EMTG Tutorial: Flybys

Tim Sullivan [*]

August 8, 2023

Revision Date	**Author**	**Description of Change**
December 2, 2022	Tim Sullivan	Initial revision.
June 30, 2023	Joseph Hauerstein	Conversion to LaTeX.
August 4, 2023	Joseph Hauerstein	Addition of Known Issues section.

[*]Aerospace Engineer, The Aerospace Corporation

Contents

List of Known Issues

List of Acronyms

EMTG Evolutionary Mission Trajectory Generator

GUI Graphical User Interface

SNOPT Sparse Nonlinear OPTimizer

SOI sphere of influence

SPICE Spacecraft Planet Instrument Camera-matrix Events

ZSOI zero-sphere-of-influenence

NLP Nonlinear Program

MBH Monotonic Basin Hopping

1 Introduction

In the previous tutorials, you've created several flybys of other planets on the way to the destination body. All these flybys used a zero-sphere-of-influenence (ZSOI) patched conics approximation, Figure 1. In this model, the gravity-assist effect of the flyby is modeled as an instantaneous change in the velocity vector defined by analytical approximations. As a result, this model is extremely computationally efficient, but lacks high accuracy.

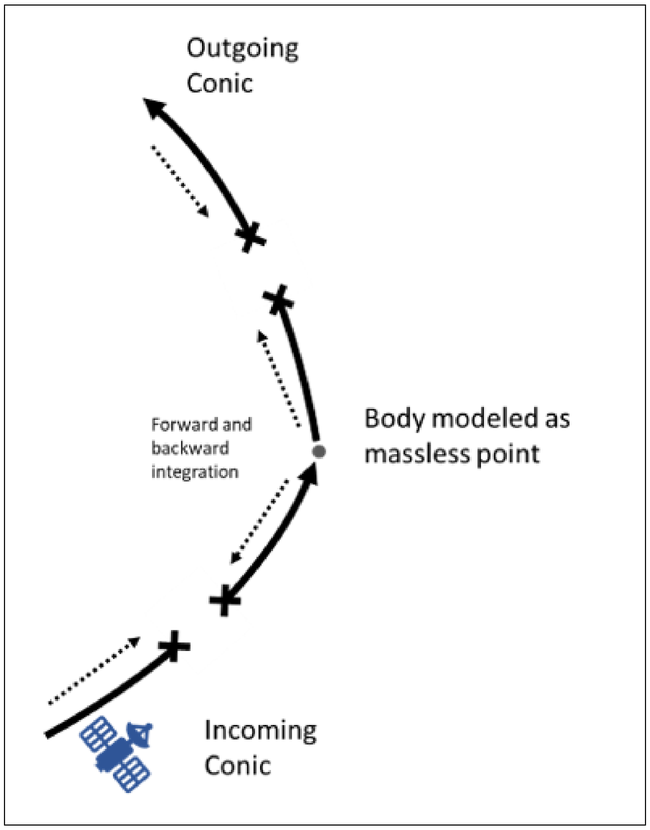

Outgoing Conic
Forward and backward integration
Body modeled as massless point
Incoming Conic

Figure 1: Patched Conic Transcription.

In this tutorial, you will walk through how to make Evolutionary Mission Trajectory Generator (EMTG) model the flyby more accurately, with the spacecraft flying around the body instead of through it, while more realistically including the effects of the flyby body's gravity during the close approach as in Figure 2.

EMTG comes with several Python scripts to help transform a ZSOI flyby into a fully propagated flyby. First, if necessary, you can convert a multi-Phase Journey (a Journey that contains one or more bodies in the "Flyby Sequence" section of the "Journey Options" tab) to a series of single-Phase journeys (each flyby or body departure/arrival occurs at a Journey boundary) with one Python script. Then, you can convert the single-Phase-Journeys Mission to a Mission with propagated flybys using another Python script. When you finish this process, each departure, flyby, and arrival body will be represented by multiple Journeys switching the primary gravitational body (the central body) at each sphere of influence (SOI) crossing. Table 1 summarizes this process.

