EMTG User Guide

Timothy Sullivan [1], Adam Trask [2], Kyle Hughes[3], Alec Mudek[4], Edwin Dove[5]

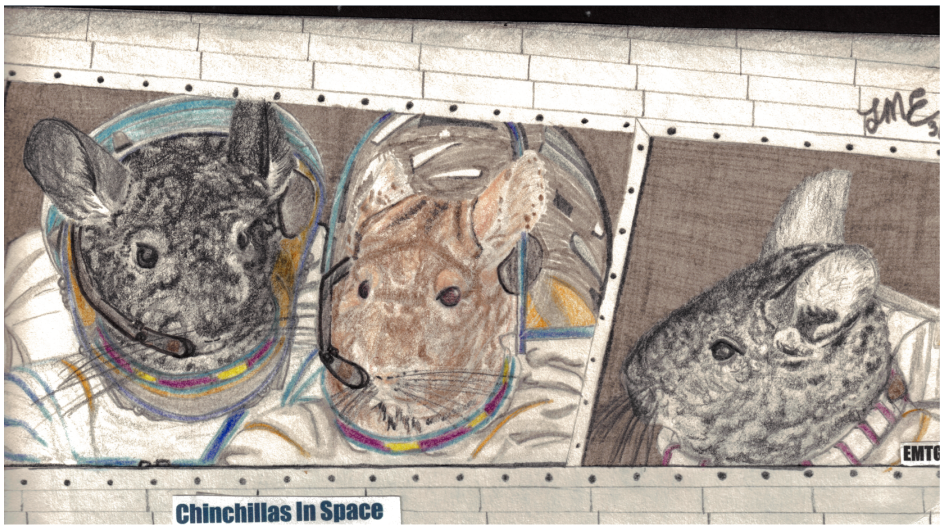
[1]Senior Member of the Technical Staff, The Aerospace Corporation, Trajectory Design and Optimization Department

[2]Member of the Technical Staff, The Aerospace Corporation, Trajectory Design and Optimization Department

[3]Aerospace Engineer, NASA Goddard Space Flight Center, Navigation and Mission Design Branch Code 595

[4]Aerospace Engineer, NASA Goddard Space Flight Center, Navigation and Mission Design Branch Code 595

[5]Aerospace Engineer, NASA Goddard Space Flight Center, Navigation and Mission Design Branch Code 595



Revision Date	Author	Description of Change
April 1, 2024	Tim Sullivan and Adam Trask	Initial Document Release

Contents

1 Introduction **1**

2 EMTG Force Models and Propagation **2**
2.1 Force Modeling . . . 2
2.1.1 Solar Radiation Pressure (SRP) . . . 2
2.1.2 Third Body Gravitational Forces . . . 3
2.1.3 Central Body Spherical Harmonics . . . 3
2.1.4 Atmospheric Drag . . . 3
2.2 Propagation . . . 4
2.2.1 Kepler . . . 4
2.2.2 Integrator . . . 4

3 EMTG Mission Types **5**
3.1 Multiple Gravity Assist with n Deep Space Maneuvers (MGAnDSMs) . . . 5
3.2 Multiple Gravity Assist with Low-Thrust (MGALT) . . . 6
3.3 Finite Burn Low Thrust (FBLT) . . . 7
3.4 Coast Phase . . . 8
3.5 Parallel Shooting Finite Burn (PSFB) . . . 8
3.6 Parallel Shooting Bounded Impulse (PSBI) . . . 9
3.7 Probe Entry Phase . . . 9

4 Journey Boundaries **12**
4.1 Boundary Classes . . . 13
4.1.1 Ephemeris-pegged . . . 13
4.1.2 Ephemeris-referenced . . . 13
4.1.3 Free Point . . . 14
4.1.4 Periapse . . . 14
4.2 Departure Types . . . 14
4.2.1 Launch or Direct Insertion . . . 14
4.2.2 Depart from Parking Orbit . . . 14
4.2.3 Free Direct Departure . . . 15
4.2.4 Flyby . . . 15
4.2.5 Flyby with Fixed V-Infinity Out . . . 15
4.2.6 Zero Turn Flyby . . . 15
4.2.7 Spiral-out from Circular Orbit . . . 15
4.3 Arrival Types . . . 16
4.3.1 Insertion into Parking Orbit (use chemical Isp) . . . 16
4.3.2 Rendezvous (with chemical maneuver) . . . 16

4.3.3 Intercept with Bounded V-Infinity . . . 16
4.3.4 Rendezvous (no maneuver) . . . 16
4.3.5 Match Final V-Infinity Vector (with chemical maneuver) . . . 17
4.3.6 Match Final V-Infinity Vector (no maneuver) . . . 17
4.3.7 Capture Spiral . . . 17
4.3.8 Momentum Transfer . . . 17

5 Universe File and Options **18**
5.1 Central Body Definition . . . 18
5.2 Menu of Bodies Details . . . 19

6 EMTG Options **23**
6.1 Global Mission Options and Constraints . . . 24
6.1.1 Global Mission Options . . . 24
6.1.2 Global Mission Constraints . . . 26
6.2 Spacecraft Options . . . 28
6.2.1 Common Hardware Options . . . 29
6.2.2 Spacecraft Model Type: 0 . . . 30
6.2.3 Spacecraft Model Type: 1 . . . 33
6.2.4 Spacecraft Model Type: 2 . . . 33
6.2.5 Additional Spacecraft Options . . . 34
6.3 Journey Options . . . 36
6.3.1 Overall Journey Options . . . 38
6.3.2 Journey Departure Options . . . 43
6.3.3 Journey Arrival Options . . . 44
6.3.4 Perturbations and Other Options . . . 46
6.3.5 Departure and Arrival Elements . . . 48
6.3.6 Journey Staging Options . . . 51
6.4 Solver Options . . . 51
6.4.1 Gradient-Based Solver . . . 53
6.4.2 Monotonic Basin Hopping . . . 57
6.5 Physics Options . . . 61
6.5.1 Ephemeris Settings . . . 62
6.5.2 Spiral Settings . . . 63
6.5.3 Integrator Settings . . . 64
6.5.4 State Representation Settings . . . 64
6.5.5 Perturbation Settings . . . 66
6.6 Output Options . . . 68
6.6.1 General Output Options . . . 68
6.6.2 Ephemeris Output Options . . . 72

7 Configuration Files **77**
7.1 Launch Vehicle Configuration . . . 77
7.2 Spacecraft Configuration . . . 78
7.2.1 Spacecraft Block . . . 79
7.2.2 Stage Block General Data . . . 81
7.2.3 Power System . . . 84
7.2.4 Propulsion System . . . 86

8 Conversion Scripts **95**
8.1 Convert To Single Phase Journeys 95
8.2 High Fidelity Conversion 96

9 Other Resources **97**
9.1 Constraint Scripting 97
9.2 Python EMTG Automated Trade Study Application 97

List of Figures

1.1 The EMTG logo featuring "Clementine", Jacob Englander's pet chinchilla 1

2.1 Example `.emtg_densityopt` File Structure 3

3.1 Multiple Gravity Assist with n Deep-Space Maneuvers using shooting (MGAnDSMs) 6
3.2 Multiple Gravity Assist with Low-Thrust (MGALT) 7
3.3 Finite-Burn Low-Thrust (FBLT) 8
3.4 Parallel Shooting with Finite-Burn (PSFB) 9
3.5 Parallel Shooting with Bounded Impulses (PSBI) 9
3.6 Probe Entry Phase 11

6.1 Creating a new EMTG Options file 23
6.2 EMTG Options tabs 24
6.3 EMTG Global Options tab 24
6.4 EMTG Journey Options tab 37
6.5 Journey Manager 38
6.6 Journey Time Steps 39
6.7 Journey Mass Increment Options 41
6.8 Journey bounded arrival date 43
6.9 EMTG Journey Arrival and Depature Elements 49
6.10 EMTG Solver Options tab 52
6.11 Monotonic Basin Hopping on a 1-dimensional function. 57
6.12 EMTG Physics Options tab 61
6.13 EMTG Physics Options - Ephemeris Settings 62
6.14 EMTG Physics Options - State Representation Settings 65
6.15 EMTG Physics Options - Perturbation Settings 66
6.16 PyEMTG Output Options 68
6.17 Journey Wait Entries 70
6.18 Forward-integrated Ephemeris Final DV Issue 73
6.19 Example Extra Journey Configuration 73

7.1 Example Launch Vehicle Options File 78
7.2 Example Structure Of `.emtg_spacecraftopt` Files 79
7.3 Example Spacecraft Block Global Data 81
7.4 Throttle Sharpness 83
7.5 Example Spacecraft Stage Block Data 84
7.6 Example Spacecraft Power Library Block 86
7.7 Example Spacecraft Propulsion Library Block 92
7.8 Example Throttle Table File 94

List of Tables

3.1 Mission Type, Propagation, and Perturbation Compatibility 5

4.1 Boundary Class and Type Compatibility 13

5.1 Universe File Central Body Information 18
5.2 Universe File Menu Of Bodies Specifications 20

7.1 Launch Vehicle Variables 78
7.2 Spacecraft Block Global Data 80
7.3 Stage Block Data 81
7.4 Power System Variables 85
7.5 Propulsion System Variables 87
7.6 Throttle Table Tabularized Data 93
7.7 Throttle Table Line Data 93

List of Known Issues

1 Issue with forward-integrated ephemeris. 73
2 Launch Vehicle configuration file issue with payload adapter mass for some scenarios. . . 78
3 High fidelity conversion scripts require universe files of precise names. 96

List of Acronyms

DLA Declination of Launch Asymptote

EMTG Evolutionary Mission Trajectory Generator

GUI Graphical User Interface

LEO Low Earth Orbit

SNOPT Sparse Nonlinear OPTimizer

SOI Sphere of Influence

SRP solar radiation pressure

COE Classical Oribital Elements

ICRF International Celestial Reference Frame

SPICE Spacecraft Planet Instrument Camera-matrix Events

STK Systems Tool Kit

NLP Nonlinear Program

MBH Monotonic Basin Hopping

MGALT Multiple Gravity Assist with Low-Thrust

MGAnDSMs Multiple Gravity Assist with n Deep-Space Maneuvers using shooting

PSFB Parallel Shooting with Finite-Burn

PSBI Parallel Shooting with Bounded Impulses

FBLT Finite-Burn Low-Thrust

SMA semi-major axis

ECC eccentricity

INC inclination

RAAN right ascension of the ascending node

MA mean anomaly

AOP argument of periapsis

Chapter 1

Introduction

The Evolutionary Mission Trajectory Generator (EMTG) is a tool primarily designed for optimizing interplanetary trajectories. It can quickly and robustly find optimal trajectories involving multiple flybys using either low-thrust or high-thrust maneuvers, with significant flexibility in defining spacecraft for these maneuvers. EMTG is extremely efficient at solving a wide range of problems but is not completely generic. It is well-suited for interplanetary trajectory design and allows modeling from low to relatively high-fidelity.

EMTG operates with minimal user intervention and is capable of running without a user provided initial guess. This is achieved through the usage of a stochastic global search algorithm called Monotonic Basin Hopping (MBH). MBH randomly searches the solution space and then performs a local gradient-based search using the Sparse Nonlinear OPTimizer (SNOPT) to obtain a local minimum. Each iteration of MBH perturbs the resulting decision vector from SNOPT and performs new local gradient-based searches, which helps to escape local minima and work towards a global optima. This process repeats until an ending condition of time or number of iterations is met.

EMTG is built in C++ and is compatible with Windows and Linux computers. A Graphical User Interface (GUI) allows convenient interface with the tool. Custom Python scripts also allow interfacing with EMTG in a scripting environment. This user guide covers the general usage of the tool for and is based on EMTG v9.02.

Figure 1.1: The EMTG logo featuring "Clementine", Jacob Englander's pet chinchilla

Chapter 2

EMTG Force Models and Propagation

This chapter describes how EMTG models gravitational and other non-maneuver forces on the spacecraft as well as spacecraft state propagation. Users should keep in mind that not all perturbing forces will apply to the spacecraft depending on the combination of Mission Type and Propagator selected for the current Journey. Refer to Table 3.1 to check if perturbations are compatible with a chosen Mission Type and Propagator.

2.1 Force Modeling

By default, EMTG only models forces due to the gravitational effects of the central body and from whatever maneuvers are applied to the spacecraft. However, additional forces can be included when higher fidelity modeling is desired. These forces are known as Perturbations in EMTG. These perturbations can accumulate over time and cause significant changes to a spacecraft's trajectory if left uncorrected. EMTG can model perturbing forces on the spacecraft caused by central body gravitational harmonics, third body gravitational effects, atmospheric drag, and solar radiation pressure.

2.1.1 Solar Radiation Pressure (SRP)

solar radiation pressure (SRP) is a force that affects spacecraft in space due to the radiation emitted by the Sun. This force can cause small but measurable changes in a spacecraft's trajectory over time. EMTG uses a spherical/cannonball solar radiation pressure model. Users must specify the spacecraft coefficient of reflectivity C_r, the surface area A_s, the illumination percentage K, solar irradiance Φ at 1 AU (in W/m^2), and the speed of light in a vacuum c. These parameters are specified on the Physics Options tab in PyEMTG.

$$\ddot{\mathbf{r}} = C_{\text{SRP}} \frac{1}{m \; r_{s/\odot}^2} \frac{\mathbf{r}_{s/\odot}}{r_{s/\odot}} \tag{2.1}$$

$$C_{\text{SRP}} = \frac{C_r A_s K \phi}{c} \tag{2.2}$$

2.1.2 Third Body Gravitational Forces

EMTG can model third body gravitational forces with the perturbing bodies configured at the Journey level using the "Perturbation bodies" field in the PyEMTG Journey Options tab (see Section 6.3.4), which is revealed when the user selects "Enable third body" on the Physics Options tab (see Section 6.5).

2.1.3 Central Body Spherical Harmonics

Central body spherical harmonics may be enabled using the Physics Options setting "Enable central-body J2", which applies J2-perturbations to all Journeys where the central body has a defined J2 coefficient in its Universe file (see Section: 6.5). Additionally, higher order perturbations can be included at the Journey-level by selecting the "Enable central-body gravity harmonics" on the Journey Options tab (see Section 6.3.4). When selecting this option, a `.grv` file must be provided. This file follows the standard format required by the Ansys Systems Tool Kit (STK).

2.1.4 Atmospheric Drag

EMTG is also capable of modeling atmospheric drag given a file containing various information about the atmosphere model. This can be included at the Journey-level by selecting the "Enable aerodynamic drag" option in the Journey Options tab (see Section 6.3.4). When selecting this option, an `.emtg_densityopt` file must be provided. This file describes the altitude versus density information fit to the chosen atmospheric density model. Various tools may be used to generate the data required for `.emtg_densityopt` files, as atmospheric data varies based on position on the planet and time of day. Figure 2.1 provides an example of the expected format. Note that the first line must be a negative altitude (typically near the center of the body) to ensure validity of the fitted model.

#altitude (km)	density (kg/m^3)
-6378	1.225
0.0	1.225
1.0	1.112
2.0	1.007
3.0	0.9093
4.0	0.8194
5.0	0.7364
...	...

Figure 2.1: Example `.emtg_densityopt` File Structure

2.2 Propagation

EMTG has two main methods for performing propagation of the spacecraft: Kepler (or analytic) and Integrator. Kepler propagation is much faster than Integrator propagation at the cost of accuracy and is not always compatible with certain perturbations depending on Mission Type (see Chapter 3). Kepler propagation is more appropriate at earlier stages of mission design.

2.2.1 Kepler

Kepler uses Kepler's equation to analytically propagate the spacecraft's orbit state between delta-v events. The "Integrator time step size option" in the Physics Options tab of PyEMTG does not affect Kepler propagation since Kepler propagation is analytical. Kepler propagation is compatible with MGALT, Coast Phase, MGAnDSMs, PSBI, and Probe Entry Phase.

2.2.2 Integrator

EMTG's Integrator option propagates the spacecraft by integrating the full equations of motion,whether they are simple two-body equations or more complex N-body equations with additional perturbations. As a result, Integrator propagation is slower than Kepler propagation but can model more realistic forces. Integrator is compatible with FBLT, Coast Phase, and MGAnDSMs, PSFB, Probe Entry Phase, and Control Law Thrust Phase. Currently, the only functional "Integrator type" is "rk8 fixed step." Do not choose "rk7813M adaptive step."

Chapter 3

EMTG Mission Types

EMTG provides several transcription methods to model the flight of a spacecraft at various levels of accuracy. These methods are known in EMTG as Mission Types. EMTG provides Mission Types for chemical and low thrust spacecraft. Not all Mission Types are compatible with all of the propagation methods and perturbations discussed in Chapter 2. Table 3.1 describes which perturbation and propagation methods can be used with each Mission Type. Setting perturbations for an incompatible Mission Type and propagation method in PyEMTG or the EMTG options file will have no effect on the spacecraft force model and EMTG will not notify the user that pertubations are not being applied.

Mission Type	Propagation	Perturbation Modeling
MGAnDSMs	Integrator or Kepler	Only with Integrator
MGALT	Kepler	Yes
FBLT	Integrator	Yes
Coast Phase	Integrator or Kepler	Only with Integrator
PSFB	Integrator	Yes
PSBI	Kepler	Yes
Probe Entry Phase	Integrator	Yes

Table 3.1: Mission Type, Propagation, and Perturbation Compatibility

3.1 Multiple Gravity Assist with n Deep Space Maneuvers (MGAnDSMs)

Multiple Gravity Assist with n Deep-Space Maneuvers using shooting (MGAnDSMs) transcription models the flight of a spacecraft using high-thrust chemical propulsion. This is done using two-point shooting to propagate a trajectory, where for a single phase the spacecraft is propagated forward in time from a left-hand boundary condition and backward in time from a right-hand boundary condition with a set of match point constraints to link the forward and backward half-phases together. The maneuvers are encoded as impulsive events, i.e. they happen instantaneously, and the optimizer may place maneuvers in any permitted location in the phase. The user chooses the maximum number of impulses a priori. If the specified number of impulses is more than what

is needed, the optimizer will reduce the magnitude of any un-needed maneuvers to zero.

Selecting MGAnDSMs for one or more Journeys activates the variable "impulses per phase" in the Journey Options tab which sets the maximum number of impulsive burns that can occur during the Journey. Spacecraft propagation can be modeled with Kepler or Integrator between instantaneous delta-v events at each DSM. Perturbations can only be applied with Integrator propagation by adding the perturbing forces to the central body gravitational force.

EMTG will decrement the mass of the spacecraft according to the engine Isp specified in the Spacecraft Options section of PyEMTG (see Section 6.2), the EMTG Options file, or Spacecraft Configuration file (see Section 7.2). This is labeled as "Chemical Isp (s)" in PyEMTG and as "IspChem" in the EMTG Options file. In Figure 3.1, m_0 refers to the mass of the spacecraft at the start of the Journey, m_n to the mass after DSM n, and m_f to the mass at the end of the Journey where $m_N = m_f$ after the final DSM in the Journey. The red arrows indicate the impulsive velocity change where $\mathbf{v}_n^-$ is the spacecraft velocity prior to the impulsive delta-v and $\mathbf{v}_n^+$ is the velocity after the impulsive delta-v.

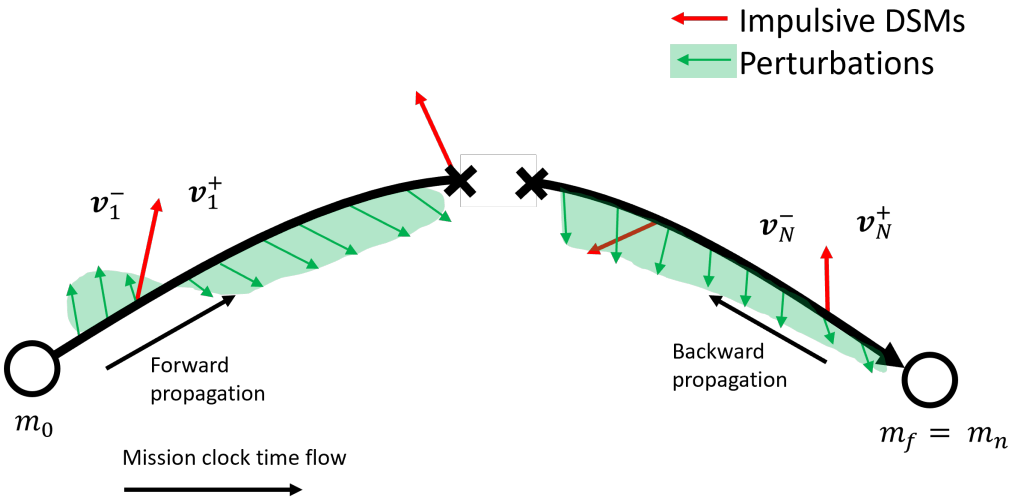


Figure 3.1: Multiple Gravity Assist with n Deep-Space Maneuvers using shooting (MGAnDSMs)

3.2 Multiple Gravity Assist with Low-Thrust (MGALT)

Multiple Gravity Assist with Low-Thrust (MGALT) uses two-point shooting to propagate a trajectory. Each phase is propagated forward in time from a left-hand boundary condition and backward in time from a right-hand boundary condition with a set of match point constraints to link the forward and backward half-phases together. MGALT then models each Journey using the Sims-Flanagan transcription and Kepler propagation. A bounded-impulse delta-v is applied at each segment to model the spacecraft's thrust; the magnitude of the impulse is limited to the maximum delta-v that could be achieved by thrusting constantly across the segment. If any perturbations are turned on, they are applied in a similar manner: the perturbing force(s) are evaluated at the point at which each propulsive delta-v is applied. The effect of the perturbing acceleration(s) across

the segment is approximated by multiplying the acceleration by the time between segments. The perturbing "delta-v" is then added to the propulsive delta-v to get the full impulse applied to the spacecraft. As a result, MGALT can approximate the effect of perturbing accelerations without using integrator propagation.

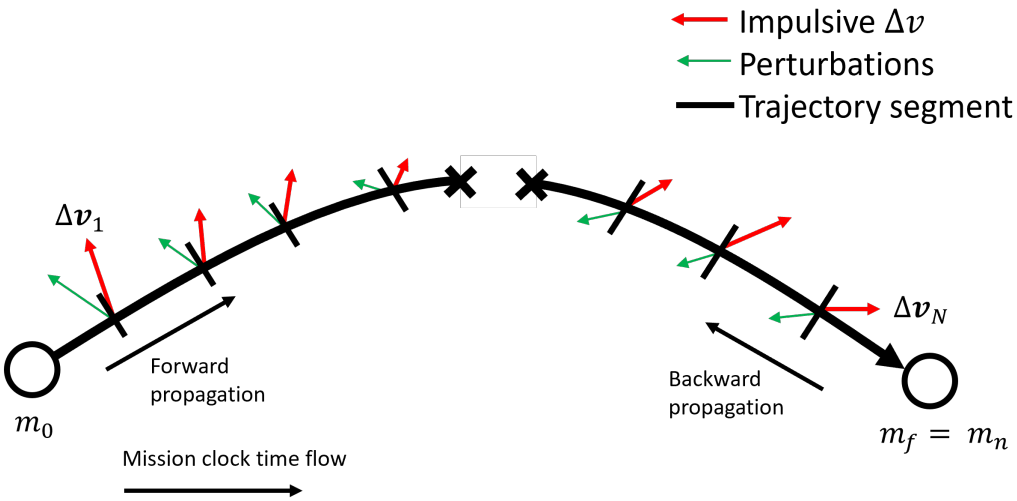


Figure 3.2: Multiple Gravity Assist with Low-Thrust (MGALT)

3.3 Finite Burn Low Thrust (FBLT)

Finite-Burn Low-Thrust (FBLT) is identical to MGALT except that numerical integration is used to propagate the equations of motion for the spacecraft rather than the Sims-Flanagan transcription. The central body, thrust, and perturbing forces (if used) sum to apply a net acceleration to the spacecraft, which is integrated with a Runge-Kutta method to generate the spacecraft trajectory. This gives a more realistic trajectory than MGALT.

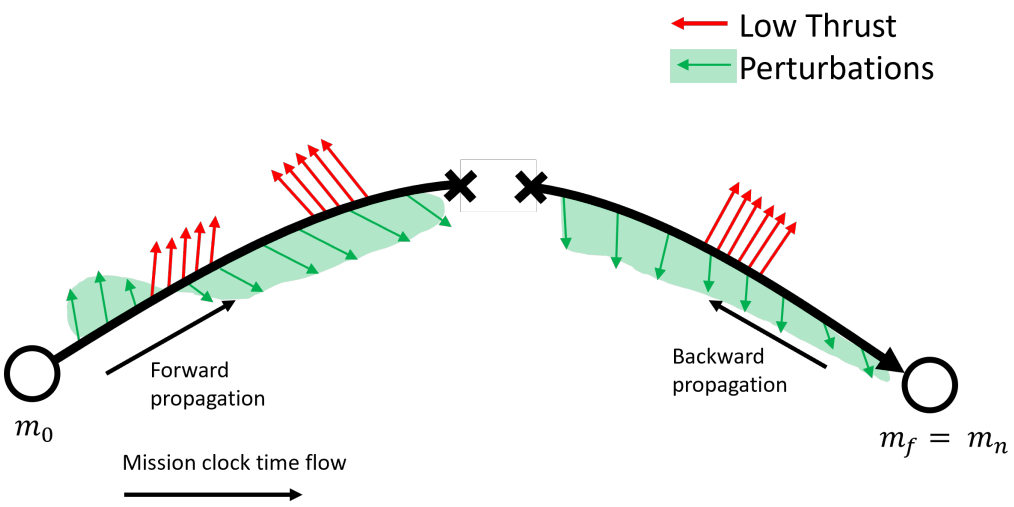


Figure 3.3: Finite-Burn Low-Thrust (FBLT)

3.4 Coast Phase

Like the name implies, Coast Phase has no spacecraft thrust. It can be propagated with either Kepler (and no perturbing forces) or Integrator with optional perturbing forces. For a Coast Phase, the "Integrator time step size" set on the "Physics Options" tab is always overridden by the forward/backward half-phase step size values on the "Journey Options" tab.

