National Aeronautics and Space Administration

Rapid Preliminary Design of Interplanetary Trajectories Using the Evolutionary Mission Trajectory Generator

code 595

Jacob Englander

3-14-2016

NAVIGATION & MISSION DESIGN BRANCH
NASA GSFC

www.nasa.gov

EMTG: The Team

- NASA Goddard Space Flight Center
  - Jacob Englander
  - Jeremy Knittel
- a.i. solutions
  - Matthew Vavrina
- University of Illinois at Urbana-Champaign
  - Donald Ellison
  - Ryne Beeson
  - Dr. Alexander Ghosh
  - Professor Bruce Conway
- University of Vermont
  - David Hinckley
- FBCM
  - Arnold Englander

Why an automated design tool?

- Interplanetary mission design is highly complex. Analysts must design maneuvers and choose launch dates, flight times, encounter epochs, and in many cases a flyby sequence or even a set of science targets.
- This process is even more complex for missions employing low-thrust electric propulsion because each possible spacecraft design dramatically affects the trajectory.
- Designing even a single interplanetary trajectory is an expensive process. It is even more expensive to explore a broad trade space of missions to find the relationship between science return, cost, and risk.
- An automated tool shifts much of the work from the analyst to a computer, allowing the analyst to spend more time collaborating with the science team and better understand their needs.

EMTG Design Capabilities

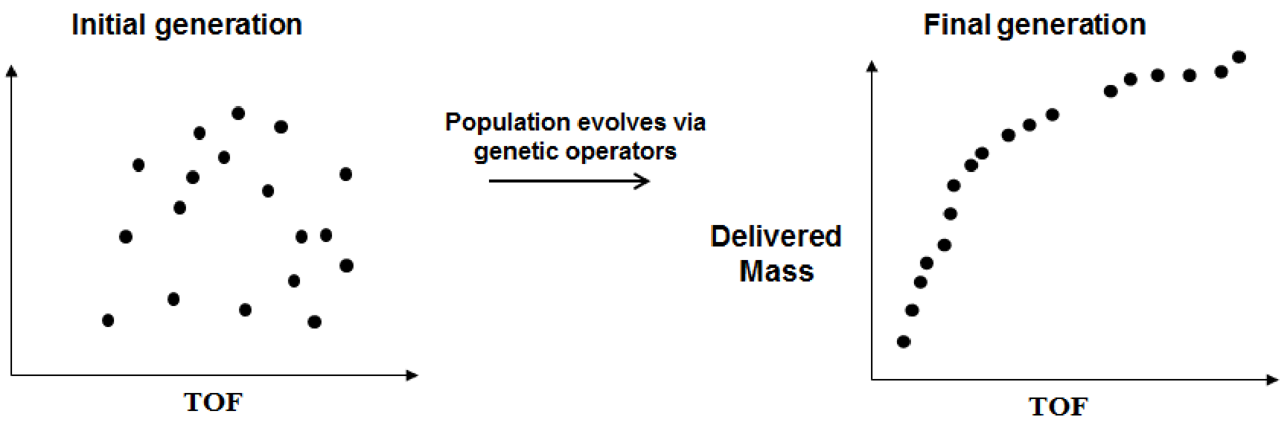
- **Propulsion Types**
  - High-thrust chemical
  - Low-thrust electric
- **Mission Components**
  - Deep-space maneuvers
  - Gravity Assists
  - Asteroid Rendezvous/Flyby
  - Sample Return/Planetary Landing
  - Launch Vehicle selection
- **Spacecraft Systems**
  - Power system sizing
  - Propulsion system sizing
- **Mission Objectives**
  - Maximize science payload
  - Minimize flight time
  - Visit as many diverse bodies as possible
  - Maximize encounter energy (for planetary defense)
- **Operational and Science Constraints**
  - Atmospheric entry
  - Solar distance
  - Any other constraints on final orbit

Mission and Systems Design via Hybrid Optimal Control

- The interplanetary mission design problem has two types of variables:
  - *Discrete* variables encoding the mission sequence and choice of spacecraft systems (launch vehicle, power, propulsion)
  - *Continuous* variables defining the trajectory
- In *Hybrid Optimal Control*, the problem is divided into two nested loops.
  - The *outer-loop* solves the discrete problem and identifies candidate missions.
  - The continuous *inner-loop* then finds the optimal trajectory for each candidate mission.
- But in this tutorial, due to time constraints, we will focus solely on the inner-loop solver

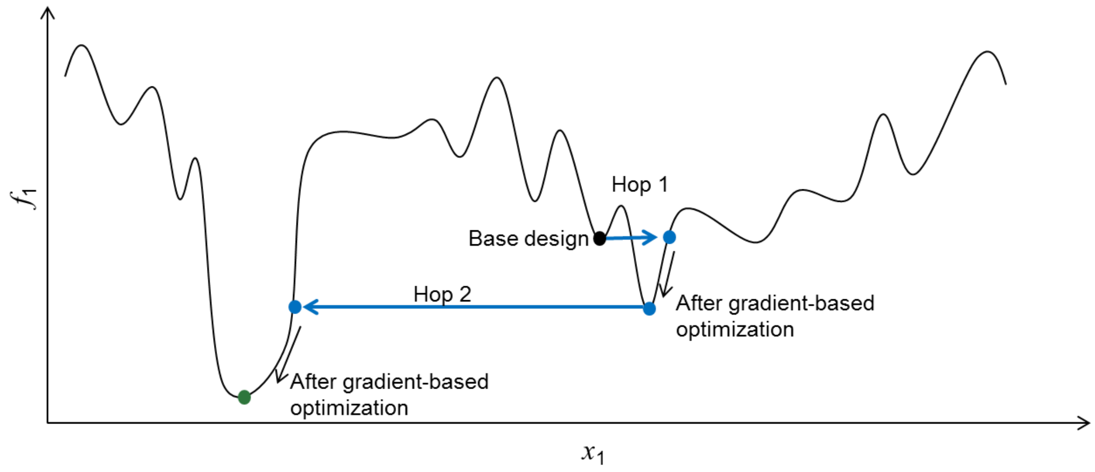
Discrete Optimization of the Mission Sequence and Spacecraft Systems

- EMTG's outer-loop finds the non-dominated set of missions, those which are not strictly better or worse than other missions in the set based on all of the analyst's chosen objective functions
- EMTG uses a version of the Non-Dominated Sorting Genetic Algorithm II (NSGAII) which can evolve to the final non-dominated trade front despite starting from complete randomness. No *a priori* knowledge of the solution is required.



Trajectory Optimization via Monotonic Basin Hopping and Nonlinear Programming

- EMTG's inner-loop finds the optimal trajectory using a stochastic global search method called Monotonic Basin Hopping (MBH) coupled with a gradient-based local search supplied by the third-party Sparse Nonlinear Optimizer (SNOPT).
- EMTG does not require an initial guess and can find the global optimum autonomously.



Some nuts and bolts

- EMTG is available open-source at https://sourceforge.net/projects/emtg/
- EMTG is written in C++ and the user interface, PyEMTG, is written in Python 2.7
  - Why Python 2.7? That's what I started with. We welcome collaborators who want to take on the challenge of migrating us to Python 3!
- EMTG is script-driven; all PyEMTG does is write scripts and read output files
- To compile EMTG, one also needs the Boost C++ extensions, the CSPICE ephemeris library, and most importantly SNOPT.
- PyEMTG depends on wxPython
- If one is using the outer-loop, it is possible to link EMTG to MPI for parallel processing. The inner-loop is serial.
- EMTG is happily cross-platform and is used operationally on OSX, Windows, and Linux.

Let's play with EMTG…

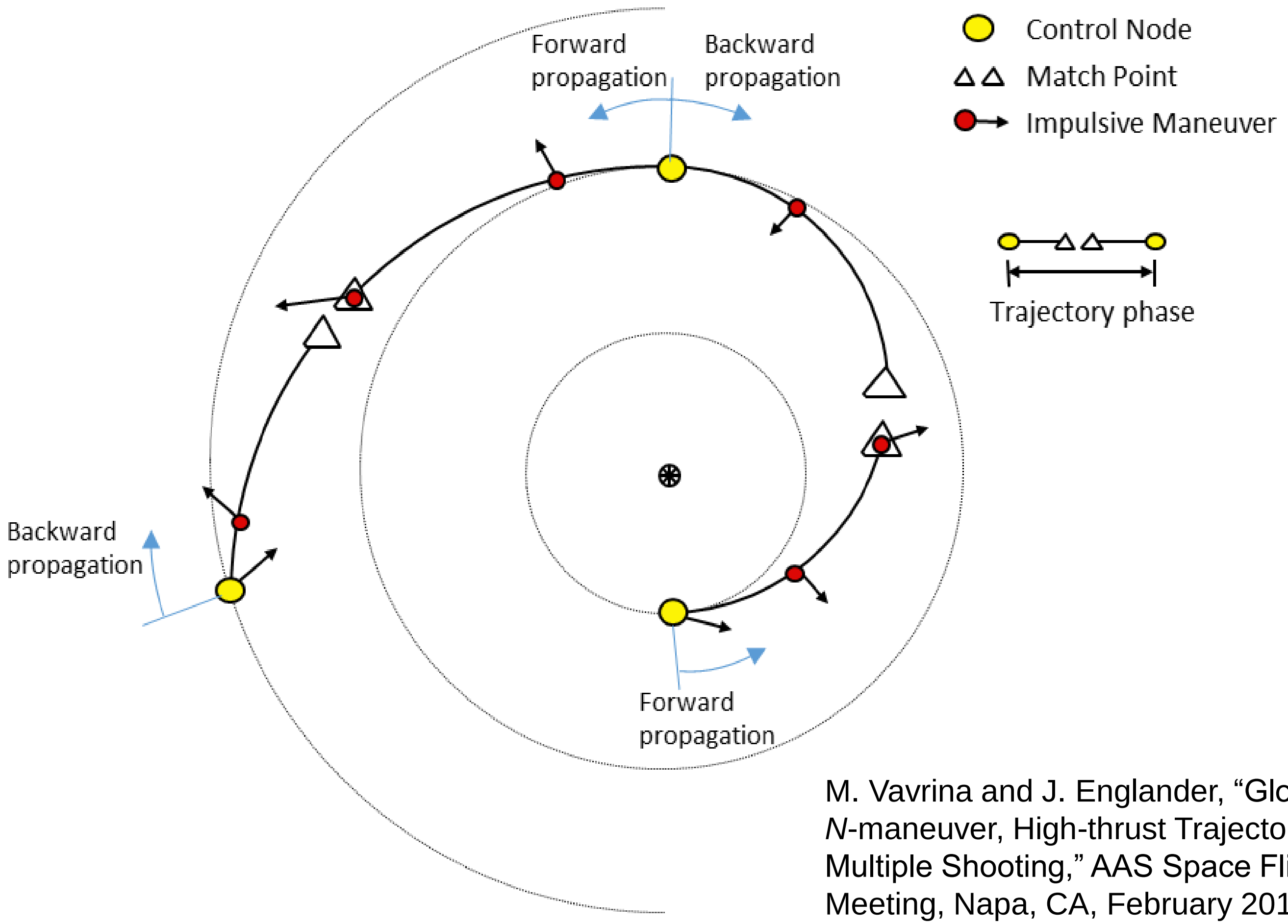
- Today we are going to design two missions in EMTG
  - OSIRIS-REx
    - A New Frontiers class mission to acquire a sample from near-Earth asteroid Bennu and return it to Earth
  - LowSIRIS-REx
    - Academic problem which is very similar to OSIRIS-REx but uses low-thrust electric propulsion
- In the next several slides, we will walk you through how to create these missions in EMTG
- Start your PyEMTG…

