Section 4.5.1.4).

Configuration

Settings for this attitude definition comprise a fixed rotation offset:

- *Sequence* is the rotation sequence of the offset
- *Euler angle 1* is the rotation angle around the 1st rotation axis of the sequence
- *Euler angle 2* is the rotation angle around the 2nd rotation axis of the sequence
- *Euler angle 3* is the rotation angle around the 3rd rotation axis of the sequence

4.3.4 Euler Angles

Description

To describe the orientation of the vehicle's body axes a set of Euler angles with respect to a coordinate frame can be used:

- **Yaw** angle ψ
- **Pitch** angle θ
- **Roll** angle ϕ is usually the angle about the body x_B -axis

The definition of each angle strongly depends on the coordinate system.

Configuration

The following settings are available for Euler angles:

- *Coordinate frame* is the coordinate system of the set of Euler angles. Available are:
 - *J2000* is the mean equator and equinox of epoch at J2000.0 coordinate system (see Section 9.3.2.1)
 - *L* is the local planetocentric coordinate system (see Section 9.3.4.1)
 - *LD* is the local planetodetic coordinate system (see Section 9.3.4.2)
 - *LO* is the local orbital coordinate system (see Section 9.3.1.1)
 - *N* is the inertial navigation frame coordinate system (see Section 9.3.6.4)
- *Control/State* defines how the Euler angles are considered in the dynamics:

DYNAMICS CONFIGURATION

- *Control* defines the set of Euler angles either as based on analytic attitude control laws, optimizable parameter or optimizable control. It depends on the selected control law of each aerodynamic angles as well as on further settings:
 - *Yaw angle* supports the following attitude control laws depending on the selected coordinate frame:
 - *Constant Law* (see Section 4.4.2)
 - *Indirect* (see Section 4.4.5)
 - *Inertial Tangential* (see Section 4.4.6)
 - *Linear Law* (see Section 4.4.7)
 - *Pointing Maneuver* (see Section 4.4.11)
 - *Profile Law* (see Section 4.4.13)
 - *Tangential* (see Section 4.4.17)
 - *Target Orbit* (see Section 4.4.18)
 - *Target Pointing* (see Section 4.4.19)
 - *Vertical* (see Section 4.4.23)
 - *Pitch angle* supports these control laws (depending on the coordinate frame):
 - *Bilinear Tangent* (see Section 4.4.1)
 - *Constant Law* (see Section 4.4.2)
 - *Gravity Turn* (see Section 4.4.4)
 - *Indirect* (see Section 4.4.5)
 - *Inertial Tangential* (see Section 4.4.6)
 - *Linear Law* (see Section 4.4.7)
 - *Pitch Over* (see Section 4.4.10)
 - *Pointing Maneuver* (see Section 4.4.11)
 - *Profile Law* (see Section 4.4.13)
 - *Required Velocity* (see Section 4.4.15)
 - *Target Pointing* (see Section 4.4.19)
 - *Thrust Arc Schwer* (see Section 4.4.21)
 - *Vertical* (see Section 4.4.23)
 - *Vertical Take Off* (see Section 4.4.25)
 - *Roll angle* is optional and by *default* it is set to zero. Depending on the coordinate frame, these control laws are supported:
 - *Constant Law* (see Section 4.4.2)
 - *Linear Law* (see Section 4.4.7)
 - *Profile Law* (see Section 4.4.13)
 - *Slip Free* (see Section 4.4.16)

DYNAMICS CONFIGURATION

- *Target Pointing* (see Section 4.4.19)
- *Vertical* (see Section 4.4.23)
- *State* is intended to be used for 6-DoF dynamics and defines the set of Euler angles as dynamic states. In this case, the quantities are integrated as function of the independent variable. No control is applied to the vehicle and the total forces and moments (aerodynamic, propulsive, gravitational, etc.) define the attitude of the vehicle that acts as an uncontrolled 6-DoF body. For more information please refer to Section 4.5.2.2.

Background

The definition of the Euler angles yaw, pitch and roll depends on their reference coordinate system:

- **Inertial axis system**, for example J2000 or TOD, has the standard Euler angles rotation sequence 3-2-1 and the Euler angles are defined as:
 - Yaw (from -180 deg to +180 deg) is the angle in the equator plane
 - Pitch (from -90 deg to +90 deg) is the angle out of the equator plane, negative in direction of the inertial z-axis
 - Roll (from -180 deg to +180 deg) is the angle between the vehicle's body-fixed lateral axis and its projection on the equator plane
- **Local planetocentric axis system** follows the 1-2-1 sequence and the Euler angles are defined as:
 - Yaw (from -180 deg to +180 deg) is the angle between the projection of body-fixed forward axis on the local horizontal plane and the x_V -axis of the local vertical reference axis (north)
 - Pitch (from -90 deg to +90 deg) is the angle between the vehicle's body-fixed forward axis and its projection on the local horizontal plane
 - Roll (from -180 deg to +180 deg) is the angle between the vehicle's body-fixed lateral axis and its projection on the local horizontal plane
- **Local orbital axis system** Euler angles are defined according to:
 - Yaw (from -90 deg to +90 deg) is the out of orbit plane angle, positive in direction of angular momentum
 - Pitch (from -180 deg to +180 deg) lies in the orbit plane, between the orbit along-track direction (perpendicular to the radial axis and pointing in velocity direction) and the body axis, clockwise positive
 - Roll (from -180 deg to +180 deg) is the angle about the body x_B -axis
- **Navigation frame** Euler angles depend on the selected axis definition:

- *Version 1* has the following definition:
 - Yaw (from -90 deg to +90 deg) is the out of x-z-plane angle, positive in direction of +y-axis
 - Pitch (from -180 deg to +180 deg) lies in the x-z-plane and is measured from the +x-axis to the projection of the body x-axis on the x-z-plane, clockwise positive
 - Roll (from -180 deg to +180 deg) is the angle about the body x_B -axis
- *Version 2* has the following definition:
 - Yaw (from -90 deg to +90 deg) is the out of x-z-plane angle, positive in direction of -z-axis
 - Pitch (from -180 deg to +180 deg) lies in the x-y-plane and is measured from the +x-axis to the projection of the body x-axis on the x-y-plane, counterclockwise positive
 - Roll (from -180 deg to +180 deg) is the angle about the body x_B -axis

4.3.5 Quaternion

Description

Quaternions uniquely describe the attitude and are free of singularities (like it is the case for angle representations). It is intended to be used for 6-DoF only.

Configuration

There is no configuration for this kind of attitude. The reference frame is always the inertial mean equator and equinox of epoch at J2000.0 axis system *J2000* (see Section 9.3.2.1). No control laws can be applied since the quaternion is defined as dynamic *State*. The forces and moments (aerodynamic, propulsive, gravitational) define the attitude of the vehicle acting as uncontrolled 6-DoF body. More information is given in Section 4.5.2.3.

Background

A quaternion consists of four elements:

- q_w is the scalar (real) part of the quaternion
- q_x is the first component of the vector part (imaginary) of the quaternion
- q_y is the second component of the vector part (imaginary) of the quaternion
- q_z is the third component of the vector part (imaginary) of the quaternion

4.3.6 Target Pointing

Description

The *Target pointing* attitude definition allows that two user defined axes of a spacecraft be aligned with two user defined target objects. The objects used for the direction definitions can be any celestial body, vehicle, ground station or point/area of interest present in the scenario. Nadir pointing and Sun pointing are two typical examples where the *Target pointing* attitude definition can be used.

Configuration

The *Target pointing* attitude definition is configured as follows:

- *Primary target* and *Secondary target* define the primary and secondary targets to which the spacecraft axes shall point. The primary pointing direction is exactly achieved, while the secondary pointing direction is achieved as close as possible. For example, in case that the primary target is the central body Earth and the secondary target of the spacecraft is the Sun, then primary pointing axis precisely points to the center of the Earth while the secondary pointing axis points to the center of the Sun as close as possible. This feature of the second pointing direction (smallest deviation from the defined direction) can be used for solar array alignment. The settings of a primary/secondary target and pointing axis are as follows:
 - *Target object* is the object at which the pointing axis is directed. The target object can be selected from a list of the available vehicles and celestial bodies in the scenario.
 - *Axis definition* is the type of the pointing axis definition:
 - *Main axis* is one of the spacecraft body axes (*Pointing axis*), namely +X, +Y, +X, +Z, -X, -Y or -Z
 - *Custom axis* defines an arbitrary axis in the spacecraft body fixed frame by:
 - *Yaw angle* is the angle in the spacecraft x-y-plane and defines the rotation about the spacecraft z-axis
 - *Pitch angle* is the angle out of the spacecraft x-y-plane and defines the rotation about the spacecraft y-axis

Background

In the *Target pointing* attitude definition it is not possible to define any additional constraints on the spacecraft attitude. In case such additional constraints may be required, the *Target pointing* control law (see Section 4.4.19) can be used.

4.3.7 Unit Vector

Description

Unit vector is one kind of attitude representation and is intended for orbital maneuvers, especially for low-thrust transfers. This unit vector consists of the following components:

- $\mathbf{U}_R$ is the radial component
- $\mathbf{U}_T$ is the transverse component (perpendicular to the radius)
- $\mathbf{U}_N$ is the normal component

Configuration

This kind of attitude representation has the options:

- *Control/State* is always set to *Control* and defines the unit vector components based on analytic attitude control laws or optimizable controls (depends on further settings of the control law)
- *Coordinate frame* is the coordinate system of the unit vector. Only *LO* is available, the local orbital axis system (see Section 9.3.1.1)
- *Control law* specifies the kind of the applied attitude control law
 - *Orbit Transfer* (see Section 4.4.9)
 - *Pollard Argument Periapsis* (see Section 4.4.12)
 - *Pollard Eccentricity* (see Section 4.4.12)
 - *Pollard Tangential* (see Section 4.4.12)
 - *Pollard Transverse* (see Section 4.4.12)
 - *Radial* (see Section 4.4.14)

4.3.8 From File

Description

From file uses the attitude history of a specified file. It is useful for situations when the attitude is already known from previous computations and no further optimization or post-computation is required.

Configuration

These configurations are required:

- *File type* defines the kind of file:
 - *CCSDS* is a text file according to the CCSDS standard for attitude ephemeris message data (AEM)

- *Simulation Structure File (*.struct)* is the struct file format, a binary output format of ASTOS and features:
 - *Offset* is optional and specifies an attitude offset of the spacecraft body axes applied on the attitude taken from the struct file. When *enabled*, the following input is required:
 - *Angle 1* is the first Euler angle applied on the attitude
 - *Angle 2* is the second Euler angle applied on the attitude
 - *Angle 3* is the third Euler angle applied on the attitude

- *Filename* specifies path and name of the file containing the attitude history

Background

In case of *CCSDS*, one attitude frame needs to be the body-fixed frame (SC_BODY, see Section 9.3.3.1) of the vehicle. The other attitude frame needs to be one of the following coordinate systems:

- EME2000, see Section 9.3.2.1
- TOD, see Section 9.3.2.3
- ITRF_93, ITRF_97, ITRF2000 or TDR, see Section 9.3.2.4

In case of *Simulation Structure File (*.struct)* and *enabled Offset*, the provided Euler angles are applied on top of the attitude as specified in the struct file. The rotation sequence is 1-2-3 and the offset angles can be understood as misalignment of the spacecraft body axes.

4.4 Control Laws

Description

A control law describes the behaviour of a control quantity. The former is usually only used during the initialization action when the scenario initial guess is created. The following kinds of control laws are available:

- **Analytic** control laws are computed according to an analytic formulas. The former are evaluated during the initialization, optimization and simulation actions and no optimizable quantities are created.
- **Optimizable** control laws also have an analytic law defined, which is usually a very simple law like a constant or linear history. But this analytic formula is only used during the initial guess generation. Whereas in simulation and optimization the behaviour of the control law is described by one or multiple optimizable quantities of these kinds:

DYNAMICS CONFIGURATION

- **Parameter** is an optimizable quantity with a constant value within a phase. It can be modified freely by the optimizer within the specified bounds
- **Control** is a function of the independent variable and its history can be freely changed by the optimizer within the specified bounds

Following is a complete list of available control laws:

- *Bilinear Tangent*, see Section 4.4.1
- *Constant Law*, see Section 4.4.2
- *Dynamic Pressure*, see Section 4.4.3
- *Gravity Turn*, see Section 4.4.4
- *Indirect*, see Section 4.4.5
- *Inertial Tangential*, see Section 4.4.6
- *Linear Law*, see Section 4.4.7
- *Maximum Lift to Drag Ratio*, see Section 4.4.8
- *Orbit Transfer*, see Section 4.4.9
- *Pitch Over*, see Section 4.4.10
- *Pointing Maneuver*, see Section 4.4.11
- *Pollard*, see Section 4.4.12
- *Profile Law*, see Section 4.4.13
- *Radial*, see Section 4.4.14
- *Required Velocity*, see Section 4.4.15
- *Slip Free*, see Section 4.4.16
- *Tangential*, see Section 4.4.17
- *Target Orbit*, see Section 4.4.18
- *Target Pointing*, see Section 4.4.19
- *Thrust Arc Schwer*, see Section 4.4.21
- *Trimmed Aero*, see Section 4.4.22
- *Vertical*, see Section 4.4.23
- *Vertical Geodetic*, see Section 4.4.24
- *Vertical Take Off*, see Section 4.4.25

Common control laws settings

Some control laws need additional input parameters that use specific independent identifiers or identifiers which are combined with the identifier of the control value. These specific input parameters are described in the corresponding section of the control law. But most control laws have the following settings in common:

- *Optimizable* is optional and by default it is *disabled*. No optimizable quantity will be created and the control law is always evaluated analytically. When *enabled*, optimizable quantity(ies) will be created for the control law:

DYNAMICS CONFIGURATION

- *Optimizable as Control* creates the corresponding control(s) of the control law
- *Optimizable as Parameter* creates the corresponding parameters of the control law
- *Optimizable as* is not present means that the optimizable quantity(ies) are automatically treated as *Control*

Furthermore the bounds of the optimizable quantity(ies) have to be specified by *lower/upper bound*. Each single optimizable quantity has its own specific bound settings and their values have to be appropriate for the control/optimizable quantity.

Note: The bounds of a *Control* are related to the control quantity itself whereas the bounds of a *Parameter* are related to the optimizable parameter of the control law, which is not necessarily the same!

Tip: Values and bounds can be imported from the currently loaded *TOPS* file, in case the related parameter or control is present in the file. Click the right mouse button on a value or bounds field of a control law and choose the menu item *Import from TOPS*. Up to three options are available: *Value*, *Bounds* or *Value and Bounds*.

Background

There might be a difference in the control laws between initial guess generation and optimization. Please note, this behaviour is not for all control laws. Some of them, like *Gravity Turn*, are analytical control laws and are always computed from the analytical formula. On the other hand there are control laws with optimizable quantities, either being Parameters or Controls.

The values specified in the GUI are always used for initial guess generation, but not necessarily for optimization or simulation. During optimization the optimizer changes all optimizable quantities. In case this is not desired, either fix the bounds of the corresponding control law, fix the bounds of the corresponding real parameter(s) in the *Real Parameters* panel or fix the bounds of the corresponding control in the *Grids* panel (both panels are located in the *Optimization* tree).

4.4.1 Bilinear Tangent

Description

The *Bilinear Tangent* control law is the solution to a very simple control problem, the "maximum velocity transfer to a rectilinear path" (see [52]). It is only available for the *Euler Angles* attitude definition's *Pitch* angle (see Section 4.3.4) of the local planetocentric axis system (L) (refer to Section 9.3.4.1), because the pitch angle is the only angle which impacts the orbit velocity.

Configuration

This control law requires the following settings:

DYNAMICS CONFIGURATION

- In the *Final orbit* pane the user can specify the values of the following two properties which shall be reached at the end of the current phase:
 - *Target altitude* specifies the altitude of the vehicle
 - *Radial velocity* specifies the radial velocity of the vehicle
- The *Thrust assumption* pane contains two settings:
 - *Fuel mass* defines the propellant mass to be spent during the phase. If set to *Default* it is computed by multiplying the fuel flow at the beginning of the phase by the phase duration.
 - *Average thrust* specifies the average thrust magnitude during the phase. If set to *Default* the thrust magnitude is set to be equal to the thrust magnitude at the beginning of the phase and is kept constant throughout the phase.
- The *Optimizable* setting is generic and is explained in Section 4.4

Background

This control law computes iteratively the initial and final pitch values such that the *Target altitude* is reached with the given *Radial velocity* at the end of the phase and while considering the *Average thrust* during the phase and the spent *Fuel mass*.

This computation is based on a bilinear tangent control law, whose general form is given by:

$$\tan\beta = \frac{c_2 t + c_4}{c_1 t + c_3} \quad (4.17)$$

where β is the control angle and c_i , $i = 1..4$ are constants. This is called a "bilinear tangent law", since the tangent of the control angle is the ratio of two linear functions. The coefficients c_i , $i = 1..4$ must be chosen in such a way that the end conditions specified (i.e. *Target altitude* and *Radial velocity* for the implementation in ASTOS) for this control law are satisfied. Assuming a constant thrust acceleration this can be done analytically. Although this bi-linear tangent law is valid only for a flat planet and a constant gravitational acceleration, it can also be used with good result for the guidance of rockets. In the latter case the control angle β is replaced by the pitch angle Θ , and the distance h by the periapsis altitude of the target orbit. Maximizing the horizontal velocity is then equivalent to maximizing the apoapsis of the orbit. This law is limited to trajectory parts where the assumptions mentioned above are nearly justified and where the atmospheric drag can be neglected.

For the application in ASTOS, since the pitch angle is known to be within the interval of $-\pi/2$ to $+\pi/2$ (a value outside this range would reduce the horizontal velocity component instead of increasing it), using the initial and the final pitch angle of the phase offers a convenient way to specify two of the required constants. The third parameter is chosen to be ξ , which is a measure of the curvature. In this case, by applying a proper substitution Eq. 4.17 can be rewritten as:

$$\tan\theta = \frac{\alpha^\xi \tan\theta_0 + (\tan\theta_f - \alpha^\xi \tan\theta_0) \hat{t}}{\alpha^\xi + (1 - \alpha^\xi) \hat{t}} \quad \text{with} \quad \hat{t} = \frac{(t - t_i)}{(t_{i+1} - t_i)} \quad (4.18)$$

DYNAMICS CONFIGURATION

where $a > 1$ is an arbitrary constant. An example for $a=100$ and ξ in the interval $[-1, +1]$ is shown in Fig. 4.4. $\xi=0$ results in a linear tangent law.

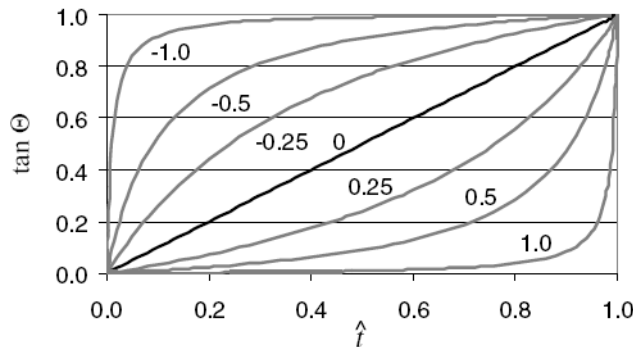


Fig. 4.4: Effect of the curvature parameter ξ for $a=100$. The boundary values are $\tan\Theta_0=0$, and $\tan\Theta_f=1$, respectively.

4.4.2 Constant Law

Description

When the *Constant Law* is used the control quantity is set to a constant value. It is available in the following attitude definitions:

- In the *Aerodynamic Angles* attitude definition this control law can be used for the *Angle of attack*, *Side slip angle* and *Bank angle* (see Section 4.3.1)
- In the *Euler Angles* attitude definition this control law can be used for the *Yaw*, *Pitch* and *Roll* angles (see Section 4.3.4)

Configuration

For the *Constant Law* the following settings are available:

- *Use (final) value from previous phase* specifies whether the control quantity value is set to be equal to the final value in the previous phase or not:
 - **Checked** means the value of the control quantity equals the final value in the previous phase. In the first phase of the scenario this value is set to zero.
 - **Unchecked** means the value does not depend on the final value in the previous phase and a value has to be specified by the user.
- *Optimizable* is generic and is explained in Section 4.4

4.4.3 Dynamic Pressure

Description

DYNAMICS CONFIGURATION

The *Dynamic pressure* control law is intended to be used for hypersonic winged vehicles during their ascent. The vehicle will follow a user specified maximum dynamic pressure trajectory. This control law is only available for the *Aerodynamic Angles* (refer to Section 4.3.1) *Angle of attack* and *Bank angle*.

The *Dynamic pressure* control law is based on a controller characterized by two gains which are the inverse of two time factors.

Note: In order to achieve the expected results, it is mandatory to configure both aerodynamic angles *Angle of attack* and *Bank angle* as *Dynamic pressure* control law with identical specifications! And the sideslip angle should be left unspecified.

Note: It is required to use this control law with a *Tabular* aerodynamics model (refer to Section 3.5.1.1 *Tabular Aerodynamics*) where the lift coefficient is specified as profile data table with *enabled Angle of attack reverse lookup* (refer to Profile Data Tables in User Manual).

Configuration

The following settings are available:

- *Dynamic pressure* is the upper dynamic pressure limit of the engine(s) specified as control law. Available options are:
 - *Constant Law*, see Section 4.4.2
 - *Linear Law*, see Section 4.4.7
 - *Profile Law*, see Section 4.4.13
- *Altitude gain* defines the gain in altitude per time unit and yields the commanded altitude rate
- *Airpath inclination gain* specifies the gain in the airpath inclination angle to yield the required vertical acceleration
- *Horizontal turn rate* defines the rate of turn of the vehicle in the local horizontal plane
- *Optimizable* is generic and explained in Section 4.4

Background

The base assumption behind this control law is that many airbreathing engines (e.g. turbojets, ramjets) show their best efficiency at their upper dynamic pressure limit. A controller characterized by two gains, the inverse of two time factors, is used to follow the specified dynamic pressure profile. This controller has been designed for advanced launch vehicles such as winged vehicles.

The difference between the current altitude and an estimated target altitude (both represented by the logarithm of the dynamic pressure) is multiplied by the *Altitude gain* (which is actually the inverse of the desired time constant). The resulting rate is compared to the air path inclination. This difference again is multiplied by the second time constant (the *Airpath inclination gain*) to give a required inclination rate. By entering this rate into the differential equation of γ , the required vertical acceleration is computed.

DYNAMICS CONFIGURATION

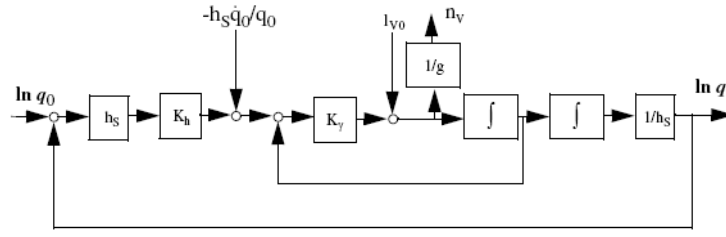


Fig. 4.5: Dynamic pressure controller

To derive the controller, a locally exponential atmosphere is assumed

$$p \approx p_0 e^{-h/h_s} \quad (4.19)$$

A part of the required altitude rate can be attributed to the change in dynamic pressure

$$\dot{q}_0 = \frac{1}{2} V^2 \dot{\rho} + \rho V \dot{V} = q_0 \left(\frac{\dot{p}}{p} + 2 \frac{\dot{V}}{V} \right) = q_0 \left(2 \frac{\dot{V}}{V} - \frac{\dot{h}}{h_s} \right) \quad (4.20)$$

or, since the velocity change is usually very small compared to the (supersonic) velocity

$$\dot{h}_q = -h_s \frac{\dot{q}_0}{q_0} \quad (4.21)$$

Since for a given velocity the dynamic pressure is proportional to the atmospheric density, the difference of the current altitude h and h_0 , the one corresponding to the commanded dynamic pressure, is

$$\Delta h = h_s \left(\ln \frac{\rho_0}{2q/V^2} - \ln \frac{\rho_0}{2q_0/V^2} \right) = h_s \ln \frac{q_0}{q} \quad (4.22)$$

By choosing a suitable time constant T_h for the altitude, we can get the commanded altitude rate

$$\dot{h}_c = \frac{\Delta h}{T_h} - h_s \frac{\dot{q}_0}{q_0} \quad (4.23)$$

This value can be compared with the current altitude rate $V \cos \gamma$ and divided by a time constant to get the altitude acceleration needed

$$\ddot{h}_c = \frac{\dot{h}_c - V \sin \gamma}{T_h} \quad (4.24)$$

When inserting this term into the differential equation of the flight-path angle (Eq. 4.54), with the approximation

$$\ddot{h}_c = \dot{V} \sin \gamma + V \dot{\gamma} \cos \gamma \approx V \dot{\gamma} \quad (4.25)$$

one can determine the vertical load factor

$$n_V g = \ddot{h}_c + \left(g - \frac{V^2}{R} \right) \cos \gamma - 2\omega_E V \sin \chi \cos \delta - R\omega_E^2 \cos \delta (\sin \delta \sin \gamma \cos \chi + \cos \delta \cos \gamma) \quad (4.26)$$

The time factors of the two loops are user-definable and should differ at least by a factor of five in order to satisfy the time separation condition.

4.4.4 Gravity Turn

Description

During a gravity turn maneuver the pitch angle in the local planetocentric axis system (L) equals the air path velocity inclination angle. Thus the vehicle's x_B -axis points in the direction of the air path velocity vector. To model this the *Gravity Turn* control law is available in the *Euler Angles* attitude definition for the *Pitch* angle (see Section 4.3.4) of the local planetocentric axis system (L) (refer to Section 9.3.4.1).

Note: The yaw angle is independent of the gravity turn maneuver and can be subject to its own control law. Therefore the angle of attack is not necessarily zero. The use of the control law *Tangential* (see Section 4.4.17) for the yaw angle ensures a zero angle of attack, but this may lead to an undesired yaw angle.

Configuration

Only the generic setting *Optimizable* is available, which is explained in Section 4.4.

Background

At the phase transition between the first phase in which a gravity turn is used and the preceding phase a jump in the value of the pitch angle may occur due to the settings in both phases. In order to prevent a jump in the value and in order to ensure that the pitch is continuous in the optimal solution, a final boundary constraint *Gravity Turn Condition* must be defined in the *Constraints* panel (see Constraints in User Manual and Section 6.4).

4.4.5 Indirect

Description

The *Indirect* control law is intended to be used with indirect equations of motion. It is only available for the *Euler Angles* attitude definition's *Yaw* and *Pitch* angles (see Section 4.3.4) of the local planetocentric axis system (L) (refer to Section 9.3.4.1).

Note: Indirect equations of motion are currently not supported. In case of interest, please contact us at service@astos.de

Configuration

Only the generic setting *Optimizable* is available, which is explained in Section 4.4.

4.4.6 Inertial Tangential

Description

DYNAMICS CONFIGURATION

By using the *Inertial Tangential* control law the yaw and pitch angles can be set to be equal to the inertial flight path heading and inclination angle, respectively. This control law is almost equivalent to *Gravity Turn* (see Section 4.4.4) and *Tangential* (see Section 4.4.17), except that the inertial flight path velocity vector is used as a reference (instead of the air path velocity vector). The *Inertial Tangential* control law is useful once the aerodynamic force is small (e.g. vehicle outside the dense atmosphere layers). It is only available for the *Euler Angles* attitude definition's *Yaw* and *Pitch* angles (see Section 4.3.4) of the local planetocentric axis system (L) (refer to Section 9.3.4.1).

Configuration

Only the generic setting *Optimizable* is available, which is explained in Section 4.4.

4.4.7 Linear Law

Description

A linear control law is a linear function of the independent variable (usually the mission time) defined by an initial and a final value of the linear function. Both the initial and the final value can be optimized. This law is available in the following attitude definitions:

- In the *Aerodynamic Angles* attitude definition this control law can be used for the *Angle of attack*, *Side slip angle* and *Bank angle* (see Section 4.3.1)
- In the *Euler Angles* attitude definition this control law can be used for the *Yaw*, *Pitch* and *Roll* angles (see Section 4.3.4)

Configuration

For the *Linear Law* the following settings are available:

- *Initial* is the value of the control quantity at the beginning of the current phase
- *Use (final) value from previous phase* specifies whether the control quantity value is set to be equal to the final value in the previous phase or not:
 - **Checked** means the initial value of the control quantity equals the final value in the previous phase. In the first phase of the scenario this value is set to zero.
 - **Unchecked** means the initial value does not depend on the final value in the previous phase and a value has to be specified by the user.
- *Slope defined by* specifies how the final value of the linear function is computed:
 - *Final* allows explicitly to specify the *Final* value. Thus the final value at the end of the phase is fixed while the slope is computed internally and depends on the phase duration.
 - *Rate* specifies the value for the linear law's slope, i.e. *Rate*. In this case, the slope of the linear law is fixed while the final value depends on the duration of the phase.

Tip: In case the control quantity is defined as optimizable *Control*, the rate can be fixed using the appropriate constraint. For example, in case of Euler angle pitch the corresponding constraint is *Pitch Rate* (see Section 6.4).

- *Optimizable* is generic and explained in Section 4.4

4.4.8 Maximum Lift to Drag Ratio

Description

The *Maximum Lift to Drag Ratio* control law ensures that the maximum lift to drag ratio for a vehicle is maintained. The control law can be used for winged vehicles during their ascent in order to achieve a big enough velocity so that the *Dynamics Pressure* control law can be used. This control law is only available for the *Angle of attack* of the *Aerodynamic Angles* attitude control (refer to Section 4.3.1).

Configuration

Only the generic setting *Optimizable* is available, which is explained in Section 4.4.

Background

The maximum lift to drag ratio is evaluated from the aerodynamics model. The resulting angle of attack value stays within the range provided in this aerodynamics model.

4.4.9 Orbit Transfer

Description

The *Orbit transfer* is an advanced control law and combines several other control laws based on the Pollard control laws (see Section 4.4.12). The *Orbit transfer* law is named after the Keplerian orbit elements which are mainly affected by it. Due to the fact that this control law combines multiple control laws in one single control law, it is suitable for special applications like multi-revolution low-thrust orbit transfers where the control has to follow a certain profile in a single phase (e.g. initial orbit is LEO, raise semi-major axis to 20,000 km, change eccentricity to 0.3, then change inclination to 25 deg). This law is only available for *Unit Vector* (see Section 4.3.7).

Configuration

For the *Orbit transfer* the following settings are available:

- *Type* specifies the kind of attitude control law used for the orbit transfer:
 - *Argument Periapsis* is based on the control law *Pollard Argument Periapsis*

DYNAMICS CONFIGURATION

- *Eccentricity* is based on the control law *Pollard Eccentricity*
- *Inclination* is based on the control law *Pollard Transverse* with an out-of-plane angle set to $\pm 90^\circ$ (depends on inclination change direction)
- *Semimajor Axis* is based on the control law *Pollard Tangential*

More information about the Pollard control laws is given in Section 4.4.12. Combinations of different control law types can be realized by clicking the **Add** button to define a new control set. Clicking the **Remove** button deletes an undesired control set.

- *Target value* is the target value to be reached for the chosen quantity (for example, in case of *Inclination* it is the desired orbit inclination value)
- *Out-of-plane angle* is the out-of-orbital plane thrust angle for a combined inclination change. This setting is not available for *Inclination*.
- *Optimizable* is generic and is explained in Section 4.4

Background

When the phase in which the *Orbit transfer* law is defined starts, the control law defined by the first *Type* is active. Once its *Target value* is reached, the control law defined by the next *Type* is executed. This procedure continues until the control law defined by the last *Type* is executed. After its *Target value* has been reached, the control law *Pollard Tangential* is applied until the phase ends. For this control law the out-of-plane angle is set to zero so that all thrust is applied in velocity direction.

Tip: The eccentricity and inclination are defined as non-negative quantities. Therefore, if their target value is set to zero by the user, the gradient based optimization algorithm will fail to compute any gradient for this value. The user can prevent this by using small positive values (for example 0.01 for eccentricity and 0.1° for inclination) instead of 0.0 as target values.

4.4.10 Pitch Over

Description

The *Pitch Over* control law is used for launchers during the trajectory transition from the vertical lift-off phase to a trajectory phase with a well defined heading. The trajectory transition consists of three different phases during the launch:

1. vertical lift-off with constant pitch angle
2. pitch-over maneuver to increase the difference between the pitch angle and the air path inclination angle
3. alignment of the pitch angle with air path inclination angle to meet the gravity turn condition (vehicle's x_B -axis pointing in the direction of the air path velocity vector)

This control law is only available for the *Euler Angles* attitude definition's *Pitch* angle (see Section 4.3.4) defined in the local planetocentric axis system (L) (refer to Section 9.3.4.1).

Configuration

DYNAMICS CONFIGURATION

For the *Pitch Over* control law the following settings are available (see the **Background** section for an explanation of the symbols used below):

- *Acceleration time* sets the duration Δt_{acc} with constant pitch angle (the vertical lift-off) between the beginning of the phase t_i and t_{acc}
- *Delta pitch* is the maximum difference between the pitch angle and the air path inclination angle at t_{po}
- *Pitchover duration* is the duration Δt_{po} between the end of the vertical lift-off (t_{acc}) and maximum delta pitch angle (t_{po})
- *Time constant* is the time t_c used in the equation defining the exponential decay reducing the delta pitch to zero
- *Optimizable* is generic and explained in Section 4.4

Background

A gravity turn maneuver generally cannot be flown right after the start of a launcher. The transitional maneuver between the start and the gravity turn maneuver is the pitch-over maneuver during which the trajectory air path inclination angle γ changes from 90 deg to a well defined value. This maneuver realizes the following sequence:

$$\theta = \gamma \quad \text{for} \quad t_i \leq t \leq t_i + \Delta t_{acc} \quad (4.27)$$

$$\theta = \gamma - \frac{t - (t_i + \Delta t_{acc})}{\Delta t_{po}} \Delta \theta \quad \text{for} \quad t_i + \Delta t_{acc} \leq t \leq t_i + \Delta t_{acc} + \Delta t_{po} \quad (4.28)$$

$$\theta = \gamma - \Delta \theta e^{-(t - (t_i + \Delta t_{acc} + \Delta t_{po})) / t_c} \quad \text{for} \quad t_i + \Delta t_{acc} + \Delta t_{po} \leq t \leq t_{i+1} \quad (4.29)$$

or

$$\theta = \gamma - \frac{t_{i+1} - t}{t_{i+1} - (t_i + \Delta t_{acc} + \Delta t_{po})} \Delta \theta \quad \text{for} \quad t_i + \Delta t_{acc} + \Delta t_{po} \leq t \leq t_{i+1} \quad (4.30)$$

where Θ is the pitch angle, $t_i \leq t_{acc} < t_{po} < t_{i+1}$ are: the initial time of the phase, the end of an acceleration time, the end of the pitch-over rotation, and the phase final time, respectively. Whenever the time constant t_c is not specified, Eq. 4.30 is used for reducing the difference between pitch and air path angle linearly; otherwise the exponential function in Eq. 4.29 is used (see Fig. 4.6 for an example).

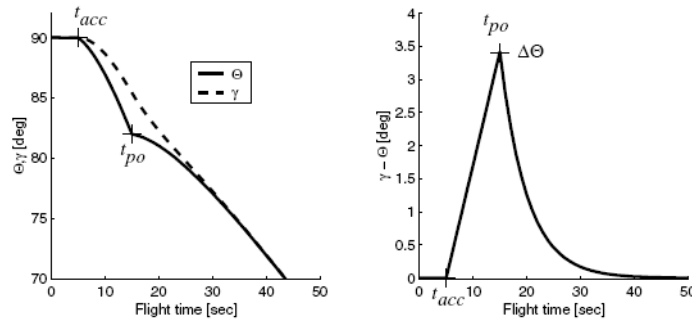


Fig. 4.6: Air path inclination angle γ and pitch angle Θ and their difference $\Delta\Theta$ over time during a pitch over maneuver

DYNAMICS CONFIGURATION

The pitch over duration of the example given is Δt_{po} . The pitch difference $\Delta\Theta = \gamma - \Theta$ is a measure for the progress of the pitch over maneuver. When the acceleration time is set to zero, the pitch over maneuver starts at the beginning of the phase.

In the last sequence of the phase the time constant t_c is governing the exponential decay. It should be sufficiently small compared to the remaining time of the phase in order to make sure that the residual difference $\Delta\Theta$ is small at the end of the phase.

This control law can also be used to reduce an existing difference between the pitch and the air path angles at the beginning of a phase. The reduction of the difference happens over the time span of the phase. In this case, *Delta pitch*, *Acceleration time* and *Pitchover duration* can be left disabled and *Delta pitch* is then initialized using the angle values at the beginning of the phase.

Optimization

This control law can be used both to generate an initial guess or during optimization. In the latter case the pitch difference $\Delta\Theta$ is the only optimizable parameter. When a collocation method is used for optimization, a collocation grid node must be placed at t_{acc} and at t_{po} , otherwise the lack of differentiability in the sharp bends of the control quantity at these points prevent the convergence of the optimization.

Instead of this special pitch-over maneuver the standard formulation modeled by two phases can also be used. In this case, the first pitch-over phase should be set to use a *Linear Law* for the *Pitch angle* with a *Final value* of about 82° , and the second phase should use a *Constant Law* for the *Pitch angle*. Both final phase times can be optimizable. In case the second phase (constant pitch angle) shall end with the gravity turn condition, a final boundary constraint (either an *Angle Of Attack* constraint equal zero or a *Gravity Turn Condition* constraint) has to be added to the optimal control problem.

4.4.11 Pointing Maneuver

Description

This control law is intended for special applications and internal use. The behaviour is similar to a linear law (see Section 4.4.7), but the final value is analytically computed based on the difference between the current position and the reference one. It is available for the *Euler Angles Yaw, Pitch and Roll* (see Section 4.3.4) defined in the local planetocentric axis system (L) (refer to Section 9.3.4.1).

Note: In case of interest please contact us at service@astos.de.

Configuration

Only the generic setting *Optimizable* is available, which is explained in Section 4.4.

4.4.12 Pollard

Description

The *Pollard* attitude control laws are implemented according to [84] and are perfectly suited for performing low-thrust orbit transfers. Four different kinds of *Pollard* attitude control laws are implemented in ASTOS:

- *Pollard Argument periapsis* sets the main vehicle axis (+x-axis) parallel to the major axis of the orbit. If in this case low thrust is applied for at least a complete orbit or longer, this will affect mainly the orbit argument of periapsis.
- *Pollard Eccentricity* sets the main vehicle axis (+x-axis) perpendicular to the major axis of the orbit. If in this case low thrust is applied for at least a complete orbit or longer, this will affect mainly the orbit eccentricity.
- *Pollard Tangential* sets the main vehicle axis (+x-axis) aligned with the inertial velocity direction (tangent to the orbit path). If in this case low thrust is applied for at least a complete orbit or longer, this will affect predominantly the orbit semi-major axis and to a smaller extent the eccentricity.
- *Pollard Transverse* sets the main vehicle axis (+x-axis) perpendicular to the orbit radius vector. Similar to the *Pollard Tangential*, if in this case low thrust is applied for at least a complete orbit or longer, this will affect predominantly the orbit semi-major axis and to a smaller extent the eccentricity.

All these attitude control laws are only available for the attitude definition *Unit Vector* (see Section 4.3.7).

Configuration

The following settings are required for each kind of *Pollard* law:

- *Use reverse law* defines whether this control law aligns the -x-axis (*enabled*) or the default +x-axis (*disabled*) of the vehicle in the directions specific to each of the Pollard control laws (see the **Description** section above). If a thrust force is active, in case of *disabled* the vehicle is accelerated and in case of *enabled* the vehicle is decelerated.
- *Out-of-plane angle* is optional and defines a constant angle between the vehicle main axis and the orbital plane. This angle can take values between -90 and +90 degree. If a thrust force is active, a positive out-of-plane angle increases continuously the orbit inclination and a negative out-of-plane angle decreases continuously the orbit inclination. Moreover, in case of a positive out-of-plane angle the angle between the thrust vector and the orbital momentum vector is smaller than 90 degrees when the vehicle is above the equatorial plane and bigger than 90 degrees when the vehicle is below the equatorial plane to ensure the continuous increase of of the orbit inclination.
- *Optimizable* is generic and explained in Section 4.4

4.4.13 Profile Law

Description

In case of a profile law the control quantity is a function of one or multiple independent variables, such as the flight time. The relationship between the control quantity and the independent variables is provided as table. This control law is available for:

- *Aerodynamic Angles* Angle of attack, Side slip angle and Bank angle (see Section 4.3.1)
- *Euler Angles* Yaw, Pitch and Roll (see Section 4.3.4)

Configuration

The following settings are available:

- *Profile* defines the control quantity as function of the independent variable(s). The configuration of this panel is detailed in chapter Profile Data Tables of the book User Manual.
- *Optimizable* is generic and explained in Section 4.4

Background

Below follows a list of the supported independent variables:

- *Flight_Time* is the actual time of flight (starting at zero at the beginning of the first phase)
- *Mission_Time* is the actual mission time (starting at the initial time offset at the beginning of the first phase)
- *Phase_Time* is the actual time of the phase (starting at zero at the beginning of the phase)
- *Altitude* is the actual altitude of the vehicle
- *Airpath_Velocity* is the magnitude of the actual air path speed of the vehicle
- *Mach* is the actual Mach number of the vehicle
- *Dynamic_Pressure* is the actual dynamic pressure of the vehicle
- *Latitude* is the actual (geodetic) latitude of the vehicle
- *Longitude* is the actual longitude of the vehicle (w.r.t. prime meridian)

4.4.14 Radial

Description

This control law applies thrust in radial direction to raise the orbit radius. It is only available for *Unit Vector* (see Section 4.3.7).

Configuration

The following settings are available:

- *Use reverse law* defines whether the control law defines the attitude in the opposite direction (*enabled*) or default direction (*disabled*). Here, the default direction is raising the orbit radius (in positive radial direction, zenith), while the opposite is lowering the orbit radius (in negative radial direction, nadir)
- *Out-of-plane angle* is optional and defines the out-of-plane thrust angle (between -90 and +90 degree)
- *Optimizable* is generic and explained in Section 4.4

Additionally, an out-of-plane thrust angle *Beta value* can be defined.

4.4.15 Required Velocity

Description

This control law is based on the required velocity concept of Battin (Q-guidance, see [32], [43], [44]). For a current point, the velocity that a vehicle on the target orbit should have is computed given the current radius. The thrust vector is then oriented reducing the difference between the current velocity vector and the required velocity. It is only available for *Euler Angles Pitch* (see Section 4.3.4) defined in the local planetocentric axis system L (refer to Section 9.3.4.1).

Configuration

The following settings are available:

- *Target apoapsis altitude* is the altitude of the apoapsis of the target orbit w.r.t. the equatorial radius of the central body
- *Target periapsis altitude* is the altitude of the periapsis of the target orbit w.r.t. the equatorial radius of the central body
- *Default pitch* is used as pitch angle when computation is possible, see section below
- *Min. thrust to effective weight ratio* specifies the minimum ratio between the thrust magnitude and the effective weight (weight minus the centrifugal force due to the horizontal velocity) using the Q-guidance for the pitch computation
- *Optimizable* is generic and explained in Section 4.4

Background

The control law computes the required pitch angle out of the differences between the current and required component of radial and horizontal velocity. A zero pitch angle is used when the current apoapsis is larger than the required apoapsis. The pitch angle becomes +/- 90° when the required horizontal velocity is already too high. Then only the radial velocity is increased or decreased according to the required change of the radial velocity. And the *Default pitch* angle is used when:

- The current radius is lower than the specified *Target periapsis altitude*. Reducing the target periapsis altitude in that particular phase might solve this issue.

- The effective weight (i.e. the weight minus the centrifugal force due to the horizontal velocity) is larger than the thrust and there is no acceleration left to change the velocity vector towards the desired direction. The minimum thrust to effective weight ratio governs the onset of the Q-guidance. *Min. thrust to effective weight ratio* can be reduced below 1.0 as the vehicle velocity reduces the necessary velocity for additional acceleration.

4.4.16 Slip Free

Description

The *Slip free* attitude control law evaluates the roll angle assuming an aerodynamic sideslip angle of zero. This law is only available for *Euler Angles Roll* (see Section 4.3.4) defined in the local planetocentric axis system (L) (refer to Section 9.3.4.1).

Configuration

Only the generic setting *Optimizable* as *Control* is available (please refer to Section 4.4).

4.4.17 Tangential

Description

Tangential defines the yaw angle in the local planetocentric axis system (L) equal to the flight path heading angle of the air path velocity. This law is only available for *Euler Angles Yaw* (see Section 4.3.4) defined in the local planetocentric axis system (L) (refer to Section 9.3.4.1).

Tip: It can be combined with the control law *Gravity Turn* (see Section 4.4.4) for the Euler angle *Pitch*.

Configuration

Only the generic setting *Optimizable* is available, which is explained in Section 4.4.

4.4.18 Target Orbit

This control law computes a yaw angle to reach the specified orbit inclination. It is suitable for launcher trajectories and is only available for *Euler Angles Yaw* (see Section 4.3.4) in the local planetocentric axis system (L) (refer to Section 9.3.4.1).

Configuration

The following settings are available:

DYNAMICS CONFIGURATION

- *Target inclination*
- *Orbit leg* specifies the orbital leg where to reach the specified target orbit inclination and has these options:
 - *Ascending* is the ascending branch of the orbit, after passing through the ascending node
 - *Descending* is the descending branch of the orbit, after passing through the descending node
 - *Automatic* computes automatically whether the ascending or descending branch is more suitable
- *Optimizable* is generic and explained in Section 4.4

Background

Whenever the current latitude is outside the specified inclination, a suitable default value for the yaw angle is selected (pointing exactly North, East, South or West), depending on the target inclination.

In case the *Orbit leg* is specified as *Automatic*, the ascending leg of the target orbit is assumed when the current heading is between -90° and $+90^\circ$. Otherwise the descending leg is assumed. Specifying the orbit leg is useful in situations without sizable horizontal velocity, for example during the first phase of a rocket launch.

Note: The inclination may slightly deviate from the inclination specified as final boundary constraint, even after optimization.

4.4.19 Target Pointing

Description

The *Target pointing* control law allows that an axis chosen by the user be directed to a target object. This is achieved through a rotation about the axis for which the target pointing control law was chosen. Nadir pointing and Sun pointing are two examples where the *Target pointing* control law can be used. The latter allows the spacecraft to pointing to any scenario defined celestial body or vehicle. Unlike the *Target Pointing* attitude definition (see Section 4.3.6) which is used to align two spacecraft axes with two user defined directions, the *Target pointing* control law allows the target pointing of only one spacecraft main axis. It additionally offers the possibility to introduce a constraint on the rotation angle about the axis for which it is defined. The *Target pointing* control law is available in the *Euler Angles* attitude definition for the *Yaw*, *Pitch* and *Roll* angles (see Section 4.3.4) according to their definition with respect to the selected *Coordinate frame*.

Configuration

The *Target pointing* control law is configured as follows:

- *Target object* specifies the target for the *Pointing axis*. and is selected from the list of available vehicles and celestial bodies in the scenario.
- *Pointing axis* is one of the spacecraft body axes, namely +X, +Y, +X, +Z, -X, -Y or -Z

Note: The pointing axis can not be identical with the rotation axis of the attitude angle, which strongly depends on the selected coordinate frame.

- *Lower/Upper bound* specifies the limits of the attitude angle to be maintained (if applicable) about the axis for which the control law was defined. This feature can be used in case the spacecraft rotation is constrained by for some reason while still retaining the target pointing within this range.

Background

A target pointing attitude definition (see Section 4.3.6) features a primary and a secondary pointing direction, where the primary pointing direction is exactly achieved and the secondary pointing direction is achieved as close as possible. Both the primary and the secondary pointing directions are distributed along the three rotations about the x-, y- and z-axis of the coordinate frame. For the *Target pointing* control law depending on the specified attitude *Coordinate frame*, a rotation about one axis corresponds to one of the Euler angles yaw, pitch and roll: the roll angle is always about the spacecraft x-axis, while yaw and pitch depend on the selected coordinate frame. For example, the yaw angle in the local planetocentric coordinate system (see Section 9.3.4.1) is about the spacecraft x-axis, while in the inertial coordinate frame (for example J2000, see Section 9.3.2.1) it is about the z-axis.

Now one example is provided: The z-axis of the vehicle shall point to the center of the central body (i.e. nadir). One option is to use the attitude *Defined as Euler Angles, Control/State* set to *Control* and the *Coordinate frame* defined as *L* (the local planetocentric coordinate system). In this coordinate system, the z_{LO} -axis is always pointing north, while the y_{LO} -axis points east and the x_{LO} -axis points upward in radial direction. To have the z_B -axis pointing to the center of the central body, the $pitch_{LO}$ angle has to be modified. The *Control law* of the *Pitch angle* needs to be *Target Pointing*, and the *Target object* must be set to the same celestial body as the current central body. As *Pointing axis* +Z needs to be specified. Further, the *Yaw angle* and *Roll angle* can be specified as *Constant Law*, since they do not have any impact on this target pointing.

4.4.20 Target Velocity Pointing

Description

The *Target Velocity Pointing* control law is only available for vehicle elements (e.g. sensors) and allows that an axis chosen by the user to be in the direction of the relative velocity vector of a target with respect to the vehicle. This can be used, for example, to turn a sensor in the direction of the relative velocity of an approaching dust cloud.

DYNAMICS CONFIGURATION

The *Target Velocity Pointing* control law is available when the *Orientation offset* is set to **Enabled** in the vehicle element's (e.g. sensor's) *Placement* settings in the *Vehicles & POIs Definition*.

Configuration

The *Target Velocity Pointing* control law is configured as follows:

- *Target object* specifies the target for the computation of the *Pointing axis* and is selected from the list of available vehicles in the scenario. *Target object* should be different from the one on which the vehicle element (e.g. sensor) is mounted.
- *Pointing axis* is one of the spacecraft body axes, namely +X, +Y, +X, +Z, -X, -Y or -Z
- *Lower/Upper bound* specifies the limits of the attitude angle to be maintained (if applicable) about the axis for which the control law is defined. This feature can be used in case the spacecraft rotation is constrained for some reason while still retaining the target pointing within this range.

Background

The velocity of the target object is computed relative to the vehicle on which the vehicle element (e.g. sensor) is mounted. The pointing axis $\mathbf{p}$ is then in the direction opposite to the normalized relative velocity vector $\mathbf{v}$ so that $\mathbf{p} = -\mathbf{v}$.

4.4.21 Thrust Arc Schwer

Description

The control law thrust arc Schwer is intended to be used for low-thrust orbit transfers and is based on the dissertation of Andreas Schwer. It is only available for *Euler Angles Pitch* (see Section 4.3.4) defined in the local orbit (LO) axis system (refer to Section 9.3.1.1).

Configuration

Only the generic setting *Optimizable* is available, which is explained in Section 4.4.

4.4.22 Trimmed Aero

Description

The trimmed aero control law computes the angle of attack from the actual aerodynamics model for trimmed flight (pitch moment is zero). It is only available for *Aerodynamic Angles Angle of attack* (see Section 4.3.1).

Note: The trimmed aero control law is intended for special applications only and has not been completely verified and validated.

Configuration

The following settings are available:

- *Profile* defines the control quantity as function of the independent variable(s). The configuration of this panel is detailed in chapter Profile Data Tables of the book User Manual.
- *Optimizable* is generic and explained in Section 4.4

Background

This law computes the angle of attack as a function of the independent variables (same is done in control law *Profile*, refer to Section 4.4.13). But here the aerodynamics model computes its own angle of attack value for trimmed flight condition. It means that there is no pitch moment. Since this is a root finding problem using the Wijngaarden-Dekker-Brent method, the computation can be slower.

4.4.23 Vertical

Description

The vertical law defines the aerodynamic bank angle as zero. This can be used e.g. for a pull-up maneuver when all lift is used in the vertical plane (hence the name). It is only available for *Aerodynamic Angles Bank angle* (see Section 4.3.1).

Configuration

No settings are available for this control law.

4.4.24 Vertical Geodetic

Description

For vertical take-off phases of launchers or vertical landing of boosters the pitch angle is fixed at 90 degrees in the local planetodetic coordinate system (LD) (see Section 9.3.4.2. This law is only available for the *Euler Angles Yaw* and *Pitch* defined in the *Coordinate frame J2000* (see Section 4.3.4).

Configuration

Only the generic setting *Optimizable* (refer to Section 4.4) is available to optimize the control quantity as **Control** (it is not possible to define the control quantity as optimizable **Parameter**).

Background

The pitch angle is considered in relation to the local planetodetic axis system (LD). Thus, the thrust direction is perpendicular to the local surface of the central body and pointing downwards.

This control law is usually applied for vertical landing scenarios as in case of fly-back boosters or landers on celestial bodies with thin (or even without) atmosphere.

4.4.25 Vertical Take Off

Description

For vertical take-off phases of launchers the pitch angle is fixed at 90 degrees in the local planetocentric coordinate system (L) (see Section 9.3.4.1. This law is only available for *Euler Angles Pitch* defined in *Coordinate frame L* (see Section 4.3.4).

Configuration

Only the generic setting *Optimizable* (refer to Section 4.4) is available to optimize the control quantity as **Control** (it is not possible to define the control quantity as optimizable **Parameter**).

Background

The pitch angle is considered in relation to the local planetocentric axis system (L). Thus, the thrust direction is parallel to the radius vector of the vehicle w.r.t. the central body. In case of an oblate central body, the thrust is not perpendicular to the central body surface, but the maximum angular error (mid-latitude) on Earth is 0.2°. Since this control law is usually applied only in the first 70-100 meters of vertical take-off (launch-pad height), the error in terms of the final position at the end of the phase is below 0.4 m in north (or south) direction. For example, for a launch in Kourou the error is about 0.02 m.

4.5 Equations of Motion

Description

Equations of motion describe the behavior of a physical system in terms of a function of the independent variable (e.g. the time). Typically the equations of motion (EoM) are a set of mathematical functions. The motion of the dynamics is described by the differential equations. The dynamic system models described in the following are divided into the following categories:

- **Translational** equations of motion are the dynamics of the position and velocity state, see Section 4.5.1
- **Rotational** equations of motion are the dynamics of the attitude and rotational state, see Section 4.5.2
- **Auxiliary** equations of motion are the dynamics of additional states like the propellant consumption. More details are given in Section 4.5.3

4.5.1 Translational Dynamics

Description

Translational motion is the movement of a rigid body changing its position and velocity. Various sets of translational dynamics are available:

- *Accelerate Flightpath* uses a fixed flight path velocity direction but with variable speed, see Section 4.5.1.1
- *Circular Restricted Three Body Problem* describes the translational motion within the gravitational field of two massive bodies, see Section 4.5.1.2
- *Circular Restricted Three Body Problem Analytical* describes the translational motion within the gravitational field of two massive bodies based on analytical formulas, see Section 4.5.1.3
- *Docked* considers the translational dynamics of a specified parent vehicle, see Section 4.5.1.4
- *Equinoctial Elements* provides the modified equinoctial elements, see Section 4.5.1.5
- *Flightpath Velocity* provides the flight path velocity with the relative position vector, see Section 4.5.1.6
- *Inertial Cartesian* is based on the cartesian components of the inertial position and velocity vector, see Section 4.5.1.7
- *Inertial Velocity* is based on the inertial flight path velocity and the relative position vector components, see Section 4.5.1.8
- *Kepler Orbit* provides the Keplerian orbital elements with all but the true anomaly to be considered constant, see Section 4.5.1.9
- *Launch Pad* provides zero flight path velocity and zero derivatives of the relative position vector, see Section 4.5.1.10
- *(Flexible) Multibody Dynamics* is for multibody systems where each vehicle element has its own translational dynamics, see Section 4.5.1.11
- *Perturbed Kepler Orbit* provides the Keplerian orbital elements including perturbing forces, see Section 4.5.1.12
- *SGP4* is based on NASA's SGP4 algorithm for the calculation of near Earth satellites, see Section 4.5.1.13
- *State from File* considers the translational motion provided in an external file, see Section 4.5.1.14

Background

To decide which set of equations of motion is most suitable for a certain kind of trajectory/ scenario the following hints are given:

- Zero relative velocity w.r.t. the central body surface (e.g. vehicle on launch pad) cannot be modelled with flight path states, instead *Launch Pad* EoM should be used
- Flight path inclination angle of 90° , a vertical ascent, cannot be modelled with flight path states as the heading angle is ill-defined, instead *Inertial Velocity* EoM should be used

DYNAMICS CONFIGURATION

- Polar flyovers cannot be modelled with flight path or inertial velocity states as the longitude is ill-defined at the poles, instead *Inertial Cartesian* EoM should be used
- Rail trajectories, for example during the launch of sounding rockets, should use *Accelerate Flightpath* EoM
- Interplanetary trajectories should use *Equinoctial Elements*, *Inertial Cartesian*, *Kepler Orbit* or *Perturbed Kepler Orbit* EoM

4.5.1.1 Accelerate Flight Path Velocity States

Description

Accelerate Flightpath consists of the relative position vector and the magnitude of the flight path velocity vector (i.e. flight path speed), in particular:

- R is the distance from the central's body center
- λ is the longitude measured from the prime meridian (e.g. Greenwich) in eastern direction
- δ is the declination, the planeto-centric latitude
- V is the magnitude (speed) of the flight path velocity measured in the local planetocentric coordinate system (see also Section 9.3.4.1)

The equations of motion are functions of these four state variables where the direction of the flight path velocity vector, the flight path heading/azimuth χ and the flight path inclination angle γ , is kept constant. This is useful to model a vehicle launched by a rail with fixed inclination, for example. The direction of the flight path velocity vector is defined through the attitude definition of the vehicle at the end of the previous phase. In case *Accelerate Flightpath* is used in the first phase, the direction of the flight path velocity vector is set by the vehicle attitude at the beginning of the phase.

Note: The flight path velocity direction is undefined for vertical flight and above the poles of the central body (north direction is undefined there). Therefore in such cases the *Accelerate Flightpath* dynamics should not be used.

Background

The Cartesian components of the velocity vector along the local L -frame are given by:

$$\hat{V}_k = V \begin{bmatrix} \sin\gamma \\ \cos\gamma\sin\chi \\ \cos\gamma\cos\chi \end{bmatrix}_L \quad (4.31)$$

The kinematic state equations represent the kinematic relationship established by the definition of the position and the velocity states:

$$\frac{d}{dt} \begin{bmatrix} R \\ \lambda \\ \delta \end{bmatrix} = \begin{bmatrix} V \sin \gamma \\ \frac{V \cos \gamma \sin \chi}{R \cos \delta} \\ \frac{V \cos \gamma \cos \chi}{R} \end{bmatrix} \quad (4.32)$$

The dynamic state equations follow Newton's second law

$$\frac{dV}{dt} = X + \Omega_E^2 R \cos \delta (\cos \delta \sin \gamma - \sin \delta \cos \gamma \cos \chi) \quad (4.33)$$

where Ω_E is the angular velocity of the central body about the inertial planet-centered z-axis. X , Y , Z are the components of the acceleration vector in the trajectory coordinate system (see Section 9.3.5.1):

$$\frac{\hat{F}}{m} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}_T \quad (4.34)$$

The acceleration acting on the vehicle results from gravity, aerodynamic forces, thrust or other perturbations. Note that only X is considered in the dynamic, whereas Y and Z are neglected (i.e. supposed to be balanced by the rail structure).

4.5.1.2 Circular Restricted Three Body Problem

Description

The Circular Restricted 3-Body Problem (CR3BP) is a particular case of the n-body problem. It describes the motion of a body with negligible mass (m_3) within the gravitational field generated by two massive bodies (called primaries, with $m_1 > m_2 \gg m_3$). Since one of these two masses is negligible if compared to the others, it is called restricted. Besides, the two primaries are moving on circular orbits around the center of mass. For this reason it is called circular. The CR3BP is the most used formulation of the 3-body problem, which allows to make use of special trajectories and orbits as manifold, Halo and Lissajous orbits, etc.

Configuration

The following setting is available:

- *The central body is* defines whether the central body specified in environment is used as:
 - *First primary*, while the *Second primary* needs to be specified
 - *Second primary*, while the first primary is determined automatically.

Background

The reference frame used to analyse the CR3BP is a co-rotating normalized reference frame. In this framework the equations of motion can assume a simpler expression. Moreover, the system is normalized by using the following units of mass, length and time [131]:

DYNAMICS CONFIGURATION

- Unit of mass = $m_1 + m_2$.
- Unit of length = mean distance between the primaries.
- Unit of time = $T/2\pi$ (orbital period of m_2 divided by 2π).

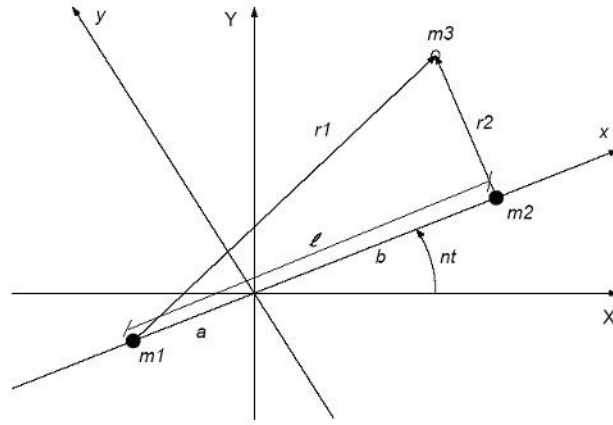


Fig. 4.7: Inertial (XYZ) and co-rotating (xyz) reference frames.

The following relations follow from the previous definitions:

- $a = \mu$
- $b = 1 - \mu$
- $l = n = 1$.

By using the previous definitions and the second Newton's law, the equations of motion to be solved can be written, in the co-rotating reference frame, in the following way [131]:

$$\begin{aligned}\ddot{x} - 2\dot{y} &= x - \frac{(1-\mu)(x+\mu)}{r_1^3} + \frac{\mu(1-x-\mu)}{r_2^3} = \frac{\partial U}{\partial x} \\ \ddot{y} + 2\dot{x} &= y - \frac{(1-\mu)y}{r_1^3} - \frac{\mu y}{r_2^3} = \frac{\partial U}{\partial y} \\ \ddot{z} &= -\frac{(1-\mu)z}{r_1^3} - \frac{\mu z}{r_2^3} = \frac{\partial U}{\partial z}\end{aligned}\tag{4.35}$$

where r_1 and r_2 are the distances of the third body from the primaries and U is a scalar function of the spatial coordinates, defined as the potential accounting both the gravitational forces and the centrifugal force and μ is the mass parameter equal to:

$$\mu = \frac{m_2}{m_1 + m_2}\tag{4.36}$$

4.5.1.3 Circular Restricted Three Body Problem Analytical

Description

These equations of motion allow to (analytically) propagate the initial state defined as analytical Halo or Lissajous orbit (see Section 4.2.3.2.10), which is determined setting the time to 0 in equations Eq. 4.6. No perturbations or maneuvers are therefore considered. For this reason these equations of motion can be used only in combination with the aforementioned initial state types. All the input parameters required to compute the states are taken over from the initial state definition, while the coefficients are calculated according to [133] and [137] for both Halo and Lissajous analytical orbits.

4.5.1.4 Docked

Description

With the *Docked* equations of motion a vehicle is considered joined with a reference vehicle. Since its movement will follow exactly the movement of the reference vehicle the translational dynamics specified for the reference vehicle also describe the translational movement of the docked vehicle. At least two vehicles need to be defined in the scenario to use it.

Configuration

The following settings have to be made:

- *Reference vehicle* is the vehicle (master) used for the docking. It can be chosen from a list of available vehicles in the scenario.
- *Reference element* is the element of the reference vehicle used for the docking
- *Reference node* is the corresponding node of the reference element used for the docking
- *Anchor element* is the element of the current vehicle used for the docking
- *Anchor node* is the node of the anchor element used for docking

4.5.1.5 Equinoctial Orbital Elements

Description

The modified *Equinoctial Elements* are a set of nonsingular orbital elements, in particular:

- *Equinoctial p* is the semiparameter p (semi-latus rectum) of the orbit, a measure for the shape of the orbit
- *Equinoctial f* is the cosine component of the equinoctial eccentricity vector
- *Equinoctial g* is the sine component of the equinoctial eccentricity vector
- *Equinoctial h* is the cosine component of the equinoctial inclination vector
- *Equinoctial k* is the sine component of the equinoctial inclination vector

DYNAMICS CONFIGURATION

- *Equinoctial* L is the true longitude of the spacecraft position and is equal to the sum of the true anomaly and the longitude of the periapsis. The longitude of the periapsis is equal to the sum of longitude of ascending node and argument of periapsis.

Since most of these elements change very slowly - if at all - along the trajectory, the trajectory integration is fast. In contrary to the Keplerian element dynamics, equinoctial elements avoid singularities for circular, equatorial, polar and hyperbolic orbits. Note that singularities still occur for retrograde orbits and rectilinear motion. More details about this set of equations of motion can be found in [50] and [51].

Background

The relationship between the modified equinoctial orbit elements and the Keplerian orbit elements is given by:

$$p = a \cdot (1 - e^2) \quad (4.37)$$

$$f = e \cos(\omega + \Omega) \quad (4.38)$$

$$g = e \sin(\omega + \Omega) \quad (4.39)$$

$$h = \tan\left(\frac{i}{2}\right) \cos\Omega \quad (4.40)$$

$$k = \tan\left(\frac{i}{2}\right) \sin\Omega \quad (4.41)$$

$$L = \Omega + \omega + \nu \quad (4.42)$$

And the dynamics of the modified equinoctial elements are:

$$\frac{dp}{dt} = \frac{2p}{w} \sqrt{\frac{p}{\mu}} \frac{F_T}{m} \quad (4.43)$$

$$\frac{df}{dt} = \sqrt{\frac{p}{\mu}} \left(\frac{F_N}{m} \sin L + ((w+1) \cos L + f) \cdot \frac{F_T}{m \cdot w} - (h \sin L - k \cos L) g \cdot \frac{F_O}{m \cdot w} \right) \quad (4.44)$$

$$\frac{dg}{dt} = \sqrt{\frac{p}{\mu}} \left(\left(-\frac{F_N}{m} \right) \cos L + ((w+1) \sin L + g) \cdot \frac{F_T}{m \cdot w} + (h \sin L - k \cos L) f \cdot \frac{F_O}{m \cdot w} \right) \quad (4.45)$$

$$\frac{dh}{dt} = \frac{1}{2} \cdot \sqrt{\frac{p}{\mu}} s^2 \cdot \frac{F_O}{m \cdot w} \cos L \quad (4.46)$$

$$\frac{dk}{dt} = \frac{1}{2} \cdot \sqrt{\frac{p}{\mu}} \cdot s^2 \cdot \frac{F_O}{m \cdot w} \sin L \quad (4.47)$$

$$\frac{dL}{dt} = \sqrt{p\mu} \left(\frac{w}{p} \right)^2 + \sqrt{\frac{p}{\mu}} \cdot \frac{1}{w} (h \sin L - k \cos L) \cdot \frac{F_O}{m} \quad (4.48)$$

where m , F_O , F_N and F_T are defined as in Section 4.5.1.12. The following abbreviations are used:

$$\frac{dL}{dt} = \sqrt{p\mu} \left(\frac{w}{p} \right)^2 + \sqrt{\frac{p}{\mu}} \cdot \frac{1}{w} (h \sin L - k \cos L) \cdot \frac{F_O}{m} \quad (4.49)$$

$$s^2 = 1 + h^2 + k^2 \quad (4.50)$$

4.5.1.6 Flight Path Velocity States

Description

Flightpath Velocity consists of the relative position vector and the flight path velocity vector [10], [22], in particular:

- R is the distance from the central's body center
- λ is the longitude measured from the prime meridian (e.g. Greenwich) in eastern direction
- δ is the declination, the planeto-centric latitude
- V is the magnitude (speed) of the flight path velocity measured in the local planetocentric coordinate system (see Section 9.3.4.1)
- χ is the heading angle, the angle between north and the flight path velocity component projected onto the horizontal plane of the local planetocentric coordinate system (see Section 9.3.4.1), clockwise positive. It is undetermined in vertical flight and above the poles of the central body.
- γ is the inclination angle, the angle between the horizontal plane of the local planetocentric coordinate system (see Section 9.3.4.1) and the flight path velocity (positive for ascent)

This set of equations of motion is useful for atmospheric flight (ascent or re-entry) as long as the trajectory is not close to the poles of the central body.