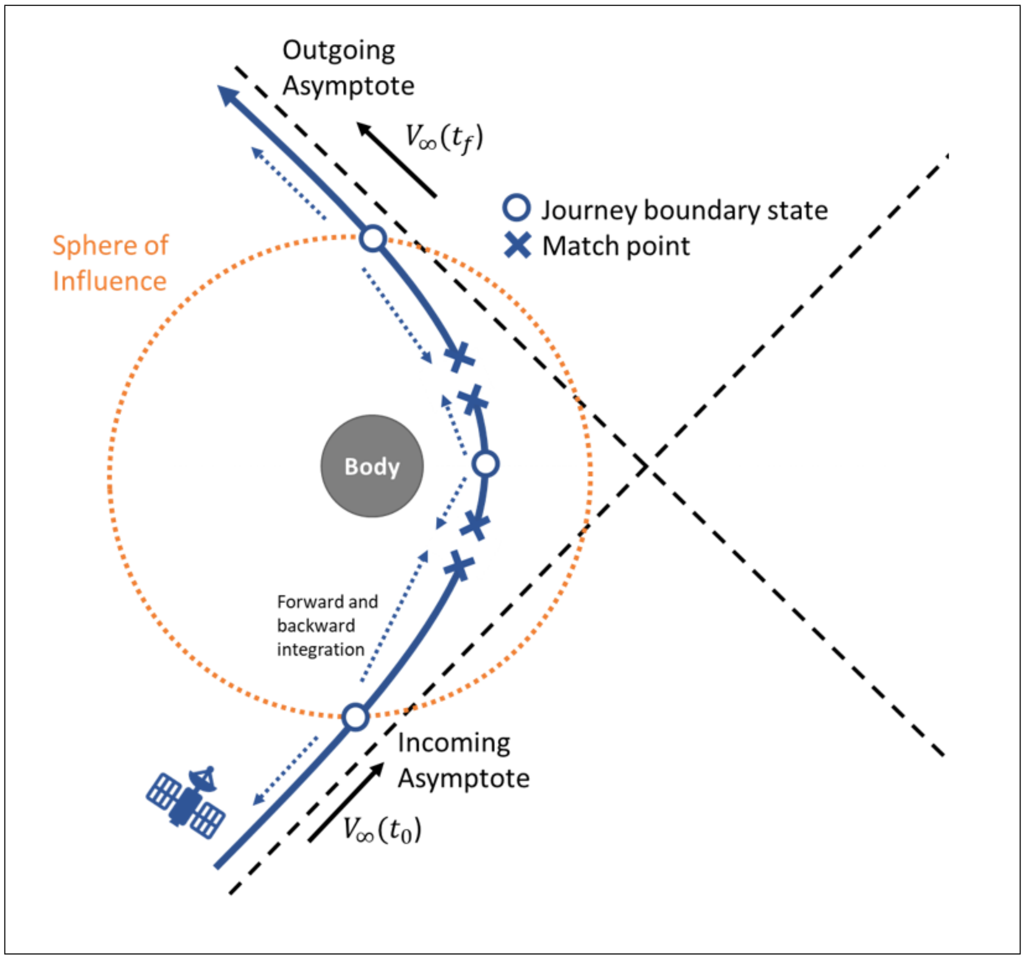


Figure 2: Ephemeris-referenced Gravity-assist Multiple-shooting Transcription.

Starting EMTG Options Files	Converter	Output EMTG Options File
Multi-Phase Journey(s) Example: Earth to Mars with Venus in flyby sequence (Ephemeris-pegged boundaries)	PyEMTG/Converters/convert_to_single_phase_journeys.py	Single-Phase Journeys with flybys Example: Journey 0: Earth to Venus Journey 1: Venus to Mars (Ephemeris-pegged boundaries)
Single-Phase Journeys with flybys Example: Journey 0: Earth to Venus Journey 1: Venus to Mars (Ephemeris-pegged boundaries)	PyEMTG/HighFidelity/ HighFidelityDriverFunction.py	High-fidelity EMTG Mission Example: Journey 0: Earth periapse to Earth SOI Journey 1: Earth SOI to Venus SOI Journey 2: Venus SOI to Venus periapsis Journey 3: Venus periapse to Venus SOI Journey 4: Venus SOI to Mars center of mass (Multiple Journey arrival/departure classes)

Table 1: Journey Arrival Elements.

2 Setup

Make a new directory for this tutorial called `Flybys` and copy and paste in the `EVM.emtgopt` file, and `hardware_models` folders from the Journey Boundaries tutorial. Open the `EVM.emtgopt` file and change the paths for “Hardware library path” in the “Spacecraft Options” tab, and “Working directory” in the “Output Options” tab to a new `results` directory the new `Flybys` directory. Don’t forget the trailing slash in “Hardware library path”.

You will also want to have the solution to this low-fidelity mission as an initial guess for the higher fidelity conversions, so either run mission again without changing any additional options or have the solution folder from the first EVM run in the Journey Boundaries tutorial handy.

2.1 Universe Files

You need to make new Universe files before you can run the converted missions. You already have a Universe file for the Sun and for Mars, but you need them for Earth and Venus, as well, because those two bodies will now be the central bodies of Journeys. In the `EVM_universe` directory, make two copies of the `Sun.emtg_universe` file and call them `Earth.emtg_universe` and `Venus.emtg_universe`, respectively. Copy the "#menu of bodies" line for the Sun from `Mars.emtg_universe` and paste it into the Earth and Venus Universe files. Delete the "#menu of bodies" line for Earth from the Earth file and the Venus line from the Venus file. Like when making the Mars universe file in the Journey Boundaries tutorial, make sure that the ID number for each body in the menu of bodies is unique. (E.g., Sun and Mercury cannot both have ID 1.) Finally, update the central body section of the Earth and Venus Universe files.