Configuring PyEMTG

- Edit your PyEMTG.options to ensure that PyEMTG has paths to:
  - EMTG executable
  - Universe folder (where all ephemeris-related things live)
  - Your small bodies file (optional file for searching for targets of opportunity)

```
EMTG_path D:\Projects\EMTGv8_sourceforge\branch\Jacob_MGANDSM_and_GMAT\bin\emtg.exe
default_universe_path D:\Projects\EMTGv8_sourceforge\EMTG_libraries\Universe
default_small_bodies_file D:\Projects\EMTGv8_sourceforge\EMTG_libraries\Universe\ephemeris_files\AllAsteroids.SmallBody
```

Chemical Mission Modeling in EMTG



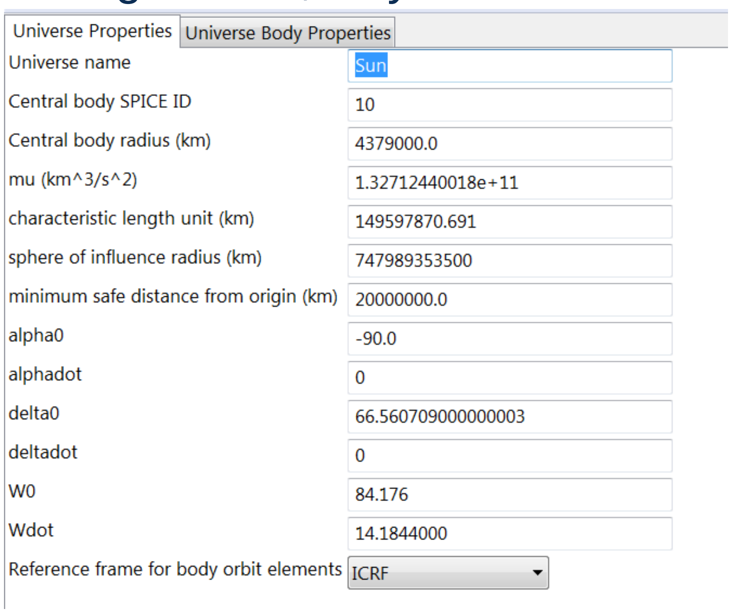
M. Vavrina and J. Englander, "Global Optimization Of *N*-maneuver, High-thrust Trajectories Using Direct Multiple Shooting," AAS Space Flight Mechanics Meeting, Napa, CA, February 2016.

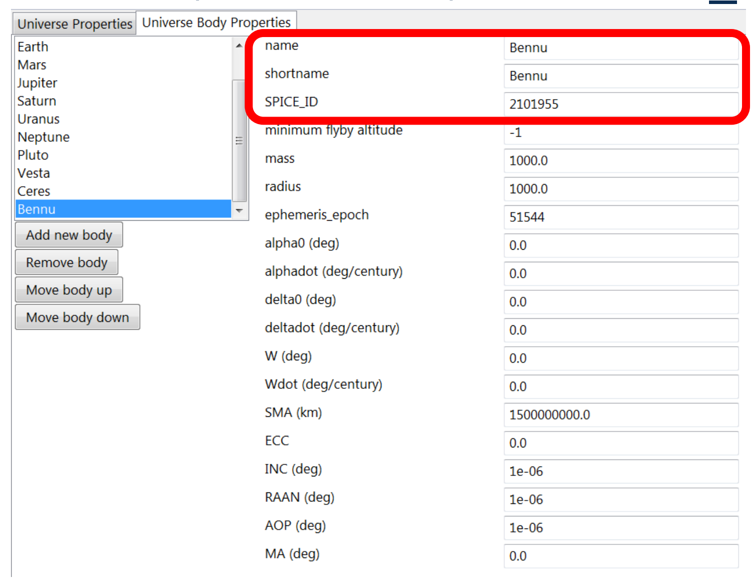
OSIRIS-REx Step 1a: Acquire body ephemeris data

- EMTG is most accurate when a SPICE kernel is provided for each body of interest
- You can create a SPICE kernel for any body in the HORIZONS database at: http://ssd.jpl.nasa.gov/x/spk.html
  - Place SPICE kernels in your 'EMTG_root/Universe/ephemeris_files/' directory
- Alternatively you may specify Keplerian orbit elements. This results in faster EMTG execution but less accuracy.

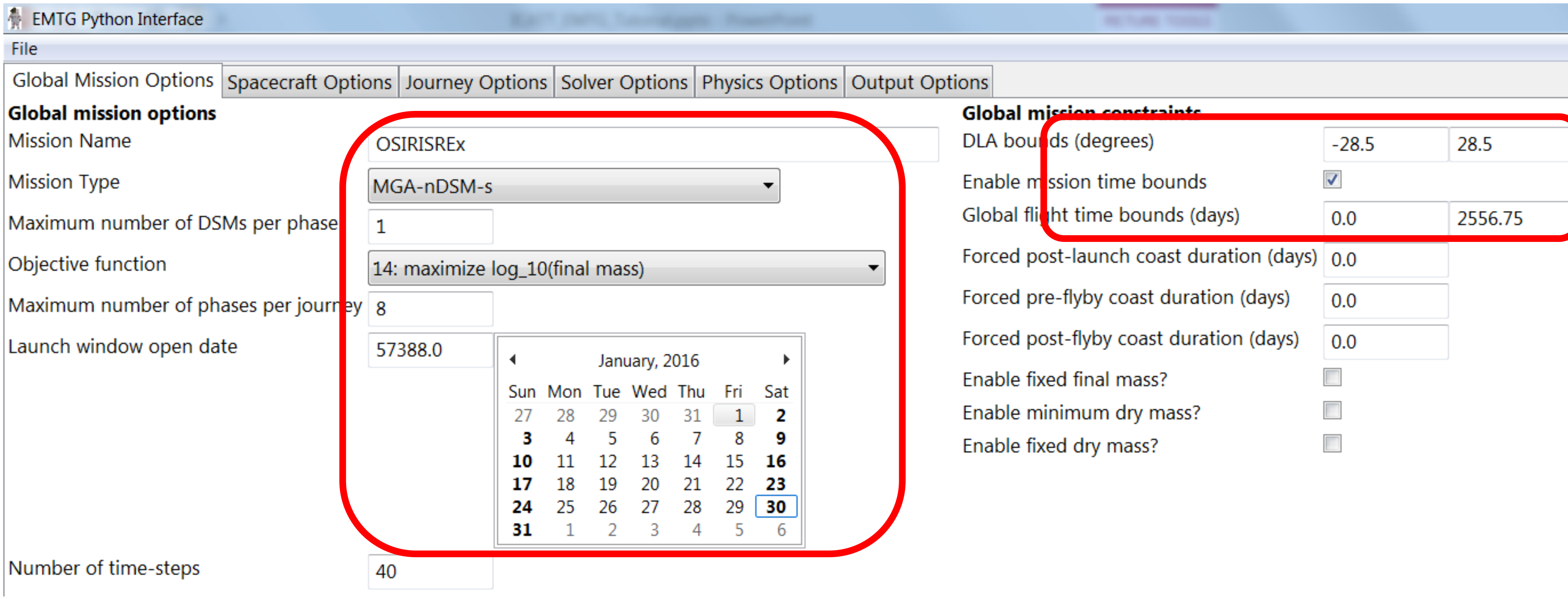
OSIRIS-REx Step 1b: Create a Universe file to teach EMTG about your body(ies)

- Users make EMTG aware of a body by putting entering it into a “Universe” file
- Every journey in an EMTG mission can have a different Universe file if you so desire but you can use on Universe file for the whole mission.
- Create a body and tell EMTG your new body’s SPICE ID and name. If you are using SPICE, all you need to enter is name, shortname, and SPICE_ID.