Background

The velocity states (V, γ, χ) are the polar representation of the flight path velocity vector $\hat{V}_k$, and the Cartesian components are given by

$$\hat{V}_k = V \begin{bmatrix} \sin\gamma \\ \cos\gamma \sin\chi \\ \cos\gamma \cos\chi \end{bmatrix}_L \quad (4.51)$$

The kinematic state equations represent the kinematic relationship established by the definition of the position and the velocity states

$$\frac{d}{dt} \begin{bmatrix} R \\ \lambda \\ \delta \end{bmatrix} = \begin{bmatrix} V \sin\gamma \\ \frac{V \cos\gamma \sin\chi}{R \cos\delta} \\ \frac{V \cos\gamma \cos\chi}{R} \end{bmatrix} \quad (4.52)$$

and the dynamic state equations are

$$\frac{dV}{dt} = X + \Omega_E^2 R \cos\delta (\cos\delta \sin\gamma - \sin\delta \cos\gamma \cos\chi) \quad (4.53)$$

$$\begin{aligned} \frac{d\gamma}{dt} = & -\frac{1}{V} Z + \frac{V}{R} \cos\gamma \\ & + 2\Omega_E \cos\delta \sin\chi \\ & + \frac{\Omega_E^2 R \cos\delta}{V} (\cos\delta \cos\gamma + \sin\delta \sin\gamma \cos\chi) \end{aligned} \quad (4.54)$$

$$\begin{aligned} \frac{d\chi}{dt} = & \frac{Y}{V\cos\gamma} + \frac{V\cos\gamma}{R}\tan\delta\sin\chi \\ & + 2\Omega_E(\sin\delta - \cos\delta\tan\gamma\cos\chi) \\ & + \frac{\Omega_E^2 R\cos\delta\sin\delta\sin\chi}{V\cos\gamma} \end{aligned} \quad (4.55)$$

where Ω_E is the angular velocity of the central body about its rotation axis. X , Y , Z are the components of the acceleration vector in the trajectory coordinate system (see Section 9.3.5.1):

$$\frac{\hat{F}}{m} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}_T \quad (4.56)$$

The acceleration acting on the vehicle results from gravity, aerodynamic forces, thrust or other perturbations.

4.5.1.7 Inertial Cartesian Components

Description

Inertial Cartesian consists of the inertial position and velocity vector in Cartesian representation [14], [23]:

- X is the x-component of the inertial position vector
- Y is the y-component of the inertial position vector
- Z is the z-component of the inertial position vector
- VX is the x-component of the inertial velocity vector
- VY is the y-component of the inertial velocity vector
- VZ is the z-component of the inertial velocity vector

Position and velocity are represented in an inertial coordinate system, for example J2000 (see Section 9.3.2.1). This set is especially useful where relative relative dynamics have singularities such as close to the poles of the central body. For orbital trajectories it is not the primary choice because the step size during integration needs to be small for a good precision. On the other hand, they are perfectly working for hyperbolic orbits with very high eccentricities.

Background

The states V_X , V_Y and V_Z specify the Cartesian components of the inertial velocity vector

$$\hat{V} \equiv \begin{bmatrix} V_X \\ V_Y \\ V_Z \end{bmatrix}_G \quad (4.57)$$

The kinematic state equations represent the kinematic relationship established by the definition of the position and the velocity states

$$\frac{d}{dt} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} V_X \\ V_Y \\ V_Z \end{bmatrix} \quad (4.58)$$

and the dynamic state equations are

$$\frac{d}{dt} \begin{bmatrix} V_X \\ V_Y \\ V_Z \end{bmatrix}_G = \frac{\hat{F}}{m} \quad (4.59)$$

Input to the system of Eq. 4.59 is the acceleration vector acting on the vehicle resulting from gravity, aerodynamic forces, thrust or other perturbations.

4.5.1.8 Inertial Velocity Components

Description

Inertial velocity consists of the relative position vector in polar coordinates and the inertial velocity vector in Cartesian coordinates [14], [23], in particular:

- R is the distance from the central's body center
- λ is the longitude measured from the prime meridian (e.g. Greenwich) in eastern direction
- δ is the declination, the planeto-centric latitude
- V_R is the radial component of the inertial velocity vector, positive upward
- V_λ is the component along the local parallel of the inertial velocity vector, positive in east direction
- V_δ is the component along the local meridian of the inertial velocity vector, positive in north direction

The inertial velocity vector is measured in inertial space and represented in the local planetocentric coordinate system L (see Section 9.3.4.1). This set of equations of motion is useful for atmospheric and orbital flight as long as the trajectory is not close to the poles of the central body.

Background

The states V_R , V_λ and V_δ specify the Cartesian components of the inertial velocity vector

$$\hat{V} \equiv \begin{bmatrix} V_R \\ V_\lambda \\ V_\delta \end{bmatrix}_L \quad (4.60)$$

The kinematic state equations represent the kinematic relationship established by the definition of the position and the velocity states

$$\frac{d}{dt} \begin{bmatrix} R \\ \lambda \\ \delta \end{bmatrix} = \begin{bmatrix} V_R \\ \frac{V_\lambda}{R \cos \delta} - \Omega_E \\ \frac{V_\delta}{R} \end{bmatrix} \quad (4.61)$$

and the dynamic state equations are

$$\frac{d}{dt} \begin{bmatrix} V_R \\ V_\lambda \\ V_\delta \end{bmatrix} = \begin{bmatrix} \frac{1}{R} \cdot (V_\lambda^2 + V_\delta^2) + \frac{F_R}{m} \\ \frac{1}{R} \cdot V_\lambda \cdot (V_\delta \cdot \tan \delta - V_R) + \frac{F_\lambda}{m} \\ -\frac{1}{R} \cdot (V_\lambda^2 \cdot \tan \delta + V_\delta \cdot V_R) + \frac{F_\delta}{m} \end{bmatrix} \quad (4.62)$$

Input to the system of Eq. 4.62 is the acceleration vector acting on the vehicle resulting from gravity, aerodynamic forces, thrust or other perturbations.

4.5.1.9 Keplerian Orbital Elements

Description

Kepler Orbit is an unperturbed orbit with the following set of orbital elements which describe the vehicle position and velocity:

- a is the semi-major axis of the orbit
- e is the eccentricity of the orbit
- i is the inclination of the orbit
- ω is the argument of the periapsis of the orbit
- Ω is the right ascension of the ascending node of the orbit
- v is the true anomaly of the orbit

This set of equations of motion is only useful to describe a ballistic arc since perturbations are neglected here. Perturbations are considered in the set *Perturbed Kepler Orbit*, see Section 4.5.1.12.

Note: The *Kepler Orbit* equations of motion do not consider any perturbations, i.e. the aerodynamic drag, propulsion forces, third body perturbations, etc. have no effect on the vehicle motion.

Note: The *Kepler Orbit* equations cannot be used with parabolic orbits (eccentricity equal to 1.0), because the semi-major axis is undefined.

Note: For an orbital scenario it is recommended to use the *Equinoctial Elements* (see Section 4.5.1.12 for details).

Background

DYNAMICS CONFIGURATION

Only the true anomaly changes with time (see Eq. 4.63)

$$\frac{dv}{dt} = \sqrt{\frac{\mu}{p^3}} (1 + e \cdot \cos v)^2 \quad (4.63)$$

where p is the semi-parameter of the orbit (also called "semilatus rectum"),

$$p = a \cdot (1 - e^2) \quad (4.64)$$

and μ denotes the standard gravitational parameter of the central body. All other elements remain constant.

4.5.1.10 Launch Pad

Description

This set of equation of motion are required to model the ignition phase of a launcher: the thrust is not yet higher than the weight and the vehicle is fixed at the launch pad.

This package provides equations of motions of a rigid body with respect to a rotating planet using flight path velocity states [10], [22]. In this special case the velocity is fixed to zero. In state space form, the equations of motion are functions of three state variables, which specify the vehicle's position. The three auxiliary states define its velocity don't appear in the state space as they are fixed.

The position of the vehicle is specified by a set of planet-fixed spherical coordinates (R , λ , δ). As shown in Fig. 9.5, R denotes the distance from the planet center, λ the east longitude measured from the Greenwich meridian, and δ the declination measured from the equatorial plane. The derivatives are always zero. This state representation is indistinct above the poles of the planet, where the definition of north is ambiguous, and for vertical flight. In both cases, the value of the flight path heading angle χ is undetermined.

Table 4.2: Definition of flight path velocity state variables

State Variable	Definition
R	Radial distance from planet center
λ	East longitude / Angle between the Greenwich meridian and the meridian of the current position (positive east of Greenwich)
δ	Declination angle between the equatorial plane and the current position vector (positive on the northern hemisphere)

The kinematic state equations represent the kinematic relationship established by the definition of the position:

$$\frac{d}{dt} \begin{bmatrix} R \\ \lambda \\ \delta \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \quad (4.65)$$

The sum of all external acceleration acting on the vehicle is zero, as the vehicle is assumed to be fixed to a launch pad. In a sequence of phases with different EOM's, this type of EOM

must be the first one, if it exists. Directly following phases can have the this EOM with zero derivatives, but the first phase with optimizable velocity must not be a relative flight path velocity state, as it is not defined for a zero velocity.

As the velocity is zero, there is no need for initial constraints. The direction of the velocity with respect to the reference frame L (γ , χ) is not defined: the flight path inclination γ is set to zero, and the heading angle χ is set to 90° with respect to north.

4.5.1.11 (Flexible) Multibody Dynamics

(Flexible) Multibody Dynamics is the set of equations of motion for multibody systems.

Each single element of the vehicle with positive mass has its own set of translational and rotational equations of motion. The position and the velocity vector are specified by their Cartesian elements, while the orientation is specified as quaternion and the rotation is specified by the body-axis rates.

For a description of the configuration, please refer to chapter (Flexible) Multibody Dynamics in the User Manual.

For a description of the model background, please refer to chapter Multibody Dynamics in the Multibody and Flexible Dynamics Models book.

4.5.1.12 Perturbed Keplerian Orbital Elements

Description

Perturbed Kepler Orbit is a set of orbital elements describing the position and velocity state ([48], [49] and [50]), in particular:

- a is the semi-major axis of the orbit
- e is the eccentricity of the orbit
- i is the inclination of the orbit
- ω is the argument of the periapsis of the orbit
- Ω is the right ascension of the ascending node of the orbit
- v is the true anomaly of the orbit

This set of equations of motion is useful for orbital trajectories. Unlike *Kepler Orbit* (see Section 4.5.1.9), perturbations are considered.

Note: These equations of motion can be neither used for circular orbits (zero eccentricity), because the derivatives of eccentricity, true anomaly and argument of periapsis are ill-defined, nor for equatorial orbits (zero inclination), because the derivative of the right ascension of the ascending node is ill-defined, nor for parabolic orbits (eccentricity equal to 1.0), because the semi-major axis is undefined.

Background

The dynamic state equations are

$$\frac{da}{dt} = \frac{2 \cdot e \cdot \sin v}{n \cdot x} \cdot \frac{F_N}{m} + \frac{2 \cdot a \cdot x}{n \cdot r} \cdot \frac{F_T}{m} \quad (4.66)$$

$$\frac{de}{dt} = \frac{x \cdot \sin v}{n \cdot a} \cdot \frac{F_N}{m} + \frac{x}{n \cdot a^2 \cdot e} \cdot \left(\frac{a^2 \cdot x^2}{r} - r \right) \cdot \frac{F_T}{m} \quad (4.67)$$

$$\frac{di}{dt} = \frac{r \cdot \cos u}{n \cdot a^2 \cdot x} \cdot \frac{F_0}{m} \quad (4.68)$$

$$\frac{d\omega}{dt} = -\frac{x \cdot \cos v}{n \cdot a \cdot e} \cdot \frac{F_N}{m} + \frac{p}{e \cdot h} \cdot \sin v \cdot \left(1 + \frac{r}{p} \right) \cdot \frac{F_T}{m} - \frac{r \cdot \cot i \cdot \sin u}{n \cdot a^2 \cdot x} \cdot \frac{F_0}{m} \quad (4.69)$$

$$\frac{d\Omega}{dt} = \frac{r \cdot \sin u}{n \cdot a^2 \cdot x \cdot \sin i} \cdot \frac{F_0}{m} \quad (4.70)$$

$$\frac{dv}{dt} = \sqrt{\frac{\mu}{p^3}} (1 + e \cos v)^2 - \left[\frac{p}{e \cdot h} \cdot \sin v \cdot \left(1 + \frac{r}{p} \right) \cdot \frac{F_T}{m} - \frac{x \cdot \cos v}{n \cdot a \cdot e} \cdot \frac{F_N}{m} \right] \quad (4.71)$$

with

$$x = \sqrt{1 - e^2}, \quad h = \sqrt{\mu \cdot p}, \quad u = v + \omega, \quad n = \sqrt{\frac{\mu}{a^3}}, \quad r = \frac{p}{(1 + e \cdot \cos v)}. \quad (4.72)$$

where p is the semi-parameter of the orbit (also called "semilatus rectum")

$$p = a \cdot (1 - e^2) \quad (4.73)$$

where m is the vehicle mass, F_N , F_T and F_O are the force components in the local orbital coordinate system LO (see Section 9.3.1.1) and μ is the standard gravitational parameter of the central body.

4.5.1.13 SGP4

Description

SGP4 (Simplified General Perturbations Satellite Orbit Model 4) is a NASA/NORAD algorithm that calculates near Earth satellites.

4.5.1.14 State from File

Description

State from File is actually not a set of equations of motion. Instead, the position and velocity states are provided through an external file. The date/time of the provided file has to coincide with the actual phase time (depends on the specified mission start date and the duration of all preceding phases).

Configuration

The following settings have to be made:

- *File type* is the kind of the file format:
 - CCSDS is a text file according to the CCSDS standard for orbit ephemeris message data (OEM)
 - *Simulation Structure File (*.struct)* is the struct file format, a binary output format of ASTOS
- *Filename* is the name and path of the file containing the position and velocity

4.5.2 Rotational Dynamics

Description

Rotational motion is the movement of a rigid body keeping at least one point fixed (unlike translation). Modeling the rotational motion of a rigid body is required for 6-DoF dynamics [30]. The equations of motion are functions of two state sets describing the orientation and rotation of the vehicle:

- Orientation defines the vehicle's attitude. In other words, it is the orientation of the spacecraft body-axis with respect to another frame, for example an inertial reference frame. Different representations are supported:
 - Aerodynamic angles (see Section 4.5.2.1)
 - Euler angles (see Section 4.5.2.2)
 - Quaternion (see Section 4.5.2.3)
- Rotation describes the change of the orientation of the vehicle's body axes and is the angular velocity of the vehicle. Only body-axis rates are supported and their explanation is provided below since they are common for all orientation representations.

Definition of Rotation Vector

DYNAMICS CONFIGURATION

Three state variables specify the vehicle's angular velocity with respect to the inertial frame. It is also known as the rotation vector ω of the body frame with respect to the inertial frame expressed in components along the body axes:

- *OMEGA_X* is the body-axis rate ω_x , the rate (angular velocity) of the body x-axis (vehicle's forward axis) of the vehicle w.r.t. inertial frame. It is also known as p .
- *OMEGA_Y* is the body-axis rate ω_y , the rate (angular velocity) of the body y-axis (vehicle's starboard/lateral axis) of the vehicle w.r.t. inertial frame. It is also known as q .
- *OMEGA_Z* is the body-axis rate ω_z , the rate (angular velocity) of the body z-axis (vehicle's vertical axis) of the vehicle w.r.t. inertial frame. It is also known as r .

Dynamics of Rotation Vector

The dynamics of the body-axis rates are given by the Euler equations:

$$\frac{d}{dt}\hat{\omega} = I^{-1}(\hat{M} - \hat{\omega} \times I\hat{\omega}) \quad (4.74)$$

where the inertial body-fixed angular velocity vector is given by

$$\hat{\omega} = \begin{bmatrix} P \\ Q \\ R \end{bmatrix}_B \quad (4.75)$$

and the inertia matrix I is defined with respect to the body frame with the following components:

$$I = \begin{bmatrix} I_{XX} & I_{XY} & I_{XZ} \\ I_{XY} & I_{YY} & I_{YZ} \\ I_{XZ} & I_{YZ} & I_{ZZ} \end{bmatrix} \quad (4.76)$$

The definition of the products of inertia I_{XY} , I_{XZ} and I_{YZ} in *ASTOS* is as follows: $I_{XY} = - \int xy \, dm$, $I_{XZ} = - \int xz \, dm$ and $I_{YZ} = - \int yz \, dm$.

Note: There exist at least two different definitions of the products of inertia I_{XY} , I_{XZ} and I_{YZ} . The one used above is in line with the notation in Eq. 4.76 where all elements of the inertia matrix I on the minor diagonals lack the “-” sign.

Further input to the Euler equations are the external moments $\hat{M}$ acting on the vehicle resulting from aerodynamic and other external torques. They are expected in the body-fixed axis system.

4.5.2.1 Aerodynamic Angles

Description

Aerodynamic angles define the orientation of the vehicle's body frame with respect to the air path frame and consist of the following states:

- *ALPHA* is the angle of attack α
- *BETA* is the side slip angle β

DYNAMICS CONFIGURATION

- MU is the air path bank angle μ_a (the subscript 'a' is omitted in the following equations)

Equations of Motion

Aerodynamic angles have the following dynamic equations:

$$\begin{aligned} \dot{\alpha} = & -P \cos \alpha \tan \beta + Q - R \sin \alpha \tan \beta + \frac{\cos \mu}{\cos \beta} \left[\dot{\delta} \cos \chi - \dot{\gamma} + (\dot{\lambda} + \Omega_E) \cos \delta \sin \chi \right] \\ & + \frac{\sin \mu}{\cos \beta} \left[\dot{\delta} \sin \chi \cos \gamma - \dot{\chi} \cos \gamma + (\dot{\lambda} + \Omega_E) (\sin \delta \cos \gamma - \cos \delta \cos \chi \sin \gamma) \right] \end{aligned} \quad (4.77)$$

$$\begin{aligned} \dot{\beta} = & P \sin \alpha + R \cos \alpha + \sin \mu \left[\dot{\delta} \cos \chi - \dot{\gamma} + (\dot{\lambda} + \Omega_E) \cos \delta \sin \chi \right] \\ & + \cos \mu \left[\dot{\chi} \cos \gamma - \dot{\delta} \sin \chi \sin \gamma + (\dot{\lambda} + \Omega_E) (\cos \delta \cos \chi \sin \gamma - \sin \delta \cos \gamma) \right] \end{aligned} \quad (4.78)$$

$$\begin{aligned} \dot{\mu} = & P \cos \alpha \cos \beta + Q \sin \beta + R \sin \alpha \cos \beta - \dot{\alpha} \sin \beta + \dot{\chi} \sin \gamma + \dot{\delta} \sin \chi \cos \gamma \\ & - (\dot{\lambda} + \Omega_E) (\cos \delta \cos \chi \cos \gamma + \sin \delta \sin \gamma) \end{aligned} \quad (4.79)$$

where the dot indicates differentiation with respect to time and Ω_E is the angular velocity of the planet about the inertial planet-centered z-axis. Input to these equations are also the derivatives of the translational flight path velocity state (see Section 4.5.1.6).

4.5.2.2 Euler Angles

Description

A set of Euler angles defines the orientation of the vehicle's body frame with respect to another frame and consists of the following states:

- *YAW* is the yaw angle ψ
- *PITCH* is the pitch angle θ
- *ROLL* is the roll angle ϕ

Depending on the reference frame, the Euler angles are differently defined.

Equations of Motion

For the inertial frame, the dynamic equations for the Euler angles are given by

$$\dot{\phi} = p + (\sin \phi \tan \theta) q + (\cos \phi \tan \theta) r \quad (4.80)$$

$$\dot{\theta} = (\cos \phi) q - (\sin \phi) r \quad (4.81)$$

$$\dot{\psi} = \frac{\sin \phi}{\cos \theta} q - \frac{\cos \phi}{\cos \theta} r \quad (4.82)$$

where the dot indicates differentiation with respect to time.

4.5.2.3 Quaternion

Definition

A quaternion defines the orientation of the vehicle's body frame with respect to another frame. It is specified by the following four states:

- QW is the scalar (real) part of the quaternion
- QX is the first component of the vector part (imaginary) of the quaternion
- QY is the second component of the vector part (imaginary) of the quaternion
- QZ is the third component of the vector part (imaginary) of the quaternion

Equations of Motion

The dynamic equations for the quaternion are given by

$$\dot{Q}_1 = \frac{1}{2} (-Q_4 p - Q_3 q + Q_2 r) \quad (4.83)$$

$$\dot{Q}_2 = \frac{1}{2} (Q_3 p - Q_4 q - Q_1 r) \quad (4.84)$$

$$\dot{Q}_3 = \frac{1}{2} (-Q_2 p + Q_1 q - Q_4 r) \quad (4.85)$$

$$\dot{Q}_4 = \frac{1}{2} (Q_1 p + Q_2 q + Q_3 r) \quad (4.86)$$

where the dot indicates differentiation with respect to time.

4.5.3 Auxiliary States

Description

Auxiliary states are additional differential equations besides the translational and rotational dynamics. They are optional and are activated in the panel default settings of the dynamics configuration of each vehicle (see Auxiliary States paragraph in User Manual). Several auxiliary states are available:

- *Horizontal range* is the integrated length of the ground track, see Section 4.5.3.1
- *Velocity increment (delta-V)* is the integrated velocity increment without losses, see Section 4.5.3.2
- *Drag loss* is the loss in delta-V due to aerodynamic forces, see Section 4.5.3.8
- *Steering loss* is the loss in delta-V due to thrust vector misalignments, see Section 4.5.3.4
- *Heat load* is the integrated heat load on the vehicle, see Section 4.5.3.5
- *Average drag deviation* calculates an average drag profile, see Section 4.5.3.6
- *Trajectory smoothness* is a measure to identify e.g. oscillating behavior of a trajectory, see Section 4.5.3.7
- *Total impulse* is the integrated thrust/force of all actuators, see Section 4.5.3.9

- *Gravity loss* is the loss in delta-V due to gravitational forces, see Section 4.5.3.3
- *Ozone load* is the integrated ozone concentration along the trajectory, see Section 4.5.3.10

Background

Enabled auxiliary states are modeled as an optimizable state quantity and always propagated during a simulation. They are present in all phases of the vehicle. Usually they are only required for optimization of a certain physical quantity like a cost function or constraint, but they can be also used for information only.

4.5.3.1 Horizontal Range

Description

Horizontal range integrates the distance along the trajectory.

Background

The dynamic state equation is

$$\frac{ds}{dt} = \frac{v \cdot \cos(\gamma) \cdot r}{R_e} \quad (4.87)$$

where v is the flight path velocity of the vehicle, γ is the flight path inclination angle, r is the radius of the vehicle w.r.t. center of the central body and R_e is the radius of the central body.

4.5.3.2 Velocity Increment (Delta V)

Description

Velocity increment (delta V) is the integrated velocity increment (delta-v) without losses. It is required for the cost function *Max DeltaV* (see Section 7.5).

Background

The velocity increment is defined by

$$\Delta V = \int_{t_0}^{t_f} (T / m) dt \quad (4.88)$$

with the thrust T and the vehicle mass m .

4.5.3.3 Gravity Loss

Description

DYNAMICS CONFIGURATION

Gravity loss is the integrated velocity (delta-V) loss due to gravitational forces. It is required for the cost function *Min Loss* (see Section 7.14).

Background

The gravity loss L_G is given by

$$L_G = \int_{t_0}^{t_f} \min(g \cdot \sin(\gamma); acc_{T_r}) dt \quad (4.89)$$

where g is the local gravity acceleration, γ is the flight path inclination angle and acc_{T_r} is the radial component of the thrust acceleration. The integral considers only the minimum between the two elements to avoid the overestimation of the losses in case of an orbiting satellite with low thrust over weight ratio.

4.5.3.4 Steering Loss

Description

Steering Loss is the integrated velocity (delta-V) loss due to thrust not aligned with the velocity vector. It is required for the cost function *Min Loss* (see Section 7.14).

Background

The steering loss L_T is calculated according to

$$L_T = \int_{t_0}^{t_f} \frac{T}{m} (1 - \cos \alpha_t) dt \quad (4.90)$$

where T is the thrust, m the vehicle mass and α_t the total angle of attack.

4.5.3.5 Heat Load

Description

Heat load is the integrated heat flux at the stagnation point and a measure for re-entry trajectories. It is required for the cost function *Min Heatload* (see Section 7.13).

Configuration

The following configuration has to be specified:

- *Aerothermodynamics* is the model computing the heat flux in the stagnation point. If *custom*, the corresponding identifier of the aerothermodynamics model has to be chosen, otherwise the *default* model is used.

Background

The dynamic state equation equals the heat flux provided by the aerothermodynamics model (see Section 3.5.2).

4.5.3.6 Average Drag Deviation

Description

Average drag deviation calculates an average drag profile. It is required for the cost function *Max Safety* (see Section 7.11).

Configuration

The following configuration is required:

- *Method* defines the kind of the average drag computation:
 - *Profile* defines the complete average drag as function of the air path velocity by user input. It has to be specified as profile data table, which is described in detail in Profile Data Tables in User Manual.
 - *Average* computes the average drag and has the following specifications:
 - *Heat flux limit* is the scaling factor Q_{max} of the heat flux acceleration
 - *Aerothermodynamics* is optional and specifies the identifier of the aerothermodynamics model required for the heat flux computation, if *enabled*. By *default* the aerothermodynamics model of the vehicle is used.
 - *Load factor limit* is optional and defines the scaling factor n_{max} of the load factor acceleration
 - *Dynamic pressure limit* is optional and defines the scaling factor q_{max} of the dynamic pressure acceleration
 - *Hinge moment limit* is optional and is required for the computation of the hinge moment acceleration
 - *Maximum Hinge Moment* is the scaling factor H_{max}
 - *CHmax* is the maximum hinge moment coefficient
 - *Lref* is the flap reference length
 - *Sref* is the flap reference area
 - *Equilibrium glide* is optional. If no specification is made, a roll angle of 0° is used to calculate a_{min}

Note: The equilibrium glide condition is also calculated with a bank angle of zero if the total acceleration is below a specified value *Min_Acceleration*. This acceleration limit can be used for entry cases, where the Mach number is not unique in the beginning.

DYNAMICS CONFIGURATION

- *Maximum velocity* is optional and defines the limit over which the derivative is always zero. This option can be useful for velocities above this value, where the corresponding cost function (see Section 7.11) cannot follow the average profile.

Note: This feature may be inappropriate for optimization as it results in a non-differentiable function. It should not be used if the gradient is too large or in case of collocation.

All additional parameters for the computation of the hinge moment and equilibrium glide are described in detail in Section 6.4 and Section 6.3, respectively.

Note: At least one limit of dynamic pressure, load factor or hinge moment need to be specified.

Background

This auxiliary state calculates an average drag acceleration profile using the formula

$$d = \int_{t_0}^{t_f} \frac{V_{\text{rel}}}{R_e} \left(\frac{a_D - a_{\text{ref}}}{\Delta a} \right) dt \quad (4.91)$$

with

$$a_D = \frac{D}{m}, a_{\text{ref}} = \frac{a_{\text{max}} + a_{\text{min}}}{2}, \Delta a = \frac{a_{\text{max}} - a_{\text{min}}}{2} \quad (4.92)$$

The minimum and maximum accelerations are defined by

$$a_{\text{min}} = -a_D \frac{V_r \dot{\gamma} + a_z + a_y \sin \mu_{\text{min}}}{a_L \cos \mu_{\text{min}}} \quad (4.93)$$

and

$$a_{\text{max}} = \min \left\{ a_D \left(\frac{\dot{Q}_{\text{max}}}{Q} \right)^{1/n}, a_D \frac{q_{\text{max}}}{q}, a_D \frac{n_{\text{max}}}{n}, a_D \frac{H_{\text{max}}}{H} \right\} \quad (4.94)$$

where a_z is the acceleration along the z axis of the trajectory coordinate system T (see Section 9.3.5.1) excluding gravity, a_y is the aerodynamic side force acceleration and a_L is the aerodynamic lift acceleration.

4.5.3.7 Trajectory Smoothness

Description

Trajectory smoothness is the integrated square of the derivative of the flight path inclination angle. This can be used to dampen trajectories which show oscillating behaviour. It is required for the cost function *Trajectory smoothing weight factor* (see Chapter 7).

Background

The dynamic state equation is

$$\frac{dS}{dt} = \dot{\gamma}^2 \quad (4.95)$$

with the flight path inclination angle γ .

4.5.3.8 Drag Loss

Description

Drag loss describes the integrated velocity (delta-V) loss due to aerodynamic drag. It is required for the *Min Loss* cost function (see Section 7.14).

Background

The drag loss L_D is calculated according to

$$L_D = \int_{t_0}^{t_f} a_D dt \quad (4.96)$$

where a_D is the drag acceleration.

4.5.3.9 Total Impulse

Description

Total impulse is the integrated thrust/force of all actuators of a vehicle.

Background

The total impulse J is defined by

$$J = \int_{t_0}^{t_f} T dt \quad (4.97)$$

where T is the resultant thrust/force applied on the vehicle.

4.5.3.10 Ozone Load

Description

Ozone load integrates the current concentration of ozone in the atmosphere. In order to not disturb the ozone layer, an advanced launcher vehicle should cross it fast.

Background

The dynamic state equation of the ozone load O equals the ozone density

$$\frac{dO}{dt} = \rho_{OZ} \quad (4.98)$$

4.6 Maneuvers at the Beginning of a Phase

Description

Maneuvers at the beginning of a phase change instantaneously one or multiple states of the vehicle, such as velocity magnitude and/or direction. Optionally, the related propellant consumption required for such a maneuver can be computed. The following kinds of maneuver are available:

- *Impulse* to change the velocity vector of the vehicle, see Section 4.6.1
- *Spin impulse* to change the rotation rate of the vehicle, see Section 4.6.2

Note: *Maneuver at beginning of phase* is not available for the first phase!

Configuration

All maneuvers have the following setting in common:

- *Propellant consumption* has to be *enabled* (default is *disabled*) to calculate the required propellant mass of the maneuver:
 - *Isp* is the specific impulse of the (imaginary) propulsion system used for the maneuver
 - *Tank ID* is the identifier of the related tank component where the propellant mass of the maneuver is taken from. This tank component must be already defined by the user (see Section 3.3).

Background

Some general remarks can be stated for maneuvers at the beginning of a phase:

- Any maneuver takes place instantaneously at the beginning of a phase
- The vehicle velocity required to perform the desired maneuver and reach the required end conditions is computed without considering the effect of perturbations. Therefore if perturbations are active in the phase following the maneuver, the reached end condition may deviate from the requested end condition.
- If, for example, a drift, orbit transfer or spin up phase is started, it usually needs to be finalized by a second maneuver at the beginning of the following phase.

- Many maneuvers are suitable if they take place at a specific orbit locations. Hence they should be combined with *Additional phase end conditions* in *Dynamics Configuration* -> *Phases* -> *Phase* (see chapter Additional Phase End Conditions in the User Manual book).

4.6.1 Impulse Maneuvers

Description

Impulsive maneuvers change "instantly" the velocity vector of the vehicle. Typical applications are orbit maneuvers like Hohmann transfers or rendezvous scenarios. But they are also suitable for interplanetary transfers.

Configuration

An *Impulse* maneuver has to be *enabled* before further settings become available:

- *Type* allows to choose the kind of impulsive maneuver:
 - *User Defined* defines the change of the inertial velocity vector's components *X*, *Y* and *Z* in the selected *Representation frame*:
 - *B* is the body-fixed axis system as defined at the end of the previous phase, see Section 9.3.3.1

Note: The B-frame corresponds to the attitude definition at the end of the previous phase, and not at the beginning of the current one!

 - *L* is the local planetocentric axis system, see Section 9.3.4.1
 - *LO* is the local orbit axis frame, see Section 9.3.1.1
 - *T* is the trajectory axis system, see Section 9.3.5.1
 - *TO* is the tangential orbital axis system, see Section 9.3.1.3
 - *Radial* defines a maneuver in radial direction which leads to a drift along the satellite V-bar direction (see Section 9.3.6.3) by a specified *Distance*.
 - *Radial to Target* defines a radial maneuver which leads to a drift that ends at a specified *Distance* to a target vehicle specified by *Target ID*.
 - *Tangential* defines a tangential maneuver which leads to a drift along the satellite V-bar direction (see Section 9.3.6.3) by a specified *Distance*
 - *Tangential to Target* defines a tangential maneuver which leads to a drift that ends at a specified *Distance* to a target vehicle specified by *Target ID*.
 - *Fly Around Target* defines a radial maneuver which results in a fly-around of the target vehicle specified by *Target ID*.
 - *Circularize* defines a maneuver that circularizes the orbit (eccentricity zero)
 - *Hold Point Target* defines a maneuver that maintains the current distance to a target vehicle specified by *Target ID*.

DYNAMICS CONFIGURATION

- *Target Radius at next Apsis Crossing* defines classical Hohmann transfer maneuver with a velocity increment required to reach the specified *Target Radius* at the next apsis (either apoapsis or periapsis)
- *Out of Plane* defines a maneuver in normal orbit direction (direction of angular momentum) changing the inclination of the orbital plane to the specified *Target Inclination*
- *Transfer Plane Injection* is one of the maneuvers for interplanetary transfers. It changes the inclination of the orbital plane for optimal interplanetary injection.
- *Interplanetary Injection* is one of the maneuvers for interplanetary transfers and changes the orbit energy to escape the current central body and to reach the next celestial body.
- *Deep Space* is one of the maneuvers for interplanetary transfers and changes the orbital velocity to reach the sphere of influence of the next celestial body.

Most type-dependent inputs can be made *Optimizable* by left-click on the respective checkbox. Then *Lower/upper bound* values can be specified and a corresponding real parameter is created in the next initialization.

Note: All maneuvers with input *Target ID* require at least two vehicles in the scenario!

The interplanetary transfer maneuvers (*Transfer Plane Injection*, *Interplanetary Injection* and *Deep Space*) solve Lambert's problem to evaluate the required velocity changes. A transfer starts at the initial celestial body and ends at the final celestial body. Both bodies orbit their common center body. A transfer leg is around the central body from the initial celestial body to the final one and requires the following specifications:

- *Number of complete revolutions* defines the number of orbits around the center body of the transfer leg. Zero means, the transfer leg is less than one revolution.
- *Orbit direction* is the orbit direction of the transfer leg. Options are *Prograde* and *Retrograde*. If retrograde, the angular momentum vector of the transfer leg points opposite to the angular momentum vector of the initial celestial body.
- *Transfer type* is the kind of the transfer leg, either *Short arc* or *Long arc*. Both options represent two solutions of Lambert's problem. In general, the short arc is a kind of "direct transfer" within short time, while the long arc is typically outbound and requires a much longer transfer duration.

Note: All maneuvers of the same transfer leg should have the identical parameter settings to work properly.

Note: The initial and final celestial body as well as the central body of these interplanetary maneuvers are automatically determined from the scenario phases' dynamics configurations.

Background

For the interplanetary transfer maneuvers the Lambert's problem is solved. It describes an orbit between two position vectors and a given time of flight. Here, the first position vector is the position of the initial celestial body at the epoch when the transfer starts. Adding the transfer duration to the initial epoch yields the final epoch when the transfer ends. This epoch results in the second position vector, the position vector of the final celestial body. Both positions, the time of flight and the specified transfer parameters are input of Lambert's problem (a boundary value problem of a two-body problem). The transfer leg is a conic trajectory between the position vectors. Both positions originate at the common central body of the initial celestial body, the final celestial body and the conic arc. For an interplanetary transfer this is the Sun.

4.6.2 Spin Impulse Maneuvers

Description

Spin impulsive maneuvers change "instantly" the rotation rate of the vehicle.

Configuration

For a *Spin Impulse* maneuver the following options are available:

- *Type* defines the kind of the spin impulse maneuver:
 - *User Defined* defines the change of the angular velocity about X-axis, Y-axis and Z-axis of the *Frame B* (body-fixed axis system, see Section 9.3.3.1). Each component can be *Optimizable* by left-click on the respective checkbox. Then *Lower/upper bound* values can be specified for the optimization and a corresponding real parameter is created in the next initialization.
 - *Stabilize* reduces the angular velocity about an axis. By default, all axes are *disabled*. An axis has to be *enabled* so that the spin about it is reduced to zero. Multiple axes can be combined. Only body-fixed axes are supported (*Frame* is always *B*, the body-fixed axis system, see Section 9.3.3.1).

4.7 Linearised Flexible Dynamics

Flexible dynamics panel in the vehicle *Default Settings* settings (*Modelling*→*Dynamics Configuration*→*Vehicle & POIs Dynamics*) shall be enabled in the in order to that flexibility effect are considered.

For a description of the configuration, please refer to chapter *Linearised Flexible Dynamics* in the *User Manual*.

DYNAMICS CONFIGURATION

For a description of the model background, please refer to chapter Linearised Flexible Dynamics in the Multibody and Flexible Dynamics book.

5 Analyses

The **Analysis** feature in *ASTOS* allows the user to perform computations of further quantities of interest, which require additional computational time and the data of a completely simulated or optimized scenario as input. The **Analysis** feature uses as input the `struct` file created during the simulation of a scenario. The output created by the analyses can be appended to the same input file or to a different file, depending on the type of analysis and on the user settings. The following list gives an overview of the available analyses:

- *Chemical Plume Dispersion Analysis* computes the data required to generate 2D maps showing the concentration of the launcher vehicles exhaust plume
- *Coverage* analysis evaluates the visibility of a target based on a user-defined set of visibility, link or relay conditions, see
- *Data Timeliness* analysis calculates the time required to transmit all accumulated data in a vehicle to a desired receiver, see Section 5.2
- *Destructive Reentry* analysis computes the reentry trajectories and impact locations of space debris and performs a risk analysis based on this data, see Section 5.3
- *Earth Observation Mission* analysis allows the automatic generation of an Earth observation mission report according to the ESA mission analysis guidelines for EO missions [145] and automatically populates some parts of the report based on the chosen scenarios by the user, see Section 5.5
- *Eclipse* analysis computes the eclipse pattern of a vehicle including longest duration of an eclipse and shortest time between eclipses, see Section 5.6
- *GNC* analysis, see Section 5.7
- *Interplanetary Transfer* analysis processes the data of interplanetary transfers to be visualized in plots, see Section 5.8
- *Launch Safety* analysis evaluates the risk of casualties and fatalities of a launch trajectory in case of failures and computes the flight corridor, see Section 5.9
- *Link Budget* analysis evaluates the gains and losses of an electromagnetic signal between transmitter and receiver, see Section 5.10
- *Load Case* analysis identifies the dimensioning load cases of launcher substructures based on the launcher trajectory, see Section 5.11
- *Navigation* analysis calculates the accuracy of the position determination which can be achieved if a satellite was navigated only by means of Global Navigation Satellite System (GNSS) receivers, see Section 5.12
- *Operational Lifetime Prediction* analysis calculates the operational phase duration of a satellite considering the maneuvers to be performed, see Section 5.13

ANALYSES

- *Requirements* analysis allows the definition of boolean-logic-based requirements and evaluates these for user-specified time points, see Section 5.14
- *Rocket Motor* analysis calculates the performance and characteristics of a *Throat Design* rocket engine, see Section 5.15
- *Tracking* analysis, see Section 5.16
- *Visibility* analysis evaluates the observability of the vehicle from ground stations as well as the visibility of targets from the vehicle payload, see Section 5.17

Common analysis settings

Some of the available analyses require the following settings:

- The *Simulation output file* path specifies the location of the `struct` file where the output of this analysis should be saved. If this output simulation file already exists and has the proper format (same number of phases and nodes as the current `simulation.struct`), the new data is appended, otherwise an error is raised suggesting to delete the old output file. If the specified `struct` file does not exist, a new one with proper format is created.
- *Write analysis data only* allows the user to add only the data computed by this analysis to an empty or new output `struct` file, i.e. without adding the complete information contained in the input `struct`.
- *Phase* specifies the phase whose beginning shall be considered as the beginning for the analysis calculations.
- *Vehicle* is a drop-down vehicle selector to choose the vehicle for which the analysis shall be performed.
- *Evaluation time* denotes if the beginning or the end of the phase shall be considered as the start of the analysis calculations.

5.1 Chemical Plume Dispersion Analysis

Rocket propellants can contain and have poisonous or environmentally hazardous components and combustion products. In order to estimate the safety of the public and protected environments, an *ASTOS* user can perform a simulation of the distribution of the rocket engine gases before any real rocket flight or especially before planning new space ports. This can be done by using the *Chemical Plume Dispersion Analysis* which computes the data required to generate 2D maps showing the concentration of the launcher vehicles exhaust plume at a user defined altitude after a given time after launch. Details on this analysis configuration, model and output is given in the following three subsections:

- Section 5.1.1 provides a description how to configure the analysis
- Section 5.1.2 contains information on the model background
- In Section 5.1.3 the output generated by the analysis is listed

5.1.1 Configuration

The *Chemical Plume Dispersion* analysis contains the following settings:

- *Simulation output file* is common for several analyses and is explained in Chapter 5, subsection *Common analysis settings*
- *Vehicle* identifies the spacecraft, whose exhaust plume shall be analysed
- *Mass fraction* is the fraction of the engine exhaust mass flow which shall be considered in the computation
- *Wind* configures the features of the wind field, which is the driving factor of the plume dispersion:
 - *Velocity* is the velocity of the wind, which is assumed to be constant for all altitudes and the whole area of analysis. This value shall not be zero, since the underlying calculation is not valid for windless dispersion processes. A value equal to or greater than 1 m/s shall be used for sensible results.
 - *Direction* is the physical direction of the wind vector. It is assumed to be constant for all altitudes and the whole area of simulation.
- *Flight time* defines the observed flight section whose exhaust dispersion shall be analysed
 - *Lower bound* is the beginning of the observed flight section
 - *Upper bound* is the end of the observed flight section
- *Snapshot* serves the definition of the output which is a concentration map
 - *Time* is the time at which the plume dispersion shall be evaluated
 - *Altitude* is the altitude for which the evaluation shall be executed
 - *Longitude* serves the definition of the analysis area for which the concentration map shall be created
 - *Lower bound* denotes the western boundary of the snapshot area
 - *Upper bound* denotes the eastern boundary of the snapshot area
 - *Number of discretisation points* defines the resolution of the evaluation in longitudinal direction
 - *Latitude* serves the definition of the analysis area for which the concentration map shall be created
 - *Lower bound* denotes the southern edge of the snapshot area
 - *Upper bound* denotes the northern edge of the snapshot area
 - *Number of discretisation points* defines the resolution of the evaluation in latitudinal direction.

ANALYSES

- *Eddy diffusion coefficient* is a parameter modelling the scale of diffusion of the plume with increasing downrange distance

5.1.2 Background

A common approach of modelling the transport of airborne contaminants due to turbulent diffusion and advection by the wind is the utilisation of the Gaussian plume model. The most important assumptions in the implementation of this model within the *Chemical Plume Dispersion Analysis* are the following:

- The wind direction does not change during the observed time span
- The effects of transport processes due to wind/advection are much larger than the effects due to diffusion
- The initial velocity of the exhaust plume is neglected. So, this analysis should not be used for modelling the shape of exhaust cloud right after leaving the rocket engine but for the distribution of the plume concentration after several minutes. The exhaust plume of the rocket has roughly similar features like air and its eddy diffusion parameter K is constant.

For a detailed description of derivation of the Gaussian plume model and its variants, please refer to [157].

The *Chemical Plume Dispersion Analysis* is based on the so-called Gaussian Puff solution, which considers an instantaneous release of a limited amount of plume at a defined point:

$$c(r', x', y', z, t) = \frac{Q_T}{8(\pi r')^{\frac{3}{2}}} e^{\frac{-(x'-ut)^2 + y'^2}{4r'}} \left[e^{\frac{-(z-H_S)^2}{4r'}} + e^{\frac{-(z+H_S)^2}{4r'}} \right] \quad (5.1)$$

where:

- c is the concentration of the contaminant
- Q_T is the mass of the instantaneously released contaminant
- u is the wind velocity
- $x' = x - X_S$ is the downwind distance between a contaminant source and the evaluation point
- $y' = y - Y_S$ is the crosswind distance between a contaminant source and the evaluation point
- z is the altitude of the point being evaluated with $z = 0 \text{ m}$ being ground altitude
- t is the time with $t = 0 \text{ s}$ being the point in time of the release of the contaminant
- r' is an independent variable which is introduced in the scope of the derivation of the original Gaussian Plume Solution with an area-unit. It is elaborated in the following.
- H_S is the altitude of a contaminant source

The independent variable r is defined as follows:

$$r' = \frac{1}{u} \int_0^{x'} K(\xi) d\xi \quad (5.2)$$

ANALYSES

where K is the eddy diffusion coefficient.

As already mentioned above (see [157] for details), the Gaussian plume solution is only strictly valid for a constant K .

In order to model the behaviour of the exhaust plumes of a complete ascend trajectory for instantaneous plume, sources are positioned at every trajectory point. The Gaussian Puff solution is solved for every point with their respective time span between the release of the exhaust gases and the evaluation time. Afterwards the resulting concentration solutions are superposed in order to obtain the total result.

5.1.3 Output

The output of the *Chemical Plume Dispersion Analysis* is the data required for concentration maps showing the concentrations at the user defined altitude as colour gradient above the landscape. The area is limited by the *Longitude and Latitude Upper and Lower Bounds*. In the case that the user wants to analyse the temporal course of the propagation of the plume concentrations, several maps can be created via parameter variation using the *Batch mode* in *ASTOS*.

5.2 Data Timeliness Analysis

The time span (i.e. delay) between data generation (e.g. scientific data generated by sensors) on board a vehicle and receiving this data by a ground station can be an important mission analysis aspect. The *Data Timeliness* analysis considers the amount of data stored on all on-board data storage devices and calculates the time until this data has been transmitted to the receivers specified as *Link Partners* in the phase specific settings of the transmitter.

5.2.1 Configuration

The *Data Timeliness* analysis contains the following settings:

- *Simulation output file* is common for several analyses and is explained in Chapter 5, subsection *Common analysis settings*
- *Write analysis data only* is common for several analyses and is explained in Chapter 5, subsection *Common analysis settings*
- *Vehicle* is a drop-down list selector for the vehicle whose data timeliness shall be evaluated.

Note: The selected vehicle must contain at least one data system.

5.2.2 Background

The analysis evaluates for every time step and for all data buffers of the selected vehicle over all phases the time required for the total data amount stored on board until this time step to be transmitted by the assigned transmitter. The data buffers on board the vehicle do not assign priorities to data packages and operate according to a "first in, first out" (FIFO) logic with the predefined data transmission speed *Maximum data rate* in the *Configuration Panel* of the selected transmitter. This may cause that the whole data acquired by the sensors and stored on board the vehicle cannot be transmitted instantly even when there is visibility between transmitter and receiver, resulting in a delay between data acquisition and transmission at some time of the mission. Since the simulation of the scenario most likely does not end at a time when there is visibility between the transmitter of the vehicle and a receiver, for any data added after the last visibility window or for data which could not be transmitted during the last transmission session no delay can be computed (i.e. this data cannot be considered in the delay computation). In this case, the delay value is set to the last computed value.

5.3 Destructive Reentry Analysis

General Settings

The *Destructive Reentry* analysis requires the following general settings:

- *Simulation input file*
- *Simulation output file* is common for several analyses and is explained in Chapter 5, subsection *Common analysis settings*
- *Vehicle* is common for several analyses and is explained in Chapter 5, subsection *Common analysis settings*
- *Phase* is common for several analyses and is explained in Chapter 5, subsection *Common analysis settings*
- *Evaluation Time* is common for several analyses and is explained in Chapter 5, subsection *Common analysis settings*
- *Computation type* may be a *DARS* analysis (see Section 5.3.1 for details) or a *Ballistic Analysis* (see Section 5.3.2 for details). For both analysis types a risk analysis with *RAM* is possible (see Section 5.3.3 for details).

The environment specified in the selected phase is used by the destructive reentry analysis. *DARS* also considers the wind model, while the *Ballistic Analysis* does not.

5.3.1 DARS

DARS (Debris Analysis for Re-entry Spacecraft) is a tool developed by ESA (ESTEC) that analyses the atmospheric re-entry of a spacecraft or the different stages of launcher vehicles. Through a trajectory propagator and an aero-thermal module, *DARS* supplies the trajectory and the thermal state of each fragment. It is able to determine if the object reaches the surface of the planet or if it is demised on its path through the atmosphere. If the object reaches the surface of the planet, *DARS* computes the energy at the impact point and provides the data necessary to calculate the probability of casualty and fatality for a re-entry risk analysis. Additional outputs such as shape, position and dimensions of the footprint are computed by *ASTOS*.

ASTOS internally stores the impact points calculated by *DARS* as auxiliary items which can be visualized in a *Map-Plot* or *3D Map-Plot* using the *Viewer*.

Note: This analysis method requires the *ASTOS Reentry Safety* feature.

5.3.1.1 Setup of DARS

First select the previously defined analysis item in the *Analyses* tree (i.e. *Splash_analysis* when having followed the tutorial up to that point). Make sure that all settings described in chapter Scenario Setup in the Safety and Debris Risk Assessment Tutorial book are already applied and select *DARS* from the *Computation type* drop-down list. Continue preparing the analysis according to the following steps:

1. Insert a suitable *Maximum perigee radius* (i.e. 6978.0 Kilo-Meter).

Note: *ASTOS* computes the perigee radius (dependent on the *Evaluation time* setting) at the *Final* or *Initial* time of the specified *Phase*. If this value is larger than the *Maximum perigee radius* entry, the splash-down computation is not performed. This is necessary to avoid endless computation in case of a non-re-entry stage or vehicle. A perigee radius as defined for this scenario (corresponding to a perigee altitude of 100 km) guarantees the re-entry of the vehicle.

2. Select a *Rocket Body* as *Explosion model* (see Section 5.3.1.1.1).
3. Keep the *default* settings for the *Pre-breakup phase*.

Note: Instead of using aerodynamics computed by *DARS*, *custom* settings for the *Pre-phase breakup* allow to define alternative aerodynamics for the parent object using a ballistic coefficient approach. But only aerodynamics models of type *Tabular* and *User* are supported!

4. Leave all *Scaling factors* at their 1.0 default value. *Scaling factors* are useful in conjunction with the batch mode for a Monte Carlo analysis (see Section 5.3.1.1.2).
5. Concerning the *Simulation settings*, select a *Stepsize* of 1.0 Second and a *Stepsize output* of 3.0 Second for the *Dormand-Prince 5 integrator*.

ANALYSES

10. The *Materials* table (see Section 5.3.1.1.3) can be read from a *File*, from *ODBC* or defined by *Local* settings. Choose the latter option for this tutorial. For a description of the *source* options, please refer to chapter Profile Data Tables in the User Manual.
12. Add *Materials* to the table as shown in Fig. 5.1.

	Name	Tensile Strength	Density	Cp	Melting Temperat...	Spec Melting Heat	Emissivity
Units	-	Newton/Meter/Meter	Kilogram/Meter**3	Joule/Kilogram/Kelvin	Kelvin	Joule/Kilogram	None
1	SS304	3.0E8	7805.0	482.0	1700.0	286098.0	0.5
2	SS316	3.0E8	8030.0	461.0	1644.0	286098.0	0.5
3	AL2219	3.0E8	2713.0	864.0	867.0	386116.0	0.5
4	AL7075	5.4E8	2713.0	960.0	750.0	386116.0	0.5
5	AL6061	3.0E8	2713.0	896.0	855.0	386116.0	0.5
6	INC718	3.0E8	8190.0	435.0	1571.0	311664.0	0.5
7	TI6AL4V	3.0E8	4512.0	560.0	1943.0	393559.0	0.5
8	Columbium	3.0E8	8600.0	272.0	2740.0	290000.0	0.5
9	AL5083	3.0E8	2713.0	900.0	865.0	386116.0	0.5
10	Fiberglass	3.0E8	1268.0	1046.8	1200.0	232.6	0.5
11	AL7475	3.0E8	2713.0	880.0	750.0	386116.0	0.5
12	Plastic	3.0E8	110.0	1555.0	294.25	413000.0	0.5
13	Copper	3.0E8	8960.0	434.1	1356.0	243000.0	0.5
14	TI6AL4V_NOMELT	3.0E8	4512.0	560.0	101943.0	393559.0	0.5

Fig. 5.1: Materials settings for the scenario

9. Add triggers (see Section 5.3.1.1.4) to the *Triggers* section. Modify the existing default trigger *ID* (*Trigger_1*) and use the **Add** button to create the remaining ones.
10. Set the source of the *Fragments* table (see Section 5.3.1.1.5) as for the *Materials* table (i.e. *Local*).
12. Add all *Fragments* to the table as shown in Fig. 5.2.

	Name	Parent	Delta FP Speed	Delta FP Angle	Delta FP Azimuth	Shape	Quantity	Width Dia...	Length	Height	Mass	Material	Wall Tem...	Status	Triggers
Units	-	-	Meter/Second	Degree	Degree	-	-	Meter	Meter	Meter	Kilogram	-	Kelvin	-	-
1	DARS_parent	-	0.0	0.0	0.0	Cylinder	1	4.0	10.0	0.0	1050.746	TI6AL4V_NOMELT	300.0	Active	Alt_78.3
2	frag_1	DARS_parent	0.0	0.0	0.0	Cylinder	1	1.0	2.0	0.0	400.0	AL7075	300.0	Standby	
3	frag_2	DARS_parent	0.0	0.0	0.0	Box	1	1.0	2.0	1.0	600.0	AL5083	300.0	Standby	Alt_60
4	frag_3	DARS_parent	0.0	0.0	0.0	Sphere	1	0.2	0.0	0.0	5.0	TI6AL4V	300.0	Standby	
5	frag_4	frag_2	0.0	0.0	0.0	Box	1	1.0	0.5	1.0	150.0	AL5083	300.0	Standby	
6	frag_5	frag_2	0.0	0.0	0.0	Box	1	1.0	1.5	1.0	450.0	AL5083	300.0	Standby	

Fig. 5.2: Fragments settings for the scenario

13. For the *Risk analysis* (see Section 5.3.3), enable *Apply atmospheric uncertainties* and select *Average* for the *Population density data*.
14. Press **Save**.

5.3.1.1.1 DARS Explosion Model

The explosion model used in *DARS* is derived from the *EVOLVE* model developed by NASA [121]. Based on statistical information it creates distributions for the diameter, mass and velocity, which can be used to populate the *DARS* fragment table in case of an explosion (or fragmentation). Currently the direction of the velocity increment due to the explosion is spherically distributed using random numbers for heading and inclination.

ANALYSES

The explosion model combines three normal distributions to calculate the area-to-mass ratio distribution, which is needed for the computation of the fragment masses. In an iterative process the fragments' diameter and mass distribution is defined such that the summed-up mass equates to the mass from the source fragment.

The employed normal distributions depend on the diameter of the created fragment and following cases are considered:

- Fragment diameter $L_c \leq 0.08\text{m}$:

The area-to-mass ratio is calculated with the normal distribution N_0

$$D\left(\frac{A}{M}\right) = N_0(L_c) \quad (5.3)$$

using the values for the mean $\mu_0(L_c)$ and the variance $\sigma_0(L_c)$. The user must provide profile data table objects for these two coefficients.

- Fragment diameter $L_c \geq 0.11\text{m}$:

The fact, that the diameter can increase to a large value, two normal distributions are combined with a scaling factor α using the formula:

$$D\left(\frac{A}{M}\right) = \alpha(L_c) \cdot N_1(L_c) + (1 - \alpha(L_c)) \cdot N_2(L_c) \quad (5.4)$$

Therefore profile data table objects must be provided to calculate the values for the factor alpha and the normal distribution coefficients μ_1 , σ_1 , μ_2 , σ_2 for the requested fragment diameter L_c .

- Fragment diameter $0.08\text{m} < L_c < 0.11\text{m}$:

The area-to-mass ratio is interpolated linearly with the border values of the two cases before.

The data can be provided by using the dimensionless variable $\lambda = \log(L_c)$ or directly the diameter L_c .

5.3.1.1.2 Monte Carlo Analysis with DARS

Even if *DARS* follows a deterministic approach, normally some quantities are not precisely known. Therefore in parallel to the deterministic approach, the user can analyse the dispersion of the impact points via the *Batch-Mode Inspector* window. This is a functionality perfectly integrated in the *ASTOS* simulation platform. A set of *Scaling factors* can be defined as parameters in order to modify the nominal properties inside the fragments and material tables (refer to chapter Batch-Mode Inspector in the User Manual for details and examples).

5.3.1.1.3 DARS Material Table

The materials used by *DARS* for the definition of the fragment properties are configured in the *Materials* table. All materials are completely user-definable. Their definition requires the following parameters:

- *Name*: an arbitrary the name describing the defined material
- *Tensile Strength*: the maximum stress a material can withstand when stretched
- *Density*
- *Cp*: the material specific heat capacity
- *Melting Temperature*
- *Specific Melting Heat*: the material specific heat during melting
- *Emissivity*: the emission coefficient

All above properties already have suitable default *units*. If desired, they can be altered by the user by double-clicking on the respective unit cell in the *materials* table and then by clicking the button  in the selected cell.

5.3.1.1.4 DARS Trigger Table

The triggers used by *DARS* are configured using the *Triggers* section in the configuration panel. They require the following properties:

- *ID*: an arbitrary name describing the trigger
- *Type*: the trigger condition can be either:
 - *Altitude*: the altitude of the action
 - *Ablation*: the smelting percentage (0% means no mass difference and 100% means the complete mass is burned up. If a heat shield is defined for the fragment, than the heat-shield mass instead of the total fragment mass is used.)
 - *Pressure*: the dynamic pressure
 - *Temperature*: the wall temperature

Enter the desired value in the bottommost cell of the *Triggers* section. Its label varies according to the selected *Type*.

- *Action*: select the action to be performed after all trigger conditions are met. Possible values are:
 - *Automatic Fragmentation*: generates an explosion with a very low delta V and does not use the predefined fragments by the user
 - *Predefined Fragmentation*: activates the fragments defined by the user that are children of the fragment for which the reached trigger is specified.
 - *Explosion*: creates fragments according to:

ANALYSES

- total mass equal to the parent fragment mass
- a distribution of the fragment diameter defined by the explosion parameters
- a velocity change to a random direction

The new fragments are modeled as boxes with length and height calculated with random numbers.

5.3.1.1.5 DARS Fragment Table

DARS, requires that all objects that are analysed after the break-up of the stage or re-entry vehicle are specified. The *Fragments* table is used to configure the "parent" and other fragments' characteristics. The first object in the table is always treated as the parent object. Fragments are specified using the following properties:

- *Name*: an arbitrary identifier for the fragment
- *Parent*: the name of the fragment ancestor (empty for the parent fragment)
- *Delta FP Speed/Angle/ Azimuth*: the three component of the delta flight path (FP) velocity vector. Nominally the final state of the parent is used as initial state of the child fragment; for fragments with *status Standby* or *Sheltered*, the delta-V vector is added to the final velocity vector of the parent.
- *Shape*: double-click the cell to define the primitive shape of the fragment
 - *Sphere*
 - *Cylinder*
 - *Plate*
 - *Box*
- *Quantity*: number of identical fragments of this kind (used during the risk computation)
- *Width/Diameter*: the diameter of a *Sphere/Cylinder* or the first edge length of a *Box* or *Plate* (dependent on the selected *Shape*)
- *Length*: length of a *Cylinder* or the second edge length of a *Box* or *Plate*, unused in case of a *Sphere* (please insert 0.0 then)
- *Height*: Third edge length of a *Box* or thickness of a *Plate* unused in case of *Cylinder* or *Sphere* (please insert 0.0 then).
- *Mass*: initial mass of the fragment
- *Material*: name of this fragments's material as defined in the material table
- *Wall Temperature*: initial wall temperature of the fragment
- *Status*: initial status of the fragment with possible values:
 - *Active*: the fragment is propagated

ANALYSES

- *Standby*: the fragment is not propagated; its position and temperature is identical to the fragment specified in the *Parent* cell. The propagation starts only if the *Triggers* of the parent has the *Action* defined as *Predefined Fragmentation*. Once the propagation starts the initial position and the wall temperature is the final one of the parent.
- *Sheltered*: similar to *Standby*; the trajectory propagation starts only when the parent reaches a trigger. The difference is that the sheltered objects receive just a small amount of heat, so the temperature can differ from the parent one. Additionally a trigger check for the temperature can be performed on the fragment.
- *Triggers*: list of names for active triggers as defined in the *Triggers* section.

All above properties already have suitable default *units*. If desired, they can be altered by the user by double-clicking on the respective unit cell in the *materials* table and then clicking the button  that appears in the cell.

5.3.1.2 Simulation and Results Plotting

If you have followed the tutorial up to that point, now the scenario is ready for simulation and analysis. In this case, skip the paragraph "Open the Scenario" below and continue with the paragraph "Simulation and Results".

Open the scenario

In case you enter the tutorial here: the installation of *ASTOS* (with the **Reentry Safety** feature) contains a scenario that provides an example to demonstrate the use and capability of *DARS*. The analysis integrates the re-entry of the vehicle with user-defined multiple fragmentations, demise of debris and object impacts.

To import an example scenario prepared for *DARS* follow these steps:

1. Open *ASTOS*.
2. From the *Scenario* menu, select *New...* The *New Scenario* window automatically opens.
3. Specify an arbitrary *name of the scenario* and the *parent path* to where you want to locate your scenario.
4. Select the *Copy from existing scenario* checkbox and enter %ASTOS%\examples\Analysis\Reentry\Satellite_Reentry_Analysis.gtp where %ASTOS% is the installation path of *ASTOS*.
5. Press the **Create** button.

The scenario is loaded and ready for simulation and analysis.

Simulation and results

If not already applied, the scenario first has to be initialized and simulated. For that purpose:

- Select the root element *Analyses* in the *Analyses* tree. Then select the *Automatic* box for the defined analysis (i.e. *Splash_analysis*) in configuration panel.

ANALYSES

- Start an Initialization action by clicking the ribbon *Start→Actions→Initialize* button.
- Simulate with connected phases by clicking the ribbon *Start→Actions→Simulate Connected Phases* button.

The initialization and simulation process should terminate with status 0 and no errors should be printed in the *Execution Log*.

The visualization of the impacts and dispersion areas are now realized using the *Viewer* window. The result summary additionally provides fatality and casualty information for each fragment.

For the plots, first proceed as follows:

- Click on the *Window* menu, choose *Viewer*: the *Viewer* window already contains a 2D and a 3D Map-plot. (see also chapter Curve Plots in the User Manual)
- In the *Map Plot* tab the initial position is represented by a blue dot at the coordinates 48S-75W, while three impacting objects are displayed next to the Falkland Islands.
- The rectangles around the object visualize the dispersion area due to atmospheric uncertainties (see Section 5.3.3.2 and [122]).
- In the *Viewer* window, select *Data → Auxiliary Items...* (see also the **Auxiliary Items** section in chapter Types of Output and Naming Conventions of the User Manual) book on how to modify the display style of any auxiliary item.

The *3D Map Plot* tab contains the same objects from an aerial perspective. Especially note:

- the fragmentation events, displayed as red stars (see also the legend)
- the demised fragment displayed as green rhombus

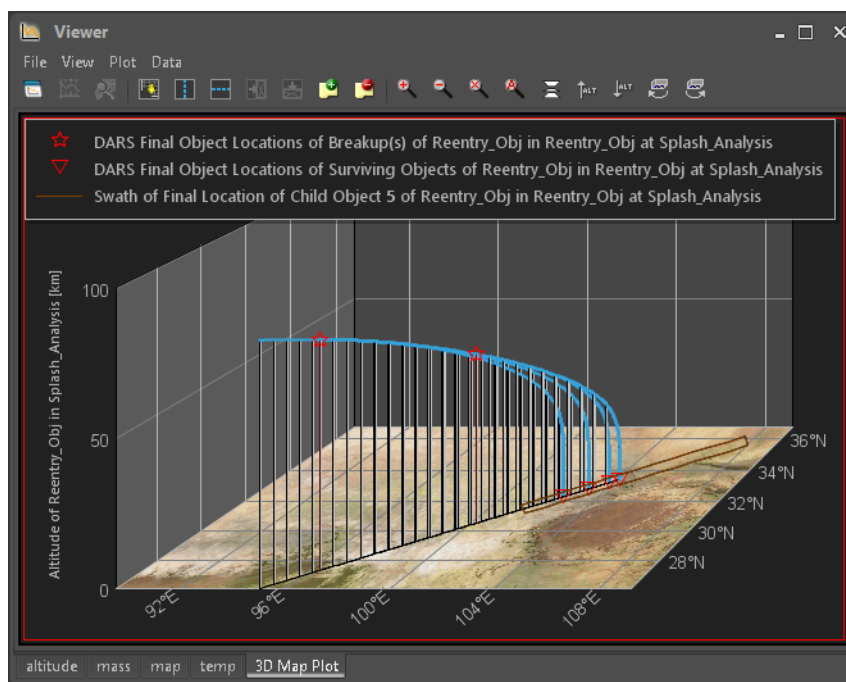


Fig. 5.3: DARS scenario: 3D view

ANALYSES

In the *Results* tree, select *Results Summary* → *Summary*: a textual information is printed in the *Execution Log*. Among others scalars, the fatality and casualty probability of each fragment are listed. Since the impacting objects are not on populated areas, the probabilities for population density are zeros, while the probabilities for ship or oil-rig strike are quite small.

Modify the impact position

Moving the impact fragments over the Falkland Islands verifies the risk computation over populated areas of *ASTOS*.

- Change the initial heading from 110.0
- Press **Save**.
- Initialize and simulate the scenario again.

The *Viewer* tabs automatically load the updated results and now one surviving fragment hits the islands (as well as its dispersion area). Recomputing the Result Summary, it can be noted that the fatality and casualty probability of the population density associated to the fragment 2 is not zero anymore.

5.3.1.3 DARS Output Files

The *Destructive Reentry* analysis using *DARS* generates the following output files:

- `DARS_Impacts.txt`: contains the states and properties of all fragments at the time of their impact. The destination folder for this file is the local `.\integ` folder of the scenario and its content is plain text.
- `reentry_analysis.struct`: contains the trajectory data and time dependent properties for all fragments as a binary file. This enables the user to plot the fragment trajectories as well as breakup, demise and impact positions, among others.

Note: The demise events are either due to melting or due to a interrupted integration due to the fact that the fragment's kinetic energy has become smaller than 15.0 Joule.

The destination folder for this file can be specified in the *Destructive Reentry* analysis settings and has to be an existing one.

5.3.2 Ballistic Coefficient Analysis

This stochastic analysis uses the ballistic coefficient and the vehicle mass (respectively jettisoned component mass) as an initial (respectively final) boundary constraint to calculate an equivalent product of the drag coefficient and a reference surface. From this result the drag acceleration is computed and used to integrate of the trajectory until impact (while lift is **not** considered).

Note: This method requires the *ASTOS Reentry Safety* feature.

5.3.2.1 Configuration

First select the previously defined analysis item in the *Analyses* tree (i.e. *Splash_analysis* when having followed the tutorial up to that point). Make sure that all settings described in chapter Scenario Setup in the Safety and Debris Risk Assessment Tutorial are already applied and select *Ballistic Analysis* from the *Computation type* drop-down list. Continue preparing the analysis according to the following steps:

- In case the impact altitude is known and not 0.0, define it as *Final altitude*. In this scenario, a value close to the ground (i.e. *1.0 Meter*) is chosen.
- Enable the *Pre-breakup phase* and define a *Breakup altitude* of 78km (i.e. *78000.0 Meter*) and the *Ballistic coefficient* for the computation before break-up. Use *150.0 Kilogram/Meter**2* in this case.
- In the *Ballistic coefficient* section select either *Table* or *Fixed loop* from the *Type* drop-down list. In the case of *Fixed loop*, *ASTOS* performs an impact analysis with a specified range of ballistic coefficients. Use the latter for this scenario.

Note: In case *Table* is used for the *Ballistic coefficient Type*, fill the provided table with the ballistic coefficients and the cross section areas of the desired fragment objects.

- Specify the range of ballistic coefficient using the following values:
 - *Count:* 5
 - *Step:* *10.0 Kilogram/Meter**2*
 - *Start:* *250.0 Kilogram/Meter**2*
- Specify the *Velocity increment* range for X/Y/Z trajectories (only an Y component is assumed for this scenario):
 - *Count:* *1 / 5 / 1*
 - *Step:* *0.0 / 50.0 / 0.0 Kilogram/Meter**2*
 - *Start:* *0.0 / -100.0 / 0.0 Kilogram/Meter**2*

With the *Velocity increment* loops the user can specify a perturbation to the velocity vector of the initial state. To apply no perturbation, set the loop *Count* to 1 and leave *Step* and *Start* at zero. The risk analysis is computed with the velocity perturbation representing the worst case.