3.5 Parallel Shooting Finite Burn (PSFB)

PSFB is a phase transcription which models the path of a low-thrust spacecraft using direct parallel-shooting and a high-fidelity model of both the natural and spacecraft dynamics. PSFB integrates the full set of dynamics with an explicit Runge-Kutta method with all central body, thrust, and perturbing forces (if used) applied. Since PSFB is a parallel-shooting method and all steps propagate forward in time, each time-step encodes the left-hand state in the decision vector and includes a set of continuity constraints to enforce equivalency between the encoded left-hand state and the previous step's propagated right-hand state. This parallel-shooting method allows for easy implementation of maneuver constraints as compared to the two-point shooting used in FBLT.

The controls are piecewise constant across a segment. Users may choose to allow more than one control opportunities per segment when setting the Journey options. This option appears as "Number of interior control points" in the Journey Options tab. When more than one control opportunities are selected, the segment is broken into equal-length control steps. EMTG has a high-fidelity duty cycle variant of PSFB that is used when a realistic duty cycle option is selected. The duty cycle is discussed in more detail in Section 6.2.

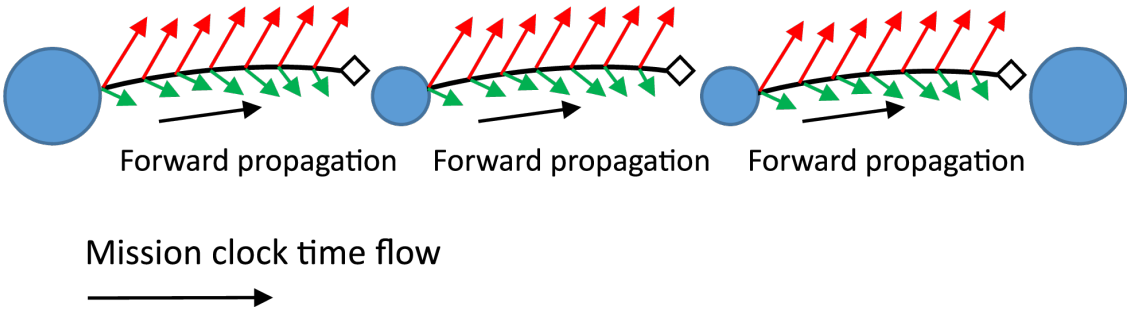


Figure 3.4: Parallel Shooting with Finite-Burn (PSFB)

3.6 Parallel Shooting Bounded Impulse (PSBI)

PSBI combines the Sims-Flanagan transcription with EMTG's base parallel-shooting phase classes, resulting in a low-fidelity parallel-shooting transcription. The low-thrust acceleration is modeled as a bounded impulse in the center of each time step, similar to the MGALT transcription. Since PSBI is a parallel-shooting method and all steps propagate forward in time, each time-step encodes the left-hand state in the decision vector and includes a set of continuity constraints to enforce equivalency between the encoded left-hand state and the previous step's propagated right-hand state. This parallel-shooting method allows for easy implementation of maneuver constraints as compared to the two-point shooting used in MGALT.

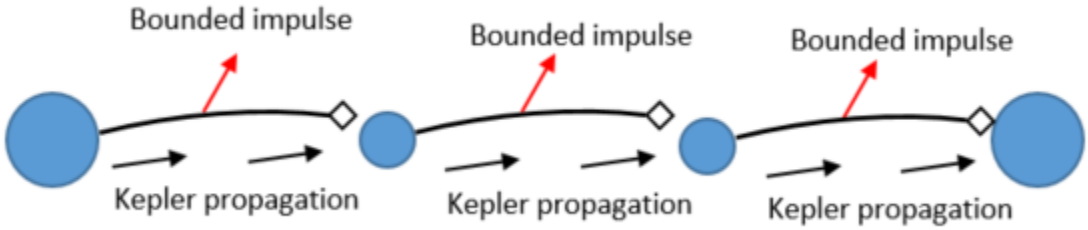


Figure 3.5: Parallel Shooting with Bounded Impulses (PSBI)

3.7 Probe Entry Phase

Probe Entry Phase is a special phase type that tracks both the trajectory of the spacecraft and that of an entry probe which separates from the spacecraft. Following separation, the probe and spacecraft are propagated separately but for the same time of flight. The entry probe propagates to some point defining atmospheric entry and then to some surface target or parachute open state. The spacecraft performs a divert maneuver to enter some desired final state, defined by the Journey arrival conditions.

This phase type derives from the MGAnDSMs base class and thus inherits the same departure

and arrival criteria for the main spacecraft. To handle the probe, an additional arrival event is introduced to define the desired target location for the probe independent of the rest of the problem. This right-hand boundary is modeled as a Free Point Intercept, thus when using the Probe Entry Phase mission type users must set some arrival elements using the same interface as any Free Point departure or arrival class. Separation is modeled using a user-defined constant separation impulse in Newton-seconds. This impulse is applied in the direction of the probe's velocity vector at entry.

Since Probe Entry Phase inherits from MGAnDSMs, this is done using a two-point shooting method with a set of match points linking the forward and backward half-phases. Since the probe does not perform maneuvers after separation, the match point location is specified as a fraction of the phase time of flight, which is a user-defined constant. This allows users to force the match point to be nearer to the right-hand boundary condition, allowing small time steps to be used when the probe is near the planet. This allows some representation of atmospheric entry. The second coast subphase following the match point then includes a unique time of flight decision variable to allow the optimizer to determine the arrival event.

Since the spacecraft is propagated alongside the probe, its propagation also inherits from MGAnDSMs and thus uses the expected two-point shooting method. The match point for the propagation of the spacecraft is attached to the divert maneuver and may be placed in any permitted location in the phase just as maneuvers are in an MGAnDSMs phase. Effectively, the spacecraft operates as if it is a normal MGAnDSMs phase, with an initial mass loss equal to the weight of the entry probe.

An additional constraint is enforced on the distance between the probe and the spacecraft to ensure the solution maintains a reasonably high-bandwidth communication between the entry probe and the carrier spacecraft. This constraint is applied on the state at the end of the probe's trajectory and the state at the end of the spacecraft's trajectory and is set by the user in the Journey Options via the "Probe communication distance bounds (km)" input.

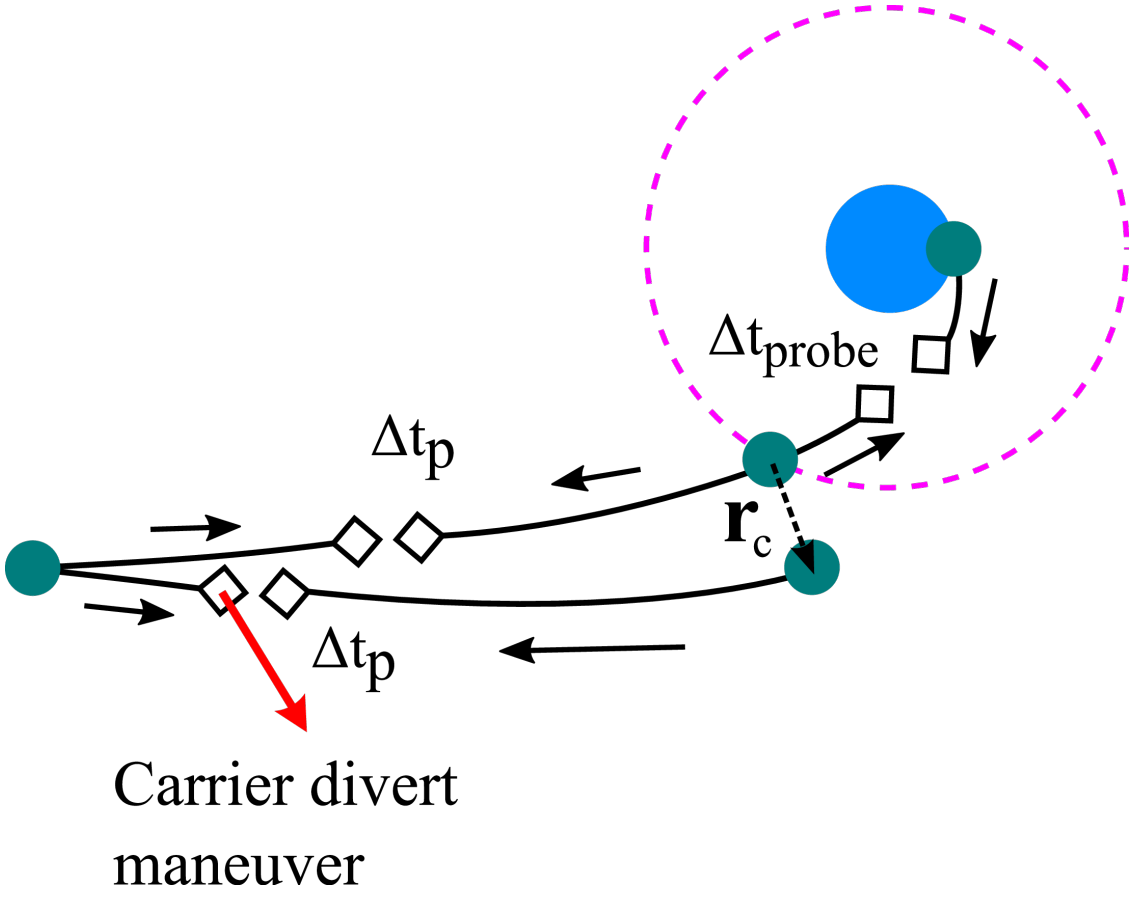


Figure 3.6: Probe Entry Phase

Chapter 4

Journey Boundaries

EMTG missions consist of at least one Journey, each having a Departure and Arrival Boundary. A Boundary includes a Type (Arrival or Departure) that determines how the spacecraft arrives at or departs from the ephemeris body, and a Class that represents the spacecraft state relative to the ephemeris body. The Journey Options tab specifies the specific combination of Boundary Class and Type for each Journey's Arrival and Departure. Table 4.1 specifies the compatibility of the various Boundary Classes and Types.

		Boundary Classes			
		Ephemeris-pegged	Free Point	Ephemeris-referenced	Periapse
Departure Types	Launch or Direct Insertion	**x**	**x**		**x**
	Depart from Parking Orbit	**x**			
	Free Direct Departure	**x**	**x**	**x**	
	Flyby	**x**			
	Flyby with Fixed V-Infinity out				
	Spiral-out from Circular Orbit	**x**			
	Zero Turn Flyby	**x**			
Arrival Types	Insertion into Parking Orbit (use chemical Isp)	**x**			
	Rendezvous (with chemical maneuver)	**x**	**x**		
	Intercept with Bounded V-Infinity	**x**	**x**	**x**	**x**
	Rendezvous (no maneuver)	**x**	**x**	**x**	
	Match Final V-Infinity Vector (with chemical maneuver)				
	Match Final V-Infinity Vector (no maneuver)				
	Capture Spiral	**x**			
	Momentum Transfer	**x**			

Table 4.1: Boundary Class and Type Compatibility

4.1 Boundary Classes

4.1.1 Ephemeris-pegged

The Ephemeris-pegged Boundary Class represents a boundary at the center of an ephemeris body such as the center of mass of a planet. Typically this Boundary Class is only useful for initial mission design such as patched-conic models of planetary flybys. The "ephemeris source" setting on the Physics Options tab controls how EMTG determines the location of the ephemeris body. See Section 6.5.

4.1.2 Ephemeris-referenced

The Ephemeris-referenced Boundary Class defined relative to an ephemeris point but not on the ephemeris point. In other words, the boundary point is "referenced" to an ephemeris point and

moves with it, but additional information is needed to define the boundary relative to the ephemeris point. In EMTG, Ephemeris-referenced boundary conditions are defined as lying on a triaxial ellipsoid centered on an ephemeris point. For example, this could be the surface of the sphere of influence of a planet or a triaxial ellipsoid representing the surface of a non-spherical body like Ceres. The user provides the three semi-axes of the ellipsoid in the International Celestial Reference Frame (ICRF) where the center of the ellipsoid is the ephemeris body.

4.1.3 Free Point

The Free Point Boundary Class represents a point in space that is defined as a cartesian, orbit element, spherical, or b-plane state relative to the central body. The user may choose to fix or vary within bounds any of the six elements of the position and velocity state on the left-hand side of the boundary.

4.1.4 Periapse

The Periapse Boundary Class represents events that happen at periapse of the spacecraft's orbit about the central body. In the current implementation, this class only guarantees that the spacecraft be at an apse, not necessarily the right one. Note that if a state representation that includes true anomaly (Classical Oribital Elements (COE), IncomingBplane, or OutgoingBplane) is chosen, then a periapse is guaranteed.

4.2 Departure Types

4.2.1 Launch or Direct Insertion

This Departure Type models launch from a body or an impulsive departure using three decision variables, the magnitude of the departure v_∞, and the right ascension and declination of the departure asymptote in the ICRF.

4.2.2 Depart from Parking Orbit

Depart from a parking orbit around the departure body. Use a Free Point Boundary Class to specify parameters of the orbit.

4.2.3 Free Direct Departure

Free Direct Departure is a type of boundary event in which a spacecraft departs from a point in space that is defined as a cartesian or COE state relative to the central body. The user may choose to fix any of the state elements relative to the central body or allow them to vary within specified bounds on the left-hand side of the boundary.

4.2.4 Flyby

A flyby is a trajectory segment in which a spacecraft passes close to a celestial body and uses the body's gravity to alter its trajectory. In EMTG, flybys are modeled using patched-conic approximations, which divide the trajectory into segments based on the gravitational influence of each celestial body. The spacecraft's trajectory is approximated as a series of conic sections that are tangent at the points where they transition from one body's sphere of influence to another. At later phases of mission design, flybys are modeled at higher fidelity using other Boundary Event combinations to go from incoming Sphere of Influence (SOI) crossing to periapse to outgoing SOI crossing.

4.2.5 Flyby with Fixed V-Infinity Out

Flyby with Fixed V-Infinity Out is a type of outgoing flyby used when the user wants to specify the value of $v_{\infty-out}$. This boundary event includes two equality constraints to guarantee that the magnitude and direction of $v_{\infty-out}$ match the user-specified values.

4.2.6 Zero Turn Flyby

This Departure Type is used when the flyby is of a very small body and therefore the change in the direction of the spacecraft velocity due to the body is small enough to not be modeled. When zero turn flyby and ephemeris point are used together, EMTG enforces the constraint that the spacecraft $v_{\infty-out}$ matches $v_{\infty-in}$. Users must take care that the small body assumption is sufficient, as this will ignore any velocity change that would normally occur even if a larger planetary body is chosen as the start location for the journey.

4.2.7 Spiral-out from Circular Orbit

Spiral-out from circular orbit is a technique used in EMTG to approximate a many-revolution low-thrust spiral about a body in the current universe. For example, one may wish to spiral from Low Earth Orbit (LEO) to escape from the Earth, or from the edge of the Mars sphere of influence down to the orbital distance of Phobos or Deimos. Edelbaum's approximation provides a fast, sufficiently accurate model that allows spirals to be included with an EMTG broad search without

having to explicitly model and optimize the path of the spacecraft during the spiral. Edelbaum's approximation consists of modeling the initial and final orbits about the body as co-planar circles and then assuming that the thrust level is sufficiently low that the transfer orbit is also nearly circular.

4.3 Arrival Types

4.3.1 Insertion into Parking Orbit (use chemical Isp)

Specifies that EMTG should solve for a maneuver that places the spacecraft into a parking orbit using its chemical thruster upon arrival at this body. Depending on the Boundary Class selected, users may specify only the semi-major axis (SMA) and eccentricity (ECC) of the parking orbit (Ephemeris-pegged Class) or a full set of varying or specified state variables (Free Point Class).

4.3.2 Rendezvous (with chemical maneuver)

This arrival type specifies that the spacecraft perform a chemical maneuver upon reaching the arrival body to either match velocities with the body in the case of an Ephemeris-pegged Boundary Class or achieve a state relative to the ephemeris body when combined with another Boundary Class.

4.3.3 Intercept with Bounded V-Infinity

This arrival type specifies that the spacecraft arrive at the ephemeris body with some maximum magnitude of velocity at infinity, v_∞, given by

$$v_\infty = \sqrt{v^2 - \frac{2\mu}{r}}. \tag{4.1}$$

4.3.4 Rendezvous (no maneuver)

This arrival type specifies that the spacecraft matches position and velocity with some point upon arrival. This point may be defined as some ephemeris point in the case of an Ephemeris-pegged Boundary Class, some point on the edge of the ephemeris bounding ellipse in the case of an Ephemeris-referenced Boundary Class, or some user defined Free Point in the case of the Free Point Boundary Class. This is generally intended for use explicitly with low-thrust transcriptions.

4.3.5 Match Final V-Infinity Vector (with chemical maneuver)

Arrival Type that forces the spacecraft to match a specified v_∞ vector upon arrival using a chemical maneuver. Selecting this Arrival Type in the Journey Options tab reveals additional options to specify the v_∞ vector.

4.3.6 Match Final V-Infinity Vector (no maneuver)

Arrival Type that forces the spacecraft to match a specified v_∞ vector upon arrival using a chemical maneuver. Selecting this Arrival Type in the Journey Options tab reveals additional options to specify the v_∞ vector.

4.3.7 Capture Spiral

Models a many-revolution low-thrust spiral capture about a body in the current universe similarly to the "spiral-out from circular orbit" Departure Type

4.3.8 Momentum Transfer

Momentum Transfer is a specific type of Arrival Type used in unique scenarios where the spacecraft collides with the destination body, resulting in the transfer of momentum to the body.

Chapter 5

Universe File and Options

EMTG Universe files define all celestial body and ephemeris data required by EMTG. This mostly consists of astrodynamics data pertaining to the central body. For a given Journey, the geometric center of the central body serves as the center of propagation in EMTG. The file takes the extension `.emtg_universe` and consists of two major sections: the central body information and a menu of bodies which define the list of available flyby objects for the Journey.

5.1 Central Body Definition

The central body information is used to define the central body for a given Journey, and thus multiple `.emtg_universe` files are required when multiple central bodies are needed in a mission. This information is listed first in an `.emtg_universe` file but is not contained in any specific block. The lines and expected data types are shown in Table 5.1, with additional details following.

Line	Line Name	Data Type
1	central_body_name	String
2	central_body_SPICE_ID	Integer
3	central_body_radius	Real (km)
4	central_body_J2	Real
5	central_body_J2_reference_radius	Real (km)
6	central_body_flattening_coefficient	Real
7	mu	Real (km^3/s^2)
8	LU	Real (km)
9	reference_angles	Real (degrees)
10	r_SOI	Real (km)
11	minimum_safe_distance	Real (km)

Table 5.1: Universe File Central Body Information

central_body_name:

The name of the central body.

central_body_SPICE_ID:

The Spacecraft Planet Instrument Camera-matrix Events (SPICE) ID associated with the central body.

central_body_radius:

The radius of the central body.

central_body_J2:

Optionally provided J2 value for the central body. Defaults to 0.

central_body_J2_reference_radius:

Optionally provided J2 reference radius for the central body. Defaults to 0.

central_body_flattening_coefficient:

Optionally provided flattening coefficient to define oblateness of central body. Defaults to 0.

mu:

Gravitational constant of central body.

LU:

The characteristic length unit used in scaling the problem internally.

reference_angles:

Six space-delimited values: alpha0, alphadot, delta0, deltadot, W, Wdot. These angles define the local reference frame relative to ICRF, in degrees and degrees per century. The angle alpha0 defines the right ascension angle used to describe the north pole of the central body relative to the invariable plane of the solar system. The angle delta0 defines the declination angle describing the angular distance of the central body's north pole from the celestial equator. The angle W defines the location of the prime meridian of the central body measured easterly along the body's equator from a node defined as an intersection of the body equator with the celestial equator. The angular rates alphadot, deltadot, and Wdot are the rates of change of the associated angles due to precession of the axis of rotation.

r_SOI:

Radius of the central body's sphere of influence.

minimum_safe_distance:

The minimum safe distance from the central body, some value greater than the radius of the central body.

5.2 Menu of Bodies Details

The menu of bodies details information similar to that of the central body, but instead for a list of bodies available for flybys or third body perturbation effects. The menu of bodies is attached to the `.emtg_universe` file much the same way as a central body, so for missions where multiple central bodies are used the menu of bodies must be present in each `.emtg_universe` file. Unlike

the central body information however, each body has all of its information listed on one space delimited line. The indices, varibale names, and expected data types are shown in Table 5.2, with additional details following.

Index	Variable Name	Data Type
1	Name	String
2	Short Name	String
3	Number	Integer
4	SPICE_ID	Integer
5	minimum_flyby_altitude	Real (km)
6	GM	Real (km^3/s^2)
7	Radius	Real (km)
8	body_flattening_coefficient	Real
9	body_J2	Real
10	body_AbsoluteMagnitude	Real
11	body_albedo	Real
12	ephemeris_epoch	Real
13	alpha0	Real (degrees)
14	alphadot	Real (degrees/century)
15	delta0	Real (degrees)
16	deltadot	Real (degrees/century)
17	W	Real (degrees)
18	Wdot	Real (degrees/century)
19	SMA	Real (km)
20	ECC	Real
21	INC	Real (degrees)
22	RAAN	Real (degrees)
23	AOP	Real (degrees)
24	MA	Real (degrees)

Table 5.2: Universe File Menu Of Bodies Specifications

Name:

Details a full name for a body appearing on the menu of bodies.

Short Name:

A short hand call out to a body which is used to name results files. Often a single letter (e.g., J for Jupiter).

Number:

A number associated with the body, used to set destination lists and flyby sequences as an easy to parse list of integers.

SPICE_ID:

The SPICE ID associated with the body being set. Lists of IDs are available at https://naif.jpl.nasa.gov/pub/naif/toolkit_docs/C/req/naif_ids.html.

minimum_flyby_altitude:

A lower bound on the safe flyby altitude for a body appearing on the list of available flyby objects. If the value is ≤ 0, the object is not placed on the flyby menu.

GM:

The gravitational parameter of the body.

Radius:

The radius of the body.

body_flattening_coefficient:

A flattening coefficient defining the oblateness of the body.

body_J2:

The J2 zonal harmonic term of the body.

body_AbsoluteMagnitude:

The brightness of the body. This is included for future use in a genetic algorithm to aid in defining a potentially interesting small body flyby.

body_albedo:

The fraction of light a body reflects. This is used with the body absolute magnitude to estimate the body radius.

ephemeris_epoch:

A reference epoch for the body provided as a modified Julian date, used to verify the body exists in the provided SPICE ephemeris at this point.

alpha0:

The right ascension angle used to define the pole of rotation that lies on the north side of the invariable plane of the solar system. This right ascension describes the angular distance from the Earth equinox at J2000 and the hour circle passing through the object, the hour circle being the great circle passing through a celestial object and the two celestial poles.

alphadot:

The rate of change of the right ascension angle used to define the pole of rotation of the celestial body due to precession of the axis of rotation of the body.

delta0:

The declination angle used to define the pole of rotation that lies on the north side of the invariable plane of the solar system. This declination describes the angular distance of said north pole to the celestial (ICRF) equator.

deltadot:

The rate of change of the declination angle used to define the pole of rotation of the celestial body due to precession of the axis of rotation of the body

W:

The angle defining the location of the prime meridian of the celestial body, measured easterly along the body's equator from a node defined as an intersection of the body equator with the celestial (ICRF) equator.

Wdot:

The rate of change of the angle used to define the location of the prime meridian of the celestial body due to precession of the axis of rotation of the body

SMA:

Semi-major axis of the body at the provided reference epoch with respect to the central body local reference frame. Overridden if ephemeris is drawn from SPICE.

ECC:
Eccentricity of the body at the provided reference epoch with respect to the central body local reference frame. Overridden if ephemeris is drawn from SPICE.

INC:
Inclination of the body at the provided reference epoch with respect to the central body local reference frame. Overridden if ephemeris is drawn from SPICE.

RAAN:
Right Ascension of the Ascending Node of the body at the provided reference epoch with respect to the central body local reference frame. Overridden if ephemeris is drawn from SPICE.

AOP:
Argument of Perigee of the body at the provided reference epoch with respect to the central body local reference frame. Overridden if ephemeris is drawn from SPICE.

MA:
Mean Anomaly of the body at the provided reference epoch with respect to the central body local reference frame. Overridden if ephemeris is drawn from SPICE.