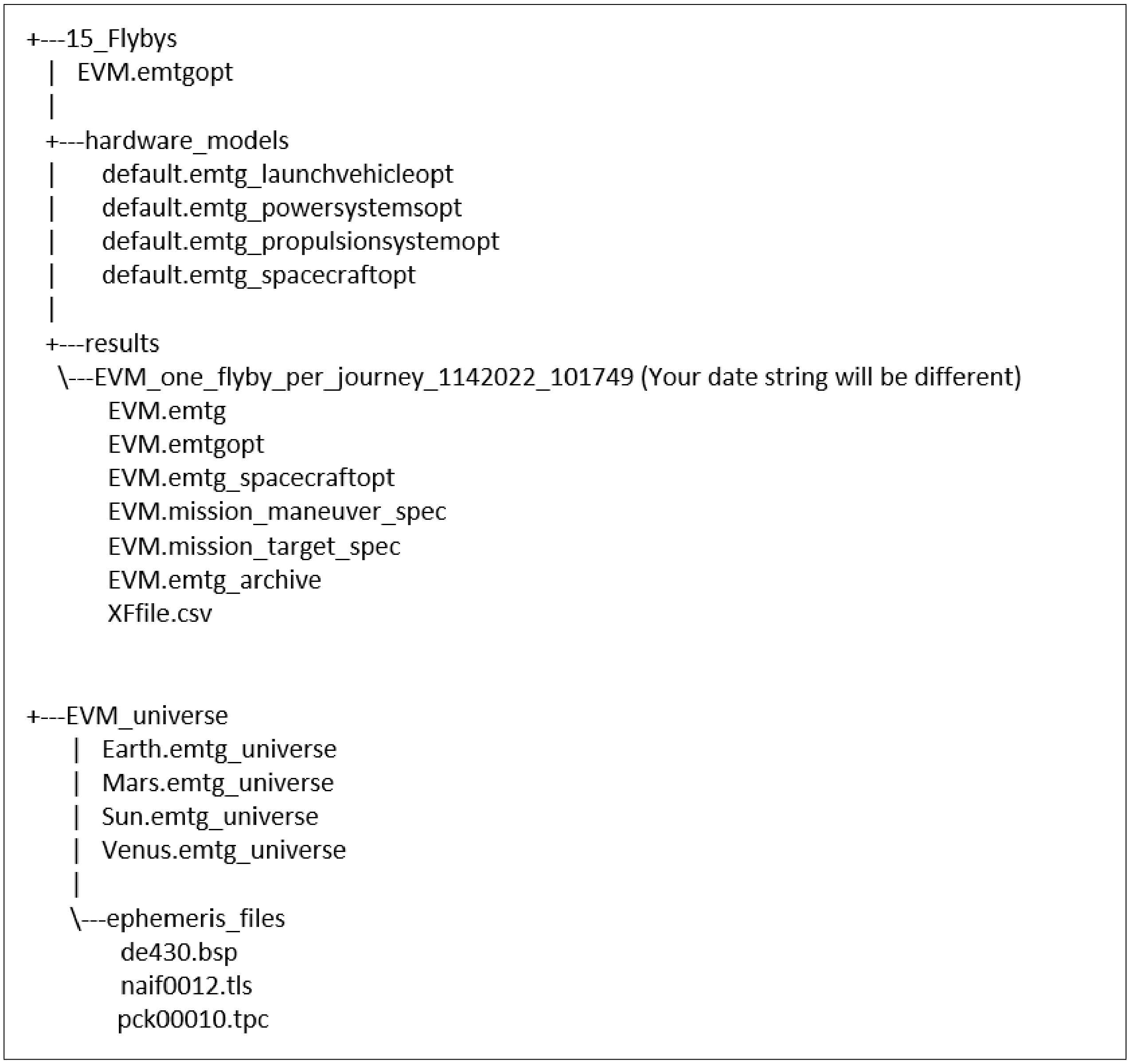
Reference the text below for Venus.

```
#Central body name
central_body_name Venus
#Central body SPICE ID
central_body_SPICE_ID 299
#central body radius
central_body_radius 6051.8
#gravitational constant of central body, in km^3/s^2
mu 324858.592
#characteristic length unit, in km
LU 6051.8
#angles defining the local reference frame relative to ICRF, given in degrees
#alpha0, alphadot, delta0, deltadot, W, Wdot
reference_angles 272.76 0.0 67.16 0.0 160.2 -1.4813688
#radius of the central body's sphere of influence
r_SOI 616000
#minimum safe distance from the central body
minimum_safe_distance 6351
```

Reference the text below for Earth.

```
#Central body name
central_body_name Earth
#Central body SPICE ID
central_body_SPICE_ID 399
#central body radius
central_body_radius 6378.14
#gravitational constant of central body, in km^3/s^2
mu 398600.436296
#characteristic length unit, in km
LU 6378.14
#angles defining the local reference frame relative to ICRF, given in degrees
#alpha0, alphadot, delta0, deltadot, W, Wdot
reference_angles -90.0 -0.641 90.0 -0.557 280.147 360.9856235
#radius of the central body's sphere of influence
r_SOI 924500
#minimum safe distance from the central body
minimum_safe_distance 6678
```

When finished, you should have a working directory that looks like the one in Figure 3. (Your results directory will be empty until later in the tutorial.) Refer to the Universe files provided with this tutorial if you are unsure about the Universe files or directory setup. You now have everything you need to convert the `EVM.emtgopt` file to higher fidelity missions.



```
+---15_Flybys
  |  EVM.emtgopt
  |
  +---hardware_models
  |      default.emtg_launchvehicleopt
  |      default.emtg_powersystemsopt
  |      default.emtg_propulsionsystemopt
  |      default.emtg_spacecraftopt
  |
  +---results
   \---EVM_one_flyby_per_journey_1142022_101749 (Your date string will be different)
         EVM.emtg
         EVM.emtgopt
         EVM.emtg_spacecraftopt
         EVM.mission_maneuver_spec
         EVM.mission_target_spec
         EVM.emtg_archive
         XFfile.csv


+---EVM_universe
     |  Earth.emtg_universe
     |  Mars.emtg_universe
     |  Sun.emtg_universe
     |  Venus.emtg_universe
     |
     \---ephemeris_files
           de430.bsp
           naif0012.tls
           pck00010.tpc
```

Figure 3: Working Directory Example.

3 Conversion to Single-Phase Journeys

The first step towards a high-fidelity EMTG mission is removing Venus from the flyby sequence and inserting it as a new Journey departure and arrival point. This is easy to do with PyEMTG, but you're going to take advantage of a Python converter script to show how it can be done automatically. One advantage of using the Python converter script is that it automatically populates the initial guess for the new Journey structure with the values from the old Journey structure, including appropriately renaming the relevant decision variables. As a result, if the Journey structure is the only thing that is changed, EMTG should produce exactly the same trajectory with either Journey structure.