- Enable *Apply atmospheric uncertainties* for the *Risk analysis* (see Section 5.3.3) and choose *Average Population density data* from the drop-down list. This selection is optional.
- Press **Save**.

5.3.3 Risk Analysis

The Risk Assessment Module (RAM) is a subcomponent of the destructive reentry analysis. It takes the final state of the propagated fragments and calculates the risk for humans, ships, oil-rig and airplanes due to each individual object. These numbers are summarized in overall risk values for each category. The calculation of the risk is similar to the method described in [122]. In contrast to this method, RAM uses population density information with a higher resolution of 2.5' in longitudinal and lateral direction [123]. It also provides the option to consider user-defined high-resolution population density data for local areas. Moreover, the risk assessment is computed also in respect of air-water traffic information. To find information about the generation and usage of ship and air traffic data, see Section 5.3.3.1.

In order to consider uncertainties the atmospheric density is varied by 20% and an additional impact uncertainty of ± 40 kilometers in cross range direction is assumed. The resulting impact swath and the nominal impact may be plotted with the integrated plotting tools of ASTOS (see Section 5.3.3.3).

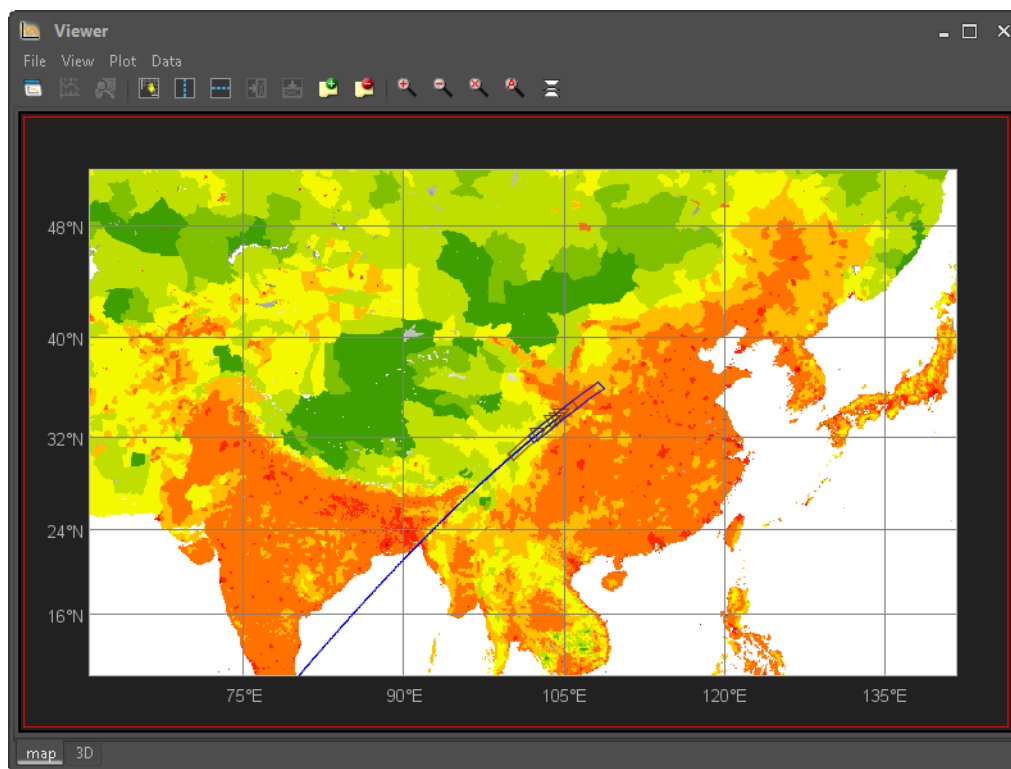


Fig. 5.4: Impact swath as calculated by RAM

5.3.3.1 Data sources and calculation comments for the risk assessment module

Population density

Population density data is retrieved from [123]. The most recent data refers to the year 2000, while a growing factor is used to compute the population density in different years. Precise

ANALYSES

population density data is used in the risk computation, while the background map presents a post-processing in order to create a limited number of density ranges. Each density range is associated to a colour.

The *RAM* user can choose between a standard population density (*Average*) and a modified one with increased peaks (*Peak*). For the latter the population density in strongly populated districts has been increased. The population density for the whole urban area is compared to the highest local density for several metropolises to gather information about a scaling factor in relation to the population density.

With this correlation the population density data is scaled only for cities using information about the rural and urban worldwide distribution to better reflect districts and small areas with high density. Custom population data can be provided as text file in tabular format for a given longitude and latitude range.

Ship traffic density and oil-rig distribution

The positions of commercial ships are taken from the World Meteorological Organization [125]. For this project, weather information with position data is provided by several ships, eg. 3374 vessels in 2005 representing roughly 11% of the 30851 merchant ships greater than 1000 gross tonnage. The data includes unique identifier codes for ships. Connecting the position data for each ship with straight lines leads to a good approximation of the ship tracks. To transform these individually tracked routes to generalized traffic density information the data has been blurred depending on the distance from the next land mass.

The oil-rig distribution data is generated from the light information on satellite pictures taken by night. On these images only lights on the oceans are considered and analysed for their position changes and spectrum. Typically the lights registered are the gas flares emitted by the oil-rigs. The National Oceanic and Atmospheric Administration (NOAA) provides such information with a resolution of 30 arc-seconds (about 1 km²).

Air traffic density

Real flight tracking data is analysed for variation of the flight route through bad weather condition or jet streams seasonal changes. Then for around 58200 collected routes the aircraft tracks are approximated with great circle arcs applying regional bandwidths for the spreading of the flight route. Therefore an elliptic distribution is used in horizontal direction for the arc form. All calculated routes, respectively the distribution of the aircraft on the route, are summed up to compute the air traffic density map for the time span of a day.

Advanced risk calculation

If checked the calculation of the casualty area changes and the method described in [126] is used. Thus also the effects of sheltering, collapsing buildings and explosive objects are considered. Only the unburned propellant mass is used to calculate the equivalent TNT mass and explosion strength (currently supported only in *DARS*).

Calculation comments

Some assumptions and approximations are summarized in this paragraph.

ANALYSES

- For the probability calculation for air traffic, a fixed vertical bandwidth of 5 kilometer is used for the air lane.
- The front surface area of the aircraft is fixed to 60 m², the cruising speed is considered 850 km/h.
- To distinguish between fatal and non-fatal hit between objects and aircrafts, the kinetic energy of the impact is compared with the energy of a bird-strike defined by several aviation authorities. Additionally the probability of a fatal aircraft strike is linearly scaled with the object mass from 0 up to 10 kg.
- For the casualty and fatality probability of a passenger, the casualty area is calculated with the front view area of 8 seated persons (4 m²).
- The kinetic energy which the object loses while penetrating the aircraft is equated with the kinetic energy of a bird-strike and is subtracted from the object energy for the determination of the fatality index.
- For ships a fixed top surface area of 20000 m² and velocity of 12.8 m/s is used representing an average commercial tanker
- For oil-rigs a fixed top view area of 2400 m² is assumed.
- When a building is hit by a fragment, one of two things can happen. Either the structure of the building is massive enough to resist the impact and no casualties are expected or the impacting fragment is heavy enough to penetrate the building. In the latter case, the probability of harmed population is increased with the strength of the structure due to more and heavier debris originating from the building damage. This circumstance is implemented similarly to the FSC Appendix I [126] with the following simplifications and a customization option:
 - According to Table 9 in the appendix of [126], fragments with a mass below 0.4 kg lead to a casualty area of zero square meters. This is implemented as a threshold and can be customized by the user.
 - A difference to the tables in the appendix of [126] is that the implemented risk analysis does not differentiate between different roof strengths and fragment densities. Instead, the values of the tables have been combined into a single formula using the impact energy as input: $C_{sheltering} = 0.108159 * 1.653288^{\log_{10}(E_{kin})}$. Its output is a sheltering coefficient which increases the calculated impact area.

5.3.3.2 Set up the risk assessment

To perform a risk analysis with *DARS* or *Ballistic Analysis* the *Risk_Analysis* flag should be activated.

Note: In case of *Ballistic Analysis*, the option *Table* must be selected in order to compute the safety analysis, see Section 5.3.2.1 for more details.

ANALYSES

The user can adjust the risk analysis with various settings to meet his needs and applicable regulations:

- *Apply atmospheric uncertainties* - A variation of the atmospheric density by 20% in either direction is applied to compute down range distribution; a fixed value of $\pm 40\text{km}$ in cross range direction is assumed. This approach is described in [122].
- *Population density data* - The user can select between an average or a peak population density in the population cell. Latter is more conservative and better considers urban areas with population density much higher than in the remaining area of the population cell.
- *Custom population density data* - Additionally the user can superimpose the global population data selected above with own data for a local area.
- *Casualty area definition according to* - The user can select from two different definitions of the casualty area: FAA Flight Safety Analysis Handbook or NASA Safety Standard 1740.14.
- *Casualty kinetic energy threshold* - The minimum kinetic energy of a fragment that can cause a casualty can be customized (default is 11J).
- *Aircraft penetration kinetic energy* - The kinetic energy required to penetrate an aircraft can be customized here.
- *Sheltering effect threshold* - Minimum fragment mass to consider sheltering effects according to Australian Flight Safety Code (default is 0.4kg).

Custom population density data - file format

The data file format is currently limited to a text format. Further formats will be added in the future according to customer needs.

The text file may contain only floating point numbers separated by space or tab. The first line must contain minimum and maximum longitude. The second line must contain minimum and maximum latitude. All values must be specified in degree.

Note: If the date crossing border intersects the area, then the minimum is the left but mathematically larger value (e.g. $+160^\circ$) while the maximum is the mathematically smaller value (e.g. -160).

Starting with the third line the population density values have to be given in matrix format, i.e. each line represents values for a constant latitude, starting with the minimum longitude value and ending with the value for the maximum longitude. The rows must be sorted from minimum to maximum latitude. The values are given in units of people per square kilometer. A constant angular step size between the values is assumed in longitudinal and in latitude direction.

Below is an example for a valid file with a longitude range from -78 to -72 degree and a latitude range from -16 to -12 degree. The value for $16^\circ\text{S } 78^\circ\text{W}$ is 1000.0 inhabitants/kilometer.

```
-78.0 -72.0
```

ANALYSES

```
-16.0 -12.0
1000.0 2000.0 3000.0 4000.0
10000.0 20000.0 30000.0 40000.0
100.0 200.0 300.0 400.0
10.0 20.0 30.0 40.0
1.0 2.0 3.0 4.0
```

5.3.3.3 Simulation and Results

from RAM

To simulate and create the risk assessment results, proceed as follows:

- Start an Initialization action by clicking the ribbon *Start→Actions→Initialize* button.
- Simulate with connected phases by clicking the ribbon *Start→Actions→Simulate Connected Phases* button.
- In the *Results* tree, select *Result Summary* and press **Summary** in the configuration panel. The summary information is printed in the *Execution Log* and a text file (*Result_summary.dat*) is created in the scenario .gtp folder.

The result summary presents the relevant information about the analysed scenario in a formatted text. It contains different sections:

- Scenario introduction, with date, time and scenario description.
- COST FUNCTIONS: relevant in case of an optimization procedure.
- PERFORMANCE INFO: shows the constraints of the last phase.
- SCALAR QUANTITIES OF PHASE x with the resulting Impact area (total area and semi-major/minor axes), the Human risk, the risk for air traffic, ship traffic and oil-rigs due to each object analysed. For all probabilities except for the ship traffic and oil-rig the casualty and fatality probability is computed.

Once the scenario is simulated, only few steps are necessary to plot the impact points:

1. Select *Curve Plots* in the *Results* tree (see also chapter *Curve Plots* in the User Manual).
2. In the *plot* section, select *Map-Plot* from the *plot type* drop-down list.
3. Press the **Add Defaults** button.
4. The trajectory is displayed in the *Viewer* window after pressing **Show in New Tab**. To show the impact points and swaths activate the auxiliary items: select *Data → Auxiliary Items...* from the *Viewer* menu (see also the **Auxiliary Items** section in chapter *Types of Output and Naming Conventions*, of the User Manual book).
5. In the *Viewer* window (see Fig. 5.4), use the mouse to obtain the desired view: left click to pan, right click to zoom (see also chapter *Viewer* in the User Manual).

ANALYSES

6. Change the background map to *Earth - Population Density* by selecting *Plot → Background...* from the *Viewer* window

To plot a 3D view:

1. In the *Plot* section of the *Curve Plots* configuration panel, select *3D Map-Plot* from the *Plot type* drop-down list.
2. Press the **Add Defaults** button.
3. Use the mouse to adjust the desired view: left click to pan, right click to zoom.
4. Repeat from step 4 above to display the impact points.

5.4 DrNUM Aerodynamics Analysis

DrNUM is a CFD software developed by engits that utilizes the performance of modern graphics cards. This analysis creates the required input configuration for DrNUM and calls it to run the computation. The results are written back into the configuration file. The format of the file allows using it as file import for the tabular aerodynamics. The geometry of the outer shell is automatically created by ASTOS during the simulation in terms of a Wavefront OBJ file. Such file can be used as the geometry input file for DrNUM. Of course, OBJ files from other sources could be used as well.

Configuration

- *DrNUM executable* specifies the script or executable that is automatically run after the configuration file has been created.
- *DrNUM configuration file* defines the target configuration file.
- *Geometry file* specifies the input OBJ file.
- *Reference area* is used to derive coefficients from the calculated forces.
- *Reference length* is used to derive coefficients from the calculated moments.

Further parameters define the ranges of Machnumber, angle of attack and side-slip angle as well as the number of steps.

Note: For the angles, a zero-crossing detection will make sure that 0° will be one of the listed steps in case 0° is within the specified range and the number of steps is at least 3. For the Machnumber a 1-crossing detection will make sure that Mach 1 will **not** be one of the intermediate steps, as this would cause problems in the CFD solver.

5.5 Earth Observation Mission Analysis

Description

ANALYSES

The *Earth Observation Mission* analysis creates a report template in line with the structure outlined in the annex of [145] and partially fills the chapters with content of the current and also additional scenarios, e.g. one scenario for the Launch and Early Operations phase (LEOP) and one for the operational mission phase. This chapter is structured as follows:

- Section 5.5.1 contains the configuration panel of this analysis.
- Section 5.5.2 contains some background information, assumptions and equations used by this analysis.
- Section 5.5.3 provides details about the report created from this analysis.

As indicated by the name of this analysis it can only be performed for vehicles in orbit around Earth.

5.5.1 Configuration

The *Earth Observation Mission* analysis contains the following settings:

- *Add report template chapter information text* allows to add a description to each chapter in the report containing information about what should be the content of this chapter.
- *Used scenarios* is a list of scenarios to be considered when filling the report with content. The current scenario is already in the list. Additional scenarios can be added or removed from this list by using the corresponding buttons. The *Selected item* panel contains additional information about the selected item in the list.
- *General assumptions and models* is the title of one of the chapters automatically filled with content. The *Satellite* drop down menu allows to select a vehicle from the current scenario. The models, dimensions and masses of this vehicles are used to fill this chapter. The file selector *Simulation file* allows to select the simulation file from which the content is extracted. By default this is the `simulation.struct` file from the current scenario.
- *Launch and early operations phase* is the title of the chapter containing information about the LEOP phase. The *Scenario ID* selector allows to select any scenario from the *Used scenarios* list and with the *Simulation file* option any simulation file from within this scenario. The selected scenario for the LEOP chapter should contain a coverage analysis for ground station coverage and an eclipse analysis for the orbit of the LEOP phase.
- *Operational orbit* is the title of the chapter containing information about the operational phase of the mission. The *Scenario ID* selector allows to select any scenario from the *Used scenarios* list and with the *Simulation file* option any simulation file from within this scenario. The selected scenario for the Operational Orbit chapter should contain a coverage analysis for ground station coverage, an eclipse analysis for the operational orbit and an *Operational Lifetime Prediction* analysis to compute delta-V budget for orbit maintenance and disposal strategy. The operational orbit to be maintained is assumed to be constant. Changing the reference orbit during the operational phase is not supported by this analysis!

5.5.2 Background

This analysis does not compute anything on its own but merely takes information available from simulation results and other analysis to partially fill the report structure foreseen in the annex of [145]. Details about which information is taken from which source can be found in Section 5.5.3. Check the Model Reference of these sources to get background information on the assumptions and models behind the output used in the generation of the report of this analysis.

5.5.3 Output

This chapter provides some background information about the report content which is automatically added and the sources from which the information is taken. The report generated by this analysis is most useful if the *Format* is set to *Word*. The generated docx file can then be easily modified and completed manually. But caution is necessary since recreating this report would overwrite modifications if the file name was not changed! For more information on report generation see Reports in the in the User Manual.

General Assumptions and Models

The content added to the chapter *General Assumptions and Models* contains the following:

■ Basic Spacecraft Data

- **Dry mass** is the mass of the vehicle without propellant.
- **Wet mass** is the total mass of the vehicle including propellant.
- **Payload mass** is the total mass of the assembly of type *Payload*.
- **Dimensions** are the dimensions of the vehicle in the body frame of the vehicle. This is computed by getting the coordinates of all 8 vertexes of the bounding box of each vehicle part in the body frame of the vehicle and then calculating the difference between the minimum and the maximum coordinate of all parts for each axis.
- A **Vehicle plot** is added at the end of the **Basic Spacecraft Data** section with the x_B -axis of the body frame pointing upwards and the y_B -axis pointing to the left

■ Orbit Perturbations

- **Gravity Field** adds information about the gravity model of the central body, e.g. ELLIPSOID gravity and the values used for J2 up to J6.
- **Atmospheric Drag** will contain the following information if the vehicle in the scenario uses an atmospheric model and an aerodynamic model:
 - Type of atmosphere model used, e.g. JB2006
 - Aerodynamic cross section
 - Space weather coefficients, e.g. F10.7 index (if supported by the atmospheric model used)

ANALYSES

- Ballistic coefficient
- Drag coefficient
- **Solar radiation pressure** will describe the model used, the reference surface and the reflectivity coefficient
- **Third Body Perturbations** will list the IAU identifiers of celestial bodies selected for perturbations.
- **Other Mission Relevant Perturbations** will contain IAU identifiers selected for relativistic effects.
- **Selection of Operational Orbit**
 - **Injection Orbit** will list Keplerian Elements of the orbit at the initial time of phase 1. The epoch is given as julian date in the time standard TT.

Launch and Early Operations Phase

The following list provides some information about the content automatically added to the chapter *Launch and Early Operations Phase*:

- **Initial Ground Track and Ground Station Coverage**: In case the scenario selected for this chapter contains a *Coverage* analysis two plots are automatically added to this section. The first plot is a 2D plot over time with the coverage of the vehicle from all ground stations used in this scenario. The second is a map plot also showing the coverage of the vehicle from all ground stations. A list below these plots contains the name of all ground stations in this scenario. For details and background information about the *Coverage* analysis refer to .
- **Eclipse Analysis during LEOP**: In case the scenario selected for this chapter contains an *Eclipse* analysis the following content is added to this section:
 - A table containing the maximum duration of all eclipses in the simulation and the minimum duration of sunlight. Both values are reported in seconds.
 - A 2D plot of the eclipse pattern over time.

For details and background information about the *Eclipse* analysis refer to Section 5.6.

Operational Orbit

The following list provides some information about the content automatically added to the chapter *Operational Orbit*:

- **Reference Orbit**: The table will list Keplerian Elements of the orbit at the initial time of phase 1. The epoch is given as julian date in the time standard TT.
- **Orbit Evolution** provides 2D plots of the evolution of the Keplerian Elements semi-major axis, eccentricity, inclination, right ascension of ascending node and argument of periapsis.
- **Ground Station Coverage** is similar to the section *Initial Ground Track and Ground Station Coverage* in the LEOP chapter.

- **Eclipse Analysis** is similar to the section *Eclipse Analysis during LEOP* in the LEOP chapter.
- **End of Life Operations. De-Orbit Strategy:** If the scenario contains an *Operational Lifetime Prediction* analysis the deorbit strategy is added to this section. For details see Section 5.13.
- **Lifetime Delta-V Budget** will contain delta-V requirements for orbit maintenance depending on the type of operational orbit if the scenario contains an *Operational Lifetime Prediction* analysis. For details refer to Section 5.13.

5.6 Eclipse Analysis

The amount of sun radiation energy reaching a satellite along its trajectory is a key information, which has a strong impact on several subsystems (e.g. power and thermal management) and instruments. Depending on the vehicle illumination conditions, solar panels collect energy to recharge batteries on board, the thermal system has to deal with a different amount of heat and sensors (e.g. cameras, antennas, etc) may be overheated or blinded by sunlight.

The *Eclipse* analysis can determine when and for how long the satellite is shaded by any celestial body and cannot receive direct sunlight or is fully or only partially illuminated. In the latter case, a percentage of the non-visible Sun surface is provided.

Tip: Since the duration of partial eclipses may be rather short (e.g. in LEO a few seconds) a small simulation output step size is needed to detect them. Instead of using a small output step size in the whole simulation, using the Automatic simulation output refinement (ASOR) (see chapter Running Analyses in the User Manual on how to activate ASOR) is recommended for an eclipse analysis if the required accuracy is below one second for reported partial eclipse durations.

5.6.1 Configuration

The *Eclipse Analysis* analysis contains the following settings:

- *Simulation output file* is common for several analyses and is explained in Chapter 5, subsection *Common analysis settings*
- *Write analysis data only* is common for several analyses and is explained in Chapter 5, subsection *Common analysis settings*
- *Vehicle* identifies the spacecraft whose eclipses shall be analysed.

5.6.2 Background

Three different types of eclipse are considered by the *Eclipse* analysis [117]:

- *Partial eclipse*: the vehicle is within the penumbra, but outside the umbra. In this case, a portion of the Sun's disk is overlapped by the obstructing body
- *Total eclipse*: the Sun is not visible at all from the vehicle which is in the umbra region
- *Annular eclipse*: the vehicle is farther from the obstructing celestial body and an annulus or ring of the bright solar disk surrounding the disk of the obstructing body is visible by the vehicle

The different eclipse types are shown in Fig. 5.5.

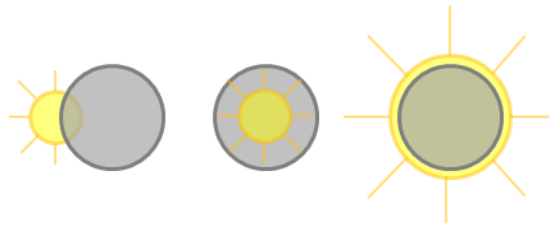


Fig. 5.5: Eclipse geometry: partial, total, annular

Partial and annular eclipses are two different types of penumbra conditions. The formulas to compute the eclipse function value *Eclipse*, which is always between 0.0 (full light condition) and 1.0 (total eclipse), are given as follows [129]:

- Partial eclipse:

$$Eclipse = \left(\pi - \cos \rho_s \arccos \left(\frac{\cos \rho_p - \cos \rho_s \cos \theta}{\sin \rho_s \sin \theta} \right) - \cos \rho_p \arccos \left(\frac{\cos \rho_s - \cos \rho_p \cos \theta}{\sin \rho_p \sin \theta} \right) - \arccos \left(\frac{\cos \theta - \cos \rho_s \cos \rho_p}{\sin \rho_s \sin \rho_p} \right) \right) \cdot \frac{1}{\pi(1 - \cos \rho_s)} \quad (5.5)$$

- Annular eclipse:

$$Eclipse = 1 - \left(\frac{1 - \cos \rho_p}{1 - \cos \rho_s} \right) \quad (5.6)$$

All of the aforementioned eclipse conditions are illustrated in Fig. 5.6

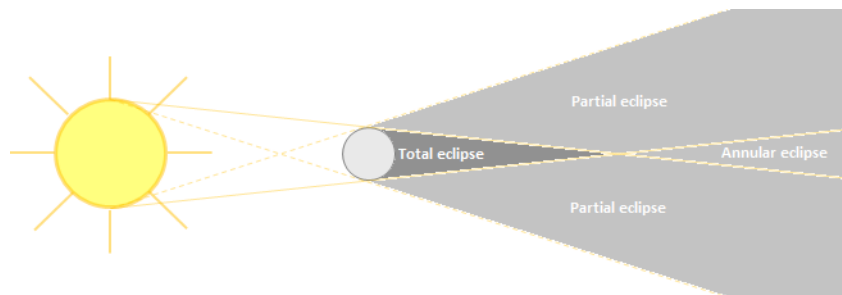


Fig. 5.6: Umbra/penumbra conditions

5.6.3 Output

The following output is generated by the *Eclipse* analysis:

- *Eclipse* is an auxiliary function whose value is equal to 1.0 if a full eclipse occurs and equal to 0.0 if the vehicle is fully illuminated. Intermediate values are used for partial eclipses giving the percentage of non-visible Sun surface.
- *Duration light* is an auxiliary function which stores the duration without interruption for which the *Eclipse* function is equal to 0.0. Its value is reset to 0.0 each time the *Eclipse* value becomes bigger than 0.0.
- *Duration umbra* is an auxiliary function which stores the duration for which the *Eclipse* function is equal to 1.0. Its value is reset to 0.0 each time the *Eclipse* value becomes smaller than 1.0.
- *Duration penumbra* is an auxiliary function which stores the duration for which the *Eclipse* function is not equal to 0.0 or 1.0. Its value is reset to 0.0 each time the *Eclipse* value becomes equal to 0.0 or 1.0.
- *Duration eclipse total* is an auxiliary function storing the total duration for which the *Eclipse* function is not equal to 0.0. Is is the sum of the umbra and penumbra duration and is never reset.
- *Minimum light duration* is an auxiliary item which stores the minimal of all maximal values (i.e. the peak value before a reset) of the *Duration light* function. This is the minimal of all peak values at the time steps before the beginning of an eclipse.
- *Maximum umbra duration* is an auxiliary item which stores the maximal of all maximal values (i.e. the peak value before a reset) of the *Duration umbra* function. This is the maximal of all peak values at the time steps before the beginning of a light condition or penumbra.
- *Maximum penumbra duration* is an auxiliary item which stores the maximal of all maximal values (i.e. the peak value before a reset) of the *Duration penumbra* function. This is the maximal of all peak values at the time steps before the beginning of a light condition or umbra.
- *Mean light duration* is an auxiliary item which stores the average duration of sunlight between eclipses. It is computed as the sum of all peak values of the *Duration light* function divided by the number of periods with sunlight.
- *Mean umbra duration* is an auxiliary item which stores the average duration of eclipse phases. It is computed as the sum of all peak values of the *Duration umbra* function divided by the number of umbra periods.
- *Mean penumbra duration* is an auxiliary item which stores the average duration of partial eclipse phases. It is computed as the sum of all peak values of the *Duration penumbra* function divided by the number of penumbra periods.

5.7 GNC Analysis

The *GNC* analysis provides additional output functions which can be used for the design process of a controller algorithm. The analysis description is structured as follows:

- Configuration of the analysis, see Section 5.7.1
- Background information, see Section 5.7.1
- Generated output by the analysis, see Section 5.7.3

5.7.1 Configuration

The *GNC* analysis contains the following settings:

- *Simulation output file* is common for several analyses and is explained in Chapter 5, subsection *Common analysis settings*
- *Write analysis data only* is common for several analyses and is explained in Chapter 5, subsection *Common analysis settings*
- *Vehicle* identifies the spacecraft whose eclipses shall be analysed.

Note: The specified *Vehicle* needs to use aerodynamics models of type *Tabular* or *MisDat* throughout all phases.

5.7.2 Background

The aerodynamics output is computed according to

$$C_{N,\alpha} = \frac{C_N - C_N(\alpha = 0)}{\alpha} \quad (5.7)$$

and

$$C_{M,Q} = \frac{C_M - C_M(Q = 0)}{Q} \quad (5.8)$$

with the normal force coefficient C_N , the angle of attack α , the pitching moment coefficient C_M and the body pitch rate Q .

Note: All coefficients are represented in the body-fixed (B) coordinate system (refer to Section 9.3.3.1 for details).

5.7.3 Output

The following output functions are generated by the GNC analysis:

- Related to the aerodynamics of the vehicle the following output functions are provided:
 - The aerodynamic reference area related to the provided aerodynamic coefficients
 - The aerodynamic reference length related to the provided aerodynamic moment coefficient
 - The (non-total) angle of attack slope of the normal force coefficient ($C_{N,\alpha}$)
 - The body rate Q slope of the pitching moment coefficient ($C_{M,Q}$)

Both aerodynamic coefficients are represented in the body-fixed coordinate system (refer to Section 9.3.3.1 for details).

- Related to the controllability of the vehicle, the following output functions are generated:
 - The aerodynamic efficiency $A6$
 - The thruster efficiency $K1$ of the thrust vector control system
 - The required deflection angle of the thrust vector control system
 - The control requirement CR of the thrust vector control system
 - The maximum angle of attack during wind gust
 - The shear wind magnitude
 - The normal force component due to shear wind

The thrust vector control system of the vehicle comprises all actuators specified as *Main_Engine_TVC*.

- For each thruster used as reaction control system the nominal (i.e. full throttle) mass flow rate and thrust are generated

5.8 Interplanetary Transfer Analysis

In the case of an interplanetary transfers scenario, it may be of interest to find departure and arrival dates such that a minimum propellant mass is required for the mission. For such a scenario, the departure date, the arrival date and the corresponding required fuel mass can be considered as triples and can be representation either as a 3-D surface or as a 2-D chart showing contours of equal propellant consumption, both representations featuring the departure and arrival dates along the x and y axis respectively. In some cases, instead of the propellant consumption the characteristic energy $C3$ or the delta- V is used in the chart. The *Interplanetary Transfer* analysis computes all data required for the generation of the 3-D surface plot mentioned above. Details on this analysis configuration, model and output is given in the following three subsections:

ANALYSES

- How to configure the analysis is described in Section 5.8.1
- The model background is given in Section 5.8.2
- The output generated by the analysis is listed in Section 5.8.3

5.8.1 Configuration

The *Interplanetary Transfer* analysis contains the following settings:

- *Simulation output file* is common for several analyses and is explained in Chapter 5, subsection *Common analysis settings*
- *Initial date* configures the boundaries for the departure date of the transfer and requires the following settings:
 - *Lower bound* is the earliest departure date of the transfer
 - *Upper bound* is the latest departure date of the transfer
 - *Stepsize* defines the stepsize for the computation of the actual departure date

For a more detailed description how to configure a date, please refer to Section 4.5.

- *End condition* sets the boundaries of the arrival date of the transfer and supports the following options:
 - *Transfer duration* defines the duration boundaries of the transfer. The arrival date is then computed from initial date and transfer duration. The following settings are required for the *Transfer duration*:
 - *Lower bound* is the minimum duration of the transfer
 - *Upper bound* is the maximum duration of the transfer
 - *Stepsize* defines the stepsize for the computation of the actual arrival date
 - *Final date* defines the arrival date boundaries of the transfer. The transfer duration is then computed from initial date and final date. The *Final date* settings are:
 - *Lower bound* is the earliest arrival date of the transfer
 - *Upper bound* is the latest arrival date of the transfer
 - *Stepsize* defines the stepsize for the computation of the actual arrival date

For a more detailed description how to configure a date, please refer to Section 4.5.

- *Truncation of delta-V* is an optional setting which allows the user to specify a threshold value for the delta-V, which is saved in the analysis output file. Any computed delta-V during the transfer which is bigger than the specified value will be replaced by the latter in the *Simulation output file*. Since the delta-V of some interplanetary transfers can become

ANALYSES

quite large depending on the initial and final date, this setting can be used for the purpose of scaling the analysis data before plotting.

- *Initial celestial body* sets the departure celestial body for the transfer from a list of all defined celestial bodies in the scenario.
- *Final celestial body* sets the arrival celestial body for the transfer from a list of all defined celestial bodies in the scenario.
- *Number of complete orbits* defines the number of orbits around the center body of the transfer leg. Zero means, the transfer leg is less than one revolution. Here, the following input is required:
 - *Lower bound* is the minimum number of complete orbits to be considered for the transfer
 - *Upper bound* is the maximum number of complete orbits to be considered for the transfer

In case the number of complete orbits shall be fixed, both *Lower bound* and *Upper bound* shall be set to the same value.

- *Orbit direction* defines the orbit direction of the transfer leg, i.e. the direction of motion of the vehicle. The options thereof are:
 - *Prograde* limits the motion of the vehicle in the same direction as the motion of the initial celestial body
 - *Retrograde* limits the motion of the vehicle in the direction as the motion of the initial celestial body
 - *Both* considers the motion of the vehicle in both directions: prograde and retrograde
- *Transfer type* is the kind of the transfer leg since there are two solutions of the Lambert's problem. The options thereof are:
 - *Short arc* limits the transfer to inbound trajectories, i.e. moving towards the central body
 - *Long arc* limits the transfer to outbound trajectories, i.e. moving away from the central body
 - *Both* considers both (inbound and outbound) solutions of Lambert's problem

5.8.2 Background

The *Interplanetary Transfer* analysis computes the delta-V required in order for a spacecraft to leave the initial celestial body with zero initial relative velocity with respect to it and arrive at the final celestial body with zero relative velocity with respect to it. In order to determine the delta-V, the Lambert's problem is solved by this analysis. The formulation of the Lambert's problem (see [49], pp. 227 for more details) in the case of an interplanetary transfer is as follows.

- The spacecraft initial position is the same as the position of the initial celestial body at the initial time for the transfer.

ANALYSES

- The spacecraft final position is the same as the position of the final celestial body at the final time of the transfer.
- The time interval for the transfer is defined by the pair of initial and final times.
- The orbit for the interplanetary transfer is a Keplerian orbit with the Sun as a central body.

The solution of the Lambert's problem gives the *required* velocity vectors $\mathbf{v}_1$ and $\mathbf{v}_2$ for the interplanetary transfer from the initial position to the final position. More precisely, $\mathbf{v}_1$ is the velocity vector at the end of the initial spacecraft maneuver. $\mathbf{v}_2$ is the velocity vector just before the engaging maneuver when arriving at the final celestial body.

The vector difference between $\mathbf{v}_1$ and the initial celestial body velocity vector yields $\Delta\mathbf{v}_1$ which is the required delta-V vector to leave the initial celestial body. Analogously, $\mathbf{v}_2$ minus the final celestial body velocity vector yields $\Delta\mathbf{v}_2$ which is the required delta-V vector to engage into the final celestial body orbit. The sum $\|\Delta\mathbf{v}_1\| + \|\Delta\mathbf{v}_2\|$ is the total delta-V magnitude computed by the *Interplanetary Transfer* analysis.

5.8.3 Output

All output data of this analysis is stored in a dedicated output file `./integ/interplanetary_transfer.struct` and it consists of the following functions:

- `julian_date_initial` is the date of the departure as Julian date in terrestrial time (TT)
- `julian_date_final` is the date of the arrival as Julian date in terrestrial time (TT)
- `transfer_duration` is the duration of the transfer
- `delta_v` is the required delta-v of the transfer

It is suggested to use the *Surface plot* (see Surface Plots in User Manual) in the ASTOS GUI in order to plot the output data of the *Interplanetary Transfer* analysis. For example, this can be done as follows:

- `julian_date_initial` on the X-axis
- `julian_date_final` on the Y-axis
- `delta_v` on the Z1-axis

5.9 Launch Safety Analysis

The *Launch Safety* analysis estimates the risk of casualty and fatality for launchers when performing a launch from any launch pad. This analysis requires the launcher trajectory to be already simulated before running the analysis.

The *Launch Safety* analysis combines the debris impact points resulting from the vehicle fragmentation with the results by the risk analysis module (see Section 5.3) to compute the risk

ANALYSES

of casualty and fatality in case of launcher failure during the vehicle ascent while considering the population density of the areas crossed by the instantaneous impact point line (see Section 5.9.2). The user has the possibility of using the explosion model available in ASTOS for the generation of vehicle debris.

This analysis is also capable of computing the flight corridor (see Section 5.9.2) for any ascending trajectory as currently defined by US regulations [113]. This can help the user to assess the risks when performing a launch from a new launch pad or when launching from an existing launch pad into a different trajectory.

5.9.1 Configuration

The *Launch Safety* analysis contains the following settings:

- *Simulation output file* is common for several analyses and is explained in Chapter 5, subsection *Common analysis settings*
- *Write analysis data only* is common for several analyses and is explained in Chapter 5, subsection *Common analysis settings*
- *Launcher* defines the launched vehicle for which the analysis shall be performed. The list contains the vehicles listed under *Vehicles & POIs Definition*.
- *Launcher class* defines a list of constants used to compute the flight corridor. The classes are defined according to [113] and depend on the mass (in kg) which shall be put into orbit. The table below gives this classification:

Table 5.1: Launcher classes

185 km Orbit	Small	Medium	Medium Large	Large
28° inclination	≤ 2000	2000-5000	5000-8400	>8400
90° inclination	≤ 1500	1500-3800	3800-6800	>6800

- The user can specify a *maximum perigee altitude*, which defines up to what perigee altitude (of the launcher) the analysis is run. Once the launcher reaches that perigee altitude, the analysis is stopped.
- The *Explosion model* consist of two separate modules which can be run independently from each other. Dedicated output (see Section 5.9.2) is generated by each module only if it is activated by the user. If both modules are disabled only the flight corridor and the instantaneous impact point are computed. The two modules of the *Explosion model* are:
 - The *Fragments* module simulates the generation of explosion fragments (see Section 5.3 for more details) at each time point along the whole trajectory and propagates their trajectories in order to evaluate the impact risks. The explosion model generates a relatively large number of fragments and if this feature is active the computational time may become very high, depending on the duration of the trajectory itself. The generation model for the explosion fragments is a statistical one and requires a seed number for its computations. The user can choose how the seed number is defined in the *Random number generator* setting:

ANALYSES

- A *Random seed* generates different seed numbers every time the analysis is run. In this way two consecutive runs of the same scenario will lead to different results, because different fragments will be generated at one and the same time step.
- A *Fixed seed* has the effect that the same seed value is used each time the explosion model is run (i.e. every time step). Thus the analysis generates fragments with the same **distribution** for every time step along the whole trajectory.
- An *Incremental seed* has the effect that the seed value is incremented during every time step along the trajectory and the resulting explosion fragments for each time step have different distributions. The sequence of seed numbers is the same each time the analysis is run and the results of the analysis are reproducible.

The user can choose between different options concerning the computation of the induced velocities of the explosion fragments in the *Induced velocity scaling* setting:

- If *None (native Evolve results)* is selected, the native Evolve results are used.
 - By choosing *fixed fraction of propellant energy*, the induced velocity of the explosion fragments is adjusted such that the total kinetic energy increment of all explosion fragments equals the given fraction of the propellant energy. The propellant energy is based on the *Specific energies* provided by the user and the content of the selected propellant tanks.
 - By choosing *Gurney equation*, the induced velocities are scaled such that the highest velocity equals the initial fragment velocity gained by the cylindrical Gurney-equation, see [156]. The explosive mass is based on the content of the selected propellant tanks. In this case, the *Specific energies* themselves are ignored since the information about the energy of the explosion is stored in the Gurney-constant, which has to be provided by the user. This constant is usually very close to $1/3$ of the detonation velocity and depends on the exploding material.
- By activating the *Blast* module, a blast wave at each time step along the whole trajectory is simulated in order to evaluate the effect of it to the area around the launch pad. The peak overpressure generated by the shockwave decreases while moving away from the explosion point. The isobaric areas of the shock wave have the form of spheres and the user needs to provide the *Peak overpressure* threshold which shall be considered to affect the area around the vehicle. The *Incidence angle* is used to consider the effect of the overpressure on structure in case of a certain incidence angle of interest. At 0° incidence angle the blast effect on the structure is the strongest while the higher the blast overpressure, the higher the peak pressure experienced by the structure. [141]. Several *Peak overpressure formulations* are implemented in ASTOS, providing different models to compute the scaled distance between the explosion point and the point where the *Peak overpressure* is reached [140]. This distance is scaled with the energy of the explosion, which is computed from the residual propellant using the *Specific energies* provided by the user. If *Full_Energy_into_Blast* is set to *false* the total blast energy is reduced by 15% to account for the energy used for fragmentation of the launcher and acceleration of its fragments. On the other hand in case of explosion on ground the total energy is increased by 80% to consider risks and damages derived from the vibrations

ANALYSES

generated and transmitted through the ground, if *Onground_Explosion_Extra_Energy* is set to *true*.

Overpressure levels corresponding to different types of damage are provided in the following table [142]:

Table 5.2: Level of damage expected at different overpressure values

Overpressure [psi]	Expected Damage
0.04	Loud noise (143 dB); sonic boom glass failure
0.15	Typical pressure for glass failure
0.4	Limited minor structural damage
0.5-1.0	Windows usually shattered; some window frame damage
0.7	Minor damage to house structures
1.0	Partial demolition to houses; made inhabitable
1.0-2.0	Corrugated metal panels fail and buckle; housing wood panels blown in
1.0-8.0	Range for slight to serious laceration injuries from flying glass and other missiles
2.0	Partial collapse of walls and roofs of houses
2.0-3.0	Non-reinforced concrete or cinder block walls shattered
2.4-12.2	Range for 1-90% eardrum rupture among exposed populations
2.5	50% destruction of home brickwork
3.0	Steel frame buildings distorted and pulled away from foundation
5.0	Wooden utility poles snapped
5.0-7.0	Nearly complete destruction of houses
7.0	Loaded train cars overturned
9.0	Loaded train box cars demolished
10.0	Probably total building destruction
14.5-29.0	Range for 1-99% fatalities among exposed populations due to direct blast effects

- The *Specific_Energies* for each tank propellant have to be provided. They are used by the blast computation and also in case the induced velocities that are applied to the explosion fragments shall be scaled towards the propellant energy.

The central body used for all the computations is the one selected for the first phase of the scenario.

5.9.2 Output

Flight corridor

ANALYSES

The definition of the flight corridor can be found at [113] and is illustrated in the figure below:

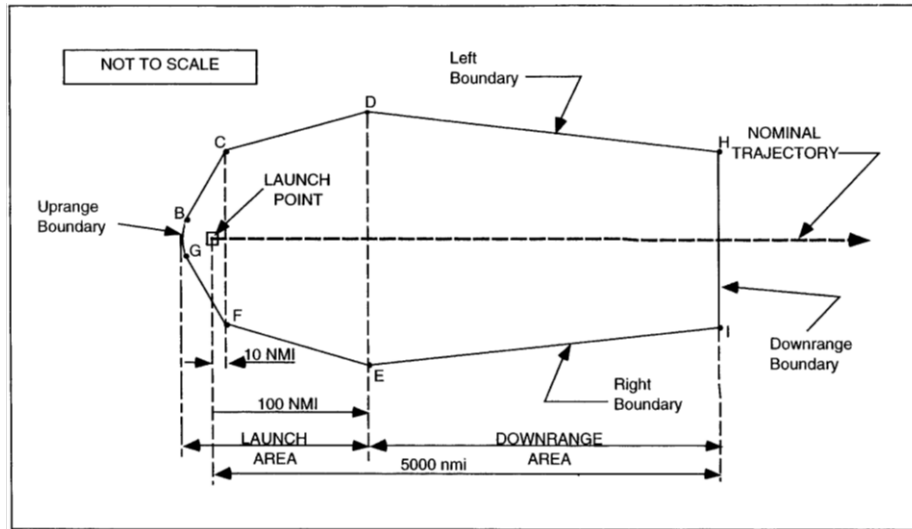


Figure A-2
Flight Corridor

Fig. 5.7: Flight corridor

In order to define the orientation of the flight corridor, the flight azimuth angle of the launcher is evaluated at a horizontal distance of 10000 m from the launch site, since the orientation of the nominal trajectory is not constant in time.

Depending on the selected *Launcher class*, the geometry of the flight corridor is determined by the following lengths:

Table 5.3: Flight corridor line segment lengths [km]

Launch class	D_{max}	CF	DE	HI
Small	2.23	73.06	218.30	3265.58
Medium	2.83	75.49	219.52	3265.58
Medium large	3.23	77.08	220.31	3265.58
Large	3.9624	80.00	221.78	3265.58

The flight corridor is output by this analysis as an auxiliary item. In this way its plotting along the vehicle trajectory is possible.

Impact risks

The launch safety analysis computes the *instantaneous impact point (IIP)* for a vehicle. The computation of the *IIP* is performed assuming no thrust and zero-drag for the vehicle by solely propagating an elliptical orbit starting at a certain point of the trajectory until the latter intersects the oblate Earth surface. The intersection point is the *IIP*. The *instantaneous impact point line (IIPPL)* is the line connecting all *IIP* generated for one vehicle ascent trajectory. For every time step of the trajectory propagation, the corresponding vehicle state is taken as an initial state for the computation of one *IIP*, whose location is output as auxiliary function.

ANALYSES

If the *Fragments* setting is enabled, the *casualty* and *fatality probabilities*, the *total casualty area*, *impact mass* and *impact kinetic energy* are computed and output as function of the time (auxiliary function). An explosion is simulated at each time step in order to estimate the risk as a function of time. The current position and velocity and the dry mass of the launcher (assumed spherical in shape and made of aluminum 7075) are used as an initial condition for the explosion. Each fragment is propagated until it demises or impacts the Earth surface. The *surviving objects* at each time point are collected in an auxiliary item, which can be visualized in a map plot.

The propagation of the fragments is performed in the same way as described Section 5.3. In case the launcher trajectory reaches a perigee altitude higher than the value provided in the GUI (see Section 5.9.1) then the explosion of the vehicle is not triggered any more since for this (or higher) orbit perigees, the explosion fragments are assumed not to re-enter.

Blast envelope

If the *Blast* setting is enabled, the software provides as output the blast radius computed at each point along the trajectory as long as the peak overpressure sphere intersects the Earth's surface.

The blast radius decreases along the trajectory, because the launcher continuously burns propellant mass and thus less potential energy is available on board to be transformed into blast energy. The envelope of the isobaric spheres projected on the Earth surface is provided as an auxiliary item to the user and defines the area where the peak overpressure is equal or greater than the user provided value. The Earth surface altitude is assumed to be 0 meters above sea level as a best guess since the surface altitude changes along the vehicle trajectory and depending if an ascent or a descent scenario is considered.

5.10 Link Budget Analysis

A link budget accounts for of all gains and losses in the transfer of an electromagnetic signal from a source (**transmitter**) to a target (**receiver**). All contributions that increase the transmitted signal power are defined as gain, e.g. antennas as the most important example. On the other hand, all contributions that decrease the transmitted signal power are defined as loss. Many different types of losses exist, some of them taking place in the transmitter or receiver (e.g. thermal noise, cable losses, etc), while others being caused by the propagation of the signal through a medium (e.g. free space, atmospheric attenuation, etc).

A simple link budget equation can be expressed as follows:

$$\text{Received Power} = \text{Transmitted Power} + \text{Gains} - \text{Losses}.$$

The main objective of the *Link Budget* analysis is to compute the signal power reaching a receiver, which is a measure for the signal quality after propagation. The following figures of merit are computed within the analysis in order to evaluate the performance: the Signal-to-Noise ratio (S/N or SNR), the Carrier-to-Noise ratio (C/N₀ or CNR) and the Bit Error Rate (BER). The link budget analysis can be subdivided into three main blocks:

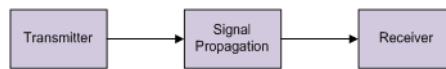


Fig. 5.8: Link budget functional blocks

Transmitter and receivers have already been discussed in Section 3.4.7. This chapter shall focus on the signal propagation and on the performance assessment.

5.10.1 Configuration

The *Link Budget* analysis contains the following settings:

- *Simulation output file* is common for several analyses and is explained in Chapter 5, subsection *Common analysis settings*
- *Write analysis data only* is common for several analyses and is explained in Chapter 5, subsection *Common analysis settings*
- *Transmitter* allows the user to select the model for the transmitter block in Fig. 5.8 and specifies the device transmitting the signal. The list of transmitters contains the transmitter models defined in *Vehicle Parts & Properties -> Sensors & Transmitters*.
- *Receiver* allows the user to select the model for the receiver block in Fig. 5.8. The list of receivers contains the receiver models defined in *Vehicle Parts & Properties -> Sensors & Transmitters*.

Note: At least one receiver and one transmitter model must be present in the scenario in order to perform a *Link Budget* analysis.

- *Minimum elevation at the receiver* allows to define a *custom* elevation (evaluated at the receiver) above which the link budget is computed (0.1° by *default*). This setting is considered only if the receiver is located at a ground station and does not affect the *Viewing Window* of the sensor (see Section 3.4).
- *Absorption* allows the user to set the atmospheric attenuation models to be considered during the propagation of the signal (see Section 5.10.2):
 - *Rain*
 - *Clouds and fog*
 - *Gaseous*
 - *Tropospheric scintillation*

By default, all absorption models are *disabled*. When *enabled*, no further user inputs are required.

- *Environment definition* enables the user to set the environment surrounding the sensors and to be considered during the propagation of the signal (see Section 5.10.2). The link budget integration is performed by discretizing the line-of-sight vector between transmitter and receiver. By *default*, the respectively "closest" environment for the transmitter and for the receiver is used for this integration. Thus for the first half of the distance on the line-

ANALYSES

of-sight vector the environment of the transmitter is used, while the environment of the receiver is used for the second half of the distance. In case of a *custom* environment, the environment resulting from the combination of the following models is used for the entire integration:

- *Celestial body* specifies the celestial body to be used (to be chosen from the list of celestial bodies defined in *Environment* -> *Celestial Bodies*, see also Section 2.1)
- *Atmosphere* specifies the atmosphere to be used (to be chosen from the list of atmospheres defined in *Environment* -> *Atmospheres*, see also Section 2.3)
- *Hydrosphere* specifies the hydrosphere to be used (to be chosen from the list of hydrospheres defined in *Environment* -> *Hydrospheres*, see also Section 2.4)
- *Compute dynamics MDEVs* allows the user to enable the computation of the MDEVs (Modified Allan DEVIation) [118] of relative range, range rate and acceleration, between transmitter and receiver (*disabled* by default). MDEVs is a measure for the stability of the link.
- *Phase measurements* allow the estimation of the carrier and code phase measurements and the modelling of the signal noise. The following error models can be selected within the *Phase measurements* setting (all *disabled* by default):
 - *White noise*
 - *Flicker noise*
 - *AM-PM conversion*
 - *Frequency filter response*
 - *DLL/PLL time lag*
 - *Time tagging calibration*
 - *Antenna phase pattern*

These error models and their individual settings are further explained in the next section.

5.10.2 Background

In aerospace scenarios once an electromagnetic wave leaves the transmitter, it usually travels a long distance before it reaches a receiver. Within the *Link Budget* analysis before propagating the signal, first the **visibility conditions** are checked. As explained in Section 3.4, the receiver must be within cone angle and viewing window of the transmitter and vice versa. Moreover, the line-of-sight between transmitter and receiver must not be obstructed by celestial bodies. If one of these conditions is not met, the whole main output of the analysis (e.g. G/T, EIRP and losses) is set to 0, and SNR and CNR are set to a default value of -200 dB/(Hz) for practical reasons (e.g. plotting); otherwise the signal is propagated.

During this phase, the signal is subject to many different sources of loss, where the main contribution is due to the propagation itself (free-space loss). Losses in the signal also occur due to the presence of an atmosphere. Especially in the troposphere, several effects have to be considered, e.g. the effect of meteorological phenomena such as rain, clouds and fog or effects

ANALYSES

of certain gases (oxygen and water vapor) on the signal. In the ionosphere the presence of free electrons influences the propagation of the signal. The full list of loss sources considered by ASTOS is provided below:

- Free-space path loss
- Rain loss
- Clouds and fog loss
- Gaseous absorption
- Tropospheric scintillation

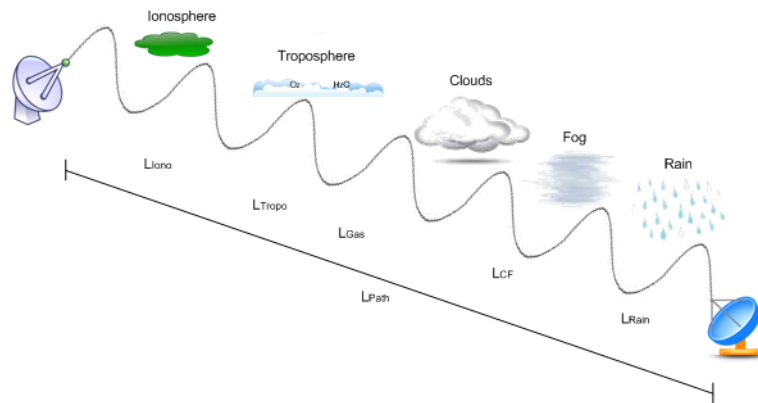


Fig. 5.9: Loss sources during propagation

The total loss [dB] then results from [87]:

$$L_{Tot} = L_{Path} + L_{Gas} + \sqrt{(L_{CF} + L_{Rain})^2 + L_{Tropo}^2} \quad (5.9)$$

Each atmospheric loss can be computed at different exceedance probabilities, defined by the selected hydrosphere (for more details see Section 2.4) Usually, the same numerical value should be used for all the exceedance probabilities. In case this probability is set to a value below 1.0% (which means that all the losses are computed considering values exceeded only for 1% of time of the year), the absorptions by gas, clouds and fog are computed at a percentage of 1%, since a large part of these attenuations is already included in the rain attenuation prediction for time percentages below 1%, as explained in [87].

Apart from the free-space path loss which is always computed, all other contributions to the total loss are optional. If they are used, an atmosphere model has to be defined in the scenario (since its physical properties are required during the computation) along with a hydrosphere model. The latter provides data on the distribution of water and on meteorological phenomena in the atmosphere. This data, except for the spectroscopic coefficients, is stored in external files and is summarized in the next figure.

Note: When the *Absorption* models are *enabled*, both *Atmosphere* and *Hydrosphere* models must be present in the scenario.

ANALYSES

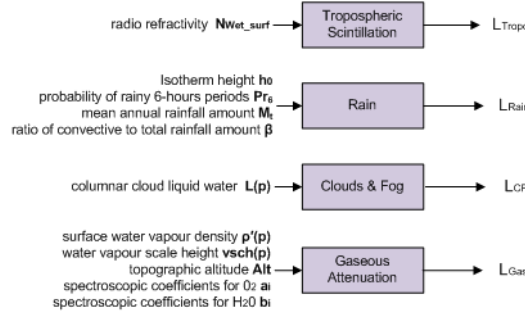


Fig. 5.10: External data required for the propagation losses

Atmospheric attenuation

Atmospheric attenuation along the line-of-sight are accurately determined by discretizing the line-of-sight vector starting from the transmitter towards the receiver. At each step, the local altitude is determined and if it is below 100 km (default limit for the Earth atmosphere), the attenuations are computed. For that purpose, also the atmosphere is discretized into horizontal layers (as shown in Fig. 5.11 [88]) in which the pressure, temperature and humidity are considered constant.

In order to calculate the total attenuation for a satellite link, it is not only necessary to know the specific attenuation at each point of the link but also the length of path that is affected by a specific attenuation. To derive the latter, also the ray bending (assuming a spherical Earth) needs to be considered.

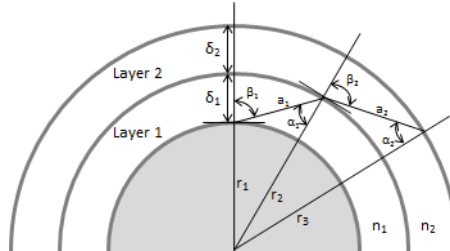


Fig. 5.11: Atmosphere discretization

The total atmospheric attenuation is given as:

$$L_{Tot} = \sum_{n=1}^k a_n \gamma_n \quad (5.10)$$

where γ_n is the total specific attenuation valid for the n -th layer of the atmosphere and a_n is the path length in layer n with the layer thickness δ_n :

$$\delta_i = 0.0001 e^{\left(\frac{i-1}{100}\right)} \quad (5.11)$$

The n -th path length can be determined using the following expressions:

$$a_n = -r_n \cos \beta_n + 0.5 \sqrt{4r_n^2 \cos^2 \beta_n + 8r_n \delta_n + 4\delta_n^2}; \quad \alpha_n = \pi - \arccos \left(\frac{-a_n^2 - 2r_n \delta_n - \delta_n^2}{2a_n r_n + 2a_n \delta_n} \right); \quad \beta_{n+1} = \arcsin \left(\frac{n_n}{n_{n+1}} \sin \alpha \right) \quad (5.12)$$

where n_n is the refractive index, α_n and β_n are the entry and exit incidence angles and r_n is the radius from the centre of the Earth to the beginning of layer n . The incidence angle at the next

ANALYSES

layer is calculated according to Snell's law while the refractive indices are calculated according to [86].

Free-space path loss

The free-space path loss is the loss in signal strength of an electromagnetic wave that occurs along a line-of-sight path through free-space (usually air), with no obstacles nearby to cause reflection or diffraction. It does not include device-specific factors such as antenna gains at the transmitter and receiver nor any losses associated with hardware imperfections. The expression for the free-space path loss in terms of dB is given by the expression:

$$L_{path}[\text{dB}] = 10 \log_{10} \left(\frac{4\pi r}{\lambda} \right) \quad (5.13)$$

where r is the distance between transmitter and receiver and λ the wavelength of the signal.

Rain loss

Rainfall represents an important climatic effect on a satellite link. Rain results in an attenuation of radio waves due to scattering and energy absorption. The attenuation effects increase with frequency. Thus, enough extra power must be transmitted to overcome the additional attenuation induced by rain in order to provide an adequate link availability.

The procedure to compute the rain loss is given by [87] for frequencies up to 55 GHz. The computation is a function of many parameters such as altitude above mean sea level of the Earth, elevation angle, latitude of receiver and frequency.

The rain-specific attenuation expressed in dB/km can be computed using the following expression:

$$\gamma_R = k R_p^\alpha \quad (5.14)$$

where the coefficients k and α are functions of the signal frequency and polarization and include the scattering effect, while R is the rain rate in mm/h.

The specific attenuation γ_R is multiplied by the slant-path length, by a vertical adjustment factor, and by a coefficient function related to the percentage of time that exceeds the average annual level of rain fall. The full detailed model behind the rain loss computation can be found in [85] and [87].

Clouds and fog loss

For clouds or fog consisting entirely of small droplets (generally with a diameter less than 0.01 cm), the Rayleigh approximation is valid for frequencies below 200 GHz and it is possible to express the attenuation in terms of the total water content per unit volume. Thus the specific attenuation in dB/km within a cloud or fog can be written as [91]:

$$\gamma_{CF} = k_{CF} M \quad (5.15)$$

where k_{CF} is the specific attenuation coefficient and M is the liquid water density in the cloud/fog.

The value of k_{CF} is determined from mathematical model based on Rayleigh scattering, which uses a double-Debye model for the dielectric permittivity of water. To obtain the attenuation due to clouds for a given probability value, the statistical data of the total columnar content

ANALYSES

of liquid water is required (provided by the hydrosphere model) and altitude/ elevation angle must be taken into account. The detailed mathematical model can be found in [85] and [91].

Gaseous absorption

Signal loss also can occur due to absorption by atmospheric gases such as oxygen and water vapor. This kind of loss depends on the signal frequency, elevation angle, altitude above sea level, and absolute humidity. Its importance increases with frequencies above 10 GHz, especially for low elevation angles. The specific gaseous attenuation (for frequencies below 300 GHz) in dB/km is given by [88]:

$$\gamma_G = \gamma_O + \gamma_W = 0.182N''(f, p) \quad (5.16)$$

where γ_O and γ_W are the specific attenuations due to dry air and water vapour, respectively, f is the frequency and N is the imaginary part of the frequency and pressure-dependent complex refractivity (refer to [88] for a detailed model description).

Tropospheric scintillation

The troposphere is constantly in motion as energy from the sun heats the surface of the Earth and the resultant convective activity agitates this layer. The resulting turbulent mixture of air mass causes variations of the refractive index of air. The tropospheric scintillation is the occurrence of small-scale fluctuations of the refractive index due to turbulence and produces random fades and enhancements of the received signal amplitude [115].

The magnitude of tropospheric scintillations depends on the magnitude and structure of the refractive index variations. It increases with frequency and with the path length through a medium, and decreases with decreasing antenna beamwidth because of aperture averaging. A general technique for predicting the cumulative distribution of tropospheric scintillation at elevation angles $> 4^\circ$ is adopted here. It is based on monthly or longer averages of temperature and relative humidity, and reflects the specific climatic conditions of a site. The procedure is recommended for signal frequencies of 7-50 GHz. The detailed mathematical model can be found in [85] and [87].

Phase measurements

The *Phase measurements* options allow an estimation of phase time measurement errors. Phase measurements compare the phase of the signal received by a device to the phase of the device response signal (reproduced signal). Assuming that an accurate calibration has been performed, the difference in phase between the two signals (known as phase shift) is a result of the electrical characteristics of the device itself.

Two noise models are available in this analysis: *White noise* and *Flicker noise*. White noise is random noise with constant power spectral density, while flicker noise has a power spectral density inversely proportional to the signal frequency. Currently, noise data can only be *Defined by* via a user-defined *File* and is added to the total error (in seconds). The flicker noise has to be provided in seconds and a *Code scaling constant* (i.e. scaling factor) can be applied to the code flicker noise before adding it to the total error. White noise has to be provided normalized and it is added in the following way respectively to the code and carrier errors:

ANALYSES

$$E_{wn}(t) = \frac{N_w}{0.6f_b} \sqrt{\frac{0.5}{CNR}} \quad (5.17)$$

$$E_{wn}(t) = \frac{N_w}{0.6\pi f} \sqrt{\frac{0.5}{CNR}} \quad (5.18)$$

where N_w is the white noise (extracted from file), f is the signal frequency, f_b is the *Code chip rate* (in Chips/s).

The *AM-PM conversion* error is a measure of the amount of undesired phase deviation (PM) that is caused by amplitude variations (AM) of the system. AM-PM conversion is defined as the change in output phase for a 1-dB increment in the input power of an amplifier. The conversion error can be estimated with the following formula:

$$E_{AM-PM} = \frac{k \cdot s}{c} f \cdot range \quad (5.19)$$

where c is the speed of light, s is a *Scaling constant*, k is a *Conversion factor* and *range* is the (dynamic) distance between transmitter and receiver.

Bandpass filters are usually symmetric around the center frequency, but a close look at the characteristics around the center frequency shows that a real filter is not perfectly symmetric. This leads to a *Frequency filter response* error, since the incoming signal frequency is not constant but dynamically changes due to the Doppler effect. This error can be estimated using the following relation:

$$E_{f_{rf}} = s \cdot \frac{l_{slope}}{c} f \cdot rangerate \quad (5.20)$$

where c is the speed of light, s is a *Scaling constant*, l_{slope} is the *Linear slope* of the filter response (first order) and *rangerate* is the dynamic relative velocity between transmitter and receiver.

The code-phase and carrier-phase time measurements (performed by delay-locked-loops (DLL) and phase-locked-loops (PLL) respectively) lag behind the incoming signal, if the tracking loop needs to adjust to significant variations. These variations are related to changes of the range rate and cause a *DLL/PLL time lag* error given by:

$$E_{DLL-PLL} = s \cdot \frac{acc}{cf_n^2} \quad (5.21)$$

where c is the speed of light, s is a *Scaling constant*, f_n is the *Natural frequency* of the tracking loop and *acc* is the dynamic relative acceleration between transmitter and receiver.

A Doppler shift can lead to a *Time tagging calibration* error in the phase measurement, if the measurement epochs onboard the satellite and at the ground station are not perfectly synchronised. In first order, neglecting relativistic effects, the phase error due to time-tagging errors is given by:

$$E_{time-tag} = \frac{\delta t}{c} rangerate \quad (5.22)$$

where c is the speed of light, δt is the *Timestamp desynchronization* and *rangerate* is the dynamic relative velocity between transmitter and receiver.

ANALYSES

Antenna phase pattern errors which arise from hardware imperfections can be estimated using the following formula:

$$E_{ant} = s \cdot \sin(\phi \cdot k_{arg}) \quad (5.23)$$

where s is a *Scaling constant*, ϕ is the off-boresight angle and k_{arg} is the *Sine argument constant*.

5.10.3 Output

The following three main figures of merit are used to evaluate the performance of a link budget:

- Signal-to-Noise Ratio (**SNR**)
- Carrier-to-Noise Ratio (**CNR**)
- Bit Error Rate (**BER**)

Note: A **BER** can only be evaluated if a *Maximum data rate* has been specified for the transmitter model.

SNR

A schematic overview of the **SNR** determination, the **SNR** being also input for the computation of **CNR** and **BER**, is shown in Fig. 5.12.

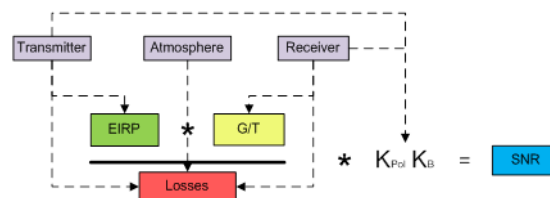


Fig. 5.12: **SNR** computation diagram

The EIRP (see the transmitter chapter of Section 3.4.7) represents the main positive contribution to the **SNR** and depends on the transmitter characteristics. On the other hand the G/T computation (see the receiver chapter of Section 3.4.7) contains both a positive contribution (due to the receiver gain) and a negative contribution (due to the noise at the receiver). The main negative contribution is caused by the propagation losses. Inputs for the **SNR** computation are the atmosphere and hydrosphere models, the positions of receiver and transmitter and the signal characteristics.

The coefficient K_{Pol} (values ranging from 0 and 1) accounts for polarization mismatches. The polarization of the receiver has to be compatible with the polarization of the incoming signal. If the two polarizations are of the same type, then there is no loss in the transmission. However, if the polarization types are different, signal degradation might occur. The signal quality is measured on a scale from 0 to 1 where 0 represents no match in the polarization of receiver and incoming signal (and thus no detected signal) and 1 represents the ideal case without losses.

ANALYSES

In case the transmitting antenna is linearly polarized, it is important that also the receiving antenna has the same polarization type. Otherwise the received signal strength might be greatly reduced. Therefore, only horizontal polarizations should be used in combination (analogously for vertical polarizations). Partial matchings cause a loss of signal strength, although not as much as in the case of a complete mismatch. Transmitters mounted on moving vehicles usually make use of circularly polarized antennas such that there is never a complete mismatch with signals from other sources.

Table 5.4 lists K_{Pol} values depending on the geometry of the problem. In case the polarization of one antenna is linear (e.g. vertical, horizontal, etc.) and horizontal or vertical for the other, the coefficient needs to be computed taking into account the specific geometric configuration (analogously in case of elliptical polarizations).

Table 5.4: Polarization mismatch

Transmitter Polarization	Receiver Polarization	K_{Pol}
Horizontal	Vertical	0
Vertical	Horizontal	0
Linear	Circular	1
Circular	Linear	0.5
Circular	Circular	1
Horizontal/vertical	Linear	computed
Linear	Horizontal/Vertical	computed

The coefficient K_B accounts for bandwidth mismatches. If the receiver bandwidth is manually defined by the user (i.e. *Auto-scale* is not used in the receiver model) and it does not match the bandwidth of the transmitter, a loss in the transmission occurs which is represented using K_B . It is defined as a fraction (between 0.0 and 1.0) of transmitted power which is contained within the receiver's bandwidth. The transmitted power spectrum is considered flat over the full bandwidth.