Chapter 6

EMTG Options

An EMTG mission is created through a text file known as an EMTG Options file with the extension `.emtgopt`. EMTG Options files define the mission parameters as well as link to the ephemeris and hardware configuration needed for the mission. The ephemeris configuration is another text file known as an EMTG Universe file with extension `.emtg_universe`. Hardware configurations are stored in another set of text files for the launch vehicle and spacecraft with extensions `.emtg_launchvehicleopt` and `.emtg_spacecraftopt`. EMTG Options files can be configured using a text editor or the PyEMTG GUI. This chapter will discuss the EMTG Options file and PyEMTG interface.

Most interaction with the EMTG Options file can be accomplished through PyEMTG though some options are only present in the `.emtgopt` file and must be changed there using a text editor. An EMTG Options file can be created in PyEMTG by selecting "File, New, Mission" (Ctrl + m). An existing EMTG Options file can be opened by selecting "File, Open" (Ctrl + o) and choosing an existing `.emtgopt` file.

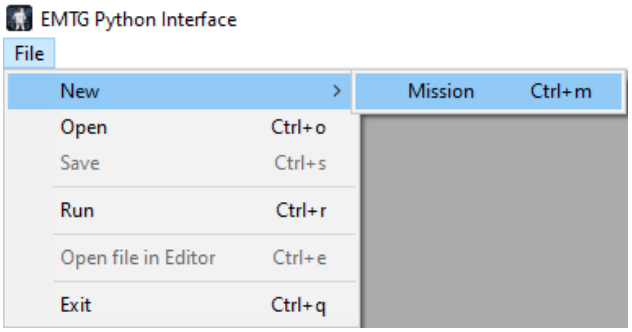


Figure 6.1: Creating a new EMTG Options file

An EMTG Options file is broken up into several sections shown with tabs in PyEMTG: Global Mission Options, Spacecraft Options, Journey Options, Solver Options, Physics Options, and Output Options. The rest of the chapter will cover each of the sections in turn as well as the options that can only be accessed through the `.emtgopt` text file.

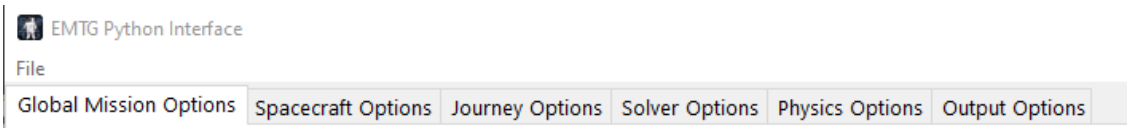


Figure 6.2: EMTG Options tabs

6.1 Global Mission Options and Constraints

The Global Mission Options tab covers options and constraints that apply to the entire mission such as the objective function, launch bounds, overall mission time bounds, and other settings.

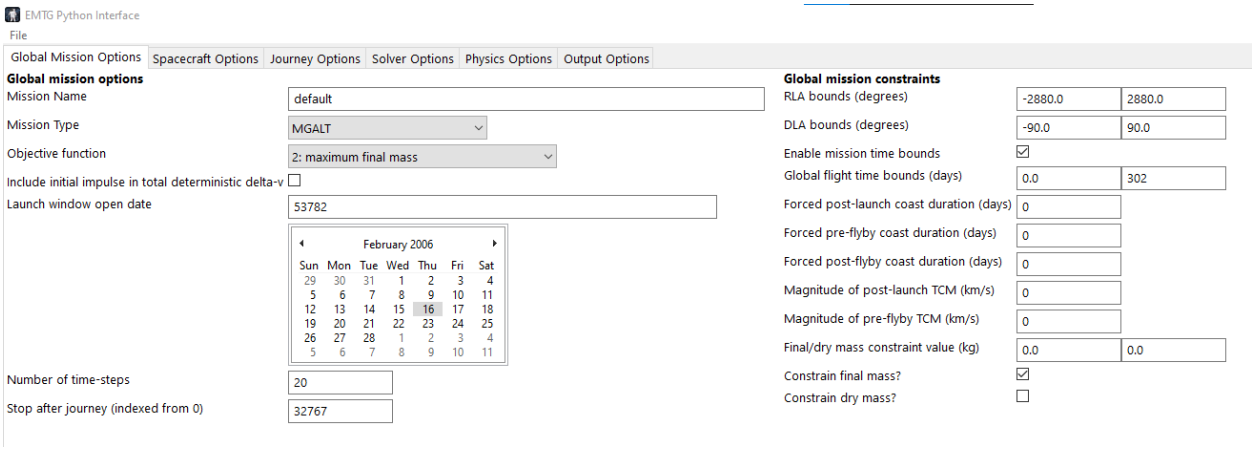


Figure 6.3: EMTG Global Options tab

6.1.1 Global Mission Options

1. **Mission Name:** The mission name is used to distinguish EMTG missions and will prefix the output directories and files EMTG produces when a mission is run.

Data Type	`string`
Default Value	"default"

2. **Mission Type:** EMTG provides several different Mission Types which control how an EMTG mission is modeled. Mission Types come in various levels of fidelity and may be selected for the entire mission or on a per-Journey basis. Mission Types are covered in more detail in Chapter 3.

Data Type	`PhaseType`
Allowed Values	`MGALT, FBLT, PSBI, PSFB, MGAnDSMs, CoastPhase, SundmanCoastPhase, Variable phase type, ProbeEntryPhase, ControlLawThrustPhase`
Default Value	`MGALT`

3. **Objective Function:** This option specifies the mission parameter to be maximized or minimized such as maximizing final mass or minimizing delta-v.

Data Type	`PhaseType`

Allowed Values	`0: minimum deterministic deltaV,`
	`1: minimum time,`
	`2: maximum final mass,`
	`3: maximize initial mass,`
	`4: depart as late as possible,`
	`5: depart as early as possible,`
	`6: maximize orbit energy,`
	`7: minimize launch mass,`
	`8: arrive as early as possible,`
	`9: arrive as late as possible,`
	`10: minimum propellant,`
	`11: maximum dry/wet ratio,`
	`12: maximum arrival kinetic energy,`
	`13: minimum BOL power,`
	`14: maximize log\_10(final mass),`
	`15: maximize log\_e(final mass),`
	`16: maximum dry mass margin,`
	`17: maximum dry mass,`
	`18: maximum log\_10(dry mass),`
	`19: maximum log\_e(dry mass),`
	`20: minimize chemical fuel,`
	`21: minimize chemical oxidizer,`
	`22: minimize electric propellant,`
	`23: minimize total propellant,`
	`24: minimize waypoint tracking error,`
	`25: minimize initial impulse magnitude,`
	`26: maximize distance from central body`

Default Value	`2: maximum final mass`

4. **Include initial impulse in total deterministic delta-v:** This options controls whether or not the delta-v provided by the launch vehicle is included in the objective function, "0: minimum deterministic deltaV."

Data Type	`bool`
Allowed Values	true, false
Default Value	false
Units	NA

5. **Launch window open date:** The earliest possible launch date expressed as an MJD2000 value. This value can be set using the text box or the calendar tool below.

Data Type	`double`
Allowed Values	$0 <$ Real $< \infty$
Default Value	0.0
Units	days since MJD2000 epoch

6. **Number of time-steps:** Number of time-steps per journey/phase. This controls how many segments define each Journey. A higher value increases the mission fidelity at the cost of run time. This option can be overridden on a per-Journey basis in the Journey Options tab.

Data Type	`int`
Allowed Values	$0 <$ Integer $< \infty$
Default Value	20
Units	NA

7. **Stop after journey (indexed from 0):** This will cause EMTG to only optimize up to and including the integer Journey number set in the text box. For example, if the EMTG Options file has 3 Journeys (Journey 0 to Journey 2) and this text box is set to 1, EMTG will not solve Journey 2.

Data Type	`int`
Allowed Values	0 < Integer < (Number of Journeys - 1)
Default Value	32767
Units	NA

6.1.2 Global Mission Constraints

1. **RLA bounds (degrees):** Bound the right ascension of the launch asymptote (equatorial east longitude from the vernal equinox).

Data Type	`std::vector<double, 2>`
Allowed Values	$-\infty <$ Real $< \infty$
Default Value	[-2880.0, 2880.0]
Units	degrees

2. **DLA bounds (degrees):** Bound the declination (latitude) of the launch asymptote.

Data Type	`std::vector<double, 2>`
Allowed Values	-90 < Real < 90
Default Value	[-90.0, 90.0]
Units	degrees

3. **Enable mission time bounds:** Selecting this box will reveal the Global flight time upper and lower bounds boxes to constrain the mission length.

Data Type	`bool`
Allowed Values	true, false
Default Value	true
Units	NA

4. **Global flight time bounds (days):** Used to set boundaries on mission length in days where 0.0 is the launch epoch. Revealed and active only if "Enable mission time bounds" is checked.

Data Type	`std::vector<double, 2>`
Allowed Values	$0 <$ Real $< \infty$
Default Value	[0, 302]
Units	days

5. **Forced post-launch coast duration (days):** Restrict EMTG from placing any maneuvers between launch and this option's value in days. This can be used to ensure adequate time for spacecraft checkout after launch.

Data Type	`double`
Allowed Values	$0 <$ Real $< \infty$
Default Value	0.0
Units	days

6. **Forced pre-flyby coast duration (days):** Restrict EMTG from placing any maneuvers this many days prior to a planetary flyby. This can be used to ensure adequate time for trajectory error cleanup prior to the flyby.

Data Type	`double`
Allowed Values	$0 <$ Real $< \infty$
Default Value	0.0
Units	days

7. **Forced post-flyby coast duration (days):** Restrict EMTG from placing any maneuvers this many days after a planetary flyby.

Data Type	`double`
Allowed Values	$0 <$ Real $< \infty$
Default Value	0.0
Units	days

8. **Magnitude of post-launch TCM (km/s):** Sets the upper bound on a post-launch TCM. EMTG sets the lower bound to 0 km/s internally. The post-launch TCM is assumed to be performed by the monopropellant propulsion system and is used to compute the mass loss due to the TCM on the left boundary of a phase.

Data Type	`double`
Allowed Values	$0 <$ Real $< \infty$
Default Value	0.0
Units	km/s

9. **Magnitude of pre-flyby TCM (km/s):** Sets the upper bound on a pre-flyby TCM. EMTG sets the lower bound to 0 km/s internally. The pre-flyby TCM is assumed to be performed by the monopropellant propulson system and is used to compute the mass loss due to TCM immediately before a flyby.

Data Type	`double`
Allowed Values	$0 <$ Real $< \infty$
Default Value	0.0
Units	km/s

10. **Constrain final/dry mass?:** Checking either of these options reveals a set of text boxes to bound either the final mass or the dry mass of the spacecraft at the end of the mission. Final mass is also known as final wet (including propellant) mass. These options are applied separately, allowing users to constrain both the vehicle dry mass and the final mission mass at the same time.

Data Type	bool
Allowed Values	true, false
Default Value	false
Units	NA

11. **Final/dry mass constraint value (kg):** Mass values for the final/dry mass constraint. Revealed and active only if either the constrain final or dry mass is active.

Data Type	`std::vector<double, 2>`
Allowed Values	$0 <$ Real $< \infty$
Default Value	[0.0, 0.0]
Units	kg

6.2 Spacecraft Options

The Spacecraft Options tab allows users to specify the launch vehicle and spacecraft used to define a mission. The launch vehicle is defined explicitly with a configuration file, while a spacecraft may be defined either via configuration files or within the Spacecraft Options tab directly. The spacecraft configuration files allow for greater flexibility when defining a spacecraft than may be easily achieved via the PyEMTG window. For details on how to create configuration files for launch vehicles and spacecraft, as well as the available propulsion, power, and constraint options which may be defined, refer to Chapter 7.

While many of the options defining spacecraft are explicitly defined in the configuration files, some of the options defining how EMTG handles a spacecraft still must be set in the Spacecraft Options tab.

6.2.1 Common Hardware Options

Common hardware options define the most general parameters for a missions vehicles as well as the spacecraft model type used to define the spacecraft.

1. **Maximum mass (kg):** A "maximum mass" field which scales the optimization problem. In EMTG, the actual launch mass is limited to whatever the launch vehicle can carry to the C3 chosen by the optimizer. The launch vehicle configuration file contains settings which define the function relating injected mass to C3 (see Section 7.1).

EMTG Variable Name	`maximum_mass`
Data Type	`double`
Allowed Values	1.0E-10 < Real < 1.0E-30
Default Value	525.2
Units	kg

2. **Allow initial mass to vary:** Option to allow EMTG to use less initial mass than the launch vehicle can inject to a given C3 if that results in a higher final mass. Initial mass is still bounded by the capabilities of the launch vehicle (i.e., injected mass to C3).

EMTG Variable Name	`allow_initial_mass_to_vary`
Data Type	`bool`
Allowed Values	true, false
Default Value	false

3. **Spacecraft model type:** Specifies the method used to define the spacecraft.

EMTG Variable Name	`SpacecraftModelInput`
Data Type	`(SpacecraftModelInputType) enum`

Allowed Values	`0: Assemble from libraries,`
	`1: Read .emtg_spacecraftoptions file,`
	`2: Assemble from missionoptions object`

Default Value	`2: Assemble from missionoptions object`

   Descriptions of the available options follows.

   `0: Assemble from libraries`:

Uses `.emtg_powersystemsopt` and `.emtg_propulsionsystemopt` files to define the power and propulsion systems of a spacecraft, while opening up a "Tanks" section in PyEMTG which allows users to set and define propellant tank constraints for both electric and chemical propulsion systems. Generally, it is recommended to avoid this mode as it is more complex than mode 2 while providing less flexibility than mode 1.

`1: Read .emtg_spacecraftoptions file`:
Uses a `.emtg_spacecraftopt` file as defined in Section 7.2 to define the spacecraft. Where possible it is recommended for users to use this option as it provides the greatest flexibility.

`2: Assemble from missionoptions object`:
Allows users to set options defining a spacecraft directly in the Spacecraft Options tab by opening up "Propulsion options", "Power options", and "Tanks" sections. The options which may be set here are discussed in greater detail in Section 7.2.

4. **Hardware library path:** Specifies the path to the folder containing all options files for launch vehicles and spacecraft. A **trailing slash is required** to avoid an error in PyEMTG.

EMTG Variable Name	`HardwarePath`
Data Type	`string`
Default Value	"c:/Utilities/HardwareModels/"

5. **Launch vehicle library file:** Specifies which `.emtg_launchvehicleopt` file in the hardware library path to use to obtain available launch vehicles for the mission. See Section 7.1 for details on the configuration of these files.

EMTG Variable Name	`LaunchVehicleLibraryFile`
Data Type	`string`
Default Value	"NLSII_August2018.emtg_launchvehicleopt"

6. **Launch vehicle:** Sets the launch vehicle for the mission. Must be the name of a launch vehicle defined in the `.emtg_launchvehicleopt` file.

EMTG Variable Name	`LaunchVehicleKey`
Data Type	`string`
Default Value	"Atlas_V_401"

6.2.2 Spacecraft Model Type: 0

In "Spacecraft model type" mode 0, a launch vehicle options file, power systems option file, and propulsion system options file are required. The following are the options available when using this mode:

1. **Power systems library file:** Specifies which `.emtg_powersystemsopt` file in the hardware library path to use to obtain available power systems for the mission. See Section 7.2 for details on the configuration of these files.

EMTG Variable Name	`PowerSystemsLibraryFile`
Data Type	`string`
Default Value	"default.emtg_powersystemsopt"

2. **Propulsion systems library file:** Specifies which `.emtg_propulsionsystemopt` file in the hardware library path to use to obtain available propulsion systems for the mission. See Section 7.2 for details on the configuration of these files.

EMTG Variable Name	`PropulsionSystemsLibraryFile`
Data Type	`string`
Default Value	"4_18_2017.emtg_propulsionsystemopt"

3. **Power system:** Sets the power system for the mission. Must be the name of a power system defined in the `.emtg_powersystemsopt` file to use.

EMTG Variable Name	`PowerSystemKey`
Data Type	`string`
Default Value	"5kW_basic"

4. **Electric propulsion system:** Sets the electric propulsion system for the mission. Must be the name of a power system defined in the `.emtg_propulsionsystemopt` file.

EMTG Variable Name	`ElectricPropulsionSystemKey`
Data Type	`string`
Default Value	"NSTAR"

5. **Chemical propulsion system:** Sets the chemical propulsion system for the mission. Must be the name of a power system defined in the `.emtg_propulsionsystemopt` file.

EMTG Variable Name	`ChemicalPropulsionSystemKey`
Data Type	`string`
Default Value	"DefaultChemicalPropulsionSystem"

6. **Number of thrusters:** Specifies the number of thrusters to use for the power and propulsion system calculations. Further detail on these calculations is provided in Section 7.2.

EMTG Variable Name	`number_of_electric_propulsion_systems`
Data Type	`int`
Allowed Values	$1 <$ Real < 2147483647
Default Value	1

7. **Enable electric propulsion propellant tank constraint?:** Option to enable an upper bound constraint on available propellant for the electric propulsion system.

EMTG Variable Name	`enable_electric_propellant_tank_constraint`
Data Type	`bool`
Allowed Values	true, false
Default Value	false

8. **Maximum electric propulsion propellant (kg):** Only used when the electric tank constraint is active. Defines the upper bound on allowed propellant for the electric propulsion system.

EMTG Variable Name	`maximum_electric_propellant`
Data Type	`int`
Allowed Values	$0 <$ Real $<$ 1.0E30
Default Value	1000
Units	kg

9. **Enable chemical propulsion tank constraints?:** Option to enable an upper bound constraint on available fuel for the chemical propulsion system.

EMTG Variable Name	`enable_chemical_propellant_tank_constraint`
Data Type	`bool`
Allowed Values	true, false
Default Value	false
Units	kg

10. **Maximum chemical fuel (kg):** Only used when the chemical tank constraint is active. Defines the upper bound on allowed chemical fuel.

EMTG Variable Name	`maximum_chemical_fuel`
Data Type	`double`
Allowed Values	$0 <$ Real $<$ 1.0E30
Default Value	1000
Units	kg

11. **Maximum chemical oxidizer (kg):** Only used when the chemical tank constraint is active. Defines the upper bound on allowed chemical oxidizer.

EMTG Variable Name	`maximum_chemical_oxidizer`
Data Type	`double`
Allowed Values	$0 <$ Real $<$ 1.0E30
Default Value	1000
Units	kg

12. **Bipropellant mixture ratio:** Only used when both electric and chemical tank constraints are inactive. Defines the mixture ratio for the chemical thruster.

EMTG Variable Name	`bipropellant_mixture_ratio`
Data Type	`double`
Allowed Values	1.0E-10 $<$ Real $<$ 1
Default Value	0.925

6.2.3 Spacecraft Model Type: 1

In "Spacecraft model type" mode 1, a launch vehicle options file and spacecraft options file are required. Since the spacecraft options are all defined in the spacecraft options file, less options are present in the PyEMTG window. The following are the options available when using this mode:

1. **Spacecraft file:** Specifies which `.emtg_spacecraftopt` file in the hardware library path to use to define the spacecraft for the mission. See Section 7.2 for details on the configuration of these files.

EMTG Variable Name	`SpacecraftOptionsFile`
Data Type	`string`
Default Value	"default.emtg_spacecraftopt"

6.2.4 Spacecraft Model Type: 2

In "Spacecraft model type" mode 2 only a launch vehicle options file is required. All other options are set by the user. The following are the options used to define a launch vehicle in this mode:

1. **Engine type:** Allows the user to specify a specific thruster mode which opens up many additional options defining how thrust and mass flow rate are calculated. There are 32 available options a user may choose in this field in PyEMTG, though not all are functional. Furthermore, the numbering of these modes does not match that used for defining an engine type in a `.emtg_spacecraftopt` file. Given the variety of options present here and to avoid repeated details in this document, refer to Section 7.2.4 for detailed explanations of these options. These options are also discussed in Section 5.6 of the EMTG Software Design Document.

EMTG Variable Name	`engine_type`
Data Type	`int`
Default Value	`5: custom thrust and mass flow rate polynomial`

6.2.5 Additional Spacecraft Options

The Spacecraft Options tab contains other various options which are present no matter the chosen "Spacecraft model type". These options follow:

1. **Launch vehicle margin (fraction):** Sets a margin which scales down the calculated injected mass to a C3 value for a launch vehicle by some percentage. The calculation is detailed in Section 7.1.

EMTG Variable Name	`LV_margin`
Data Type	`double`
Allowed Values	$0 <$ Real < 1
Default Value	0

2. **Power margin (fraction):** Sets a margin which scales down the calculated available power for the propulsion system by some percentage. A value of zero causes no reduction in available power other than that required by the spacecraft bus for non-propulsive functions. The calculation is detailed in Section 7.2.3.

EMTG Variable Name	`power_margin`
Data Type	`double`
Allowed Values	$0 <$ Real < 1
Default Value	0

3. **Thruster duty cycle:** Defines the percentage of time the engine can operate.

EMTG Variable Name	`engine_duty_cycle`
Data Type	`double`
Allowed Values	1.0E-10 $<$ Real < 1
Default Value	1

4. **Duty cycle type:** Defines whether EMTG uses "averaged" or "realistic" duty cycles for thrust arcs in the PSFB transcription (defined in Chapter 3). Defaults to "0: Averaged".

EMTG Variable Name	`duty_cycle_type`
Data Type	`(DutyCycleType) enum`
Allowed Values	`0: Averaged,` `1: Realistic`
Default Value	`0: Averaged`

5. **Electric propulsion propellant margin:** Sets a electric propellant margin for an objective to ensure some percentage of propellant for the electric propulsion system is retained when maximizing mass as an objective or applying dry mass constraints. This bound also modifies the upper bound on available electric propellant by scaling the user-defined tank size. A value of 0 specifies that no propellant need be saved.

EMTG Variable Name	`electric_propellant_margin`
Data Type	`double`
Allowed Values	$0 <$ Real < 1
Default Value	1

6. **Chemical propulsion propellant margin:** Sets a chemical propellant margin for an objective to ensure some percentage of propellant for the chemical propulsion system is retained when maximizing mass as an objective or applying dry mass constraints. This bound also modifies the upper bound on available chemical propellant by scaling the user-defined tank size. A value of 0 specifies that no propellant need be saved.

EMTG Variable Name	`chemical_propellant_margin`
Data Type	`double`
Allowed Values	$0 <$ Real < 1
Default Value	1

7. **Track ACS propellant?:** Enables a constant mass leak term to approximate propellant consumption due to the Attitude Control System (ACS) maneuvers. Propellant loss is applied against the spacecraft's chemical fuel tank. Opens an additional option to define ACS propellant use per day.

EMTG Variable Name	`trackACS`
Data Type	`bool`
Allowed Values	true, false
Default Value	false

8. **ACS propellant use per day (kg):** Only available when "Track ACS propellant" is enabled. Defines the constant mass leak in kilograms of chemical propellant a spacecraft will lose per day due to ACS maneuvers.

EMTG Variable Name	`ACS_kg_per_day`
Data Type	`double`
Allowed Values	$0 <$ Real $< 1.0E30$
Default Value	0
Units	kg

9. **Throttle logic mode:** Defines the logic in calculating the number of thrusters to use for a propulsion system. Defaults to minimum number of thrusters. Is overridden when using an `.emtg_spacecraftopt` file to define a spacecraft.

EMTG Variable Name	`throttle_logic_mode`
Data Type	`(ThrottleLogic) enum`
Allowed Values	`0: Maximum Number Of Thrusters,` `1: Minimum Number Of Thrusters`
Default Value	`1: Minimum Number Of Thrusters`

10. **Throttle sharpness:** Defines how quickly the thruster transitions between different settings in some thruster modes. Additional detail is provided in Section 7.2.2. Is overridden when using an `.emtg_spacecraftopt` file to define a spacecraft.

EMTG Variable Name	`throttle_sharpness`
Data Type	`double`
Allowed Values	$1 <$ Real $<$ 1.0E5
Default Value	1.0E2

11. **Power Source Decay Reference Epoch:** Defines the reference epoch used when calculating how the decay rate of the power supplied affects the actual generated power as compared to the expected nominal value. This calculation is described in further detail in Section 7.2.3. This date may be set manually as a Julian date or set by the provided calendar in PyEMTG.

EMTG Variable Name	`power_system_decay_reference_epoch`
Data Type	`double`
Allowed Values	$0 <$ Real $<$ 1.0E30 $\cdot$86400.0
Default Value	$51544.5 \cdot 86400.0$

6.3 Journey Options

The Journey Options tab contains all of the settings necessary to define the spacecraft trajectory and the bodies it encounters along the way. An EMTG mission requires at least one Journey, and each Journey contains an arrival and departure event. In early stages of mission design, a single Journey can model at low fidelity an entire mission including multiple flybys of Universe bodies. As mission design progresses, this low fidelity trajectory can be converted into multiple Journeys modeling different phases of the mission in higher accuracy. A single gravity assist may be modeled as a Journey from the SOI crossing to periapse and another Journey from periapse to the next SOI crossing. Chapter 8 discusses the tools EMTG provides for converting low-fidelity missions into higher fidelity. Users should also reference Chapter 4 for more information on configuration of Journey arrival and departure events.