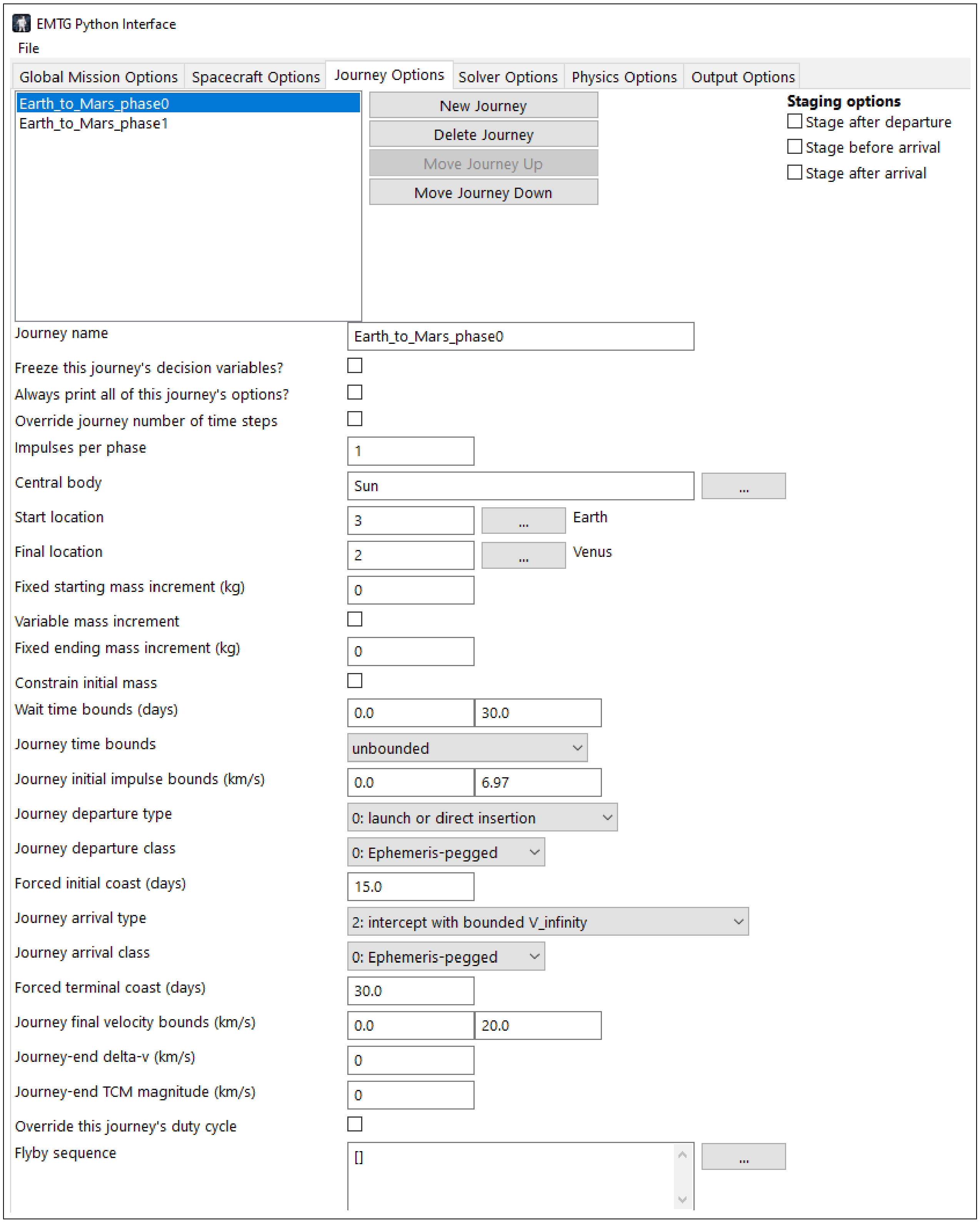
The script is located in your EMTG repository at `EMTG/PyEMTG/Converters/convert_to_single_phase_journeys.py`. Open this file with a text editor like Notepad++ or VSCode. Notice that there is a `sys.path.append` line near the top of the file in the imports section. This path needs to point to your PyEMTG directory. If you installed EMTG somewhere other than what is shown, update this line so that the script has the correct path to your PyEMTG directory. Save the file. Now, you will perform the first conversion to single-phase Journeys.

Open a terminal and execute the script with Python, providing two arguments: one for the path to the EMTG options file (`EVM.emtgopt`) and the other for the path to the EVM solution file (`EVM.emtg`). As an example (all in one line), `python C:\<EMTG-FOLDER>\emtg\PyEMTG\Converters \convert_to_single_phase_journeys.py C:\<PATH-TO-WORKING-DIR>\EVM.emtgopt C:\<PATH-TO-WORKING-DIR>\results\EVM_DATE_TIME\EVM.emtg`

You should now have a file called `EVM_singlePhase.emtgopt` alongside your `EVM.emtgopt` file. Open `EVM_singlePhase.emtgopt` in PyEMTG. You should have two Journeys called "Earth_to_Mars_phase0" and "Earth_to_Mars_phase1" with the Journey arrival and departures shown in Table 2. The full list of settings is shown in Figure 4. Phase 0 refers to the Earth-to-Venus trajectory, and phase 1 refers to the Venus-to-Mars trajectory. Note that this does not change the Mars arrival state to the free point Journey arrival state you saw in the Journey Boundaries tutorial.