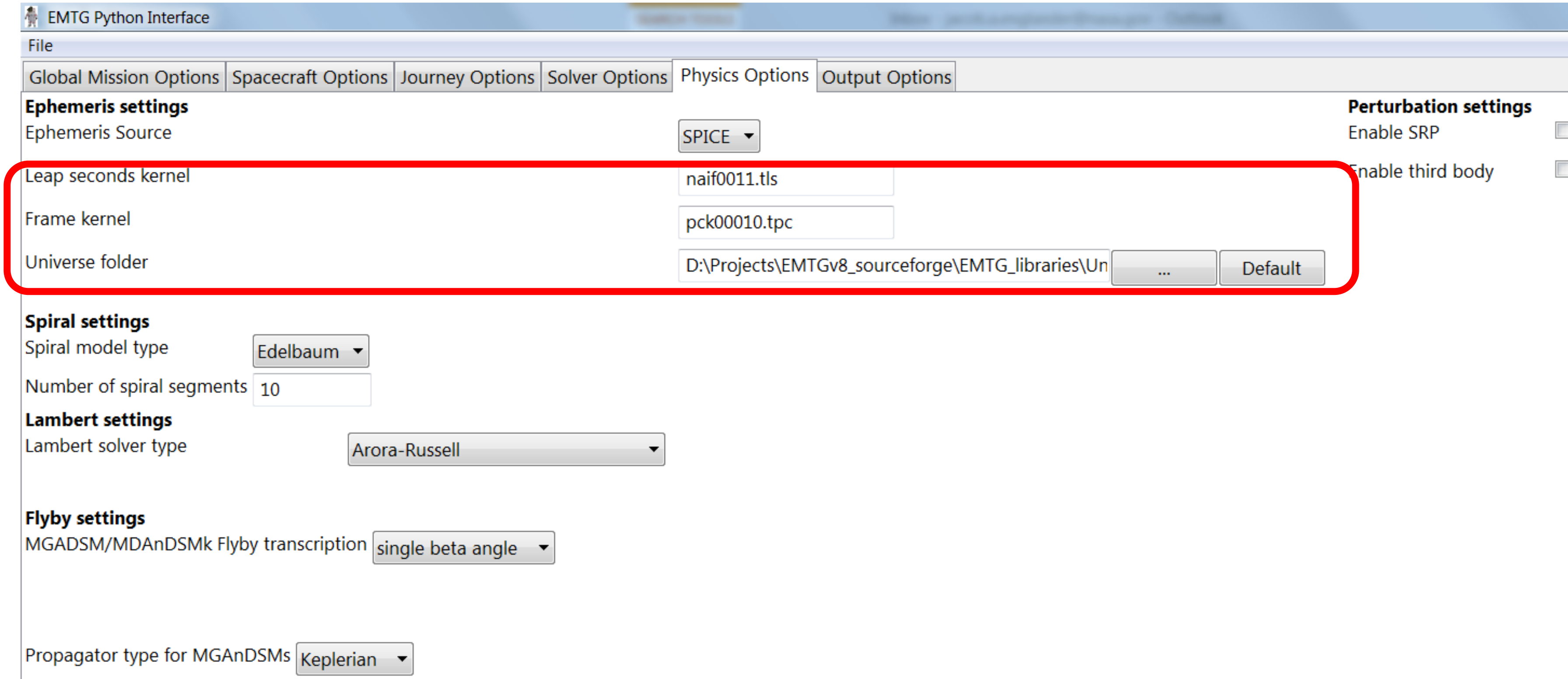


OSIRIS-REx Step 2: The Global Options Tab



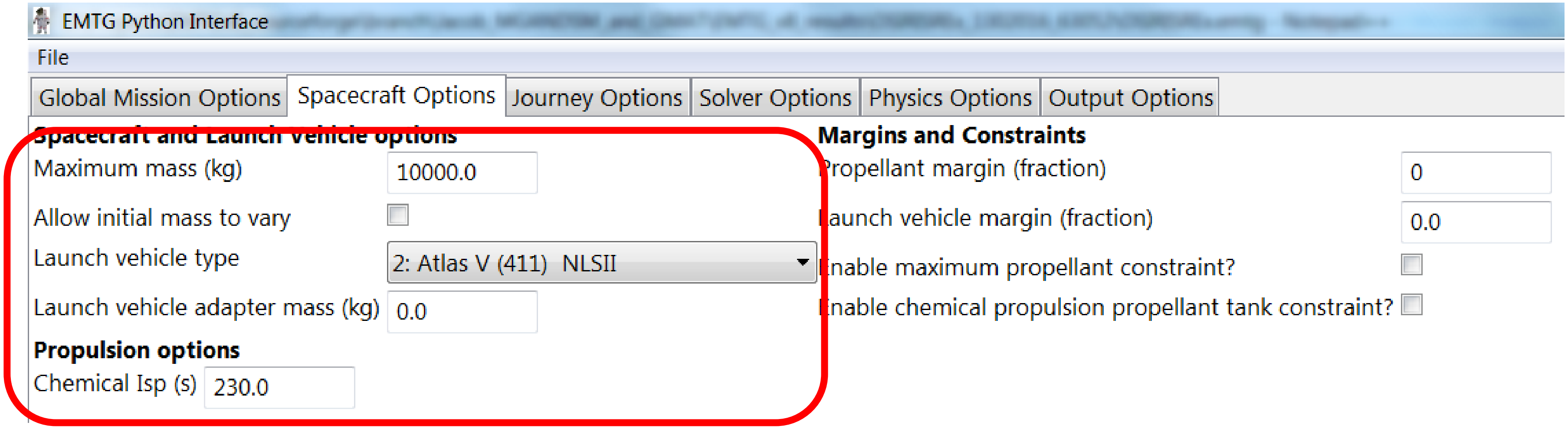
- We recommend the MGAnDSMs transcription for chemical missions
- There are many objective functions. Log10(final mass) works very well for chemical missions.
- PyEMTG does math! You can, for example, write “365.25 * 7” in the flight time bounds field.
- For chemical missions, the “number of time-steps” field only affects the resolution of the final trajectory plot and not the solution itself.

OSIRIS-REx Step 3: The Physics Options Tab



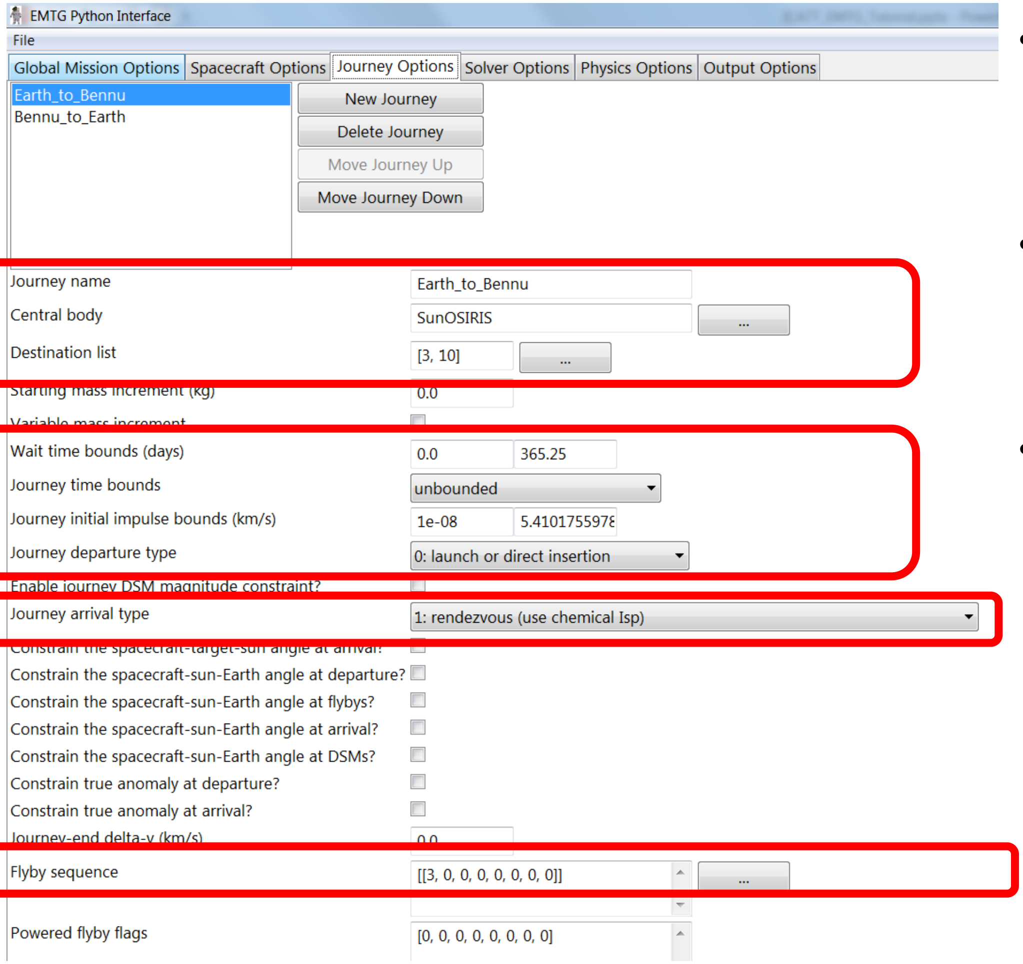
- Make sure your script points to the correct Universe folder or else nothing will work. If you configured PyEMTG.options correctly you can just click the “Default” button.
- Third body and solar radiation pressure perturbations only work if your propagator is set to “Integrator.” This is very slow and unnecessary for this stage of design.

OSIRIS-REx Step 4: The Spacecraft Options Tab



- In this case we are optimizing mass, not Δv, so a suitable launch vehicle model and chemical thruster I_{sp} must be chosen. EMTG will trade between the thruster and the launch vehicle to deliver the most mass to the target.
- The “maximum mass” field in this case just scales the optimization problem; the actual launch mass is limited to whatever the launch vehicle can carry to the C3 that the optimizer chooses.

OSIRIS-REx Step 5a: The Journey Options Tab (Journey 1)



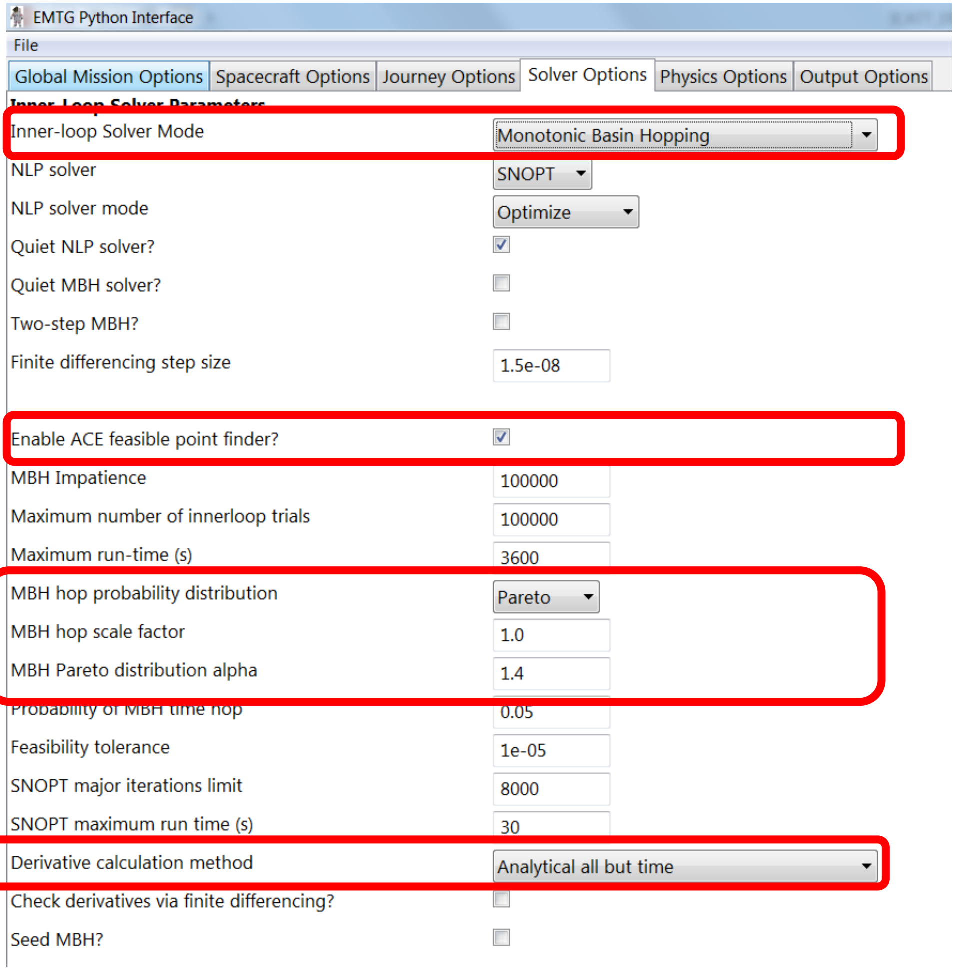
- Each “journey” represents a set of events with user-defined boundary conditions – in this case launch from Earth and arrival at Bennu.
- There are many types of journey departure and arrival conditions. The ones shown here are appropriate for a small body rendezvous.
- Each journey can contain any number of phases. If a journey has more than one phase, they are separated by planetary flybys. In this case we have selected an Earth flyby.