Figure 6.4 show the default Journey Options tab in PyEMTG. There are many other Journey settings which are revealed based on other selections such as specifying the arrival state at a body. There are also multiple Journey settings which will override settings on other tabs that apply to the overall mission.

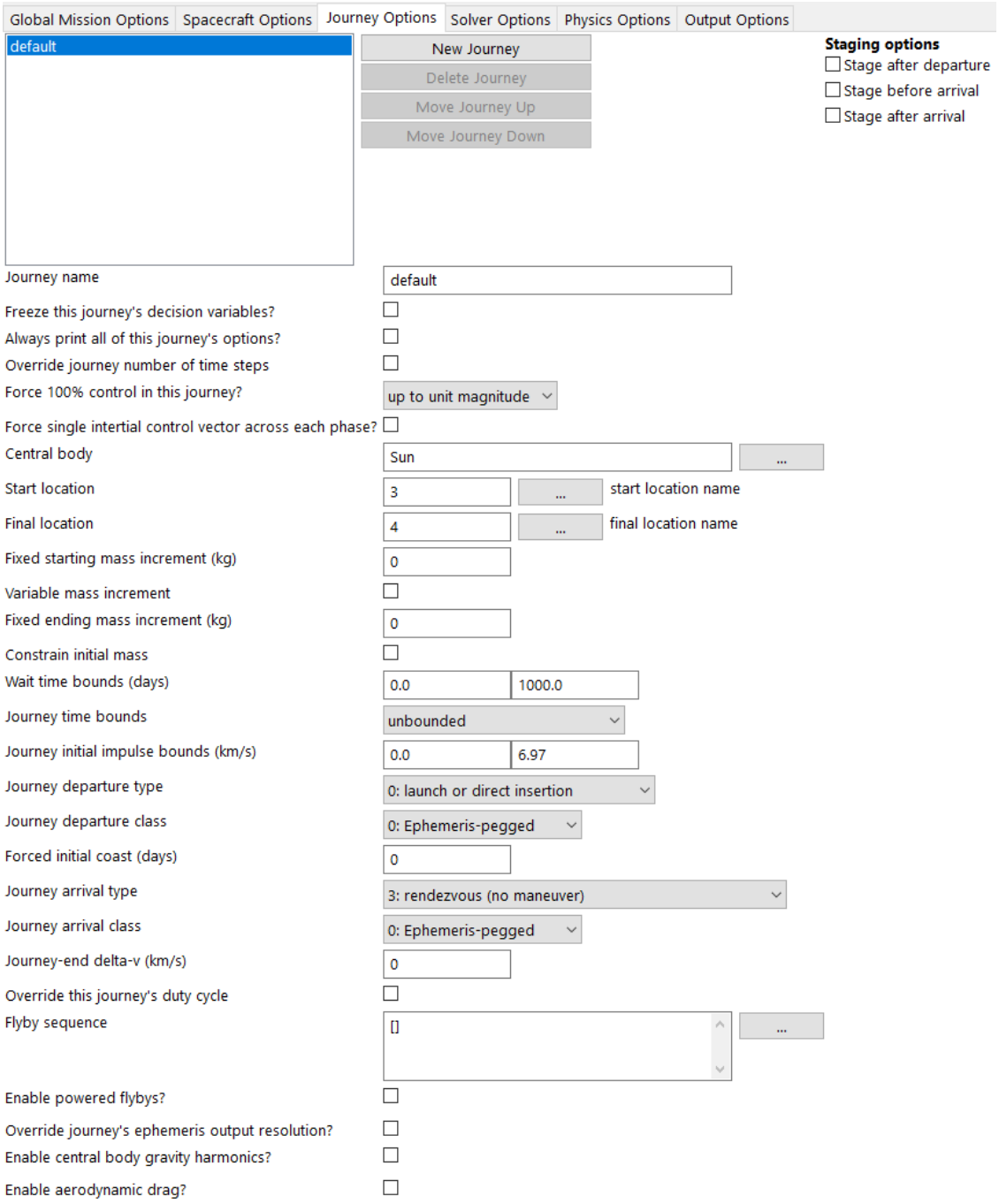


Figure 6.4: EMTG Journey Options tab

In the upper left corner of the Journey Options tab is a set of buttons and a selector to manage the Journeys in the EMTG options file. The rest of the options on the Journey Options tab correspond only to the currently selected Journey which will be highlighted in this window. The "New Journey" and "Delete Journey" buttons are used to create and remove Journeys from the EMTG options file. The "Move Journey Up" and "Move Journey Down" buttons are used to change the order of the Journeys in the mission sequence. An EMTG mission with several Journeys and a planetary

flyby might look like the example in Figure 6.5.

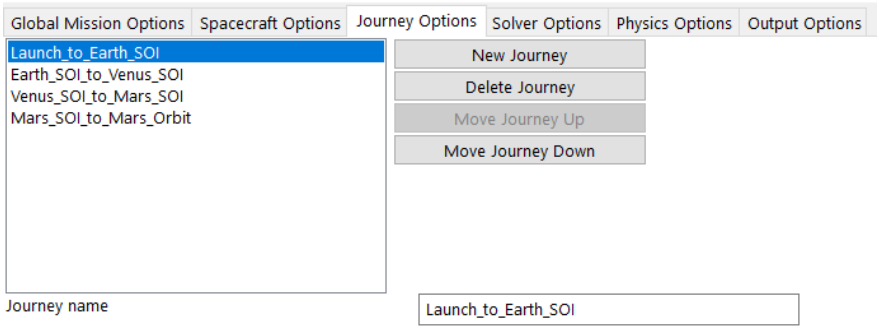


Figure 6.5: Journey Manager

6.3.1 Overall Journey Options

1. **Journey Name:** The Journey name is used to distinguish the settings pertaining only to that Journey in the EMTG options file and output files. Often these names specify the departure and arrival boundaries covered by the Journey.

Data Type	`string`
Default Value	"default"

2. **Freeze this journey's decision variables:** This options allows the user to hold the decision variables for the selected Journey constant when running EMTG. The variables pertaining to any Journey with this setting selected in the EMTG options file will not be changed by MBH or SNOPT at runtime. This can be used if only part of an EMTG mission needs to be adjusted without interfering with other Journeys and can decrease EMTG's runtime.

Data Type	`bool`
Allowed Values	true, false
Default Value	false
Units	NA

3. **Always print all of this journey's options?:** This setting is similar to the "Print only non-default options to .emtgopt file?" setting on the Output Options tab (Section 6.6). Selecting this option for a Journey will cause the EMTG options file to show all options for the Journey even those which have not been changed from their default value. This setting overrides the "Print only non-default options to .emtgopt file?" setting on the Output Options tab.

Data Type	`bool`
Allowed Values	true, false
Default Value	false
Units	NA

4. **Override journey number of time steps?:** This setting allows the user to override the

"Number of time-steps" setting on the Global Mission Options tab for the selected Journey. Selecting this option reveals and activates the "Number of time steps" setting in PyEMTG.

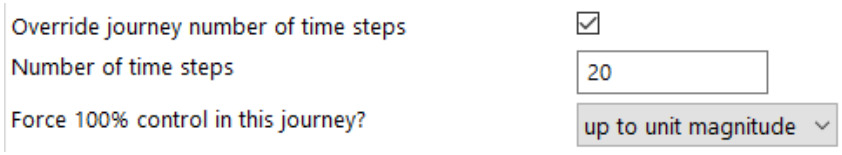


Figure 6.6: Journey Time Steps

Data Type	`bool`
Allowed Values	true, false
Default Value	false
Units	NA

5. **Number of time steps:** Override the number of time steps specified in the Global Options tab for this Journey only.

Data Type	`int`
Allowed Values	$0 <$ Integer $< \infty$
Default Value	20
Units	NA

6. **Force 100% control in this journey?:** A control 3-vector **u** is applied to each control segment in a low-thrust EMTG Mission Type. The magnitude of **u** represents the duty cycle during that segment and is constrained to be ≤ 1. This corresponds to the default option for this setting "up to unit magnitude." The user may also select "unit magnitude" forcing the duty cycle to be 1.0 or "zero magnitude" forcing control to be off. This setting is only active when Journey Mission Type is MGALT, FBLT, PSBI, or PSFB.

Data Type	`enum`

Allowed Values	`0: up to unit magnitude,`
	`1: unit magnitude,`
	`2: zero magnitude`

Default Value	0

7. **Force single inertial control vector across each phase?:** This option forces EMTG to point the control vector in a constant, inertial direction for the duration of the phase by forcing all control vectors to match each other.

Data Type	`bool`
Allowed Values	true, false
Default Value	false
Units	NA

8. **Central body:** Specifies the name of the EMTG Universe file which contains the central body for this Journey as well as any other bodies necessary for this Journey.

Data Type	`string`
Default Value	"Sun"

9. **Start location:** Specifies the starting body for the Journey from the Universe file specified in the "Central body" option according to the number in the Universe file not the SPICE ID.

Data Type	`int`
Allowed Values	$1 <$ Integer $\leq$ number of bodies in Universe file
Default Value	3
Units	NA

10. **Final location:** Specifies the final body for the Journey from the Universe file specified in the "Central body" option according to the number in the Universe file not the SPICE ID.

Data Type	`int`
Allowed Values	$1 <$ Integer $\leq$ number of bodies in Universe file
Default Value	4
Units	NA

11. **Fixed starting mass increment (kg):** Increase the mass of the spacecraft at the start of this Journey by a fixed amount. Use a negative number to indicate a mass drop at the start of this Journey.

Data Type	`double`
Allowed Values	$-\infty <$ Real $< \infty$
Default Value	0.0
Units	kg

12. **Variable mass increment (kg):** Selecting this option will convert the "Fixed starting mass increment" setting to minimum and maximimum mass increment settings to bound the mass change at the start of the Journey.

Data Type	`bool`
Allowed Values	true, false
Default Value	false
Units	NA

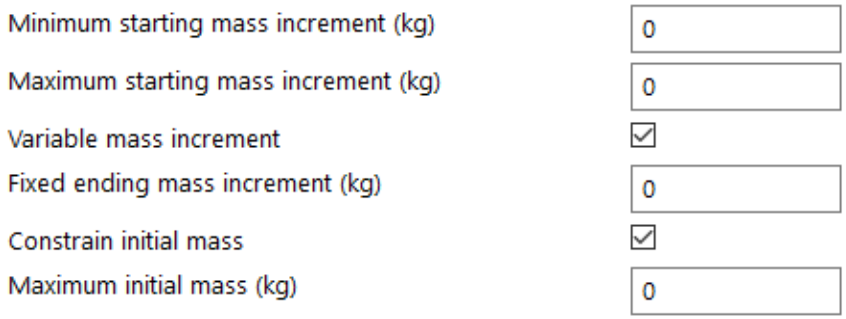


Figure 6.7: Journey Mass Increment Options

13. **Minimum starting mass increment (kg):** Lower bound on Journey initial mass change. Use a negative number to indicate a mass drop at the start of this Journey.

Data Type	`double`
Allowed Values	$-\infty <$ Real $<$ `maximum starting mass increment`
Default Value	0.0
Units	kg

14. **Maximum starting mass increment (kg):** Upper bound on Journey initial mass change. Use a negative number to indicate a mass drop at the start of this Journey.

Data Type	`double`
Allowed Values	`minimum starting mass increment` $<$ Real $< \infty$
Default Value	0.0
Units	kg

15. **Fixed ending mass increment (kg):** Increase the mass of the spacecraft at the end of this Journey by a fixed amount. Use a negative number to indicate a mass drop at the end of this Journey.

Data Type	`double`
Allowed Values	$-\infty <$ Real $< \infty$
Default Value	0.0
Units	kg

16. **Constrain initial mass (kg):** Constrain the maximum value for the spacecraft mass at the start of this Journey. Reveals the "Maximum initial mass" setting.

Data Type	`bool`
Allowed Values	true, false
Default Value	false
Units	NA

17. **Maximum initial mass (kg):** Set the maximum value for the spacecraft mass at the start of this Journey.

Data Type	`double`
Allowed Values	$0 <$ Real $< \infty$
Default Value	0.0
Units	kg

18. **Wait time bounds (days):** Bounds the time the spacecraft can remain at it's current body before departing on the next Journey. Can be used to remain in orbit around a body for a variable amount of time or delay launch from a body.

Data Type	`double`
Allowed Values	$0 <$ Real $< \infty$
Default Value	0.0 and 1000.0
Units	day

19. **Journey time bounds:** Select whether to place bounds on the timespan of the Journey. Reveals additional settings depending on method chosen. Flight time refers to the sum of all phase time of flight variables and boundary event time widths in the Journey. Aggregate flight time refers to the sum of all phase flight time and boundary event time width variables up to and including the Journey. Arrival epoch computed by adding the launch epoch to the aggregate flight time.

Data Type	`int`

Allowed Values	`0: unbounded,`
	`1: bounded flight time,`
	`2: bounded arrival date,`
	`3: bounded aggregate flight time`

Default Value	0

20. **Journey flight time bounds (days):** Set bounds on the flight time of the Journey in days. Revealed when "Journey time bounds" is set to "bounded flight time" or "bounded aggregate flight time."

Data Type	`double`
Allowed Values	$0 <$ Real $< \infty$
Default Value	0.0 and 1000.0
Units	day

21. **Journey arrival date bounds:** Set bounds on the arrival date of the Journey in days. Revealed when "Journey time bounds" is set to "bounded arrival date." Enter a date in MJD2000 or select date using the calendars revealed in PyEMTG.

Data Type	`double`
Allowed Values	$0 <$ Real $< \infty$
Default Value	51544.5 and 60000.0
Units	days since MJD2000 epoch

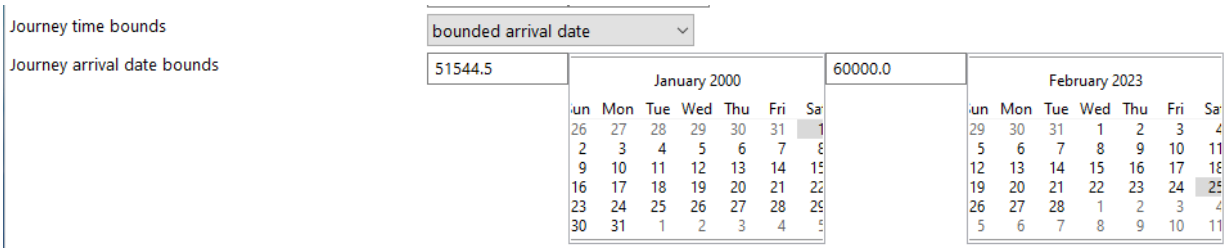


Figure 6.8: Journey bounded arrival date

6.3.2 Journey Departure Options

1. **Journey initial impulse bounds (km/s):** Set bounds on the initial impulsive velocity change occurring at the departure phase of the Journey. Only applicable when "Journey departure type" is "launch or direct insertion," or "depart parking orbit."

Data Type	`double`
Allowed Values	$0 < \text{Real} < \infty$
Default Value	0.0 and 6.97
Units	km/s

2. **Journey departure type:** Set the departure type for the Journey. See Chapter 4 for discussion on departure type and departure class combinations. Reveals additional options depending on setting.

Data Type	`(DepartureType) enum`

Allowed Values	`0: launch or direct insertion,`
	`1: depart parking orbit,`
	`2: free direct departure,`
	`3: flyby`
	`4: flyby with fixed v-infinity-out,`
	`5: spiral out from circular orbit,`
	`6: zero-turn flyby (for small bodies)`

Default Value	`0: launch or direct insertion`

3. **Journey departure class:** Set the departure class for the Journey. See Chapter 4 for discussion on departure type and departure class combinations.

Data Type	`(BoundaryClass) enum`

Allowed Values	`0: Ephemeris-pegged,`
	`1: Free point,`
	`2: Ephemeris-referenced,`
	`3: Periapse`

Default Value	`0: Ephemeris-pegged`

4. **Forced initial coast (days):** Forces the spacecraft to coast from it's initial state for some

number of days before performing any maneuvers. Similar to "forced post-flyby coast duration" on the Global Mission Options tab (Section 6.1).

Data Type	`double`
Allowed Values	$0 <$ Real $< \infty$
Default Value	0
Units	days

5. **Orbital radius for beginning of escape spiral (km):** Revealed when "Journey departure type" is set to "spiral out from circular orbit." Sets the starting circular orbit radius to begin the low-thrust spiral escape from the departure body.

Data Type	`double`
Allowed Values	$0 <$ Real $< \infty$
Default Value	0
Units	km

6. **Orbital radius for end of escape spiral (km):** Revealed when "Journey departure type" is set to "spiral out from circular orbit." Sets the final circular orbit radius to finish the low-thrust spiral escape from the departure body.

Data Type	`double`
Allowed Values	$0 <$ Real $< \infty$
Default Value	0
Units	km

6.3.3 Journey Arrival Options

1. **Journey arrival type:** Set the arrival type for the Journey. See Chapter 4 for discussion on arrival type and arrival class combinations. Reveals additional options depending on setting.

Data Type	`(ArrivalType) enum`

Allowed Values	`0: insertion into parking orbit (use chemical Isp),`
	`1: rendezvous (with chemical maneuver),`
	`2: intercept with bounded V_infinity,`
	`3: rendezvous (no maneuver)`
	`4: match final v-infinity vector (with chemical maneuver),`
	`5: match final v-infinity vector (no maneuver),`
	`6: capture spiral,`
	`7: momentum transfer`

Default Value	`3: launch or direct insertion`

2. **Journey arrival class:** Set the arrival class for the Journey. See Chapter 4 for discussion on arrival type and arrival class combinations. Reveals additional options depending on setting.

Data Type	`(BoundaryClass) enum`
Allowed Values	`0: Ephemeris-pegged, 1: Free point,` `2: Ephemeris-referenced, 3: Periapse`
Default Value	`0: Ephemeris-pegged`

3. **Impact momentum enhancement factor (beta):** Set a scale factor that encompasses the plasticity of the impact, the crater formation, and the ejecta released by the impact for the special case where the spacecraft collides with the destination body and transfers momentum to it. See the EMTG Software Design Document, Section 13.5.2.7 for more information.

Data Type	`double`
Allowed Values	$0 < \text{Real} < \infty$
Default Value	1
Units	NA

4. **Orbital radius for beginning of capture spiral (km):** Revealed when "Journey arrival type" is set to "capture spiral." Sets the starting circular orbit radius to begin the low-thrust capture spiral at the arrival body.

Data Type	`double`
Allowed Values	$0 < \text{Real} < \infty$
Default Value	6678
Units	km

5. **Orbital radius for end of capture spiral (km):** Revealed when "Journey arrival type" is set to "capture spiral." Sets the final circular orbit radius to end the low-thrust capture spiral at the arrival body.

Data Type	`double`
Allowed Values	$0 < \text{Real} < \infty$
Default Value	6678
Units	km

6. **Journey-end delta-v (km/s):** Reduce the propellant mass of the spacecraft at the end of the Journey. In order to account for maneuvers that will be performed after arrival at the target body. This could be expressed as a fixed mass drop, but journey end delta-v allows the user to define a delta-v and apply it against the spacecraft mass at the end of the journey, using the appropriate spacecraft stage monopropellant system.

Data Type	`double`
Allowed Values	$0 < \text{Real} < \infty$
Default Value	0
Units	km/s

6.3.4 Perturbations and Other Options

1. **Override this journey's duty cycle:** Override the spacecraft duty cycle specified on the Spacecraft Options tab for this Journey only. Reveals the Journey duty cycle setting.

Data Type	`bool`
Allowed Values	true, false
Default Value	false
Units	NA

2. **Journey duty cycle:** Set the spacecraft duty cycle for this Journey only overriding the setting specified on the Spacecraft Options tab.

Data Type	`double`
Allowed Values	$0 < \text{Real} < \infty$
Default Value	1
Units	NA

3. **Flyby sequence:** Specify a sequence of bodies the spacecraft should flyby during this Journey modeled using patched conics. The integers listed should correspond to the body number in the "menu of bodies" section of the Universe file specified for this Journey.

Data Type	`std::vector<int, n>`
Allowed Values	Integers from the set of bodies in Universe file
Default Value	[ ]
Units	NA

4. **Enable powered flybys?:** Allow delta-v changes to occur during body flybys.

Data Type	`bool`
Allowed Values	true, false
Default Value	false
Units	NA

5. **Perturbation_bodies:** Select bodies by number in the Universe file for this Journey that will contribute to third body gravitational perturbing forces on the spacecraft. Revealed when the user selects "Enable third body" on the Physics Options tab (see Sec:6.5).

Data Type	`std::vector<int, n>`
Allowed Values	Integers from the set of bodies in Universe file
Default Value	[ ]
Units	NA

6. **Override journey's ephemeris output resolution?:** Override the forward propagated epehemeris resolution set in the Output Options tab. This setting has no effect if the "Generate forward-integrated ephemeris" setting is not active.

Data Type	`bool`
Allowed Values	true, false
Default Value	false
Units	NA

7. **Ephemeris output resolution (seconds):** Set the forward propagated ephemeris resolution for this Journey only.

Data Type	`double`
Allowed Values	1e-3 $<$ Real $<$ ∞
Default Value	1
Units	seconds

8. **Enable central body gravity harmonics?:** Activate spherical harmonics gravity perturbations for the central body of this Journey. This setting requires an appropriate propagator and Mission Type combination (see Chapter 2) and a gravity file for the central body with a `.grv` extension.

Data Type	`bool`
Allowed Values	true, false
Default Value	false
Units	NA

9. **Central body gravity data file:** Path to the gravity file for the central body of the Journey specifying the spherical harmonics information. See Section 2.1.3 for details on the expected structure of the `.grv` file.

Data Type	`string`
Default Value	"DoesNotExist.grv"

10. **Central body maximum degree:** Set the maximum degree for spherical harmonics perturbations. Information up to this degree must be specified in the gravity file.

Data Type	`int`
Allowed Values	0 $<$ Integer $<$ ∞
Default Value	2

11. **Central body maximum order:** Set the maximum order for spherical harmonics perturbations. Information up to this order must be specified in the gravity file.

Data Type	`int`
Allowed Values	0 $<$ Integer $<$ ∞
Default Value	0

12. **Enable aerodynamic drag?:** Enable aerodynamic drag acceleration modeling due to the central body during this Journey. Requires a `.emtg_densityopt` options file defining density

information. See Section 2.1.4 for details on the expected structure of the `.emtg_densityopt` file.

Data Type	`bool`
Allowed Values	true, false
Default Value	false
Units	NA

13. **Spacecraft drag area (m^2):** Set spacecraft area used to aerodynamic drag. Only active when "Enable aerodynamic drag" is selected.

Data Type	`double`
Allowed Values	$0 <$ Real $< \infty$
Default Value	70
Units	m^2

14. **Spacecraft drag coefficient:** Set spacecraft drag coefficient used to aerodynamic drag. Only active when "Enable aerodynamic drag" is selected.

Data Type	`double`
Allowed Values	$0 <$ Real $< \infty$
Default Value	2.2
Units	NA

15. **Atmospheric density model:** Select the atmospheric density model to use. Only Exponential is currently available. Only active when "Enable aerodynamic drag" is selected.

Data Type	`str`
Allowed Values	`Exponential`
Default Value	`Exponential`

16. **Atmospheric density model data file:** Specifies the atmospheric density model file, a `.emtg_densityopt` file containing the altitude versus density information used in drag calculations. See Section 2.1.4 for details on the expected structure of the `.emtg_densityopt` file. Only active when "Enable aerodynamic drag" is selected.

Data Type	`string`
Default Value	"DoesNotExist.emtg_densityopt"

6.3.5 Departure and Arrival Elements

Selecting Free-point on either the Departure or Arrival Class will reveal new options allowing the user to specify specific state elements expressed in cartesian, spherical, orbital elements, or b-plane representations. State elements may be fixed or allowed to vary within user-specified bounds.