Journey Name	Journey Departure Type	Journey Departure Class	Journey Arrival Type	Journey Arrival Class
Earth_to_Mars_phase0	0: launch or direct insertion	0: Ephemeris-pegged	2: Intercept with bounded V_infinity	0: Ephemeris-pegged
Earth_to_Mars_phase1	3: flyby	0:Ephemeris-pegged	2: Insertion into parking orbit (use chemical Isp)	0: Ephemeris-pegged

Table 2: Single Phase Journey State Types.

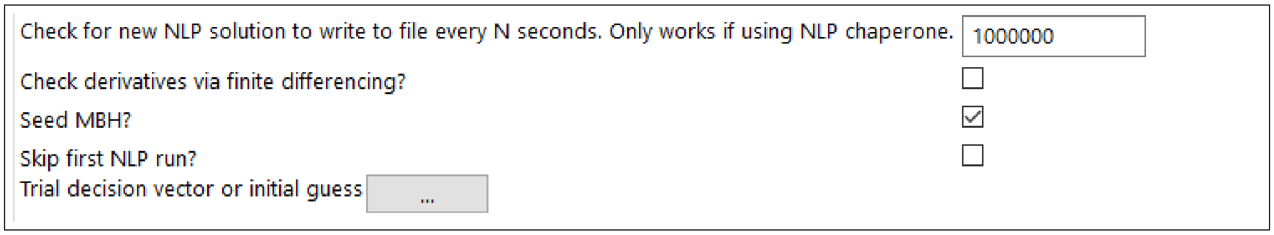


Earth_to_Mars_phase0
Earth_to_Mars_phase1

Staging options
- ☐ Stage after departure
- ☐ Stage before arrival
- ☐ Stage after arrival

Journey name	Earth_to_Mars_phase0
Freeze this journey's decision variables?	☐
Always print all of this journey's options?	☐
Override journey number of time steps	☐
Impulses per phase	1
Central body	Sun ...
Start location	3 ... Earth
Final location	2 ... Venus
Fixed starting mass increment (kg)	0
Variable mass increment	☐
Fixed ending mass increment (kg)	0
Constrain initial mass	☐
Wait time bounds (days)	0.0 30.0
Journey time bounds	unbounded
Journey initial impulse bounds (km/s)	0.0 6.97
Journey departure type	0: launch or direct insertion
Journey departure class	0: Ephemeris-pegged
Forced initial coast (days)	15.0
Journey arrival type	2: intercept with bounded V_infinity
Journey arrival class	0: Ephemeris-pegged
Forced terminal coast (days)	30.0
Journey final velocity bounds (km/s)	0.0 20.0
Journey-end delta-v (km/s)	0
Journey-end TCM magnitude (km/s)	0
Override this journey's duty cycle	☐
Flyby sequence	[] ...

Figure 4: Single Phase Journey Options.



Check for new NLP solution to write to file every N seconds. Only works if using NLP chaperone. 1000000

Check derivatives via finite differencing? ☐

Seed MBH? ✓

Skip first NLP run? ☐

Trial decision vector or initial guess ...

Figure 5: Seed MBH.

3.1 Seeding

Open the `EVM_singlePhase.emtgopt` in a text editor or select "File" ->"open file in editor" (Ctrl+e) in PyEMTG. Scroll down to the Journeys section and look for the entry "#trial decision vector" at the end of each section of Journey options. The trial decision vector for each Journey has been "seeded" or filled in using the solution from `EVM.emtg` as an initial guess. The names of the decision variables may be difficult to parse at first due to the lack of spaces, but you will probably be able to understand at least vaguely what a lot of the variables mean.

NOTE: PyEMTG may not open the file in a text editor if the file is associated with another program.

You can also seed the new EMTG options file you just created with the results from another trial by using the "Seed MBH?" option in the "Solver Options" tab of the GUI. Seeding provides a starting point for the optimizer and can help the optimizer find a solution more quickly than starting from a random guess. Switch to the "Solver Options" tab and check this box to reveal the "Trial decision vector or initial guess" button, which you can use to select previous results (i.e., a previous .emtg file) as an initial guess for this mission. These settings are shown in Figure 5.

NOTE: The conversion script already provided a good initial guess and populated the appropriate sections of `EVM_singlePhase.emtgopt`. The new single-Phase Mission decision vector is not the same as the original Mission, and the conversion script has provided initial guesses for each new variable. Using the "Seed MBH?" box here and selecting the old `EVM.emtg` as the seed will NOT provide a guess for every decision variable in the new single-Phase Mission, and you will get EMTG console output such as:

```
Initial guess missing value for:  j0p0MGAnDSMsEphemerisPeggedIntercept:  V_infinity_x
```

3.2 Run the Mission

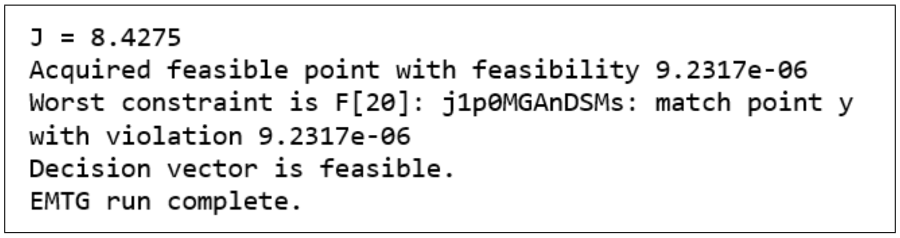
Run the file (Ctrl+r) and note that the EMTG terminal output reads something like,

```
NLP incumbent point and exit point are feasible and incumbent point is superior to
exit point.
```