OSIRIS-REx Step 5b: The Journey Options Tab (Journey 2)

Journey name	Bennu_to_Earth	
Central body	SunOSIRIS	...
Destination list	[10, 3]	...
Starting mass increment (kg)	0.0	
Variable mass increment	☐	
Wait time bounds (days)	365.25	730.5
Bounded journey departure date?	☐	
Journey time bounds	unbounded	
Journey initial impulse bounds (km/s)	1e-08	10.0
Journey departure type	0: launch or direct insertion	
Enable journey DSM magnitude constraint?	☐	
Journey arrival type	2: intercept with bounded V_infinity	
Forced pre-intercept coast (days)	0.0	
Journey final velocity bounds	0.0	10
Apply arrival declination constraint?	☐	
Atmospheric interface flight path angle (degrees)	-8.0	
Atmospheric entry interface radius (km)	6503.0	
Atmospheric entry latitude (degrees)	40.0	
Apply atmospheric entry interface azimuth constraint?	☐	
Apply atmospheric entry interface sun angle constraint?	☐	
Apply atmospheric entry interface longitude constraint?	☐	
Apply entry velocity with respect to rotating atmosphere constraint?	☑	
Bounds on entry interface velocity with respect to rotating atmosphere (km/s)	0.0	12.4

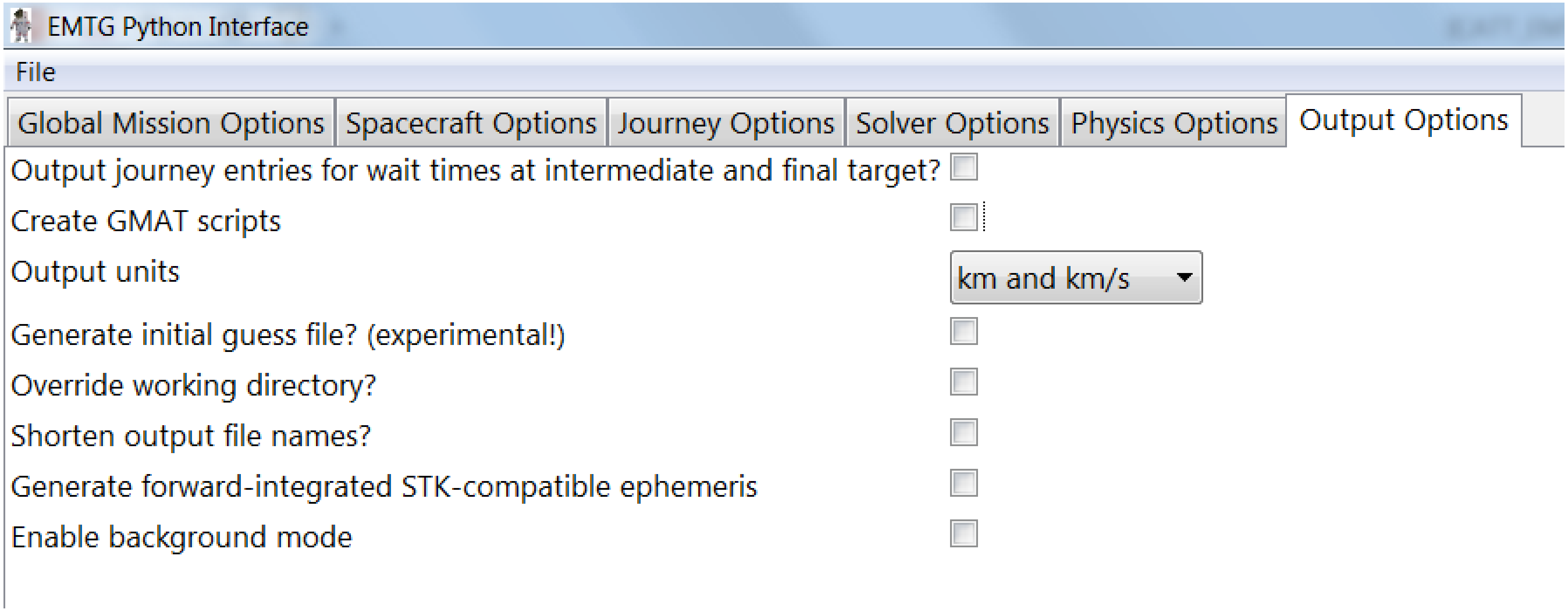
- We constrain OSIRIS-REx to spend between 1 and 2 years at Bennu.
- OSIRIS-REx ends with a landing on Earth. EMTG can constrain – using a simplified geometric model – the conditions at which the spacecraft intersects the Earth's atmosphere.
- In this example, we constrain the velocity of the spacecraft relative to the rotating atmosphere at interface.
- We must specify flight path angle, altitude, and latitude.

OSIRIS-REx Step 6: The Solver Options Tab



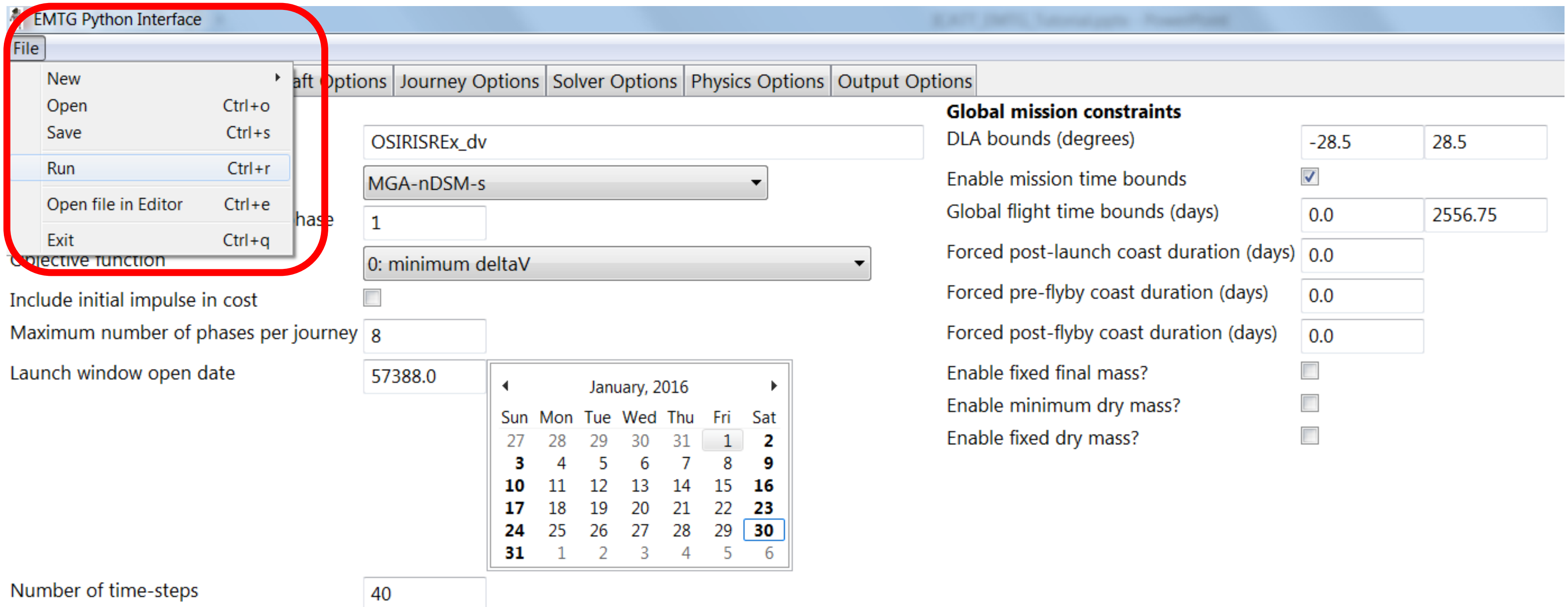
- For this example we will use Monotonic Basin Hopping (MBH) with a Pareto distribution. For chemical missions we recommend setting the “Pareto alpha” to 1.4 or, if the problem has many local optima, 1.3.
- The ACE feasible point finder allows MBH to compare infeasible solutions and search for minimum infeasibility before it finds its first feasible solution and begins to search for optimality.
- EMTG has full analytical derivatives for the MGAnDSMs transcription. However we recommend that you run with time derivatives turned off because, since SPICE does not provide smooth derivatives and must be finite-differenced, the analytical time derivatives are not accurate unless SPICE is not used.

OSIRIS-REx Step 7
The Output Options Tab



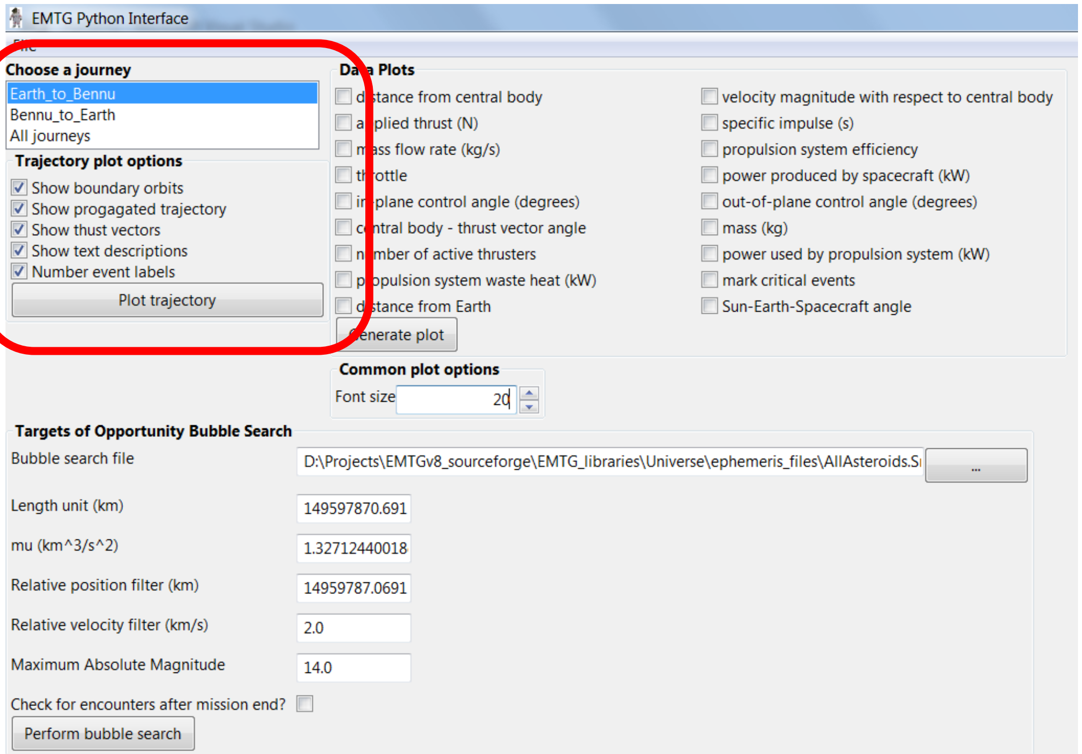
- We don’t need any of these right now.
- For the purpose of this tutorial it is sufficient to know that “background mode” means “close EMTG as soon as it is done executing.” If you are running EMTG from PyEMTG then you should leave background mode off so that you can see your results more easily.

OSIRIS-REx Step 8: Run your mission!