The **SNR** is the ratio of the received power to the noise power at the receiver in a given bandwidth and it is usually expressed in dB (a value expressed in dB is just $10 \log_{10}[\text{value}]$). If values are linked to a unit, they are attached to the dB notation (e.g. dBW). According to [92], the received power P_r and the noise power N_0 at the receiver can be expressed as:

$$P_r = \frac{EIRP \cdot G_r}{L_{Tot}} K_{Pol} K_B; \quad N_0 = K T_{Sys} B \quad (5.24)$$

where G_r is the gain of the receiver, T_{Sys} is the receiver noise temperature (see Section 3.4.7), L_{Tot} the total signal loss, B the receiver noise bandwidth and K the Boltzmann constant. Since the SNR is defined as the ratio between the received and noise power at the receiver, the following mathematical relation can be derived (as a linear power ratio, not as logarithmic decibels):

$$SNR = \frac{P_r}{N_0} = \frac{EIRP \cdot G / T}{L_{Tot} \cdot KB} K_{Pol} K_B \quad (5.25)$$

CNR

ANALYSES

The **CNR** is defined as the sum of the received power and of the noise power at the receiver per unit bandwidth and it is usually expressed in dB-Hz. The relation between SNR (in dB) and CNR (in dB-Hz) is given by:

$$CNR = SNR + 10\log_{10}B \quad (5.26)$$

BER

The **BER** is defined as the number of received bits of a data stream over a communication channel that have been altered (due to noise, interference, distortion or bit synchronization errors) divided by the total number of transferred bits during a certain time interval. BER is a dimensionless performance measure, often expressed as a percentage. A BER worst case scenario is a channel of random data, where noise totally dominates the signal. This results in a transmission BER of 50%. In a noisy channel, the BER is often expressed as a function of the normalized CNR (i.e. E_b/N_0 which is the energy per bit to noise power spectral density ratio) and is given by:

$$\frac{E_b}{N_0} = SNR \frac{B}{f_b} \quad (5.27)$$

where B is the receiver noise bandwidth and f_b is the channel data rate.

The equation which relates the SNR per bit and the BER depends on the modulation type used and can be found in [85].

Given the SNR , the channel bandwidth B and the channel capacity R_{max} , which is the theoretical tightest upper bound on the information rate of clean (low bit error rate) data that can be sent with a given average signal power through an analog communication channel subject to additive white Gaussian noise of power, the Shannon-Hartley theorem gives a maximum rate for the transmission of information R_{max} [107]:

$$R_{max} = B\log_2(1 + SNR) \quad (5.28)$$

This maximum theoretical rate is considered in case the user specifies a larger *Maximum data rate* for the transmitter.

5.11 Load Case

The *Load Case* is identifying the dimensioning load cases for each substructure of a component based on an existing trajectory. Dimensioning load cases will be used for the structural mass estimation as input to *ODIN* regression tables and will be exported to *ODIN*.

Starting a design optimization task the load cases are not known, but the initial mass estimation will depend on them. This dependency is solved by using the minimum load provided in the regression tables. This ensures to obtain a feasible trajectory during the initial guess. The regression tables provide masses for a range of loads, each load combination consists of two load pairs (pressure and flux) one relevant for strength and one relevant for buckling.

ANALYSES

Load cases changes with changing trajectories during the optimization process. It is in the responsibility of the user to interrupt the optimization process, to perform a load case analysis and to continue the optimization process with an updated load case file. Especially the user should verify at the end of the design process that the assumed loads are still valid. If they are no more valid, this will result in a non-acceptable mass error and the calculation must be repeated with an update of trajectory and loads.

ASTOS computes for purpose of a beam approximation the load distribution at predefined locations, so called cutting planes. Such cutting planes are model defined and are available at connection planes of structural elements like tanks, inter-stages, aft/fwd-skirt, strap-on connection. ASTOS determines for each cutting plane of each substructure one dimensioning load case for strength and one dimensioning load case for buckling.

5.11.1 Configuration

The *Load Case* analysis contains the following settings:

- *Struct file prefix* it contains the prefix for the name of the files generated during the analysis; the files includes the output of the loads as function of the vehicle main axis stations.
- *Max Q-Alpha* if specified, the loads are computed once more at the maximum dynamic pressure point, this time not applying the real normal forces but the real normal force divided by the current dynamic pressure and current total angle of attack and then multiplied by this given product of dynamic pressure and total angle of attack (max q-alpha).
- *Generate ODIN export file*, if selected allows the generation of an export file to be used as input by *ODIN*.
- *Automatically run ODIN after analysis*, if selected the *ODIN* software will be run after completing the analysis to check the correctness of the mass regression tables.
- *Automatically run missile DATCOM after analysis*, if selected the *Missile DATCOM* software will be run after completing the analysis.
- *ODIN installation directory* allows the user to specify the directory, where *ODIN* is located.

Additionally the **Open ODIN GUI** button opens the structural analysis software loading the input file generated during the analysis.

Note: The Max Q-Alpha option will effect only the loadcases and masses of cylindrical and conic substructures! For fairings and bulkheads the mass estimation is driven by the relative pressure, which is not affected by the *Max Q-Alpha* setting.

5.11.2 Background

Line loads at cutting planes

ANALYSES

ASTOS computes for purpose of a beam approximation the load distribution at predefined locations, so called cutting planes. Such cutting planes are model defined, i.e. *ASTOS* partitions a launcher into the following components:

- Fairing,
- Payload,
- Payload Adapter,
- Interstage,
- For each stage:
 - Forward skirt,
 - Forward bulkhead tank,
 - Cylinder tank,
 - Propellant in tank (without sloshing),
 - Intertank,
 - Aft bulkhead tank,
 - Aft skirt,
 - Thrust frame (liquid propellant only),
 - Engine with nozzle.

ASTOS computes:

- Point masses
 - at the centre of mass of each component,
 - in case of propulsion considering the filling ratio compressed at the bottom of the tank (accelerated case);
- Thrust forces at the nozzle exit area or booster connection point considering a TVC angle required to compensate wind loads assuming a control requirement of 1.5;
- Axial and normal aerodynamic force at highest point accepting a higher bending moment and considering an angle of attack due to wind loads specified by the user;
- Global acceleration.

Based on the above given information a distribution of forces and bending moments and finally line loads along the x-axis can be computed and at the interfaces between the defined components cutting planes can be introduced.

Determination of dimensioning load case

The determination of the dimensioning load case requires the steps as depicted in Fig. 5.13 and can be summarized as follows:

- Definition of load phases defined by ground and flight phases.
- Computation of maximum load case for strength and for buckling for each phase.

ANALYSES

- From all phases the maximum load case for buckling and separately the maximum load case for strength are selected as dimensioning load cases.

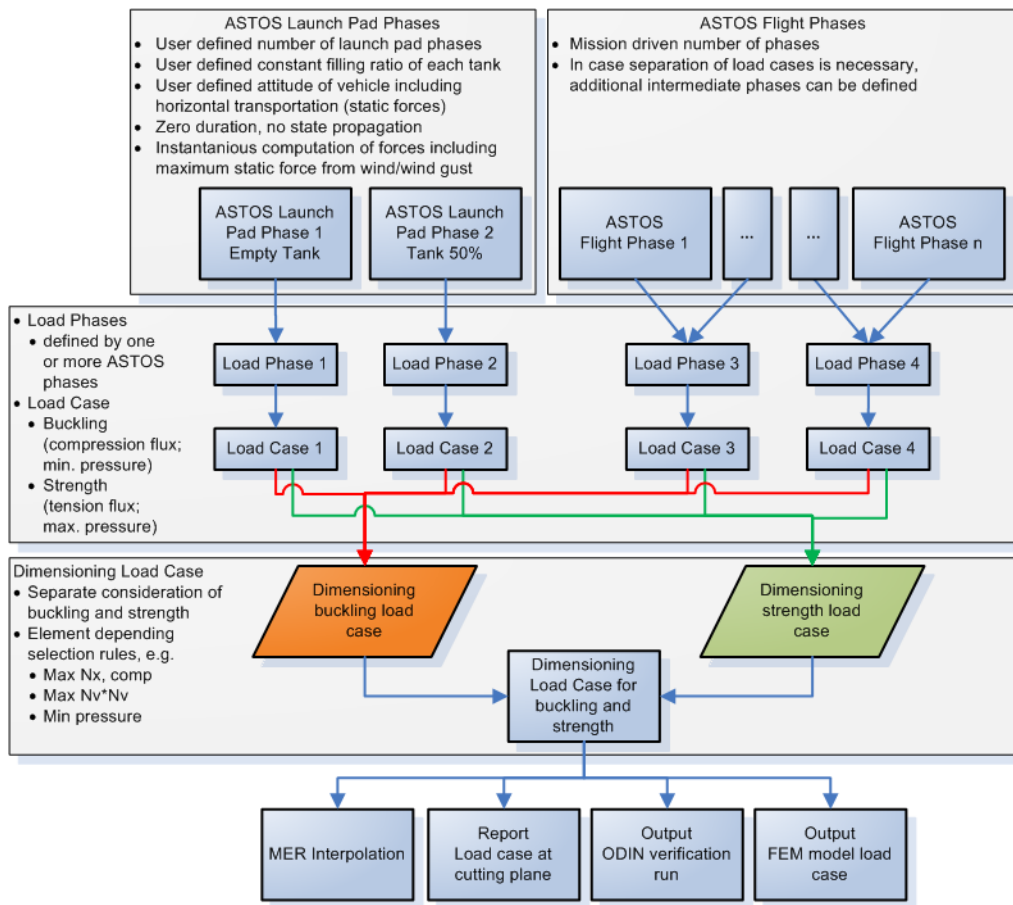


Fig. 5.13: Scheme for load case determination.

The following details need to be considered during the definition of load phases:

- Load phases separate the loads cases.
- A load phase is defined by one or more ASTOS phases. If necessary an additional ASTOS phase has to be defined (e.g. Phase on launchpad to fill the liquid tanks).
- For each load phase and each strength and buckling load case couples of extreme flux and pressure are selected independent from time.
- Compression and tension fluxes are computed as negative value, hence the maximum of the absolute values (the minimum fluxes) are selected.
- In case minimum pressure is constant for the whole flight, it should be possible to determine a single maximum compression flux for the whole flight.
- In case of a constant maximum pressure for the whole flight, it should be possible to determine a single maximum tension flux for the whole flight. Otherwise the variable hydrostatic pressure might create additional load cases which could be dimensioning.
- The dimensioning load case of a ground ignited solid booster is normally defined at ignition and can be frozen accordingly. In case of vertical ascent no buckling load case occurs.

ANALYSES

- The relative inner pressure of a 2nd stage solid motor is Zero before ignition.

Instead during the computation of pressure, these details apply:

- The pressure is computed as sum of static and hydrostatic pressure
- The static pressure shall be defined as constant value or as table data as function of time. E.g. the relative static pressure might be around 5 bar during ignition.
- The hydrostatic pressure is computed as function of total acceleration (no split into axial and lateral acceleration)
- The maximum pressure is defined either by the maximum sum of static and hydrostatic pressure or by the definition of an optional maximum pressure value (avoids maximum determination)
- Two pressure loads are required in case of common bulkhead tanks, i.e. either two user defined values or the positive and negative pressure difference between both tanks.
- The minimum tank pressure shall be larger than the turbo pump pressure of the propulsion system.

Selection of dimension load case

The selection of the dimension load case from the several load phases is performed as follows.

- Cylinder, Cone, YRing:
 - Strength
The dimensioning load case is defined by the load case with the maximum value of n_v^2 .
 - Buckling
The dimensioning load case is defined by the load case with the maximum negative flux value, e.g. the maximum compression flux.
- Bulkhead
The dimensioning load case is defined by the load case with the minimum pressure.
- Strut Cone
The dimensioning load case is defined by the load case with the maximum negative flux value, e.g. the maximum compression flux.

5.12 Navigation Analysis

A satellite navigation system is a system of satellites which provides autonomous geospatial positioning with global coverage. It allows receivers to determine their location (longitude, latitude, and altitude) to within a few meters of accuracy using time signals transmitted along a line-of-sight from satellites. A satellite navigation system with global coverage may be termed a global navigation satellite system (GNSS). As of beginning of 2014, only the American GPS and the Russian GLONASS are fully globally operational GNSSs. China is in the process of

ANALYSES

expanding its regional Beidou navigation system into the global Compass navigation system by 2020. The European Galileo positioning system is a GNSS in initial deployment phase. India and Japan are also building their regional navigational systems.

The *Navigation* analysis determines the accuracy that could be reached if a satellite was navigated only by means of GNSS receivers. Spacecraft are now beginning to use GNSS as a navigational tool. The use of a GNSS receiver on a spacecraft allows "precise" orbit determination without the need of ground segment intervention which is expensive and time-consuming.

The two main drivers for GNSS use on a spacecraft are coverage and accuracy. Since GNSS were designed to be used on Earth, they guarantee coverage on the Earth surface, but not in space. Depending on the specific trajectory and on the GNSS signals used, the coverage (and indeed the accuracy), might not be sufficient over the entire mission. Similarly, the required accuracy can restrict the use of autonomous GNSS navigation. The *Navigation* analysis allows an estimation of the coverage and the feasible accuracy of the navigation solution along a trajectory.

5.12.1 Configuration

The *Navigation* analysis contains the following settings:

- *Simulation output file* is common for several analyses and is explained in Chapter 5, subsection *Common analysis settings*
- *Write analysis data only* is common for several analyses and is explained in Chapter 5, subsection *Common analysis settings*
- *Write GNSS satellites ephemerides* allows the user to add the propagated trajectories for each GNSS satellite to the analysis output. This is especially useful when constellations are locally defined (see below), as the trajectory data is not present in the analysis output file otherwise.
- *Receivers* contains a table with all GNSS sensor instances from the list of those GNSS receiver instances defined in *Vehicle Parts & Properties->Sensors & Transmitters* that have already been assigned to a vehicle or ground stations. Multiple GNSS receivers mounted on the same vehicle/ground station can be selected.

Note: At least one receiver model (subtype *GNSS*) must be attached to a vehicle or a ground station in the scenario in order that a *Navigation* analysis can be conducted.

- *UERE budget* defines a set of constant errors to be used in the computation of the user equivalent range error (see Section 5.12.2):
 - *Broadcast clock*
 - *Broadcast ephemeris*
 - *Broadcast ionospheric delay*
 - *Broadcast tropospheric delay*

ANALYSES

- *Multipath*
- *Other*

For each error source a *default* value is given in the GUI and can be modified if desired.

- In the *Absorption* pane the atmospheric attenuation models are set which are to be considered during the propagation of the signal (as explained in Section 5.10.2).
- The *Environment definition* pane allows configuration of the environment (*Central body*, *Atmosphere* and *Hydrosphere*) for the propagation of the signal by analogy to the *Link Budget* analysis Section 5.10.2. In addition, the celestial body is used also for the propagation of the GNSS satellites (as explained in Section 5.12.2). If *default* settings are selected, the environment defined for the receiver(s) is used.
- In the *Constellations* settings are specified the GNSS constellations which are to be used for the analysis. For each constellation, the setup is determined via a radio button:
 - *Define locally* means that the initial states of the satellites have to be provided by the user since their orbits are to be propagated by the analysis. The following properties have to be specified:
 - An *ID* has to be specified for the constellation
 - The initial state can be defined by specifying either *Keplerian* elements (see Section 4.2.3.2.1) or by using a *TLE* catalog file (see Section 4.2.3.2.12).
In the case of *Keplerian* elements the orbital elements have to be provided individually for each satellite in a table (*Mean Anomaly* and *True Anomaly* are both supported). Possible *Coordinate systems* are inertial *J2000* and inertial *TOD* (see Section 9.3). The *Reference Epoch* is required as Julian Date (UTC), see Section 4.1. In both cases of initial state definition the tables requires additional input for the satellite identifiers and a selection of the *Transmitter* model (from a drop-down menu in the respective cell) which is used on the satellite.
 - If *Use existing* is specified, then a constellation already defined in *Vehicle Parts & POIs Definition* is used for the analysis. In this case, all GNSS transmitters mounted on the constellation satellites and the simulated satellite trajectories from the scenario are used in the analysis.

Multiple constellations can be introduced or deleted from the analysis using the **Add** and **Remove** buttons.

Note: At least one transmitter model (subtype *GNSS*) must be defined in the scenario for a *Navigation* analysis.

5.12.2 Background

The basic idea behind GNSS is that receivers are able to use incoming signals from GNSS satellites to compute the distance to each of them (ρ_j). Once a signal has been tracked, a receiver determines the signal transit time (Δt) between the time the signal was sent by a GNSS satellite and the time it was received. In case at least four signals have been tracked, then the absolute position (x, y, z) can uniquely be determined, since the positions of the GNSS satellites (x_j, y_j, z_j) are known as well. Three signals are required to determine the three components of the position vector and a fourth one to estimate the clock synchronization error (δt), which is common for all satellites belonging to the same constellations. In case of a multi-constellation solution - i.e. using signals from more than one constellation - one additional satellite per constellation is required, since a new unknown is introduced (e.g. for correcting the position obtained with GPS and GLONASS, five signals are required).

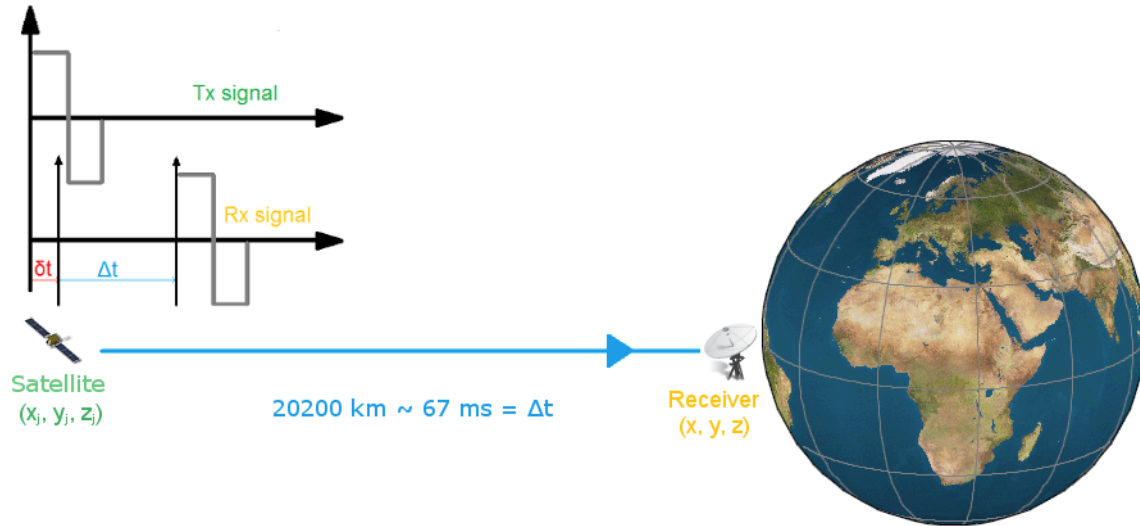


Fig. 5.14: Range measurement: signal transit time and clock synchronization error

Range measurements and GNSS satellite positions

The distance ρ_j from the vehicle to a GNSS satellite is given by the following expression [105]:

$$\rho_j = \sqrt{(x_j - x)^2 + (y_j - y)^2 + (z_j - z)^2} - c \cdot \delta t = c \cdot \Delta t \quad (5.29)$$

This range equation is applied to each available GNSS signal which results in a system with four unknowns (for mono-constellation solution) and at least four equations.

The position of the GNSS satellites which is required for the solution is usually available in real GNSS receivers; their ephemerides are usually known in order to use this information already during acquisition. Moreover, ephemeris corrections are also embedded in the navigation data.

Consequently, similar information has to be provided for the analysis in *ASTOS*. This information is directly derived from the trajectories of the satellites in the defined *Constellations*. In case of locally defined constellations, the initial state is propagated assuming ideal

ANALYSES

unperturbed Keplerian orbits. In case that existing constellation(s) from the scenario are used, the trajectory is computed as specified for the template satellites.

Visibility conditions and signal detection

Once positions and velocities of the GNSS satellites have been computed, a link budget is performed between each GNSS transmitter and GNSS receiver. If there is mutual visibility (transmitter and receiver "see" each other, as explained in Section 5.10.2), the CNR is computed in order to test if the GNSS satellite can be tracked. The CNR threshold is determined from the settings of the GNSS receiver (see Section 3.4.7.4.2). It can be either a fixed predefined value (if a *minimum* C/N_0 was specified) or it can be computed from the PLL/FLL characteristics. If the C/N_0 is above the threshold, the satellite is tracked and its signal is used to determine the GNSS satellites geometry strength (DOPs) and the range error (UERE).

DOP determination

Dilution of precision (DOP) is a measure of the geometry quality of the GNSS satellites with respect to the receiver. The idea behind the DOP computation is that the impact of the range error on the final estimated position depends on the displacement of the satellites. When visible navigation satellites are "close" together in the sky (from a receiver point of view), the geometry is said to be weak and the DOP values are high; when "far apart", the geometry is strong and the DOP values are low.

For illustration, consider two overlapping rings of different centres, each represents range measurement from a GNSS satellite to a receivers. If they intersect at right angles (case A in Fig. 5.15), the overlapping area is much smaller than if they intersect nearly parallel (case B). Thus a low DOP value represents a better positional precision due to a wider angular separation between the satellites used to determine the position.

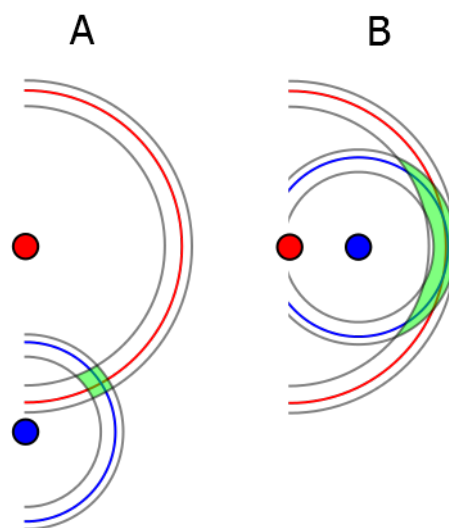


Fig. 5.15: Good (A) and poor (B) geometry

In order to determine the DOP values the following matrices are computed [105]:

ANALYSES

$$A = \begin{bmatrix} \frac{x-x_1}{R_1} & \frac{y-y_1}{R_1} & \frac{z-z_1}{R_1} & 1 \\ \frac{x-x_2}{R_2} & \frac{y-y_2}{R_2} & \frac{z-z_2}{R_2} & 1 \\ \vdots & \vdots & \vdots & \vdots \\ \frac{x-x_n}{R_n} & \frac{y-y_n}{R_n} & \frac{z-z_n}{R_n} & 1 \end{bmatrix}, \quad Q = (A^T A)^{-1} = \begin{bmatrix} D_{11} & D_{12} & D_{13} & D_{14} \\ D_{21} & D_{22} & D_{23} & D_{24} \\ D_{31} & D_{32} & D_{33} & D_{34} \\ D_{41} & D_{42} & D_{43} & D_{44} \end{bmatrix} \quad (5.30)$$

where n is the number of tracked satellites (at least four), R_i is the norm of the relative position vector, x, y, z denote the position vector components of the receiver and x_i, y_i, z_i denote the position vector components of the i -th GNSS satellite (as already shown in Fig. 5.14). Then DOP values can be computed from:

$$\begin{aligned} HDOP &= \sqrt{D_{11} + D_{22}} \\ VDOP &= \sqrt{D_{33}} \\ PDOP &= \sqrt{D_{11} + D_{22} + D_{33}} \\ GDOP &= \sqrt{D_{11} + D_{22} + D_{33} + D_{44}} \end{aligned} \quad (5.31)$$

UERE determination

Various error sources affect the (pseudo-)range measurement (see Eq. 5.29). Each error contribution to the range errors can typically be modelled by an independent and identically distributed Gaussian random variable with zero mean and with a variance σ_j . Under these hypotheses, the standard deviation of the total error is the so-called user equivalent range error (UERE) σ_{UERE} [105]:

$$\sigma_{UERE} = \sqrt{\sum_j \sigma_j^2} \quad (5.32)$$

In the following table a list of the typical error sources and the corresponding values of the variance are provided [109]:

Table 5.5: Error sources

Segment	Error Source	σ [m]
Space	Satellite clock stability	3.0
	Satellite perturbations	1.0
	Other	0.5
Control	Ephemeris prediction	4.2
	Other	0.9
User	Ionospheric delay	5.0
	Tropospheric delay	1.5
	Receiver noise	1.5
	Multipath	2.5
	Other	0.5
UERE	Total (RSS)	8.0

The UERE is the total of the contribution of all the sources. The values shown in the previous table have to be considered as residual contribution after a step in which all the predictable

ANALYSES

part of the error has been removed. All these errors can be provided by the user in the *UERE budget*, where the following entries are available:

- *Broadcast clock* for the satellite clock stability errors of the space segment
- *Broadcast ephemeris* for the ephemeris prediction errors in the control segment
- *Broadcast ionospheric delay* for respective errors in the user segment
- *Broadcast tropospheric delay* for respective errors in the user segment
- *Multipath* for respective errors in the user segment
- *Other* for various remaining errors from different segments

In addition to the previous list, also the receiver noise is included in the UERE. It is computed as explained in Section 3.4.7.4.2, if the *carrier and code loops* characteristics have been provided, otherwise it is set to 1.5 m according to Table 5.5 .

5.12.3 Output

The main results which are output of a navigation analysis are summarized below:

- The number of *visible satellites*, which is the number of GNSS satellites visible for at least one GNSS receiver.
- The number of *trackable satellites*, which is the number of visible GNSS satellites that are able to see at least one of the GNSS receiver.
- The number of *tracked satellites*, which is the number of trackable satellites effectively tracked, i.e. where also the CNR exceeds the minimum threshold required for tracking.
- The ideal *range* and its *time derivative* to each GNSS satellite. Whenever a GNSS satellite cannot be tracked, these two outputs are set to 0.0, since a range measurement cannot be obtained by the receiver, even though the C/N_0 could be evaluated.
- The *outage duration* keeps track of the time intervals, during which an insufficient number of satellites can be tracked for a position solution, i.e. four satellites for a mono-constellation solution plus one more for each additional constellation.
- The *DOP* values (HDOP, VDOP, PDOP, GDOP) computed at each time step.
- The 1σ *positioning errors* computed from the UERE settings and DOP values according to the following relation:

$$\sigma_{DOP} = DOP \cdot \sigma_{UERE} \quad (5.33)$$

Note: All receivers mounted on a vehicle are implicitly treated as "cooperative group", e.g. as a group of antennas with a common processing module.

Interpretation of DOP values

ANALYSES

The smaller the DOP factor, the better the geometry. For instance, with only four received signals the position DOP factor (PDOP) is at least 2.0. However, it can decrease down to 1.0 if more signals are received. A "classification" of the DOP values is shown in Table 5.6.

Table 5.6: Typical DOP values

DOP Value	Rating
1	Ideal
1-2	Excellent
2-5	Good
5-10	Moderate
10-20	Fair
>20	Poor

5.13 Operational Lifetime Prediction Analysis

The *Operational Lifetime Prediction* analysis estimates the duration of the operational phase for a satellite depending on its operational orbit. The estimation is based on the available fuel on board the satellite at the beginning of the mission operational phase and on the fuel amount needed for the user-defined satellite disposal strategy. The difference between these two fuel amounts is the fuel amount available for any required satellite maneuvers during the operational phase. When the fuel amount on board the satellite drops to the amount required for a disposal maneuver or its quantity is not enough for another maneuver the end of operational phase is reached. The *Operational Lifetime Prediction* analysis supports the following maneuvers:

- Launcher dispersion correction
- Orbit transfer from the launch orbit to the mission orbit
- Disposal strategy
- Orbit raising after decay due to atmospheric drag
- Collision avoidance maneuvers
- Station keeping in GEO orbits
- Desaturation of flywheels using thrusters

In Section 5.13.1 the available options to set up an *Operational Lifetime Prediction* analysis are described. The equations and the assumptions and simplifications used in this analysis are given in Section 5.13.2. The output produced by this analysis can be found in Section 5.13.3.

Note: The *Operational Lifetime Prediction* analysis can only be performed for satellites on Earth orbit.

5.13.1 Configuration

The *Operational Lifetime Prediction* analysis contains the following settings:

- *Simulation output file* is common for several analyses and is explained in Chapter 5, subsection *Common analysis settings*
- *Write analysis data only* is common for several analyses and is explained in Chapter 5, subsection *Common analysis settings*
- *Vehicle* is common for several analyses and is explained in Chapter 5, subsection *Common analysis settings*
- *Phase* is common for several analyses and is explained in Chapter 5, subsection *Common analysis settings*
- *Evaluation time* denotes if the beginning or the end of the phase shall be considered as the start of the operational phase.
- *Atmospheric drag* configures the orbit raising after a decay due to atmospheric drag. The available options for this type of maneuver are:
 - *Reference area* allows the specification of an aerodynamic reference area of the selected vehicle
 - *Apoapsis radius* margin definitions:
 - *Decrease* denotes by how much the apoapsis is allowed to decrease before executing a maneuver
 - *Increase* denotes by how much above the nominal value the reboost maneuver is allowed to increase the apoapsis radius
 - *Periapsis radius* margin definitions:
 - *Decrease* is a positive value denoting by how much the nominal periapsis value is allowed to decrease before executing a maneuver
 - *Increase* is a positive value denoting by how much above the nominal periapsis value the reboost maneuver is allowed to increase the periapsis radius
 - *Engines for orbit raising* allows the selection of the engines with which the reboost maneuvers shall be performed. The *Efficiency* takes a value between 0.0 and 1.0 and accounts for misalignment from the nominal thrust direction (tangential to the orbit) of the reboost maneuver. If a value smaller than 1.0 is used this means that the thrust vector is not in the optimal direction (tangential to the orbit) and the delta-V which the engine has to produce increases.
 - *Fuel margin* is the percentage of the actual computed amount of fuel for the maneuver which shall be reserved as safety margin.
- *Disposal* allows the modelling of a disposal maneuver at the end of the operational life. The available options for this type of maneuver are:
 - *Strategy* sets the disposal strategy as one of the following:

ANALYSES

- *Deorbit* (default option) allows the user to specify an orbit periapsis radius to trigger the deorbit burn
- *Graveyard* allows the user to specify an orbit radius of a circular graveyard orbit
- *Automatic* calculation of a graveyard altitude for GEO satellites
- *Engines for disposal* allows the selection of the engines with which the selected disposal strategy shall be performed. The *Efficiency* takes a value between 0.0 and 1.0 and accounts for misalignment from the nominal thrust direction (tangential to the orbit) of the disposal maneuver. If a value smaller than 1.0 is used this means that the thrust vector is not in the optimal direction (tangential to the orbit) and the delta-V which the engine has to produce increases.
- *Fuel margin* is the percentage of the actual computed amount of fuel for the maneuver which shall be reserved as safety margin.
- *Collision avoidance* allows the user to consider a collision avoidance maneuver in the *Operational Lifetime Prediction* analysis. The available options for this maneuver type are:
 - In the *Debris* table the user specifies a catalog of debris size classes for the mission orbit. Each debris class object is specified by:
 - *Diameter* is the diameter of the debris size class
 - *Flux* is the 2D debris flux distribution for this size class
 - *Uncertainty* denotes the debris position uncertainty (i.e. r_{deb} as explained in Section 5.13.2)

The ESA MASTER-2009 software can be used to obtain values and information about debris size classes and sources relevant for the scenario.

- *Minimum diameter considered* is a threshold of the debris size classes diameter to be considered for collision avoidance maneuvers. Any debris size class with a smaller diameter than this threshold value will be neglected.
- *Vehicle uncertainty* is uncertainty of the vehicle position (i.e. r_{veh} as explained in Section 5.13.2)
- *Engines for collision avoidance* allows the selection of the engines with which the selected disposal strategy shall be performed. The *Efficiency* takes a value between 0.0 and 1.0 and accounts for misalignment from the nominal thrust direction (tangential to the orbit) of the collision avoidance maneuver. If a value smaller than 1.0 is used this means that the thrust vector is not in the optimal direction (tangential to the orbit) and the delta-V which the engine has to produce increases.
- *Fuel margin* is the percentage of the actual computed amount of fuel for the maneuver which shall be reserved as safety margin.
- *Evasive altitude* is the desired change in altitude at the point of collision (see paragraph *Collision Avoidance* in Section 5.13.2 for further information)

ANALYSES

- *Return to previous orbit* allows the user to specify that the vehicle shall stay on the new orbit (deactivated) or shall return to the nominal mission orbit (activated) after the collision avoidance maneuver has been accomplished
- *Attitude maintenance* is a maneuver to maintain constant attitude with thrusters in order to compensate friction in momentum wheels. At least one *Flywheel* has to be present in the selected vehicle in order to be able to add *Attitude maintenance* settings to the analysis. The following settings for this maneuver type are available:
 - **Add/Remove** allow adding or removing additional wheel-thruster combinations
 - *Wheel* allows the selection of a momentum wheel in the vehicle chosen under *Vehicle* for which friction shall be considered
 - *Lower limit* is the limit of the wheel rotation speed below which the attitude maintenance is active
 - *Upper limit* is the limit of the wheel rotation speed above which the attitude maintenance is active
 - *Coulomb friction* is the friction torque inside the selected *Wheel* independent of rotational speed
 - *Viscous friction* is the friction torque inside the selected *Wheel* depending on the rotational speed
 - *Thrusters* allow the selection of one or more thrusters to counteract the torque created in the selected *Wheel*
- *Geo station keeping* considers station keeping in GEO due to north/south and east/west drift. The available options for this maneuver type are:
 - *Engines for GEO station keeping* allows the user to select the engine with which the station keeping is performed. The efficiency can be any value between 0.0 and 1.0 to account for misalignment from the nominal thrust direction.
 - *Fuel margin* is the percentage of the actual computed amount of fuel for the maneuver which shall be reserved as safety margin.
- *Launch orbit different from mission orbit* considers that the vehicle has been launched in an orbit which is different from the mission orbit and has to accomplish the orbit transfer to the target mission orbit with its own engines. If this maneuver type is selected the data taken from the simulation is assumed to be the launch orbit. The available options for this maneuver type are:
 - *Mission orbit* specifies the Keplerian elements of the mission orbit
 - The *Injection errors* setting allows the user to introduce injection errors in the target launcher orbit. Depending on each injection error the analysis computes the required Δv and the resulting required fuel mass in order to compensate these errors. The correction of the true anomaly injection error, in contrast to the other injection errors, is done by lowering the orbit until the correct position in the orbital plane is reached. In this case, setting the *Number phasing revolutions* is required. This setting is the desired number of revolutions which the vehicle should spend in the lower phasing orbit.

ANALYSES

By increasing the number of revolutions the required Δv for the phase corrections is decreasing at the expense of the time spent to reach the intended true anomaly.

- *Engines for orbit transfer* allows to select the engine with which the orbit transfer is executed. The efficiency can be any value between 0.0 and 1.0 to account for misalignment from the nominal thrust direction.
- *Fuel margin* is the percentage of the actual computed amount of fuel of a maneuver which is reserved as safety margin.

5.13.2 Background

This chapter gives an overview over the equations and assumptions used for the operational lifetime prediction analysis.

Orbit transfer

If *Launch orbit different from mission orbit* is selected, the vehicle performs an orbit transfer first from the actually reached launch orbit to the intended launch orbit (if injection errors are specified) and then from the intended launch orbit to the mission orbit. Since constraints due to the payload can not be considered (e.g. engines attitude is not nominal due to satellite attitude restrictions imposed by e.g. the sun sensor or thermal radiator), in-plane corrections and out-of-plane corrections are executed separately. The following list of maneuvers is used for correcting both the injection errors and the transfer to the mission orbit:

- Changing semi-major axis and eccentricity is done by two tangential in-plane maneuvers at periapsis and apoapsis starting with a burn at periapsis followed by a second burn at apoapsis:

$$\Delta V = \sqrt{\left(2 \cdot \mu \cdot \left(\frac{1}{R} - \frac{1}{(Peri_t + Apo_t)}\right)\right)} - \sqrt{\left(2 \cdot \mu \cdot \left(\frac{1}{R} - \frac{1}{(Peri_i + Apo_i)}\right)\right)} \quad (5.34)$$

with R being the radial position of the vehicle, $Peri_t$ and Apo_t being the target periapsis and target apoapsis radius and $Peri_i$ and Apo_i being the initial periapsis and apoapsis radius before executing the maneuver and μ being the gravitational parameter of the central body.

- Changing the inclination is done either at the ascending node or at the descending node whichever requires less Δv :

$$\Delta v = 2 \cdot v_{node} \cdot \cos(\Phi_{fpa}) \cdot \sin\left(\frac{\Delta i}{2}\right) \quad (5.35)$$

with v_{node} being the velocity at the node, Φ_{fpa} being the flight path angle and Δi being the difference in inclination to be changed.

- Changing the ascending node is done at the maximum latitude point, i.e. where the distance to the equatorial plane is greatest. The required Δv is given by:

$$\Delta v = 2 \cdot v_{lat} \cdot \sin(i) \cdot \sin\left(\frac{\Delta \Omega}{2}\right) \quad (5.36)$$

ANALYSES

with v_{lat} being the velocity at the maximum latitude point, i being the inclination and $\Delta\Omega$ being the difference in right ascension of the ascending node to be achieved.

- Changing the argument of periapsis is done with two out-of-plane maneuvers, the first performed at an eccentric anomaly of 90.0° and the second performed at an eccentric anomaly of 270.0° :

$$\Delta v = \left| \Delta\omega \right| \cdot \frac{n \cdot a \cdot e}{\sqrt{1 - e^2} + \sin([anom_{ecc}])} \quad (5.37)$$

with $\Delta\omega$ being the change in argument of periapsis for each maneuver (i.e. half of the total difference between current and target argument of periapsis), n being the mean motion of the orbit, a being the semi-major axis, e being the eccentricity and $anom_{ecc}$ being the eccentric anomaly where the maneuver is executed. The first maneuver also affects the semi-major axis as follows:

$$\Delta a = \Delta v \cdot 2 \cdot a \cdot \frac{\sqrt{1 - e^2}}{n \cdot R} \quad (5.38)$$

The second maneuver will change the argument of periapsis to the target value while also restoring the semi-major axis.

- Changing the true anomaly is done by first lowering the orbit, then drifting in this phasing orbit until the phasing is correct and finally executing an in-plane maneuver to increase the semi-major axis to its previous value. The orbital period of the phasing orbit is computed as follows:

$$T_{phase} = \frac{\left(2\pi - \frac{\Delta\theta}{rev_{phase}} \right)}{n} \quad (5.39)$$

with T_{phase} being the period of the phasing orbit, $\Delta\theta$ being the difference in true anomaly and rev_{phase} being the user-defined number of revolutions in the phasing orbit. From the orbital period the semi-major axis is computed as follows:

$$a_{phase} = \left(T_{phase} \cdot \frac{\sqrt{\mu}}{2\pi} \right)^{\frac{2}{3}} \quad (5.40)$$

with a_{phase} being the semi-major axis of the phasing orbit.

Disposal strategy

Depending on the selected disposal strategy, the following assumptions are made:

- *Deorbit burn*: the deorbit burn is executed with a single burn to lower the orbit periapsis to the specified value. The maneuver is assumed to be outside the atmosphere and therefore executed using the vacuum thrust value for the engine.
- *Graveyard orbit*: two impulsive maneuvers are executed to enter the graveyard orbit first adjusting the apoapsis and then the periapsis.
- *Automatic graveyard*: the altitude of the graveyard orbit is calculated as follows [150]:

$$h_{graveyard} = h_{GEO} + 235km + \left(1000 \cdot C_R \cdot \frac{A}{m} \right) \quad (5.41)$$

ANALYSES

where $h_{graveyard}$ is the graveyard orbit altitude, h_{GEO} is the operational orbit altitude, C_R is the solar radiation pressure coefficient, A is the reference area and m is the dry mass of the vehicle. Two impulsive maneuvers are used to enter the graveyard orbit by first raising the apogee and then the perigee.

Atmospheric drag

Atmospheric drag is assumed to affect predominantly the perigee thereby first circularizing the orbit. The change in semi major axis and eccentricity after each orbit are calculated as [107]:

$$\Delta a = -2\pi \cdot \left(C_D \cdot \frac{A}{m} \right) \cdot a^2 \cdot \rho_p \cdot \exp(-c) \cdot (I_0 + 2 \cdot I_1 \cdot e) \quad (5.42)$$

$$\Delta e = -2\pi \cdot \left(C_D \cdot \frac{A}{m} \right) \cdot a \cdot \rho_p \cdot \exp(-c) \cdot \left(I_1 + \frac{e}{2} \cdot (I_0 + I_1) \right) \quad (5.43)$$

with C_D being the drag coefficient (assumed value is 2.2), A being the reference area for the aerodynamic effects, m being the current mass of the vehicle, ρ_p being the atmospheric density at the periapsis, a being the semi major axis of the orbit and e being the eccentricity of the orbit. The quantities c , I_0 and I_1 are defined as:

$$c = \frac{a \cdot e}{H} \quad (5.44)$$

with H being the atmospheric scale height at periapsis. I_0 and I_1 are the modified Bessel functions

$$I_0(c) = \begin{cases} 1 + \frac{c^2}{4} + \frac{c^4}{64} + \frac{c^6}{2304} & c < 3 \\ \frac{\exp(c)}{\sqrt{2\pi c}} \left(1 + \frac{1}{8c} + \frac{9}{128c^2} + \frac{75}{1024c^3} \right) & c \geq 3 \end{cases} \quad (5.45)$$

$$I_1(c) = \begin{cases} \frac{c}{2} + \frac{c^3}{16} + \frac{c^5}{384} & c < 3 \\ \frac{\exp(c)}{\sqrt{2\pi c}} \left(1 - \frac{3}{8c} - \frac{15}{128c^2} - \frac{105}{1024c^3} \right) & c \geq 3 \end{cases} \quad (5.46)$$

Collision avoidance

All defined debris size classes are assumed to be equally distributed around the operational orbit. The collision cross section is calculated as

$$A_{cross} = \pi \cdot (r_{veh} + r_{deb})^2 \quad (5.47)$$

The time interval between avoidance maneuvers for this size class is calculated from the cross section as

$$t_{interval} = \frac{1}{flux \cdot A_{cross}} \quad (5.48)$$

with r_{veh} being the positional uncertainty of the vehicle, r_{deb} being the positional uncertainty of the debris size class, and $flux$ being the debris flux for the size class. The minimum interval of all size classes with a diameter larger than the given considered minimum yields the final interval between maneuvers.

Since the location of the collision on the operational orbit can not be computed, a worst-case scenario is assumed for which the collision point has an eccentric anomaly of 90.0 degrees, so that the maneuver executed in the periapsis raises the apoapsis by more than the computed altitude. This calculation is done as follows:

ANALYSES

$$r_{apo'} = \frac{(\sqrt{r_{peri} \cdot r_{apo}} + h_{evade})^2}{r_{peri}} \quad (5.49)$$

with $r_{apo'}$ being the apoapsis radius the avoidance maneuver aims at, r_{peri} and r_{apo} being the periapsis radius and apoapsis radius of the current orbit respectively.

Station keeping in GEO

For satellites in GEO orbits the north/south drift due to the gravity of the sun and moon is considered as well as the J2-effect causing east/west drifts around the nearest stable longitude. All drifts are immediately compensated and the nominal position above the Earth stays constant. The north/south drift is calculated as follows [107]:

$$\Delta V_{Moon} = 102.67 \cdot \cos(\alpha) \cdot \sin(\alpha) \quad (5.50)$$

$$\Delta V_{Sun} = 40.17 \cdot \cos(\gamma) \cdot \sin(\gamma) \quad (5.51)$$

with α being the angle between the Moon orbit and the vehicle orbit and γ being the angle between the vehicle orbit and the ecliptic. Both of the above equations yield a Δv in m/s per year. East/west drifts occur mainly due to the J2-effect of the earth's gravitational field and are calculated as

$$\Delta V_{J22} = 1.715 |\sin(2(l_D - l_S))| \quad (5.52)$$

with l_D being the longitude of the vehicle and l_S being the closest stable longitude, either 75.0 degrees or 255.0 degrees. The resulting Δv is in m/s per year.

5.13.3 Output

The following scalar outputs (*Auxiliary Scalars*) are computed by the *Operational Lifetime Prediction* analysis. All $\Delta\#$ outputs are expressed in m/s and all fuel-related outputs are expressed in kg .

- *operational_lifetime* represents the duration of the operational phase in years and is calculated so that after the maneuvers required for the selected disposal strategy the tanks contain only the *Fuel margin* amount of fuel. The bigger the *Fuel margin* value, the shorter the the computed operational lifetime.
- *total_deltaV_reboost* is the sum of Δv over all reboost maneuvers required to compensate atmospheric drag.
- *total_fuel_reboost* is the sum of propellant consumption over all reboost maneuvers required to compensate atmospheric drag.
- *total_deltaV_disposal* is the sum of Δv over all disposal maneuvers required for the selected disposal strategy.
- *total_fuel_disposal* is the amount of fuel required to perform the selected disposal strategy.
- *total_deltaV_collision_avoidance* is the Δv required for all collision avoidance maneuvers.
- *total_fuel_collision_avoidance* is the amount of fuel required to perform all collision avoidance maneuvers.

ANALYSES

- *total_deltaV_station_keeping* is the Δv required to compensate all north/south and east/west drifts in GEO orbit.
- *total_fuel_station_keeping* is the fuel required to compensate all north/south and east/west drifts in GEO orbit.
- *total_deltaV_attitude_maintenance* is the Δv required to desaturate the flywheels using thrusters.
- *total_fuel_attitude_maintenance* is the fuel required to desaturate the flywheels using thrusters.
- *total_deltaV_launcher_dispersion* is the Δv required to correct injection inaccuracies by the launcher.
- *total_fuel_launcher_dispersion* is the fuel required to correct injection inaccuracies by the launcher.
- *total_deltaV_orbit_transfer* is the Δv required to get from the launch orbit to the mission orbit.
- *total_fuel_orbit_transfer* is the fuel required to get from the launch orbit to the mission orbit.
- *mission_orbit_semi_major_axis* is the semi-major axis in kilometer of the mission orbit to be maintained.
- *mission_orbit_eccentricity* is the eccentricity of the mission orbit to be maintained.
- *mission_orbit_inclination* is the inclination in degree of the mission orbit to be maintained.
- *mission_orbit_ascending_node* is the ascending node in degree of the mission orbit to be maintained.
- *mission_orbit_periapsis_argument* is the argument of periapsis in degree of the mission orbit to be maintained.

Additionally this analysis produces one textual general info item containing the selected disposal strategy.

5.14 Requirements Analysis

Via the ASTOS Requirements Analysis it is possible to define a set of boolean expressions for the evaluation of simulation or analysis results. The results of the requirements analysis can become part of a report. For example in tabular format, all requirements and their fulfillment status can be listed. In this way a time consuming manual assessment of multiple aspects of the simulation results can be avoided. This is especially useful in larger simulation, analysis or optimization campaigns with varying parameter sets and many simulations or optimizations.

5.14.1 Configuration

The configuration of the requirements analysis is similar to the definition of a *Condition Plot* (see Condition Plots in the User Manual book). For each requirement, a boolean expression can be defined that consists of operators, constants and data source references (function values).

After adding a requirements analysis, clicking on the new button in the requirements analysis configuration panel will add a new requirement to the requirements table below. Each requirement comprises an identifier, a description and of course the boolean expression. Identifier and description are listed in the requirements table and can be modified by double-clicking on the table cell. The boolean expression is edited in a multi-row edit field below. The requirements table has a further column that shows the last results of the analysis.

The boolean expression can consist of the following elements:

- **Value** to define a constant (consists of value and unit)
- Variable, which is any output value stored in a struct simulation file
- +, -, *, / arithmetic operators
- (and) to influence the operator precedence
- <, <=, =, >=, > relational operators
- AND, OR, XOR and NOT logical operators

Via the delete buttons on the right of the *Conditions* pane either single expression elements or the whole expression can be deleted. Single expression elements can be selected with the mouse.

5.14.2 Output

The Requirements analysis stores its results inside simulation.struct. For each Requirement a General Info item will be added to phase 1 (string value). These General Info items can be printed via the report function.

5.15 Rocket Motor Analysis

The *Rocket Motor* analysis allows a rapid performance assessment of a *Throat Design* rocket engine (see Section 3.2.2.5). This analysis can only be executed if the selected engine uses *RPA* [110] for the *Equilibrium calculation*. Specific impulse, thrust, characteristic velocity, efficiencies and mass can be computed for different conditions (e.g. sea level, vacuum, etc). Moreover, if the geometry of the engine has been defined (see Section 3.2.2.5), RPA computes the contour of the engine as shown in Fig. 5.16.

ANALYSES

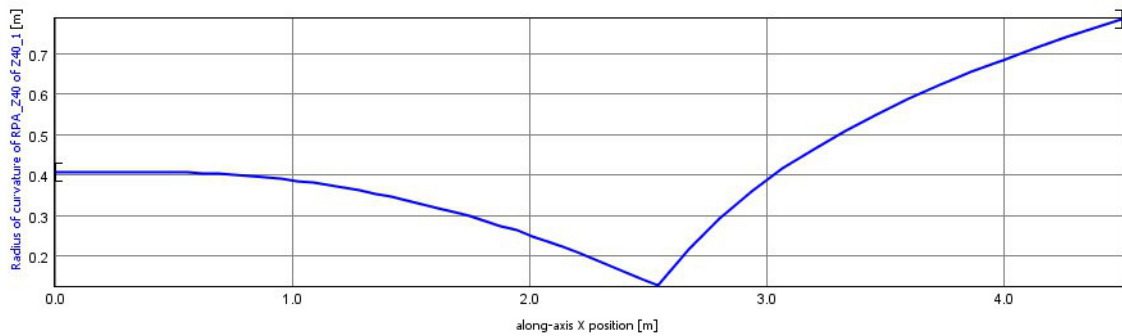


Fig. 5.16: Example of a rocket motor contour obtained with ASTOS

Together with the engine's contour, RPA can perform the thermal analysis, providing heat flux and wall temperature along the engine as output. For this computation the engine's geometry must be known too.

5.15.1 Configuration

The *Rocket Motor* analysis contains the following settings:

- *Engine* sets which rocket motor is to be analysed. The respective drop-down list contains all throat design engines defined in *Vehicle Parts & Properties* -> *Actuators* which are assigned to a vehicle.
- If the *Performance* analysis is *enabled* in the configuration panel, the following settings are available:
 - *Compute at sea level* to analyse the engine performance at sea level (at pressure of 101325 Pa)
 - *Compute at optimum expansion* to determine the performance at the optimum expansion pressure for the engine
 - *Compute in vacuum* to compute the engine performance in an ideal vacuum (i.e. a pressure of 0 Pa)
 - *Compute at custom pressure* allows the computation of the performance at a custom pressure provided by the user
- If the *Thermal* analysis is *enabled* in the configuration panel, either *default* or *custom* settings for a set of *Heat transfer parameters* can be used. In the latter case, the following parameters are present:
 - *Convective heat transfer formulation* provides three different methods to calculate the gas-side heat transfer rates [112]:
 - *levlev* approach for the calculation of convective heat transfer in the nozzle

ANALYSES

- *Bartz* approach, which uses semi-empirical correlation for gas-side heat transfer coefficients
- *Combined* provides the average value of the previous methods
- *Radiation heat transfer* specifies the emissivity of the gas-side wall surface.
- *Heat flux at wall temperature* calculates the heat flux at a fixed wall temperature (no chamber cooling). If *disabled*, the wall temperature is not kept constant along the engine.
- *Relative wall thickness* is the tube thickness-to-diameter ratio.
- *Number of stations* defines number of output points along the engine (a minimum of 30 stations is required).

All *default* values are inherited from RPA [112]. *Heat transfer parameters* are ignored in case an RPA configuration file is used to define the throat design engine as the parameter values are directly imported from the file. In this way the definition of thrust chamber cooling is possible even if not supported by the GUI (see [112] for more details).

5.15.2 Output

Performance analysis

The performance analysis provides the following output for each requested condition:

- *Exhaust velocity* in m/s
- *Specific impulse* (*Isp*) in s

Furthermore, the following time-independent output is always provided:

- *Characteristic velocity* in m/s
- *Overall efficiency* is the total efficiency (dimensionless between 0.0 and 1.0) obtained by multiplying all the individual terms listed below
 - *Reaction efficiency* is the only throttle-dependent efficiency
 - *Nozzle efficiency*
 - *Friction efficiency* accounts for the friction thrust loss
 - *Divergence efficiency* accounts for the divergence thrust loss
 - *Multiphase efficiency* accounts for the multiphase flow and phase transitions
- *Chamber pressure loss coefficient* is the coefficient of stagnation pressure loss at the nozzle inlet
- *Critical pressure* in Pa is the value of the pressure that causes sonic flow velocity
- *Optimum expansion pressure* in Pa is the value of the external pressure which provides the maximum thrust (the nozzle is said to be adapted)
- *Mass flow* in kg/s
- *Vacuum thrust* in N

ANALYSES

Together with the listed performance, the following engine dimension parameters (see Fig. 5.17) are provided by RPA:

- L_c (chamber length)
- L_e (nozzle length)
- R_e (nozzle exit radius)
- D_t (throat diameter)
- L_{cyl}
- D_c
- b
- R_1
- R_2
- R_n
- D_e
- T_n
- T_e

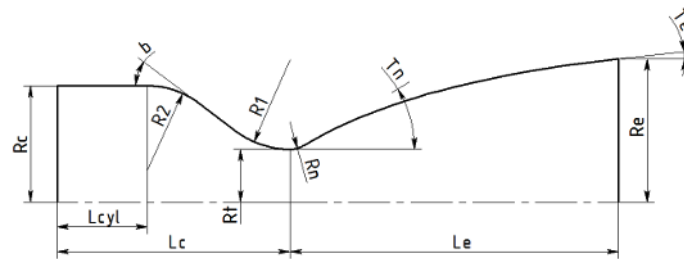


Fig. 5.17: Throat Design engine geometry [110]

If the *Engine mass correlation* of the engine is set to *RPA* (see Section 3.2.2.5), then also the computed engine mass and the *Mass of other elements* (which accounts for fixed components as pipes, wires, etc) are provided in the output.

Thermal analysis

As output of the thermal analysis, the following time-dependent parameters are provided at each station (output point along the engine, see Section 5.15.1):

- *Radius of curvature* in m
- *Convective heat coefficient* in $W/m^2/K$
- *Convective heat flux* in W/m^2
- *Radiative heat flux* in W/m^2
- *Total heat flux* in W/m^2
- *Wall temperature* in K

5.16 Tracking Analysis

The *Tracking Analysis* allows the simulation of satellite tracking by a ground station. Multiple targets can be tracked at the same time by the same ground station and for each of them typical data (e.g. range, range-rate, elevation, azimuth, etc.) is computed. The user does not need to specify any tracking sensor, since the targets are considered in view every time the elevation angle is positive. The aforementioned tracking data is computed also when the satellite is not visible from the ground station.

5.16.1 Configuration

The *Tracking* analysis contains the following settings:

- *Simulation output file* is common for several analyses and is explained in Chapter 5, subsection *Common analysis settings*
- *Write analysis data only* is common for several analyses and is explained in Chapter 5, subsection *Common analysis settings*
- *Ground station* defines the observer which tracks the targets
- *Targets* gives a list of vehicles which can be selected for tracking

5.16.2 Output

For each target the following output functions are computed:

- relative position, velocity and acceleration in the LD-frame (Section 9.3.4.2)
- Height, elevation and azimuth angles in the LD-frame (Section 9.3.4.2)
- Line of sight distance, great-circle distance between the sub-satellite point and the ground station
- X, Y and Z Cartesian relative position and velocity in the ground station LD-frame
- Pitch and yaw angles in the FL-frame (Section 9.3.6.2)

Moreover, for each target the following scalar is evaluated:

- Number of contacts, i.e. every time the elevation angle becomes positive

5.17 Visibility Analysis

One of the major mission analysis aspects of a satellite mission is the observability of a satellite during the mission. The observability of the satellite from ground stations is the condition to exchange data with the spacecraft.

In a similar way, it is important to know if and when on-board payload can observe a predefined target (e.g. vehicle, celestial body).

The *Visibility* analysis can assess both these problems providing additional output describing the geometric configuration, visibility windows and distance from target. Moreover, it checks if any of the celestial bodies defined in the scenario is obstructing the line of sight vector between observer (which can be either a vehicle or a sensor) and target.

5.17.1 Configuration

The *Visibility* analysis contains the following settings:

- *Simulation output file* is common for several analyses and is explained in Chapter 5, subsection *Common analysis settings*
- *Write analysis data only* is common for several analyses and is explained in Chapter 5, subsection *Common analysis settings*
- *Observer Type* defines if the observer is a *Vehicle* or a *Sensor*. If a vehicle is selected a simple point-to-point visibility is evaluated, while if a sensor is selected its field of view and visibility window will be considered too.
- The *Targets* panel accommodates the definition of the objects the user is interested in. Dedicated output is generated for each selected element according to its type (as described in Section 5.17.2). The following target groups are available:
 - *Ground stations* define the ground segment. For each ground station the observability by the observer defined in *Observer Type* is evaluated.
 - A *Minimum elevation at ground stations* angle can be considered in order to create an artificial mask below the specified value.
 - *Vehicles* allow the user to evaluate the visibility between the previously defined observer and spacecraft from the list of available spacecraft.
 - *Sensors* allow the user to evaluate the visibility between the observer defined in *Observer Type* and sensors from the list of available sensors.
 - *Celestial bodies* allow the user to evaluate the visibility between the observer defined in *Observer Type* and celestial bodies from the list of available celestial bodies.
- *Minimum separation angle* allows to define a *custom* minimum angle between a target and each obstructing celestial body. This setting allows the user to consider the blinding effect of a bright celestial body close to the target which would prevent observing the target.

5.17.2 Output

For each target the following output is generated:

- *Aspect angle* is the angle between the x_B -axis of the vehicle B-frame and the target
- *Range* is the distance between target and vehicle
- *Observability* is equal to 0.0, if the line of sight between vehicle and target is obstructed or at least one of the requirements regarding the *Minimum elevation at ground stations* (for ground stations) or the *Minimum separation angle* (for all target) is not met, otherwise it is equal to 1.0
- *Duration observable* gives the time window with observability equal to 1.0
- *Duration unobservable* gives the time window with observability equal to 0.0

For each ground station the *Elevation angle* and the *Azimuth angle* are also computed.

Additional output is computed considering all the ground stations together as network:

- *Ground station observability* gives the combined observability output for all ground stations
- *Ground station duration observable* gives the combined observability duration output for all ground stations
- *Ground station duration unobservable* gives the combined unobservability duration output of all ground stations
- *Daily average duration of ground station observability* gives the average duration of visibility passes per day
- *Revolutions ground station observability* gives the ground station observability duration divided by the orbital period
- *Revolutions ground station unobservability* gives the ground station unobservability duration divided by the orbital period

Additional output is computed considering all the targets together:

- *Observability* is equal to 0.0, if at least one of the target's observability is equal 0.0
- *Duration observable* gives the time window with observability equal to 1.0
- *Duration unobservable* gives the time window with observability equal to 0.0

6 Phase-End Conditions and Optimization Constraints

Description

An introduction to optimization constraints is provided in section Constraints Management of the User Manual. The following section describes all phase-end conditions and optimization constraints available in the software.

- **Phase-end conditions** are similar to optimization constraints and terminate a phase when they are fulfilled. They are used for simulation in case the **Optimization** feature is *disabled* and for initial guess generation in case the **Optimization** feature is *enabled*.
- **Optimization Constraints** are conditions to be satisfied by a solution of the optimization problem

The phase-end conditions and the optimization constraints are grouped according to their application in:

- **Position** related conditions/constraints, see Section 6.2
- **Velocity** related conditions/constraints, see Section 6.3
- **Attitude control** related conditions/constraints, see Section 6.4
- **Forces** related conditions/constraints, see Section 6.5
- **Orbit** related conditions/constraints, see Section 6.6
- **Deorbit** related conditions/constraints, see Section 6.7
- **Integrated quantity** conditions/constraints, see Section 6.8
- **Model** conditions/constraints, see Section 6.9
- **Indirect phase** conditions/constraints, see Section 6.10
- **Inter-satellite** conditions/constraints, see Section 6.11

Furthermore some automatically created optimization constraints, they are not available as phase-end conditions, are described in:

- **Automatic model** constraints, see Section 6.1

Configuration

The following generic settings are available for all constraints (but not phase-end conditions):

PHASE-END CONDITIONS AND OPTIMIZATION CONSTRAINTS

- *Description* is an optional string to describe the constraint. The description is used in the *Result Summary* (see Result Summary in User Manual).
- *Vehicle ID* is the identifier of the vehicle the constraint belongs to
- *Is enforced* describes whether the constraint is part of the active constraint set (*enabled*, default) or not (*disabled*) directly after initial guess generation. When *disabled*, the constraint is created but not considered during the optimization unless it is activated in the corresponding *Constraints Info* panel. This feature is mainly useful for an initial guess where a fulfillment of all constraints is not yet the main intention. Sometimes it is also a good strategy not to enforce a (secondary) constraint in the beginning of an optimization, but to switch it on in later iterations.

Note: This setting is only considered in the initial guess generation! For later optimization steps, constraints have to be enforced individually for each phase in the *Grids* node (for path constraints, see in User Manual) and in the *Constraints Info* node (for all other constraint types, see in User Manual).

- **Phases table** defines in which phase(s) the constraint is present. Multiple selections are possible in the *Specified* column. However, a multiple selection is not possible for boundary constraints, for example a constraint cannot be set as *Initial Boundary* and *Final Boundary*. In this situation a clone of the constraint has to be created. There are some special entries in the *Phase ID* column:
 - (*All*) can be used if a constraint shall be present in all phases of the scenario
 - (*First*) is always the first phase in the scenario
 - (*Last*) is always the last phase in the scenario

The latter two options might be useful in case the order of phases is switched or in case phases are added or removed. In this case, the phase identifier of the constraints does not need to be updated.

Furthermore also the kind of constraint is specified:

- *Initial Boundary* is a scalar boundary constraint at the beginning of a phase (e.g. initial altitude)
- *Final Boundary* is a scalar boundary constraint at the end of a phase (e.g. target orbit altitude)
- *Path* is a constraint which is active throughout the complete phase (e.g. ground station visibility)
- *Parameter* is a constraint that links two model parameters (e.g. ratio between component scaling factors). Parameter constraints are evaluated once per phase.

Note: Only the dedicated model parameter constraints (e.g. *Subtype Model Parameter Ratio*) are designed to be used as *Parameter* constraint. All other constraints should be set as *Initial Boundary*, *Final Boundary* or as *Path* constraint.

PHASE-END CONDITIONS AND OPTIMIZATION CONSTRAINTS

- *Limit* is the reference value of the constraint and determines if the constraint is used as an equality (i.e. a reference value) or an inequality (i.e. lower/upper limit) constraint via *Defined by*:
 - *Reference* is used to define a precise value (e.g. target inclination). Such a constraint should never be used as path constraint.
 - *Lower Limit* is used to specify that a function value cannot fall below the value specified (e.g. altitude cannot be lower than the launch tower when starting the rotation of the vehicle).
 - *Upper Limit* is used to specify that a function value cannot exceed the value specified (e.g. maximum load factor).
 - *Range* is used to define the acceptable range that function values can achieve (e.g. Mach number between 2 and 3).

More details on the definition of limits are provided in Constraints Data of the User Manual.

- *Infeasible limit* is optional and specifies the scale factor of the constraint violation computation. This optimization constraint violation is computed as the difference between the function value and the reference value divided by a scale factor. Usually the scale factor is the reference value, but in some situations it might be desired to specify a different scale factor: the *Infeasible limit*.

Note: In case of angular constraints the infeasible limit is ignored and an internal scaling factor based on the sine function is applied to improve the optimization performance.

Background

Additional Phase end conditions

Additional phase end conditions can be defined in each phase in the settings pane of each phase and are not part of an optimal control problem. These conditions are introduced to the scenario and are evaluated only during the initial guess generation action or during the simulation action if no optimization feature is used. As their name suggests, the *Additional phase end conditions* are used to conditionally terminate the propagation of a phase. For example, a minimum altitude of 10 km as a phase end condition would cause the propagation to stop as soon as the vehicle reaches an altitude of at least 10 km within this particular phase. This feature is very powerful and frequently used for the initial guess generation since the evolution with time of some physical quantities is not known a priori, and therefore also the duration of a phase cannot be specified in advance. Since the mathematical formulation of a phase-end condition is identical to the corresponding optimization constraint, there is no separate section listing and describing all additional phase end conditions.

The values of each path constraint on the scenario grid points are stored in a dedicated auxiliary function. If additional auxiliary functions are provided as output, they are mentioned in the corresponding condition/constraint paragraph.

6.1 Automatic Model Constraints

All constraints of this section are not configurable by the user and they are also not available as phase-end conditions. Depending on the scenario definition, they are automatically created and added during the initialization (initial guess generation). Automatic model constraints comprises:

- *Booster_Length* defines an initial boundary constraint on the length of a booster during vehicle design optimization. It ensures the length of the booster equals the distance from the lower attachment point (i.e. bottom of the vehicle) to the upper attachment point. This constraint is only created in phase 1 in case a *Detailed Vehicle Design Booster* is present, more details in Section 3.3.2.7.
- *Branching_Offset* defines a parameter constraint on the phase time offset in case of branched phases. It is only present in case of branching and unnormalized independent variable. This equality constraint ensures the phase duration equals the branching offset parameter and is only required for internal purpose. It should not be changed.

Note: The branching feature is currently not supported!

- *Chebyshev_Limit* defines a path constraint on the Chebyshev cost function (see Section 7.1). This cost function requires a path constraint in order to evaluate the complete phase, therefore it is created as inequality (i.e. upper bound) in each phase the cost function is present.
- *Initial_Ascending_Node* defines an initial boundary constraint on the right ascension of the ascending node of the user-defined initial state in phase 1. It is only available in case the initial state is defined by Keplerian orbit elements.
- *Initial_Eccentricity* defines an initial boundary constraint on the eccentricity of the user-defined initial state in phase 1. It is only available in case the initial state is defined by Keplerian orbit elements.
- *Initial_Epoch* defines a parameter constraint on the orbit epoch of the user-defined initial state in phase 1
- *Initial_Inclination* defines an initial boundary constraint on the orbit inclination of the user-defined initial state in phase 1. It is only available in case the initial state is defined by Keplerian orbit elements.
- *Initial_Latitude* defines an initial boundary constraint on the latitude of the vehicle. It is only available in case the inertial navigation coordinate system (see Section 9.3.6.4) is defined with a reset after the first phase (see Configuration of Assemblies and Elements of the User Manual).
- *Initial_Longitude* defines an initial boundary constraint on the longitude of the vehicle. It is only available in case the inertial navigation coordinate system (see Section 9.3.6.4) is defined with a reset after the first phase (see Configuration of Assemblies and Elements of the User Manual).
- *Initial_Perigee_Argument* defines an initial boundary constraint on the argument of periapsis of the user-defined initial state in phase 1. It is only available in case the initial state is defined by Keplerian orbit elements.

PHASE-END CONDITIONS AND OPTIMIZATION CONSTRAINTS

- *Initial_PROP_MASS* defines an initial boundary constraint on the propellant mass of a tank in the first phase in case the tank is already used. If a tank is not used in the first phase but in later phases, this condition is automatically fulfilled through the corresponding phase-connect condition. The initial propellant mass has to equal the propellant mass of this tank computed with the filling ration design parameter. For each tank of the scenario such a constraint is automatically created. It must always be enforced and of type equality. The constraint name includes the engine and the vehicle identifier.
- *Initial_Propellant_Volume* defines an initial boundary constraint on the propellant volume of a vehicle component (i.e. stage) in the first phase in case the component is already used. If a tank is not used in the first phase but in later phases, this condition is automatically fulfilled through the corresponding phase-connect condition. The initial propellant volume of all tanks of this component has to equal the total volume of all tanks. It must always be enforced and of type equality. The constraint is useful for dual propellant stages and will be only added in case the volume of the component is defined constant, otherwise a *Initial_PROP_MASS* constraint is created for each tank. The constraint name includes the engine and the vehicle identifier.
- *Initial_Semi_Major_Axis* defines an initial boundary constraint on the semimajor axis of the user-defined initial state in phase 1. It is only available in case the initial state is defined by Keplerian orbit elements.
- *Initial_Time_Offset* defines a parameter constraint on the mission time at the beginning of phase 1. It is automatically created if an initial time is defined or in case its bounds are not fixed. The constraint ensures that the mission time at beginning of phase 1 equals the value of the corresponding real parameter. For the definition of an initial time see Phase Settings in User Manual.

Note: The initial time offset feature is not available anymore!