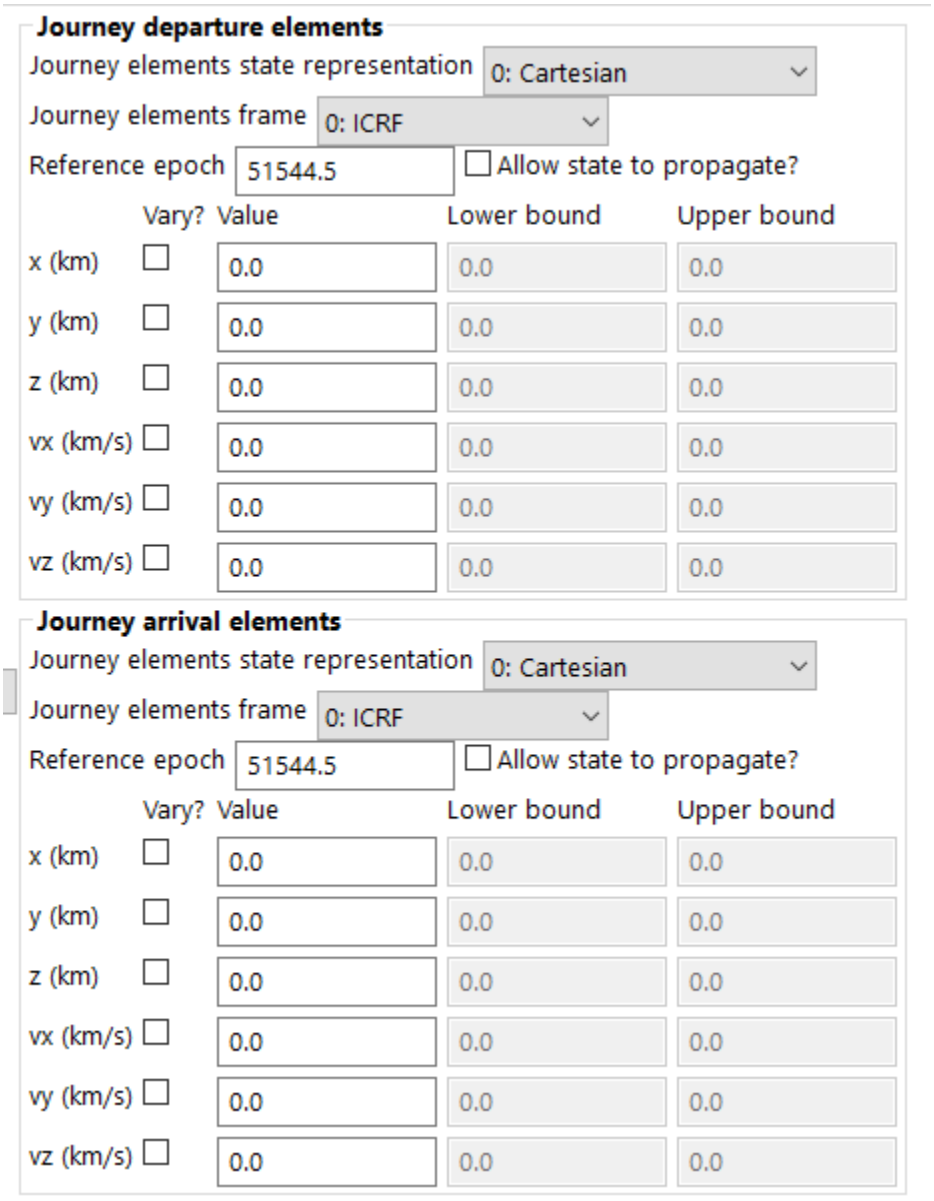


Figure 6.9: EMTG Journey Arrival and Depature Elements

1. **Journey elements state representation:** Select how the Journey Arrival/Departure state will be represented when the Boundary Class is set to Free Point.

Data Type	(StateRepresentation) enum
Allowed Values	0: Cartesian, 1: SphericalRADEC, 2: SphericalAZFPA, 3: COE, 4: MEE, 5: IncomingBplane, 6: OutgoingBplane, 7: IncomingBplaneRpTA 8: OutgoingBplaneRpTA,
Default Value	0: Cartesian,

2. **Journey elements frame:** Select the reference frame for the Journey Arrival/Departure state when the Boundary Class is set to Free Point.

Data Type	(ReferenceFrame) enum
Allowed Values	0: ICRF, 1: J2000_BCI, 2: J2000_BCF, 3: TrueOfDate_BCI 4: TrueOfDate_BCF, 5: PrincipleAxes, 6: Topocentric, 7: Polar 8: SAM 9: Object Referenced
Default Value	0: ICRF

3. **Reference epoch:** Epoch for the specified Journey arrival/departure state in MJD2000.

Data Type	double
Allowed Values	$0 <$ Real $< \infty$
Default Value	51544.5
Units	days since MJD2000 epoch

4. **Allow state to propagate?:** Selecting this option allows the state to propagate forward or backward from the specified epoch. Otherwise the Free point represents a fixed waypoint.

Data Type	bool
Allowed Values	true, false
Default Value	false
Units	NA

5. **Journey Arrival/Departure State:** Specify the six state components for the selected state representation as well as whether they can vary and their bounds. If the user chooses to fix any of these values, they are still variables but their bounds are ±1.0e-13, so the solver cannot move them.

Data Type	`double`
Allowed Values	$-\infty <$ Real $< \infty$
Default Value	0.0
Units	various

6.3.6 Journey Staging Options

1. **Stage after departure:** Stage the spacecraft according to the spacecraft configuration file immediately following departure. See Chapter 7.

Data Type	`bool`
Allowed Values	true, false
Default Value	false
Units	NA

2. **Stage before arrival:** Stage the spacecraft according to the spacecraft configuration file immediately before arrival. See Chapter 7.

Data Type	`bool`
Allowed Values	true, false
Default Value	false
Units	NA

3. **Stage after arrival:** Stage the spacecraft according to the spacecraft configuration file immediately following arrival. See Chapter 7.

Data Type	`bool`
Allowed Values	true, false
Default Value	false
Units	NA

6.4 Solver Options

The Solver Options tab contains settings defining exactly how a given problem should run. EMTG contains both a local gradient-based optimizer and a stochastic global search heuristic called (MBH), each with various options which may be modified as needed. Users may choose run the local optimizer and the stochastic search heuristic together, the local optimizer alone, or instead just evaluate a set of controls as an initial guess.

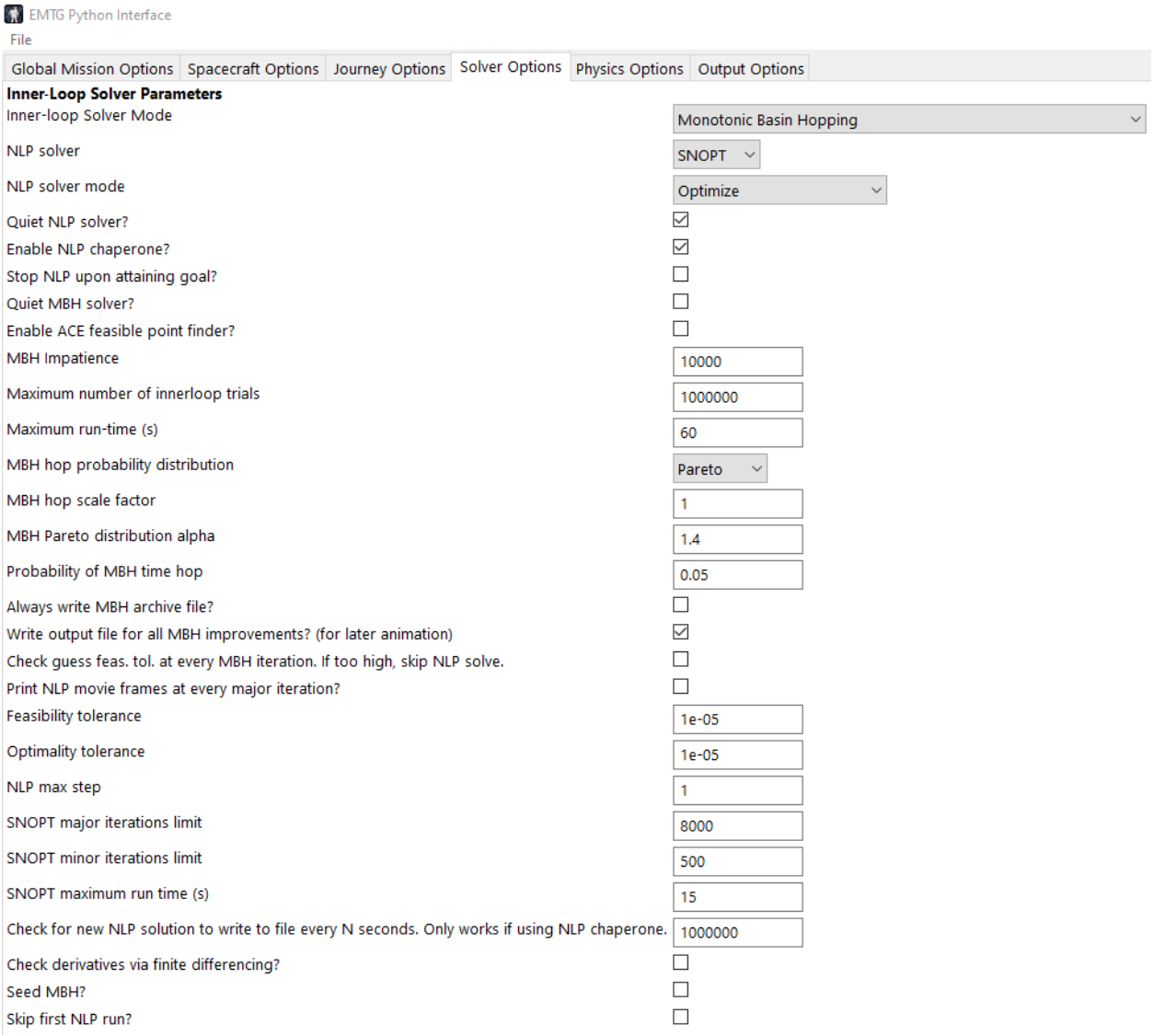


Figure 6.10: EMTG Solver Options tab

Users may choose which solver mode to use via the **Inner-loop Solver Mode** option:

EMTG Variable Name	`run_inner_loop`
Data Type	`(InnerLoopSolverType) enum`

Allowed Values	`0: None, evaluate trialX`
	`1: Monotonic Basin Hopping,`
	`2: Adaptive Constrained DE (NOT IMPLEMENTED),`
	`3: NLP with initial guess,`
	`4: Filament walker (experimental)`

Default Value	`1: Monotonic Basin Hopping`

Evaluate trialX:

Propagate an initial guess without solving. This will user either the Keplerian propagator or integrate the equations of motion, depending on the users chosen mission type and physics settings. Users must provide an initial guess as a `.emtg` solution file via the "Trial decision vector or initial guess" input option.

Nonlinear Program (NLP) with initial guess:

Leverages a gradient-based solver to solve a problem given an initial guess. Currently, the only available solver is the SNOPT. More detail is provided in 6.4.1. Users must provide an initial guess as a `.emtg` solution file via the "Trial decision vector or initial guess" input option.

Monotonic Basin Hopping:

This solver mode allows users to provide no initial guess and instead generate one with the MBH stochastic global search method. This runs in conjunction with the gradient-based solver to allow users to start with no initial guess and obtain a satisfied, locally optimized solution. In theory, if users allow MBH to run for extended periods of time, this will ensure solutions converge to globally or near-globally optimal solutions. More detail is provided in 6.4.2.

6.4.1 Gradient-Based Solver

The optimization problems in EMTG are formulated as Nonlinear Programming (NLP) problems. Thus, the optimizer solves a problem of the form:

$$\begin{aligned}
&\text{Minimize} f(\mathbf{x}) \\
&\text{Subject to:} \\
&\mathbf{x}_{lb} \leq \mathbf{x} \leq \mathbf{x}_{ub} \\
&\mathbf{c}(\mathbf{x}) \leq \mathbf{0} \\
&A \cdot \mathbf{x} \leq 0
\end{aligned}$$

where $\mathbf{x}_{lb}$ and $\mathbf{x}_{ub}$ define the lower and upper bounds on the decision (control) vector, $\mathbf{c}(\mathbf{x})$ is a vector of nonlinear constraint functions, and A is a matrix describing linear constraints.

Since most interplanetary trajectory optimization problems may consist of hundreds of variables and tens to hundreds of constraints, these problems are best solved with sparse NLP solvers. EMTG leverages the SNOPT to solve these problems. For detailed information on this optimizer beyond what is described here, see the SNOPT user guide at https://web.stanford.edu/group/SOL/guides/sndoc7.pdf.

SNOPT uses a sparse sequential quadratic programming (SQP) method and benefits greatly from partial derivatives with respect to the decision variables. EMTG is designed to provide analytical expressions for all necessary partial derivatives (objectives and constraints) which improves convergence. Users may modify the following options for the NLP solver:

1. **NLP solver:** Allows users to choose which of the available NLP solvers to use. Currently, only SNOPT may be used.

EMTG Variable Name	`NLP_solver_type`
Data Type	`int`
Allowed Values	`0: \ac{SNOPT},` `1: WORHP`
Default Value	`0: \ac{SNOPT}`

2. **NLP solver mode:** Sets the NLP solver to solve the problem in various ways, such as finding a solution which only satisfies equality constraints, finding a solution which is feasible, and finding a feasible solution and minimizing or maximizing the chosen objective.

EMTG Variable Name	`NLP_solver_mode`
Data Type	`(NLPMode) enum`

Allowed Values	`0: Feasible point,`
	`1: Optimize,`
	`2: Satisfy equality constraints`

Default Value	`1: Optimize`

3. **Quiet NLP solver?:** Sets whether the NLP solver should be quiet such that iterations are not displayed in the EMTG terminal.

EMTG Variable Name	`quiet_NLP`
Data Type	`bool`
Allowed Values	true, false
Default Value	true

4. **Enable NLP chaperone?:**Allows EMTG to recover the best solution found in a run of SNOPT even if SNOPT does not converge. It is recommended to turn this setting on in most cases.

EMTG Variable Name	`enable_NLP_chaperone`
Data Type	`bool`
Allowed Values	true, false
Default Value	true

5. **Stop NLP upon attaining goal?:** Option to force NLP solver to stop once a desired objective goal is satisfied, even if the optimizer may be able to minimize further.

EMTG Variable Name	`NLP_stop_on_goal_attain`
Data Type	`bool`
Allowed Values	true, false
Default Value	false

6. **NLP objective goal:** User specified numeric objective value used to define the "Stop NLP upon attaining goal" option criteria. This is set as an unscaled value. In PyEMTG, this option only appears after selecting the "Stop NLP upon attaining goal" option.

EMTG Variable Name	`NLP_objective_goal`
Data Type	`double`
Default Value	0
Units	various

7. **Print NLP movie frames at every major iteration?:** Option to print NLP information at each major iteration, including constraints descriptions and bounds. This will generate "NLP_Frame_#.csv" files containing the variable being constrained, the constraint lower and upper bounds, the scale factor for the constraint, and the actual value of the variable.

EMTG Variable Name	`print_NLP_movie_frames`
Data Type	`bool`
Allowed Values	true, false
Default Value	false

8. **Feasibility tolerance:** Tolerance for defining feasibility of a solution with respect to provided constraints. This is described in further detail in the SNOPT user guide.

EMTG Variable Name	`snopt_feasibility_tolerance`
Data Type	`double`
Allowed Values	$-\infty <$ Real $< \infty$
Default Value	1E-5
Units	various

9. **Optimality tolerance:** Tolerance for defining optimality of a solution. This is described in further detail in the SNOPT user guide.

EMTG Variable Name	`snopt_optimality_tolerance`
Data Type	`double`
Allowed Values	$-\infty <$ Real $< \infty$
Default Value	1E-5
Units	various

10. **NLP max step:** Sets the maximum step size during a linesearch in SNOPT.

EMTG Variable Name	`NLP_max_step`
Data Type	`double`
Allowed Values	$0 <$ Real $< \infty$
Default Value	1
Units	various

11. **SNOPT major iterations limit:** Sets the maximum number of major iterations SNOPT may perform. This defines how many times the quadratic subproblem will be solved when optimizing an objective.

EMTG Variable Name	`snopt_major_iterations`
Data Type	`size_t`
Allowed Values	$0 <$ Real $< \infty$
Default Value	8000

12. **SNOPT minor iterations limit:** Sets the maximum number of minor iterations SNOPT may perform. This defines the number of iterations allowed in solving each quadratic subproblem for each major iteration.

EMTG Variable Name	`snopt_minor_iterations`
Data Type	`size_t`
Allowed Values	$0 <$ Real $< \infty$
Default Value	500

13. **SNOPT maximum run time:** The maximum amount of time allowed for any SNOPT run.

EMTG Variable Name	`snopt_max_run_time`
Data Type	`int`
Allowed Values	$0 <$ Real $< \infty$
Default Value	15
Units	seconds

14. **Check for new NLP solution to write to file every N seconds. Only works if using NLP chaperone.:** When using the NLP chaperone, sets the number of seconds between checking for a new intermediate NLP solution. New intermediate NLP solutions are written to the file.

EMTG Variable Name	`print_NLP_movie_frames`
Data Type	`bool`
Allowed Values	true, false
Default Value	false

15. **Check derivatives via finite differencing?:** Option to tell SNOPT to check the derivatives by verifying both the individual gradients and the individual columns of the Jacobian are satisfactory.

EMTG Variable Name	`check_derivatives`
Data Type	`bool`
Allowed Values	true, false
Default Value	false

16. **Trial decision vector or initial guess:** Option to select a previous `.emtg` solution file as an initial guess for another run, even if some aspects of the problem have changed (e.g., increasing fidelity by changing the Mission Type, see Chapter 3). EMTG has built in capability to handle differences between the current `.emtgopt` file and the provided initial guess solution file.

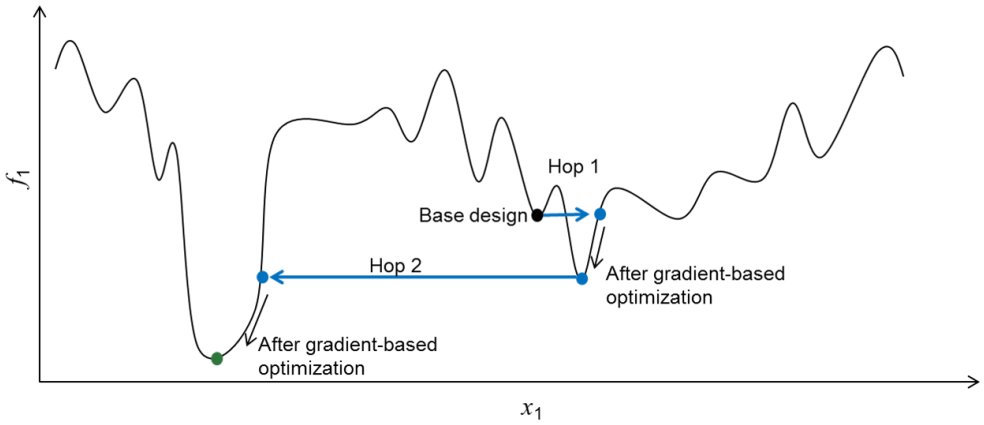


Figure 6.11: Monotonic Basin Hopping on a 1-dimensional function.

6.4.2 Monotonic Basin Hopping

(MBH) is a stochastic global search heuristic which allows EMTG to search for globally optimal solutions and run without an initial guess. This algorithm helps to find the best solutions in problems with many local optima. Often problems are structured such that individually local optimal "basins" cluster together, such that the distance in the decision space from one local optima to the next in a given cluster may be traversed in a short "hop". In EMTG, the "hop" is calculated and applied using the scaled decision variables rather than the dimensional decision variables. Additional details on MBH, including an algorithm detailing its process, may be found in in Section 15.3 of the EMTG Software Design Document. A diagram detailing how MBH may improve a solution for a problem with many local optima is shown in Figure 6.11.

The version of MBH in EMTG has two major parameters which a user has control over: the stopping criterion, and the type of random step used to generated the perturbed decision vector. The stopping criterion is defined by providing a maximum number of trial points to attempt (which directly translates to iterations) or a maximum CPU run time. The type of random step is defined by a probability distribution and various parameters which describe that distribution. Typically, a Pareto distribution described by some Pareto parameter α is used. MBH may be started from either a uniform-randomly chosen point in the decision space when not providing an initial guess or from an initial guess derived from a previous problem.

Users may modify the following options for MBH to suit their needs:

1. **Enable ACE feasible point finder?:** Allows MBH to compare infeasible solutions and search for minimum infeasibility before finding the first feasible solution. The most feasible of the infeasible solutions is then used as the next initial trial point before further hop perturbations are applied, which helps MBH continuously improve its solutions prior to obtaining a feasible solution. This runs prior to SNOPT if the initial feasibility is larger than the user specified constraint set for skipping the NLP solve, and after SNOPT every iteration until a point satisfying the SNOPT feasibility tolerance is satisfied.

EMTG Variable Name	`ACE_feasible_point_finder`
Data Type	`bool`
Allowed Values	true, false
Default Value	false

2. **MBH Impatience:** Defines the number of MBH perturbations to try near the current feasible point before obtaining a new random point. If this number of trials is reached, MBH returns a new random point before performing further perturbations.

EMTG Variable Name	`MBH_max_not_improve`
Data Type	`int`
Allowed Values	$0 <$ Real $< \infty$
Default Value	1E4

3. **MBH Impatience when skipping NLP Solve:** Defines the number of MBH perturbations to continue trying near the current feasible point before obtaining a new random point when skipping NLP solve due to not satisfying the guess feasibility tolerance requirement. If this number of trials is reached, MBH returns a new random point before performing further perturbations. Hidden until the "Check guess feas. tol." option is selected.

EMTG Variable Name	`MBH_max_not_improve_with_NLP_skip`
Data Type	`int`
Allowed Values	$0 <$ Real $< \infty$
Default Value	1E4

4. **Maximum number of innerloop trials:** Sets the maximum number of MBH trials to run before returning the best feasible point found.

EMTG Variable Name	`MBH_max_trials`
Data Type	`int`
Allowed Values	$0 <$ Real $< \infty$
Default Value	1E6

5. **Maximum run-time (s):** Sets the maximum CPU run time of MBH before returning the best feasible point found.

EMTG Variable Name	`MBH_max_run_time`
Data Type	`int`
Allowed Values	$0 <$ Real $< \infty$
Default Value	60

6. **MBH hop probability distribution:** Allows a user to choose between "Uniform", "Cauchy", "Pareto", or "Gaussian" probability distributions to randomly calculate the step size for a hop perturbation. These options have various additional parameters to help define the distribution.

EMTG Variable Name	`MBH_hop_distribution`
Data Type	`int`

Allowed Values	`0: Uniform,`
	`1: Cauchy,`
	`2: Pareto,`
	`3: Gaussian`

Default Value	`2: Pareto`

7. **MBH uniform hop ball size:** For the "Uniform" probability distribution, this scales the randomly generated perturbation on the decision variables, i.e. increases the size of the uniform "ball" used to generate the perturbations.

EMTG Variable Name	`MBH_max_step_size`
Data Type	`double`
Allowed Values	$0 <$ Real $< \infty$
Default Value	1

8. **MBH hop scale factor:** For the "Cauchy" and "Pareto" probability distributions, this scales the randomly generated perturbation on the decision variables.

EMTG Variable Name	`MBH_max_step_size`
Data Type	`double`
Allowed Values	$0 <$ Real $< \infty$
Default Value	1

9. **MBH Pareto distribution alpha:** For the "Pareto" probability distribution, sets the Pareto parameter α to define how the random step is drawn from the bi-directional Pareto distribution.

EMTG Variable Name	`MBH_Pareto_alpha`
Data Type	`double`
Allowed Values	$0 <$ Real $< \infty$
Default Value	1.4

10. **MBH hop standard deviation:** For the "Gaussian" probability distribution, this sets the standard deviation which defines the Gaussian distribution used to generate the perturbation for the decision variables.

EMTG Variable Name	`MBH_max_step_size`
Data Type	`double`
Allowed Values	$0 <$ Real $< \infty$
Default Value	1

11. **Probability of MBH time hop:** Special attention is given to decision variables that define time-of-flight between two boundary points in the MBH algorithm. These are the most significant variables defining an interplanetary trajectory, and thus it is sometimes necessary to significantly perturb them in order to successfully "hop" to a new cluster of solutions. This

is achieved via a low uniform-random probability which shifts the time-of-flight variables by ± 1 synodic period of the two boundary points.

EMTG Variable Name	`MBH_time_hop_probability`
Data Type	`double`
Allowed Values	$0 <$ Real $< \infty$
Default Value	0.05

12. **Always write MBH archive file?:** Option to choose whether to always save an archive file of the MBH runs. An archive file has extension `.emtg_archive` and stores basic information from the MBH runs, including the number of times the point was reset, the number of total iterations MBH took, the time stamp in seconds of when an iteration of MBH completed, and the objective function value of that iteration.

EMTG Variable Name	`MBH_always_write_archive`
Data Type	`bool`
Allowed Values	true, false
Default Value	false

13. **Write output file for all MBH improvements? (for later animation):** Option to print a "hop" data file for every MBH improvement, which can then be used to animate the results of MBH.

EMTG Variable Name	`MBH_write_every_improvement`
Data Type	`bool`
Allowed Values	true, false
Default Value	false

14. **Check guess feas. tol. at every MBH iteration. If too high, skip NLP solve.:** Option to have MBH check if a solution satisfies a user provided feasibility tolerance and skip the NLP solve call in an MBH trial run if the feasibility tolerance is not satisfied.

EMTG Variable Name	`checkFeasibilityTolInMBHToSkipNLP`
Data Type	`bool`
Allowed Values	true, false
Default Value	false

15. **Skip NLP solve if feas. tol. of initial guess is greater than this value. (MBH only.):** Tolerance used to determine if the MBH iteration returns a point that is acceptable enough to perform an NLP solve. If the feasibility is worse than this provided tolerance, the NLP solve is skipped and the next MBH iteration begins. Hidden until the "Check guess feas. tol." option is selected.

EMTG Variable Name	`feasibilityTolInMBHToSkipNLP`
Data Type	`double`
Allowed Values	$-\infty <$ Real $< \infty$
Default Value	1E6

6.5 Physics Options

The Physics Options tab contains settings which control gravitational and SRP effects on the spacecraft including how EMTG computes the ephemerides for bodies in the mission Universe. Settings for the EMTG integrator (if used) are also available on this tab. These options effect EMTG's accuracy, run time, and memory requirements. Additional settings for how EMTG represents states for periapse boundaries and parallel shooting constraints and decision variables are also controlled from this tab.