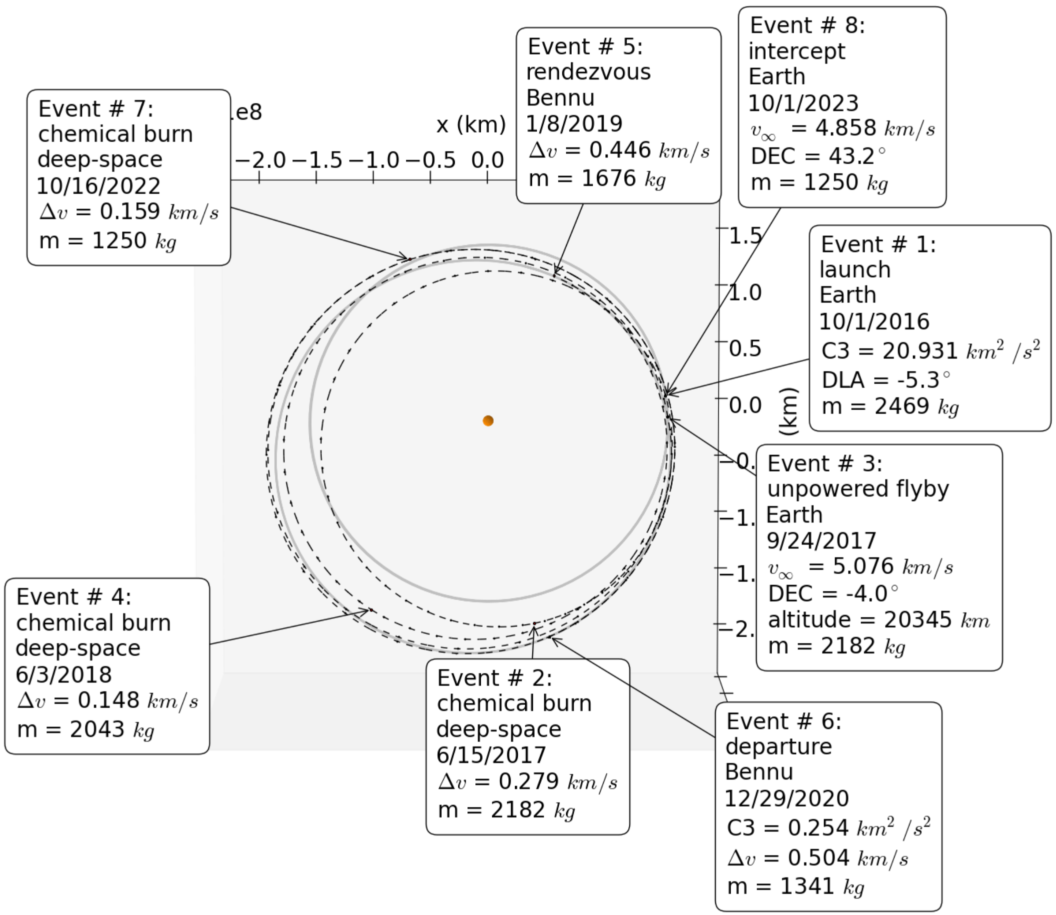
- PyEMTG will prompt you to save a ".emtgopt" options file.
- EMTG will then execute, create a new timestamped subdirectory for your results, and begin solving your problem.

OSIRIS-REx Step 9: Post-Process Your Mission



- PyEMTG can open ".emtg" options files.
- PyEMTG can create trajectory plots and systems summary plots.
- We will focus on the trajectory plot for this example.
- PyEMTG can also search for targets of opportunity along your trajectory. This is out of scope for the tutorial.

OSIRIS-REx Trajectory Plot

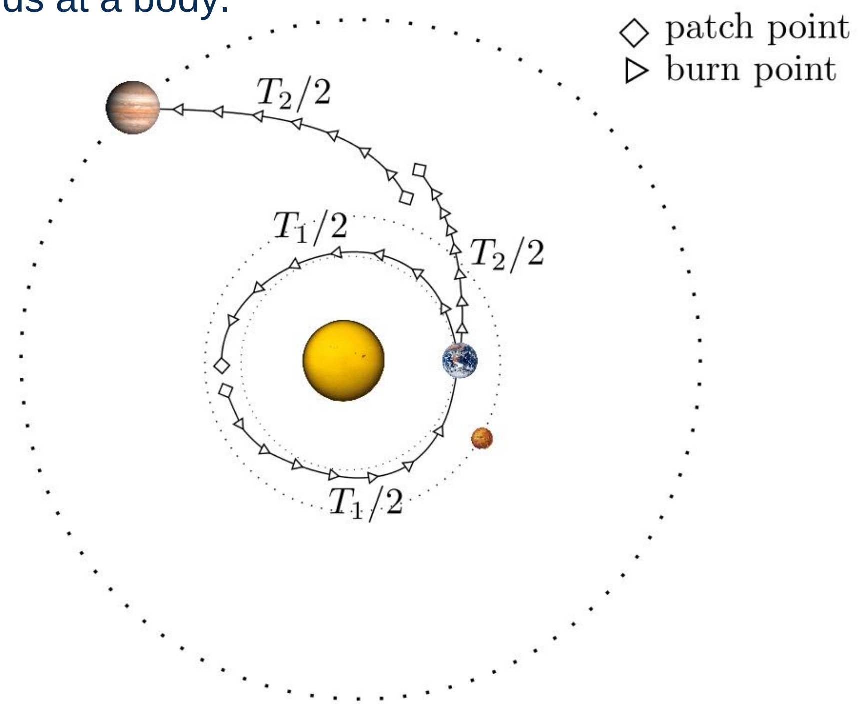


Interlude: What makes Low-Thrust Different?

- Low-thrust electric propulsion is characterized by high power requirements but also very high specific impulse (I_{sp}), leading to very good mass fractions
- Low-thrust trajectory design is a very different process from chemical trajectory design
  - Like chemical design, must find the optimal launch date, flight time, and dates of each flyby (if applicable)
  - Unlike chemical design, must find a time-history of thrust control for the entire mission
- ***Low-thrust electric propulsion mission design requires accurate modeling of propulsion and power systems. Every spacecraft design drives a unique trajectory design!***
- Audience members who are familiar with ESA's PaGMO will see many similarities between it and EMTG. This is intentional. EMTG's inner-loop was very much inspired by PaGMO although EMTG models a much wider range of mission types. If there are any PaGMO team members in the audience, we greatly appreciate you and cite you as often as we can!

Low-Thrust Modeling in EMTG Transcription

- Break mission into phases. Each phase starts and ends at a body.
- Sims-Flanagan Transcription
  - Break phases into time steps
  - Insert a small impulse in the center of each time step, with bounded magnitude
  - Optimizer Chooses:
    - Launch date
      - For each phase:
        - Initial velocity vector
        - Flight time
        - Thrust-impulse vector at each time step
        - Mass at the end of the phase
        - Terminal velocity vector
- Assume two-body force model; propagate by solving Kepler's problem
- Propagate forward and backward from phase endpoints to a "match point"
- Enforce nonlinear state continuity constraints at match point
- Enforce nonlinear velocity magnitude and altitude constraints at flyby



J. Englander, D. Ellison, and B.Conway, "Global Optimization of Low-Thrust, Multiple-Flyby Trajectories at Medium and Medium-High Fidelity," AAS Space Flight Mechanics Meeting, Santa Fe, NM, January 2014.

Low-Thrust Modeling in EMTG Spacecraft and Launch Vehicle Models

- Medium-fidelity mission design requires accurate hardware modeling
- Launch vehicles are modeled using a polynomial fit

$$m_{delivered} = (1 - \sigma_{LV})\left(a_{LV} C_3^5 + b_{LV} C_3^4 + c_{LV} C_3^3 + d_{LV} C_3^2 + e_{LV} C3 + f_{LV}\right)$$

where σ_{LV} is launch vehicle margin and C_3 is hyperbolic excess velocity

- Thrusters are modeled using either a polynomial fit to published thrust and mass flow rate data

$$\dot{m} = a_F P^4 + b_F P^3 + c_F P^2 + d_F P + e_F$$
$$T = a_T P^4 + b_T P^3 + c_T P^2 + d_T P + e_T$$

or, when detailed performance data is unavailable

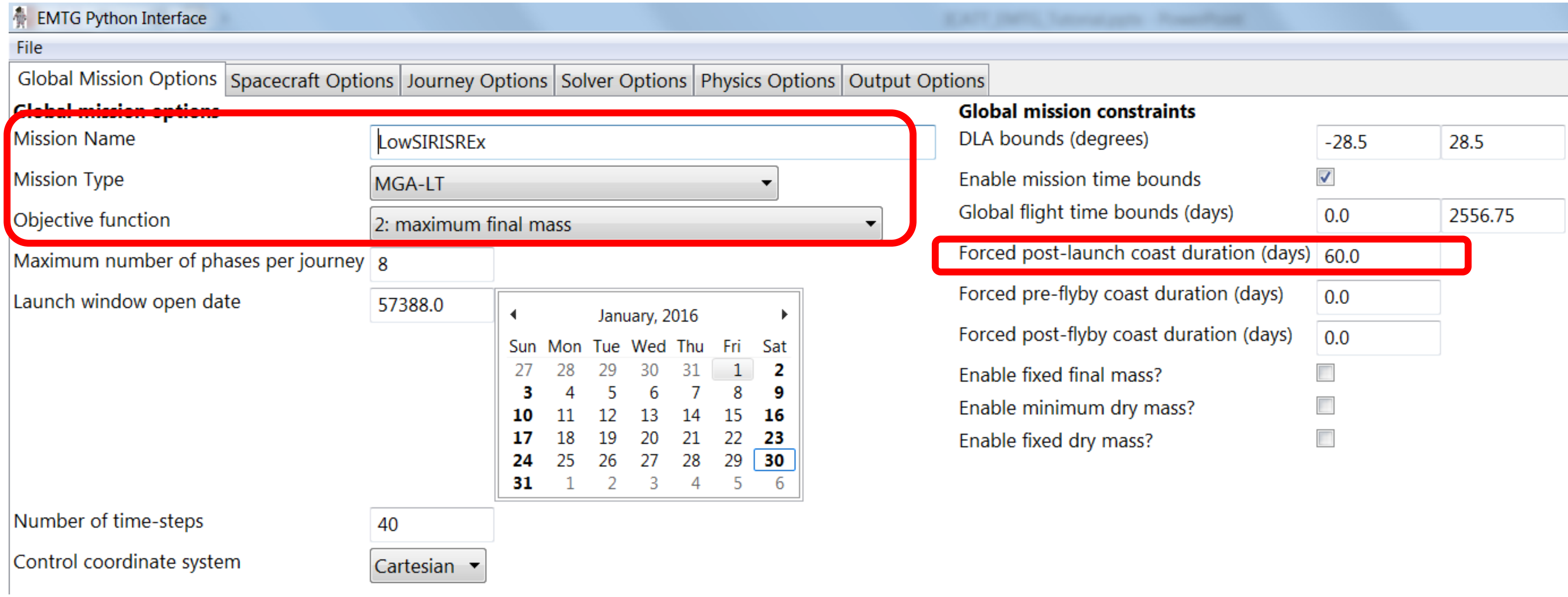
$$T = \frac{2 \eta P}{I_{sp} g_0}$$

- Power is modeled by a standard polynomial model

$$\frac{P_0}{r^2}\left(\frac{\gamma_0 + \frac{\gamma_1}{r} + \frac{\gamma_2}{r^2}}{1 + \gamma_3 r + \gamma_4 r^2}\right)(1-\tau)^t$$