- *Initial_True_Anomaly* defines an initial boundary constraint on the true anomaly of the user-defined initial state in phase 1. It is only available in case the initial state is defined by Keplerian orbit elements.
- *Normalized_Time* defines a parameter constraint on the normalization of the independent variable. It is created in all phases but only in case of normalized independent variable. It ensures the (unnormalized) duration of the phase, represented by a real parameter, equals the difference between the unnormalized initial and final phase times, represented by two real parameters.
- *Reentry_Altitude* defines an initial boundary constraint on the altitude of re-entry scenarios in phase 1. The altitude of the vehicle at the beginning of phase 1 has to equal the altitude of the user-defined re-entry orbit. This constraint is only available in case of a user-defined initial state as *Orbital Elements* with *Initial deorbit* enabled as *Transfer Orbit*.
- *Residual_PROP_MASS* defines a final boundary constraint on the propellant mass of a tank in all phase the tank is used. The propellant mass of the tank in the last phase of usage, the residual propellant mass, has to be bigger than zero. For each tank of the scenario such a constraint is automatically created. This constraint is automatically enforced in each phase where a tank is jettisoned and where the residual propellant mass can be affected by the solver. I.e. if the final phase time is fixed as well as initial propellant mass

PHASE-END CONDITIONS AND OPTIMIZATION CONSTRAINTS

and propellant mass-flow, this constraint is not enforced. See also the manual constraint *Residual Propellant* in Section 6.9.

- *Structural_Strength* defines a constraint on the wall thickness of a vehicle component, such as a fairing or a stage. It ensures the wall thickness is larger or equal its defined minimum value. This constraint is only created in case a *Detailed Vehicle Design Stage/Booster* is present in the vehicle.

Note: This constraint is currently not supported!

- *Unit_Vector_Norm* defines a path constraint on the length of the control vector in case of *Unit vector* as attitude definition (see Section 4.3.7). It ensures the length of the control vector equals 1 (equality) or is smaller than 1 (inequality, default).
- *virtual_ignition_time* defines a parameter constraint on the ignition time of a propulsion. It is an equality constraint only present in the first active phase of a propulsion system. It is required to compute the burn time at each time step. The constraint name includes the engine and the vehicle identifier.
- *virtual_shut_off_time* defines a parameter constraint on the shut-off time of a propulsion. It is an equality constraint only present in the last active phase of a propulsion system. Together with the *virtual_ignition_time* it is required to compute the normalized burn time at each time step. The constraint name includes the engine and the vehicle identifier.

6.2 Position Related Conditions/Constraints

- *Altitude* defines a constraint on the planeto-detic altitude (normal to surface) considering the radius of the sub-latitude point
- *Altitude Equatorial* defines a constraint on the planeto-detic altitude considering the equatorial radius of the central body. It is useful in connection with circular orbit constraints defined by an orbit altitude instead of a radius.
- *Altitude Machnumber* defines a constraint on the planeto-detic altitude as a function of the Mach number. The altitude profile has to be provided as profile data table (see Profile Data Tables in User Manual).
- *Altitude Velocity* defines a constraint on the planeto-detic altitude as a function of the flight path speed. The altitude profile has to be provided as profile data table (see Profile Data Tables in User Manual).
- *Cross Range* defines a constraint on the distance between the current point (i.e. landing site) and the initial orbital plane. It is defined positive in the direction of the angular momentum of the initial orbital plane (i.e. port side).
- *Declination* defines a constraint on the planeto-centric latitude (see Fig. 2.2)
- *Down Range* defines a constraint on the distance between the projection of the current point (i.e. landing site) on the initial orbital plane and the initial point (i.e. entry point)

PHASE-END CONDITIONS AND OPTIMIZATION CONSTRAINTS

- *Ground Distance* defines a constraint on the distance from the specified target object on the reference ellipsoid of the central body. These specific settings are required (for general settings, see Chapter 6):

- *Coordinates* is a location in the planeto-detic frame on the current central body.
 - *Longitude* specifies the (planeto-centric) longitude of the target location
 - *Latitude* specifies the planeto-detic latitude of the target location

The constraint evaluation is given by $d = R \cdot \text{acos}((\lambda, \delta) \cdot (\lambda_T, \delta_T))$ with the longitude λ_T and the declination δ_T of the target location. The declination is computed with the provided planeto-detic latitude.

- *Area of Interest* specifies the ID of the previously defined Area of Interest entity.
- *Point of Interest* specifies the ID of the previously defined Point of Interest entity.

- *Latitude* defines a constraint on the planeto-detic latitude (see Fig. 2.2)

- *Longitude* defines a constraint on the (planeto-centric) longitude

- *Position* defines a constraint on a single coordinate of a position vector. The following settings are required:

- *Frame* specifies the coordinate frame of the position vector
 - *Barycentric rotating* is a two-body rotating coordinate system (see Section 9.3.1.1) used to define any orbit in the Circular Restricted Three Body Problem (CR3BP, refer to Section 4.5.1.2).
 - *J2000* is the mean equator and equinox of epoch at J2000.0 axis system (see Section 9.3.2.1)
 - *PCPF* is the rotating planet-centered planet-fixed axis system (see Section 9.3.2.4)
- *Representation* specifies the mathematical representation of the position vector, in particular if it is *Cartesian* or *Spherical*
- *Coordinate* specifies the numerical coordinate, in case of *Cartesian Representation* either
 - *X*
 - *Y*
 - *Z*

and in case of *Spherical Representation* either

- *Altitude* specifies the radial distance w.r.t. surface of the central body
- *Declination* specifies the planetocentric elevation angle
- *Latitude* specifies the planetodetic elevation angle
- *Longitude* specifies the azimuthal angle

PHASE-END CONDITIONS AND OPTIMIZATION CONSTRAINTS

- *Radius* specifies the radial distance to the center of the central body
- *Pressure* defines a constraint on the atmospheric (ambient) pressure
- *Radius* defines a constraint on the radius of the vehicle, the distance to the center of the central body
- *Station Aspect Angle* defines a constraint on the angle between the x-axis of the vehicle (longitudinal axis) and the vector from the vehicle to the ground station (also called aspect angle). It is given by $\kappa = f(R, R_S) \geq \kappa_{min}$; κ with the geographic position $R_S = (h, \lambda, \Phi_G)$ of the station: altitude h , longitude λ and latitude Φ_G .
180 degree identifies a station behind the nozzle whereas 0 degree a ground station in front of the nose.
- *Station Visibility* defines a constraint evaluating the visibility between a vehicle and one or multiple ground stations and/or sensors. These specific settings are required (for general settings, see Chapter 6):
 - *Ground stations* is a list of all ground stations defined in the scenario
 - *Sensors* is a list of all sensors mounted at ground stations

The visibility is finally evaluated for all selected ground stations and sensors. The separation angle between each of the aforementioned targets and the selected vehicle is computed and the minimum value is compared with the specified input value. In case of sensors, their cone angle and viewing window are considered before computing the separation angle. If the vehicle is outside the cone angle or the viewing window the separation angle is not computed, since no visibility can be achieved.

- *Velocity* defines a constraint on a single coordinate of a velocity vector. The following settings are required:
 - *Referene_Frame* specifies the coordinate frame in which the velocity vector is measured:
 - *A* is the air-path coordinate system (see Section 9.3.5.3)
 - *J2000* is the mean equator and equinox of epoch at J2000.0 axis system (see Section 9.3.2.1)
 - *PCPF* is the rotating planet-centered planet-fixed axis system (see Section 9.3.2.4)
 - *Representation_Frame* specifies the coordinate frame in which the velocity vector is represented:
 - *Barycentric rotating* is a two-body rotating coordinate system (see Section 9.3.1.1) used to define any orbit in the Circular Restricted Three Body Problem (CR3BP, refer to Section 4.5.1.2).
 - *L* is the local planetocentric coordinate system (see Section 9.3.4.1)
 - *LD* is the local planetodetic coordinate system (see Section 9.3.4.2)
 - *LO* is the local orbit coordinate system (see Section 9.3.1.1)

- *Representation* specifies the mathematical representation of the velocity vector, in particular if it is *Cartesian* or *Spherical*
- *Coordinate* specifies the numerical coordinate, in case of *Cartesian Representation* either

- *X*
- *Y*
- *Z*

and in case of *Spherical Representation* either

- *Heading* specifies the azimuthal angle
- *Inclination* specifies the elevation angle
- *Speed* specifies the velocity magnitude

6.3 Velocity Related Conditions/Constraints

- *Active Oxidation* is present when the stagnation pressure (or the heat flux) computed by the radiative formulation is higher than the stagnation pressure computed by the convective formulation. It has experimental status since a general formulation has not been identified for this effect. The following specific settings are required (for general settings, see Chapter 6):

- *Formula* specifies the kind of formula used for the evaluation:
 - *CIRA* is according to [63] and has these settings:
 - *Exponent*
 - *Reference pressure* is the reference atmospheric pressure
 - *Rayleigh* is according to [75] and has these settings:
 - *k* is a constant of the formula
 - *w* is a constant of the formula
 - *Heat ratio* can be specified either as *Value* or as function of the Mach number (*Profile*, see Profile Data Tables in User Manual)
- *Aerothermodynamics* is optional and specifies the identifier of the aerothermodynamics model (see Section 3.5.2). If *custom*, the corresponding identifier of the aerothermodynamics model has to be chosen, otherwise the *default* model is used.

PHASE-END CONDITIONS AND OPTIMIZATION CONSTRAINTS

According to the Rayleigh formulation, the active oxidation occurs when the stagnation pressure computed from Eq. 6.1 is higher than the stagnation pressure computed from Eq. 6.2.

$$p_0 = k \cdot (T_w)^w \quad (6.1)$$

where p_0 is the stagnation pressure, k and w are constants provided by the user and T_w is the wall temperature computed from Eq. 3.176.

$$\begin{aligned} p_0 &= p_\infty \cdot \left(1 + \frac{\gamma-1}{2} \cdot M^2\right)^{\left(\frac{\gamma}{\gamma-1}\right)}; & M &= [0 \dots 1] \\ p_0 &= p_\infty \cdot \left(\frac{\gamma+1}{2} \cdot M^2\right)^{\left(\frac{\gamma}{\gamma-1}\right)} \cdot \left(\frac{2\gamma M^2 - \gamma + 1}{\gamma + 1}\right)^{\left(\frac{1}{1-\gamma}\right)}; & M &= (1 \dots 5] \end{aligned} \quad (6.2)$$

where p_∞ is the atmospheric air pressure, γ is the ratio of specific heats and M is the Mach number. As a not fully appropriate extension, the second line of Eq. 6.2 is also used for Mach numbers larger than 5.

Tip: *Active Oxidation* should be used as path constraint.

Three additional auxiliary function are the stagnation pressure of Eq. 6.2, the heat flux (`heat_flux`) and the wall temperature (`wall_temperature`).

- *Airpath Velocity* defines a constraint on the magnitude (speed) of the air path velocity vector
- *Dynamic Pressure* defines a constraint on the dynamic pressure of the vehicle

$$\bar{q} = \frac{1}{2} \rho V_a^2 \leq \bar{q}_{\max} \quad (6.3)$$

with the air path velocity V_a and the free-stream density of the atmosphere ρ .

One additional auxiliary function is the acceleration (`drag_acc_dyn_pressure`) for plotting a drag acceleration - velocity diagram, for example.

- *Dyn Press Velocity* defines a constraint on the dynamic pressure as function of the flight path velocity. The dynamic pressure profile has to be provided as profile data table (see Profile Data Tables in User Manual). This constraint is defined by $\bar{q} \leq \bar{q}_{\max}(V_k)$; with the dynamic pressure q and the flight path velocity V_k .
- *East Velocity* defines a constraint on the east component of the inertial velocity vector
- *Equilibrium Glide* defines a constraint evaluating if the derivative of the flight path inclination angle at a minimum air path bank angle must be greater or equal zero. Hence it should be always defined as lower limit, defining an acceleration of zero. The constraint can be used to define a safety condition for re-entry applications. It ensures together with a minimum bank angle not equal zero, that a safety margin to fly an $\dot{\gamma} > 0$; exists by flying a bank angle smaller than $\mu_{\min}$.

The following specific settings are required (for general settings, see Chapter 6):

- *Minimum acceleration* is the threshold until the minimum bank angle is set to zero
- *Roll angle profile* is the minimum roll angle as function of the Mach number (see Profile Data Tables in User Manual)

The computation of the constraint depends on $\mu_{\min}$. The sign of the gravity components in Eq. 6.4 comes from the local coordinate system L:

PHASE-END CONDITIONS AND OPTIMIZATION CONSTRAINTS

$$\begin{aligned} \dot{\gamma} = & \frac{V^2}{R} \cos \gamma + 2\Omega_E V \sin \chi \cos \delta \\ & + \Omega_E^2 R \cos \delta (\cos \chi \sin \gamma \sin \delta + \cos \gamma \cos \delta) \\ & + \frac{S}{m} \sin \mu_{\min} + \frac{L}{m} \cos \mu_{\min} \\ & - g_z \cos \chi \sin \gamma - g_y \sin \chi \sin \gamma + g_x \cos \gamma \end{aligned} \quad (6.4)$$

The minimum bank angle is zero if the acceleration is less than the specified *Minimum Acceleration*. Otherwise the bank angle is defined as a function of the minimum roll angle and the Mach number

$$\mu_{\min}(M, a) = \text{atan}(\tan \varphi_{\min}(M, a) \cos \alpha(M)) \quad (6.5)$$

As the Mach number M is not a unique function for a re-entry case, the minimum acceleration a defines the start value at which a minimum bank angle constraint is active. Two additional auxiliary function are the acceleration (`drag_acc_equil_glide`) with $\gamma(\mu_{\min}) = 0$ for plotting a drag acceleration - velocity diagram; and the bank angle (`bank_angle_equilibrium_glide`).

- *Flightpath Angle* defines a constraint on the (planeto-centric) inclination angle of the flight path velocity vector
- *Flightpath Angle Geodetic* defines a constraint on the planeto-detic inclination angle of the flight path velocity vector
- *Flightpath Heading* defines a constraint on the (planeto-centric) heading angle of the flight path velocity vector
- *Flightpath Heading Geodetic* defines a constraint on the planeto-detic heading angle of the flight path velocity vector
- *Flightpath Velocity* defines a constraint on the magnitude (speed) of the flight path velocity vector
- *Heat Flux* defines a constraint on the heat flux in the stagnation point. It requires the optional *Aerothermodynamics* model (see Section 3.5.2). If *custom*, the corresponding identifier of the aerothermodynamics model has to be chosen, otherwise the *default* model is used. Two additional auxiliary functions are the wall temperature computed by Boltzmann's law using the *Emissivity* and an acceleration (`drag_acc_heat_flux`) for plotting a drag acceleration - velocity diagram, for example.
- *Inertial Flightpath Angle* defines a constraint on the (planeto-centric) inclination angle of the inertial velocity vector
- *Inertial Flightpath Heading* defines a constraint on the (planeto-centric) heading angle of the inertial velocity vector
- *Inertial Velocity* defines a constraint on the magnitude (speed) of the inertial velocity vector
- *Knudsennumber* defines a constraint on the ratio between the mean free path (MFP) and the reference length. The mean free path is defined by [69]

$$MFP = \frac{KT}{p\sqrt{2}\pi d^2} \quad (6.6)$$

where K is the Boltzmann constant (see Chapter 10), T is the atmosphere temperature, p is the atmosphere pressure and $d(=3.65 \cdot 10^{-10} \text{ m})$ is the molecular diameter.

- *Machnumber* defines a constraint on ratio of the air path speed of the vehicle and the speed of sound of the atmosphere
- *North Velocity* defines a constraint on the north component of the inertial velocity vector
- *Radial Velocity* defines a constraint on the radial component of the inertial velocity vector
- *Rel East Velocity* defines a constraint on the east component of the (relative) flight path velocity vector

6.4 Attitude Control Related Conditions/Constraints

- *Airpath Bank Angle* defines a constraint on the air path bank angle
- *Airpath Bank Angle Rate* defines a constraint on the rate of the air path bank angle

Note: *Airpath Bank Angle Rate* is only usable if the bank angle is an optimizable control and the optimizer is *PROMIS* or *CAMTOS*; or if the bank angle is defined as linear law; or in case of 6-DoF dynamic systems.

- *Alpha Machnumber* defines a constraint on the angle of attack as function of the Mach number. The angle of attack profile has to be provided as profile data table (see Profile Data Tables in User Manual).
- *Angle of Attack* defines a constraint on the angle of attack
- *Angle of Attack Rate* defines a constraint on the rate of the angle of attack

Note: *Angle of Attack Rate* is only usable if the angle of attack is an optimizable control and the optimizer is *PROMIS* or *CAMTOS*; or if the angle of attack is defined as linear law; or in case of 6-DoF dynamic systems.

- *Bending Moment* defines a constraint on the product of the dynamic pressure and the total angle of attack
- *Gravity Turn Condition* evaluates the gravity turn condition where the pitch angle in the local horizontal coordinate system (see Section 9.3.4.1) equals the air path inclination angle
- *Load Factor* defines a constraint on the ratio of the total acceleration due to aerodynamic and propulsive forces and the local gravity acceleration. In re-entry scenarios this constraint can be used to minimize the structural weight of the vehicle. One auxiliary function is the acceleration (`drag_acc_load_factor`) for plotting a drag acceleration - velocity diagram.
- *Mu Machnumber* defines a constraint on the air path bank angle as function of the Mach number. The bank angle profile has to be provided as profile data table (see Profile Data Tables in User Manual).
- *Pitch* defines a constraint on the pitch angle in one of the following *Frames*:
 - *J2000* coordinate system, see Section 9.3.2.1

PHASE-END CONDITIONS AND OPTIMIZATION CONSTRAINTS

- *L* coordinate system, see Section 9.3.4.1
- *LO* coordinate system, see Section 9.3.1.1
- *N* coordinate system, see Section 9.3.6.4
- *Pitch Moment* defines a constraint on the torque about the body y-axis (vehicle starboard axis) as function of the Mach number. It requires a 6-DoF dynamic system.
- *Pitch Rate* defines a constraint on the rate of the roll angle in the current attitude frame

Note: *Pitch Rate* is only usable if the pitch angle is an optimizable control and the optimizer is *PROMIS* or *CAMTOS*; or if the pitch angle is defined as linear law; or in case of 6-DoF dynamic systems.

- *Roll* defines a constraint on the roll angle in one of the following *Frames*:
 - *J2000* coordinate system, see Section 9.3.2.1
 - *L* coordinate system, see Section 9.3.4.1
 - *LO* coordinate system, see Section 9.3.1.1
 - *N* coordinate system, see Section 9.3.6.4
- *Roll Moment* defines a constraint on the torque about the body x-axis (vehicle forward axis) as a function of the Mach number. It requires a 6-DoF dynamic system.
- *Roll Rate* defines a constraint on the rate of the roll angle in the current attitude frame

Note: *Roll Rate* is only usable if the roll angle is an optimizable control and the optimizer is *PROMIS* or *CAMTOS*; or if the roll angle is defined as linear law; or in case of 6-DoF dynamic systems.

- *Sideslip Angle* defines a constraint on the sideslip angle
- *Sideslip Angle Rate* defines a constraint on the rate of the sideslip angle

Note: *Sideslip Angle Rate* is only usable if the sideslip angle is an optimizable control and the optimizer is *PROMIS* or *CAMTOS*; or if the sideslip angle is defined as linear law; or in case of 6-DoF dynamic systems.

- *Tangential Yaw Condition* is achieved when the yaw angle in the local horizontal coordinate system (see Section 9.3.4.1) equals the air path heading angle
- *Yaw* defines a constraint on the yaw angle in one of the following *Frames*:
 - *J2000* coordinate system, see Section 9.3.2.1
 - *L* coordinate system, see Section 9.3.4.1
 - *LO* coordinate system, see Section 9.3.1.1
 - *N* coordinate system, see Section 9.3.6.4
- *Yaw Moment* defines a constraint on the torque about the body z-axis (vehicle downward axis) as a function of the Mach number. It requires a 6-DoF dynamic system.

- *Yaw Rate* defines a constraint on the rate of the roll angle in the current attitude frame

Note: *Yaw Rate* is only usable if the yaw angle is an optimizable control and the optimizer is *PROMIS* or *CAMTOS*; or if the yaw angle is defined as linear law; or in case of 6-DoF dynamic systems.

6.5 Forces Related Conditions/Constraints

- *Axial Acceleration* defines a constraint on the axial component (body +x-axis, vehicle forward axis) of the acceleration vector due to aerodynamic and propulsive forces
- *Axial Load Factor* defines a constraint on the ratio of the axial component (body +x-axis, vehicle forward axis) of the acceleration due to aerodynamic and propulsive forces and the local gravity acceleration
- *Cross Load Factor* defines a constraint on the ratio of the lateral component (body +y-axis, vehicle starboard axis) of the acceleration due to aerodynamic and propulsive forces and the local gravity acceleration
- *Drag Accel Velocity* defines a constraint on the drag acceleration as function of the flight path velocity. The drag acceleration profile has to be provided as profile data table (see Profile Data Tables in User Manual).
- *Lift Force* defines a constraint on the aerodynamic lift force (not the total lift force)
- *Lift to Drag Ratio* defines a constraint on the ratio of aerodynamic lift (not total lift) and aerodynamic drag
- *Normal Acceleration* defines a constraint on the vector sum of the lateral (body +y-axis, vehicle starboard axis) and normal component (body +z-axis, vehicle downward axis) of the acceleration vector due to aerodynamic and propulsive forces
- *Normal Load Factor* defines a constraint on the ratio of the normal component (body +z-axis, vehicle downward axis) of the acceleration due to aerodynamic and propulsive forces and the local gravity acceleration
- *Thrust to Weight Ratio* defines a constraint on the ratio of the total thrust and the total effective weight of the vehicle. Effective weight is the product of the mass and the effective gravity, which is the gravity acceleration minus the centrifugal acceleration.

6.6 Orbit Related Conditions/Constraints

Description

Below is a list of all orbit related constraints:

PHASE-END CONDITIONS AND OPTIMIZATION CONSTRAINTS

- *Angular Momentum* defines a constraint on the specific angular momentum of the orbit, the cross product of the position and velocity vector
- *Apoapsis Altitude* defines a constraint on the altitude of the apoapsis considering the equatorial radius of the celestial body
- *Apoapsis Height* defines a constraint on the apoapsis altitude (considering the equatorial radius) as function of the inertial velocity at apoapsis. The current angular momentum and the radius of the apoapsis yield the velocity in the apoapsis. It can be used to specify a trajectory envelope that is compatible with immediate abort considerations, e.g. during the launch of a manned vehicle.
- *Apoapsis Radius* defines a constraint on the radius of the apoapsis
- *Ascending Node* defines a constraint on the right ascension of the ascending node, the angle between the x-axis of the inertial coordinate system and the ascending node of the orbit
- *Asymptote Declination* defines a constraint on the declination (planeto-centric latitude) of the asymptote of the escape hyperbola
- *Circular Altitude* constrains the altitude of a circular orbit with the current circular velocity (inertial velocity without radial component). The altitude considers the equatorial radius of the celestial body.
- *Circular Radius* constrains the radius of a circular orbit with the current circular velocity (inertial velocity without radial component)
- *Eccentricity* defines a constraint on the orbital eccentricity
- *Excess Velocity* defines a constraint on the excess velocity of the orbit, the velocity at infinity
- *Injection* constrains the true anomaly to the point of the injection burn into an escape hyperbola of a subsequent interplanetary transfer
- *Orbit Inclination* defines a constraint on the orbital inclination angle, the angle between the x-y-plane of the inertial coordinate system and the orbital plane
- *Orbital Period* defines a constraint on the period of the osculating orbit
- *Periapsis Altitude* defines a constraint on the altitude of the periapsis considering the equatorial radius of the celestial body
- *Periapsis Argument* defines a constraint on the argument of the periapsis
- *Periapsis Radius* defines a constraint on the radius of the periapsis
- *Plane intersection* constrains the true anomaly to the point where the current orbital plane around a central body intersects the orbital plane containing the escape hyperbola of a subsequent interplanetary transfer.
- *Semi Major Axis* defines a constraint on the semimajor axis of the orbit
- *Specific Energy* defines a constraint on the specific orbital energy, the sum of the potential energy and the total kinetic energy
- *Sphere of Influence* defines a constraint on the radius of the vehicle equaling the radius of the sphere of influence of the celestial body w.r.t. its parent body
- *True Anomaly* defines a constraint on the true anomaly of the orbit

Configuration

PHASE-END CONDITIONS AND OPTIMIZATION CONSTRAINTS

Most orbit related constraints have the following specific settings in common (for general settings, see Chapter 6):

- *Celestial Body* is the body the orbital property shall be related to. It can be any celestial body defined in the node *Celestial Bodies* in the environment part of the *Modelling* tree. By default it is the current central body.
- *Reference frame* specifies the coordinate system of the orbital property and is only available for angular quantities (e.g. inclination) with the following choices:
 - *J2000* coordinate system, see Section 9.3.2.1
 - *TOD* coordinate system, see Section 9.3.2.3

The interplanetary transfer conditions/constraints *Injection* and *Plane Intersection* solve Lambert's problem to evaluate the required velocity changes. An interplanetary transfer starts at the initial celestial body and ends at the final celestial body. Both bodies orbit their common central body. A transfer leg is around the central body from the initial celestial body to the final one and requires the following settings:

- *Number of complete revolutions* defines the number of orbits around the center body of the transfer leg. Zero means, the transfer leg is less than one revolution.
- *Orbit direction* is the orbit direction of the transfer leg. Options are *Prograde* and *Retrograde*. If retrograde, the angular momentum vector of the transfer leg points opposite to the angular momentum vector of the initial celestial body.
- *Transfer type* is the kind of the transfer leg, either *Short arc* or *Long arc*. Both options represent two solutions of Lambert's problem. In general, the short arc is a kind of "direct transfer" within short time, while the long arc is typically outbound and requires a much longer transfer duration.

Tip: All constraints related to the same transfer leg should have identical parameter settings to work properly.

Note: The initial and final celestial body of these interplanetary conditions are automatically taken from the dynamics configuration.

Note: In order to specify a circular orbit condition, it is not advisable to define a periapsis and apoapsis constraint. This may result in a bad convergence. Instead use the following three constraints for a well defined circular condition:

1. *Radius* equal orbit radius
2. *Circular Radius* equal orbit radius
3. *Radial Velocity* equal zero

6.7 Deorbit Related Conditions/Constraints

- *Dvee Normal* defines a constraint on the de-orbit velocity impulse (delta-V) in normal direction, parallel to the inertial position vector
- *Dvee Radial* defines a constraint on the de-orbit velocity impulse (delta-V) in radial direction, parallel to the angular momentum vector
- *Dvee Tangential* defines a constraint on the de-orbit velocity impulse (delta-V) in tangential direction, parallel to the inertial velocity vector
- *Flightpath Angle at Reentry* evaluates the flight path inclination angle of jettisoned stages or re-entry vehicles at a specified altitude. It is similar to *Splash Down* constraint. Since re-entry scenarios are targeted here, the user must select a negative value for the inclination angle *Limit*. These specific settings are required (for general settings, see Chapter 6):
 - *Altitude* defines the altitude until the trajectory is propagated and when the flight path inclination angle is computed
 - *Drag computation* defines the kind of the trajectory computation with these options:
 - *No Drag* does not compute any drag component (drag is zero) and computes analytically the impact location as intersection of the impact trajectory with the surface of the central body assumed as spheroid
 - *Combined Average Aero* considers aerodynamic drag and propagates numerically the impact trajectory until an altitude of zero considering these specifications:
 - *Maximum periapsis radius* defines the maximum allowed periapsis radius of the impact object at the beginning of the computation. If the value is actually higher, the constraint is not evaluated.
 - *Aerodynamics ID* specifies the identifier of the aerodynamics model

Note: Only the aerodynamics models of type *Tabular* and *User* are supported!

- *Output file* is optional and defines the path and name of an output file containing trajectory data of the impact object

- *Splash Down* evaluates the impact position of jettisoned stages or re-entry vehicles (see *Splash Down Computation with ASTOS in Safety and DRA Tutorial*).

Note: The constraint uses the mass of the selected vehicle if it is set as initial boundary or path constraint. If it is set as final constraint the mass of the jettisoned assemblies is used instead.

The following specific settings are required (for general settings, see Chapter 6):

- *Boundary type* is the kind of the constraint evaluation with these options:
 - *Circle* evaluates the distance of the impact point (λ_{IP} , Φ_{IP}) w.r.t. a circle defined by:

PHASE-END CONDITIONS AND OPTIMIZATION CONSTRAINTS

- *Radius* specifies the radius R of the constraint circle
- *Longitude* specifies the longitude λ_0 of the constraint circle center
- *Latitude* specifies the planeto-detic latitude Φ_0 of the constraint circle center

This kind of evaluation can be used to avoid an island or to limit the impact area of debris, for example.

Tip: An upper limit constraint forbids the area outside the circle.

- *Line* evaluates the perpendicular distance of the impact point $(\lambda_{IP}, \Phi_{IP})$ w.r.t. a constraint line defined by:
 - *Heading* specifies the heading χ , the angle w.r.t. north direction (clockwise positive), of the constraint line.
 - *Longitude* specifies the longitude λ_0 of the constraint line center
 - *Latitude* specifies the planeto-detic latitude Φ_0 of the constraint line center

This kind of evaluation is normally used as a final boundary constraint in the same phase where a component is jettisoned.

Tip: A lower limit constraint forbids the area on the left side of the constraint line when walking along the line in heading direction.

- *Drag computation* defines the kind of the trajectory computation with these options:
 - *No Drag* does not compute any drag component (drag is zero) and computes analytically the impact location as intersection of the impact trajectory with the surface of the central body assumed as spheroid
 - *Combined Average Aero* considers aerodynamic drag and propagates numerically the impact trajectory until an altitude of zero considering these specifications:
 - *Maximum periapsis radius* defines the maximum allowed periapsis radius of the impact object at the beginning of the computation. If the value is actually higher, the constraint is not evaluated.
 - *Aerodynamics ID* specifies the identifier of the aerodynamics model

Note: Only the aerodynamics models of type *Tabular* and *User* are supported!

- *Output file* is optional and defines the path and name of an output file containing trajectory data of the impact object

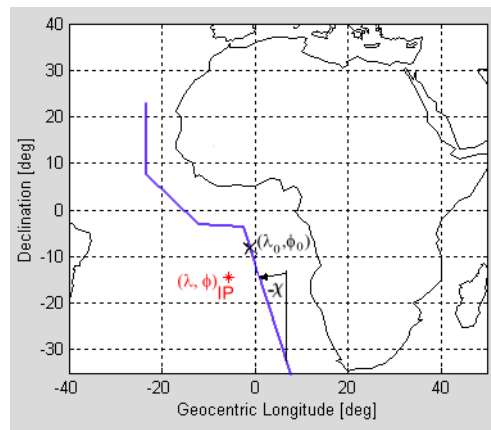


Fig. 6.1: Splash Down line constraint at the west coast of Africa: defining an upper limit of 0° forbids the area east of the lower blue line. A polygon as shown can only be defined using several constraints. But they would influence each other.

Tip: To visualize the impact point, activate the corresponding *Data → Auxiliary Items...* from the *Viewer* menu in a Map plot or 3D Map Plot containing latitude and longitude.

No auxiliary item is produced in case the splash down is defined as path constraint, but the following auxiliary functions (in the Constraints group of *Curve Plots*):

- *splash_down_declination* is the planeto-centric latitude of the stage/vehicle
- *splash_down_latitude* is the planeto-detic latitude of the stage/vehicle
- *splash_down_longitude* is the longitude of the stage/vehicle
- *splash_down_time* is the (flight) duration of the stage/vehicle starting at zero

6.8 Integrated Quantity Conditions/Constraints

- *Heat Load* defines a constraint on the integrated heat flux density and requires the auxiliary state *Heat load*, see Section 4.5.3.5
- *Ozone Load* defines the constraint on the integrated ozone density and requires the auxiliary state *Ozone load*, see Section 4.5.3.10
- *Trajectory Smoothness* defines a constraint on the trajectory smoothness, a measure of trajectory oscillations, and requires the auxiliary state *Trajectory Smoothness*, see Section 4.5.3.7

6.9 Phase Duration and Model Conditions/Constraints

The constraints of this section are connected to the phase time duration and one or more model objects. The constraints that refer to parameters (starting with "Model Param") require to be set as *Parameter* constraint (instead of *Final Boundary* constraint).

- *Equipment Design* defines a constraint modelled in one of the following vehicle parts models:

- *Actuators*
 - *Solid Design Rocket Engine*, refer to Section 3.2.2.7
 - *Throat Design Rocket Engine*, refer to Section 3.2.2.5
- *Aerodynamics*
 - *Tabular aerodynamics model*, refer to Section 3.5.1.1
- *Components*
 - *Detailed Vehicle Design Stage*, refer to Section 3.3.2.6
 - *Detailed Vehicle Design Booster*, refer to Section 3.3.2.7
 - *Reentry Vehicle*, refer to Section 3.3.4

The constraint definition allows to specify the corresponding model and the parameter to be constrained.

This constraint has the following specific settings (for general settings, see Chapter 6):

- *Equipment Type* is the vehicle parts model the user wants to use.
 - *Equipment ID* is the name of the vehicle part or environment model as defined in the *Modelling tree*.
 - *Parameter* identifying the physical parameter to be constrained.
- *Hinge Moment* defines a constraint on the hinge moment of a rudder or flap. It has these specific settings (for general settings, see Chapter 6):
 - *Defined by* specifies the kind of hinge moment computation:
 - *Aerodynamic surface* computes the hinge moment of a user-specified aerodynamic surface:
 - *Aerodynamics* specifies the identifier of an aerodynamics model (only tabular aerodynamics are supported, see Section 3.5.1.1)
 - *Surface* specifies the identifier of the aerodynamic surface the hinge moment is computed for

PHASE-END CONDITIONS AND OPTIMIZATION CONSTRAINTS

The hinge moment is assumed to be the difference of the actual aerodynamic moment of an aerodynamic surface and its aerodynamic moment with zero deflection.

- *Local* computes the hinge moment with the following provided input:
 - *CHmax* is the maximum hinge moment coefficient $C_{H_{max}}$ of the rudder/flap
 - *Reference length* is the rudder/flap reference length L_{BF}
 - *Reference area* is the rudder/flap reference area S_{BF}

The hinge moment is given by $h = \rho C_{H_{max}} L_{BF} S_{BF}$ with the dynamic pressure q .

One auxiliary function is the acceleration (`drag_acc_hinge_moment`) for plotting a drag acceleration - velocity diagram.

- *Mission Time* defines a constraint on the current mission time of the scenario
- *Model Param Value* defines a constraint on a specified parameter of a model (e.g. actuator model). The parameter of the model must be present in the same phase as the constraint. These individual settings are required (for general settings, see Chapter 6):
 - *Type* is the kind of the model, either:
 - *Aerodynamics* model, see Section 3.5.1
 - *Component* model, see Section 3.3
 - *Propulsion* model, see Section 3.2.2
 - *Name* is the parameter name as present in the *Real Parameter* node
- *Model Params Delta* defines a constraint on the difference (delta) of two specified parameters of two models (e.g. a component model and an actuator model). Both parameters can be optionally scaled, so that the sum of both values can be computed as well, for example. The following specific settings are required (for general settings, see Chapter 6):
 - *Factor A* is the optional scaling factor of the first parameter (default is 1.0)
 - *Factor B* is the optional scaling factor of the second parameter (default is 1.0)
 - *Model 1* and *Model 2* have these settings:
 - *Type* is the kind of the model, either:
 - *Aerodynamics* model, see Section 3.5.1
 - *Component* model, see Section 3.3
 - *Propulsion* model, see Section 3.2.2
 - *Name* is the parameter name as present in the *Real Parameter* node

PHASE-END CONDITIONS AND OPTIMIZATION CONSTRAINTS

Both parameters need to be present in the same phase as the constraint. By default - factors not selected - the difference between both parameters is computed.

- *Model Params Ratio* defines a constraint on the ratio of two specified parameters of two models (e.g. a component model and an actuator model). The following specific settings are required (for general settings, see Chapter 6):

- *Model 1* and *Model 2* have these settings:
 - *Type* is the kind of the model, either:
 - *Aerodynamics* model, see Section 3.5.1
 - *Component* model, see Section 3.3
 - *Propulsion* model, see Section 3.2.2
 - *Name* is the parameter name as present in the *Real Parameter* node

Both parameters need to be present in the same phase as the constraint.

- *Phase Time Duration* defines a constraint on the duration of the phase time, the difference between the initial phase time (beginning of phase) and the final phase time (end of phase). It is normally used as equality constraint to ensure a fixed phase duration. It is automatically created in each phase in case of normalized times (see Independent Variable and Normalization in the User Manual book).
- *Residual Propellant* defines a constraint on the propellant mass of a tank. It is similar to the automatic model constraint *Residual_PROP_MASS* (see Section 6.1), but here it is a general propellant mass constraint and the reference value can be specified as *Limit*. It is usually used as manual residual propellant constraint replacing the automatically created one, since a residual propellant mass can be specified while the automatically created one uses zero. The following specific settings are required (for general settings, see Chapter 6):
 - *Tank ID* is a list of available tank components and specifies the tank used for the propellant mass

Note: When this constraint is used as final boundary constraint in the last phase of usage, the corresponding automatic *Residual_PROP_MASS* constraint needs to be deactivated (uncheck *enforced* in the *Constraints Info* node)!

- *Static Stability* is a constraint for the static stability of a vehicle. For a statically stable vehicle the static stability constraint requires a *lower limit* with a value of 0.0 or greater.

Note: This constraint only works in case the center of pressure is given by the aerodynamics model. Either a *Misdat* (see Section 3.5.1.3), *Sose* (see Section 3.5.1.4) or *Tabular* aerodynamics model (see Section 3.5.1.1) is required. In case of a tabular aerodynamics model, the center of pressure is only defined if either the *Transport moment* or at least one moment coefficient is specified.

PHASE-END CONDITIONS AND OPTIMIZATION CONSTRAINTS

If the vehicle's nose points to the left and the vehicle's tail points to the right, static stability is ensured when the center of pressure (CoP) is to the right of center of mass (CoM). The static stability in ASTOS is computed as (-1.0) times the difference between the position of the center of pressure (CoP) and the position of the center of mass (CoM), i.e. $static\_stability = -1.0 * (x_{CoP} - x_{CoM})$, along the x_B -axis of the vehicle's body (B) frame. Since the CoM is the origin of the vehicle body (B) frame, $static\_stability = -1.0 * x_{CoP}$. Thus, for a statically stable vehicle, x_{CoP} has to be negative (in the body (B) frame), resulting in a value for $static\_stability \neq 0.0$.

Both the position of the CoP and of the CoM (i.e. x-offset, y-offset, z-offset) can be specified with respect to the global node. This node is always present, at least one component of the vehicle being attached to it.

In the case of the CoM, the position offset w.r.t. the global node is set up either for the full vehicle (in the *Vehicle & POIs Dynamics*, inside the mass distribution settings) or for each individual component of the vehicle (inside the corresponding component settings in the *Vehicle Parts and Properties* inside the mass distribution settings).

In the case of the CoP, the position offset setup is done in the tabular aerodynamics only if the aerodynamic moments are defined via position offsets (not the case for moment coefficients).

- *User Model* defines a constraint modelled in one of the following vehicle parts models:

- *User Propulsion* model, refer to Section 3.2.2.9
- *User Aerodynamics* model, refer to Section 3.5.1.5
- *User Component* model, refer to Section 3.3.5

The constraint definition allows to specify the corresponding model and a constraint provided by it. The input unit is related to the coded constraint. More details about the user-defined models is given in Section 11.1.5.

This constraint has the following specific settings (for general settings, see Chapter 6):

- *Model* defines the user model type
- *Identifier* is the name of the vehicle part or environment model as defined in the *Modelling* tree.
- *DLL constraint name* is the name specified in the user model constraints interface.

Note: The provided limit value is ignored in case the user model library supports the interface 'constlim', refer to Section 11.1.5. Instead the returned value of the interface 'constlim' is used as limit.

- *Vehicle Mass* defines a constraint on the mass of the vehicle, either propellant, structure or both. This specific setting is required (for general settings, see Chapter 6):

- *Mass type* specifies the kind of the vehicle mass with these options:
 - *Propellant* is the total mass of all propellants of the vehicles
 - *Structure* is the total structural mass of the vehicle

- *Total* is the total propellant and structural mass of the vehicle

6.10 Indirect Phase Conditions/Constraints

Note: Indirect constraints are not available in the standard software installation, please contact service@astos.de in case of interest.

All constraints of this section are applicable only in indirect phases. They are intended for expert users who are familiar with the application of indirect optimization methods. Some constraints are only different representations of several orbit or velocity constraints that exploit special properties of the formulation of indirect phases in order to slightly speed up the computation.

Note: The transversality conditions can be omitted if the cost function is set to maximize the final mass or payload, or minimize the fuel consumption. In such cases, the transversality conditions are included in the Karush-Kuhn-Tucker conditions of the SQP method and need not be explicitly enforced. The costates are calculated in such way that they minimize the performance index. More details can be found in [56].

- *Indirect_Apogee_Radius* is the same as the *Apoapsis Radius* constraint
- *Indirect_Circular_Transversality_1* defines a constraint on the first transversality condition for a circular target orbit. This is to be used in combination with the *Indirect_Orbit_Inclination*, *Indirect_Radius*, *Indirect_Inertial_Velocity* and *Indirect_Flightpath_Angle* constraints (with Reference = 0.0). It returns $((a_r / a_v)P + Q)^T i_f = 0$ where $a_r / a_v = (i_f^T (i_N \times V)) / (i_f^T (i_N \times R))$ and i_f is the unit vector perpendicular to the orbit plane.
- *Indirect_Circular_Transversality_2* defines a constraint on the second transversality condition for a circular target orbit. It is to be used in combination with the *Indirect_Circular_Transversality_1* constraint and returns $Q^T i_v - (v/r)P^T i_R = 0$
- *Indirect_Elliptic_Transversality_1* is similar as the *Indirect_Circular_Transversality_1* constraint but for elliptic target orbits.
- *Indirect_Elliptic_Transversality_2* is the same as the *Indirect_Circular_Transversality_2* constraint but for elliptic target orbits. This is to be used in combination with the *Indirect_Elliptic_Transversality_1*, *Indirect_Orbit_Inclination*, *Indirect_Perigee_Radius* and *Indirect_Apogee_Radius* constraints.
- *Indirect_Elliptic_Transversality_3* defines a constraint on the third transversality condition for elliptic target orbits. It is to be used in combination with the *Indirect_Elliptic_Transversality_2* constraint.
- *Indirect_Flightpath_Angle* defines a constraint on the dot product of the unit vectors R and V : $i_R^T i_V$ (the cosine of the inertial flight path angle)

PHASE-END CONDITIONS AND OPTIMIZATION CONSTRAINTS

- *Indirect_Hamiltonian* defines a constraint on the Hamiltonian $H = L + P^T \left(G + \frac{T}{m} i_b \right) + Q^T V - \lambda_m q$ where $L = q$ and q is the overall mass-flow. This constraint can also be used as an inactive path constraint in order to get output data for the Hamiltonian value within an indirect phase.
- *Indirect_Inertial_Velocity* is the same as the *Inertial Velocity* constraint
- *Indirect_Orbit_Inclination* is the same as the *Orbit Inclination* constraint
- *Indirect_Perigee_Radius* is the same as the *Periapsis Radius* constraint
- *Indirect_Radius* is the same as the *Radius* constraint
- *Lambda_m* defines a constraint on λ_m
- *Primer_Vector_Norm* defines a constraint on the two-norm (Euclidean length) of the primer vector
- *Q_Vector_Norm* defines a constraint on the two-norm (Euclidean length) of the Q vector
- *Switching_Function* defines a constraint on the switching function $S = L + \frac{T}{m} P^T i_b - \lambda_m q$ where $L = q$ and q is the overall mass-flow. This constraint can also be used as an inactive path constraint in order to get output data for the switching function within an indirect phase.
- *Switching_Function_Approximation* defines a constraint on an alternative formulation of the switching function which is based on the assumption that the value of the Hamiltonian is zero along the trajectory. It is computed as $S = -P^T G - Q^T V$. This constraint can also be used as an inactive path constraint in order to get output data for the switching function within an indirect phase.

6.11 Inter-Satellite Conditions/Constraints

Description

For scenarios with more than one vehicle some special constraints might be required involving two vehicles (the vehicle and the target vehicle):

- *Distance* defines a constraint on the distance between two vehicles. Only one specific setting is required:
 - *Evaluation at true anomaly* is optional and specifies a true anomaly for the vehicle. If *disabled*, the distance between both vehicles is evaluated at the current time. When *enabled* the distance between both vehicles is computed as when the vehicle reaches the specified true anomaly. Both orbits of the vehicle and the target vehicle are analytically computed.
- *H-bar Distance* defines a constraint on the distance w.r.t. target vehicle in direction of angular momentum (h-bar). It is one constraint of the *R-bar/V-bar/H-bar* group, see below.
- *H-bar Velocity* defines a constraint on the velocity w.r.t. target vehicle in direction of angular momentum (h-bar). It is one constraint of the *R-bar/V-bar/H-bar* group, see below.

PHASE-END CONDITIONS AND OPTIMIZATION CONSTRAINTS

- *Plume Impingement* defines a constraint on the distance between the plume of a propulsion system (part of the vehicle) and the target vehicle. The following specific settings are required:
 - *Propulsion ID* specifies the identifier of the thruster model
 - *Beamlength* specifies the length L_{Beam} of the plume lobe exiting the thruster
 - *Beamwidth* is the semiangle α_{Beam} of the plume lobe exiting the thruster

The size of the plume lobe is given by

$$Size_{Lobe} = L_{Beam} \cdot \left(\cos(\alpha_{Boresight}) \right)^{\frac{\ln(2)}{\ln(\cos(\alpha_{Beam}))}} \quad (6.7)$$

with the beam length L_{Beam} and the beamwidth α_{Beam} . The origin of the plume is located in the nozzle center (bottom node of a thruster). As illustrated in Fig. 6.2, the boresight angle ($\alpha_{Boresight}$) is the angle between thruster boresight and the target vehicle direction as seen from the thruster nozzle center. This constraint should be used as *Lower limit*, indicating a required minimum distance between plume and target vehicle. The distance between target vehicle and plume is measured in direction to the nozzle center, and not perpendicular to the plume. The red segment in Fig. 6.2 is the $Size_{Lobe}$.

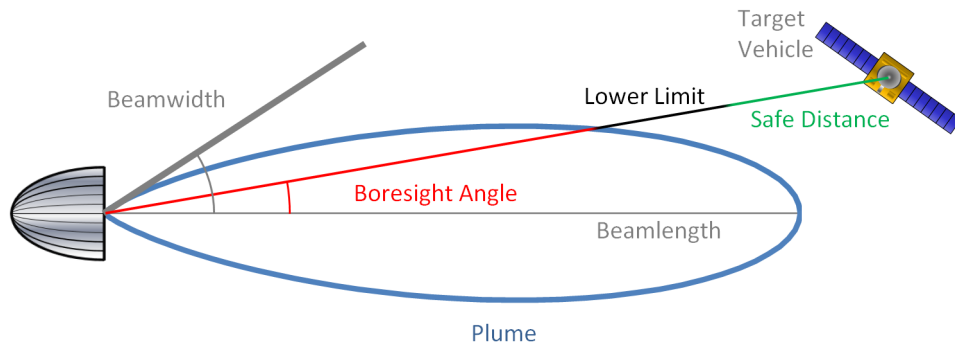


Fig. 6.2: Illustration of the plume impingement constraint with a beamwidth of 20 degree

- *R-bar Distance* defines a constraint on the distance w.r.t. target vehicle in radial (r-bar) direction. It is one constraint of the *R-bar/V-bar/H-bar* group, see below.
- *R-bar Velocity* defines a constraint on the velocity w.r.t. target vehicle in radial (r-bar) direction. It is one constraint of the *R-bar/V-bar/H-bar* group, see below.
- *V-bar Distance* defines a constraint on the distance w.r.t. target vehicle in velocity (v-bar) direction. It is one constraint of the *R-bar/V-bar/H-bar* group, see below.
- *V-bar Velocity* defines a constraint on the velocity w.r.t. target vehicle in velocity (v-bar) direction. It is one constraint of the *R-bar/V-bar/H-bar* group, see below.

Configuration

Generic settings are explained in Chapter 6. All inter-satellite constraints have the following settings in common:

PHASE-END CONDITIONS AND OPTIMIZATION CONSTRAINTS

- *Target vehicle* specifies the second/target vehicle identifier

Specific settings, if any, are provided above.

R-bar/V-bar/H-bar

In a rendezvous scenario, the spacecraft that performs orbital maneuvers is usually called chaser (here: vehicle), while the spacecraft approached by the chaser is called target (here: target vehicle). The reference frame has its origin at the center of the target vehicle, while the x-axis (or R-bar) points away from the center of the central body. The y-axis (or V-bar) is parallel to the current velocity vector of the target vehicle and the z-axis (or H-bar) is oriented in order to complete the right-handed frame: in direction of angular momentum of the target vehicle. The V-bar axis is actually not a straight line, but it follows the orbital curvature.

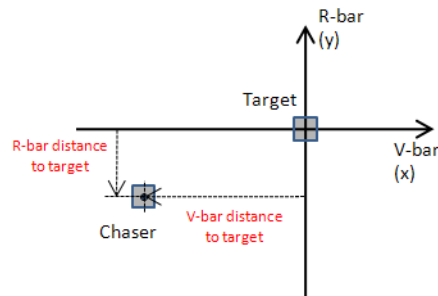


Fig. 6.3: R-bar Distance and V-bar Distance between two vehicles

7 Cost Functions

Description

This section describes all available cost function terms (see *Background* sub-section below) in the ASTOS software, which are grouped as follows:

- *Chebyshev* minimizes the maximum value of a function, see Section 7.1
- *Inter-satellite* minimizes quantities related to two vehicles (e.g. distance), see Section 7.2
- *Max Ascending Node Delta* maximizes the difference in the right ascension of the ascending node of two payload orbits, see Section 7.3
- *Max Crossrange* maximize the distances between the current point and the initial orbital plane, see Section 7.4
- *Max DeltaV* maximizes the velocity increment (delta-V), see Section 7.5
- *Max Downrange* maximizes the distance between the current point and the projection of a specified landing site on the initial orbit plane, see Section 7.4
- *Max Final Delta* maximizes the difference between the value of a function and a specified reference value at the end of a phase, see Section 7.6
- *Max Final Mass* maximizes the total mass of the vehicle, see Section 7.7
- *Max Final Value* maximizes the value of a function at the end of a phase, see Section 7.6
- *Max Great Circle Range* maximizes the distance between the entry point and the current point (i.e. landing site) measured along the great circle that connects both these points, see Section 7.4
- *Max Inclination Delta* maximizes the difference in the inclination of two payload orbits, see Section 7.8
- *Max Initial Delta* maximizes the difference between the value of a function and a specified reference value at the beginning of a phase, see Section 7.6
- *Min Initial Propellant* minimizes the initial propellant mass, see Section 7.9
- *Max Initial Value* maximizes the value of a function at the beginning of a phase, see Section 7.6
- *Max Payload* maximizes the payload mass, see Section 7.10
- *Max Safety* maximizes the safety of the trajectory, see Section 7.11
- *Max Splash Down* maximizes a property of a splash down event, see Section 7.12
- *Min Heat Load* minimizes the heat load, see Section 7.13
- *Min Loss* minimizes the losses of the velocity increment (delta-V), see Section 7.14
- *None* is a zero cost term, see Section 7.15

COST FUNCTIONS

- *User Model* enables the configuration of cost function terms for user defined models, see Section 7.17
- *Max Smoothness* optimizes the smoothness of a certain function, see Section 7.16

Tip: Some cost functions require certain model components to make sense, these restrictions are explicitly discussed in the description below.

Configuration

The following generic settings are available for each of the available cost terms:

- *Vehicle ID* is the identifier of the vehicle the cost functions belongs to
- *Phase* specifies the phase in which the objective function is evaluated. It is optional and by *default* it is evaluated in the last phase. This item has no influence on cost functions which depend on internal parameters (e.g. *Max Payload*).
- *Scaling* is a scaling factor by which the cost term value is multiplied. The scaling factor value can be used as an individual weighing factor for the cost function term. Setting the scaling factor to zero causes that this cost term is not be considered in the objective function.
- *Complex Scaling* is optional and allows a more sophisticated scaling of the cost term. The following options are supported:
 - *Quadratic* applies a quadratic scaling function
 - *Power Law* is a scaling function using a power law
 - *Exponential* is an exponential scaling function
 - *Logarithmic* is a logarithmic scaling function

Background

It is possible to switch between the cost function between two optimization runs. If the new cost function is different from the previous one, the TOPS *Verify* function detects this and opens the verify manager window to inform the user (no actions are requested to the user). The verify manager automatically activates the currently defined cost functions in the proper phases and adds/removes additional parameters and constraints as required.

Multiple cost functions can be added together resulting in the *Cost* value of the total objective function:

$$\text{Total Cost Function} = \sum_{i=1}^n (\text{Cost Function Term}_i \times \text{Scaling}_i) \quad (7.1)$$

which is **minimized** by the optimization. All objective functions are formulated such that the desired goal is attained by **minimizing** the objective function value. If the goal, for example, is to **maximize** payload, the corresponding cost function value is defined as being proportional to the **negative** payload mass. Minimizing this quantity is equivalent to maximizing the actual payload.

COST FUNCTIONS

Tip: As a rule, with a positive scaling factor the goal is described by the cost function name, e.g. *Max Payload* maximizes the Payload while *Min Loss* minimize the losses. In case the desired goal is opposite to the function name, a negative scaling factor should be used (e.g. to minimize the final altitude, the user should select *Max Final Value* with a negative scaling factor).

By default, the objective function value is scaled by the optimizer such that its magnitude is on the order of 1. This gives minimizing the cost function the same contribution to the overall optimization merit function as is meeting the constraints. In case that minimizing the cost is less important than meeting the constraints, the cost function contribution can be reduced by defining a scaling factor that places the objective between 0.01 and 0.1 (absolute values). For example, a payload mass in the range 1500 to 2000 kg and a scaling factor value equal 0.0001 yields a cost term between -0.15 and -0.2.

Note: The optimizer uses an internal scaling if the cost function is poorly scaled. Depending on the optimizer this happens when the objective function is greater than 100 or 1000. Such an action can be detected by the cost value of the first iteration output. This value is 1.0 in such a case (but it can also be 1.0 due to user scaling!) Generally, an optimizer scaling is not critical. Only in cases of multiple cost terms it should be avoided, as the relationship between the objective functions becomes completely unknown.

7.1 Chebychev

Description

The *Chebychev* cost term is a terminal cost term which minimizes the maximum value of a function, i.e. physical quantity. It is a fairly complicated cost function involving an additional real parameter and a path constraint, which are both created automatically. The following functions are supported:

- *Angle of Attack*
- *Axial Acceleration*
- *Bending Moment*
- *Dynamic Pressure*
- *Flightpath Angle*
- *Heat Flux*
- *Hinge Moment*
- *Load Factor*
- *Normal Acceleration*

The function has to be selected as *Subtype* of the *Chebychev* cost term.

Note: Putting an additional path or boundary constraint on the function you try to minimize can lead to inconsistent inequality constraints and/or infeasible problems.

Configuration

All generic cost term settings are described in Chapter 7. Furthermore the following specifications are required:

- The *Phase* setting for the Chebychev cost term results in the following behaviour, which is unique for this cost term. If no phase is specified (*default*) the (automatic) real parameter and path constraint are created in every phase, and the cost function is evaluated as terminal cost in the last phase. Thus the maximum value of the cost function is minimized over the whole interval of the independent variable, which is used for integration. In case a phase is specified (*custom*) the parameter and constraint are only created in this phase, and the terminal cost term is evaluated in this phase, too. The cost function acts in all phases where parameter and constraint are present.
- *Reference* is the reference value for the Chebychev parameter including optional lower and upper bound. These values are used for the initial guess of the Chebychev parameter and the input should be an estimate for the maximum function value.
- *Aerothermodynamics* is only present in case of *Subtype Heat Flux*. It specifies the model computing the heat flux in the stagnation point. If *custom*, the corresponding identifier of the aerothermodynamics model has to be chosen, otherwise the *default* model is used. More information about the aerothermodynamics models is given in Section 3.5.2.
- *Defined by* is only present in case of *Subtype Hinge Moment* and specifies the kind of hinge moment computation:
 - *Aerodynamic surface* computes the hinge moment of a user-specified aerodynamic surface:
 - *Aerodynamics* specifies the identifier of an aerodynamics model (only tabular aerodynamics are supported, see Section 3.5.1.1)
 - *Surface* specifies the identifier of the aerodynamic surface the hinge moment is computed for

The hinge moment is assumed to be the difference of the actual aerodynamic moment of an aerodynamic surface and its aerodynamic moment with zero deflection.

- *Local* computes the hinge moment with the following provided input:
 - *CHmax* is the maximum hinge moment coefficient $C_{H_{max}}$ of the rudder/flap
 - *Reference length* is the rudder/flap reference length L_{BF}
 - *Reference area* is the rudder/flap reference area S_{BF}

The hinge moment is given by $h = \rho C_{H_{max}} L_{BF} S_{BF}$ with the dynamic pressure q .

Background

COST FUNCTIONS

The basic problem to solve is to minimize the maximum value of a function

$$\min\{\max[f(x, u, t)]\} \quad (7.2)$$

This problem is reformulated by introducing a new real parameter c ("Chebychev") which is minimized. In addition, a path constraint is included such that

$$f(x, u, t) - c \leq 0 \quad (7.3)$$

The constraint would like to select the parameter c as large as possible, on the other hand c is minimized in the cost function. Therefore, the problem posed in Eq. 7.2 is solved.

7.2 Inter-Satellite Cost Functions

Description

For scenarios with more than one vehicle a cost function term involving two vehicles (primary vehicle and target vehicle) may be required. *Inter Satellite* is a terminal cost function terms which fulfills this requirement. Currently one subtype of this cost function term is available:

- *Distance* minimizes (*Scaling* > 0) the distance between two vehicles. Only one distance-specific setting is available:
 - *Evaluation at true anomaly* is optional and specifies a true anomaly for the vehicle. If *disabled*, the distance between both vehicles is evaluated at the current time. When *enabled* the distance between both vehicles is computed as when the vehicle reaches the specified true anomaly. Both orbits of the vehicle and the target vehicle are analytically computed.

Configuration

All generic cost term settings are described in Chapter 7. Furthermore the following specifications are required:

- *Target vehicle* specifies the second/target vehicle identifier

Specific settings, if any, are provided above.

7.3 Max Ascending Node Delta

Description

The *Max Ascending Node Delta* terminal cost function term is used to maximize the difference in right ascension of the ascending node of two orbits into which two different payloads with

COST FUNCTIONS

fixed masses are injected successively (first and last). At least two payload components (see Section 3.3.3.2) need to be defined in the *Components* branch of the *Modelling* tree. See also the cost term *Max Inclination Delta* in Section 7.8.

Configuration

The generic settings are described in Chapter 7.

Background

The cost function is given by

$$\text{Cost Function Term}_i = \text{Node}(\text{Orbit1}) - \text{Node}(\text{Orbit2}) \quad (7.4)$$

The optimizer minimizes the objective function; this leads to the maximization of the difference between the two orbits only if the second right ascension of ascending node is larger than the first one, otherwise the optimizer reduces the difference between the two orbits. In this second case, a negative scaling factor yields a maximization of the right ascension of ascending node difference.

7.4 Max Crossrange, Max Downrange, Max Great Circle Range

Description

These terminal cost function terms compute distances along the projected trajectory on the central body surface, in particular:

- *Crossrange* is the distance between the current point (i.e. the landing site) and the initial orbital plane. It is defined positive in the direction of the angular momentum of the initial orbital plane (i.e. port side).
- *Downrange* is the distance between the initial point (i.e. the entry point) and the projection of the considered point (i.e. landing site) on the initial orbital plane.
- *Great Circle Range* is the distance between the entry point and the current point (i.e. landing site) measured along the great circle that connects both these points

Note: Cross, down and great circle range are measured in inertial space and are not the relative distances measured along the rotating celestial body!

Tip: The cross range is defined positive when the ground track turns left (port). In order to maximize the cross range in the right direction (starboard) the scale factor must be negative. Similar, in order to minimize the down range the scale factor must be negative.

Configuration

The generic settings are described in Chapter 7.

COST FUNCTIONS

Background

All distances are computed as angles in the inertial frame with the central body as origin.

7.5 Max DeltaV

Description

This terminal cost term maximizes the velocity increment (delta-V) of the vehicle. It is useful for ascent trajectories with maximized final velocity increment, for example. Two *Subtypes* are available:

- *Total* is the total delta-V including the losses due to drag, gravity and steering
- *Loss Reduced* is the delta-V without the losses due to drag, gravity and steering. In this case, all losses have to be computed so that they can be subtracted from the total delta-V.

Total requires only one auxiliary state *Velocity increment (delta-V)* (see Section 4.5.3.2), while *Loss Reduced* requires further the auxiliary states *Drag Loss* (see Section 4.5.3.8), *Gravity Loss* (see Section 4.5.3.3) and *Steering Loss* (see Section 4.5.3.4).

Configuration

The generic settings are described in Chapter 7.

Background

The objective function is defined as

$$\text{Cost Function Term}_i = \Delta V - L_G - L_D - L_T \quad (7.5)$$

where ΔV is the velocity increment computed from the auxiliary state. The drag loss L_D , gravity loss L_G and steering loss L_T can be subtracted optional depending on the setting of *Subtype*.

7.6 Max Initial Value, Max Final Value, Max Initial Delta, Max Final Delta

Description

These cost functions serve to maximize either a particular value or the difference to a specified reference value at the beginning (initial cost) or at the end (final, i.e. terminal cost) of a phase, in particular:

- *Max Initial Value* maximizes the value of a function at the beginning of a phase
- *Max Final Value* maximizes the value of a function at the end of a phase

COST FUNCTIONS

- *Max Initial Delta* maximizes the difference between the value of a function and a specified reference value at the beginning of a phase
- *Max Final Delta* maximizes the difference between the value of a function and a specified reference value at the end of a phase

One of the following functions (i.e. physical quantities) can be specified as *Subtype*:

- *Airpath Velocity* is the magnitude of the air path velocity vector
- *Altitude* is the planeto-detic altitude (normal to surface) considering the radius of the sub-latitude point
- *Angular Momentum* is the magnitude of the orbital angular momentum vector, the cross product of the position and velocity vector
- *Apoapsis Radius* is the radius of the apoapsis
- *Ascending Node* is the right ascension of the ascending node, the angle between the x-axis of the inertial coordinate system and the ascending node of the orbit
- *Axial Acceleration* is the acceleration in axial direction ($+x_B$ -axis of the vehicle's body-fixed (B) frame) due to aerodynamic and propulsive forces
- *Distance* is the distance to target coordinates on the reference ellipsoid of the central body
- *Drag Force* is the force in drag direction ($-x_B$ -axis of the vehicle's body-fixed (B) frame)
- *Duration* is the duration of the phase
- *Dynamic Pressure* is the dynamic pressure of the vehicle
- *Eccentricity* is the eccentricity of the orbit
- *Excess Velocity* is the excess velocity, the velocity of the vehicle at infinity
- *Flightpath Angle* is the inclination angle of the flight path velocity vector
- *Flightpath Velocity* is the magnitude of the flight path velocity vector
- *Heat Flux* is the heat flux in the stagnation point of the vehicle
- *Inertial Velocity* is the magnitude of the inertial velocity vector
- *Latitude* is the planeto-detic latitude
- *Lift To Drag* is the lift to drag ratio
- *Load Factor* is the total aerodynamic load factor including drag, lift and side force
- *Longitude* is the (planeto-detic) longitude
- *Mach* is the ratio of the air path speed of the vehicle and the speed of the sound of the atmosphere
- *Mass* is the total mass of the vehicle
- *Model Parameter* defines a cost term on a specified parameter of the specified model (any vehicle part, e.g. actuator model). The parameter of the model must be present in the same phase as the cost term.
- *Normal Acceleration* is the acceleration in normal direction ($-z_B$ -axis of the vehicle's body-fixed (B) frame) due to aerodynamic and propulsive forces
- *Orbit Inclination* is the inclination angle of the orbit, the angle between the x-y plane of the inertial coordinate system and the orbital plane
- *Orbital Period* is the period of the osculating orbit

COST FUNCTIONS

- *Periapsis Radius* is the radius of the periapsis
- *Propellant Mass* is the total amount of all propellant masses
- *Radius* is the radius, i.e. the distance to the center of the central body
- *Semi Major Axis* is the semimajor axis of the orbit
- *Specific Energy* is the specific energy of the orbit, the sum of the potential energy and the total kinetic energy
- *Time* is the mission time

Configuration

Generic cost term settings are described in Chapter 7 and further specifications are:

- *Reference* specifies a reference value of the function. It is only available for *Max Initial Delta* and *Max Final Delta*. In this case, the cost term is the difference to this reference value.
- Further specifications are required for the following *Subtypes* only:
 - *Distance* requires the coordinates of the target location on the reference ellipsoid of the central body:
 - *Longitude* is the planeto-detic longitude
 - *Latitude* is the planeto-detic latitude
 - *Heat Flux* requires the optional *Aerothermodynamics* model (see Section 3.5.2). If *custom*, the corresponding identifier of the aerothermodynamics model has to be chosen, otherwise the *default* model is used.
 - *Model Parameter* has these settings:
 - *Model Parameter Type* is the kind of the model:
 - *Aerodynamics*
 - *Component*
 - *Propulsion*
 - *Name* is an identifier and must be identical to one of the real parameters defined by one of the models selected by *Model Parameter Type*. This identifier is exactly the same as listed in the *Result Summary* (*Results* tree) or in the *Real Parameters* (*Optimization* tree).

7.7 Max Final Mass

Description

COST FUNCTIONS

The *Max Final Mass* terminal cost term maximizes the total mass of the vehicle at the final time of a phase.

Configuration

The generic settings are described in Chapter 7.

Background

It is defined as

$$\text{Cost Function Term}_i = -\text{TotalMass}(\text{FinalTime}) \quad (7.6)$$

7.8 Max Inclination Delta

Description

This terminal cost term is used to maximize the difference in inclination of two orbits into which two different payloads with fixed masses are injected successively (first and last). At least two payload components (see Section 3.3.3.2) need to be defined in the *Components* branch of the *Modelling* tree.

See also the cost term *Max Ascending Node Delta* in Section 7.3.

Configuration

The generic settings are described in Chapter 7.

Background

The first payload must be jettisoned at an intermediate phase boundary, where usually also a set of orbit conditions is specified as intermediate boundary constraints. The last payload is always delivered to the target orbit defined by the final boundary constraints. To maximize the inclination difference, the inclination of at least one of the two target orbits must be unconstrained. A third payload jettisoned between the first and the last one is not considered. The cost function is defined as

$$\text{Cost Function Term}_i = \text{Inclination}(\text{Orbit1}) - \text{Inclination}(\text{Orbit2}) \quad (7.7)$$

The optimizer minimizes the objective function; this leads to the maximization of the difference between the two orbits only if the second inclination is bigger than the first one, otherwise the optimizer reduces the difference between the two orbits. In this second case, a negative scaling factor yields the maximization of the inclination difference.

7.9 Min Initial Propellant

Description

This terminal cost term minimizes the total propellant mass of the vehicle at the beginning of a phase. Hence the filling ratio of all tanks is minimized.

Configuration

The generic settings are described in Chapter 7.

Background

The cost function is given by

$$\text{Cost Function Term}_i = \sum_{i=1}^n \text{Ini\_Propellant}_i \quad (7.8)$$

where Ini_Propellant_i is the initial propellant mass of the i -th stage.

7.10 Max Payload

Description

This terminal cost term maximizes the total mass of all payload components of a vehicle.

Configuration

The generic settings are described in Chapter 7.

Background

It is defined as

$$\text{Cost Function Term}_i = - \text{Payload Scaling Factor} \cdot \text{NominalPayloadMass} \quad (7.9)$$

where the nominal payload mass is specified in the payload components of the *Components* branch of the *Modelling* tree (see Section 3.3). The payload scaling factor is a design parameter (i.e. real parameter) to be optimized. If the current payload mass is increased, the lift-off mass increases accordingly.

7.11 Max Safety

Description

The *Max Safety* terminal cost term maximizes the safety of the trajectory, i.e. it maximizes the distance to the vehicle constraints in the drag-energy diagram. In particular, the average drag

COST FUNCTIONS

deviation is minimized. Thus the auxiliary state *Average_Drag_Deviation* must be specified (see Section 4.5.3.6). This cost is useful for re-entry scenarios.

Configuration

The generic settings are described in Chapter 7.

Background

The cost function is calculated as the surface between the average drag profile and a reference drag profile. The constraints that are used to calculate a reference drag profile are: load factor, dynamic pressure, heat flux and equilibrium glide. More details are found in [53] and [54].