When changing settings on this tab, keep in mind that certain options only take effect when the appropriate mission type is selected for each Journey. This section and Chapters 2 and 3 should be referenced together when selecting appropriate settings for the mission design.

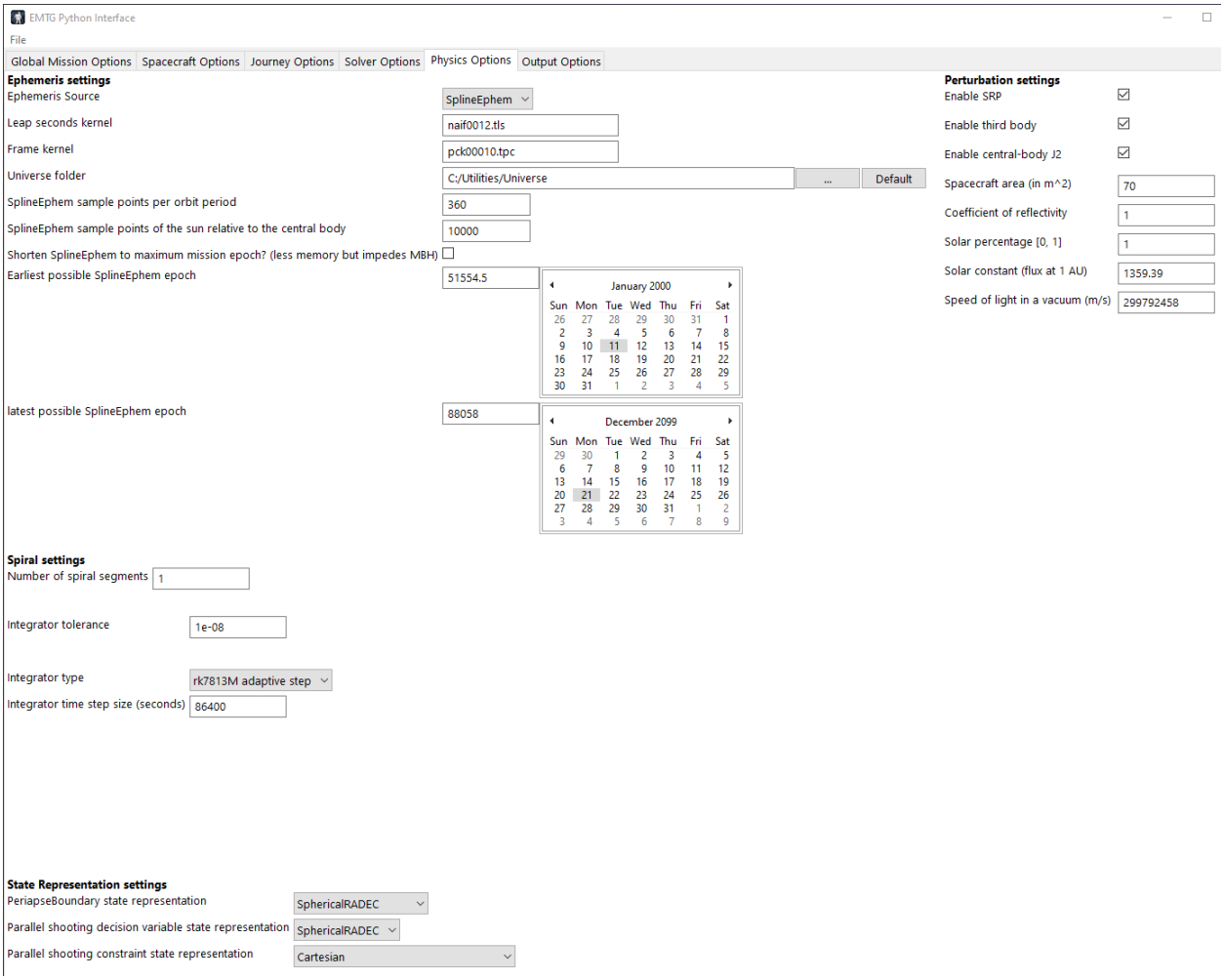


Figure 6.12: EMTG Physics Options tab

6.5.1 Ephemeris Settings

The Ephemeris settings control how EMTG uses SPICE to generate ephemeris states for Universe file bodies. For speed reasons EMTG by default does not call the SPICE library directly each time it needs a body state, but instead builds a set of ephemeris interpolations using cubic spline functions.

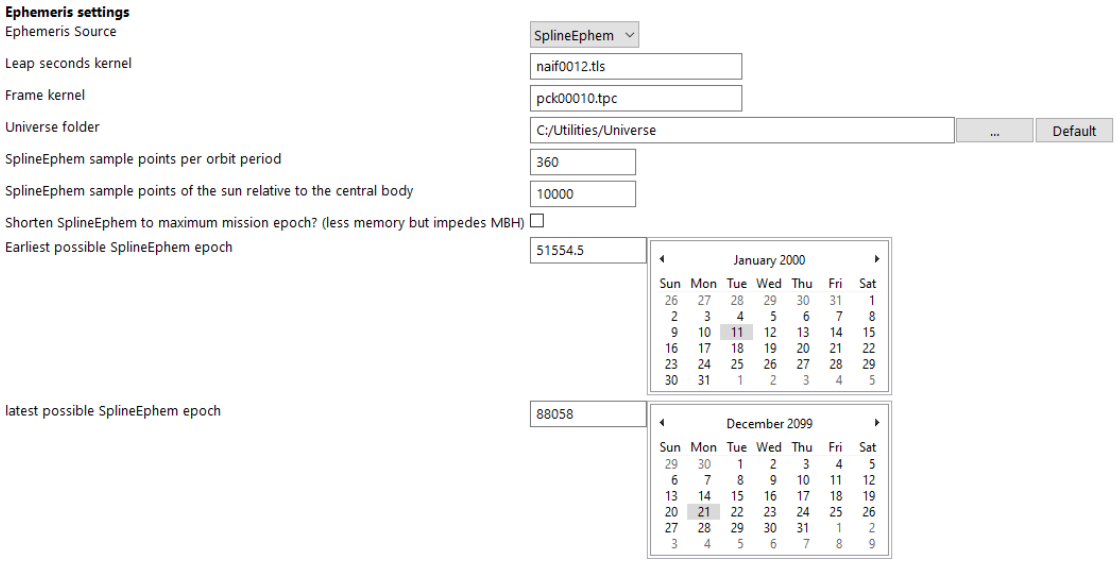


Figure 6.13: EMTG Physics Options - Ephemeris Settings

1. **Ephemeris Source:** Select the method EMTG uses to get body ephemeris. `SplineEphem` is the fastest method relying on cubic spline functions. It requires additional memory than other methods but also provides analytical partial derivatives. `\ac{SPICE}` calls the CSPICE library each time ephemeris information is required. This method is slower and relies on finite differencing for derivatives but uses less memory. It is also slightly more accurate.

Data Type	`int`
Allowed Values	0: `Static`, 1: `\ac{SPICE}`, 2: `SplineEphem`
Default Value	2: `SplineEphem`

2. **Leap seconds kernel:** Leap seconds kernel file to use. The file should be located in the Universe folder specified using the setting below. Leap second kernels can be found on the JPL NAIF website.

Data Type	`string`
Default Value	"naif0012.tls"

3. **Frame kernel:** Frame kernel file to use. The file should be located in the Universe folder specified using the setting below. Frame kernels can be found on the JPL NAIF website.

Data Type	`string`
Default Value	"pck00010.tpc"

4. **SplineEphem sample points per orbit period:** This setting controls the number of points drawn from SPICE per period of the central body which allows the user control the accuracy and memory footprint of the spline.

Data Type	`int`
Allowed Values	$1 <$ Integer $< \infty$
Default Value	360
Units	NA

5. **SplineEphem sample points of the sun relative to the central body:**This setting controls the number of points drawn from SPICE for the sun relative to the central body.

Data Type	`int`
Allowed Values	$1 <$ Integer $< \infty$
Default Value	360
Units	NA

6. **Shorten SplineEphem to maximum mission epoch?:** Sets the upper bound on the SPICE ephemeris splines to the global flight time bounds set in the Global Mission Options tab see Section 6.1. Reduces EMTG's memory requirements. The lower bound is decreased by ten days and the upper bound is increased by ten days to ensure the splines are well-defined at the global flight time bounds.

Data Type	`bool`
Allowed Values	true, false
Default Value	false
Units	NA

7. **Earliest/Latest possible SplineEphem epoch:** Sets epoch bounds on the SPICE ephemeris splines when `SplineEphem` is used as the ephemeris source. Reduces EMTG's memory requirements. The lower bound is decreased by ten days and the upper bound is increased by ten days to ensure the splines are well-defined at the epoch bounds.

Data Type	`double`
Allowed Values	33251 (1 DEC 1949) $<$ Real $<$ 100000 (31 AUG 2132)
Default Value	51554.5 (11 JAN 2000) (Earliest) and 88058 (21 DEC 2099) (Latest)
Units	days

6.5.2 Spiral Settings

1. **Number of spiral segments:** Sets the number of spiral segments when using Journey departure type, "5: Spiral-out from circular orbit" or Journey arrival type "6: capture spiral."

EMTG uses Edelbaum's spiral approximation to approximate a many-revolution low-thrust spiral about a body. See sections 4.3.7 and 4.2.7 for more information about Journey spiral arrival and departure.

Data Type	`int`
Allowed Values	$1 <$ Integer $< \infty$
Default Value	1
Units	NA

6.5.3 Integrator Settings

This section contains the settings for EMTG's numerical integrators. Note that not all mission types use numerical integration for propagation of the spacecraft. If using Keplerian propagation, the integration options set here will have no effect. Refer to chapter 3 for a discussion of the different Mission Types and propagation methods they use.

1. **Integrator tolerance:** Sets the maximum allowable error in the numerical solution of the ordinary differential equation (ODE)

Data Type	`double`
Allowed Values	1.0E-12 $<$ Real $<$ 1
Default Value	1.0E-8
Units	NA

2. **Integrator type:** Select the integrator type. Currently limited to a Runge-Kutta 8th order fixed step size method. Additional integrator methods may be implemented in the future and optionally selected here.

Data Type	`(IntegratorType) int`
Allowed Values	`0: rk7813M adaptive step, 1: rk8 fixed step`
Default Value	1

3. **Integrator time step size (seconds):**

Data Type	`double`
Allowed Values	1.0E-10 $<$ Real $< \infty$
Default Value	86400
Units	seconds

6.5.4 State Representation Settings

This section allows the user additional control over EMTG's internal state representation. Typically EMTG's default state representation is cartesian. However for periapse boundary states and parallel shooting decision variables, the default state is spherical using right ascension and declination.

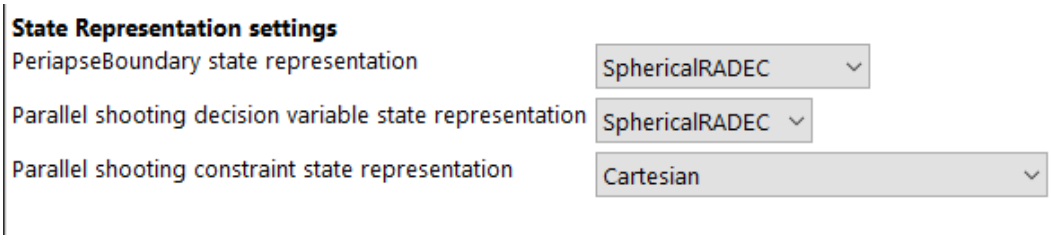


Figure 6.14: EMTG Physics Options - State Representation Settings

1. **PeriapseBoundary state representation:** Allows the user to change EMTG's internal state representation for the `PeriapseBoundary` class. See Section 4.1.4

Data Type	`(StateRepresentation) int`
Allowed Values	`0: Cartesian, 1: SphericalRADEC, 2: SphericalAZFPA, 3: COE, 4: MEE, 5: IncomingBplane, 6: OutgoingBplane, 7: IncomingBplaneRpTA, 8: OutgoingBplaneRpTA`
Default Value	1

2. **Parallel shooting decision variable state representation:** Allows the user to change EMTG's internal state representation for parallel shooting mission type decision variables. See Sections 3.6 and 3.5 for more information on parallel shooting.

Data Type	`(StateRepresentation) int`
Allowed Values	`0: Cartesian, 1: SphericalRADEC, 2: SphericalAZFPA, 3: COE, 4: MEE`
Default Value	1

3. **Parallel shooting constraint representation:** Allows the user to change EMTG's constraint representation for parallel shooting mission type. See Section 3.6 for more information on parallel shooting.

Data Type	`(ConstraintStateRepresentation) int`
Allowed Values	`0: Cartesian, 1: Same as encoded state representation`
Default Value	0

6.5.5 Perturbation Settings

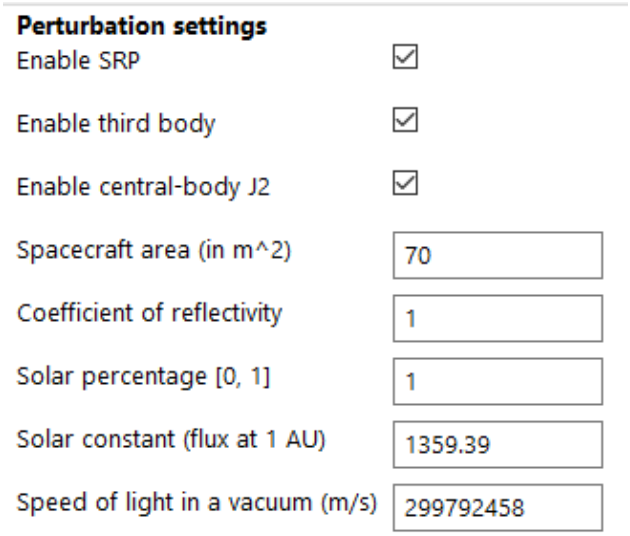


Figure 6.15: EMTG Physics Options - Perturbation Settings

1. **Enable SRP:** Activate EMTG's spherical/cannonball solar radiation pressure (SRP) model. Selecting this option will reveal additional configurable settings for EMTG's SRP model: spacecraft area, coefficient of reflectivity, solar percentage, solar constant, and speed of light in a vacuum.

Data Type	`bool`
Allowed Values	true, false
Default Value	false
Units	NA

2. **Enable third body:** Activate third body gravitational perturbations. Selecting this option reveals a "Perturbation bodies" setting on the Journey Options tab allowing the user to select which bodies will contribute to third body accelerations on a per-Journey basis. Selecting this option turns on third body perturbations for all Journeys whose mission type and propagator supports perturbations, but only those bodies listed on the Journey page will contribute. See Chapter 3 for more information.

Data Type	`bool`
Allowed Values	true, false
Default Value	false
Units	NA

3. **Enable central-body J2:** Activate J2 spherical harmonic gravitational forces for the central body of each Journey. For this force to be included, the Universe file for the central body must contain a value for J2 and a J2 reference radius (units of km) in the central body section of the Universe file. Like third body perturbations, central-body J2 activates J2 effects for

all Journeys whose mission type and propagator supports perturbations. See Chapter 3 for more information.

Data Type	`bool`
Allowed Values	true, false
Default Value	false
Units	NA

4. **Spacecraft area (in m^2):** Set spacecraft cross-sectional area used to compute SRP effects. Only active when "enable SRP" is selected.

Data Type	`double`
Allowed Values	$0 < \text{Real} < \infty$
Default Value	70
Units	m^2

5. **Coefficient of reflectivity:** Set spacecraft coefficient of reflectivity used to compute SRP effects. Only active when "Enable SRP" is selected.

Data Type	`double`
Allowed Values	$0 < \text{Real} < \infty$
Default Value	1
Units	NA

6. **Solar percentage [0, 1]:** Set solar percentage used to compute SRP effects. This setting combined with "solar constant" can be used to examine the effects of solar activity levels on the mission design. Only active when "Enable SRP" is selected.

Data Type	`double`
Allowed Values	$0 < \text{Real} < 1$
Default Value	1
Units	NA

7. **Solar constant (flux at 1 AU):** Set solar flux at 1 AU used to compute SRP effects. This setting combined with "solar percentage" can be used to examine the effects of solar activity levels on the mission design. Only active when "Enable SRP" is selected.

Data Type	`double`
Allowed Values	$0 < \text{Real} < \infty$
Default Value	1359.39
Units	W/m^2

8. **Speed of light in a vacuum (m/s):** Set speed of light in a vacuum. This is normally left at it's default value.

Data Type	`double`
Allowed Values	$0 < \text{Real} < \infty$
Default Value	299792458
Units	m/s

6.6 Output Options

The Options tab specifies the working directory, frame, configuration options for the `.emtg` solution file produced when EMTG is run and other miscellaneous settings. This tab can also activate a number of additional file outputs which can be useful during the mission planning process. These include an integrated ephemeris file for the spacecraft which can be converted into a SPICE SPK (Spacecraft and Planetary Kernel) file as well as maneuver and target specification files providing more detail about each of the maneuvers in the EMTG solution. This section covers the available options for EMTG's output files.

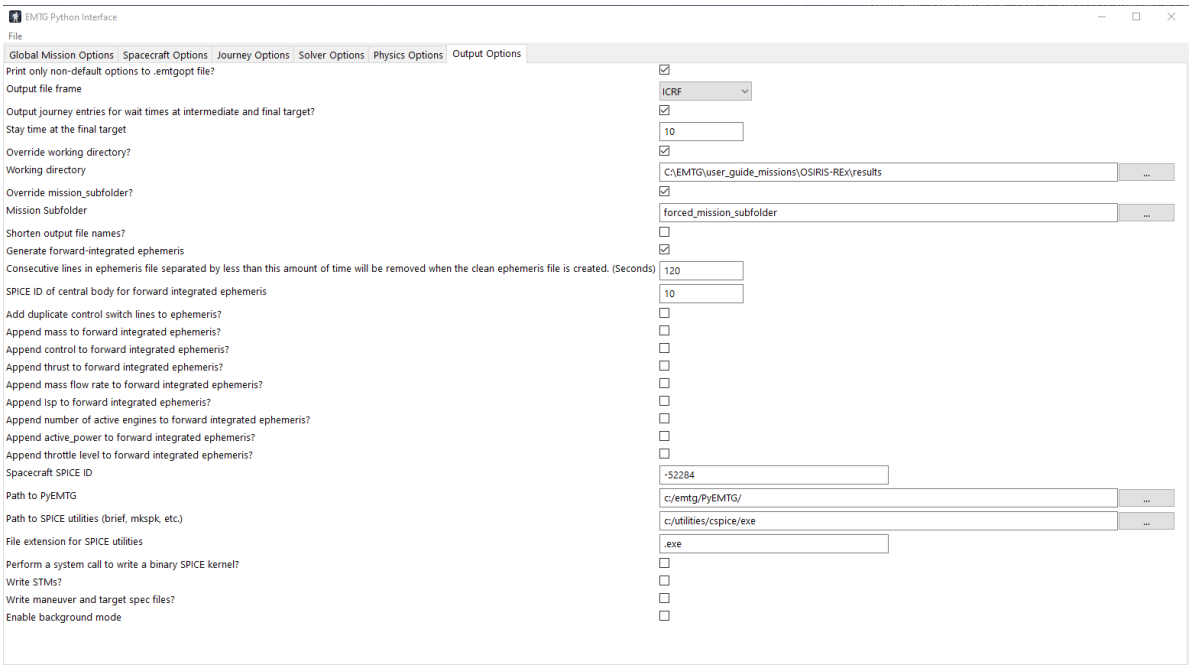


Figure 6.16: PyEMTG Output Options

6.6.1 General Output Options

1. **Print only non-default options to .emtgopt file?:** Selecting this option will cause PyEMTG to only include options in the EMTG Options file that the user has changed from their default values. This can make it easier to review the options file, but only if the user is already familiar with EMTG's default behavior. It is recommended to leave this box unchecked so the full set of options may be easily viewed by users, as some options are only present in the EMTG Options file and not in PyEMTG.

Data Type	`bool`
Allowed Values	true, false
Default Value	false
Units	NA

2. **Output file frame:** Specify the desired reference frame to be used in the EMTG output files. Note that EMTG's basic trajectory plotting tool uses ICRF. If another output frame is selected the trajectory plot will not be accurate.

Data Type	`(ReferenceFrame) enum`

Allowed Values	`ICRF,`
	`J2000_BCI,`
	`J2000_BCF,`
	`TrueOfDate_BCI,`
	`TrueOfDate_BCF,`
	`PrincipleAxes,`
	`Topocentric,`
	`Polar,`
	`SAM,`
	`ObjectReferenced`

Default Value	`ICRF`

3. **Output journey entries for wait times at intermediate and final target?:** Selecting this option will cause EMTG to include entries in the output `.emtg` file depicting the spacecraft waiting at an ephemeris body prior to departure as in Figure 6.17. Normally these entries are skipped and there will be a gap between the spacecraft's arrival at an ephemeris body and its departure at the start of the next Journey. Selecting this option also activates and reveals an option to stay at the final target ephemeris body.

Data Type	`bool`
Allowed Values	true, false
Default Value	false
Units	NA

Journey: 1
Journey name: Bennu_to_Earth
Central Body: Sun
Radius (km): 695700.00000000000000000000
mu (km^3/s^2): 132712440040.94459533691406250000
Characteristic length unit (km): 149597870.69100001454353332520
Frame: ICRF

#	JulianDate (ET/TDB)	MM/DD/YYYY	event type	location	step size (days)	altitude (km)	BdotR (km)
46	2458844.72499689	12/27/2019	waiting	Bennu	0.00000000	-	-
47	2458881.24999689	2/1/2020	waiting	Bennu	0.00000000	-	-
48	2458917.77499689	3/9/2020	waiting	Bennu	0.00000000	-	-
49	2458954.29999689	4/14/2020	waiting	Bennu	0.00000000	-	-
50	2458990.82499689	5/21/2020	waiting	Bennu	0.00000000	-	-
51	2459027.34999689	6/26/2020	waiting	Bennu	0.00000000	-	-
52	2459063.87499689	8/2/2020	waiting	Bennu	0.00000000	-	-
53	2459100.39999689	9/7/2020	waiting	Bennu	0.00000000	-	-
54	2459136.92499689	10/14/2020	waiting	Bennu	0.00000000	-	-
55	2459173.44999689	11/19/2020	waiting	Bennu	0.00000000	-	-
56	2459209.97499689	12/26/2020	waiting	Bennu	0.00000000	-	-
57	2459246.49999689	1/31/2021	waiting	Bennu	0.00000000	-	-
58	2459283.02499689	3/9/2021	waiting	Bennu	0.00000000	-	-
59	2459319.54999689	4/15/2021	waiting	Bennu	0.00000000	-	-
60	2459356.07499689	5/21/2021	waiting	Bennu	0.00000000	-	-
61	2459392.59999689	6/27/2021	waiting	Bennu	0.00000000	-	-
62	2459429.12499689	8/2/2021	waiting	Bennu	0.00000000	-	-
63	2459465.64999689	9/8/2021	waiting	Bennu	0.00000000	-	-
64	2459502.17499689	10/14/2021	waiting	Bennu	0.00000000	-	-
65	2459538.69999689	11/20/2021	waiting	Bennu	0.00000000	-	-
66	2459575.22499689	12/26/2021	departure	Bennu	0.00000000	-	-
67	2459586.69658445	1/7/2022	coast	deep-space	22.94317511	-	-
68	2459609.63975956	1/30/2022	coast	deep-space	22.94317511	-	-

Figure 6.17: Journey Wait Entries

4. **Stay time at the final target:** This setting causes EMTG to add Journey entries in the output `.emtg` file at the final mission target for the specified number of days. Only active when the "Output journey entries for wait times" option is selected.

Data Type	`double`
Allowed Values	$0 < \text{Real} < \infty$
Default Value	0.0
Units	days

5. **Override working directory:** By default when EMTG is run it will create a new run directory (also known as the mission subfolder) in the `EMTG_v9_results` folder in the EMTG install location named using the mission name in the Global Options tab followed by the current date and time. This behavior can be changed by selecting this option and specifying the output directory in the "Working directory" option activated and revealed by this setting. The provided path may be either relative to the PyEMTG executable or absolute. It is recommended to always use absolute paths.

Data Type	`bool`
Allowed Values	true, false
Default Value	false
Units	NA

6. **Working directory:** Specifies the path to the new working directory.

Data Type	`string`
Default Value	"..//EMTG_v9_Results"

7. **Override mission subfolder?:** This option changes the default run directory naming convention of the mission name followed by the current date and time. This option also activates and reveals the "Mission subfolder" option allowing the user to specify a custom directory name which will be created at runtime in the working directory specified earlier.

Data Type	`bool`
Allowed Values	true, false
Default Value	false
Units	NA

8. **Mission subfolder:** Specifies the new mission subfolder (run directory) name. This subfolder is created within the working directory. If there are any pre-existing files of the same name, they will be overwritten.

Data Type	`string`
Default Value	"forced_mission_subfolder"

9. **Shorten output file names?:** Removes additional Journey information from the name of the `.emtg` output file and uses just the mission name as the filename.