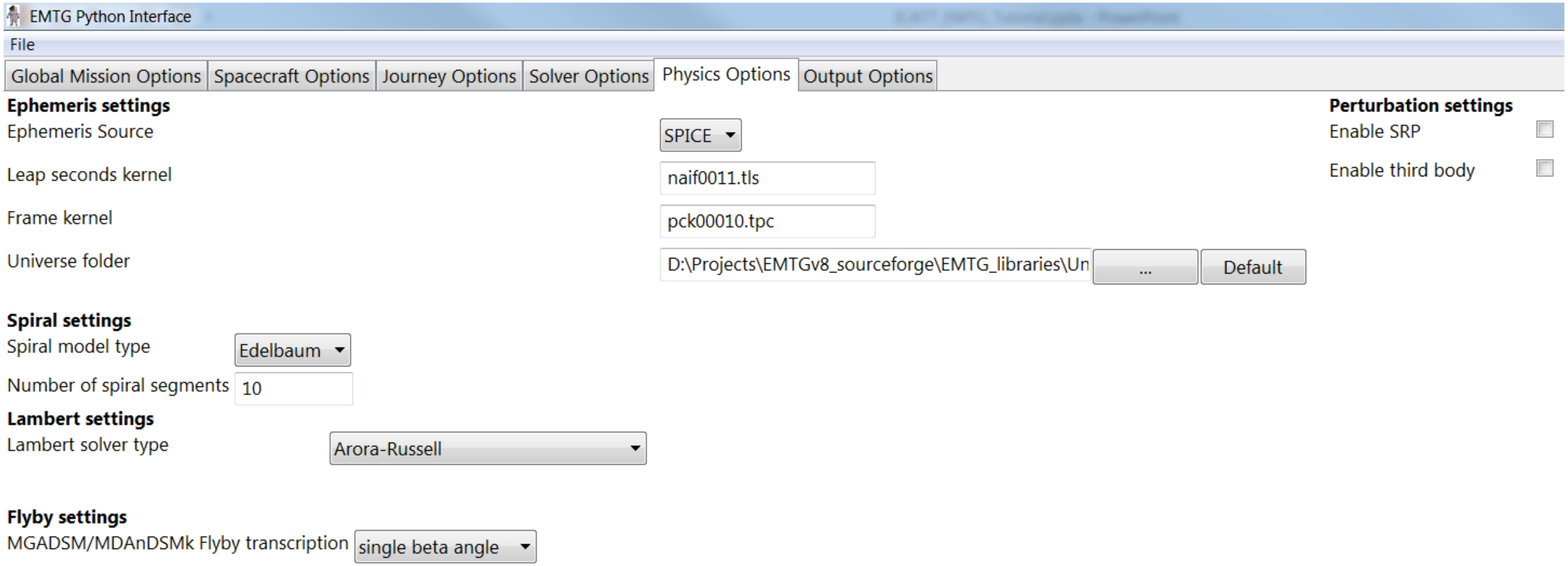
where P_0 is the power at beginning of life at 1 AU and τ is the solar array degradation constant

LowSIRIS-REx Step 1: The Global Options Tab



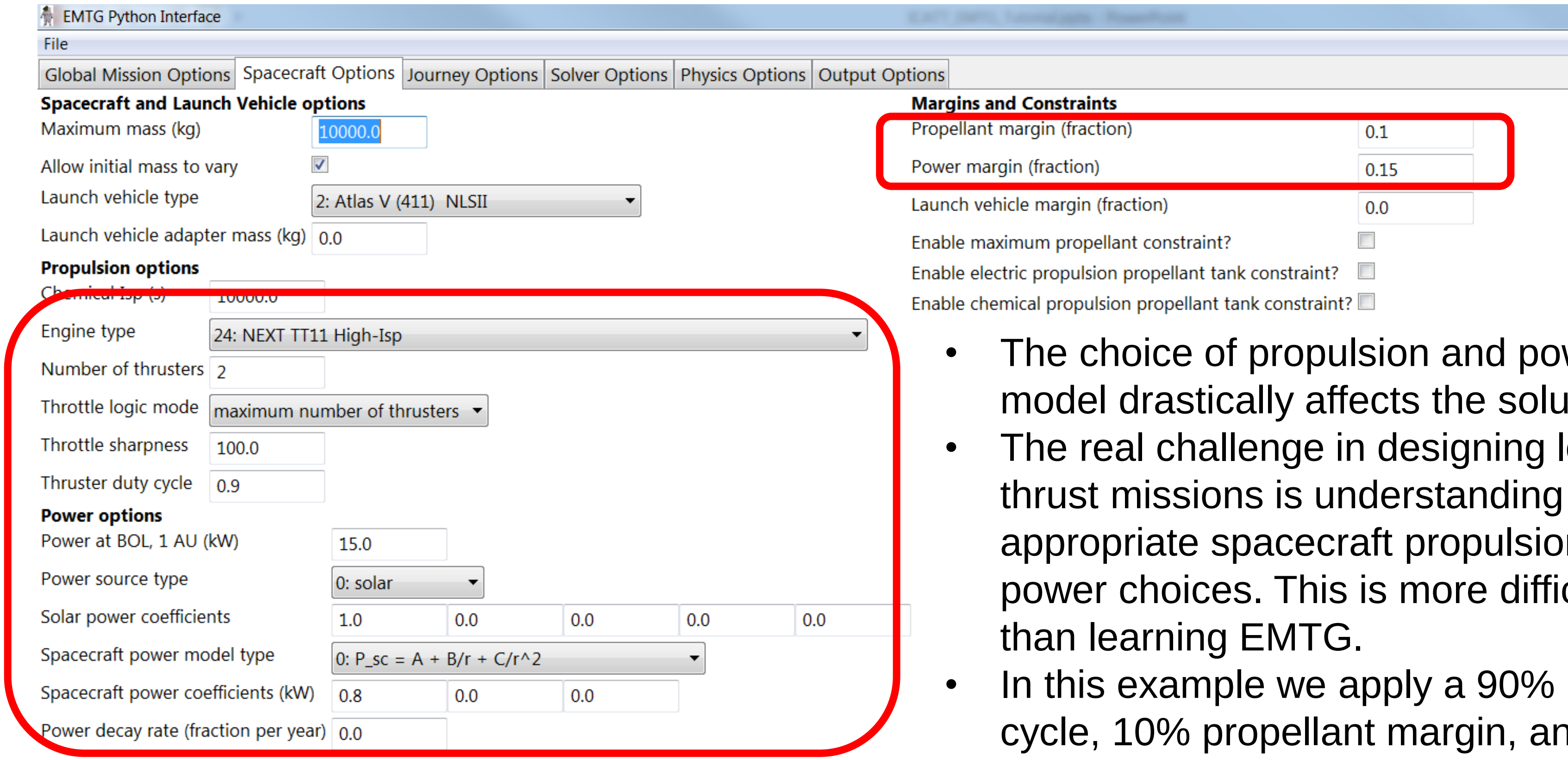
- Change the transcription to MGALT
- Change the objective function to “maximum final mass”
- Add a forced coast of 60 days after launch before the electric thrusters can used to maneuver. This period is used for spacecraft testing and is standard for NASA preliminary design.

LowSIRIS-REx Step 2: The Physics Options Tab



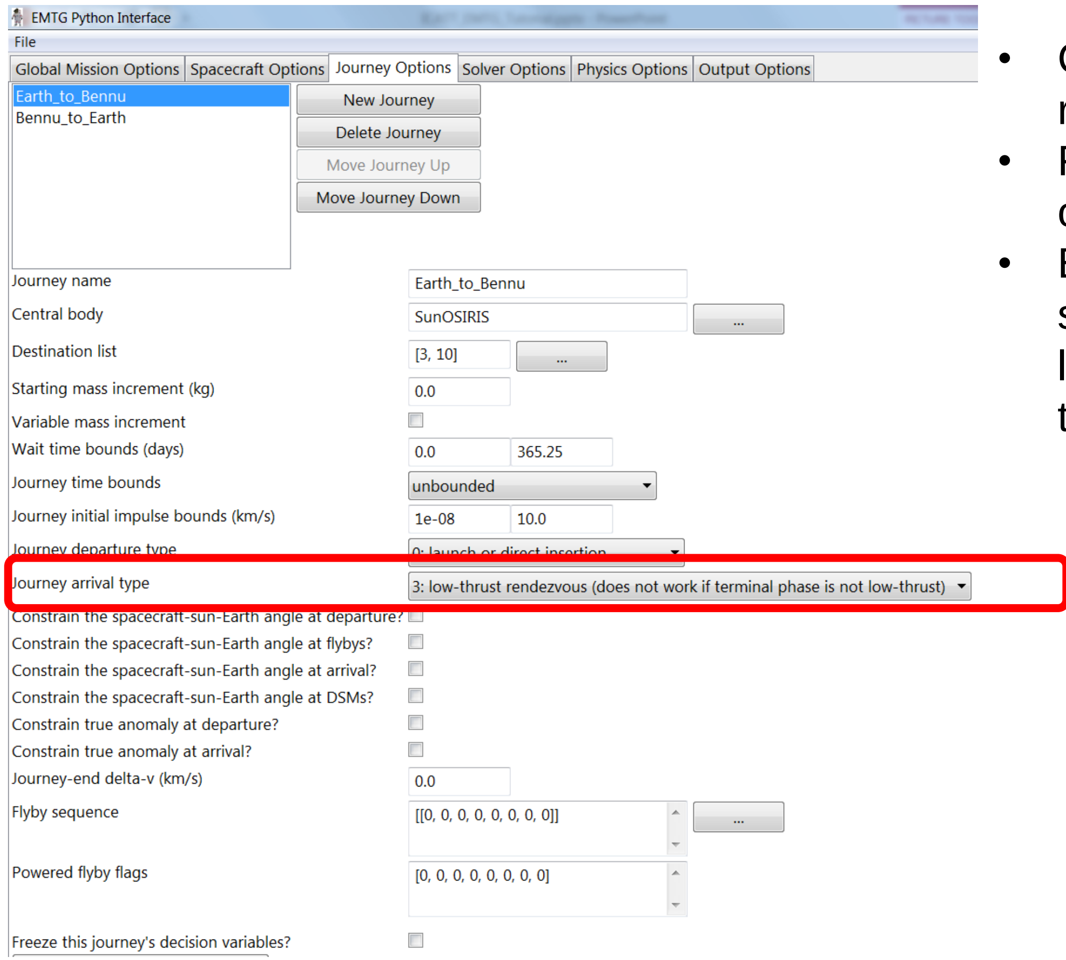
- No changes here.