7.12 Max Splash Down

Description

The splash down is the impact of a re-entry vehicle or of a jettisoned assembly and the corresponding terminal cost term depends on the chosen *Subtype*:

- *Distance* maximizes/minimizes the total angular distance between the impact location and a specified target location
- *Latitude* maximizes/minimizes is the planeto-detic latitude of the impact location
- *Longitude* maximizes/minimizes the (planeto-detic) longitude of the impact location

Configuration

The generic settings are described in Chapter 7. Only in case of *Subtype Distance* these specifications are required as well:

- *Longitude* is the planeto-detic latitude of the target location
- *Latitude* is the (planeto-detic) longitude of the target location

Background

The impact location on the reference ellipsoid of the central body is analytically computed for the specified vehicle and at the end of the specified phase taking into account the current translational state of the vehicle.

7.13 Min Heatload

Description

COST FUNCTIONS

The terminal cost term *Min Heat Load* minimizes the integrated heat load. It is useful for re-entry applications. This cost term requires the auxiliary state *Heat load* (see Section 4.5.3.5).

Configuration

The generic settings are described in Chapter 7.

Background

The cost function is given by

$$\text{Cost Function Term}_i = -\int Q dt \quad (7.10)$$

7.14 Min Loss

Description

The *Min Loss* terminal cost term minimizes the losses in the velocity increment (delta-V) due to drag, gravity and/or steering. It is useful for ascent trajectories. In order to use this cost term the auxiliary states *Drag_Loss* (see Section 4.5.3.8), *Gravity_Loss* (see Section 4.5.3.3) and *Steering_Loss* (see Section 4.5.3.4) are required.

Configuration

The following specifications are required:

- *Drag Loss Weight* is the weight/scaling factor of the drag losses
- *Gravity Loss Weight* is the weight/scaling factor of the gravity losses
- *Steering Loss Weight* is the weight/scaling factor of the steering losses

Background

The cost function is defined by

$$\text{Cost Function Term}_i = \text{Drag\_Loss\_Weight} \cdot L_D + \text{Gravity\_Loss\_Weight} \cdot L_G + \text{Steering\_Loss\_Weight} \cdot L_T \quad (7.11)$$

with the drag loss L_D , the gravity loss L_G and the steering loss L_T .

7.15 None

Description

This cost term does not add anything to the objective function of the optimal control problem. In case this is the only specified cost term, the optimization is forced to meet the constraints only. This might be useful in very difficult problems (by means of optimization) when the initial

COST FUNCTIONS

guess must be improved before computing an optimal trajectory. Moreover, it is internally automatically created in case no cost function is specified at all.

Configuration

The settings are generic and are described in Chapter 7.

7.16 Max Smoothness

Description

The *Max Smoothness* cost term type is used to maximize the smoothness of one of the following quantities below by introducing an additional term to the scenario objective function. The smoothness cost function value itself is minimized in order to maximize the smoothness.

- *Trajectory* introduces a trajectory smoothness term which is equal to the product of the auxiliary state *Trajectory Smoothness* (see Section 4.5.3.7) and a user defined scaling factor. This cost function is modeled as terminal cost.
- *Attitude control* introduces a max attitude control smoothness term which is modeled as a Lagrange cost (an integrated cost term) and is equal to the product of the attitude control derivatives and a user specified scaling factor.

Note: It is not suggested to use the attitude control smoothing cost functions in combination with the SOS optimization suite, since the latter already automatically applies a smoothing factor to all control derivatives. If this cost function is specified with a non-zero scaling factor despite of this fact, this could lead to an SOS error code IER = -1287.

- *Drag acceleration* introduces a drag acceleration smoothness term which is modeled as Lagrange cost (an integrated cost term) and is equal to the product of the drag acceleration function and the user specified scaling factor. This cost function term is useful for re-entry scenarios where a smooth drag acceleration profile is desired.

Note: The *TROPIC* optimization method does not support the *Attitude control* and the *Drag acceleration* cost functions since it does not support Lagrange cost functions in general.

Configuration

The cost functions' settings are generic and are described in Chapter 7.

Background

The user shall take care that the smoothness cost function terms have a smaller contribution to the objective function than all other cost function terms by specifying a relatively small scaling factor compared to the scaling factor of the other cost function terms.

COST FUNCTIONS

In case of a trajectory smoothness cost function, the final value of the auxiliary state can be used to find a proper scaling factor. For an attitude control smoothness cost function however there is no such possibility. In this case, it is advisable to start with a small value (e.g. 0.001) for the scaling factor and then to increase its magnitude step by step during several optimization runs until the desired smoothness is achieved.

The proper scaling factor value can be also estimated based on the maximum value of the corresponding Lagrange cost auxiliary function, which is available in *Curve Plots* after a simulation action has been performed. The actual name of the auxiliary function depends on the defined Lagrange cost functions. Possible names are:

- `Attitude_Control_Smoothing`,
- `Drag_Accel_Smoothing` or
- `Multiple_Lagrange_Cost` in case of multiple Lagrange costs

The scaling of several Lagrange cost terms can only be estimated by setting all Lagrange costs except one to zero, by simulating and finally by analysing the auxiliary function value. After all maximum Lagrange cost values have been found, the proper scaling factors can be applied to each individual Lagrange cost term.

7.17 User Model

Description

Use this cost function term if you have defined a cost function within your user-defined model (see Section 11.1.6 for details). This terminal cost function term can be also used for models including a cost function like Throat Design Rocket (refer to Section 3.2.2.5).

Configuration

The generic settings are described in Chapter 7. Furthermore the following setting is required:

- *Type* is the kind of the model with these options:
 - *Actuator*
 - *Aerodynamics*
 - *Component*
- *Identifier* is the identifier of the model

8 Error Modelling

The **Error Modelling** feature enables the user to apply configurable error models to the output generated by actuators and sensors models on board a vehicle. This allows for a validation of the design robustness with respect to uncertainties and deviations from the nominal characteristics and development models. The **Error Modelling** feature can be activated only if the *Closed-Loop Simulation* feature is active. Both features can be activated in the root node of the *Modelling* tree.

Note: Computations involving this model require the corresponding feature to be present in the *ASTOS* license.

Configuration

When the **Error Modelling** feature is enabled, the *Error Modelling* setting can be enabled in the following models:

■ Sensor:

- *Camera*
- *Generic Sensor*
- *Inertial Measurement Unit*
- *Magnetometer*
- *Sun Sensor*

■ Actuator:

- *Magnetorquer*
- *Rocket Engine* subtypes *Basic Design*, *Linear Design*, *Throttleable* and *Typical VMR*

For any of these models the *Error Modelling* setting has the following parameters:

- *Sequence* defines in which order the error sources are applied to a signal. The following sequences are available:
 - *Scaling Bias Noise Saturation* means that the signal will first be scaled and after that the bias, noise and saturation are applied.

ERROR MODELLING

- *Bias Scaling Noise Saturation* means that first the bias will be applied and then the scaling, noise and saturation are applied.
- *Bias Noise Scaling Saturation* means that first the bias will be applied and then the noise, scaling and saturation are applied.

The error models in the list below contain common error sources, which are explained further below.

■ *Scaling* contains the following error sources:

- *Constant*
- *Run-to-run random range*
- *Stability*

■ *Bias* contains the following error sources:

- *Constant*
- *Run-to-run random range*
- *Stability*

■ *Noise* contains the following error sources:

- *Band-limited white noise*
- *White noise power*
- *Random walk*
- *Bias flicker*
- *Quantization*

■ *Saturation* has the following parameters:

- *Lower bound*
- *Upper bound*

■ *Transient* configuration is a complete second set of error sources for the model. Once the transient configuration has been activated, it is used for a defined duration by the simulation. The parameters of the transient configuration are give below and have the same meaning as the parameters with the same name in the list above.

- *Scaling*
- *Bias*
- *Noise*
- *Saturation*

The error sources of the error types given above are explained in the following.

ERROR MODELLING

- The *run-time variable* check-box is available for almost all of the sub-settings listed below. Therefore it is explained here first.
- The *Constant* error source is available for *Bias* and *Scaling*. If it is enabled for the *Bias*, a user-defined value is added to the signal. If it is enabled for the *Scaling*, the signal is multiplied by the user-defined value.
- The *Run-to-run random range* error source is similar to the *Constant* error source except that the user is able to specify the range of the error. A random number generator generates the constant scaling factor or bias within the range defined by *Lower bound* and *Upper bound*. The generated value is constant during the complete simulation run. An individual *Seed* can be provided by the user.
- The *Stability over Time* is implemented as pink noise configured by the filter *Coefficient B*. If the error source is applied to the *Bias* the noise is directly added to the signal. If it is applied to the *Scaling* the noise signal is increased by one before multiplying it with the signal.
- *Stability over Temperature* and *Stability over Temperature gradient* are implemented as third order polynomials, whose coefficients have to be specified by the user. Both polynomials are available for *Bias* and *Scaling*.
- The *Band-limited white noise* is defined by a *Standard deviation*, *Cut off frequency* of the noise and an optional *Seed* for the random number generator. The *Band-limited white noise* can only be realized for a certain range of the frequencies. This range depends on the simulation sample frequency $f_s = 20\text{Hz}$. For a *cut off frequency* f_c below the Nyquist frequency $f_n = f_s/2 = 10\text{Hz}$ of the simulation (i.e. $f_s/2 > f_c$), the noise signal is realized using a digital filter with the respective cut-off. If the cut-off frequency is chosen to be larger than the simulation sampling frequency ($f_c > f_s$), the cut-off has no effect. Thus the power spectrum becomes effectively white with the same power level as the originally band-limited spectrum. This correctly reduces the *effective* standard deviation compared to the *Standard deviation* parameter defined in the GUI.

Note: It is not recommended to specify a cut-off frequency between $f_s/2 < f_c < f_s$, as no digital filter can be realized in this frequency range. In this case, the power level defined by the band-limited spectrum is simply maintained without applying any filter. As this effectively increases the effective standard deviation in the signal, either the cut-off frequency or the standard deviation parameter may be manually adapted.

- *White noise power* represents a noise with a constant magnitude defined by the *Coefficient N*.
- The *Random walk* is implemented as a *Band-limited white noise* which is scaled by the *Coefficient K*.
- The *Bias flicker* is implemented as pink noise configured by the filter *Coefficient B*.

Note: A flicker of a quantity is generally defined by a $1/f$ power spectrum (i.e. $1/\sqrt{f}$ amplitude spectrum).

- *Quantization* is implemented as violet noise configured by the filter *Coefficient Q* and the *Quantization Period T*.

ERROR MODELLING

Note: Quantization of a quantity is generally defined by an f^2 power spectrum (i.e. an amplitude spectrum linearly increasing with frequency f).

9 Coordinate Systems and Transformations

In this chapter are introduced and explained the reference frames and coordinate systems, the vector representations, the physical quantities and symbols which are required to describe the motion of vehicles in atmospheric and space flight. Most terms used and the coordinate systems notations are based on the *International Standard ISO 1511* [2], on the *American National Standard ANSI/AIAA R-004-1992* [3] and on the *Coordinate Systems and Orthogonal Transformations* [4] where standardized definitions are available. The definitions of physical quantities and symbols used in this document follow the *Recommended Practice* [3]. Additional terms and symbols have been adopted from widely recognized literature in the aerospace field [31]-[37].

The content in this chapter is organized as follows:

- Section 9.1 **Definition of terms** contains definitions of the terms used in the explanations in this chapter
- Section 9.2 **Notation** explains the notation for vectors and transformation matrices
- Section 9.3 **Coordinate systems** provides explanations of the reference frames and coordinate systems used in the software
- Section 9.4 **Conversions and Transformation** contains explanations on what operations are available to change coordinate systems as well as to compute velocities relative to moving reference frames. This section gives a summary of common direction cosine matrices.

Note: The terms *coordinate system/frame*, *axis system* and *reference frame* are used often interchangeably in this help when used in a common sense. When used in a more specific sense, the terms *coordinate system/frame* imply that the orientation of a set of coordinate axes and their origin is important while the terms *axis system* and *reference frame* imply that only the orientation of some two axis systems with respect to each other is important.

9.1 Definition of terms

Terms defined by *International Standard ISO 1511* [2] or by the *American National Standard ANSI/AIAA R-004-1992* [3] are printed in italics.

Table 9.1: Vehicle position as a set of Keplerian elements

Symbol	Explanation
Ω	Right ascension of the ascending node
i	Angle of inclination
ω	Argument of the periapsis
a	Semi-major axis
e	Eccentricity
v	True anomaly (measured from the periapsis)

Table 9.2: Vehicle position as a set of modified equinoctial elements

Symbol	Explanation
p	Equinoctial element p (semilatus rectum)
f	Equinoctial element f
g	Equinoctial element g
h	Equinoctial element h
k	Equinoctial element k
L	Equinoctial element L (true longitude)

Table 9.3: Vehicle position w.r.t. a central body

Symbol	Explanation
δ	Declination
Φ_G	Planetodetic latitude (see Fig. 2.2)
Λ	Right ascension
λ	Longitude
Ω_E	Angular velocity (magnitude) of the planet
Δt	Elapsed time since beginning of the movement
Θ_{TOD}	Sidereal time
$\vec{\Omega}_E$	Angular velocity vector of the planet
$\hat{R}$	Distance to the planet center

Table 9.4: Components of the flight path velocity and of the air path velocity

Symbol	Explanation
V_k	Flight path speed / Planeto-relative speed
γ	Flight path inclination angle / Flight path angle
χ	Flight path azimuth angle / Heading
V_a	Air path speed / Airspeed

COORDINATE SYSTEMS AND TRANSFORMATIONS

Symbol	Explanation
γ_a	Air path inclination angle / Air path angle
χ_a	Air path azimuth angle / Air path heading
μ_a	Air path bank angle
V_w	Wind speed / Speed of the air relative to the planet
χ_w	Wind heading / Heading of the wind relative to north
γ_w	Wind inclination angle / Inclination of the wind relative to the horizontal plane
γ_i	Inertial velocity angle / Inclination between the LO and TO Frame (inertial)

Table 9.5: Vehicle orientation w.r.t. vehicle-carried coordinate systems

Symbol	Explanation
Ψ	Yaw angle
Θ	Pitch angle
Φ	Roll angle

Table 9.6: Vehicle orientation w.r.t. air path velocity

Symbol	Explanation
α	angle of attack
α_t	Total angle of attack
β	Sideslip angle
Φ'	Aerodynamic roll angle
μ_t	Aerodynamic bank angle

Table 9.7: Aerodynamic forces and moments

Symbol	Explanation
D	Drag
L	Lift
Y_a	Side force / Cross-stream force
A	Axial force
N	Normal force
Y	Lateral force
D_t	Total drag along total force axis
L_t	Total lift along total force axis
A_t	Total axial force along total angle of attack axis
N_t	Total normal force along total angle of attack axis
l	Moment about body-fixed longitudinal axis
m	Moment about body-fixed lateral axis
n	Moment about body-fixed normal axis
l_t	Moment about total angle of attack x-axis

Symbol	Explanation
m_t	Moment about total angle of attack y-axis
n_t	Moment about total angle of attack z-axis

9.2 Notation

In this section the mathematical notation for working with vectors and matrices used in *ASTOS* is explained. It follows the recommendations of the International Organization of Standardization.

Vectors

Vectors are commonly represented by Cartesian or spherical coordinates. A conversion procedure exists between the different representations which strongly depends on the coordinate frame and the physical quantity (i.e. position or velocity). Elevation and heading angles, E and H , are defined according to the engineering definition of spherical coordinates as shown in Fig. 9.1. The conversion from polar coordinates (R, E, H) to Cartesian coordinates (x, y, z) is given by:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = R \begin{bmatrix} \cos E \cos H \\ \cos E \sin H \\ \sin E \end{bmatrix} \quad (9.1)$$

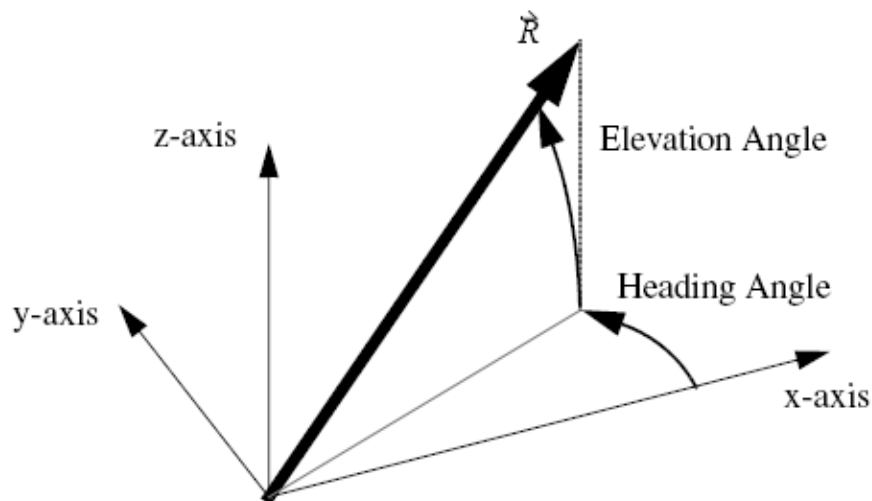


Fig. 9.1: Definition of heading and elevation angles in polar vector representation

The polar planetodetic coordinates consider the perpendicular to the local horizon (sub-latitude point) instead of the radial direction which is used in polar coordinates.

Matrices

The standard notation for a transformation matrix is $\mathbf{T}$. In this case, the matrix represents a single unit-axis rotation then it is called a direction cosine matrix. In case the transformation matrix is a product of three combined direction cosine matrices about different axes, then it is called a full transformation matrix. Typical matrix notations are summarized in the following table.

Table 9.8: Notations

$\mathbf{T}^{-1}$	Inverse of a transformation matrix
$\mathbf{T}^T$	Transpose of a transformation matrix
$\mathbf{I}$	Identity matrix
$\mathbf{T}_A^B$	Direction cosine matrix for transforming a vector from axis system A coordinates into axis system B coordinates
$\mathbf{T}_1(i)$	Transformation matrix to compute the new vector coordinates after a rotation about axis 1 by an angle i

9.3 Coordinate systems

This chapter provides a description of the coordinate/axis systems which may be used when setting up an *ASTOS* scenario. Each coordinate system has a unique name and subscript abbreviation (index). An overview of the available coordinate systems is given in Table 9.9 .

Table 9.9: Coordinate systems overview

Index	Name	Category	Chapter
<i>A</i>	Air path coordinate system	Velocity-carried	Section 9.3.5.3
<i>AV</i>	Vertical air path coordinate system	Velocity-carried	Section 9.3.5.2
<i>B</i>	Body-fixed coordinate system	Body-axis	Section 9.3.3.1
<i>EA</i>	Euler angles coordinate system	Other	Section 9.3.6.1
<i>FL</i>	Launcher frame according to ROSI definition [83]	Other	Section 9.3.6.2
<i>H</i>	Local horizontal coordinate system	Vehicle-carried	Section 9.3.4.3
<i>J2000</i>	Inertial J2000 equator coordinate system	Planetocentric	Section 9.3.2.1
<i>J2000Ec</i>	Inertial J2000 ecliptic coordinate system	Planetocentric	Section 9.3.2.2

COORDINATE SYSTEMS AND TRANSFORMATIONS

Index	Name	Category	Chapter
<i>L</i>	Local planetocentric coordinate system	Vehicle-carried	Section 9.3.4.1
<i>LD</i>	Local planetodetic coordinate system	Other	Section 9.3.4.2
<i>LO</i>	Local orbit coordinate system	Orbit-defining	Section 9.3.1.1
<i>N</i>	Inertial navigation coordinate system	Other	Section 9.3.6.4
<i>PF</i>	Perifocal coordinate system	Orbit-defining	Section 9.3.1.2
<i>PCPF</i>	Planet-centered planet-fixed coordinate system	Planetocentric	Section 9.3.2.4
<i>R-Bar_V-Bar</i>	R-Bar/V-Bar coordinate system	Other	Section 9.3.6.3
<i>S</i>	Sun-pointing coordinate system	Vehicle-Carried	Section 9.3.4.5
<i>T</i>	Trajectory coordinate system	Velocity-carried	Section 9.3.5.1
<i>TA</i>	Total angle of attack coordinate system	Body	Section 9.3.3.2
<i>TF</i>	Total force coordinate system	Body	Section 9.3.3.3
<i>TO</i>	Tangential orbit coordinate system	Orbit-defining	Section 9.3.1.3
<i>TOD</i>	Inertial True-Of-Date coordinate system	Planetocentric	Section 9.3.2.3
<i>V</i>	Local vertical coordinate system	Vehicle-Carried	Section 9.3.4.4

In the following section are provided the basic definitions required for the explanation of the coordinate/axis systems in the table above. A detailed explanation to each of these coordinate/axis systems is given in a separate section further below.

Coordinate system definitions at a glance

Orbit-defining coordinate systems

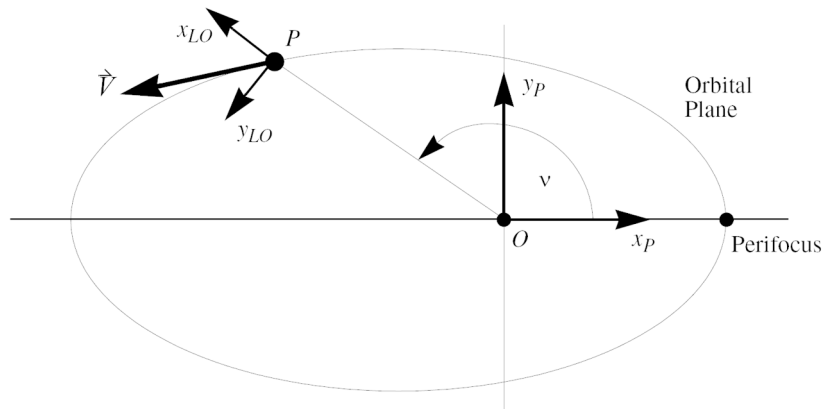


Fig. 9.2: Orbit-defining coordinate systems: (1) Perifocal coordinate system (index p) and (2) Local orbital coordinate system (index LO)

The vehicle is located at the origin P of the local orbital system (index LO). The origin of the perifocal system (index p) is the center of the planet O which is assumed to be at rest.

Relative motion coordinate system

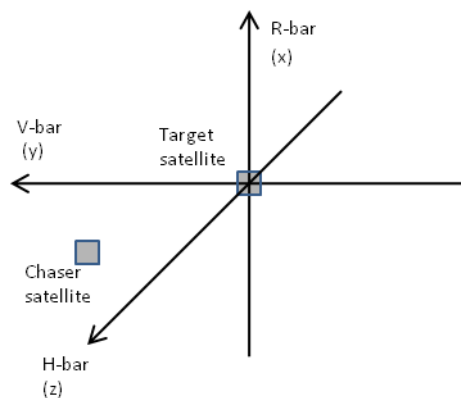


Fig. 9.3: Relative motion coordinate system

The vehicle is located at the origin of the reference frame. The x-axis (R-bar) is in line with the vector from the center of the planet and the CoM of the satellite, and points away from the planet. The z-axis (H-bar) is in line with the angular momentum of the orbit and the y-axis (V-bar) completes the right-handed frame. In low-eccentricity orbits, the V-bar usually coincides with the velocity vector, in higher eccentric orbits the vectors point in different directions.

Definition of the Keplerian elements

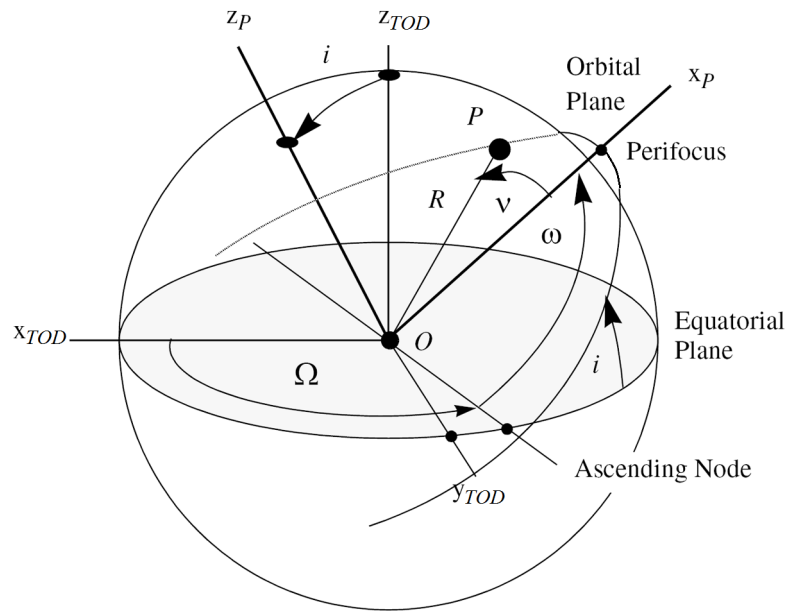


Fig. 9.4: Definition of the Keplerian elements in an inertial coordinate system (here: true of date) and the perifocal coordinate system

All angles in Fig. 9.4 are measured with respect to the planet-centered inertial axis-system (index $_{TOD}$). The orbit inclination angle i is the angle between the orbital plane and the equatorial plane.

Planet-centered and local horizontal coordinate systems

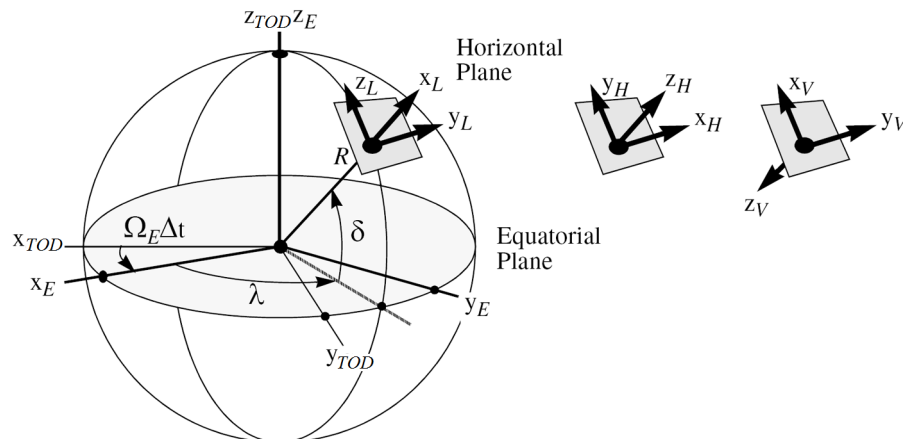


Fig. 9.5: Planet-centered and local horizontal coordinate systems: (1) True of date coordinate system (index $_{TOD}$), (2) Planet-centered planet-fixed coordinate system (index $_E$), (3) Local planetocentric coordinate system (index $_L$), (4) Local horizontal coordinate system (index $_H$), (5) Local vertical coordinate system (index $_V$)

The x-y plane of the inertial geocentric axis system (index $_{TOD}$) and of the rotating planet-fixed axis system (index $_E$) coincides with the planet equatorial plane, their z axes coincide with the planet's axis of rotation. The local horizontal axis system (index $_L$), the horizontal axis system (index $_H$), and the vertical axis system (index $_V$) are attached to the vehicle's center of mass.

Trajectory coordinate system

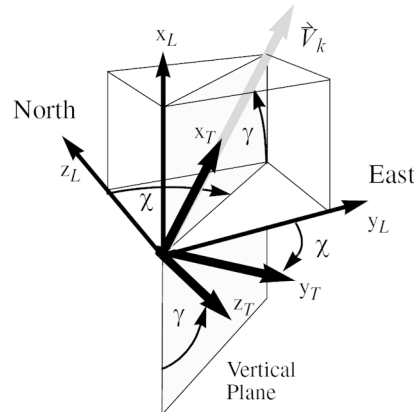


Fig. 9.6: The trajectory coordinate system (index τ)

The x_T and z_T axes lie in the local vertical plane. The x_T -axis is parallel to the planet-relative velocity vector. The flight path angle γ is the angle between the velocity vector and the local horizon and is positive for ascent. The heading angle χ lies in the local horizontal plane and is positive clockwise starting from the north.

Vertical air path coordinate system

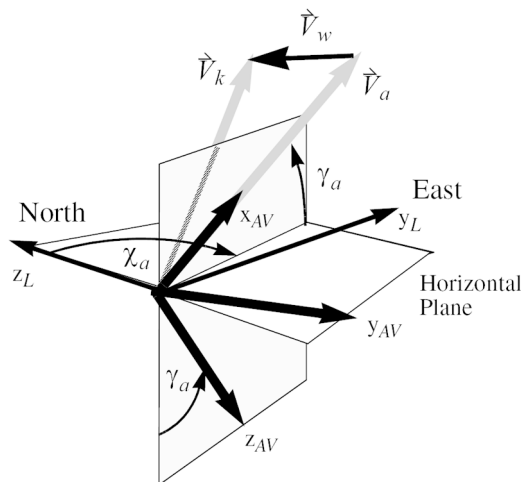


Fig. 9.7: Definition of the vertical air path coordinate system (index $_{AV}$)

The x_{AV} and z_{AV} axes lie in the local vertical plane.

Note: The x_{AV} is parallel to the air velocity.

The air path flight path angle γ_a is the angle between the air path velocity vector and the local horizon and is positive for ascent. The air path heading angle χ_a lies in the local horizontal plane.

Air path coordinate system

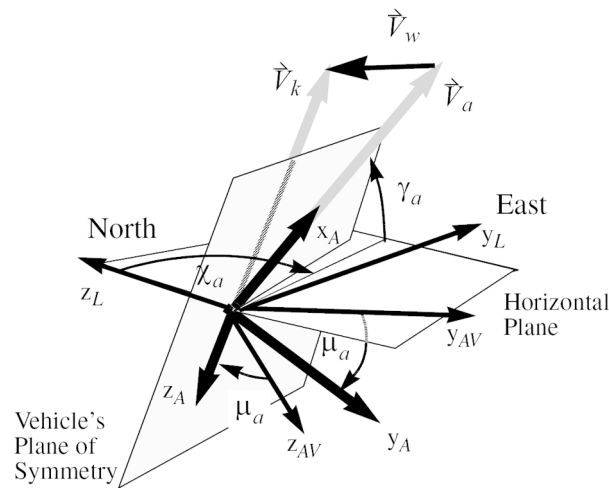


Fig. 9.8: Definition of the air path coordinate system (index A)

The x_A and z_A axes span the vehicle's plane of symmetry. The y_A -axis completes the x_A and z_A axis to a right handed system. The air path bank angle μ_a is the angle between the y_A -axis and the y_{AV} -axis which lie in a plane perpendicular to the air path velocity vector.

Body-fixed and Aerodynamic coordinate systems

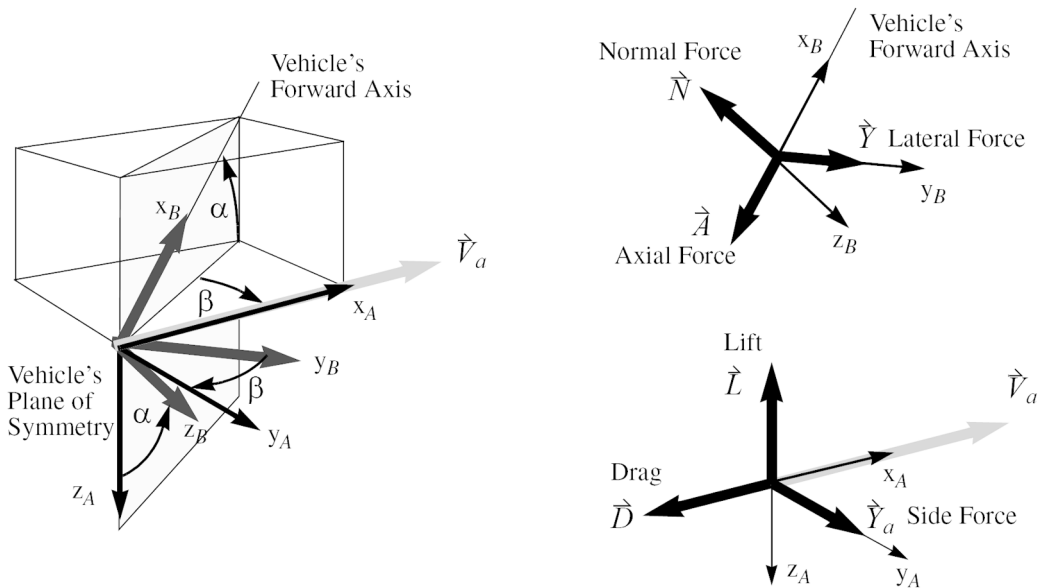


Fig. 9.9: Definition of the body-fixed coordinate system (index B) and aerodynamic angles

The normal, axial and lateral force are defined to be parallel to the body axes, whereas lift, drag and side force are along the air path velocity axes.

Euler angles in local planetocentric coordinate system

COORDINATE SYSTEMS AND TRANSFORMATIONS

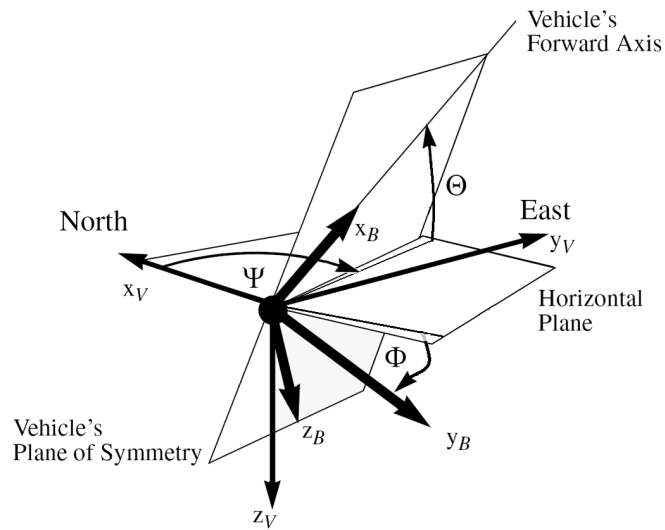


Fig. 9.10: Definition of the Euler angles in the local planetocentric coordinate systems

The Euler angles are a measure for the attitude of the vehicle-carried vertical system (*index* $_V$) with respect to the body-fixed axis system (*index* $_B$).

Total angle of attack coordinate system

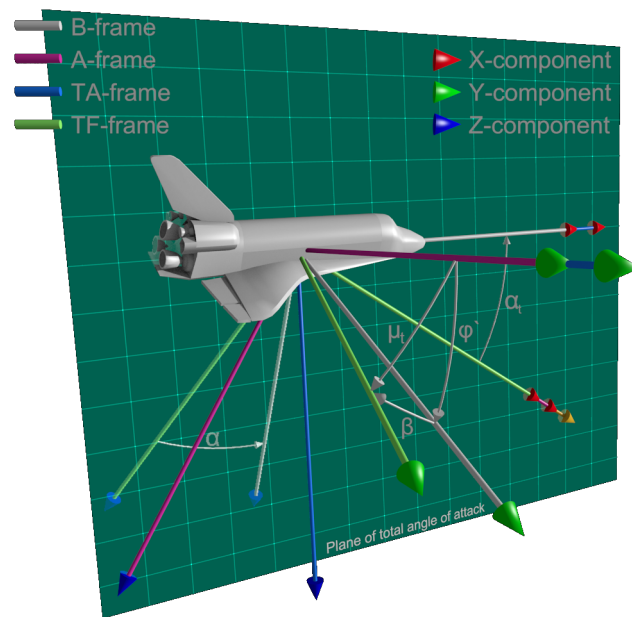


Fig. 9.11: Definition of the aerodynamics related coordinate systems and their angles: (1) air path coordinate system (*index* $_A$), (2) body-fixed coordinate system (*index* $_B$), (3) total angle of attack coordinate system (*index* $_{TA}$), (4) total force coordinate system (*index* $_{TF}$)

The total angle of attack is the angle between the vehicle's forward axis and the air path velocity vector. The total force coordinate system (*index* $_{TF}$) is aligned with the air path x_A axis, but rotated by the aerodynamic bank angle μ_t , such that the negative z_{TF} -axis is along

the total lift (the combination of lift and side force). The total angle of attack coordinate system ($index_{TA}$) is aligned with the body axis x_B axis, but rotated by the aerodynamic roll angle Φ' , such that the negative z_{TA} -axis is along the total lift force which is the vector sum of the lift and side forces. Both y_{TA} - and y_{TF} -axes are identical and point in the direction of no aerodynamic force to complete right-handed coordinate systems.

9.3.1 Orbit-defining axis systems

The orbit-defining coordinate system in this section are based on [31]-[33]. These coordinate systems are available:

- The **local orbit coordinate system (LO)** defines the local horizontal and the local vertical direction, see Section 9.3.1.1.
- The **perifocal coordinate system (PF)** is based on the orbital plane of the vehicle, see Section 9.3.1.2.
- The **tangential orbit coordinate system (TO)** is defined by the local vertical direction and the inertial velocity vector, see Section 9.3.1.3.

9.3.1.1 Local orbit (LO)

Description

The local orbit (LO) coordinate system defines the local vertical and the local horizontal direction (see Fig. 9.2). It is a non-inertial coordinate frame rotating with the inertial position vector of the vehicle and can be used for the attitude definition of orbital scenarios. The frame is defined as follows:

- the origin is the center of mass of the vehicle
- the fundamental plane is the orbital plane of the vehicle
- the x_{LO} -axis (radial) lies in the fundamental plane pointing in the same direction as the vehicle's radius vector
- the y_{LO} -axis (transverse) lies in the fundamental plane and completes the right-handed coordinate system (illustrated in Fig. 9.12)
- the z_{LO} -axis (normal) points in the direction of the angular momentum vector of the orbit

The local orbit coordinate system is also known as LVLH (local vertical, local horizontal), QSW, or RTN (radial, transverse, normal) [65], as well as RSW in [67].

Background

A set of Euler angles (yaw, pitch and roll) is required for the transformation from the local orbital frame (LO) to the body-fixed (B) coordinate system (refer to Section 9.3.3.1):

COORDINATE SYSTEMS AND TRANSFORMATIONS

- the yaw angle Ψ_{LO} is the out-of-orbit plane angle, defined positive in direction of the orbit angular momentum (see Fig. 9.13)
- the in-orbit plane angle pitch Θ_{LO} is measured in the x-y-plane from the y-axis (transverse direction) and is clockwise positive (illustrated in Fig. 9.12)
- the roll angle Φ_{LO} is the rotation about the vehicle's longitudinal axis (x-axis of the body-fixed frame)

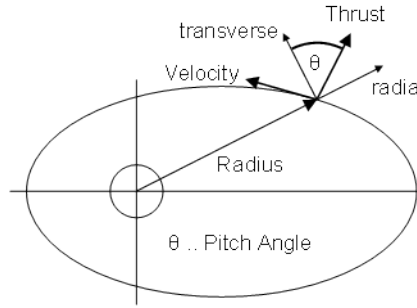


Fig. 9.12: Definition of the pitch $_{LO}$ angle in the x-y-plane (radial-transverse plane)

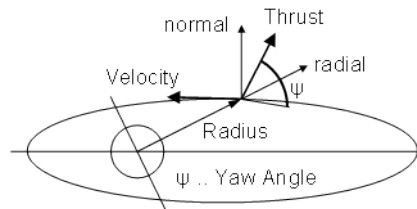


Fig. 9.13: Definition of the out-of-orbit plane angle yaw $_{LO}$

The transformation from the body-fixed (B) coordinate system to the local orbit (LO) system is defined by

$$\mathbf{T}_B^{LO} = \mathbf{T}_3\left(-\frac{\pi}{2} + \theta_{LO}\right) \mathbf{T}_2(\Psi_{LO}) \mathbf{T}_1(-\Phi_{LO}) \quad (9.2)$$

The transformation matrix from perifocal (PF) coordinate system (refer to Section 9.3.1.2) to the local orbit (LO) coordinate system is given by

$$\mathbf{T}_{PF}^{LO} = \mathbf{T}_3(v) \quad (9.3)$$

with the true anomaly v . The transformation from an inertial coordinate system to the local orbit (LO) coordinate system can be expressed by these orbital elements (see Fig. 9.4):

- the right ascension of the ascending node Ω_{TOD}
- the inclination of the orbital plane i_{TOD}
- the argument of periapsis ω_{TOD}
- the true anomaly v

COORDINATE SYSTEMS AND TRANSFORMATIONS

The subscript $_{TOD}$ means that those are with respect to the true of date (TOD) reference frame (refer to Section 9.3.2.3) and the direction cosine matrix to the local orbit (LO) system is defined by

$$\mathbf{T}_{TOD}^{LO} = \mathbf{T}_3(\omega_{TOD} + \nu) \mathbf{T}_1(i_{TOD}) \mathbf{T}_3(\Omega_{TOD}) \quad (9.4)$$

Similar, the transformation from the inertial planetocentric mean of J2000.0 (J2000) coordinate system (refer to Section 9.3.2.1) to the local orbit (LO) system is defined by

$$\mathbf{T}_{J2000}^{LO} = \mathbf{T}_3(\omega_{J2000} + \nu) \mathbf{T}_1(i_{J2000}) \mathbf{T}_3(\Omega_{J2000}) \quad (9.5)$$

The transformation from the planet-centered planet-fixed (PCPF) coordinate system (refer to Section 9.3.2.4) to the local orbit (LO) coordinate system is given by

$$\mathbf{T}_{PCPF}^{LO} = \mathbf{T}_{TOD}^{LO} \mathbf{T}_{PCPF}^{TOD} = \mathbf{T}_3(\omega_{TOD} + \nu) \mathbf{T}_1(i_{TOD}) \mathbf{T}_3(\Omega_{TOD} - \Omega_E \Delta t - \lambda_{i0}) \quad (9.6)$$

with the right ascension of the prime meridian ($\Omega_E \Delta t + \lambda_{i0}$) composed of the angular velocity of the planet Ω_E , the elapsed time since a reference epoch Δt and λ_{i0} , the angle between the vernal equinox (x-axis of TOD frame) and prime meridian at that reference epoch.

The transformation from the tangential orbit (TO) coordinate system (refer to Section 9.3.1.3) to the local orbit (LO) coordinate is defined by

$$\mathbf{T}_{TO}^{LO} = \mathbf{T}_3\left(-\frac{\pi}{2} - \gamma_i\right) \mathbf{T}_1\left(-\frac{\pi}{2}\right) \quad (9.7)$$

with the inertial velocity inclination angle γ_i .

9.3.1.2 Perifocal (PF)

Description

The origin of the non-inertial perifocal (PF) coordinate system is the center of the central body, which is assumed to be fixed in space (see Fig. 9.2). It is rotating with the position of the periapsis of the orbit as well as the orientation of the orbital plane. It is defined as follows:

- the origin is the center of the central body of the vehicle
- the fundamental plane is the orbital plane
- the x_{PF} -axis lies in the fundamental plane and points to the periapsis of the orbit
- the y_{PF} -axis lies in the fundamental plane and completes a right-handed system
- the z_{PF} -axis is parallel to the angular momentum vector of the orbit

Background

The transformation from the TOD coordinate system (refer to Section 9.3.2.3) to the perifocal (PF) system can be expressed using these orbital elements:

- the right ascension of the ascending node Ω_{TOD}
- the inclination of the orbital plane i_{TOD}

COORDINATE SYSTEMS AND TRANSFORMATIONS

- the argument of periapsis ω_{TOD}

The subscript $_{TOD}$ for the angles means that those are with respect to the TOD reference frame. The transformation from the inertial planetocentric true of date (TOD) coordinate system to the perifocal (PF) coordinate system is given by

$$\mathbf{T}_{TOD}^{PF} = \mathbf{T}_3(\omega_{TOD}) \mathbf{T}_1(i_{TOD}) \mathbf{T}_3(\Omega_{TOD}) \quad (9.8)$$

Similar, the transformation from the inertial planetocentric mean of J2000.0 (J2000) coordinate system (refer to Section 9.3.2.1) to the perifocal (PF) system is defined by

$$\mathbf{T}_{J2000}^{PF} = \mathbf{T}_3(\omega_{J2000}) \mathbf{T}_1(i_{J2000}) \mathbf{T}_3(\Omega_{J2000}) \quad (9.9)$$

The transformation matrix from the local orbit (LO) coordinate system (refer to Section 9.3.1.1) to the perifocal (PF) coordinate system is given by

$$\mathbf{T}_{LO}^{PF} = \mathbf{T}_3(-\nu) \quad (9.10)$$

with the true anomaly ν .

9.3.1.3 Tangential orbit (TO)

Description

The tangential orbit (TO) coordinate system is a non-inertial in-orbit coordinate system rotating with the inertial velocity vector of the vehicle along the orbit. It is currently only used for impulsive maneuvers (see Section 4.6). This coordinate system is defined as follows:

- the origin is the center of mass of the vehicle
- the fundamental plane is the orbital plane
- the x_{TO} -axis lies in the fundamental plane and is aligned with the velocity vector
- the y_{TO} -axis is in the direction opposite to the orbital angular momentum
- the z_{TO} -axis lies in the fundamental plane and completes the right-handed system being positive in opposite radial direction (approximately nadir)

The tangential orbit coordinate system is similar to the NTW coordinate system defined in [67], where the x_{TO} -axis corresponds to T, the y_{TO} -axis is opposite to W, and the z_{TO} -axis is opposite to N.

Background

The transformation from the local orbit (LO) coordinate system (refer to Section 9.3.1.1) to the tangential orbit (TO) coordinate is defined by

$$\mathbf{T}_{LO}^{TO} = \mathbf{T}_1\left(\frac{\pi}{2}\right) \mathbf{T}_3\left(\frac{\pi}{2} + \gamma_i\right) \quad (9.11)$$

with the inertial velocity inclination angle γ_i .

9.3.2 Planetocentric axis systems

The coordinate system definitions in this section are based on [3]. The definitions of the orbital elements are according to [32]. These coordinate systems are available:

- The **J2000 equator coordinate system (J2000)** is an inertial mean equator and mean equinox of epoch J2000.0 coordinate system, see Section 9.3.2.1.
- The **J2000 ecliptic coordinate system (J2000Ec)** is an inertial mean ecliptic and mean equinox of epoch J2000.0 coordinate system, see Section 9.3.2.2.
- The **true of date coordinate system (TOD)** is a quasi-inertial true equator and true equinox of date coordinate system, see Section 9.3.2.3.
- The **planet-centered planet-fixed coordinate system (PCPF)** rotates with the surface of a celestial body, see Section 9.3.2.4.

9.3.2.1 Mean Equator and Equinox of Epoch at J2000.0 (J2000)

Description

The inertial planetocentric coordinate system J2000 equator is a "Mean of Epoch at J2000.0" coordinate system, also known as "Mean of J2000". It is based on the mean equator and mean equinox of the Earth at the epoch J2000.0 (12:00 TT on 1 January 2000) and stays inertially fixed with respect to space. This coordinate system is defined as follows:

- the origin is the geometric center of mass of a celestial body which is assumed to be at rest (see Fig. 9.4)
- the fundamental plane is the mean Earth equator at epoch J2000.0
- the x_{J2000} -axis lies in the fundamental plane pointing to a direction fixed in space: the vernal equinox at epoch J2000.0. The vernal equinox is defined as the ascending node of the ecliptic on the Earth's mean equator at epoch J2000.0
- the y_{J2000} -axis lies in the fundamental plane and completes the right-handed coordinate system
- the z_{J2000} -axis is parallel to the mean angular momentum vector of the Earth at epoch J2000.0

Background

For the inertial frame J2000 equator the Euler angles are defined as standard rotation sequence:

1. positive rotation about inertial z-axis through the yaw angle Ψ_{J2000}
2. positive rotation about the intermediate y-axis through the pitch angle Θ_{J2000}
3. positive rotation about the intermediate x-axis through the roll angle Φ_{J2000} with x-axis coinciding with the x-axis of the body frame.

COORDINATE SYSTEMS AND TRANSFORMATIONS

and the direction cosine matrix from the body-fixed (B) coordinate system (refer to Section 9.3.3.1) to the inertial planetocentric mean of J2000.0 equator (J2000) coordinate system is defined by

$$\mathbf{T}_B^{J2000} = \mathbf{T}_3(-\psi_{J2000})\mathbf{T}_2(-\theta_{J2000})\mathbf{T}_1(-\phi_{J2000}) \quad (9.12)$$

The transformation from the perifocal (PF) coordinate system (refer to Section 9.3.1.2) to the mean of epoch J2000.0 equator (J2000) coordinate system can be expressed by these orbital elements (considering J2000 instead of TOD, see Fig. 9.4):

- the right ascension of the ascending node Ω_{J2000}
- the inclination of the orbital plane i_{J2000}
- the argument of periapsis ω_{J2000}

The subscript $_{J2000}$ for the angles means that those are with respect to the J2000 equator reference frame and the direction cosine matrix is defined by

$$\mathbf{T}_{PF}^{J2000} = \mathbf{T}_3(-\Omega_{J2000})\mathbf{T}_1(-i_{J2000})\mathbf{T}_3(-\omega_{J2000}) \quad (9.13)$$

The transformation from the local orbit (LO) coordinate system (refer to Section 9.3.1.1) to the mean of J2000.0 equator (J2000) coordinate system is given by

$$\mathbf{T}_{LO}^{J2000} = \mathbf{T}_3(-\Omega_{J2000})\mathbf{T}_1(-i_{J2000})\mathbf{T}_3(-\omega_{J2000} - \nu) \quad (9.14)$$

with the true anomaly ν .

From the mean of J2000.0 ecliptic (J2000Ec) coordinate system (refer to Section 9.3.2.2) to the mean of J2000.0 equator (J2000) coordinate system the direction cosine matrix is defined by

$$\mathbf{T}_{J2000Ec}^{J2000} = \mathbf{T}_1(-\varepsilon_0) \quad (9.15)$$

with the mean obliquity of the ecliptic ε_0 .

9.3.2.2 Mean Ecliptic and Equinox of Epoch at J2000.0 (J2000Ec)

Description

The inertial planetocentric coordinate system J2000 Ecliptic is a "Mean of Epoch at J2000.0" coordinate system, also known as "Mean of J2000". It is based on the mean ecliptic and mean equinox of the Earth at the epoch J2000.0 (12:00 TT on 1 January 2000) and stays inertially fixed with respect to space. This coordinate system is defined as follows:

- the origin is the geometric center of mass of a celestial body which is assumed to be at rest (see Fig. 9.4)
- the fundamental plane is the mean ecliptic at epoch J2000.0

COORDINATE SYSTEMS AND TRANSFORMATIONS

- the $x_{J2000Ec}$ -axis lies in the fundamental plane pointing to a direction fixed in space: the vernal equinox at epoch J2000.0. The vernal equinox is defined as the ascending node of the ecliptic on the Earth's mean equator at epoch J2000.0
- the $y_{J2000Ec}$ -axis lies in the fundamental plane and completes the right-handed coordinate system
- the $z_{J2000Ec}$ -axis is parallel to the mean angular momentum vector of the Earth orbit at epoch J2000.0

Background

For the inertial frame J2000 ecliptic the Euler angles are defined as standard rotation sequence:

1. positive rotation about inertial z-axis through the yaw angle $\Psi_{J2000Ec}$
2. positive rotation about the intermediate y-axis through the pitch angle $\Theta_{J2000Ec}$
3. positive rotation about the intermediate x-axis through the roll angle $\Phi_{J2000Ec}$ with x-axis coinciding with the x-axis of the body frame.

and the direction cosine matrix from the body-fixed (B) coordinate system (refer to Section 9.3.3.1) to the inertial planetocentric mean of J2000.0 (J2000) ecliptic coordinate system is defined by

$$\mathbf{T}_B^{J2000Ec} = \mathbf{T}_3(-\Psi_{J2000Ec}) \mathbf{T}_2(-\Theta_{J2000Ec}) \mathbf{T}_1(-\Phi_{J2000Ec}) \quad (9.16)$$

The transformation from the mean of J2000.0 equator (J2000) coordinate system (refer to Section 9.3.2.1) to the mean of J2000.0 ecliptic (J2000Ec) coordinate system is given by

$$\mathbf{T}_{J2000}^{J2000Ec} = \mathbf{T}_1(\varepsilon_0) \quad (9.17)$$

with the mean obliquity of the ecliptic ε_0 .

The transformation from Sun-pointing (S) coordinates (refer to Section 9.3.4.5) to the mean of J2000.0 ecliptic (J2000Ec) coordinate system is given by

$$\mathbf{T}_S^{J2000Ec} = \mathbf{T}_3(-\sigma) \mathbf{T}_2(\phi) \quad (9.18)$$

with σ and ϕ denoting the in-plane and out-of-plane transformation angles, respectively.

9.3.2.3 True Equator and Equinox of Date (TOD)

Description

The quasi-inertial planetocentric true of date (TOD) coordinate system is based either on the true equator and true equinox (in case of Earth) or on the true equator and the node of a celestial body (in case of another celestial body) at a particular epoch. "True" means that both the precession and nutation effects are included and computed at that epoch. This frame is also known as "True of Epoch" or "True of Date" and is defined as follows:

- the origin is the geometric center of mass of a celestial body which is assumed to be at rest (see Fig. 9.4)

COORDINATE SYSTEMS AND TRANSFORMATIONS

- the fundamental plane is the true equator of the celestial body at the current epoch
- the x_{TOD} -axis lies in the fundamental plane pointing to a direction fixed in space dependent on the celestial body:
 - in case of Earth it points to the dynamical equinox, which is the ascending node of the ecliptic on the Earth's true equator
 - in case of any other celestial body it points to the ascending node of the true equator on the ICRF/J2000 reference plane (x-y-plane)
- the y_{TOD} -axis lies in the fundamental plane and completes the right-handed coordinate system
- the z_{TOD} -axis is parallel to the true angular momentum vector of the celestial body at the current epoch

A TOD coordinate system is quasi-inertial and rotates due to precession and the nutation of the angular momentum vector of the celestial body. Within short timespans it can be considered inertially fixed.

Background

The transformation from the TOD coordinate system to the perifocal (PF) system (refer to Section 9.3.1.2) can be expressed using these orbital elements:

- the right ascension of the ascending node Ω_{TOD}
- the inclination of the orbital plane i_{TOD}
- the argument of periapsis ω_{TOD}

The subscript $_{TOD}$ for the angles means that those are with respect to the TOD reference frame. The right ascension of the ascending node Ω_{TOD} is the angle between the inertial planetocentric x_{TOD} -axis and the line where the equatorial TOD plane and the orbital plane intersect and where the vehicle has a velocity component pointing north (ascending node). The inclination i_{TOD} is the angle between the TOD equatorial plane and the orbital plane. The argument of periapsis ω_{TOD} angle is measured in the orbital plane from the ascending node to the periapsis. The transformation from the perifocal (PF) system to the true of date (TOD) coordinate system is defined by

$$\mathbf{T}_{PF}^{TOD} = \mathbf{T}_3(-\Omega_{TOD})\mathbf{T}_1(-i_{TOD})\mathbf{T}_3(-\omega_{TOD}) \quad (9.19)$$

The transformation from the local orbit (LO) coordinate system (refer to Section 9.3.1.1) to the true of date (TOD) coordinate system is given by

$$\mathbf{T}_{LO}^{TOD} = \mathbf{T}_3(-\Omega_{TOD})\mathbf{T}_1(-i_{TOD})\mathbf{T}_3(-\omega_{TOD} - \nu) \quad (9.20)$$

with the true anomaly ν .

The transformation from the planet-centered planet-fixed (PCPF) coordinates system (refer to Section 9.3.2.4) to the inertial true of date (TOD) coordinate system is defined by

$$\mathbf{T}_{PCPF}^{TOD} = \mathbf{T}_3(-\Omega_E \Delta t - \lambda_{i_0}) \quad (9.21)$$

COORDINATE SYSTEMS AND TRANSFORMATIONS

with the right ascension of the prime meridian ($\Omega_E \Delta_t + \lambda_{i0}$) composed of the angular velocity of the planet Ω_E , the elapsed time since a reference epoch Δ_t and λ_{i0} , the angle between the vernal equinox (x-axis of TOD frame) and prime meridian at that reference epoch.

The transformation from the local planetocentric (L) coordinate system (refer to Section 9.3.4.1) to the true of date (TOD) coordinate system is defined by

$$\mathbf{T}_L^{TOD} = \mathbf{T}_{PCPF}^{TOD} \mathbf{T}_L^{PCPF} = \mathbf{T}_3(-\lambda - \Omega_E \Delta_t) \mathbf{T}_2(\delta) \quad (9.22)$$

with the longitude λ and the declination δ . Similar, the transformation from the local horizontal (H) coordinate system (refer to Section 9.3.4.3) to the true of date (TOD) coordinate system is defined by

$$\mathbf{T}_H^{TOD} = \mathbf{T}_{PCPF}^{TOD} \mathbf{T}_H^{PCPF} = \mathbf{T}_3\left(-\frac{\pi}{2} - \lambda - \Omega_E \Delta_t\right) \mathbf{T}_1\left(-\frac{\pi}{2} + \delta\right) \quad (9.23)$$

and the transformation from the local vertical (V) coordinate system (refer to Section 9.3.4.4) to the true of date (TOD) coordinate system is defined by

$$\mathbf{T}_V^{TOD} = \mathbf{T}_{PCPF}^{TOD} \mathbf{T}_V^{PCPF} = \mathbf{T}_3(-\lambda - \Omega_E \Delta_t) \mathbf{T}_2\left(\frac{\pi}{2} + \delta\right) \quad (9.24)$$

The direction cosine matrix from the body-fixed (B) coordinate system (refer to Section 9.3.3.1) to the true of date (TOD) coordinate system is defined by

$$\mathbf{T}_B^{TOD} = \mathbf{T}_3(-\Psi_{TOD}) \mathbf{T}_2(-\Theta_{TOD}) \mathbf{T}_1(-\Phi_{TOD}) \quad (9.25)$$

with the Euler angles yaw Ψ_{TOD} , pitch Θ_{TOD} and roll Φ_{TOD} .

9.3.2.4 Planet-centered planet-fixed (PCPF)

Description

The planet-centered planet-fixed (PCPF) coordinate system rotates with the surface of a celestial body (see Fig. 9.5). It is defined as follows:

- the origin is the geometric center of mass of a celestial body which is assumed to be at rest (see Fig. 9.4)
- the fundamental plane is the true equator of the celestial body at the current epoch
- the x_{PCPF} -axis lies in the fundamental plane pointing to the prime meridian (defined as zero longitude)
- the y_{PCPF} -axis lies in the fundamental plane and completes the right-handed coordinate system
- the z_{PCPF} -axis is parallel to the true angular momentum vector of the celestial body at the current epoch

Background

The angular velocity of the planet is denoted by Ω_E , Δ_t is the elapsed time since a reference epoch and λ_{i0} is the angle between the vernal equinox (x-axis of TOD frame) and prime meridian at that reference epoch. Then the angle ($\Omega_E \Delta_t + \lambda_{i0}$) represents the right ascension of the prime meridian with respect to the inertial planetocentric system TOD (refer to Section 9.3.2.3). The

COORDINATE SYSTEMS AND TRANSFORMATIONS

transformation from the inertial true of date (TOD) coordinate system to the planet-centered planet-fixed coordinates (PCPF) is given by

$$\mathbf{T}_{TOD}^{PCPF} = \mathbf{T}_3(\Omega_E \Delta t + \lambda_{i_0}) \quad (9.26)$$

The complete transformation from local orbit (LO) coordinate system (refer to Section 9.3.1.1) to the planet-centered planet-fixed (PCPF) coordinate system is defined by

$$\mathbf{T}_{LO}^{PCPF} = \mathbf{T}_{TOD}^{PCPF} \mathbf{T}_{LO}^{TOD} = \mathbf{T}_3(\Omega_E \Delta t + \lambda_{i_0} - \Omega_{TOD}) \mathbf{T}_1(-i_{TOD}) \mathbf{T}_3(-\omega_{TOD} - \nu) \quad (9.27)$$

The angles λ and δ denote the vehicle east longitude and declination, respectively. The transformation from the local planetocentric (L) coordinate system (refer to Section 9.3.4.1) to the planet-centered planet-fixed (PCPF) coordinate system is defined by

$$\mathbf{T}_L^{PCPF} = \mathbf{T}_3(-\lambda) \mathbf{T}_2(\delta) \quad (9.28)$$

Similar, the transformation from the local horizontal (H) coordinate system (refer to Section 9.3.4.3) to the planet-centered planet-fixed (PCPF) coordinate system is defined by

$$\mathbf{T}_H^{PCPF} = \mathbf{T}_3\left(-\frac{\pi}{2} - \lambda\right) \mathbf{T}_1\left(-\frac{\pi}{2} + \delta\right) \quad (9.29)$$

and the transformation from the local vertical (V) coordinate system (refer to Section 9.3.4.4) to the planet-centered planet-fixed (PCPF) coordinate system is given by

$$\mathbf{T}_V^{PCPF} = \mathbf{T}_3(-\lambda) \mathbf{T}_2\left(\frac{\pi}{2} + \delta\right) \quad (9.30)$$

The direction cosine transformation matrix from the launcher (FL) coordinate system (refer to Section 9.3.6.2) to the planet-centered planet-fixed (PCPF) coordinates is given by

$$\mathbf{T}_{FL}^{PCPF} = \mathbf{T}_3(-\lambda_0) \mathbf{T}_2\left(\frac{\pi}{2} + \Phi_0\right) \quad (9.31)$$

with the initial longitude λ_0 measured from the prime meridian to the initial meridian of the vehicle and with the initial latitude Φ_0 being the angle between the equatorial plane and the normal to the surface of the ellipsoid at time zero (see Fig. 2.2).

9.3.3 Body-axis system

The body coordinate systems are located in the center of mass of a vehicle and at least one axis coincides with one of the principal axes of the vehicle:

- The **body-fixed coordinate system (B)** is defined by aerospace standards [3], see Section 9.3.3.1.
- The **total angle of attack coordinate system (TA)** is defined by aerospace standards [3], see Section 9.3.3.2.
- The **total force coordinate system (TF)** has been defined expressly for the ASTOS software, see Section 9.3.3.3.

9.3.3.1 Body-axes (B)

Description

The body-fixed (B) coordinate system is shown in Fig. 9.9 and Fig. 9.11. It is fixed to the vehicle structure and defined as follows:

- the origin is the center of mass of the vehicle
- the fundamental plane is the vehicle plane of symmetry
- the x_B -axis points to the front and spans the fundamental plane
- the y_B -axis points to starboard and is perpendicular to the fundamental plane
- the z_B -axis points down and spans the fundamental plane

Background

The angle of attack α and the side slip angle β determine the orientation of the vehicle forward axis with respect to the air path velocity (see Fig. 9.9). The transformation from the air path (A) coordinate system (refer to Section 9.3.5.3) to the body-fixed (B) coordinate system is given by

$$\mathbf{T}_A^B = \mathbf{T}_2(\alpha) \mathbf{T}_3(-\beta) \quad (9.32)$$

The transformation from the total angle of attack (TA) coordinate system (refer to Section 9.3.3.2) to the body-fixed (B) coordinates is given by

$$\mathbf{T}_{TA}^B = \mathbf{T}_1(\phi') \quad (9.33)$$

with the aerodynamic roll angle ϕ' . A definition of the aerodynamic roll angle is provided in Section 9.3.3.2.

The yaw angle Ψ , the pitch angle Θ and the roll angle Φ determine the vehicle orientation with respect to all other axis systems (see Fig. 9.10). Each set of Euler angles is unique for a specific coordinate system. The transformation matrix from the local vertical (V) system (refer to Section 9.3.4.4) to the body-fixed (B) coordinates is given by

$$\mathbf{T}_V^B = \mathbf{T}_1(\phi_L) \mathbf{T}_2(\theta_L) \mathbf{T}_3(\psi_L) \quad (9.34)$$

with the Euler angles of the local planetocentric (L) coordinate system, since the Euler angles of both systems are identical. The equivalent direction cosine matrix transforming the local planetocentric (L) system (refer to Section 9.3.4.1) to the body-fixed (B) system is given by

$$\mathbf{T}_L^B = \mathbf{T}_1(\phi_L) \mathbf{T}_2\left(-\frac{\pi}{2} + \theta_L\right) \mathbf{T}_1(-\psi_L) \quad (9.35)$$

The transformation matrix from the local planetodetic (LD) system (refer to Section 9.3.4.2) to the body-fixed (B) system is defined by

$$\mathbf{T}_{LD}^B = \mathbf{T}_1(\phi_{LD}) \mathbf{T}_2\left(-\frac{\pi}{2} + \theta_{LD}\right) \mathbf{T}_1(-\psi_{LD}) \quad (9.36)$$

The direction cosine matrix from the launcher (FL) coordinate system (refer to Section 9.3.6.2) to the body-fixed (B) system is defined by

$$\mathbf{T}_{FL}^B = \mathbf{T}_1(\phi_{FL}) \mathbf{T}_2(\theta_{FL}) \mathbf{T}_3(\psi_{FL}) \quad (9.37)$$

COORDINATE SYSTEMS AND TRANSFORMATIONS

The transformation matrix from the local orbit (LO) system (refer to Section 9.3.1.1) to the body-fixed (B) system is given by

$$\mathbf{T}_{LO}^B = \mathbf{T}_1(\phi_{LO}) \mathbf{T}_2(-\psi_{LO}) \mathbf{T}_3\left(\frac{\pi}{2} - \theta_{LO}\right) \quad (9.38)$$

For inertial frames, the direction cosine matrix from mean of epoch J2000.0 equator (J2000) coordinate system (refer to Section 9.3.2.1) to body-fixed (B) coordinates is defined by

$$\mathbf{T}_{J2000}^B = \mathbf{T}_1(\phi_{J2000}) \mathbf{T}_2(\theta_{J2000}) \mathbf{T}_3(\psi_{J2000}) \quad (9.39)$$

the direction cosine matrix from the mean of epoch J2000.0 ecliptic (J2000Ec) coordinate system (refer to Section 9.3.2.2) to body-fixed (B) coordinates is given by

$$\mathbf{T}_{J2000Ec}^B = \mathbf{T}_1(\phi_{J2000Ec}) \mathbf{T}_2(\theta_{J2000Ec}) \mathbf{T}_3(\psi_{J2000Ec}) \quad (9.40)$$

and the direction cosine matrix from true of date (TOD) (refer to Section 9.3.2.3) to body-fixed (B) coordinates is defined by

$$\mathbf{T}_{TOD}^B = \mathbf{T}_1(\phi_{TOD}) \mathbf{T}_2(\theta_{TOD}) \mathbf{T}_3(\psi_{TOD}) \quad (9.41)$$

Similar, but with an important difference, the transformation from a Euler angles (EA) coordinate system of a vehicle element (see Section 9.3.6.1) to the body-fixed (B) coordinate system is given by

$$\mathbf{T}_{EA}^B = \mathbf{T}_3(-\psi_{EA}) \mathbf{T}_2(-\theta_{EA}) \mathbf{T}_1(-\phi_{EA}) \quad (9.42)$$

where the Euler angles yaw ψ_{EA} , pitch θ_{EA} and roll ϕ_{EA} (often denoted as angle 1, angle 2 and angle 3, respectively) determine the vehicle element orientation w.r.t. the vehicle body axes.

Note: Here, the Euler angles of the EA coordinate system describe the attitude of a vehicle element wrt the vehicle body axes, whereas the Euler angles of the other coordinate systems mentioned before define the attitude of the vehicle body axes!

9.3.3.2 Total angle of attack (TA)

Description

The total angle of attack (TA) coordinate system (also called aeroballistic axis system) is an aerodynamic reference frame applied to space flight vehicles whose forward axis is an axis of rotational symmetry. The x-z plane of such a vehicle is arbitrarily defined by the designer and is called the total angle of attack plane. Consequently, for a vehicle of rotational symmetry the distinction between angle of attack and sideslip angle is negligible. The aerodynamic forces and moments acting on the vehicle depend only on the total angle of attack α_t between the body-fixed forward axis and the air path velocity (see Fig. 9.11). This coordinate system is defined as follows:

- the origin is the center of mass of the vehicle

COORDINATE SYSTEMS AND TRANSFORMATIONS

- the fundamental plane is the total angle of attack plane, the plane spanned by the vehicle's forward axis and the air path velocity vector
- The x_{TA} -axis lies in the fundamental plane and is collinear with the x_B -axis (body fixed coordinate system). It points in the direction opposite to the total axial aerodynamic force.
- The y_{TA} -axis points starboard and is perpendicular to the fundamental plane
- The z_{TA} -axis lies in the fundamental plane pointing in the opposite direction of the total normal aerodynamic force

As shown in Fig. 9.11, the x_{TA} -axis is identical to the x_B -axis of the body-fixed (B) coordinate system (see Section 9.3.3.1) and the y_{TA} -axis is a vector perpendicular to the total angle of attack plane and identical to the y_{TF} -axis of the total force (TF) coordinate system (see Section 9.3.3.3).