Or

```
NLP incumbent point was infeasible but less infeasible than the exit point.
```

after loading the SPICE files. Depending on your mission and the solution provided to the conversion script, the output of the single-phase script may or may not already have a feasible initial guess. (If the `EVM.emtg` file you provided to the conversion script was feasible, the new Mission should be feasible, too. You can check this directly by switching to the "Solver Options" tab, selecting "Evaluate trialX" in "Inner-loop Solver Mode", then running EMTG. If your output looks like that shown in Figure 6 with the text "Decision vector is feasible", the initial guess is feasible. (Do not expect the exact numbers to be the same because EMTG is a stochastic optimizer.) Let's increase the fidelity of this mission further.



```
J = 8.4275
Acquired feasible point with feasibility 9.2317e-06
Worst constraint is F[20]: j1p0MGAnDSMs: match point y
with violation 9.2317e-06
Decision vector is feasible.
EMTG run complete.
```

Figure 6: Evaluate TrialX Output.

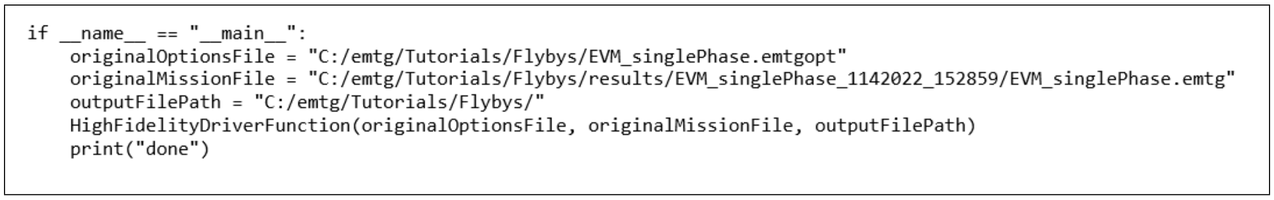
4 Conversion to High-fidelity Mission

The high-fidelity conversion script takes an EMTG options file with single-Phase Journeys and solution as input. It's located in `EMTG/PyEMTG/HighFidelity/HighFidelityDriverFunction.py`. Before running it, you need to check another system path command in the file `EMTG/PyEMTG/HighFidelity/HighFidelityJourney.py`. As before, open the file in a text editor and check that the `sys.path.append` line near the top of the file points to your PyEMTG folder, or the Python import statements will not work correctly.

Now open `HighFidelityDriverFunction.py` in a text editor. This conversion script does not use command-line arguments. Instead, scroll down to the "`if __name__ == "__main__":`" section. Change the `originalOptionsFile` variable to point to the single-Phase Journeys EMTG options file. Change the `originalMissionsFile` variable to point to the single-Phase Journeys EMTG solution file, and the `outputFilePath` variable to point to the directory where you wish to save the high-fidelity EMTG options file. This section in your file should now look something like the one in Figure 7.

Save and run the script using the terminal command:

```
python EMTG/PyEMTG/HighFidelity/HighFidelityDriverFunction.py
```



```
if __name__ == "__main__":
    originalOptionsFile = "C:/emtg/Tutorials/Flybys/EVM_singlePhase.emtgopt"
    originalMissionFile = "C:/emtg/Tutorials/Flybys/results/EVM_singlePhase_1142022_152859/EVM_singlePhase.emtg"
    outputFilePath = "C:/emtg/Tutorials/Flybys/"
    HighFidelityDriverFunction(originalOptionsFile, originalMissionFile, outputFilePath)
    print("done")
```

Figure 7: HighFidelityDriverFunction.py Example.

A new file called `HighFidelity.emtgopt` should be created in the directory you specified in `outputFilePath`. Open it with PyEMTG.

Notice that you now have several more Journeys. The EMTG mission now proceeds from Earth periapse to Earth SOI to Venus SOI to Venus periapse to Venus SOI to Mars. The converter has also included the SOI radius from the Earth and Venus Universe files when defining the ephemeris-referenced arrival states. Table 3 shows the new Journey configuration.

Journey Name	Journey Departure Type	Journey Departure Class	Journey Arrival Type	Journey Arrival Class
EarthLaunch	0: launch or direct insertion	3: Periapse	2: Intercept with bounded V_infinity	2: Ephemeris-referenced
Earth_to_Mars_phase0	2: free direct departure	1: Free point	2: Intercept with bounded V_infinity	0: Ephemeris-referenced
VenusGAi	2: free direct departure	1: Free point	2: Intercept with bounded V_infinity	3: Periapse
Venus GAo	2: free direct departure	1: Free point	2: Intercept with bounded V_infinity	2: Ephemeris-referenced
Earth_to_Mars_phase1	3: free direct departure	0: Free point	2: Insertion into parking orbit (use chemical Isp)	0: Ephemeris-pegged

Table 3: High Fidelity Journey State Types.

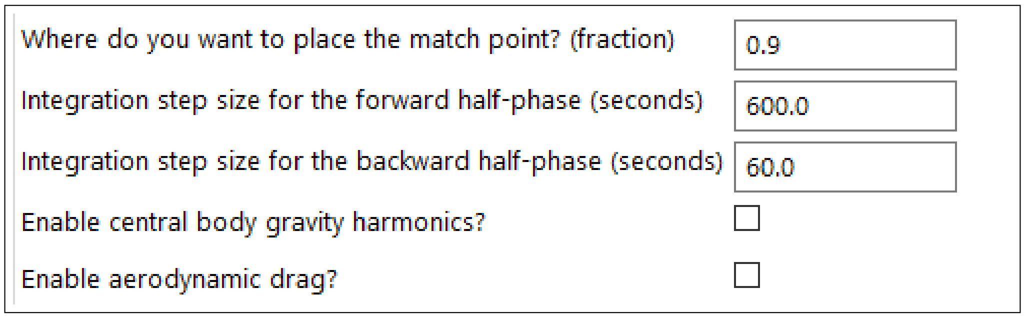
Note that neither of the conversion scripts changed the Mars ephemeris-pegged arrival to the more realistic Mars SOI to Mars free point arrival orbit you created in the Journey Boundaries tutorial. You'll need to do that yourself. Since it was covered previously, you won't update the high-fidelity mission here. The high-fidelity conversion script only converts ephemeris-pegged launches and ephemeris-pegged flybys to propagated trajectories.