Data Type	`bool`
Allowed Values	true, false
Default Value	true
Units	NA

10. **Write STMs?:** Creates state transition matrix `.stm` output files for every Journey Phase along with a set of `.stm_description` files containing the row and column variable names.

Data Type	`bool`
Allowed Values	true, false
Default Value	false
Units	NA

11. **Write maneuver and target spec files?:** These files provide detailed information for each maneuver in the mission and can be used when converting an EMTG trajectory to a final, flight-ready trajectory. The Maneuver Specification file lists the epoch, thrust direction, margin, duration, DV, and other information about each maneuver. The Target Specification file stores the epoch, frame, central body, and state vector (position, velocity, mass), and B-plane coordinates (if applicable) associated with the target for each maneuver.

Data Type	`bool`
Allowed Values	true, false
Default Value	false
Units	NA

12. **Enable background mode:** Disables almost all terminal output when running EMTG.

Data Type	`bool`
Allowed Values	true, false
Default Value	false
Units	NA

6.6.2 Ephemeris Output Options

EMTG can produce a forward integrated Ephemeris Table or timeseries of the spacecraft's position, velocity and other information as a text file. On the Output Options tab in PyEMTG select "Generate forward-integrated ephemeris." This will reveal additional options to configure the ephemeris table output. In addition to spacecraft position and velocity which are written by default, users may select to show spacecraft mass, control, thrust, mass flow rate, ISP, and other engine parameters.

The forward-integrated ephemeris can be used to create a binary SPICE kernel `.bsp` file for use with other software through the use of a short Python script `bspwriter.py` which is automatically created in the output directory when the "Generate forward-integrated ephemeris option" is active. The `bspwriter.py` script also "cleans" the ephemeris removing duplicate entries and enforcing the timestep resolution set by the "Consecutive lines in ephemeris file separated by less than..." option described below.

Forward-integrated Ephemeris Issue

If the EMTG mission (final Journey) ends with an impulsive maneuver such as a capture burn at a body, the final line of the ephemeris may not contain the impulsive delta-v. Instead the ephemeris file will show two repeated entries at the final epoch of the mission. The second of these entries is intended to contain the spacecraft state immediately following the final impulsive delta-v. This typically would only be an issue at initial stages of mission design prior to the conversion to finite burns. If desired, the final impulsive maneuver can be made to appear by adding another "fake" Journey to the end of the mission.

```
1464   2027 Mar 02  05:59:18.882572, -1136.2659251982, 2211.4557214533, 5222.1510172554, -3.4783995790688, 7.8291861477086, 1.5903876795102
1465   2027 Mar 02  06:00:00.000000, -1279.0752737045, 2532.953630776, 5286.5974263935, -3.4680006680451, 7.808763002488, 1.5450997098253
1466   2027 Mar 02  06:00:00.000000, -1279.0752736804, 2532.9536307293, 5286.5974263992, -3.4680006680466, 7.8087630024874, 1.5450997098366
1467
```

Figure 6.18: Forward-integrated Ephemeris Final DV Issue

The example Journey options below shows one configuration of an additional coast Journey lasting 0.01 days which can be added as a final Journey to trigger the post-impulsive delta-v state to appear in the forward-integrated ephemeris. This method should be used with the "evaluate trialX" solver mode rather than the NLP or MBH modes to avoid the Journey impacting the overall mission or solution feasibility.

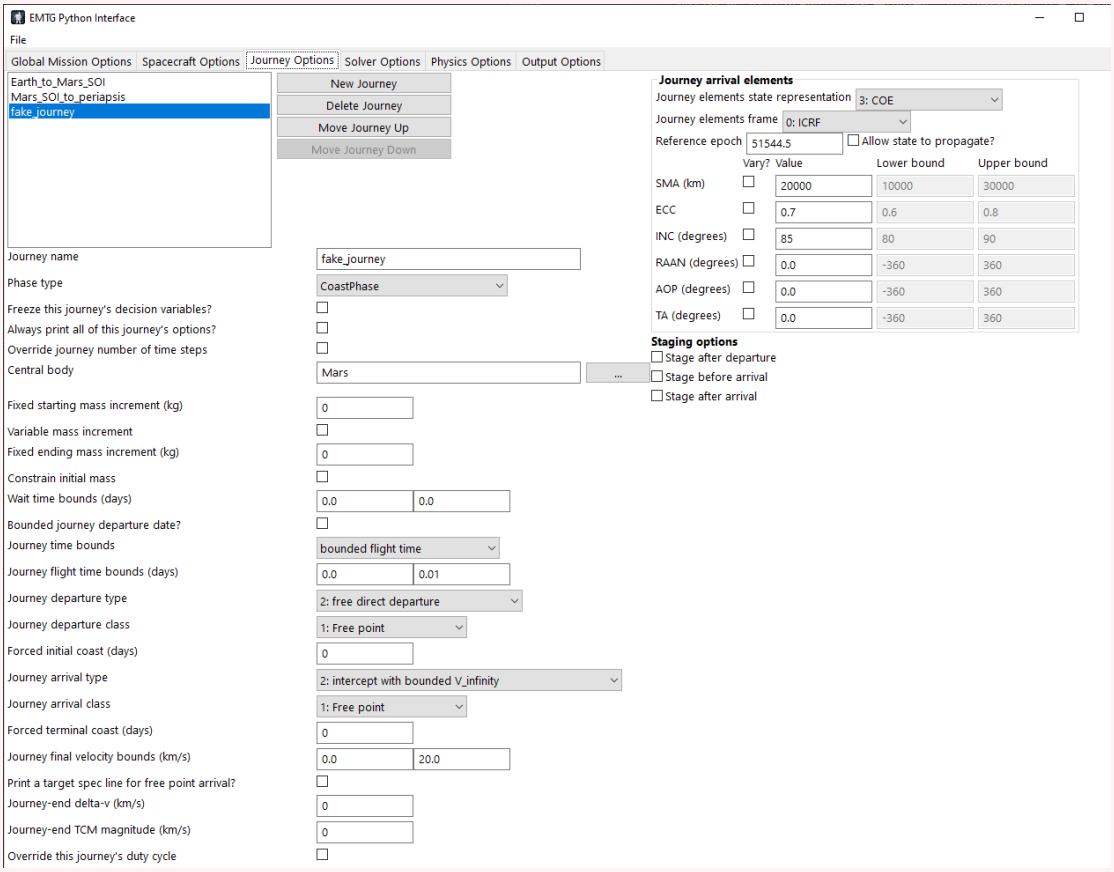


Figure 6.19: Example Extra Journey Configuration

1. **Generate forward-integrated ephemeris:** Turns on creation of the forward-integrated ephemeris file in the working directory specified above. Reveals the additional options below.

Data Type	`bool`
Allowed Values	true, false
Default Value	false
Units	NA

2. **Consecutive lines in ephemeris file separated by less than this amount of time will be removed:** Set the timestep resolution in the forward-integrated ephemeris file. Can be overridden at the Journey-level see Section 6.3.

Data Type	`double`
Allowed Values	1.0E-5 $<$ Real $< \infty$
Default Value	120
Units	seconds

3. **SPICE ID of central body for forward integrated ephemeris:** The forward-integrated ephemeris can be output using any body in the Universe as a central body by setting this option to the appropriate SPICE ID. The body must be present in one of the SPICE kernels provided in the `ephemeris_files` directory located with the mission Universe files.

Data Type	`integer`
Allowed Values	$-\infty <$ Real $< \infty$
Default Value	10
Units	NA

4. **Add duplicate control switch lines to ephemeris?:** This option was used prior to the maneuver spec files and will be removed in a future release.

5. **Append mass to forward integrated epehemeris?:** Add spacecraft mass in kilograms as an additional column in the forward-integrated ephemeris.

Data Type	`bool`
Allowed Values	true, false
Default Value	false
Units	NA

6. **Append control to forward integrated epehemris?:** Add thrust direction unit vector as additional columns in the forward-integrated ephemeris.

Data Type	`bool`
Allowed Values	true, false
Default Value	false
Units	NA

7. **Append thrust to forward integrated epehemris?:** Add thrust in Newtons as an additional column in the forward-integrated ephemeris.

Data Type	`bool`
Allowed Values	true, false
Default Value	false
Units	NA

8. **Append mass flow rate to forward integrated epehemeris?:** Add mass flow rate in kilograms/second as an additional column in the forward-integrated ephemeris.

Data Type	`bool`
Allowed Values	true, false
Default Value	false
Units	NA

9. **Append Isp to forward integrated ephemeris?:** Add Isp in seconds as an additional column in the forward-integrated ephemeris.

Data Type	`bool`
Allowed Values	true, false
Default Value	false
Units	NA

10. **Append number of active engines to forward integrated ephemris?:** Add number of active engines as an additional column in the forward-integrated ephemeris.

Data Type	`bool`
Allowed Values	true, false
Default Value	false
Units	NA

11. **Append active_power to forward integrated ephemeris?:** Add active spacecraft power in kiloWatts as an additional column in the forward-integrated ephemeris.

Data Type	`bool`
Allowed Values	true, false
Default Value	false
Units	NA

12. **Append throttle level to forward integrated ephemeris?:** Add throttle level as an additional column in the forward-integrated ephemeris.

Data Type	`bool`
Allowed Values	true, false
Default Value	false
Units	NA

13. **Spacecraft SPICE ID:** SPICE ID to be used when creating a binary SPICE kernel. It's recommended to use a negative integer. By convention, spacecraft SPICE IDs are negative integer values to avoid ID collision with a body. EMTG does not enforce this requirement, but other software which reads SPICE kernels may.

Data Type	`integer`
Allowed Values	$-\infty <$ Real $< \infty$
Default Value	-52284
Units	NA

14. **Path to PyEMTG:** Path to the directory containing the `PyEMTG.py` file.

Data Type	`string`
Default Value	"c:/emtg/PyEMTG/"

15. **Path to SPICE utilities:** Path to the directory containing the SPICE executable utilities.

Data Type	`string`
Default Value	"c:/utilities/cspice/exe"

16. **File extension for SPICE utilities:** File extension for the SPICE executable utilities.

Data Type	`string`
Default Value	".exe"

17. **Perform a system call to write a binary SPICE kernel?:** When forward-integrated epehemeris is generated a short Python script, `bspwriter.py`, is also created in the output directory which when executed will create the SPICE kernel `.bsp` file and a cleaned copy of the `.ephemeris` file. Selecting this option will automatically execute the `bspwriter.py` script. Alternatively users may execute the script themselves.

Data Type	`bool`
Allowed Values	true, false
Default Value	false
Units	NA

Chapter 7

Configuration Files

This chapter describes how configuration files may be used to model performance characteristics of the spacecraft and launch vehicles used when obtaining solutions. While vehicle characteristics may be set in the PyEMTG GUI, configuration files are the most flexible option for defining vehicle characteristics. Additionally, these files may be shared across missions, allowing users to define configuration files for specific vehicles that may be consistently re-used, such as a particular launch vehicle. These files are used exclusively for options *0: Assemble from libraries* and *1: Read .emtg_spacecraftoptions file* in the Spacecraft Option tab's "Spacecraft model type" field in PyEMTG, or by respectively setting the `SpacecraftModelInput` value in the `.emtgopt` file.

7.1 Launch Vehicle Configuration

The Launch Vehicle Options file (extension: `.emtg_launchvehicleopt`) is used to specify the capabilities of one or more launch vehicles. These options are used to calculate the delivered mass of a launch vehicle for a given input hyperbolic excess energy (C_3), as well as the derivative of the delivered mass with respect to a changing C_3. In EMTG, Launch Vehicle Options files are relatively simple. In one line, they specify the launch vehicle name, the Declination of Launch Asymptote (DLA) and C_3 (Hyperbolic Excess Energy) bounds, the launch vehicle payload Adapter Mass, and the polynomial coefficients for injected mass as a function of C_3. The format is shown in Table 7.1.

The C_3 coefficients are used to model the capability of the launch vehicle within the C_3 bounds set by values 5 and 6 in the launch vehicle line. The DLA and C_3 bounds set define the range over which the coefficients are valid. Typically the C_3 coefficients are set for polynomials up to fifth order ($n = 5$). Data for polynomial fits may be obtained from NASA's Launch Services Program Launch Vehicle Performance Website: https://elvperf.ksc.nasa.gov/Pages/Default.aspx

Index	Variable Name	Data Type
1	Name	String
2	ModelType	Integer (0: polynomial, only type currently available)
3	DLA_Lowerbound	Real (deg)
4	DLA_Upperbound	Real (deg)
5	C3Lowerbound	Real (km^2/s^2)
6	C3Upperbound	Real (km^2/s^2)
7	AdapterMass	Real (kg)
8-¿end of line	C3Coefficient	Real (C_3 is in km^2/s^2)

Table 7.1: Launch Vehicle Variables

The injected mass is governed by the following equation:

$$m(C_3) = (1 - \eta_{LV}) \sum_{i=0}^{n} \left(c_i \cdot C_3^i - m_{adapter} \right) \tag{7.1}$$

Where η_{LV} is the launch vehicle margin, c_i is the ith C_3 Coefficient, and $m_{adapter}$ is the launch vehicle adapter mass.

Launch Vehicle Configuration Issue

Payload adapter mass calculation may not behave as expected when modifying the adapter mass for a solution where the obtained C_3 is against the provided upper bound. To workaround this issue in this scenario, in PyEMTG use the "Fixed starting mass increment (kg)" option for Journey 1 and set it to the negative value of the desired adapter mass, ensuring that the adapter mass set in the `.emtg_launchvehicleopt` file is set to zero.

An example `.emtg_launchvehicleopt` file is shown in Figure 7.1. Note that lines beginning with # are read as comments.

```
#name ModelType DLA_lb DLA_ub C3_lb C3_ub AdapterMass coeff[0] ...  coeff[n]
ExampleRocket 0 -28.5 28.5 0 50 0.0 3000.0 -65.0 0.35 0.0 0.0 0.0
SmallExampleRocket 0 -28.5 28.5 0 6.5 0.0 3000.0 -65.0 0.35 0.0 0.0 0.0
```

Figure 7.1: Example Launch Vehicle Options File

7.2 Spacecraft Configuration

The Spacecraft Options file (extension: `.emtg_spacecraftopt`) specifies the capabilities of the spacecraft used in obtaining a solution with EMTG, including information regarding the propulsion and power systems. These options files are considerably more detailed than the Launch Vehicle Options files.

The spacecraft options file is organized into various sections of data labeled "Blocks". A spacecraft is defined in the file within the `#BeginSpacecraftBlock` section of data. This section contains all required information in defining a spacecraft, including globally relevant data as well as additional blocks which define the power system and propulsion system information. The

spacecraft options file also allows for multiple "stages", where each stage allows a user to change the spacecraft hardware characteristics. This information is contained within the `#BeginStageBlock` section of data. Each stage block contains information which is independent of data in other stage blocks. These blocks contain some unblocked data (such as masses and flags for constraints), a power system block identified by `#BeginStagePowerLibraryBlock`, and a propulsion system block identified by `#BeginStagePropulsionLibraryBlock`. The power and propulsion blocks are closed via a line `#EndHardwareBlock`, the stage blocks are ended via `#EndStageBlock`, and the spacecraft block is ended with `#EndSpacecraftBlock`. A `.emtg_spacecraftopt` file is thus organized as shown in Figure 7.2.

```
#BeginSpacecraftBlock
    Global Spacecraft Data
    #BeginStageBlock
        General Information for Stage 1
        #BeginStagePowerLibraryBlock
            Power System Definition
        #EndHardwareBlock
        #BeginStagePropulsionLibraryBlock
            Propulsion System Definition
        #EndHardwareBlock
    #EndStageBlock
#EndSpacecraftBlock
```

Figure 7.2: Example Structure Of `.emtg_spacecraftopt` Files

7.2.1 Spacecraft Block

The spacecraft block section contains the globally relevant data for defining a spacecraft in EMTG as well as at least one stage block containing power and propulsion system definitions. The global data contained in the spacecraft block sets the name of the spacecraft, flags for enabling or disabling maximum values on propellant and dry mass, and minimum/maximum values for constraints that are enabled. The lines and expected data types are shown in Table 7.2 followed by additional details per item.

Line	Variable	Line Name	Data Type
1	1	Name	String
2	1	EnableGlobalElectricPropellantTankConstraint	Boolean Integer
3	1	EnableGlobalChemicalPropellantTankConstraint	Boolean Integer
4	1	EnableGlobalDryMassConstraint	Boolean Integer
5	1	GlobalElectricPropellantTankCapacity	Real (kg)
6	1	GlobalFuelTankCapacity	Real (kg)
7	1	GlobalOxidizerTankCapacity	Real (kg)
8	1	GlobalDryMassBounds	Real (kg) (≥ 0, $\leq$ GlobalDryMassBounds[1])
8	2	GlobalDryMassBounds	Real (kg) ($\geq$ GlobalDryMassBounds[0])

Table 7.2: Spacecraft Block Global Data

Name:

The name of the spacecraft.

EnableGlobalElectricPropellantTankConstraint:

A boolean integer enabling a global bound on the electric propellant tank maximum capacity.

EnableGlobalChemicalPropellantTankConstraint:

A boolean integer enabling a global bound on the chemical propellant tank maximum capacity.

EnableGlobalDryMassConstraint:

A boolean integer enabling lower and upper bounds on the global dry mass.

GlobalElectricPropellantTankCapacity:

Maximum electric propellant mass, in kilograms.

GlobalFuelTankCapacity:

Maximum chemical fuel mass, in kilograms.

GlobalOxidizerTankCapacity:

Maximum oxidizer mass, in kilograms.

GlobalDryMassBounds:

Lower and upper bounds on the global dry mass, provided as two space-delimited variables, where the first defines the lower bound and the second defines the upper bound. The bound values are only considered if the corresponding constraint is active.

The spacecraft block example shown in Figure 7.3 demonstrates the usage of these mass constraints. While the chemical fuel tank capacity has a value of 1000 kg, the constraint is not active since the corresponding enable flag is set to "0". Conversely, the global dry mass constraint is active and thus the lower and upper bounds of 1000 kg and 1500 kg as set by the GlobalDryMassBounds line are active.

```
#BeginSpacecraftBlock
name example_spacecraft
EnableGlobalElectricPropellantTankConstraint 1
EnableGlobalChemicalPropellantTankConstraint 0
EnableGlobalDryMassConstraint 1
GlobalElectricPropellantTankCapacity 1000
GlobalFuelTankCapacity 1000
GlobalOxidizerTankCapacity 1000
GlobalDryMassBounds 1000 1500
```

Figure 7.3: Example Spacecraft Block Global Data

7.2.2 Stage Block General Data

After the Spacecraft Block data and still within the `#BeginSpacecraftBlock`, there are one or more Stage Block sections, bounded by `#BeginStageBlock` and `#EndStageBlock`. This section is similar to the Spacecraft Block and contains stage masses, boolean integers for enabling or disabling mass constraints, maximum masses for the fuel tanks of a stage if the associated constraint is applied, throttle logic and sharpness settings, and information regarding the power and propulsion systems. The lines and expected data types for the general stage information are shown in Table 7.3 with additional details subsequently provided, and an example is provided in Figure 7.5. The power system is defined within a Stage Power Library Block, while the propulsion systems are defined within a Stage Propulsion Library Block contained within the Stage Block. This terminology is also detailed in Section 5.4 of the EMTG Software Design Document.

Line	Line Name	Data Type
1	Name	String
2	BaseDryMass	Real (kg)
3	AdapterMass	Real (kg)
4	EnableElectricPropellantTankConstraint	Boolean Integer
5	EnableChemicalPropellantTankConstraint	Boolean Integer
6	EnableDryMassConstraint	Boolean Integer
7	ElectricPropellantTankCapacity	Real (kg)
8	ChemicalFuelTankCapacity	Real (kg)
9	ChemicalOxidizerTankCapacity	Real (kg)
10	ThrottleLogic	Integer (1: min number of thrusters, 2: max number of thrusters)
11	ThrottleSharpness	Real (¿ 0)
12	PowerSystem	String (name of power system)
13	ElectricPropulsionSystem	String (name of electric propulsion system)
14	ChemicalPropulsionSystem	String (name of chemical propulsion system)

Table 7.3: Stage Block Data

Name:

The name of the stage.

BaseDryMass:

The dry mass of the propulsion system base, in kilograms.

AdapterMass:

Mass of the spacecraft-to-power-system adapter, in kilograms.

EnableElectricPropellantTankConstraint:

A boolean integer enabling a bound on the electric propellant tank maximum capacity.

EnableChemicalPropellantTankConstraint:

A boolean integer enabling a bound on the chemical propellant tank maximum capacity.

EnableDryMassConstraint:

A boolean integer option for bounding the final mass at the end of a stage to be greater than the required mass, which is a sum of dry mass and other system masses. Users do not set bounds for this constraint. Instead, if this constraint is active, then the final mass at the end of the stage must be greater than the required mass at the end of the stage. The required mass is the sum of the stage block dry mass, stage electric propulsion system mass, stage chemical propulsion system mass, stage power system mass, electric propellant mass margin, chemical fuel mass margin, and chemical oxidizer mass margin. The fractional margins are set in the `.emtgopt` file as `electric_propellant_margin` and `chemical_propellant_margin` and may be set in the PyEMTG interface in the "Spacecraft Options" tab under the "Margins" header.

ElectricPropellantTankCapacity:

Maximum electric propellant mass, in kilograms.

ChemicalFuelTankCapacity:

Maximum chemical fuel mass, in kilograms.

ChemicalOxidizerTankCapacity:

Maximum oxidizer mass, in kilograms.

ThrottleLogic:

Integer (1 or 2) defining logic for selecting the number of thrusters. Users may choose between firing either the minimum (`ThrottleLogic = 1`) or maximum (`ThrottleLogic = 2`) number of thrusters possible for a given amount of available power. The `MaxThrusters` option typically results in thrusters fired close to their minimum power, resulting in poor performance, while the `MinThrusters` often results in thrusters being fired closer to their maximum power, resulting in better performance. While at first glance it may seem that the `MinThrusters` option would consistently be best, the choice of throttle logic for a given problem can sometimes be unintuitive.

ThrottleSharpness:

Defines how quickly the thruster transitions between different settings. A larger value indicates a steeper curve and faster transition. The throttle sharpness model uses an approximation of the discontinuous Heaviside step function $\bar{H}_i$ via a logistic function, where k is the throttle sharpness, P_a is the continuously defined power available to the Power Processing Unit (PPU), and P_i defines a power level, where $P_i \in P_1, P_2, ..., P_n$. The function can then be defined as shown in equation 7.2 and is shown in Figure 7.4, where $x = (P_a - P_i)$.

$$\bar{H}_i = \frac{1}{1 + e^{-k(P_a - P_i)}} \tag{7.2}$$

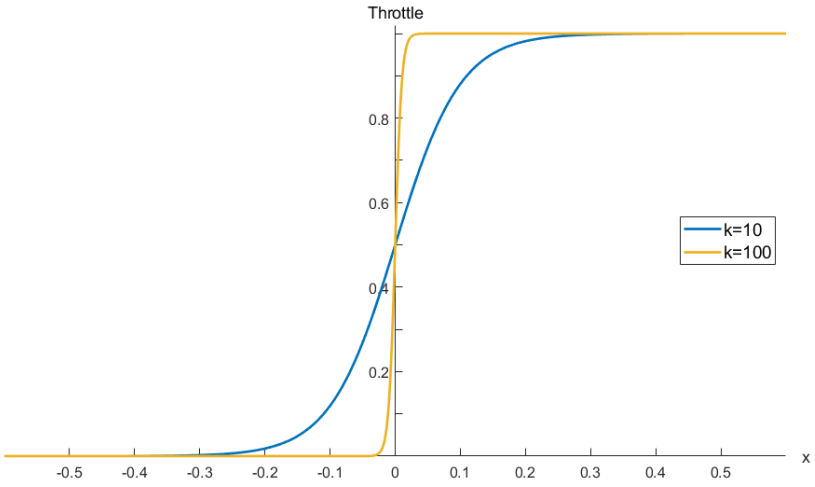


Figure 7.4: Throttle Sharpness

These throttle options are available to allow for more precise tuning of throttle table calculations. The Electric Propulsion System in particular allows for 1D and 2D smooth-stepped electric thruster models which operate using a user-supplied set of discrete throttle points (provided in the `.ThrottleTable` file). Since EMTG is a gradient based optimization tool, the thrust and I_{sp} provided in the throttle table must be first-differentiable with respect to input power. The logistic function approximation of the Heaviside function solves this differentiability problem. More precise mathematical detail of the usage of these throttle options including detailed information on the implementation of 1D throttling, 2D throttling, and multi-thruster switching is provided in section 7.2.4.2. For further detail, see Section 5.6.2 (Electric Propulsion System Model) of the EMTG Software Design Document.

PowerSystem:

Name of the power system for the stage, defined in the Power Library Block.

ElectricPropulsionSystem:

Name of the electric propulsion system for the stage, defined in the Propulsion Library Block.