Background

The total angle of attack α_t can be expressed in terms of the angle of attack α and the sideslip angle β as follows:

$$\alpha_t = \arccos(\cos\alpha\cos\beta) \quad (9.43)$$

This expression holds for $-\pi \leq \alpha \leq \pi$ and $\beta \leq \pi$. The total angle of attack is always positive and the transformation from the total force (TF) system (refer to Section 9.3.3.3) to the total angle of attack (TA) system is given by

$$\mathbf{T}_{TF}^{TA} = \mathbf{T}_2(\alpha_t) \quad (9.44)$$

The angle between the y_{TA} -axis and the body-fixed y_B -axis is known as the aerodynamic roll angle ϕ' and is defined by the following expressions:

$$\sin\phi' = \frac{\sin\beta}{\sin\alpha_t} \quad \cos\phi' = \frac{\sin\alpha\cos\beta}{\sin\alpha_t} \quad (9.45)$$

The transformation from the body-fixed (B) coordinates (refer to Section 9.3.3.1) to the total angle of attack (TA) system is given by

$$\mathbf{T}_B^{TA} = \mathbf{T}_1(-\phi') \quad (9.46)$$

9.3.3.3 Total force (TF)

Description

The total force (TF) coordinate system is an aerodynamic reference frame aligned with the total aerodynamic lift and drag acting on a vehicle of rotational symmetry. This coordinate system is defined as follows:

- the origin is the center of mass of the vehicle
- the fundamental plane is the total angle of attack plane, the plane spanned by the vehicle's forward axis and the air path velocity vector

COORDINATE SYSTEMS AND TRANSFORMATIONS

- The x_{TF} -axis lies in the fundamental plane and is collinear with the air path velocity vector (x_A -axis of the air path coordinate system). It points in the direction opposite to the total aerodynamic drag force.
- the y_{TF} -axis points starboard and is perpendicular to the fundamental plane
- the z_{TF} -axis lies in the fundamental plane pointing in the opposite direction as the total aerodynamic lift force

As shown in Fig. 9.11, the x_{TF} -axis is identical to the x_A -axis of the air path (A) coordinate system (see Section 9.3.5.3) and the y_{TF} -axis is a vector perpendicular to the total angle of attack plane and identical to the y_{TA} -axis of the total angle of attack (TA) coordinate system (see Section 9.3.3.2).

Background

The magnitude of the total lift and drag are functions of the total angle of attack α_t defined as function of the angle of attack α and the sideslip angle β :

$$\alpha_t = \arccos(\cos\alpha\cos\beta) \quad (9.47)$$

This expressions holds for $-\pi \leq \alpha \leq \pi$ and $\beta \leq \pi$. The total angle of attack is always positive and the transformation from the total angle of attack (TA) system (refer to Section 9.3.3.2) to the total force (TF) system is given by

$$\mathbf{T}_{TA}^{TF} = \mathbf{T}_2(-\alpha_t) \quad (9.48)$$

The angle between the y_{TF} -axis and the y_A -axis is not defined by aerospace standards. It is denoted here as the aerodynamic bank angle μ_t given by

$$\sin\mu_t = \frac{\cos\alpha\sin\beta}{\sin\alpha_t} \quad \cos\mu_t = \frac{\sin\alpha}{\sin\alpha_t} \quad (9.49)$$

The transformation from the air path (A) coordinate system (refer to Section 9.3.5.3) to the total force (TF) system is given by

$$\mathbf{T}_A^{TF} = \mathbf{T}_1(-\mu_t) \quad (9.50)$$

9.3.4 Vehicle-carried axis systems

The vehicle-carried axis systems are used to specify position and velocity of the vehicle center of mass with respect to a celestial body. Their coordinate origins are attached to the vehicle (see Fig. 9.5). These coordinate systems are available:

- The **local planetocentric coordinate system (L)** is typically used in space flight mechanics [36], see Section 9.3.4.1.
- The **local planetodetic coordinate system (LD)** is typically used in space flight mechanics [36], see Section 9.3.4.2.
- The **local horizontal coordinate system (H)** is typically used in space flight mechanics [36], see Section 9.3.4.3.

- The **local vertical coordinate system (V)** is typically used in atmospheric flight mechanics [35], see Section 9.3.4.4.
- The **Sun-pointing coordinate system (S)** is based on the direction of the Sun and the ecliptic and is used to analyse potential sensor blinding, see Section 9.3.4.5.

9.3.4.1 Local planetocentric (L)

Description

The local planetocentric (L) coordinate system is defined as follows (see Fig. 9.5):

- the origin is the center of mass of the vehicle
- the fundamental plane is the local planetocentric horizontal plane
- the x_L -axis points upward in radius direction
- the y_L -axis lies in the fundamental plane pointing east along the local parallel
- the z_L -axis lies in the fundamental plane pointing north along the local meridian

This coordinate system has the same orientation as the local horizontal coordinate system H (see Section 9.3.4.3) and the local vertical coordinate system V (see Section 9.3.4.4), but the axes are labeled differently.

Background

The angles λ and δ denote the vehicle east longitude and declination, respectively. The transformation from planet-centered planet-fixed (PCPF) coordinates (refer to Section 9.3.2.4) to the local planetocentric (L) system is given by

$$\mathbf{T}_{PCPF}^L = \mathbf{T}_2(-\delta) \mathbf{T}_3(\lambda) \quad (9.51)$$

The transformation from inertial true of date (TOD) coordinates (refer to Section 9.3.2.3) to the local planetocentric (L) system is defined by

$$\mathbf{T}_{TOD}^L = \mathbf{T}_{PCPF}^L \mathbf{T}_{TOD}^{PCPF} = \mathbf{T}_2(-\delta) \mathbf{T}_3(\lambda + \Omega_E \Delta t) \quad (9.52)$$

The transformation from the trajectory (T) coordinate system (refer to Section 9.3.5.1) to the local planetocentric (L) system is given by

$$\mathbf{T}_T^L = \mathbf{T}_1(\chi) \mathbf{T}_2\left(\frac{\pi}{2} - \gamma\right) \quad (9.53)$$

with the flight path inclination γ and the flight path azimuth χ . For the transformation from the air path (A) coordinate system (refer to Section 9.3.5.3) to the local planetocentric (L) coordinate system the following transformation is used

$$\mathbf{T}_A^L = \mathbf{T}_1(\chi_a) \mathbf{T}_2\left(\frac{\pi}{2} - \gamma_a\right) \mathbf{T}_1(-\mu_a) \quad (9.54)$$

where γ_a and χ_a are the air path inclination and heading angles, respectively, and μ_a is the air path bank angle (see Fig. 9.8).

COORDINATE SYSTEMS AND TRANSFORMATIONS

The transformation from the vertical air path (AV) coordinate system (refer to Section 9.3.5.2) to the local planetocentric (L) coordinate system is defined by

$$\mathbf{T}_{AV}^L = \mathbf{T}_1(\chi_a) \mathbf{T}_2\left(\frac{\pi}{2} - \gamma_a\right) \quad (9.55)$$

The transformation from the body-fixed (B) coordinate system (refer to Section 9.3.3.1) to the local planetocentric (L) coordinate system is given by

$$\mathbf{T}_B^L = \mathbf{T}_1(\Psi_L) \mathbf{T}_2\left(\frac{\pi}{2} - \theta_L\right) \mathbf{T}_1(-\Phi_L) \quad (9.56)$$

where Ψ_L , θ_L and Φ_L denote the Euler angles yaw, pitch and roll, respectively.

The transformation from the local planetodetic (LD) coordinate system (refer to Section 9.3.4.2) to the local planetocentric (L) coordinate system is defined by

$$\mathbf{T}_{LD}^L = \mathbf{T}_2(\alpha) \quad (9.57)$$

with the angle α defined as the difference between the latitude (planetodetic latitude) and the declination (planetocentric latitude). A visual representation of this angle is presented in Fig. 2.2.

9.3.4.2 Local planetodetic (LD)

Description

The definition of the local planetodetic (LD) coordinate system is similar to the local planetocentric (L) axis system (see Section 9.3.4.1) but considers the perpendicular direction to the local planetodetic horizon (sub-latitude point) instead of the radial direction. It is defined as follows:

- the origin is the center of mass of the vehicle
- the fundamental plane is the local planetodetic horizontal plane
- the x_{LD} -axis points upward perpendicular to the fundamental plane
- the y_{LD} -axis lies in the fundamental plane pointing east along the local parallel
- the z_{LD} -axis lies in the fundamental plane pointing north along the local meridian

Background

The angle α is the difference between the latitude (planetodetic latitude) and the declination (planetocentric latitude). A visual representation of this angle is presented in Fig. 2.2. The transformation from the local planetocentric (L) coordinate system (refer to Section 9.3.4.1) to the local planetodetic (LD) coordinate system is given by

$$\mathbf{T}_L^{LD} = \mathbf{T}_2(-\alpha) \quad (9.58)$$

The transformation from the body-fixed (B) coordinate system (refer to Section 9.3.3.1) to the local planetodetic (LD) coordinate system is given by

$$\mathbf{T}_B^{LD} = \mathbf{T}_1(\Psi_{LD}) \mathbf{T}_2\left(\frac{\pi}{2} - \theta_{LD}\right) \mathbf{T}_1(-\Phi_{LD}) \quad (9.59)$$

where Ψ_{LD} , Θ_{LD} and Φ_{LD} denote the Euler angles yaw, pitch and roll, respectively. Please note, the roll angle Φ_{LD} is identical with the roll angle Φ_L defined in the local planetocentric (L) coordinate system.

9.3.4.3 Local horizontal (H)

Description

The local horizontal (H) coordinate system is defined as follows (see Fig. 9.5):

- the origin is the center of mass of the vehicle
- the fundamental plane is the local planetocentric horizontal plane
- the x_H -axis lies in the fundamental plane pointing east along the local parallel
- the y_H -axis lies in the fundamental plane pointing north along the local meridian
- the z_H -axis points upward in radial direction

This coordinate system has the same orientation as the local planetocentric coordinate system L (see Section 9.3.4.1) and the local vertical coordinate system V (see Section 9.3.4.4), but the axes are labeled differently.

Background

With λ and δ denoting the vehicle east longitude and declination, respectively, the transformation from planet-centered planet-fixed (PCPF) coordinates (refer to Section 9.3.2.4) to the local horizontal (H) coordinate system is defined by

$$\mathbf{T}_{PCPF}^H = \mathbf{T}_1\left(\frac{\pi}{2} - \delta\right) \mathbf{T}_3\left(\frac{\pi}{2} + \lambda\right) \quad (9.60)$$

The transformation from the inertial true of date (TOD) coordinate system (refer to Section 9.3.2.3) to the local horizontal (H) coordinate system is given by

$$\mathbf{T}_{TOD}^H = \mathbf{T}_{PCPF}^H \mathbf{T}_{TOD}^{PCPF} = \mathbf{T}_1\left(\frac{\pi}{2} - \delta\right) \mathbf{T}_3\left(\frac{\pi}{2} + \lambda + \Omega_E \Delta t\right) \quad (9.61)$$

9.3.4.4 Local vertical (V)

Description

The local vertical (V) coordinate system conforms to the normal Earth-axis system as defined in flight mechanics [2]. It is defined as follows (see Fig. 9.5):

- the origin is the center of mass of the vehicle
- the fundamental plane is the local planetocentric horizontal plane
- the x_V -axis lies in the fundamental plane pointing north along the local meridian
- the y_V -axis lies in the fundamental plane pointing east along the local parallel

- the z_V -axis points down to the center of the celestial body

This coordinate system has the same orientation as the local planetocentric coordinate system L (see Section 9.3.4.1) and the local horizontal coordinate system H (see Section 9.3.4.3), but the axes are labeled differently.

Background

The east longitude λ angle is measured from the prime meridian (i.e. Greenwich) to the vehicle's current meridian in east direction. The declination δ is the angle between the vehicle's radius vector and the equatorial plane. The transformation from the planet-centered planet-fixed (PCPF) coordinate system (refer to Section 9.3.2.4) to the local vertical (V) coordinate system is given by

$$\mathbf{T}_{PCPF}^V = \mathbf{T}_2\left(-\frac{\pi}{2} - \delta\right) \mathbf{T}_3(\lambda) \quad (9.62)$$

The transformation from inertial true of date (TOD) coordinates (refer to Section 9.3.2.3) to the local vertical (V) coordinate system is defined by

$$\mathbf{T}_{TOD}^V = \mathbf{T}_{PCPF}^V \mathbf{T}_{TOD}^{PCPF} = \mathbf{T}_2\left(-\frac{\pi}{2} - \delta\right) \mathbf{T}_3(\lambda + \Omega_E \Delta t) \quad (9.63)$$

The transformation from the trajectory (T) coordinate system (refer to Section 9.3.5.1) to the local vertical (V) system is given by

$$\mathbf{T}_T^V = \mathbf{T}_3(-\chi) \mathbf{T}_2(-\gamma) \quad (9.64)$$

with the flight path inclination γ and the flight path azimuth χ . The transformation from the air path (A) coordinate system (refer to Section 9.3.5.3) to the local vertical (V) coordinate system is given by

$$\mathbf{T}_A^V = \mathbf{T}_3(-\chi_a) \mathbf{T}_2(-\gamma_a) \mathbf{T}_1(-\mu_a) \quad (9.65)$$

with the air path azimuth χ_a , the air path inclination γ_a and the air path bank angle μ_a . The transformation from the body-fixed (B) coordinate system (refer to Section 9.3.3.1) to the local vertical (V) coordinate system is given by

$$\mathbf{T}_B^V = \mathbf{T}_3(-\Psi_L) \mathbf{T}_2(-\Theta_L) \mathbf{T}_1(-\Phi_L) \quad (9.66)$$

where Ψ_L , Θ_L and Φ_L denote the Euler angles yaw, pitch and roll, respectively. Note, the index states (L) as coordinate system since the Euler angles in the coordinate systems (V), (H) and (L) are identical.

9.3.4.5 Sun-pointing (S)

Description

The Sun-pointing (S) coordinate system is based on the direction of the Sun and the ecliptic and rotates with the vehicle position around the Sun (non-inertial). The coordinate system is used to analyse for potential sensor blinding. It can be used to derive illumination conditions of sensors and solar generators. The Sun-pointing (S) coordinate system is defined as follows:

- the origin is the center of mass of the vehicle
- the fundamental plane is the ecliptic
- the x_S -axis is collinear with the vector from the Sun center of mass to the vehicle center of mass (i.e. lies in the fundamental plane), pointing away from the Sun
- the y_S -axis is parallel to the fundamental plane and points towards the rotation of the Earth around the Sun
- the z_S -axis completes the right-handed coordinate system

Background

With σ and ϕ denoting the in-plane and out-of-plane transformation angles, respectively, the transformation from J2000 ecliptic (J2000Ec) coordinates (refer to Section 9.3.2.2) to the Sun-pointing (S) coordinate system is defined by

$$\mathbf{T}_{J2000Ec}^S = \mathbf{T}_2(-\phi) \mathbf{T}_3(\sigma) \quad (9.67)$$

9.3.5 Velocity-carried axis systems

The coordinate systems introduced in this section are standard aeronautical reference frames according to [2] and [3]:

- The **trajectory coordinate system (T)** is used to determine the course of the vehicle with respect to a planet-fixed observer, see Section 9.3.5.1.
- The **vertical air path coordinate system (AV)** is used to determine the direction of the aerodynamic forces of a vehicle, see Section 9.3.5.2.
- The **air path coordinate system (A)** is used to determine the direction of the aerodynamic forces of a vehicle, see Section 9.3.5.3.

9.3.5.1 Trajectory (T)

Description

The trajectory (T) coordinate system is currently only used for impulsive maneuvers (see Section 4.6) and is defined as follows (see Fig. 9.6):

- the origin is the center of mass of the vehicle
- the fundamental plane is spanned by the vehicle flight path velocity vector and the planetocentric position vector and is identical with the local vertical plane
- the x_T -axis lies in the fundamental plane and is collinear with the flight path velocity vector
- the y_T -axis lies in the local horizontal plane and completes the right-handed coordinate system

COORDINATE SYSTEMS AND TRANSFORMATIONS

- the z_T -axis lies in the fundamental plane pointing down

It is equivalent to the vertical air path coordinate system (AV) (see Section 9.3.5.2) but its x -axis is parallel to the flight path velocity vector and not to the air path velocity vector.

Background

The flight path inclination γ indicates ascent or descent with respect to the local horizontal plane. And the flight path azimuth χ is the heading angle in the horizontal plane measured clockwise from north. The transformation from the local vertical (V) coordinate system (refer to Section 9.3.4.4) to the trajectory (T) coordinate system is given by

$$\mathbf{T}_V^T = \mathbf{T}_2(\gamma) \mathbf{T}_3(\chi) \quad (9.68)$$

Similar, the transformation from the local planetocentric (L) system (refer to Section 9.3.4.1) to the trajectory (T) system is given by

$$\mathbf{T}_L^T = \mathbf{T}_2\left(-\frac{\pi}{2} + \gamma\right) \mathbf{T}_1(-\chi) \quad (9.69)$$

9.3.5.2 Vertical air path (AV)

Description

The vertical air path (AV) coordinate system is defined as follows (see Fig. 9.7):

- the origin is the center of the vehicle
- the fundamental plane is spanned by the vehicle air path velocity vector and the planetocentric position vector and is identical with the local vertical plane
- the x_T -axis lies in the fundamental plane and is collinear with the air path velocity vector
- the y_T -axis lies in the local horizontal plane and completes the right-handed coordinate system
- the z_T -axis lies in the fundamental plane pointing down

It is equivalent to the trajectory coordinate system (T) (see Section 9.3.5.1) but its x -axis is parallel to the air path velocity vector and not to the flight path velocity vector.

Background

The air path inclination γ_a indicates ascent or descent with respect to the local horizontal plane. And the air path azimuth χ_a is the heading angle in the horizontal plane measured clockwise from north. The transformation from the local vertical (V) system (refer to Section 9.3.4.4) to the vertical air path (AV) system is given by

$$\mathbf{T}_V^{AV} = \mathbf{T}_2(\gamma_a) \mathbf{T}_3(\chi_a) \quad (9.70)$$

The transformation from the local planetocentric (L) coordinate system (refer to Section 9.3.4.1) to the vertical air path (AV) coordinate system is given by

$$\mathbf{T}_L^{AV} = \mathbf{T}_2\left(-\frac{\pi}{2} + \gamma_a\right) \mathbf{T}_1(-\chi_a) \quad (9.71)$$

The transformation from the air path (A) coordinate system (refer to Section 9.3.5.3) to the vertical air path (AV) coordinate system is defined by

$$\mathbf{T}_A^{AV} = \mathbf{T}_1(-\mu_a) \quad (9.72)$$

with the air path bank angle μ_a . The air path bank angle results from a rotation about the air path velocity vector and is formed by the y_A -axis and the local horizontal plane (y_{AV} -axis).

9.3.5.3 Air path (A)

Description

The air path (A) coordinate system is also known as aerodynamics frame (see Fig. 9.8 and Fig. 9.9). It is defined as follows:

- the origin is the center of the vehicle
- the fundamental plane is spanned by the vehicle air path velocity vector and the z_A -axis
- the x_A -axis lies in the fundamental plane and is collinear with the air path velocity vector
- the y_A -axis completes the right-handed coordinate system
- the z_A -axis lies in the vehicle's x_B - z_B plane of symmetry pointing down

It is equivalent to the vertical air path system (AV) (see Section 9.3.5.2) but its z -axis lies in the vehicle's x_B - z_B plane of symmetry and not in the local vertical.

Background

The transformation from the body-fixed (B) coordinate system (refer to Section 9.3.3.1) to the air path (A) coordinate system is given by

$$\mathbf{T}_B^A = \mathbf{T}_3(\beta) \mathbf{T}_2(-\alpha) \quad (9.73)$$

where α and β denote the angle of attack and sideslip angle, respectively (see Fig. 9.9). They are defined by

$$\alpha = \arctan\left(\frac{v_{a,z}}{v_{a,x}}\right); \quad \beta = \arcsin(v_{a,y}) \quad (9.74)$$

where $v_{a,i}$ are the body-fixed components of the unit vector along the air path velocity vector obtained by

$$\hat{\mathbf{v}}_a = \frac{\hat{\mathbf{v}}_a}{|\hat{\mathbf{v}}_a|} = \begin{bmatrix} v_{a,x} \\ v_{a,y} \\ v_{a,z} \end{bmatrix}_B = \mathbf{T}_L^B \mathbf{T}_{AV}^L \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}_{AV} \quad (9.75)$$

The transformation from the vertical air path (AV) coordinates (refer to Section 9.3.5.2) to the air path (A) system is defined by

$$\mathbf{T}_{AV}^A = \mathbf{T}_1(\mu_a) \quad (9.76)$$

COORDINATE SYSTEMS AND TRANSFORMATIONS

with the air path bank angle μ_a . The air path bank angle results from a rotation about the air path velocity vector and is formed by the y_A -axis and the local horizontal plane (y_{AV} -axis). The transformation from local vertical (V) coordinates (refer to Section 9.3.4.4) to the air path (A) coordinate system is given by

$$\mathbf{T}_V^A = \mathbf{T}_1(\mu_a) \mathbf{T}_2(\gamma_a) \mathbf{T}_3(\chi_a) \quad (9.77)$$

with the air path inclination γ_a and the air path azimuth χ_a . The transformation from the local planetocentric (L) system (refer to Section 9.3.4.1) to the air path (A) system is given by

$$\mathbf{T}_L^A = \mathbf{T}_1(\mu_a) \mathbf{T}_2\left(-\frac{\pi}{2} + \gamma_a\right) \mathbf{T}_1(-\chi_a) \quad (9.78)$$

Fig. 9.9 shows the direction of the aerodynamic lift L and drag D forces as well as the side, or cross-stream, force Y_a along the air path axis system (A). Aerospace standards define the positive lift force acting in the negative z -axis direction, the drag force being positive along the negative x -axis, and the side force acting in the positive y -direction. The transformation from the total force (TF) coordinate system (refer to Section 9.3.3.3) to the air path (A) coordinate system is given by

$$\mathbf{T}_{TF}^A = \mathbf{T}_1(\mu_t) \quad (9.79)$$

with the aerodynamic bank angle μ_t . Its definition is provided in Section 9.3.3.3

The wind velocity component of the air path speed obeys the following expression:

$$V_a = V_k - V_w \quad (9.80)$$

with V_k the flight path velocity and V_w the wind velocity.

9.3.6 Other axis systems

The following coordinate systems do not belong to any of the previous coordinate system groups and are described in the following subsections:

- The **Euler angles coordinate system (EA)** is used to define the orientation of vehicle elements such as components, actuators and sensors, see Section 9.3.6.1.
- The **launcher coordinate system (FL)** is used to define velocity and attitude of a vehicle, see Section 9.3.6.2.
- The **r-bar/v-bar coordinate system** is used for relative motion like formation flying or rendezvous scenarios, see Section 9.3.6.3.
- The **navigation coordinate system (N)** is used to define the attitude of a vehicle, see Section 9.3.6.4.

9.3.6.1 Euler angles (EA)

Description

COORDINATE SYSTEMS AND TRANSFORMATIONS

The Euler angles (EA) coordinate system is used to define the orientation of vehicle elements such as actuators, components and sensors wrt their vehicle. The coordinate system is fixed to the structure of a vehicle element and is defined as follows:

- the origin is the geometrical center of the vehicle element
- the fundamental plane is the plane of symmetry of the vehicle element
- the x_{EA} -axis points to the front and spans the fundamental plane
- the y_{EA} -axis points to starboard and is perpendicular to the fundamental plane
- the z_{EA} -axis points down and spans the fundamental plane

This coordinate system has a similar orientation as the body-axes (B) coordinate system (see Section 9.3.3.1). In other words, the Euler angles coordinate system can be understood as body-fixed coordinate system of a vehicle element.

Background

The Euler angles (EA) coordinate system of a vehicle element is based on the body-fixed (B) coordinate system (see Section 9.3.3.1). The direction cosine transformation matrix from body-fixed (B) coordinates to Euler angles (EA) coordinates is given by

$$\mathbf{T}_B^{EA} = \mathbf{T}_1(\phi_{EA})\mathbf{T}_2(\theta_{EA})\mathbf{T}_3(\psi_{EA}) \quad (9.81)$$

with the Euler angles yaw ψ_{EA} , pitch θ_{EA} and roll ϕ_{EA} (often denoted as angle 1, angle 2 and angle 3, respectively) determining the vehicle element orientation w.r.t. the vehicle body axes.

9.3.6.2 Launcher (FL)

Description

The launcher (FL) coordinate system is used to define velocity and attitude of launch vehicles. Its definition is provided by the ROcket Simulator [83]. The coordinate system rotates with the surface of the celestial body (non-inertial) and is defined as follows:

- the origin is fixed to the relative initial state of the vehicle (relative position at time zero), e.g. the launchpad
- the fundamental plane is the local planetodetic horizontal plane
- the x_{FL} -axis lies in the fundamental plane pointing north along the local meridian
- the y_{FL} -axis lies in the fundamental plane pointing east along the local parallel
- the z_{FL} -axis is perpendicular to the fundamental plane pointing nadir

This coordinate system has a similar orientation as the local vertical (V) coordinate system (see Section 9.3.4.4), but (FL) is planetodetic oriented while (V) is planetocentric. Further, the (FL) system stays fixed at the initial state of the vehicle while (V) is moving with the vehicle.

Configuration

COORDINATE SYSTEMS AND TRANSFORMATIONS

The launcher frame is used only by the tracking analysis (see Section 5.16). In order to get the pitch and yaw angles of a launcher in this frame, the user needs to create a ground station located at the launch pad (same initial state as the launcher). Then a tracking analysis has to be created, where the launcher and the ground station are selected, and performed.

Background

The launcher frame is based on the planet-centered planet-fixed (PCPF) coordinate system (see Section 9.3.2.4). The direction cosine transformation matrix from planet-centered planet-fixed (PCPF) coordinates to launcher (FL) coordinates is given by

$$\mathbf{T}_{PCPF}^{FL} = \mathbf{T}_2\left(-\frac{\pi}{2} - \Phi_0\right) \mathbf{T}_3(\lambda_0) \quad (9.82)$$

with the initial longitude λ_0 measured from the prime meridian to the initial meridian of the vehicle and with the initial latitude Φ_0 being the angle between the equatorial plane and the normal to the surface of the ellipsoid at time zero (see Fig. 2.2).

The direction cosine matrix from the body-fixed (B) coordinate system (refer to Section 9.3.3.1) to the launcher (FL) coordinate system is defined by

$$\mathbf{T}_B^{FL} = \mathbf{T}_3(-\Psi_{FL}) \mathbf{T}_2(-\Theta_{FL}) \mathbf{T}_1(-\Phi_{FL}) \quad (9.83)$$

with the Euler angles yaw Ψ_{FL} , pitch Θ_{FL} and roll Φ_{FL} determining the vehicle orientation (the orientation of the vehicle body axes) w.r.t. the launcher frame.

9.3.6.3 R-bar/v-bar

Description

The R-Bar/V-Bar coordinate system is suitable for formation flying, rendezvous and docking. Its relative motion coordinates describe the position of an object/spacecraft (called follower, slave or chaser) with respect to another object/spacecraft (called leader, master, target) in Cartesian coordinates (Fig. 9.14). This coordinate system is defined as follows:

- the origin is the center of of the target vehicle
- the fundamental plane is the orbital plane of the target vehicle
- the x-axis (R-bar, radial) lies in the fundamental plane pointing in the same direction as the target vehicle's radius vector
- the y-axis (V-bar, in-track/along-track) lies in the fundamental plane and completes the right-handed coordinate system
- the z-axis (H-bar, cross-track) points in the direction of the angular momentum vector of the target vehicle's orbit

The R-Bar/V-Bar coordinate system is similar with the local orbit coordinate system (LO) (see Section 9.3.1.1) but it is centered at the target vehicle and not at the vehicle itself (the follower).

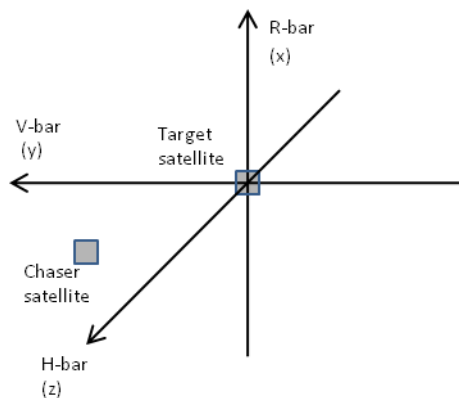


Fig. 9.14: Illustration of the R-bar/V-bar reference frame

Background

The V-bar axis (y) is non-linear and curved in space. Thus the transformation to other coordinate systems requires rotations and translations.

9.3.6.4 Navigation (N)

Description

The navigation (N) coordinate system is inertially fixed with a user-defined orientation. It is defined as follows:

- the origin is the center of mass of the vehicle
- the fundamental plane is local planetocentric horizontal plane
- one axis lies in the fundamental plane pointing in the direction of the specified azimuth (north if azimuth is zero)
- one axis points either to zenith or nadir and is perpendicular to the fundamental plane
- the third axis lies in the fundamental plane and completes the right-handed coordinate system

This frame can be used to define the inertial navigation frame of a launch vehicle for example. Each vehicle defined in the scenario has its own navigation frame.

Configuration

The configuration is optional and has to be specified in the vehicle builder of the *Vehicles & POIs Definition* branch of the *Modelling* tree (refer to The Vehicle Builder in User Manual). In case the frame definition is set to *default* the default values are used. The following options are available:

- *Axis definition* specifies the orientation of the axes with the following options:

COORDINATE SYSTEMS AND TRANSFORMATIONS

- *Version 1* has the +Z-axis in the fundamental plane pointing in the direction of the azimuth, while the +X-axis points in zenith direction
- *Version 2* has the +X-axis in the fundamental plane pointing in the direction of the azimuth, while the +Y-axis points in zenith direction
- *Reference* defines the date and time and the initial orientation of the zenith and azimuth direction w.r.t. inertial axes:
 - *Beginning of phase* causes that the date and time are at the beginning of the specified phase. The initial orientation of the zenith and azimuth direction w.r.t. inertial axes is defined by the vehicle's position at that time. By default, the first phase is used.
 - *Date* specifies the date and time at which the frame's initial orientation is defined. The zenith direction is indirectly set by the planetocentric *Longitude* and *Latitude*. The purpose of this setting is to define the inertial navigation frame based on a time outside the ASTOS simulation.
- *Azimuth* is the angle between north and the azimuth direction measured in the horizontal plane. It has to be specified since in some cases the azimuth direction of the velocity vector might be ill-defined (e.g. launcher on launch pad with a heading angle of 90 degree). The default value is zero (i.e. north direction).

Background

An inertial navigation frame is based on the planetocentric planet-fixed (PCPF) coordinate system (refer to Section 9.3.2.4): the corresponding direction cosine matrices for *Version 1* are

$$\mathbf{T}_{PCPF}^N = \mathbf{T}_1(\chi) \mathbf{T}_2(-\delta) \mathbf{T}_3(\lambda) \quad (9.84)$$

$$\mathbf{T}_N^{PCPF} = \mathbf{T}_3(-\lambda) \mathbf{T}_2(\delta) \mathbf{T}_1(-\chi) \quad (9.85)$$

and for *Version 2* are

$$\mathbf{T}_{PCPF}^N = \mathbf{T}_2\left(-\chi - \frac{\pi}{2}\right) \mathbf{T}_1(\delta) \mathbf{T}_3\left(\lambda - \frac{\pi}{2}\right) \quad (9.86)$$

$$\mathbf{T}_N^{PCPF} = \mathbf{T}_3\left(-\lambda + \frac{\pi}{2}\right) \mathbf{T}_1(-\delta) \mathbf{T}_2\left(\chi + \frac{\pi}{2}\right) \quad (9.87)$$

with the initial longitude λ , the initial planetodetic latitude δ and the specified azimuth angle χ . The relationship between the navigation frame and the body-fixed (B) coordinate system (refer to Section 9.3.3.1) of the vehicle is defined for *Version 1* by

$$\mathbf{T}_B^N = \mathbf{T}_2(-\theta_N) \mathbf{T}_3(-\psi_N) \mathbf{T}_1(-\phi_N) \quad (9.88)$$

$$\mathbf{T}_N^B = \mathbf{T}_1(\phi_N) \mathbf{T}_3(\psi_N) \mathbf{T}_2(\theta_N) \quad (9.89)$$

and for *Version 2* by

$$\mathbf{T}_B^N = \mathbf{T}_3(-\theta_N) \mathbf{T}_2(-\psi_N) \mathbf{T}_1(-\phi_N) \quad (9.90)$$

$$\mathbf{T}_N^B = \mathbf{T}_1(\phi_N) \mathbf{T}_2(\psi_N) \mathbf{T}_3(\theta_N) \quad (9.91)$$

The Euler angles yaw Ψ_N , pitch Θ_N and roll Φ_N determine the vehicle orientation (the orientation of the vehicle body axes) w.r.t. the navigation frame. They strongly depend on the selected axis definition. In case of *Version 1* the Euler angles *pitch* and *yaw* are given by

$$\begin{aligned}\theta_N &= \arctan(-z, x) \\ \psi_N &= \arctan\left(y, \sqrt{x^2 + z^2}\right)\end{aligned}\quad (9.92)$$

while in case of *Version 2* they are defined by

$$\begin{aligned}\theta_N &= \arctan(y, x) \\ \psi_N &= \arctan\left(-z, \sqrt{x^2 + y^2}\right)\end{aligned}\quad (9.93)$$

where x, y, z are the normalized components of the body x-axis (the main axis of the vehicle) represented in the N-frame. On the other hand, the *roll* angle always identifies the rotation angle about the main vehicle axis (x-axis).

9.4 Conversions and transformations

Coordinate transformations are required to transform the coordinates of a vector from one coordinate system into a target coordinate systems. One way of transforming is by a set of successive unit-axis rotations. For most coordinate systems the transformations are based on standard flight mechanics angles.

The unit-axis rotation sequence is mathematically expressed as a pre-multiply by a matrix. For example the transformation T defined by the angle sequence (a,b,c) about the axes $(1,2,3)$ is given by:

$$T = T_3(c) (T_2(b) (T_1(a))) \quad \begin{array}{l} \text{Sequence} = (1,2,3) \\ \text{Angles} = (a, b, c) \end{array} \quad (9.94)$$

This chapters describes

- **Orthogonal transformations**, see Section 9.4.1
- **Conversion of velocity vectors**, see Section 9.4.2
- **Pre-defined coordinate transformations**, see Section 9.4.3

9.4.1 Orthogonal transformation

A vector $\hat{x}$ in the Euclidean 3-dimensional space may be represented by a linear combination of Cartesian unit vectors $\mathbf{i}, \mathbf{j}, \mathbf{k}$. The scalar components x_i along each unit vector depend on the chosen reference frame, such that

$$\hat{x} = x_{A1}\mathbf{i}_A + x_{A2}\mathbf{j}_A + x_{A3}\mathbf{k}_A = x_{B1}\mathbf{i}_B + x_{B2}\mathbf{j}_B + x_{B3}\mathbf{k}_B \quad (9.95)$$

COORDINATE SYSTEMS AND TRANSFORMATIONS

To map the scalar components x_{Ai} into components x_{Bi} the vector components are listed in a column vector, and a linear transformation is performed [38], [39]

$$\begin{bmatrix} x_{B1} \\ x_{B2} \\ x_{B3} \end{bmatrix}_B = \mathbf{T}_A^B \begin{bmatrix} x_{A1} \\ x_{A2} \\ x_{A3} \end{bmatrix}_A; \quad \text{where } \mathbf{T}_A^B = \begin{bmatrix} \mathbf{i}_A \mathbf{i}_B & \mathbf{j}_A \mathbf{i}_B & \mathbf{k}_A \mathbf{i}_B \\ \mathbf{i}_A \mathbf{j}_B & \mathbf{j}_A \mathbf{j}_B & \mathbf{k}_A \mathbf{j}_B \\ \mathbf{i}_A \mathbf{k}_B & \mathbf{j}_A \mathbf{k}_B & \mathbf{k}_A \mathbf{k}_B \end{bmatrix} \quad (9.96)$$

The transformation matrix $\mathbf{T}_A^B$ contains the direction cosines of the two sets of unit vectors. In this notation, the subscript A denotes the original system and the superscript B denotes the system into which the vector components are mapped. Since the direction cosine matrix is orthonormal, the inverse is identical to its transpose [39]:

$$\mathbf{T}_A^B = (\mathbf{T}_A^B)^{-1} = (\mathbf{T}_A^B)^T \quad (9.97)$$

A direction cosine matrix relating two axis systems can be expressed as the successive unit-axis rotations which transform one axis system into the other. The direction cosine matrices of a rotation about each of the three unit axes are given as follows

$$\mathbf{T}_1(i) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos i & \sin i \\ 0 & -\sin i & \cos i \end{bmatrix}; \quad \mathbf{T}_2(i) = \begin{bmatrix} \cos i & 0 & -\sin i \\ 0 & 1 & 0 \\ \sin i & 0 & \cos i \end{bmatrix}; \quad \mathbf{T}_3(i) = \begin{bmatrix} \cos i & \sin i & 0 \\ -\sin i & \cos i & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (9.98)$$

In this notation, the subscript 1, 2, or 3 indicates the axis of rotation, and the argument in parentheses i designates the angle of rotation, which is positive for a right-handed rotation.

Unit-axis transformation matrices have the several formal properties which are used in the explanations further below. These properties are explained here. Successive unit-axis rotations about the same axis are identical to a single rotation by the sum of the angles

$$\mathbf{T}_k(i_1) \mathbf{T}_k(i_2) = \mathbf{T}_k(i_1 + i_2) \quad k = 1, 2, 3 \quad (9.99)$$

The inverse unit-axis transformation is equal to the inverse rotation about the same axis

$$(\mathbf{T}_k(i))^{-1} = (\mathbf{T}_k(i))^T = \mathbf{T}_k(-i) \quad k = 1, 2, 3 \quad (9.100)$$

The inverse of a product of unit-axis transformations is equal to the product of the inverse rotations performed in the reversed order

$$(\mathbf{T}_k(i_1) \mathbf{T}_l(i_2) \mathbf{T}_m(i_3))^{-1} = \mathbf{T}_m(-i_3) \mathbf{T}_l(-i_2) \mathbf{T}_k(-i_1) \quad k, l, m = 1, 2, 3 \quad (9.101)$$

9.4.2 Conversion of velocities

The velocity vector of a vehicle is often available in one coordinate system but calculations require its representation in a different coordinate system. Therefore there is a need of transforming the velocity vector from one coordinate system into another. In Table 9.10 the

COORDINATE SYSTEMS AND TRANSFORMATIONS

terms used further below in this section are introduced. An overview of the typical velocity vector conversions is shown in Fig. 9.15.

Table 9.10: Aeronautical standard terms and symbols for relative velocities

Velocity of	Relative to	Term	Symbol
Vehicle	Planet	Flight path velocity	V_k
Vehicle	Atmosphere	Air path velocity	V_a
Atmosphere	Planet	Wind velocity	V_w

The velocity vector transformation involves the position vector. A position vector defined in the inertial geocentric reference frame TOD can be expressed using the radius R , the inertial longitude Λ and the declination δ . Its Cartesian representation is given by

$$\hat{P} = R \begin{bmatrix} \cos\delta\cos\Lambda \\ \cos\delta\sin\Lambda \\ \sin\delta \end{bmatrix}_{TOD} \quad (9.102)$$

With respect to the Earth-fixed reference frame E , the Cartesian representation is given by

$$\hat{P} = R \begin{bmatrix} \cos\delta\cos\lambda \\ \cos\delta\sin\lambda \\ \sin\delta \end{bmatrix}_E \quad (9.103)$$

where $\lambda = \Lambda - \Omega_E \Delta t$ is the Earth-fixed longitude, Ω_E is angular velocity of the Earth and Δt is the elapsed time.

The flight path velocity vector $\hat{V}_k$ is the vehicle velocity vector with respect to the planet-fixed axis system E . Given the vehicle inertial velocity $\hat{V}_i$, the flight path velocity vector $\hat{V}_k$ is computed using the following expression

$$\hat{V}_k = \hat{V}_i - \hat{\Omega}_E \times \hat{R} \quad (9.104)$$

where $\hat{\Omega}_E$ is the angular velocity vector of the planet and $\hat{R}$ is the vehicle radius vector in the system E . Given the wind velocity vector $\hat{V}_w$ relative to the planet, the air path velocity vector $\hat{V}_a$ can be expressed as follows

$$\hat{V}_a = \hat{V}_k - \hat{V}_w = \hat{V}_i - \hat{\Omega}_E \times \hat{R} - \hat{V}_w \quad (9.105)$$

If the flight path velocity vector is expressed using the flight path speed V_k , flight path inclination γ and heading χ , its Cartesian representation along the local frame L is given by

$$\hat{V}_k = V_k \begin{bmatrix} \sin\gamma \\ \cos\gamma\sin\chi \\ \cos\gamma\cos\chi \end{bmatrix}_L \quad (9.106)$$

Similarly, the air path velocity vector is given by

$$\hat{V}_a = V_a \begin{bmatrix} \sin\gamma_a \\ \cos\gamma_a\sin\chi_a \\ \cos\gamma_a\cos\chi_a \end{bmatrix}_L \quad (9.107)$$

where V_k is the air path speed, and γ_a and χ_a are the air path inclination heading, respectively.

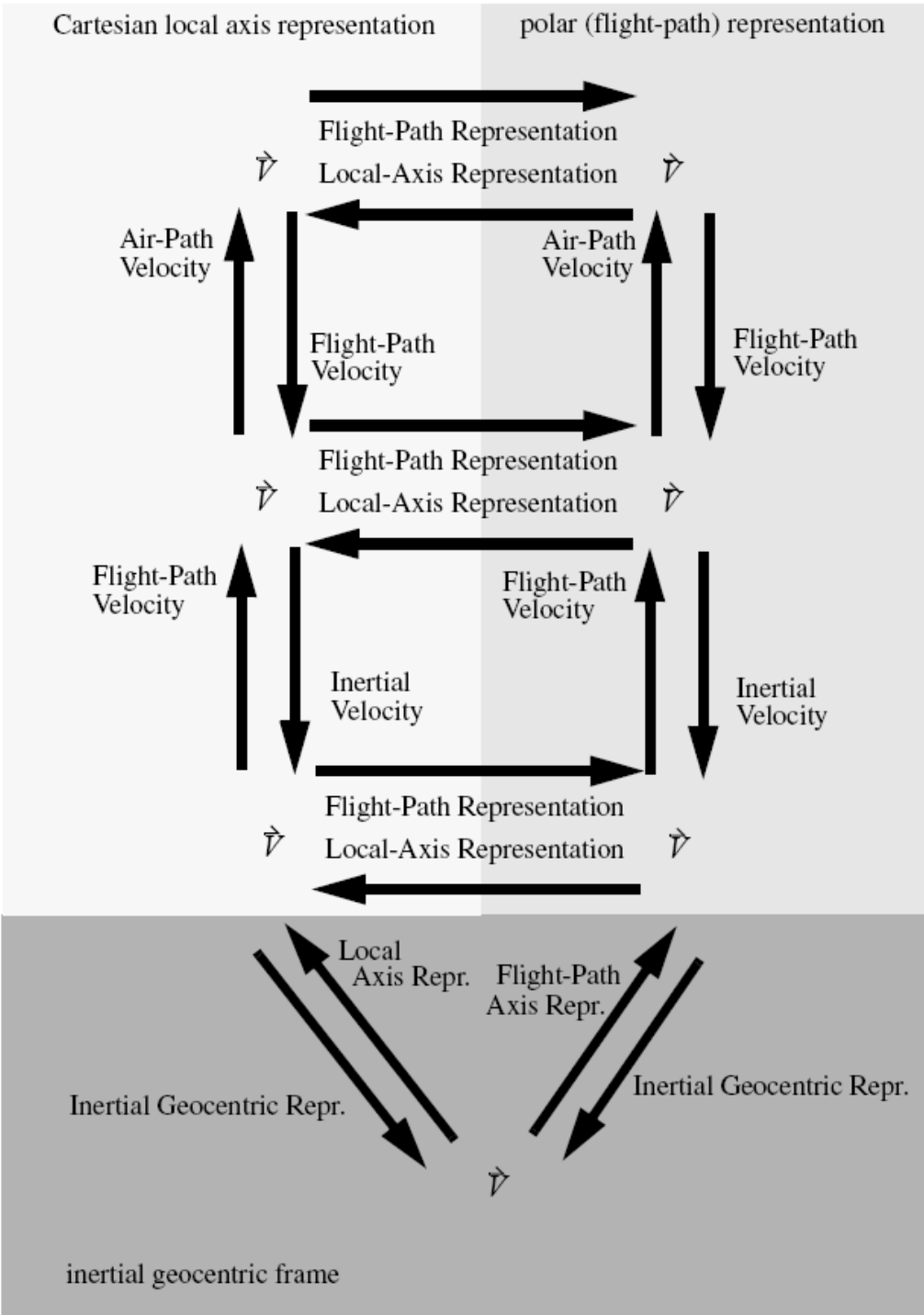


Fig. 9.15: Relationships between the inertial and Earth-fixed velocity vectors

9.4.3 Coordinate transformations

The following tables contain the direction cosine transformation matrices relating the coordinate systems given in Section 9.3 expressed in terms of unit-axis rotation matrices according to the nomenclature introduced in Section 9.4.

Table 9.11: Transformations between orbital systems and planet-centered systems

		To		
		P	TOD	E
From	LO	$T_3(-v)$		
	P	I	$T_3(-\Omega) T_1(-i) T_3(-\omega)$	
	TOD		I	$T_3(\Omega_E \Delta t)$

Table 9.12: Transformations between planet-centered systems and vehicle-carried systems

		To		
		L	H	V
From	E	$T_2(-\delta) T_3(\lambda)$	$T_1(\pi/2-\delta) T_3(\pi/2+\lambda)$	$T_2(-\pi/2-\delta) T_3(\lambda)$
	L	I	$T_1(\pi/2) T_3(\pi/2)$	$T_2(-\pi/2)$
	H		I	$T_3(\pi/2) T_2(\pi/2)$

Table 9.13: Transformations between velocity-carried and vehicle-carried systems

		To			
		L	V	AV	A
From	L	I	$T_2(-\pi/2)$	$T_2(\gamma_a-\pi/2) T_1(-\chi_a)$	$T_1(\mu_a) T_2(\gamma_a-\pi/2) T_1(-\chi_a)$
	T	$T_1(\chi) T_2(\pi/2-\gamma)$	$T_3(-\chi) T_2(-\gamma)$		
	AV	$T_1(\chi_a) T_2(\pi/2-\gamma_a)$	$T_3(-\chi_a) T_2(-\gamma_a)$	I	$T_1(\mu_a)$
	A	$T_1(\chi_a) T_2(\pi/2-\gamma_a) T_1(-\mu_a)$	$T_3(-\chi_a) T_2(-\gamma_a) T_1(-\mu_a)$	$T_1(-\mu_a)$	I

Table 9.14: Transformations between body-fixed system and vehicle-carried systems

		To			
		L	N 1 or 2	V	B
From	L	I			$T_1(\Phi) T_2(-\pi/2+\Theta) T_1(-\Psi)$
	N 1 or 2		I		$T_1(\Phi) T_3(\Psi) T_2(\Theta)$ or $T_1(\Phi) T_2(\Psi) T_3(\Theta)$
	V			I	$T_1(\Phi) T_2(\Theta) T_3(\Psi)$
	B	$T_1(\Psi) T_2(\pi/2-\Theta) T_1(-\Phi)$	$T_2(-\Theta) T_3(-\Psi) T_1(-\Phi)$ or $T_3(-\Theta) T_2(-\Psi) T_1(-\Phi)$	$T_3(-\Psi) T_2(-\Theta) T_1(-\Phi)$	I

Table 9.15: Transformations between aerodynamic systems and body-fixed systems

COORDINATE SYSTEMS AND TRANSFORMATIONS

		To		
		A	B	TF
From	A	I	$T_2(\alpha) T_3(-\beta)$	$T_1(\mu_\theta)$
	B	$T_3(\beta) T_2(-\alpha)$	I	$T_2(-\alpha_\theta) T_1(\Phi')$
	TF	$T_1(\mu_\theta)$	$T_1(\Phi') T_2(\alpha_\theta)$	I
	TA	$T_1(\mu_\theta) T_2(-\alpha_\theta)$	$T_1(\Phi')$	$T_2(-\alpha_\theta)$

10 Physical Properties and Constants

This chapter provides an overview of physical constants and parameters used in *ASTOS*. All listed values are either internally used by the software for calculations, represent default values for various model parameters or can serve the user as a lookup wizard for the scenario and model definition.

Table 10.1: Physical constants

Name	Value	Unit
Avogadro constant	$6.022140857 \cdot 10^{23}$	1/mol
Boltzmann constant	$1.380662 \cdot 10^{-23}$	J/K
Elementary charge	$1.602176565 \cdot 10^{-19}$	C
Gas constant	8.31432	J/mol/K
Gravitational constant	$6.67428 \cdot 10^{-11}$	$\text{m}^3/\text{kg}/\text{s}^2$
Speed of light	299792458.0	m/s
Stefan-Boltzmann constant	$5.670367 \cdot 10^{-8}$	$\text{W}/\text{m}^2/\text{K}^4$
Vacuum permeability	$4\pi \cdot 10^{-7}$	N/A^2

Table 10.2: Physical properties

Property	Value	Unit	Remarks
Molar mass of air	28.9644	kg/kmol	At sea level on Earth
Standard gravity	9.80665	m/s^2	Standard acceleration due to gravity at Earth's surface
Zero Temperature	273.16	K	Triple point of water

11 User-Defined Models

Description

In case the implemented models in the *ASTOS* software do not fully satisfy the mission requirements, the user is given the possibility to program a custom version for one of the following models:

- Actuator
- Aerodynamics
- Atmosphere
- Component

The user has to compile the programming code of the user-defined model to a dynamic library, which is then accessed by *ASTOS*. Any programming language and any compiler which support the dynamic library architecture can be used. In this chapter the procedure is shown using the programming languages C, Fortran, and Ada. The templates and test scenarios delivered with this distribution of *ASTOS* provide source code and makefiles for the compilers Microsoft Visual C++, Microsoft Visual Fortran, and Gnat Ada, respectively. For a different compiler the provided makefiles may need to be adapted. Please refer to the user manual of your compiler to get the proper compiler options.

Note: This software uses C-calling conventions. It is important to know, especially for Fortran libraries.

All provided interface routines can be grouped into two categories:

- Routines that are **common** to most of the user-defined models, they are detailed in Section 11.1
- **Model-specific** routines depending on the kind of the user-defined model:
 1. Actuator, see Section 11.2
 2. Aerodynamics, see Section 11.3
 3. Atmosphere, see Section 11.4
 4. Component, see Section 11.5

Background

A dynamic library can be used to link a user-coded library to an existing program. Therefore a well defined interface must be provided. The *ASTOS* user-defined model interface is described

USER-DEFINED MODELS

in the following sections. On *Microsoft Windows* platforms these libraries are known as DLLs (suffix `.dll`) while on *Linux* systems they are called Shared Objects (suffix `.so`).

For the reasons of language compatibility it is necessary to define general parameter lists for each interface routine, which are valid for any language. Therefore any routine which includes a string or a character variable includes also a variable with the length of this string or character array.

Note: Strings with the length of 80 characters are used in the interface. In case of C and Ada all variables defining the length of a string do not need to be mentioned. They are only required for Fortran where they are hidden parameters.

Note: Every input and output value part of the interface is assumed to be in the corresponding SI-unit. It is the full responsibility of the user to convert the units, if required. For more information, see Available Units in User Manual.

Auxiliary variables

Several routines of the user interface provide an array of auxiliary variables. The content of this array depends on the kind of the user-defined model as shown in table Table 11.1 .

Note: When using the C interface, the array index value is between zero and the number of items in the array minus one. In Ada and Fortran the array index value is between one and the number of items in the array.

Table 11.1: Index values of auxiliary variables

Variable Name	Actuator	Aerodynamics	Atmosphere	Component
Burn time (s)	1			
Flight time (s)	2	1		1
Altitude (m)	3	2		2
Atmospheric pressure (Pa)	4	3		3
Atmospheric density (kg/m ³)	5	4		4
Mach number (-)	6	5		5
Angle of attack (rad)	7	6		6
Side slip angle (rad)	11	7		
Dynamic pressure (Pa)	8	8		7
Dynamic viscosity (Pas)		9		
Vehicle mass (kg)	9	10		8
Illumination (-)	10			
Position x bottom node w.r.t. global node (m)	12			
Position y bottom node w.r.t. global node (m)	13			

USER-DEFINED MODELS

Variable Name	Actuator	Aerodynamics	Atmosphere	Component
Position z bottom node w.r.t. global node (m)	14			
Quaternion w w.r.t. reference frame (-)	15			
Quaternion x w.r.t. reference frame (-)	16			
Quaternion y w.r.t. reference frame (-)	17			
Quaternion z w.r.t. reference frame (-)	18			
Yaw angle L-frame of reference object (rad)	19			
Pitch angle L-frame of reference object (rad)	19			
Roll angle L-frame of reference object (rad)	19			

The illumination variable represents the fraction of the visible sun disk, i.e. in case of full eclipse the value is 0.0. Vehicle attitude and local shadowing by vehicle components are not considered. The position vector of the bottom w.r.t. global node is represented in the reference frame of the reference object. The reference object is typically the vehicle (or ground facility) the actuator belongs to. The reference frame is the body fixed (B) frame of the vehicle the actuator belongs to.

11.1 Common Model Routines

This chapter describes all routines and functions common to more than one of the user-defined models. They are grouped into the following categories:

- **Initialization** of the user-defined model, see Section 11.1.1
- **Finalization** of the user-defined model, see Section 11.1.2
- **Auxiliary States** provides access to time-dependent quantities, see Section 11.1.3
- **Control Variables** provides access to optimizable time-dependent quantities such as parameters and controls, see Section 11.1.4
- **Constraints** of the user-defined model, see Section 11.1.5
- **Cost functions** for the cost functions of the user-defined model, see Section 11.1.6
- **Algebraic functions** for additional time-dependent output functions, see Section 11.1.7
- **Info** for additional information about the user-defined model, see Section 11.1.8

11.1.1 Model Initialization Routine

Description

It only consists of the Initialize routine and is always called first. This routine can be used to read data from external files, for example. Further the user can return a pointer to a work space. This is useful if the user model library is used by several model instances (e.g. actuator model is part of first and second stage) but certain data should not be shared between them (e.g. throttle).

Note: In case of Ada 95 the initialization routine has to initialize the user-defined model library using the function `Adalnit`.

Configuration

All interface parameters are:

- `User_String` holds the string defined in the *User parameters* field of the corresponding user model in the *ASTOS GUI*. This string can be used in any way the user likes and has a maximum length of 1024 characters.
- `Identifier` is the identifier of the user model instance (the one used in the *Dynamics Configuration*)
- `Vehicle` is the identifier of the vehicle the user model belongs to.

Note: For user atmosphere models the provided vehicle identifier is an empty string!

- `Instance` is an access to memory to be allocated by the user model within this interface. The access is provided in any other interface routine to clearly identify the current instance of the user model. For example, a basic implementation would provide access to an integer value or to a string.

Ada interface

```

procedure Initialize
  (User_String_Adr : in System.Address;
   User_String_Len : in Integer;
   Identifier_Adr  : in System.Address;
   Identifier_Len  : in Integer;
   Vehicle_Adr    : in System.Address;
   Vehicle_Len    : in Integer;
   Instance_Adr   : in System.Address)
is
  User_String : String (1 .. User_String_Len);
  for User_String'Address use User_String_Adr;
  Identifier : String (1 .. Identifier_Len);
  for Identifier'Address use Identifier_Adr;
  Vehicle : String (1 .. Vehicle_Len);
  for Vehicle'Address use Vehicle_Adr;

```


USER-DEFINED MODELS

```
Instance : System.Address;  
for Instance'Address use Instance_Adr;  
pragma Export (C, Initialize, "init");
```

C interface

```
__declspec(dllexport) void __cdecl init(  
char *user_string,  
int user_string_len,  
char *identifier,  
int identifier_len,  
char *vehicle,  
int vehicle_len,  
char *instance)
```

Fortran interface

```
subroutine init (userstr, ident, vehicle, instance)  
  cDEC$ ATTRIBUTES DLLEXPORT :: init  
  character (*) userstr  
  character (*) ident  
  character (*) vehicle  
  character (*) instance
```

11.1.2 Model Finalization Routine

Description

The optional Finalize routine is always called last and is intended to be used for memory cleanup and closure of external files. It is available in all user models.

Configuration

It only has one parameter: the Instance pointer, an access to the memory allocated during initialization.

Ada interface

```
procedure Finalize  
(Instance_Adr : in System.Address)  
is  
  Instance : System.Address;
```

USER-DEFINED MODELS

```
for Instance'Address use Instance_Adr;  
pragma Export (C, Finalize, "finalize");
```

C interface

```
__declspec(dllexport) void __cdecl finalize(  
char *instance)
```

Fortran interface

```
subroutine finalize (instance)  
  cDEC$ ATTRIBUTES DLLEXPORT :: finalize  
  character *(*) instance
```

11.1.3 Model Auxiliary States Routines

Description

These routines are optional and required to model additional time-dependent quantities (states) describing the dynamics of the optimal control problem. For each state their derivatives (equations of motion) have to be provided. Auxiliary states are available for the following user models:

- Actuators (see Section 11.2)
- Components (see Section 11.5)

Configuration

Five routines are part of the auxiliary states:

1. **nostat** returns the number `Result` of auxiliary states. If no auxiliary states are defined, the returned value must be zero. `Instance` is the only input parameter of this routine.
2. **statinfo** provides additional information about the auxiliary states and is called once for each auxiliary state specified by the input parameter `Index`. Another input parameter is `Instance`. Output parameters are the strings `Name`, `Description` and `Unit` with a maximum length of 80 characters. `Unit` must be a valid *ASTOS* SI-unit string. Please refer to *Available Units in the User Manual*.
3. **statini** returns the initial values (`Values`) and initial bounds (`Lower_Bounds`, `Upper_Bounds`) of all defined auxiliary states as arrays. `Instance` is the only input parameter of this routine.

Tip: Specify the same value for `Values`, `Lower_Bounds` and `Upper_Bounds` if the initial value of an auxiliary state shall not be optimizable.

Note: The bounds provided through this interface are only used for the initial state (time zero of the first phase the auxiliary state is present).

4. **statbnds** returns the `Lower_Bound` and `Upper_Bound` values of an auxiliary state specified by the input parameter `Index`. Other input parameters are the instance pointer `Instance` and the value `Aux_State` of the auxiliary state specified by `Index`.
5. **statderi** provides the derivatives `Result` of all defined auxiliary states. Input parameters are the `Instance`, the auxiliary variables `Vars` (see Chapter 11), see auxiliary states `Aux_States` and the control variables `Control_Vars`.

Ada interface

```

procedure Number_Auxiliary_States
  (Instance_Adr : in System.Address;
   Result_Adr   : in System.Address)
is
  Instance : System.Address;
  for Instance'Address use Instance_Adr;
  Result : Natural;
  for Result'Address use Result_Adr;
pragma Export (C, Number_Auxiliary_States, "nostat");

procedure Auxiliary_States_Info
  (Instance_Adr : in System.Address;
   Index_Adr    : in System.Address;
   Name_Adr     : in System.Address;
   Desc_Adr     : in System.Address;
   Unit_Adr     : in System.Address;
   Name_Len     : in Integer;
   Desc_Len     : in Integer;
   Unit_Len     : in Integer)
is
  Instance : System.Address;
  for Instance'Address use Instance_Adr;
  Index : Natural;
  for Index'Address use Index_Adr;
  Name : Item_String;
  for Name'Address use Name_Adr;
  Desc : Item_String;
  for Desc'Address use Desc_Adr;
  Unit : Item_String;
  for Unit'Address use Unit_Adr;

```

USER-DEFINED MODELS

```
pragma Export (C, Auxiliary_States_Info, "statinfo");

procedure Auxiliary_States_Initial
  (Instance_Adr      : in System.Address;
   Values_Adr       : in System.Address;
   Lower_Bounds_Adr : in System.Address;
   Upper_Bounds_Adr : in System.Address)
is
  Instance : System.Address;
  for Instance'Address use Instance_Adr;
  Value : array (1 .. No_Aux_States) of Long_Float;
  for Values'Address use Values_Adr;
  Lower_Bounds : array (1 .. No_Aux_States) of Long_Float;
  for Lower_Bounds'Address use Lower_Bounds_Adr;
  Upper_Bounds : array (1 .. No_Aux_States) of Long_Float;
  for Upper_Bounds'Address use Upper_Bounds_Adr;
pragma Export (C, Auxiliary_States_Initial, "statini");

procedure Auxiliary_States_Bounds
  (Instance_Adr      : in System.Address;
   Index_Adr         : in System.Address;
   Aux_State_Adr     : in System.Address;
   Lower_Bound_Adr   : in System.Address;
   Upper_Bound_Adr   : in System.Address)
is
  Instance : System.Address;
  for Instance'Address use Instance_Adr;
  Index : Natural;
  for Index'Address use Index_Adr;
  Aux_State : Long_Float;
  for Aux_State'Address use Aux_State_Adr;
  Lower_Bound : Long_Float;
  for Lower_Bound'Address use Lower_Bound_Adr;
  Upper_Bound : Long_Float;
  for Upper_Bound'Address use Upper_Bound_Adr;
pragma Export (C, Auxiliary_States_Bounds, "statbnds");

procedure Auxiliary_States_Derivatives
  (Instance_Adr      : in System.Address;
   No_Vars_Adr       : in System.Address;
   Vars_Adr          : in System.Address;
   Aux_States_Adr    : in System.Address;
   Control_Vars_Adr  : in System.Address;
   Result_Adr        : in System.Address)
is
  Instance : System.Address;
  for Instance'Address use Instance_Adr;
```

USER-DEFINED MODELS

```
No_Vars : Natural;
for No_Vars'Address use No_Vars_Adr;
Vars : array (1 .. No_Vars) of Long_Float;
for Vars'Address use Vars_Adr;
Aux_States : array (1 .. No_Aux_States) of Long_Float;
for Aux_States'Address use Aux_States_Adr;
Control_Vars : array (1 .. No_Ctrl_Vars) of Long_Float;
for Control_Vars'Address use Control_Vars_Adr;
Result : array (1 .. No_Aux_States) of Long_Float;
for Result'Address use Result_Adr;
pragma Export (C, Auxiliary_States_Derivatives, "statderi");
```

C interface

```
__declspec(dllexport) void __cdecl nostat(
    char *instance,
    int *result)

__declspec(dllexport) void __cdecl statinfo(
    char *instance,
    const int *index,
    char *name,
    char *desc,
    char *unit,
    int nameLen,
    int descLen,
    int unitLen)

__declspec(dllexport) void __cdecl statini(
    char *instance,
    double *values,
    double *lowerBounds,
    double *upperBounds)

__declspec(dllexport) void __cdecl statbnds(
    char *instance,
    const int *index,
    double *auxState,
    double *lowerBound,
    double *upperBound)

__declspec(dllexport) void __cdecl statderi(
    char *instance,
    const int *noVars,
    double *vars,
    double *auxStates,
```

USER-DEFINED MODELS

```
double *ctrlVars,  
double *result)
```

Fortran interface

```
subroutine nostat (instance, result)  
  cDEC$ ATTRIBUTES DLLEXPORT :: nostat  
  character *(*) instance  
  integer result  
  
subroutine statinfo (instance, index, name, desc, unit)  
  cDEC$ ATTRIBUTES DLLEXPORT :: statinfo  
  character *(*) instance  
  integer index  
  character*(80) name, desc, unit  
  
subroutine statini (instance, values, lowbnds, uppbnds)  
  cDEC$ ATTRIBUTES DLLEXPORT :: statini  
  character *(*) instance  
  double precision values(*), lowbnds(*), uppbnds(*)  
  
subroutine statbnds (instance, index, auxstat, lowbnd, uppbnd)  
  cDEC$ ATTRIBUTES DLLEXPORT :: statbnds  
  character *(*) instance  
  integer index  
  double precision auxstat  
  double precision lowbnd, uppbnd  
  
subroutine statderi (instance, novars, vars, auxstats, ctrlvars, result)  
  cDEC$ ATTRIBUTES DLLEXPORT :: statderi  
  character *(*) instance  
  integer novars  
  double precision vars(novars), auxstats(*), ctrlvars(*)  
  double precision result(*)
```

11.1.4 Model Control Variables Routines

Description

Control variables are time-dependent (**Controls**) and time-independent (**Parameters**) optimizable quantities. The routines are optional and available for the following user models:

- Actuators (see Section 11.2)

USER-DEFINED MODELS

- Aerodynamics (see Section 11.3)
- Components (see Section 11.5)

Configuration

The following routines are present:

1. **nocvars** returns the number `Result` of control variables. If no control variables are defined, the returned value must be zero. `Instance` is the only input parameter of this routine.
2. **varsinfo** provides additional information about the control variables and is called once for each control variable specified by the input parameter `Index`. Another input parameter is `Instance`. Output parameters are the strings `Name`, `Description` and `Unit` with a maximum length of 80 characters. `Unit` must be a valid *ASTOS* SI-unit string. Please refer to Available Units in the User Manual. The output parameter `Is_Connected` specifies whether the control variable is connected to the previous phase (1) or not (0). Currently it can be connected only to itself. And the output parameter `Time_Variable` specifies whether the control variables is a control (1) or a parameter (0).
3. **varsbnds** returns the `Lower_Bound` and `Upper_Bound` values of a control variable specified by the input parameter `Index`. Another input parameter is `Instance`.
4. **varsesti** returns the initial guess value `Result` of a control variable specified by the input parameter `Index`. Another input parameter is `Instance`. A set of auxiliary variables as described in Chapter 11 may be used to calculate the time-dependent initial guess value, where the integer-valued input parameter `No_Vars` (novars) contains the number of variables and `Vars` contains the array of variables itself.