Like with the single-Phase conversion script, the high-fidelity flyby conversion script also generates initial guesses for the new decision variables, and you can examine them in the new `HighFidelity.emtgopt` file. Unlike the results of the single-Phase conversion, though, you should not necessarily expect the high-fidelity decision variables to produce a feasible solution without optimization, even if the starting low-fidelity EMTG case was feasible. However, the initial guess is usually quite good, though the quality of the initial guess tends to degrade as the flyby speed decreases (i.e., for near-parabolic flybys).

4.1 Flyby Sensitivity

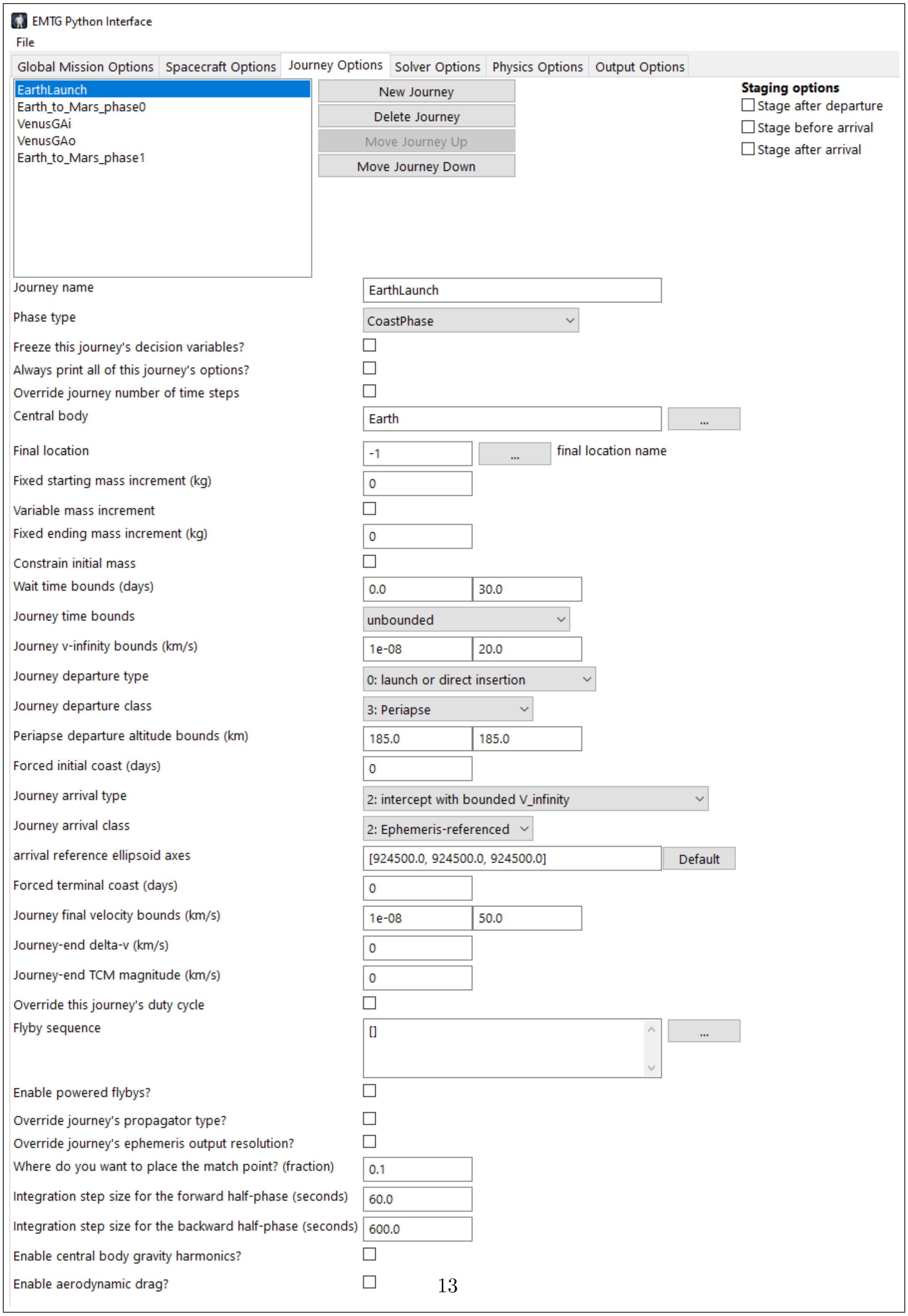
Select the “VenusGAi” (Venus gravity assist in) journey. An example is shown in Figure 9. Notice at the bottom of the Journey options, there is a section of options related to the match point and integration step size. Between each Journey departure and arrival, EMTG places a “match point” (sometimes also known as a “break point”) where the state integrated forwards from current Journey departure state and the state integrated backwards from the current Journey arrival state need to match, see Figure 2. Matching states near a gravity assist periapse is tricky because of the sensitive dynamics, and it can be difficult for SNOPT to converge if the match point is not ideally placed.