ChemicalPropulsionSystem:

Name of the chemical propulsion system for the stage, defined in the Propulsion Library Block.

```
#BeginStageBlock
name Example_Stage
BaseDryMass 0
AdapterMass 0
EnableElectricPropellantTankConstraint 0
EnableChemicalPropellantTankConstraint 0
EnableDryMassConstraint 0
ElectricPropellantTankCapacity 1000.0
ChemicalFuelTankCapacity 1000.0
ChemicalOxidizerTankCapacity 1000.0
ThrottleLogic 1
ThrottleSharpness 100
PowerSystem LowSIRIS_Power
ElectricPropulsionSystem LowSIRIS_ElectricProp
ChemicalPropulsionSystem LowSIRIS_ChemicalProp
```

Figure 7.5: Example Spacecraft Stage Block Data

7.2.3 Power System

The power system is defined as an option contained in a Power Library Block. This is either defined with a `.emtg_powersystemsopt` file when setting "Spacecraft model type" to "0: Assemble from libraries" or within the Stage Block when using "1: Read .emtg_spacecraftoptions file". In each case, the format is the same as detailed in this section and demonstrated in Figure 7.6.

The power systems block allows for defining multiple power system options that can be easily interchanged, similar to the launch vehicle options file. This block is opened with `#BeginStagePowerLibraryBlock` and closed with `#EndHardwareBlock`. The information set here is that which would be set on the PyEMTG Spacecraft Options page in the "Power options" section if "Spacecraft model type" were set to "2: Assemble from missionoptions object".

In the Power Library Block, power systems are specified by a name followed by a series of comma- or space-delimited numbers on a single line. A stage can have multiple power systems specified, but only that specified in the general data in the Stage Block as the `PowerSystem` will be used. Table 7.4 specifies the parameters and data types for each Power System, with additional details for each option subsequently provided.

Items 2, 3, and 5-14 specify the power produced for the propulsion system, while items 4 and 15-17 specify the spacecraft power use for purposes other than electric propulsion (i.e., a non-zero bus power means that there is some amount of power produced by the spacecraft that cannot be used by the electric propulsion system). Figure 7.6 provides an example power library block. In

Index	Variable Name	Data Type
1	Name	String
2	Spacecraft_Power_Supply_Type	Integer (0: Constant, 1: Solar)
3	Spacecraft_Power_Supply_Curve_Type	Integer (0: Sauer, 1: Polynomial)
4	Spacecraft_Bus_Power_Type	Integer (0: Quadratic, 1: Conditional)
5	P0	Real (kW)
6	mass_per_kW	Real (kg)
7	decay_rate	Real (1/years)
8-14	gamma[0:6]	Real
15-17	BusPower[0:2]	Real

Table 7.4: Power System Variables

practice, early designs frequently make use of simple power models (e.g., the `LowSIRIS_Power` example in Figure 7.6) and move to more accurately modeled systems as the trajectory design process proceeds. Definitions of these variables follows. In some of the calculations defined by these variables, the distance of the spacecraft from the Sun, r_s, is used.

Name:

The name of the power system.

Spacecraft_Power_Supply_Type:

An integer specifying whether a constant power supply or one dependent on solar power is used by the spacecraft.

Spacecraft_Power_Supply_Curve_Type:

When the power supply type is set to solar, this option specifies the type of solar power curve, with options being Sauer or polynomial curves. The coefficients γ_i for both types are specified in indices 8-14 for a polynomial curve, or 8-11 for a Sauer curve. The resulting $P_{generated}$ is obtained in kilowatts.

$$\begin{aligned} \text{0: Sauer} \rightarrow P_{generated} &= \frac{P_0}{r_s^2}\left(\frac{\gamma_0 + \gamma_1/r_s + \gamma_2/r_s^2}{1 + \gamma_3 r_s + \gamma_4 r_s^2}\right) \\ \text{1: Polynomial} \rightarrow P_{generated} &= P_0 \cdot \sum_{i=0}^{6} \gamma_i r_s^{i-2} \end{aligned} \tag{7.3}$$

Spacecraft_Bus_Power_Type:

The spacecraft bus power reduces the available power for the electric propulsion system by assuming some amount of power must be maintained for the spacecraft itself. There are two options for modeling the bus power, where β_i is the i^{th} coefficient of the bus power.

$$\begin{aligned} \text{0: Quadratic} \rightarrow P_{Bus} &= \sum_{i=0}^{2} \beta_i / r_s^i \\ \text{1: Conditional} \rightarrow P_{Bus} &= \beta_0 && \text{if } P_{generated} > \beta_0 \\ &= \beta_0 + \beta_1 \left(\beta_2 - P_{generated}\right) && \text{otherwise} \end{aligned} \tag{7.4}$$

P0:

The power, in kilowatts, delivered at 1 AU at the start of the mission.

mass_per_kW:

The spacecraft stage mass increase per kilowatt of power delivered.

decay_rate:

The decay rate of the power supplied by the power supply curve or constant power source, in units of 1/years. The change in supplied power is given by the following equation, where t is in years and t_{ref} is the epoch, set in the `.emtgopt` file as `power_system_decay_reference_epoch` or in PyEMTG in the "Spacecraft Options" tab under "Power Source Decay Reference Epoch".

$$P_{generated} = P_{generated,nominal}\ e^{-decay\_rate\cdot(t-t_{ref})} \tag{7.5}$$

gamma[0:6]:

Coefficients for the power supply curves. In the case of a Sauer curve, the values of gamma[4:6] are inconsequential, however a total of seven are always provided.

```
#BeginStagePowerLibraryBlock
LowSIRIS_Power 1 0 0 15 10 0 1 0 0 0 0 0 0 0 0 0
example_power 1 0 0 5 10 0 1.32077 -0.10848 -0.11665 0.10843 -0.01279 0 0 0 0 0
#EndHardwareBlock
```

Figure 7.6: Example Spacecraft Power Library Block

Given the chosen settings, the power margin η_{power}, the power required by the spacecraft bus for non-propulsive functions $P_{bus}(r)$, and the distance from the spacecraft to the sun in AU r, the power available to the propulsion system (for electric propulsion systems) is calculated as:

$$P_{available}(r,t) = (1-\eta_{power})\cdot(P_{produced}(r,t) - P_{bus}(r)) \tag{7.6}$$

7.2.4 Propulsion System

Similar to the Power Library Block, the Propulsion Library Block contains one or more rows with the name of a propulsion system and a series of parameters that specify its performance. This block contains general information regarding a propulsion system. Users must appropriately define chemical or electric propulsion systems using the available parameters as desired. The lines defining these systems must have an equivalent name to that defined in the Stage Block. Table 7.5 specifies the parameters and data types for each Propulsion System, with additional details for each option subsequently provided. As with power systems, propulsion systems may be defined within a `.emtg_propulsionsystemopt` file when setting the "Spacecraft model type" to "0: Assemble from libraries" or within the Stage Block when using "1: Read .emtg_spacecraftoptions file". In each case, the format is the same as detailed in this section and demonstrated in Figure 7.7.

Index	Variable Name	Data Type
1	Name	String
2	ThrusterMode	Integer (0: ConstantThrustIsp, 1: FixedEfficiencyCSI) 2: FixedEfficiencyVSI, 3: Poly1D, 4: SteppedHThrust1D, 5: SteppedLMdot1D, 6: SteppedHIsp1D, 7: SteppedHefficiency1D, 8: SteppedFullSet1D, 9: Stepped2D, 10: Poly2D)
3	ThrottleTableFile	String ("none" or path to external file)
4	MassPerString	Real (kg)
5	NumberOfStrings	Integer
6	Pmin	Real (kW)
7	Pmax	Real (kW)
8	ConstantThrust	Real (N)
9	ConstantIsp	Real (s)
10	MinimumIsp/MonopropIsp	Real (s)
11	FixedEfficiency	Real [0, 1]
12	MixtureRatio	Real [0, 1]
13	ThrustScaleFactor	Real ($\geq$0)
14-20	ThrustCoefficients[0:6]	Real (mN)
21-27	MassFlowCoefficients	Real (mg/s)

Table 7.5: Propulsion System Variables

Name:

Name of propulsion system.

ThrusterMode:

Integer specifying type of thruster. Not all variables are relevant for each type of thruster.

ThrottleTableFile:

EMTG can read in a file specifying power processing unit efficiency and power as well as thruster throttle levels and various associated parameters at each level. Details on the format and information provided in this file are provided in 7.2.4.3.

MassPerString:

The mass of the propulsion system for one thruster "string".

NumberOfStrings:

The number of propulsion system "strings", i.e. the number of thrusters. The logic for the number of strings to use is set with the `ThrottleLogic` setting. The number of strings that can be used is dependant on the available power. EMTG uses Heaviside step function approximations to continuously transition from n strings to $n+1$ or $n-1$ strings.

Pmin:

Minimum power necessary for operation of one thruster.

Pmax:

Maximum power able to be used by one thruster if supplied by the power system.

ConstantThrust:

Thrust value for `ThrusterMode` 0, "ConstantThrustIsp".

ConstantIsp:

Isp for constant Isp thruster modes.

MinimumIsp/MonopropIsp:

Isp for a monopropellant system, if used.

MixtureRatio:

Bipropellant mixture ratio for chemical propulsion system.

FixedEfficiency:

Value for fixed efficiency if in a fixed efficiency thruster mode.

ThrustScaleFactor:

Constant factor by which to scale thrust. May be used to take into account (to first order) cosine losses caused by a canted thruster.

ThrustCoefficients:

Thrust coefficients of the input power to the thruster. Power is provided in kW and thrust in mN. Only used if the ThrottleTableFile is set to "none". For thrust coefficients η_i and Power P, thrust may be calculated as:

$$\text{Thrust} = \sum_{i=0}^{6} \eta_i \cdot P^i$$

MassFlowCoefficients:

Mass flow coefficients of the input power to the thruster. Power is provided in kW and mass flow rate in mg/s. Only used if the ThrottleTableFile is set to "none". For mass flow coefficients ζ_i and Power P, the mass flow may be calculated as:

$$\text{Mass Flow} = \sum_{i=0}^{6} \zeta_i \cdot P^i$$

7.2.4.1 ChemicalPropulsionSystem

The `ChemicalPropulsionSystem` class in EMTG computes the propellant consumed by the spacecraft in both biprop and monoprop modes. Fuel and oxidizer consumption for a maneuver are computed along with their derivatives with respect to the decision variables. An assumption is made that major maneuvers (i.e. those chosen by the optimizer) are biprop, while proportional TCMs and attitude control mass drops are monoprop. Biprop maneuvers may be overridden by the user to define them as monoprop. All chemical maneuvers are currently modeled as impulses. Thrust is considered when writing a maneuver specification file. Additionally, only one value of thrust is provided which is applied to both monoprop and biprop modes; this will be updated to allow for two thrust values in a future release.

7.2.4.2 ElectricPropulsionSystem

The `ElectricPropulsionSystem` class in EMTG computes the performance characteristics of and may also be used to size the spacecraft's provided electric propulsion system. Derivatives of thrust, I_{sp}, mass flow rate, etc. are also computed with respect to the input power.

Many different electric thruster models are provided in EMTG, each of which return the active thrust, the I_{sp}, the mass flow rate $\dot{m}$, and the active power P_{active} used by the propulsion system. For each of the models, P_{active} may be less than or equal to $P_{available}$. The desired electric thruster model is chosen using the `ThrusterMode` option. Detailed information of these models and their implementation in EMTG is provided in the following pages.

The thrust and mass flow rate outputs from any of these models are scaled by a user-defined operational duty cycle. The duty cycle may be set in the `.emtgopt` file using the `engine_duty_cycle` parameter, which defines the percentage of time the engine can operate (1E-10 -¿ 1), and the `duty_cycle_type` parameter, set to either 0 (averaged) or 1 (realistic). These options may also be set in the PyEMTG interface in the "Spacecraft Options" tab under the "Margins" header as `Thruster duty cycle` and `Duty cycle type` respectively.

Constant Thrust and Isp — `Thruster Mode 1`:

Simplest electric thruster model, with constant thrust and Isp. Requires no input power and returns zero for all derivatives. Models the propulsion system as a single, "super thruster".

Fixed Efficiency, Constant Isp — `Thruster Mode 2`:

Models the performance of a propulsion system with a fixed Isp and a propulsion system efficiency η_{prop}, which corresponds to the FixedEfficiency input in the propulsion system configuration. The user must also provide bounds on P_{active}, which are provided as Pmin and Pmax in the propulsion system configuration. If $P_{available}$ is greater than $P_{active,max}$, then it is clipped to $P_{active,max}$. If $P_{available}$ is less than $P_{active,min}$, then no thrusting occurs. Models the propulsion system as a single, "super thruster". The thrust and mass flow rate are calculated as follows, with g_0 being the acceleration due to gravity at sea level on Earth, equal to 9.80665 $\frac{m}{s^2}$.

$$T = \frac{2000\eta_{prop}}{I_{sp}g_0}P_{active}$$

$$\dot{m} = \frac{T}{I_{sp}g_0}$$

1D Polynomial — `Thruster Mode 3`:

Models the thrust and Isp using a polynomial approximation as a function of input power. Models the propulsion system using the number of active thrusters and the input power per thruster. The thrust and mass flow rate per thruster is then computed as follows.

$$T = a_t + b_t P + c_t P^2 + d_t P^3 + e_t P^4 + f_t P^5 + g_t P^6$$

$$\dot{m} = a_f + b_f P + C_f P^2 + d_f P^3 + e_f P^4 + f_f P^5 + g_f P_6$$

The resulting values are multiplied by the number of active thrusters to determine the total thrust and mass flow rate for the system. The input power per thruster is multiplied by the number of active thrusters to obtain the total P_{active}.

1D Smooth-Stepped — `Thruster Modes 4 - 8`:

Electric thrusters are typically operated at set performance points, where mass flow rate and input voltage may be adjusted on grid. This results in discrete throttle settings. Given enough available power, the propulsion system may take any of these discrete settings. The 1D smooth-stepped throttle model operates directly on the discrete throttle points. A user supplied list of throttle points are provided via a `ThrottleTable`. A non-dominated set of `ThrottleSetting` objects are created with the `ThrottleTable` using a sorting rule. Thruster modes 4 through 8 specify which 1D smooth-stepped sorting mode is to be used. The available sorting rules are:

- `Thruster Mode 4`: Highest thrust per available power.
- `Thruster Mode 5`: Lowest mass flow rate per available power.
- `Thruster Mode 6`: Highest I_{sp} per available power.
- `Thruster Mode 7`: Highest system efficiency per available power.
- `Thruster Mode 8`: Full set - use all user-supplied throttle points.

Since EMTG is a gradient based optimization tool, thrust and I_{sp} must be first-differentiable with respect to input power, thus a Heaviside function is used. This function is defined in equation 7.7, where P_a is the continuously defined power available to the Power Processing Unit (PPU), and P_i defines a power level, where $P_i \in P_1, P_2, ..., P_n$.

$$H_i = \begin{cases} 1 & \text{if } (P_a - P_i) \geq 0 \\ 0 & \text{if } (P_a - P_i) < 0 \end{cases} \tag{7.7}$$

The mass flow rate or thrust at any given available power is then:

$$x = x_1 H_1 + \sum_{i=2}^{n} (x_i - x_{i-1}) H_i \tag{7.8}$$

However this does not allow for first order differentiability. This is solved by replacing the discontinuous Heaviside functions with logistic function approximations, where k is the throttle sharpness as provided in the StageBlock which determines how closely the logistic functions model the step increases of the Heaviside function. For values of k greater than 1000, the Heaviside step function and logistics function approximation are nearly indistinguishable, however this negatively impacts the differentiability of the function. It is assumed that only one value of k is used everywhere.

$$\bar{H}_i = \frac{1}{1 + e^{-k(P_a - P_i)}} \tag{7.9}$$

The resulting model for mass flow rate at any given available power follows, where x is either mass flow rate or thrust.

$$x = x_1 \bar{H}_1 + \sum_{i=2}^{n} (x_i - x_{i-1}) \bar{H}_i \tag{7.10}$$

As mentioned a `ThrottleTable` file provides the discrete throttle points. In practice, the optimizer performs poorly when multiplying two logistics functions together, likely due to the derivatives of these function being sharp. It is a generally good idea to encode multi-thruster throttle points directly into the provided throttle table, rather than having a file represent a single thruster and allowing EMTG to switch thrusters on and off.

2D Throttling — `Thruster Modes 9/10`:

To remove the need for a user to choose a priori how to traverse a throttle grid (as is needed in the smooth-stepped methods), allowing the optimizer to choose how to do so is preferable. **This is an in-progress capability in EMTG and is not currently implemented.** The two methods in progress are a fixed efficiency variable specific impulse approximation and a 2D smooth-stepped method. Since these methods are not currently available, their specifics are left to Section 5.6.2.5 of the EMTG Software Design Document.

```
#BeginStagePropulsionLibraryBlock
Poly 3 none 0 1 0.5 2.6 0 1 0 0 0 1 26 -51.6 90.4 -36.7 5 0 0 2.5 -5 6 -2 0.3 0 0
Chem 0 none 0 1 0.5 1.0 22.0 320.0 220.0 0.5 0.9 1.0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Warp 0 none 0 1 0.5 1.0 22.0 1.0e+6 1.0e+6 0.5 0.9 1.0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
#EndHardwareBlock
```

Figure 7.7: Example Spacecraft Propulsion Library Block

7.2.4.3 ThrottleTable

A `.ThrottleTable` file is a comma-delimited file where lines beginning with `#` are comment lines. This file defines power processing unit capabilities, tabular data defining the throttle level, and polynomial coefficients for polynomial thruster modes. The structure of this file has some additional complexity since it contains both singular defined parameters, coefficients for 1D and 2D polynomials, and tabularized throttle level information. Information may be provided in various orders. The structure of each line in the throttle table itself is defined in Table 7.6. Data that is provided on single lines is defined in Table 7.7. Since these files have some additional complexity, the structure is demonstrated in Figure 7.8.

Index	Variable Name	Data Type
1	Throttle level	String, TL##
2	Mass flow rate	Real (mg/s)
3	Beam Current	Real (A)
4	Beam Voltage	Real (V)
5	Thrust	Real (mN)
6	Isp	Real (s)
7	Efficiency	Real (≥ 0, ≤ 1)
9	PPU input power	Real (kW)

Table 7.6: Throttle Table Tabularized Data

Variable Name	Description & Data Type
PPU efficiency	Defines efficiency of power processing unit, Real (≥ 0, ≤ 1)
PPU min power (kW)	Minimum power for power processing unit, Real
PPU max power (kW)	Maximum power for power processing unit, Real
high_thrust_Thrust	Thrust coefficients 1-5 for high thrust Poly1D thruster mode, Real
high_thrust_Mdot	Mass flow coefficients 1-5 for high thrust Poly1D thruster mode, Real
high_Isp_Thrust	Thrust coefficients 1-5 for high efficiency Poly1D thruster mode, Real
high_Isp_Mdot	Mass flow coefficients 1-5 for high efficiency Poly1D thruster mode, Real
2Dpolyrow1	Coefficients 1-5 for row one of Poly2D thruster mode, Real
2Dpolyrow2	Coefficients 1-5 for row two of Poly2D thruster mode, Real
2Dpolyrow3	Coefficients 1-5 for row three of Poly2D thruster mode, Real
2Dpolyrow4	Coefficients 1-5 for row four of Poly2D thruster mode, Real
2Dpolyrow5	Coefficients 1-5 for row five of Poly2D thruster mode, Real

Table 7.7: Throttle Table Line Data

```
PPU efficiency, 1.0
PPU min power (kW), 0.63
PPU max power (kW), 7.26
#Throttle table data
TL01, 1.0, 1.0, 1000, 100, 3000, 1.0, 1.0
TL02, 1.5, 1.0, 1000, 120, 3100, 1.0, 1.4
TL##, ..., ...,  ..., ...,  ..., ..., ...
.
.
.
#coefficients for 1D polynomials, lowest order on the left, random example numbers
high_thrust_Thrust, 1.2E-02, 2E-02, 1E-01, -1E-03, 1E-04
high_thrust_Mdot, 3E-06, -1.5E-06, 1E-06, -2E-07, 1E-08
high_Isp_Thrust, 3E-03, 4E-02, -6E-03, 2E-03, -1E-04
high_Isp_Mdot, 2E-06, -2E-07, 2.5E-08, 3E-08, -3E-09
#coefficients for 2D polynomials
2Dpolyrow1, 0.0, 0.0, 0.0, 0.0, 0.0
2Dpolyrow2, 0.0, 0.0, 0.0, 0.0, 0.0
2Dpolyrow3, 0.0, 0.0, 0.0, 0.0, 0.0
2Dpolyrow4, 0.0, 0.0, 0.0, 0.0, 0.0
2Dpolyrow5, 0.0, 0.0, 0.0, 0.0, 0.0
```

Figure 7.8: Example Throttle Table File

Chapter 8

Conversion Scripts

EMTG also comes packaged with convenient conversion scripts to aid users in the process of iteratively increasing the fidelity of a mission. This allows users to, for example, generate a low-fidelity trajectory to some planetary body that may include some intermediate flyby, improve the fidelity of that trajectory by automatically transitioning flybys into multiple phases, and then further improve the fidelity by properly modeling transitions in and out of spheres of influence of the visited bodies. This requires some small amount of manual scripting which is covered in the Flybys tutorial located at `docs/0_Users/tutorial/Tutorial_Docs/4_Flybys.pdf`. These conversion scripts are available in `PyEMTG/Converters/` and `PyEMTG/HighFidelity`.

8.1 Convert To Single Phase Journeys

A typical process for converting a low fidelity EMTG mission which contains flybys of some planetary bodies to a higher fidelity mission starts with the `convert_to_single_phase_journeys.py` script available in `PyEMTG/Converters/`. This script removes bodies from the flyby sequence, and instead insert them as new Journey departure and arrival points. Thus, a mission is expanded into more Journeys, however each of these Journeys are of a single phase rather than multi-phase (pre- and post- flyby). This does not improve upon the fidelity of the mission, but does provide a stepping stone for later conversion scripts to do so. This change can be made manually, however the converter script also automatically populates the initial guess for the new Journey with values and appropriately renames the relevant decision variables. Thus, if the Journey structure is the only thing that is changed, EMTG should produce exactly the same trajectory with either Journey structure. For specific details and an example of the usage of this converter, see the Flybys tutorial at `docs/0_Users/tutorial/Tutorial_Docs/4_Flybys.pdf`.

8.2 High Fidelity Conversion

Converting to a high-fidelity mission requires an EMTG options file with single phase Journeys and an obtained solution. The conversion scripts that are used to accomplish this are found in `PyEMTG/HighFidelity`. Users interact exclusively with the `HighFidelityDriverFunction.py` script. This conversion transitions the launch and any flybys into a more accurate representation that includes body sphere of influence departure and entry. Thus, additional Universe files defining central bodies of the launch body and any flyby bodies are required.

High Fidelity Conversion Universe File Name Issue

High fidelity `.emtgopt` files created when using the high fidelity conversion scripts will not work unless the universe files for the launch body and any converted flyby bodies share the exact name as that used in the menu of bodies for the main Sun `.emtg_universe` file used in the lower fidelity `.emtgopt` file.

It is important to note that the high-fidelity conversion script applies to flybys only. The final Journey's arrival is not converted to a higher fidelity sphere of influence to free point arrival orbit. This must instead be done manually, which is demonstrated in the Journey Boundaries tutorial located at `docs/0_Users/tutorial/Tutorial_Docs/3_JourneyBoundaries.pdf`. Like the single phase conversion script, the high-fidelity flyby conversion script also generates initial guesses for the new decision variables. These decision variables do not necessarily produce a feasible solution without optimization; however, they do provide a good starting point for EMTG. For specific details and an example of the usage of this converter, see the Flybys tutorial at `docs/0_Users/tutorial/Tutorial_Docs/4_Flybys.pdf`.

Chapter 9

Other Resources

While this user guide is intended to cover the vast majority of the usage of EMTG, some of the more advanced capabilities benefit from independent explanation. There are two main advanced capabilities to cover: manual constraint scripting and the Python EMTG Automated Trade Study Application (PEATSA).

9.1 Constraint Scripting

Manual constraint scripting refers to the ability to script more complex constraints directly in the `.emtgopt` file. This allows users to design constraints that exceed the limitations of PyEMTG. There are three classes of constraints available: boundary constraints, maneuver constraints, and phase distance constraints (otherwise known as path constraints). Additionally, users may as needed create new boundary constraints by developing custom EMTG C++ code. For more information on constraint scripting, refer to the EMTG Constraint Scripting documentation located at `docs/0_Users/constraint_scripting/EMTG_constraint_scripting.pdf`. For a brief tutorial on basic usage of constraint scripting in EMTG, refer to the Constraint Scripting tutorial located at `docs/0_Users/tutorial/Tutorial_Docs/8_Constraint_Scripting.pdf`.

9.2 Python EMTG Automated Trade Study Application

The Python EMTG Automated Trade Study Application, or PEATSA, is a tool used to create and execute trade studies. It is a set of Python scripts which take an EMTG options file, known as the "base case", and varies parameters as defined by the user to generate and execute variations from said base case. For more information on PEATSA, including how to design and execute a PEATSA run, refer to the PEATSA user guide located at `docs/0_Users/PyEMTG_User_Guide/PyEMTG_User_Guide.pdf`. For a brief tutorial on the usage of PEATSA, refer to the PEATSA tutorial located at `docs/0_Users/tutorial/Tutorial_Docs/7_PEATSA.pdf`.