Ada interface

```
procedure Number_Control_Variables
  (Instance_Adr : in System.Address;
   Result_Adr   : in System.Address)
is
  Instance : System.Address;
  for Instance'Address use Instance_Adr;
  Result : Natural;
  for Result'Address use Result_Adr;
pragma Export (C, Number_Control_Variables, "nocvars");

procedure Control_Variables_Info
  (Instance_Adr      : in System.Address;
   Index_Adr         : in System.Address;
   Name_Adr          : in System.Address;
   Desc_Adr          : in System.Address;
   Unit_Adr          : in System.Address;
   Is_Connected_Adr  : in System.Address;
   Time_Variable_Adr : in System.Address;
```

USER-DEFINED MODELS

```
Name_Len      : in Integer;
Desc_Len      : in Integer;
Unit_Len      : in Integer)
is
  Instance : System.Address;
  for Instance'Address use Instance_Adr;
  Index : Integer;
  for Index'Address use Index_Adr;
  Name : String (1 .. Name_Len);
  for Name'Address use Name_Adr;
  Desc : String (1 .. Desc_Len);
  for Desc'Address use Desc_Adr;
  Unit : String (1 .. Unit_Len);
  for Unit'Address use Unit_Adr;
  Is_Connected : Natural;
  for Is_Connected'Address use Is_Connected_Adr;
  Time_Variable : Natural;
  for Time_Variable'Address use Time_Variable_Adr;
pragma Export (C, Control_Variables_Info, "varsinfo");

procedure Control_Variables_Estimate
  (Instance_Adr : in System.Address;
   No_Vars_Adr : in System.Address;
   Vars_Adr : in System.Address;
   Aux_States_Adr : in System.Address;
   Index_Adr : in System.Address;
   Result_Adr : in System.Address)
is
  Instance : System.Address;
  for Instance'Address use Instance_Adr;
  No_Vars : Integer;
  for No_Vars'Address use No_Vars_Adr;
  Vars : array (1 .. No_Vars) of Long_Float;
  for Vars'Address use Vars_Adr;
  Aux_States : array (1 .. No_Aux_States) of Long_Float;
  for Aux_States'Address use Aux_States_Adr;
  Index : Natural;
  for Index'Address use Index_Adr;
  Result : Long_Float;
  for Result'Address use Result_Adr;
pragma Export (C, Control_Variables_Estimate, "varsesti");

procedure Control_Variables_Bounds
  (Instance_Adr : in System.Address;
   Index_Adr : in System.Address;
   Lower_Bound_Adr : in System.Address;
   Upper_Bound_Adr : in System.Address)
```


USER-DEFINED MODELS

```
is
    Instance : System.Address;
    for Instance'Address use Instance_Adr;
    Index : Natural;
    for Index'Address use Index_Adr;
    Lower_Bound : Long_Float;
    for Lower_Bound'Address use Lower_Bound_Adr;
    Upper_Bound : Long_Float;
    for Upper_Bound'Address use Upper_Bound_Adr;
pragma Export (C, Control_Variables_Bounds, "varsbnds");
```

C interface

```
__declspec(dllexport) void __cdecl nocvars(
    char *instance,
    int *result)

__declspec(dllexport) void __cdecl varsinfo(
    char *instance,
    const int *index,
    char *name,
    char *desc,
    char *unit,
    int *isConnected,
    int *timeVariable,
    int namelen,
    int descLen,
    int unitLen)

__declspec(dllexport) void __cdecl varsesti(
    char *instance,
    const int *noVars,
    double *vars,
    double *auxStates,
    const int *index,
    double *result)

__declspec(dllexport) void __cdecl varsbnds(
    char *instance,
    const int *index,
    double *lowerBound,
    double *upperBound)
```

Fortran interface

USER-DEFINED MODELS

```
subroutine nocvars (instance, result)
  cDEC$ ATTRIBUTES DLLEXPORT :: nocvars
  character *(*) instance
  integer result

subroutine varsinfo (instance, index, name, desc, unit, isconn, timevar)
  cDEC$ ATTRIBUTES DLLEXPORT :: varsinfo
  character *(*) instance
  integer index
  character*(80) name,desc,unit
  integer isconn, timevavr

subroutine varsesti (instance, novars, vars, auxstats, index, result)
  cDEC$ ATTRIBUTES DLLEXPORT :: varsesti
  character *(*) instance
  integer novars, index
  double precision vars(novars),auxstats(*)
  double precision result

subroutine varsbnds (instance, index, lowbnd, uppbnd)
  cDEC$ ATTRIBUTES DLLEXPORT :: varsbnds
  character *(*) instance
  integer index
  double precision lowbnd, uppbnd
```

11.1.5 Model Constraints Routines

Description

These routines allows the definition of constraints by the user which are to be considered in the optimal control problem. In order to consider a user-coded constraint in the optimal control problem, a corresponding constraint has to be declared in the *Constraints* configuration panel of the *ASTOS GUI* as well (see Section 6.9). The routines are available for the following user models:

- Actuators (see Section 11.2)
- Aerodynamics (see Section 11.3)
- Components (see Section 11.5)

Configuration

The functions given below are part of the constraints:

1. **noconst** returns the number `Result` of constraints. If no constraints are defined, the returned value must be zero. `Instance` is the only input parameter of this routine.

USER-DEFINED MODELS

2. **constinf** provides additional information about the constraints and is called once for each constraint specified by the input parameter *Index*. Another input parameter is *Instance*. Output parameters are the strings *Name* and *Unit* with a maximum length of 80 characters. *Name* has to be identical to the name given by the *DLL constraint name* parameter in the corresponding *Constraints* configuration panel. *Unit* must be a valid ASTOS SI-unit string (please refer to Available Units in the User Manual) and should correspond to the unit of the limit or reference defined in the *Constraints* configuration panel.
3. **constval** returns the value (*Result*) of each constraint specified by the input parameter *Index*. As basis for the calculation an array of auxiliary variables (*Vars*), its length (*No_Vars*), an array of auxiliary states (*Aux_States*), an array of control variables (*Control_Vars*) and an array of control derivatives (*Control_Derivs*) are given as further input parameters. Another input parameter is *Instance*.

Note: An array of the auxiliary states is not provided in case of an **Aerodynamics** model.

Note: In case of an **Actuator** model an additional array of the current propellant masses (*Prop_Masses*) is provided as input parameter.

4. **constlim** is an optional routine similar to **constval**. But it returns the constraint limit (*Result*) for each constraint specified by the input parameter *Index*. This limit value can be understood as 'reference' value and overwrites the specified limit value in the corresponding *Constraints* configuration panel (either *Upper_Limit*, *Lower_Limit* or *Reference*). Input parameters are *Instance*, an array of auxiliary variables (*Vars*) and its length (*No_Vars*), an array of auxiliary states (*Aux_States*), an array of control variables (*Control_Vars*) and an array of control derivatives (*Control_Derivs*).

Note: An array of the auxiliary states is not provided in case of an **Aerodynamics** model.

Note: In case of an **Actuator** model an additional array of the current propellant masses (*Prop_Masses*) is provided as input parameter.

Ada interface

```
procedure Number_Constraints
  (Instance_Adr : in System.Address;
   Result_Adr   : in System.Address)
is
  Instance : System.Address;
  for Instance'Address use Instance_Adr;
  Result : Natural;
  for Result'Address use Result_Adr;
pragma Export (C, Number_Constraints, "noconst");
```

USER-DEFINED MODELS

```
procedure Constraint_Info
  (Instance_Adr : in System.Address;
   Index_Adr    : in System.Address;
   Name_Adr     : in System.Address;
   Unit_Adr     : in System.Address;
   Name_Len_Adr : in System.Address;
   Unit_Len_Adr : in System.Address)
is
  Instance : System.Address;
  for Instance'Address use Instance_Adr;
  Index : Natural;
  for Index'Address use Index_Adr;
  Name : Item_String;
  for Name'Address use Name_Adr;
  Unit : Item_String;
  for Unit'Address use Unit_Adr;
pragma Export (C, Constraint_Info, "constinf");

procedure Constraint_Value
  (Instance_Adr      : in System.Address;
   Index_Adr         : in System.Address;
   No_Vars_Adr       : in System.Address;
   Vars_Adr          : in System.Address;
   Prop_Masses_Adr   : in System.Address;
   Aux_States_Adr    : in System.Address;
   Control_Vars_Adr  : in System.Address;
   Control_Derivs_Adr : in System.Address;
   Result_Adr        : in System.Address)
is
  Instance : System.Address;
  for Instance'Address use Instance_Adr;
  Index : Natural;
  for Index'Address use Index_Adr;
  No_Vars : Integer;
  for No_Vars'Address use No_Vars_Adr;
  Vars : array (1..No_Vars) of Long_Float;
  for Vars'Address use Vars_Adr;
  Prop_Masses : array (1 .. No_Massflows);
  for Prop_Masses'Address use Prop_Masses_Adr;
  Aux_States : array (1 .. No_Aux_States);
  for Aux_States'Address use Aux_States_Adr;
  Control_Vars : array (1 .. No_Ctrl_Vars) of Long_Float;
  for Control_Vars'Address use Control_Vars_Adr;
  Control_Derivs : array (1 .. No_Ctrl_Vars) of Long_Float;
  for Control_Derivs'Address use Control_Derivs_Adr;
  Result : Long_Float;
```

USER-DEFINED MODELS

```
    for Result'Address use Result_Adr;
pragma Export (C, Constraint_Value, "constval");

procedure Constraint_Limit
  (Instance_Adr      : in System.Address;
   Index_Adr        : in System.Address;
   No_Vars_Adr       : in System.Address;
   Vars_Adr          : in System.Address;
   Prop_Masses_Adr   : in System.Address;
   Aux_States_Adr    : in System.Address;
   Control_Vars_Adr  : in System.Address;
   Control_Derivs_Adr : in System.Address;
   Result_Adr        : in System.Address)
is
  Instance : System.Address;
  for Instance'Address use Instance_Adr;
  Index : Natural;
  for Index'Address use Index_Adr;
  No_Vars : Integer;
  for No_Vars'Address use No_Vars_Adr;
  Vars : array (1..No_Vars) of Long_Float;
  for Vars'Address use Vars_Adr;
  Prop_Masses : array (1 .. No_Massflows);
  for Prop_Masses'Address use Prop_Masses_Adr;
  Aux_States : array (1 .. No_Aux_States);
  for Aux_States'Address use Aux_States_Adr;
  Control_Vars : array (1 .. No_Ctrl_Vars) of Long_Float;
  for Control_Vars'Address use Control_Vars_Adr;
  Control_Derivs : array (1 .. No_Ctrl_Vars) of Long_Float;
  for Control_Derivs'Address use Control_Derivs_Adr;
  Result : Long_Float;
  for Result'Address use Result_Adr;
pragma Export (C, Constraint_Limit, "constlim");
```

C interface

```
__declspec(dllexport) void __cdecl noconst(
  char *instance,
  int *result)

__declspec(dllexport) void __cdecl constinf(
  char *instance,
  const int *index,
  char *name,
  char *unit,
  int nameLen,
```

USER-DEFINED MODELS

```
int unitLen)

__declspec(dllexport) void __cdecl constval(
char *instance,
const int *index,
const int *noVars,
double *vars,
double *propMass,
double *auxStates,
double *ctrlVars,
double *ctrlDerivs,
double *result)

__declspec(dllexport) void __cdecl constlim(
char *instance,
const int *index,
const int *noVars,
double *vars,
double *propMass,
double *auxStates,
double *ctrlVars,
double *ctrlDerivs,
double *result)
```

Fortran interface

```
subroutine noconst (instance, result)
  cDEC$ ATTRIBUTES DLLEXPORT :: noconst
  character *(*) instance
  integer result

subroutine constinf (instance, index, name, unit)
  cDEC$ ATTRIBUTES DLLEXPORT :: constinf
  character *(*) instance
  integer index
  character*(80) name,unit

subroutine constval (instance, index, novars, vars, propmass, auxstats, ctrlvars, ctrldrvs, result)
  cDEC$ ATTRIBUTES DLLEXPORT :: constval
  character *(*) instance
  integer index, novars
  double precision vars(novars), propmass(*), auxstats(*), ctrlvars(*), ctrldrvs(*)
  double precision result

subroutine constlim (instance, index, novars, vars, propmass, auxstats, ctrlvars, ctrldrvs, result)
```

USER-DEFINED MODELS

```
    ctrldrvs, result)
cDEC$ ATTRIBUTES DLLEXPORT :: constlim
character *(*) instance
integer index, novars
double precision vars(novars), propmass(*), auxstats(*), ctrlvars(*), ctrldrvs(*)
double precision result
```

11.1.6 Model Cost Functions Routines

Description

These optional routines allows the definition of a cost function in the optimization problem. They are available for the following user models:

- Actuators (see Section 11.2)
- Aerodynamics (see Section 11.3)
- Components (see Section 11.5)

Configuration

Two routines are present:

1. **costinfo** returns the name (Name), description (Desc) and unit (Unit) of the cost function as strings with a maximum length of 80 characters. Unit must be a valid ASTOS SI-unit string (please refer to Available Units in the User Manual). The only input parameter is Instance.
2. **costval** returns the actual cost function value (Result). To calculate the cost value an array of auxiliary variables (Vars) and its length (No_Vars), an array of propellant masses (Prop_Masses), an array of auxiliary states (Aux_States), an array of control variables (Control_Vars) and an array of control derivatives (Control_Derivs) of the user-defined model are provided as input parameters. Another input parameter is Instance.

Ada interface

```
procedure Cost_Function_Info
  (Instance_Adr : in System.Address;
   Name_Adr     : in System.Address;
   Desc_Adr     : in System.Address;
   Unit_Adr     : in System.Address;
   Name_Len     : in Integer;
   Desc_Len     : in Integer;
   Unit_Len     : in Integer)
is
  Instance : System.Address;
```

USER-DEFINED MODELS

```
    for Instance'Address use Instance_Adr;
    Name : Item_String;
    for Name'Address use Name_Adr;
    Desc : Item_String;
    for Desc'Address use Desc_Adr;
    Unit : Item_String;
    for Unit'Address use Unit_Adr;
pragma Export (C, Cost_Function_Info, "costinfo");

procedure Cost_Function_Value
  (Instance_Adr      : in System.Address;
   No_Vars_Adr       : in System.Address;
   Vars_Adr          : in System.Address;
   Prop_Masses_Adr   : in System.Address;
   Aux_States_Adr    : in System.Address;
   Control_Vars_Adr  : in System.Address;
   Control_Derivs_Adr : in System.Address;
   Result_Adr        : in System.Address)
is
  Instance : System.Address;
  for Instance'Address use Instance_Adr;
  No_Vars : Integer;
  for No_Vars'Address use No_Vars_Adr;
  Vars : array (1..No_Vars) of Long_Float;
  for Vars'Address use Vars_Adr;
  Prop_Masses : array (1 .. No_Massflows);
  for Prop_Masses'Address use Prop_Masses_Adr;
  Aux_States : array (1 .. No_Aux_States) of Long_Float;
  for Aux_States'Address use Aux_States_Adr;
  Control_Vars : array (1 .. No_Ctrl_Vars) of Long_Float;
  for Control_Vars'Address use Control_Vars_Adr;
  Control_Derivs : array (1 .. No_Ctrl_Vars) of Long_Float;
  for Control_Derivs'Address use Control_Derivs_Adr;
  Result : Long_Float;
  for Result'Address use Result_Adr;
pragma Export (C, Cost_Function_Value, "costval");
```

C interface

```
__declspec(dllexport) void __cdecl costinfo(
  char *instance,
  char *name,
  char *desc,
  char *unit,
  int nameLen,
  int descLen,
```


USER-DEFINED MODELS

```
int unitLen)

__declspec(dllexport) void __cdecl costval(
char *instance,
const int *noVars,
double *vars,
double *propMass,
double *auxStates,
double *ctrlVars,
double *ctrlDerivs,
double *result)
```

Fortran interface

```
subroutine costinfo (instance, name, desc, unit)
  cDEC$ ATTRIBUTES DLLEXPORT :: costinfo
  character *(*) instance
  character*(80) name, desc, unit

subroutine costval (instance, novars, vars, propmass, auxstats, ctrlvars, ctrldrvs, result)
  cDEC$ ATTRIBUTES DLLEXPORT :: costval
  character *(*) instance
  integer novars
  double precision vars(novars), propmass(*), auxstats(*), ctrlvars(*), ctrldrvs(*)
  double precision result
```

11.1.7 Model Algebraic Functions Routines

Description

Further routines are provided to have user-defined time-dependent auxiliary output functions. They are not called during optimizations but only during simulations. Since no input parameters are available to the routines, any data required for the computation of the output values must be stored in static variables (or common blocks) during the computation (or alternatively the output values are stored once they have been computed). The following user models support algebraic functions:

- Actuators (see Section 11.2)
- Aerodynamics (see Section 11.3)
- Components (see Section 11.5)

Configuration

USER-DEFINED MODELS

The following routines are required:

1. **noalgun** returns the number `Result` of auxiliary functions. If no auxiliary functions are defined, the returned value must be zero. `Instance` is the only input parameter of this routine.
2. **alginfo** provides additional information about the auxiliary functions and is called once for every auxiliary function specified by the input parameter `Index`. Another input parameter is `Instance`. Output parameters are the strings `Name` (a unique identifier of an auxiliary function), `Desc` (a description of the auxiliary function) and the unit `Unit` with a maximum length of 80 characters. `Unit` must be a valid *ASTOS* SI-unit string (please refer to Available Units in the User Manual).
3. **algun** returns an array of double values (`Result`) containing one value for each corresponding auxiliary function. The only input parameter is `Instance`.

Ada interface

```
procedure Number_Algebraic_Functions
  (Instance_Adr : in System.Address;
   Result_Adr   : in System.Address)
is
  Instance : System.Address;
  for Instance'Address use Instance_Adr;
  Result : Natural;
  for Result'Address use Result_Adr;
pragma Export (C, Number_Algebraic_Functions, "noalgun");

procedure Algebraic_Functions_Info
  (Instance_Adr : in System.Address;
   Index_Adr    : in System.Address;
   Name_Adr     : in System.Address;
   Desc_Adr     : in System.Address;
   Unit_Adr     : in System.Address;
   Name_Len     : in Integer;
   Desc_Len     : in Integer;
   Unit_Len     : in Integer)
is
  Instance : System.Address;
  for Instance'Address use Instance_Adr;
  Index : Natural;
  for Index'Address use Index_Adr;
  Name : Item_String;
  for Name'Address use Name_Adr;
  Desc : Item_String;
  for Desc'Address use Desc_Adr;
  Unit : Item_String;
  for Unit'Address use Unit_Adr;
pragma Export (C, Algebraic_Functions_Info, "alginfo");
```

USER-DEFINED MODELS

```
procedure Algebraic_Functions
  (Instance_Adr : in System.Address;
   Result_Adr   : in System.Address)
is
  Instance : System.Address;
  for Instance'Address use Instance_Adr;
  Result : Array (1 .. No_Algebraic_Func) of Long_Float;
  for Result'Address use Result_Adr;
pragma Export (C, Algebraic_Functions, "algfunc");
```

C interface

```
__declspec(dllexport) void __cdecl noalgfun(
  char *instance,
  int *result)

__declspec(dllexport) void __cdecl alginfo(
  char *instance,
  const int *index,
  char *name,
  char *desc,
  char *unit,
  int nameLen,
  int descLen,
  int unitLen)

__declspec(dllexport) void __cdecl algfunc(
  char *instance,
  double *result)
```

Fortran interface

```
subroutine noalgfun (instance, result)
  cDEC$ ATTRIBUTES DLLEXPORT :: noalgfun
  character *(*) instance
  integer result

subroutine alginfo (instance, index, name, desc, unit)
  cDEC$ ATTRIBUTES DLLEXPORT :: alginfo
  character *(*) instance
  integer index
  character*(80) name, desc, unit
```

USER-DEFINED MODELS

```
subroutine algfunc (instance, result)
  cDEC$ ATTRIBUTES DLLEXPORT :: algfunc
  character *(*) instance
  double precision result (*)
```

11.1.8 Model Info Routines

Description

These routines collect required information for the *Result Summary*. Any data created by these routines will be printed in the model definition section of the *Scenario Summary* (see chapter Scenario Summary in User Manual). Besides, the information is also stored in the STRUCT file and can be accessed by **Reports**. All user models are supported.

Configuration

Two routines are provided:

1. **noinfo** returns the number `Result` of information objects. If no information objects are defined, the returned value must be zero. `Instance` is the only input parameter of this routine.
2. **info** returns the strings `Name` and `Value` with a maximum length of 80 characters for the specified `Index` and `Instance`. Each information object consisting of one name and one value is separately printed in the corresponding section of the *Result Summary* depending on the kind of the user model. Further input parameters are an array of auxiliary states (`Aux_States`) and an array of control variables (`Control_Vars`).

Note: In case of a user **Aerodynamics** model the provided array `Aux_States` has zero length because it is not supported!

Note: In case of a user **Atmosphere** model the provided arrays `Aux_States` and `Control_Vars` have zero length because they are not supported!

Ada interface

```
procedure Number_Info
  (Instance_Adr : in System.Address;
   Result_Adr   : in System.Address)
is
  Instance : System.Address;
  for Instance'Address use Instance_Adr;
  Result : Natural;
  for Result'Address use Result_Adr;
pragma Export (C, Number_Info, "noinfo");
```

USER-DEFINED MODELS

```
procedure Info
  (Instance_Adr      : in System.Address;
   Index_Adr         : in System.Address;
   Aux_States_Adr    : in System.Address;
   Control_Vars_Adr  : in System.Address;
   Name_Adr          : in System.Address;
   Value_Adr         : in system.Address;
   Name_Len          : in Integer;
   Value_Len         : in Integer)
is
  Instance : System.Address;
  for Instance'Address use Instance_Adr;
  Index : Natural;
  for Index'Address use Index_Adr;
  Aux_States : array (1 .. No_Aux_States) of Long_Float;
  for Aux_States'Address use Aux_States_Adr;
  Control_Vars : array (1 .. No_Control_Vars) of Long_Float;
  for Control_Vars'Address use Control_Vars_Adr;
  Name : Item_String;
  for Name'Address use Name_Adr;
  Value : Item_String;
  for Value'Address use Value_Adr;
pragma Export (C, Info, "info");
```

C interface

```
__declspec(dllexport) void __cdecl noinfo(
  char *instance,
  int *result)

__declspec(dllexport) void __cdecl info(
  char *instance,
  const int *index,
  double *auxStates,
  double *controlVars,
  char *name,
  char *value,
  int nameLen,
  int valueLen)
```

Fortran interface

```
subroutine noinfo (instance, result)
```

USER-DEFINED MODELS

```
cDEC$ ATTRIBUTES DLLEXPORT :: noinfo
character *(*) instance
integer result

subroutine info (instance, index, auxstats, ctrlvars, name, value)
  cDEC$ ATTRIBUTES DLLEXPORT :: info
  character *(*) instance
  integer index
  double precision auxstats(*), ctrlvars(*)
  character*(80) name, value
```

11.2 User-Defined Actuator Model Routines

Description

These routines allow the design of a custom actuator model, for example a hybrid propulsion system. An overview of the corresponding configuration panel is given in Section 3.2.2.9. This user-defined model supports the following common routines:

- **Initialization**, see Section 11.1.1
- **Finalization**, see Section 11.1.2
- **Auxiliary States**, see Section 11.1.3
- **Control Variables**, see Section 11.1.4
- **Constraints**, see Section 11.1.5
- **Cost Function**, see Section 11.1.6
- **Algebraic Functions**, see Section 11.1.7
- **Info**, see Section 11.1.8

Configuration

Furthermore, additional actuator model-specific routines are available:

1. **nomassfl** returns the number `Result` of mass flows of the actuator model. This value must be equal to the number of tanks of the associated component (e.g. stage). If no mass flows are present, the returned value must be zero. `Instance` is the only input parameter of this routine.
2. **compute** evaluates the mass flow rates (`Flow_Rates`) as well as force (`Force`) and moment vector (`Moment`) represented in the body fixed frame. Input parameters to this routine are an array of auxiliary variables (`Vars`) and their number (`No_Vars`), an array of the actual propellant masses of the associated tanks (`Prop_Masses`), an array of auxiliary states (`Aux_States`), an array of control variables (`Control_Vars`) and the `Instance`.

USER-DEFINED MODELS

3. **massstru** is optional and returns the structural mass Result (i.e. engine mass) of the actuator. The input parameters are the Instance, an array of auxiliary states (Aux_States) and an array of the control variables (Control_Vars). If this routine is not provided, the structural mass is assumed to be zero.

Ada interface

```
procedure Number_Massflows
  (Instance_Adr : in System.Address;
   Result_Adr   : in System.Address)
is
  Instance : System.Address;
  for Instance'Address use Instance_Adr;
  Result : Natural;
  for Result'Address use Result_Adr;
pragma Export (C, Number_Massflows, "nomassfl");

procedure Compute
  (Instance_Adr      : in System.Address;
   No_Vars_Adr       : in System.Address;
   Vars_Adr          : in System.Address;
   Prop_Masses_Adr   : in System.Address;
   Aux_States_Adr    : in System.Address;
   Control_Vars_Adr  : in System.Address;
   Flowrate_Adr      : in System.Address;
   Force_Adr         : in System.Address;
   Moment_Adr        : in System.Address)
is
  Instance : System.Address;
  for Instance'Address use Instance_Adr;
  No_Vars : Natural;
  for No_Vars'Address use No_Vars_Adr;
  Vars : Array (1 .. No_Vars) of Long_Float;
  for Vars'Address use Vars_Adr;
  Prop_Masses : array (1 .. No_Massflows) of Long_Float;
  for Prop_Masses'Address use Prop_Masses_Adr;
  Aux_States : array (1 .. No_Aux_States) of Long_Float;
  for Aux_States'Address use Aux_States_Adr;
  Control_Vars : array (1 .. No_Ctrl_Vars) of Long_Float;
  for Control_Vars'Address use Control_Vars_Adr;
  Flowrate : array (1 .. No_Massflows) of Long_Float;
  for Flowrate'Address use Flowrate_Adr;
  Force : array (1 .. 3) of Long_Float;
  for Force'Address use Force_Adr;
  Moment : array (1 .. 3) of Long_Float;
  for Moment'Address use Moment_Adr;
pragma Export (C, Compute, "compute");
```

USER-DEFINED MODELS

```
procedure Structural_Mass
  (Instance_Adr      : in System.Address;
   Aux_States_Adr    : in System.Address;
   Control_Vars_Adr  : in System.Address;
   Result_Adr        : in System.Address)
is
  Instance : System.Address;
  for Instance'Address use Instance_Adr;
  Aux_States : array (1 .. No_Aux_States) of Long_Float;
  for Aux_States'Address use Aux_States_Adr;
  Control_Vars : array (1 .. No_Control_Vars) of Long_Float;
  for Control_Vars'Address use Control_Vars_Adr;
  Result : Long_Float;
  for Result'Address use Result_Adr;
pragma Export (C, Structural_Mass, "massstru");
```

C interface

```
__declspec(dllexport) void __cdecl nomassfl(
  char *instance,
  int *result)

__declspec(dllexport) void __cdecl compute(
  char *instance,
  int *novars,
  double *vars,
  double *propMasses,
  double *auxStates,
  double *ctrlVars,
  double *flowRates,
  double *forceVector;
  double *momentVector)

__declspec(dllexport) void __cdecl massstru(
  char *instance,
  double *auxStates,
  double *ctrlvars,
  double *result)
```

Fortran interface

```
subroutine nomassfl (instance, result)
  CDEC$ ATTRIBUTES DLLEXPORT :: nomassfl
```


USER-DEFINED MODELS

```
character *(*) instance
integer result

subroutine compute (instance, novars, vars, propmass, auxstats, ctrlvars, flow, force, moment)
  cDEC$ ATTRIBUTES DLLEXPORT :: compute
  character *(*) instance
  integer novars
  double precision vars(novars), propmass(*), auxstats(*), ctrlvars(*)
  double precision flow(*), force(3), moment(3)

subroutine massstru (instance, auxstats, ctrlvars, result)
  cDEC$ ATTRIBUTES DLLEXPORT :: massstru
  character *(*) instance
  double precision auxstats(*), ctrlvars(*)
  double precision result
```

11.3 User-Defined Aerodynamics Model Routines

Description

These routines can be used to program a custom aerodynamics model, for example a six degrees-of-freedom aerodynamics model for a winged body. An overview of the corresponding configuration panel is given in Section 3.5.1.5. This user-defined model supports the following common routines:

- **Initialization**, see Section 11.1.1
- **Finalization**, see Section 11.1.2
- **Control Variables**, see Section 11.1.4
- **Constraints**, see Section 11.1.5
- **Cost Function**, see Section 11.1.6
- **Algebraic Functions**, see Section 11.1.7
- **Info**, see Section 11.1.8

Configuration

Furthermore, additional specific routines are available:

1. **compute** evaluates the aerodynamic force (Force) and moment (Moment) vector represented in the body-axis frame. Input parameters are Instance, an array of auxiliary variables (Vars) and their number (No_Vars) and an array of control variables (Control_Vars).

USER-DEFINED MODELS

2. **aref** returns the reference area (Result). Input parameters are Instance and an array of control variables (Control_Vars).
3. **lref** returns the reference length (Result). Input parameters are Instance and an array of control variables (Control_Vars).

Ada interface

```
procedure Compute
  (Instance_Adr      : in System.Address;
   No_Vars_Adr       : in System.Address;
   Vars_Adr          : in System.Address;
   Control_Vars_Adr  : in System.Address;
   Force_Adr         : in System.Address;
   Moment_Adr        : in System.Address)
is
  Instance : System.Address;
  for Instance'Address use Instance_Adr;
  No_Vars : Natural;
  for No_Vars'Address use No_Vars_Adr;
  Vars : Array (1 .. No_Vars) of Long_Float;
  for Vars'Address use Vars_Adr;
  Control_Vars : array (1 .. No_Ctrl_Vars) of Long_Float;
  for Control_Vars'Address use Control_Vars_Adr;
  Force : array (1 .. 3) of Long_Float;
  for Force'Address use Force_Adr;
  Moment : array (1 .. 3) of Long_Float;
  for Moment'Address use Moment_Adr;
pragma Export (C, Compute, "compute");

procedure Reference_Area
  (Instance_Adr      : in System.Address;
   Control_Vars_Adr  : in System.Address;
   Result_Adr        : in System.Address)
is
  Instance : System.Address;
  for Instance'Address use Instance_Adr;
  Control_Vars : array (1 .. No_Params) of Long_Float;
  for Control_Vars'Address use Control_Vars_Adr;
  Result : Long_Float;
  for Result'Address use Result_Adr;
pragma Export (C, Reference_Area, "aref");

procedure Reference_Length
  (Instance_Adr      : in System.Address;
   Control_Vars_Adr  : in System.Address;
   Result_Adr        : in System.Address)
```

USER-DEFINED MODELS

```
is
  Instance : System.Address;
  for Instance'Address use Instance_Adr;
  Control_Vars : array (1 .. No_Params) of Long_Float;
  for Control_Vars'Address use Control_Vars_Adr;
  Result : Long_Float;
  for Result'Address use Result_Adr;
pragma Export (C, Reference_Length, "lref");
```

C interface

```
__declspec(dllexport) void __cdecl compute(
  char *instance,
  const int *noVars,
  double *vars,
  double *ctrlVars,
  double *force,
  double *moment)

__declspec(dllexport) void __cdecl aref(
  char *instance,
  double *ctrlVars,
  double *result)

__declspec(dllexport) void __cdecl lref(
  char *instance,
  double *ctrlVars,
  double *result)
```

Fortran interface

```
subroutine compute (instance, novars, vars, ctrlvars, force, moment)
  cDEC$ ATTRIBUTES DLLEXPORT :: compute
  character *(*) instance
  integer novars
  double precision vars(novars), ctrlvars(*)
  double precision force(3), moment(3)

subroutine aref (instance, ctrlvars, result)
  cDEC$ ATTRIBUTES DLLEXPORT :: aref
  character *(*) instance
  double precision ctrlvars(*)
  double precision result
```

USER-DEFINED MODELS

```
subroutine lref (instance, ctrlvars, result)
  cDEC$ ATTRIBUTES DLLEXPORT :: lref
  character *(*) instance
  double precision ctrlvars(*)
  double precision result
```

11.4 User-Defined Atmosphere Model Routines

Description

These routines can be used to program a custom atmosphere model. An overview of the corresponding configuration panel is given in Section 2.3.7. This user-defined model supports the following common routines:

- **Initialization**, see Section 11.1.1
- **Finalization**, see Section 11.1.2
- **Info**, see Section 11.1.8

Description

Furthermore, additional specific routines are given below. Each of them has five input parameters: the instance pointer *Instance*, the altitude (*Altitude*), the longitude (*Longitude*), the geodetic latitude (*Latitude*) and the Julian date (TT) (*Date*).

1. **compute** evaluates and returns the atmospheric properties density (*Density*), ambient pressure (*Pressure*) and speed of sound (*Speed_Of_Sound*).
2. **temp** is optional and returns the ambient temperature (*Temperature*).
3. **dynvisc** is optional and returns the dynamic viscosity (*Dyn_Viscosity*).
4. **densoz** is optional and returns the ozone density (*Ozone_Dens*).

Ada interface

```
procedure Compute
  (Instance_Adr      : in System.Address;
   Altitude_Adr      : in System.Address;
   Longitude_Adr     : in System.Address;
   Latitude_Adr      : in System.Address;
   Date_Adr          : in System.Address;
   Density_Adr       : in System.Address;
   Pressure_Adr      : in System.Address;
   Speed_Of_Sound_Adr : in System.Address)
is
  Instance : System.Address;
```

USER-DEFINED MODELS

```
    for Instance'Address use Instance_Adr;
    Altitude : Long_Float;
    for Altitude'Address use Altitude_Adr;
    Longitude : Long_Float;
    for Longitude'Address use Longitude_Adr;
    Latitude : Long_Float;
    for Latitude'Address use Latitude_Adr;
    Date : Long_Float;
    for Date'Address use Date_Adr;
    Density : Long_Float;
    for Density'Address use Density_Adr;
    Pressure : Long_Float;
    for Pressure'Address use Pressure_Adr;
    Speed_Of_Sound : Long_Float;
    for Speed_Of_Sound'Address use Speed_Of_Sound_Adr;
pragma Export (C, Compute, "compute");
```

procedure Temperature

```
(Instance_Adr  : in System.Address;
 Altitude_Adr  : in System.Address;
 Longitude_Adr : in System.Address;
 Latitude_Adr  : in System.Address;
 Date_Adr      : in System.Address;
 Result_Adr    : in System.Address)
```

is

```
    Instance : System.Address;
    for Instance'Address use Instance_Adr;
    Altitude : Long_Float;
    for Altitude'Address use Altitude_Adr;
    Longitude : Long_Float;
    for Longitude'Address use Longitude_Adr;
    Latitude : Long_Float;
    for Latitude'Address use Latitude_Adr;
    Date : Long_Float;
    for Date'Address use Date_Adr;
    Result : Long_Float;
    for Result'Address use Result_Adr;
pragma Export (C, Temperature, "temp");
```

procedure Dynamic_Viscosity

```
(Instance_Adr  : in System.Address;
 Altitude_Adr  : in System.Address;
 Longitude_Adr : in System.Address;
 Latitude_Adr  : in System.Address;
 Date_Adr      : in System.Address;
 Result_Adr    : in System.Address)
```

is

USER-DEFINED MODELS

```
Instance : System.Address;
for Instance'Address use Instance_Adr;
Altitude : Long_Float;
for Altitude'Address use Altitude_Adr;
Longitude : Long_Float;
for Longitude'Address use Longitude_Adr;
Latitude : Long_Float;
for Latitude'Address use Latitude_Adr;
Date : Long_Float;
for Date'Address use Date_Adr;
Result : Long_Float;
for Result'Address use Result_Adr;
pragma Export (C, Dynamic_Viscosity, "dynvisc");

procedure Ozone_Density
  (Instance_Adr : in System.Address;
   Altitude_Adr : in System.Address;
   Longitude_Adr : in System.Address;
   Latitude_Adr : in System.Address;
   Date_Adr : in System.Address;
   Result_Adr : in System.Address)
is
  Instance : System.Address;
  for Instance'Address use Instance_Adr;
  Altitude : Long_Float;
  for Altitude'Address use Altitude_Adr;
  Longitude : Long_Float;
  for Longitude'Address use Longitude_Adr;
  Latitude : Long_Float;
  for Latitude'Address use Latitude_Adr;
  Date : Long_Float;
  for Date'Address use Date_Adr;
  Result : Long_Float;
  for Result'Address use Result_Adr;
  pragma Export (C, Ozone_Density, "densoz");
```

C interface

```
__declspec(dllexport) void __cdecl compute(
  char *instance,
  const double *altitude,
  const double *longitude,
  const double *latitude,
  const double *date,
  double *density,
  double *pressure,
```

USER-DEFINED MODELS

```
double *sonicvel)

__declspec(dllexport) void __cdecl temp(
    char *instance,
    const double *altitude,
    const double *longitude,
    const double *latitude,
    const double *date,
    double *result)

__declspec(dllexport) void __cdecl dynvisc(
    char *instance,
    const double *altitude,
    const double *longitude,
    const double *latitude,
    const double *date,
    double *result)

__declspec(dllexport) void __cdecl densoz(
    char *instance,
    const double *altitude,
    const double *longitude,
    const double *latitude,
    const double *date,
    double *result)
```

Fortran interface

```
subroutine compute (instance, alti, long, lati, date, dens, press, sos)
  cDEC$ ATTRIBUTES DLLEXPORT :: compute
  character (*) instance
  double precision alti, long, lati, date
  double precision dens, press, sos

subroutine temp (instance, alti, long, lati, date, result)
  cDEC$ ATTRIBUTES DLLEXPORT :: compute
  character (*) instance
  double precision alti, long, lati, date
  double precision result

subroutine dynvisc (instance, alti, long, lati, date, result)
  cDEC$ ATTRIBUTES DLLEXPORT :: compute
  character (*) instance
  double precision alti, long, lati, date
  double precision result
```

USER-DEFINED MODELS

```
subroutine densoz (instance, alti, long, lati, date, result)
  cDEC$ ATTRIBUTES DLLEXPORT :: compute
  character *(*) instance
  double precision alti, long, lati, date
  double precision result
```

11.5 User-Defined Component Model Routines

Description

These routines can be used to program an own vehicle component model, for example an advanced stage with multiple tanks. An overview of the corresponding configuration panel is given in Section 3.3.5. This user-defined model supports the following common routines:

- **Initialization**, see Section 11.1.1
- **Finalization**, see Section 11.1.2
- **Auxiliary States**, see Section 11.1.3
- **Control Variables**, see Section 11.1.4
- **Constraint**, see Section 11.1.5
- **Cost Function**, see Section 11.1.6
- **Algebraic Functions**, see Section 11.1.7
- **Info**, see Section 11.1.8

Configuration

Furthermore, additional specific routines are available:

1. **massstru** returns the structural mass `Result` of the component. The input parameters are `Instance`, an array of auxiliary states (`Aux_States`) and an array of control variables (`Control_Vars`).
2. **notanks** returns the number of integrated tanks. The only input parameter is the instance pointer `Instance`.
3. **massprop** returns an array of propellant masses (`Result`). The input parameters are `Instance`, an array of auxiliary states (`Aux_States`) and an array of control variables (`Control_Vars`).
4. **maxmassp** returns an array of maximum propellant masses (`Result`) which represents the upper bounds of all propellant mass states. In contrast to the interface **massprop** it should consider the upper bound of an available tank sizing parameter. The length of the array must be equal to the number of tanks. Input parameters are `Instance`, an array of auxiliary states (`Aux_States`) and an array of control variables (`Control_Vars`).
5. **isoptim** should return 1 if the propellant masses are optimizable, otherwise 0. The only input parameter is the instance pointer `Instance`.

Ada interface

```

procedure Structural_Mass
  (Instance_Adr      : in System.Address;
   Aux_States_Adr    : in System.Address;
   Control_Vars_Adr  : in System.Address;
   Result_Adr        : in System.Address)
is
  Instance : System.Address;
  for Instance'Address use Instance_Adr;
  Aux_States : array (1 .. No_Aux_States) of Long_Float;
  for Aux_States'Address use Aux_States_Adr;
  Control_Vars : array (1 .. No_Control_Vars) of Long_Float;
  for Control_Vars'Address use Control_Vars_Adr;
  Result : Long_Float;
  for Result'Address use Result_Adr;
pragma Export (C, Structural_Mass, "massstru");

procedure Number_Tanks
  (Instance_Adr : in System.Address;
   Result_Adr   : in System.Address)
is
  Instance : System.Address;
  for Instance'Address use Instance_Adr;
  Result : Natural;
  for Result'Address use Result_Adr;
pragma Export (C, Number_Tanks, "notanks");

procedure Propellant_Masses
  (Instance_Adr      : in System.Address;
   Aux_States_Adr    : in System.Address;
   Control_Vars_Adr  : in System.Address;
   Result_Adr        : in System.Address)
is
  Instance : System.Address;
  for Instance'Address use Instance_Adr;
  Aux_States : array (1 .. No_Aux_States) of Long_Float;
  for Aux_States'Address use Aux_States_Adr;
  Control_Vars : array (1 .. No_Control_Vars) of Long_Float;
  for Control_Vars'Address use Control_Vars_Adr;
  Result : array (1 .. No_Tanks) of Long_Float;
  for Result'Address use Result_Adr;
pragma Export (C, Propellant_Masses, "massprop");

procedure Maximum_Propellant_Masses
  (Instance_Adr      : in System.Address;
   Aux_States_Adr    : in System.Address;

```

USER-DEFINED MODELS

```
Control_Vars_Adr : in System.Address;
Result_Adr       : in System.Address)
is
  Instance : System.Address;
  for Instance'Address use Instance_Adr;
  Aux_States : array (1 .. No_Aux_States) of Long_Float;
  for Aux_States'Address use Aux_States_Adr;
  Control_Vars : array (1 .. No_Control_Vars) of Long_Float;
  for Control_Vars'Address use Control_Vars_Adr;
  Result : array (1 .. No_Tanks) of Long_Float;
  for Result'Address use Result_Adr;
pragma Export (C, Maximum_Propellant_Masses, "maxmassp");

procedure Is_Optimizable
  (Instance_Adr : in System.Address;
   Result_Adr   : in System.Address)
is
  Instance : System.Address;
  for Instance'Address use Instance_Adr;
  Result : Natural;
  for Result'Address use Result_Adr;
pragma Export (C, Is_Optimizable, "isoptim");
```

C interface

```
__declspec(dllexport) void __cdecl massstru(
  char *instance,
  double *auxStates,
  double *ctrlVars,
  double *result)

__declspec(dllexport) void __cdecl notanks(
  char *instance,
  int *result)

__declspec(dllexport) void __cdecl massprop(
  char *instance,
  double *auxStates,
  double *ctrlVars,
  double *result)

__declspec(dllexport) void __cdecl maxmassp(
  char *instance,
  double *auxStates,
  double *ctrlVars,
  double *result)
```

USER-DEFINED MODELS

```
__declspec(dllexport) void __cdecl isoptim(  
    char *instance,  
    int *result)
```

Fortran interface

```
subroutine massstru (instance, auxstats, ctrlvars, result)  
    cDEC$ ATTRIBUTES DLLEXPORT :: massstru  
    character *(*) instance  
    double precision auxstats(*), ctrlvars(*)  
    double precision result  
  
subroutine notanks (instance, result)  
    cDEC$ ATTRIBUTES DLLEXPORT :: notanks  
    character *(*) instance  
    integer result  
  
subroutine massprop (instance, auxstats, ctrlvars, result)  
    cDEC$ ATTRIBUTES DLLEXPORT :: massprop  
    character *(*) instance  
    double precision auxstats(*), ctrlvars(*)  
    double precision result(*)  
  
subroutine maxmassp (instance, auxstats, ctrlvars, result)  
    cDEC$ ATTRIBUTES DLLEXPORT :: maxmassp  
    character *(*) instance  
    double precision auxstats(*), ctrlvars(*)  
    double precision result(*)  
  
subroutine isoptim (instance, result)  
    cDEC$ ATTRIBUTES DLLEXPORT :: isoptim  
    character *(*) instance  
    integer result
```

References

- [1] International Organization for Standardization. *Ada 95 Reference Manual - Language and Standard Libraries*, International Standard ANSI/ISO/IEC 8652, Version 6.0, Intermetrics Inc., 1995
- [2] International Organization for Standardization. *Flight Dynamics - Concepts, Quantities and Symbols*, International Standard ISO 1151, 1988.
- [3] American National Standards Institute. *Recommended Practice for Atmospheric and Space Flight Vehicle Coordinate Systems*, American National Standard ANSI/AIAA R-004-1992, Feb. 1992.
- [4] Roenneke, A. J. *Coordinate Systems and Orthogonal Transformations*, Institute of Flight Mechanics and Control, University of Stuttgart, Germany, IFR TR 94-01.1, Jan. 1995.
- [5] de Boor, D. *A Practical Guide to Splines*, Applied Mathematical Sciences, Vol. 27, Springer Verlag, 1978.
- [6] Akima, H. *A New Method of Interpolation and Smooth Curve Fitting Based on Local Procedures*, J. of Assoc. for Comp. Mech., Vo. 17, No. 4, Oct. 1970, pp.589-602.
- [7] Barnes, J.G.P. *Programming in Ada 95*, Addison-Wesley, 1995.
- [8] Skansholm, J. *Ada from the Beginning*, Addison-Wesley, 1994.
- [9] Bray, G., and Pokrass, D. *Understanding Ada*, Wiley & Sons, 1985.
- [10] Roenneke, A. J. *ALTOS-2 Reentry Problem Set*, Advanced Launch and Reentry Trajectory Optimization Software Documentation, Technical Report, Institute of Flight Mechanics and Control, University of Stuttgart, Rept. IFR TR 95-06, Jan. 1995.
- [11] Amblard, P. *Preliminary Requirements for Stabilizing Function Hermes R-Shape*, Avions Marcel Dassault Breguet Aviation Espace, Rept. H-NT-1C-1004-AMD, Jan. 1989.
- [12] Roenneke, A. J. *Aerodynamic Model of a High-Lift Reentry Vehicle for Controller Design and Simulation*, Institute of Flight Mechanics and Control, University of Stuttgart, Rept. IFR TR 94-04, Nov. 1994.

REFERENCES

- [13] Vignials, I. *Hermes R-Shape Aerodynamic Data Base*, Avions Marcel Dassault Breguet Aviation Espace, Rept. H-NT-1-1074-AMD, Nov. 1988.
- [14] Buhl, W., Ebert, K., Wolff, H. *Technical Report 2 - Modeling*, Advanced Launcher Trajectory Optimization Software Technical Documentation, European Space and Technology Center, Noordwijk, The Netherlands, Contract No. 8046-88-NL-MAC, Nov. 1992.
- [15] Roenneke, A. J., Ludorf, R. *Technical Report 2 - Extension ARTOS*, Advanced Launcher Trajectory Optimization Software Technical Documentation, Draft 1, Institute of Flight Mechanics and Control, University of Stuttgart, Oct. 1993.
- [16] Shau, G.-C. *Der Einfluß flugmechanischer Parameter auf die Aufstiegsbahn von horizontalstartenden Raumtransportern bei gleichzeitiger Bahn- und Stufungsoptimierung*, German Aerospace Research Establishment, DLR Rept. 73-86, Oberpfaffenhofen, Germany, 1973.
- [17] Buhl, W. *Private Communications*, Daimler-Benz Aerospace Co., Ottobrunn, Germany, Aug. 1995.
- [18] Buhl, W. *SSTO Rocket Engine Modeling*, Advanced Launcher Trajectory Optimization Software, Software Verification Document, European Space and Technology Center, Noordwijk, The Netherlands, Contract No. 8046-88-NL-MAC, March 1992.
- [19] Buhl, W. *Approximation of Airbreathing Engines for ALTOS*, Advanced Launcher Trajectory Optimization Software, Software Verification Document, European Space Research and Technology Center, Noordwijk, The Netherlands, Contract No. 8046-88- NL-MAC, March 1992.
- [20] Buhl, W., and Herbst, H. *User Manual of Program PROFIT - Fit of Propulsion System Data*, Messerschmidt- Bölkow-Blohm GmbH, Space Communications and Propulsion Systems Dept., Draft 0, July 1992.
- [21] Schöttle, U., and Rahn, U. *Fahrzeugmodelle für Sensitivitätsstudien konventioneller Trägerraketen*, Institute of Space Systems, University of Stuttgart, Rept. IRS 95-IB-11, Aug. 1995.
- [22] Roenneke, A. J. *Atmospheric Flight Dynamics Based on Flight Path Coordinates*, Institute of Flight Mechanics and Control, University of Stuttgart, Rept. IFR TR 95-08, July 1995.
- [23] Roenneke, A. J. *Space Flight Dynamics Based on Cartesian Velocity Components*, Institute of Flight Mechanics and Control, University of Stuttgart, Rept. IFR TR 94-02, Nov. 1994.
- [24] <http://ccmc.gsfc.nasa.gov/modelweb/atmos/msis.html>

REFERENCES

- [25] <http://ccmc.gsfc.nasa.gov/modelweb/atmos/msise.html>
- [26] <http://ccmc.gsfc.nasa.gov/modelweb/atmos/nrlmsise00.html>
- [27] Justus, C. G., D. L. Johnson. "NASA/TM-1999-209630" *The NASA/MSFC Global Reference Atmospheric Model - 1999 Version (GRAM-99)*, National Aeronautics and Space Administration, Marshall Space Flight Center, Alabama, May 1999.
- [28] Justus, C. G., W. R. Jeffries III, S. P. Yung, and D. L. Johnson. "NASA TM-4715" *The NASA/MSFC Global Reference Atmospheric Model - 1995 Version (GRAM-95)*, National Aeronautics and Space Administration, Marshall Space Flight Center, Alabama, August 1995.
- [29] <http://gcmd.nasa.gov/records/GRAM-99.html>
- [30] Mooij, E. Development of START - A Six Degrees-of-Freedom Simulation Tool for Atmospheric Reentry Trajectories, European Space Research and Technology Center, Noordwijk, The Netherlands, E.W.P. 1633, Oct. 1991.
- [31] Escobal, P. R. *Methods of Orbit Determination*, Krieger Publ., Malabar, Fl., 1976
- [32] Battin, R. H. *Astronautical Guidance*, McGraw-Hill, New York, 1964.
- [33] Kane, T. R., Likins, P. W., Levinson, D.A. *Spacecraft Dynamics*, McGraw-Hill, New York, 1983.
- [34] Regan, F. J. *Re-Entry Vehicle Dynamics*, American Institute of Aeronautics and Astronautics, New York, 1984.
- [35] Miele, A. *Flight Mechanics*, Addison-Wesley, Reading, Mass., 1962.
- [36] Vinh, N. X. *Optimal Trajectories in Atmospheric Flight*, Elsevier, New York, 1981.
- [37] Brockhaus, R. *Flugregelung*, Springer-Verlag, Berlin, 1994.
- [38] Goldstein, H. *Classical Mechanics*, Addison-Wesley, Reading, Mass., 1950.
- [39] Greenwood, D. T. *Principles of Dynamics*, Prentice-Hall, Englewood Cliffs, N. J., 1988.
- [40] U.S. Standard Atmosphere, 1962, NOAA, NASA, US Air Force, Washington, D.C., 1962.
- [41] U.S. Standard Atmosphere, 1976, NOAA, NASA, US Air Force, Washington, D.C., 1976; ANSI/AIAA G-003B-2004, *Guide to Reference and Standard Atmosphere Models*, 2004.

REFERENCES

- [42] Flury, W. Vorlesung Raumfahrtmechanik 1992, ESOC, TH Darmstadt, 1992.
- [43] Bhat, M.S, and Shrivastava, S.K., "An optimal Q-guidance scheme for satellite launch vehicles", Journal of Guidance, Control and Dynamics, Vol. 10, No. 1, Jan.-Feb. 1987.
- [44] Pohl, A., "Lenkgesetz für konventionelle Träger mittels Required-Velocity-Konzept", IFR_SR_96_002, Stuttgart, 1996.
- [45] Roenneke, A. J., and Markl, A. "Re-Entry Control to a Drag-vs-Energy Profile", Journal of Guidance, Control and Dynamics, Vol. 17, No. 5, Sep.-Oct. 1994, pp. 916-920.
- [46] Roenneke, A. J., and Well, K. H. "Nonlinear Flight Control for a High-Lift Reentry Vehicle," Proc. of the 1995 AIAA Guidance, Navigation, and Control Conf. (Baltimore, Md.), AIAA 95-3370-CP, Aug. 1995, pp. 1798-1805.
- [47] Roenneke, A. J., Schütz, K., and Well, K. H. "Trajectory Optimization Using 6-DOF Vehicle Models," 1995 IFAC Workshop on Control Applications of Optimization (Haifa, Israel), Dec. 1995.
- [48] Chobotov, V. A. Orbit Mechanics, American Institute of Aeronautics and Astronautics, Washington DC, 1991.
- [49] Bate, R., Fundamentals of Astrodynamics, Dover Publications, New York, 1971
- [50] Keppeler, J., "Calculation of Perturbed Keplerian Orbits with Non-Singular Orbit Elements", IFR_SR_00_004, Stuttgart, May 2000
- [51] Betts, J.T., "Optimal Interplanetary Orbit Transfer", Boeing Computer Services, BCSTECH-94-003, 1994
- [52] Bryson, A.E. Jr., and Ho, Y.-C. "Applied Optimal Control", 2nd, revised edition. Hemisphere Publishing Corp., Washington D.C., 1975.
- [53] Roenneke, A.J., "Ein bordautonomes Verfahren zur Flugführung und Regelung von Rückkehrfahrzeugen", Dissertation, Institute of Flight Mechanics and Control, University of Stuttgart, 2001 (in German)
- [54] Roenneke, A.J., "Adaptive On-Board Guidance for Entry Vehicles", AIAA 2001-4048, AIAA Guidance, Navigation, and Control Conference, Montréal, Québec, Canada, 6-9 August 2001
- [55] Betts, J.T., "SOS User's Guide - Release 2012.08", published by AMA, Issaquah (US) 2012.

REFERENCES

- [56] Gath, P.F., "Design and Development of a Hybrid Trajectory Optimization Software", Dissertation, Institute of Flight Mechanics and Control, University of Stuttgart, Germany, to be published 2002
- [57] Brown, K.R., Harrold, E.F., Johnson, G.W., "Rapid Optimization of Multiple-Burn Rocket Flights", IBM/NASA, NASA Contractor Report CR-1430, Marshall Space Flight Center, AL, September 1969
- [58] Thorne, J.D., Hall, C.D., "Approximate Initial Lagrange Costates for Continuous Thrust Spacecraft", *Journal of Guidance, Control, and Dynamics*, Vol. 19, No. 2, March-April 1996, pp. 283-288
- [59] Tauber, M.E., Sutton, K., "Stagnation-Point Radiative Heating Relations for Earth and Mars Entries", *Journal of Spacecrafts and Rockets JSR*, Vol. 28, No. 1, 1991, pp. 40-42
- [60] Cequel, Chemical Equilibrium in Excel, Software and Engineering Associates, Inc, <http://www.seainc.com/cequel.d.html>
- [61] CEA, Chemical Equilibrium with Applications, NASA Glenn Research Center, <http://www.grc.nasa.gov/WWW/CEAWeb/>
- [62] Montenbruck, O., Gill, E., *Satellite orbits: models, methods and applications*, Springer-Verlag, 2001
- [63] Pastena, M., *Formulations for Aerothermodynamic quantities from CIRA (Centro Italiano Ricerche Aerospaziali)*, Private Communication, 2002.
- [64] Consultative Committee for Space Data Systems, *Recommendation for Space Data System Standards, Time Code Formats*, CCSDS 301.0-B-3, Blue Book, Issue 3, CCSDS, January 2002.
- [65] Consultative Committee for Space Data Systems, *Navigation Data - Definitions and Conventions*, CCSDS 500.0-G-2, Green Book, Issue 2, CCSDS, November 2005.
- [66] International Astronomical Union, *SOFA Time Scale and Calendar Tools*, available online (Mar 2015) at http://www.iausofa.org/sofa_ts_c.pdf
- [67] Vallado, David A., *Fundamentals of Astrodynamics and Applications*, 3rd Edition, Space Technology Library, Springer, New York, 2007.
- [68] Rosser, J., Barkley, *Mathematical Theory of Rocket Flight*, McGraw-Hill Book Company, New York and London, 1947.
- [69] Martin, J., John, *Atmospheric Reentry*, Prentice Hall, Englewood Cliffs, New Jersey, 1966.

REFERENCES

- [70] Akin, David L., *Principles of Space Systems Design*, Mass Estimating Relationships, Course Material, University of Maryland, 2004.
- [71] Zandbergen, B. T. C., *Data for liquid rocket engine size estimation*, retrieved 2009 10 07, from <http://www.lr.tudelft.nl>
- [72] Bowman, B., *Jacchia-Bowman Thermospheric Density Model*, retrieved 2009 10 01, from <http://sol.spacenvironment.net/~JB2006/introduction.html>
- [73] Hoots, F.R. ; Roehrich, R.L., *SPACETRACK REPORT NO. 3, Models for Propagation of NORAD Element Sets*, December 1980
- [74] Detra, R. W., Kemp, N. H., and Riddell, F. R., "Addendum to Heat Transfer to Satellite Vehicles Reentering the Atmosphere", *Jet Propulsion*, Vol. 27, No. 12, pp. 1256-1257, Dec. 1957
- [75] Mattioli, E., *Aerodinamica*, III edition, Levrotto & Bella, Torino, 1994.
- [76] Seidelmann, P. K., Abalakin, V. K., Bursa, M., Davies, M. E., de Bergh, C., Lieske, J. H., Oberst, J., Simon, J. L., Standish, E. M., Stooke, P. and Thomas, P. C., *Report of the IAU/IAG Working Group on Cartographic Coordinates and Rotational Elements of the Planets and Satellites: 2000*, *Celestial Mechanics and Dynamical Astronomy*, Vol. 82, No. 1, January 2002.
- [77] Truckenbrodt, E., *Fluidmechanik*, Band 2: Elementare Strömungsvorgänge dichte- und gasförmiger Fluide sowie Potential- und Grenzschichtströmungen, Springer, Berlin 1980.
- [78] Inalhan, G., Tillerson, M. and How, J. P., "Relative Dynamics and Control of Spacecraft Formations in Eccentric Orbits", *Journal of Guidance, Control and Dynamics*, Vol. 25, No. 1, January-February 2002
- [79] Mars Climate Database official web site - <http://www-mars.lmd.jussieu.fr/>
- [80] Justus C.G. et al., *The NASA/MSFC Global Reference Atmospheric Model - 1995 Version (GRAM-95)*, NASA Technical Memorandum 4715.
- [81] Justus C.G., Johnson D.K., *The NASA/MSFC Global Reference Atmospheric Model - 1999 Version (GRAM-99)*, NASA/TM-1999-209630.
- [82] Gallais, P., *Atmospheric Re-Entry Vehicle Mechanics*, Springer, Berlin Heidelberg 2007.
- [83] Kramer, H.J.; Craubner, A.; Ziegltrum, W. *ROSI Rocket Simulation*, DFVLR TN 12/76. 1976

REFERENCES

- [84] Pollard J.E. *Simplified Analysis of Low-Thrust Orbital Maneuvers*, Aerospace Report No. TR-2000(8565)-10, 2000.
- [85] Astos Solutions, *ASTOS-Internal-TN001-Sensors and Link Budget*, January 2014.
- [86] International Telecommunication Union Recommendations, *The Radio Refractive Index: its Formula and Refractivity Data*, ITU-R P.453-8, February 2012.
- [87] International Telecommunication Union Recommendations, *Propagation Data and Prediction Methods Required for the Design of Earth-Space Telecommunication Systems*, ITU-R P.618-10, October 2009.
- [88] International Telecommunication Union Recommendations, *Attenuation by Atmospheric Gases*, ITU-R P.676-5, February 2012.
- [89] International Telecommunication Union Recommendations, *Water Vapour: Surface Density and Total Columnar Content*, ITU-R P.836-4, September 2010.
- [90] International Telecommunication Union Recommendations, *Characteristics of Precipitation for Propagation Modelling*, ITU-R P.837-6, February 2012.
- [91] International Telecommunication Union Recommendations, *Attenuation due to Clouds and Fog*, ITU-R P.840-5, February 2012.
- [92] Silver J.P., *Satellite Communications Tutorial*, RF, RFIC & Microwave Theory, Design, 2003.
- [93] Blake L.V., *Antenna and Receiving-System Noise-Temperature Calculation*, US Naval Research Laboratory, 1961.
- [94] *Antenna Introduction/Basics*, <http://www.phys.hawaii.edu/~anita/new/papers/militaryHandbook/antennas.pdf>
- [95] *Antenna Basics Concepts*, Department of Electrical and Computer Engineering at the Durham Campus of UNH, Course material, 2012.
- [96] Balanis C. A., *Antenna Theory Analysis and Design*, 3rd Edition, 2005.
- [97] *Helical Antenna*, <http://www.antenna-theory.com/antennas/travelling/helix.php>
- [98] Navstar, *Global Positioning Systems Directorate Systems Engineering & Integration Interface Specification*, September 2012.
- [99] European Union, *European GNSS (Galileo) Open Service Signal in Space Interface Control Document*, September 2010.

REFERENCES

- [100] Russian Institute of Space Device Engineering, *GLONASS Interface Control Document* Navigational Radiosignal in bands L1, L2, Edition 5.1, Moscow 2008.
- [101] Samama N., *Global Positioning Technologies and Performance* Wiley, 2008.
- [102] Kronman J.D., *Experience Using GPS For Orbit Determination of a Geosynchronous Satellite*.
- [103] Kuehl C.T.F., Filippi H., Barrios-Montalvo A., Krauss P.A., Heim J., Mueller C., Gottzein E., Dauphin H., *Potential of Multi Frequency, Multi Constellation Receivers for Spacecraft Navigation in GEO and Higher Altiudes*, GNC 2011.
- [104] International Telecommunication Union Recommendations, *Considerations for Sharing between Systems of Other Servicing Operating in Bands Allocated to the Radionavigation-satellite and Aeronautical Radionavigation Services and GNSS (GLONASS-M)*, ITU-R M.1317, 1997.
- [105] Kaplan E. D., Hegarty C. J., *Understanding GPS Priciples and Applications*, Artech House, Second Edition, 2005.
- [106] Misra R., Palod S., *Code and Carrier Tracking Loops for GPS C/A Code*, IJPAST, ISSN 2229-6107, 2011.
- [107] Larson W.J., Wertz J.R., *Space Mission Analysis and Design*, Kuwer Academic, Third Edition, 2003.
- [108] *NORAD Two-Line Element Set Format*, <http://www.celestrak.com/NORAD/documentation/tle-fmt.asp>
- [109] Dosis F., *Error Source*, GSA GENIUS Tutorial, Prague 2013.
- [110] *Rocket Propulsion Analysis Software*, <http://propulsion-analysis.com/>
- [111] Ponomarenko A., *Assessment of Delivered Performance of Thrust Chamber*, March 2013
- [112] Ponomarenko A., *RPA User Manual*, 2013
- [113] Federal Aviation Administration, *Licensing and Safety Requirements for Operation of a Launch Site; Rule*, 14 CFR Parts 401, 417, and 420, Vol. 65, No. 203, 19-10-2000.
- [114] Fehse, W., *Automated Rendezvous and Docking of Spacecraft*, Cambridge University Press, 2003.
- [115] Garcia P., Riera J.M., Benarroch A., *Propagation Impairment Mitigation for Millimetre wave Radio Systems: Statistics of dry and wet scintillation in Madrid*

REFERENCES

- using Intelsat 50 GHz beacon*, PM3-013 1st International workshop, COST Action 280, 2002.
- [116] Gorski K.M., Hivon E., Banday A.J., Wandelt B.D., Hansen F.K., Reinecke M., Bartelman M., *HEALPix a Framework for High Resolution Discretization, and Fast Analysis of Data Distributed on the Sphere*, 2004.
- [117] Wertz J.R., *Mission Geometry; Orbit and Constellation Design and Management*, Microcosm Press, 2002.
- [118] International Telecommunication Union, *Definitions and terminology for synchronization networks*, ITU-T G.810, August 1996.
- [119] Gallais, P., *Atmospheric Re-Entry Vehicle Mechanics*, Springer, Berlin Heidelberg 2007.
- [120] Abramowitz, M., Stegun, I. A., *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*, Dover Publications, New York, 1972.
- [121] Krisko P.H., Johnson N.L., Opiela J.N, *EVOLVE 4.0 Orbital Debris Mitigation Studies*, Adv. Space Res. Vol. 28, No. 9, pp. 1385-1390, 2001
- [122] Klinkrad, H. Fritsche, B. & Lips, T. *A Standardized Method for Re-Entry Risk Evaluation*, IAC-04-IAA.5.12.2.07, Vancouver, 2004.
- [123] GPWv3 Population Density Data; 2003-2005 Center for International Earth Science Information Network (CIESIN), Columbia University.
- [124] Klinkrad H., *Space Debris: Models and Risk Analysis*, Springer-Praxis Books, ISBN 3-540-25448-X, Springer-Verlag.
- [125] World Meteorological Organization, The Voluntary Observing Ship (VOS) scheme, Feb. 2011;
<http://www.wmo.int/pages/prog/amp/mmop/JCOMM/OPA/SOT/vos.html>
- [126] Flight Safety Code (2nd Edition), Space Licensing and Safety Office, Department of Industry, Tourism and Resources Australia, 1 July 2002
- [127] Dallas S.S., *Equations of Motion for Rotating Finite Bodies in the Extended PPN Formalism*, Celestial Mechanics, Volume 15, pp. 111-123, 1977
- [128] Will C.M., Nordtvedt K., *Conservation Laws and Preferred Frames in Relativistic Gravity I Preferred-Frame Theories and an Extended PPN Formalism*, The Astrophysical Journal, Volume 177, pp. 757-774, 1972.
- [129] Wertz J.R., *Space Attitude Determination and Control*, Kuwer Academic, 1990.

REFERENCES

- [130] Bruevich E.A., Bruevich V.V., Yakunina G.V., *Changed Relation between Solar 10.7-cm Radio Flux and some Activity Indices which describe the Radiation at Different Altitudes of Atmosphere during Cycles 21-23*, 2013.
- [131] Wiesel W. E., *Spaceflight Dynamics*, McGraw-Hill, New York, 1989.
- [132] Howell K.C., *Three Dimensional Periodic Halo Orbits in the Restricted Three-Body Problem*, Ph.D. dissertation, Stanford University, 1983.
- [133] Richardson D. L., *Analytical Construction of Periodic Orbits about the Collinear Points*, University of Cincinnati, 1979.
- [134] Tremblay O., Dessant L.-A., *Experimental Validation of a Battery Dynamic Model for EV Applications*, Electrical Engineering Department, École de Technologie Supérieure, Montréal, Canada, 2009.
- [135] David L. King, Jay A. Kratochvil, and William E. Boyson, *Temperature Coefficients for PV Modules and Arrays: Measurement Methods, Difficulties, and Results*, Photovoltaic Systems Department, Sandia National Laboratories, Albuquerque, September 1997
- [136] McCarthy D.D., Petit G., *IERS Conventions (2003)*, IERS Conventions Centre, Technical Note No. 32, 2004.
- [137] Archambeau G., Augros P., Trelat E., *Eight Lissajous Orbits in the Earth-Moon System*, Mathematics in Action, 2011.
- [138] International Geomagnetic Reference Field, <http://www.ngdc.noaa.gov/IAGA/vmod/igrf.html>
- [139] Department of Defence Interface Standard MIL-STD-1797A, "Flying Qualities Of Piloted Aircraft", pp. 678 - 685
- [140] Manmoha Dass Goel et al, An Abridged Review of Blast Wave Parameters, Defence Science Journal, Vol. 62 Number 5, pp. 300-306, 2012.
- [141] Neff M., A Visual Model for Blast Waves and Fracture, Master Thesis, University of Toronto, 1998.
- [142] Lees F.P., Loss Prevention in the Process Industries, Vol. 1, London and Boston: Butterworths, 1980.
- [143] Ronald Humble, *Space Propulsion Analysis and Design*
- [144] Goddard Earth Sciences Data and Information Services Center (GES DISC), *Total Ozone Mapping Spectrometer (TOMS) project*. https://disc.gsfc.nasa.gov/acdisc/TOMS/tomsl3_dataset.gd.html

REFERENCES

- [145] ESA mission analysis guidelines for EO missions (DOPS-GS-RM-1002-OPS-OSA)
- [146] The SPICE Toolkit, NASA JPL - <https://naif.jpl.nasa.gov/naif/toolkit.html> - last access: 2017-07-25
- [147] Index of /pub/naif/generic_kernels/spk/planets - https://naif.jpl.nasa.gov/pub/naif/generic_kernels/spk/planets/ - last access: 2017-07-25
- [148] David A. Vallado, Paul Crawford, Richard Hujak and T. S. Kelso, *Revisiting Spacetrack Report #3: Rev 2*, AIAA 2006-6753-Rev2
- [149] Bowman, B., *A New Empirical Thermospheric Density Model JB2008 Using New Solar and Geomagnetic Indices*, retrieved 2017 10 20, from <http://sol.spacenvironment.net/jb2008/index.html>
- [150] *Proceedings of the 15th Inter-Agency Space Debris Coordination Committee Meeting*, NASA JSC, Houston, USA, 1997
- [151] Richardson, D. L., *Analytic construction of periodic orbits about the collinear points*, *Celestial Mechanics*, vol. 22, Oct. 1980, p. 241-253.
- [152] Howell, K. C., *Three-Dimensional Periodic Halo Orbits*, *Celestial Mechanics*, Vol. 32, Issue 1, p. 53, 1984
- [153] Werner, R. A., Scheeres, D. J., *Exterior Gravitation of a Polyhedron Derived and Compared with Harmonic and Mascon Gravitation Representations of Asteroid 4769 Castalia*, *Celestial Mechanics and Dynamical Astronomy*, Vol. 65, p. 313-344, 1997
- [154] Dodge, F., *The New Dynamic Behavior of Liquids in Moving Containers*, Southwest Research Institute, 2000
- [155] Abramson, N., *The Dynamic behaviour of Liquids in Moving Containers*, NASA report N67-15884, 1966
- [156] International Ammunition Technical Guideline 01.80, Second edition, 2015-02-01, United Nations Office for Disarmament Affairs - <https://unoda-web.s3.amazonaws.com/wp-content/uploads/assets/convarms/Ammunition/IATG/docs/IATG01.80.pdf#page=15> - last access: 2020-05-14
- [157] J. Stockie, *The Mathematics of Atmospheric Dispersion Modelling* *SIAM Review* Vol. 53, No. 2, pp. 349-372 (2011), <https://doi.org/10.1137/10080991X>