LowSIRIS-REx Step 3: The Spacecraft Options Tab



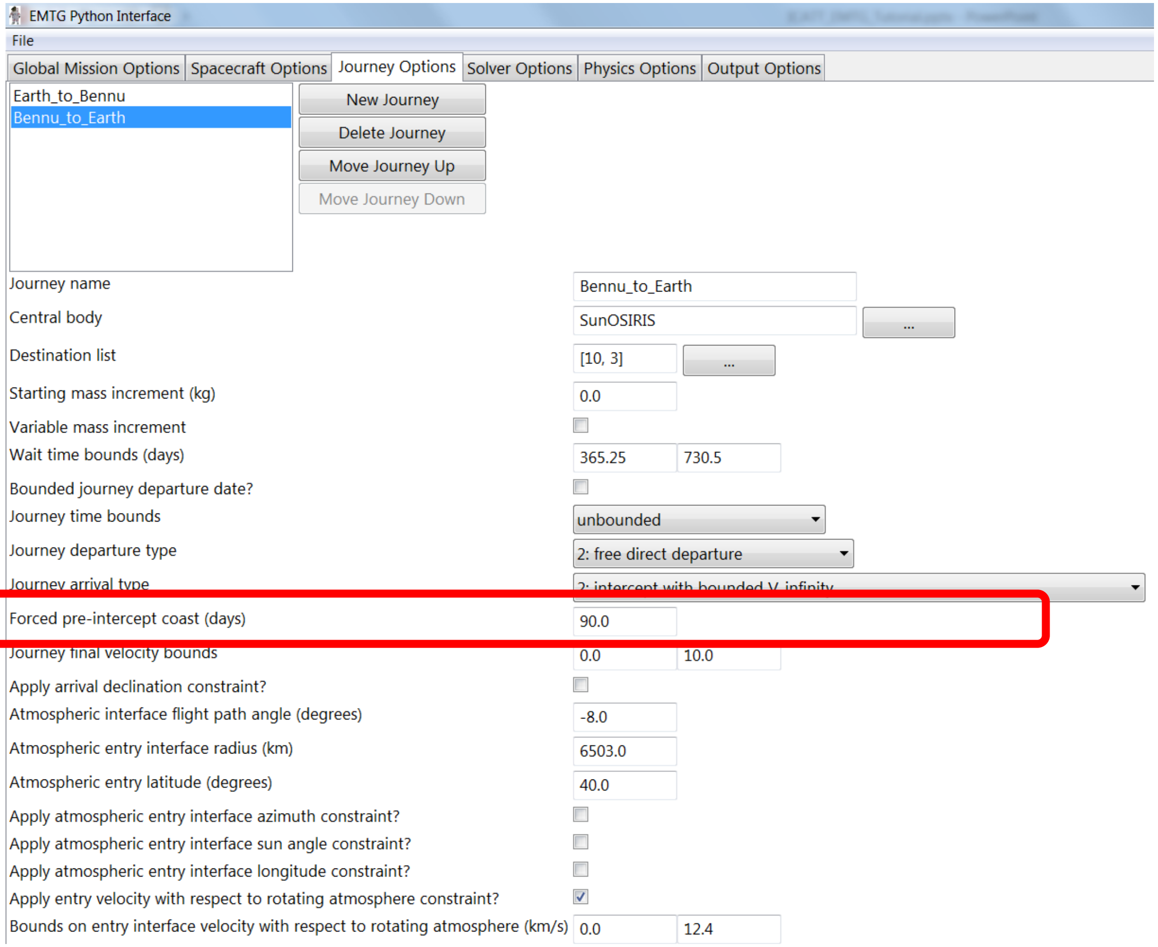
- The choice of propulsion and power model drastically affects the solution.
- The real challenge in designing low-thrust missions is understanding the appropriate spacecraft propulsion and power choices. This is more difficult than learning EMTG.
- In this example we apply a 90% duty cycle, 10% propellant margin, and 15% power margin which are standard for NASA preliminary design.

LowSIRIS-REx Step 4a: The Journey Options Tab



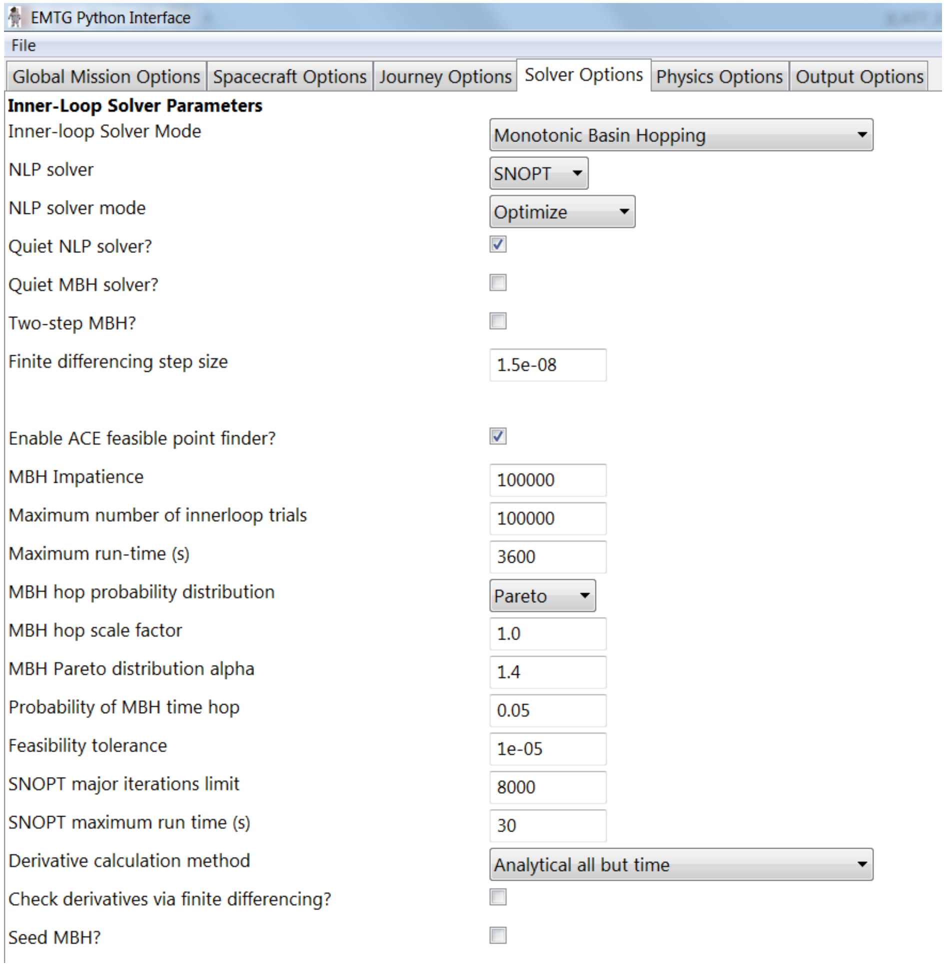
- Change the arrival type from chemical rendezvous to low-thrust rendezvous.
- For LowSIRIS-REx, an Earth flyby is not optimal and so this is a direct mission.
- EMTG can determine the optimal flyby sequence for you by using the outer-loop solver, but that is not part of this tutorial.

LowSIRIS-REx Step 4b: The Journey Options Tab



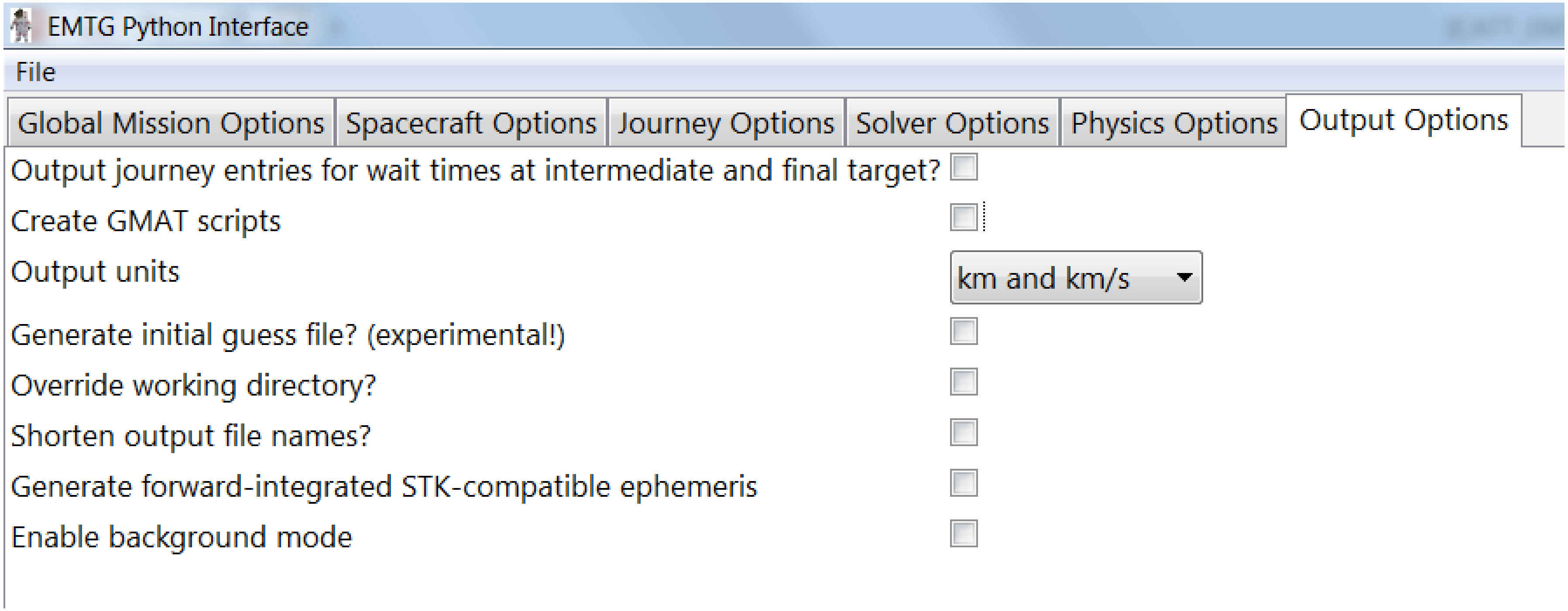
- A forced coast of 90 days is inserted prior to Earth return. The spacecraft can make small adjustments using chemical thrusters during that time but the electric propulsion system will not be used.

LowSIRIS-REx Step 5: The Solver Options Tab



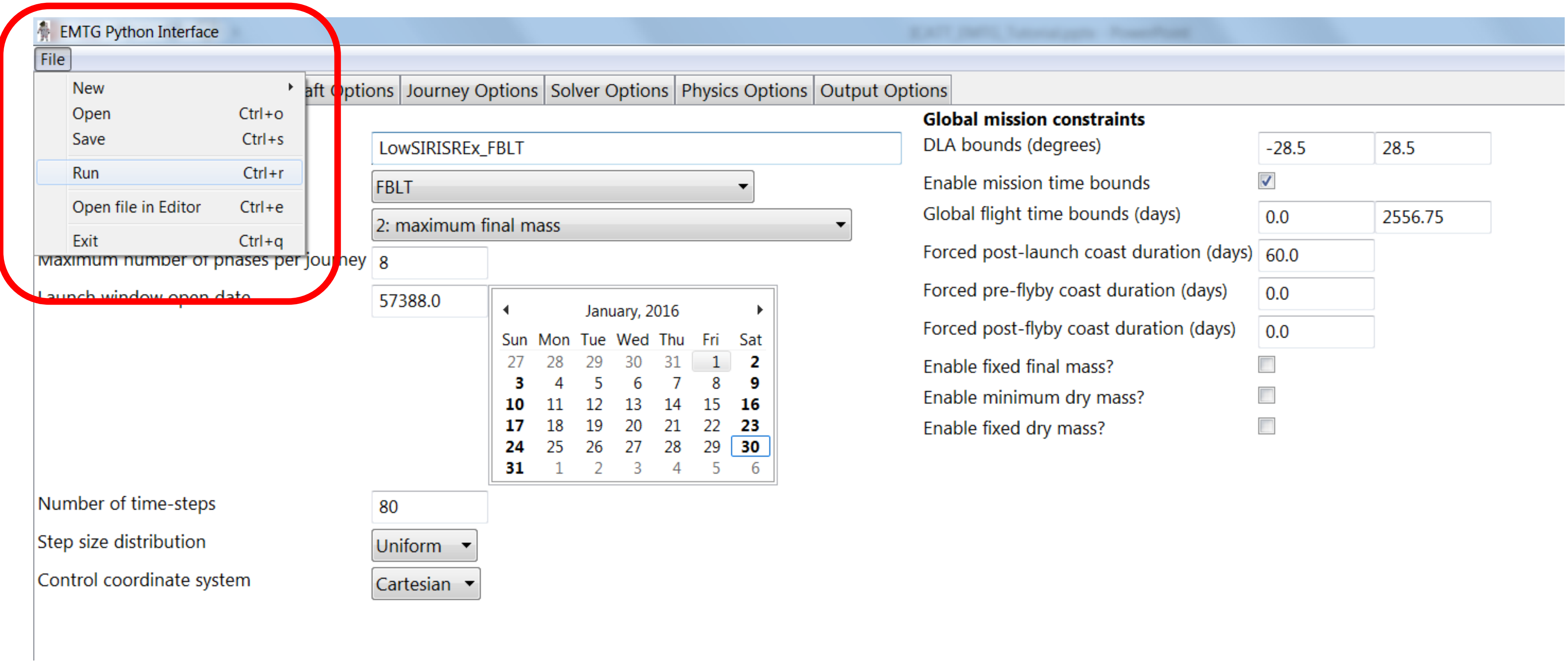
- No changes here, although in general a higher value of Pareto alpha – 1.4 or 1.5, is appropriate for low-thrust missions.