To deal with this, the conversion script overrode the default match point location (halfway in time between Journey arrival and departure) and placed the match point close to periapse. This results in a value for “Where do you want to place the match point? (fraction)” of 0.9 (90% of the way from the SOI to periapse or equivalently 10% of the way from periapse to the SOI). The conversion script has also reduced the integration step size backwards from periapse to 60 seconds rather than 600 seconds at the SOI forwards because the spacecraft is moving much faster near periapse than near the SOI boundary. Switch to the “VenusGAo” (Venus gravity assist out) Journey and notice that these values are reversed. The values pre-populated in these fields by the conversion script are rules of thumb and should be overridden by the user as appropriate for each unique scenario. These changes can be seen in Figure 8.



Where do you want to place the match point? (fraction)	0.9
Integration step size for the forward half-phase (seconds)	600.0
Integration step size for the backward half-phase (seconds)	60.0
Enable central body gravity harmonics?	☐
Enable aerodynamic drag?	☐

Figure 8: Incoming Gravity Assist Match Point Options.



EarthLaunch
Earth_to_Mars_phase0
VenusGAi
VenusGAo
Earth_to_Mars_phase1

Staging options
- ☐ Stage after departure
- ☐ Stage before arrival
- ☐ Stage after arrival

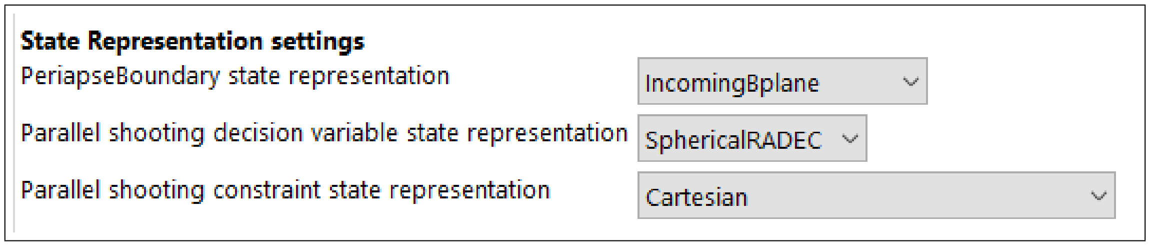
Journey name	EarthLaunch
Phase type	CoastPhase
Freeze this journey's decision variables?	☐
Always print all of this journey's options?	☐
Override journey number of time steps	☐
Central body	Earth
Final location	-1 ... final location name
Fixed starting mass increment (kg)	0
Variable mass increment	☐
Fixed ending mass increment (kg)	0
Constrain initial mass	☐
Wait time bounds (days)	0.0 / 30.0
Journey time bounds	unbounded
Journey v-infinity bounds (km/s)	1e-08 / 20.0
Journey departure type	0: launch or direct insertion
Journey departure class	3: Periapse
Periapse departure altitude bounds (km)	185.0 / 185.0
Forced initial coast (days)	0
Journey arrival type	2: intercept with bounded V_infinity
Journey arrival class	2: Ephemeris-referenced
arrival reference ellipsoid axes	[924500.0, 924500.0, 924500.0]
Forced terminal coast (days)	0
Journey final velocity bounds (km/s)	1e-08 / 50.0
Journey-end delta-v (km/s)	0
Journey-end TCM magnitude (km/s)	0
Override this journey's duty cycle	☐
Flyby sequence	[]
Enable powered flybys?	☐
Override journey's propagator type?	☐
Override journey's ephemeris output resolution?	☐
Where do you want to place the match point? (fraction)	0.1
Integration step size for the forward half-phase (seconds)	60.0
Integration step size for the backward half-phase (seconds)	600.0
Enable central body gravity harmonics?	☐
Enable aerodynamic drag?	☐

Figure 9: High Fidelity Journey Options Tab.

4.2 Periapse State Representation

Switch to the "Physics Options" tab. At the bottom of the PyEMTG window are additional options to specify how EMTG defines states in the multiple shooting problem. The "PeriapseBoundary state representation" option specifies the flyby periapse state definition. The default is "SphericalRADEC" for spherical right-ascension and declination. Click the drop-down list to see the other state representations available to define the periapse state. Sometimes switching the periapse state encoding can make it easier for SNOPT to converge on a solution.

It is possible to switch this state representation and define additional flyby constraints such as B-plane targets using this feature as shown in Figure 10. This is beyond the scope of this tutorial.


State Representation settings

PeriapseBoundary state representation: IncomingBplane

Parallel shooting decision variable state representation: SphericalRADEC

Parallel shooting constraint state representation: Cartesian

Figure 10: Example of Different State Representation Settings.

4.3 Run the Mission

Run the high-fidelity mission (Ctrl + r) with MBH using the initial guess created by the high-fidelity conversion script encoded automatically in the generated EMTG options file. You should not expect the initial guess to be feasible. However, the guess should help EMTG find a solution much faster than running this high-fidelity script without a guess due to the larger decision vector, larger constraint vector, and increased sensitivity of the problem. In practice, EMTG can often converge to a feasible high-fidelity trajectory based on this process with a single NLP solve (i.e., no MBH). Try it yourself and see!

This concludes the tutorial on high-fidelity flybys. You can now convert a low-fidelity but easy-to-solve EMTG options file using a "Flyby sequence" to a sequence of Journeys which more accurately model the spacecraft's trajectory. In the Force Models tutorial, you will learn how to increase the realism further by adding additional gravitational bodies and other perturbing forces.