LowSIRIS-REx Step 6
The Output Options Tab

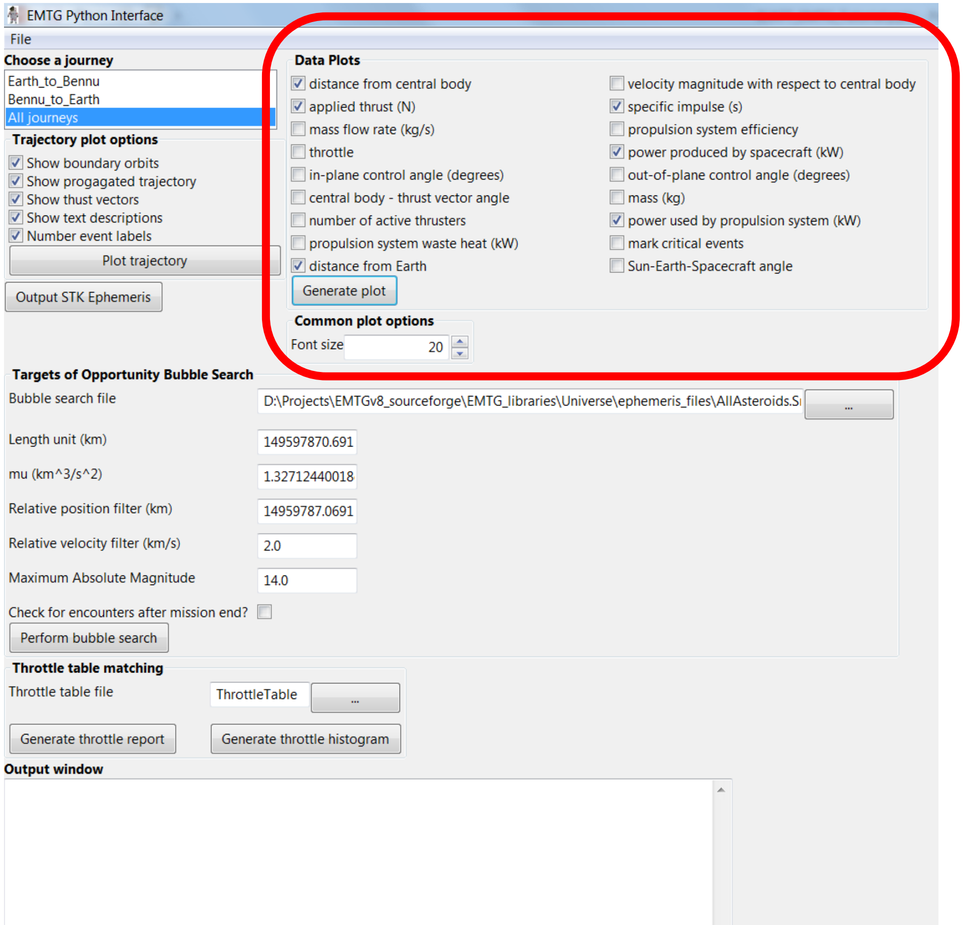


- No changes here.

LowSIRIS-REx Step 7:
Run your mission!

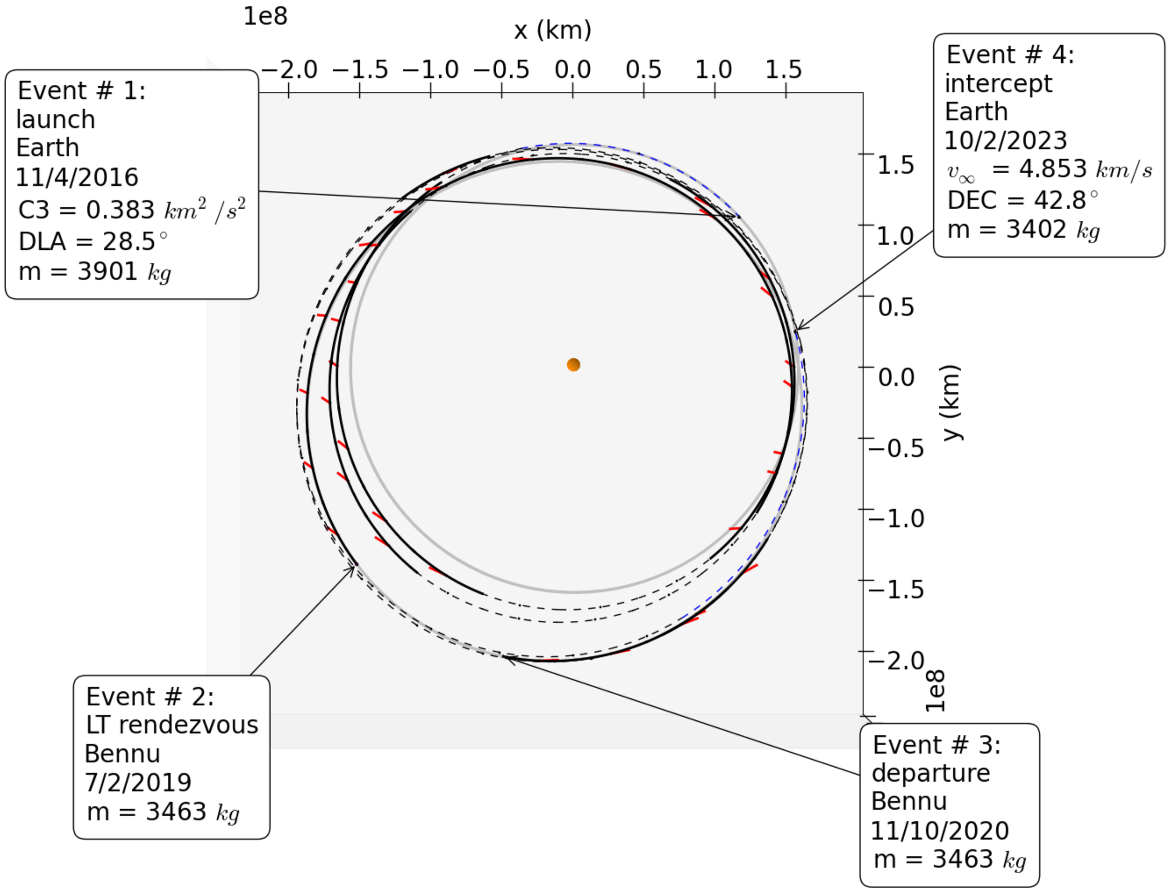


LowSIRIS-REx Step 8: Post-Process Your Mission

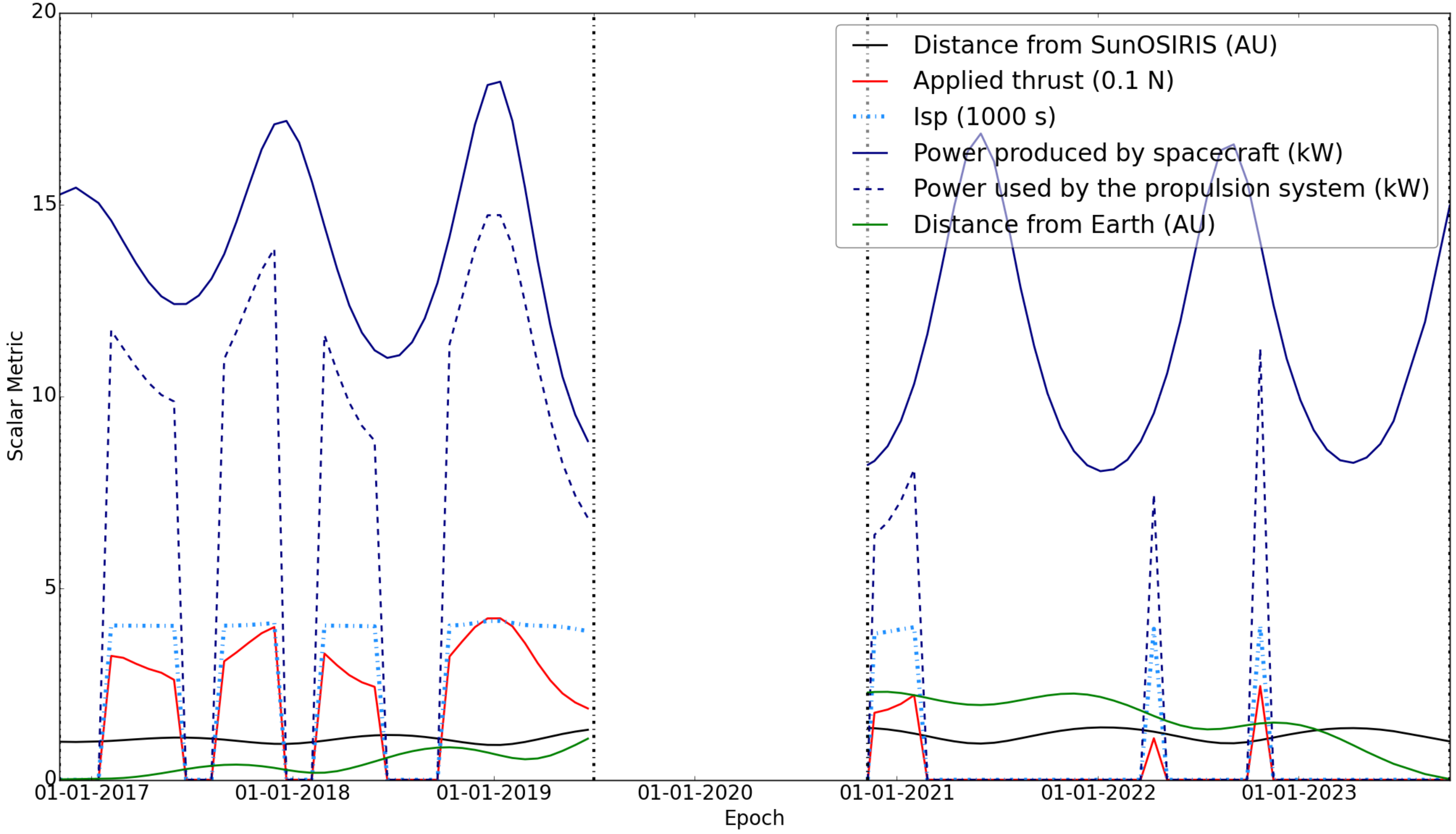


- In addition to trajectory plots, PyEMTG can create plots of systems parameters, such as power, propulsion, and distance from the Sun and Earth.
- Plotting distance from the Earth requires installing the “jplephem” and “de423” Python packages.
- PyEMTG can also map the continuous low-thrust model to a discrete throttle table if you have one. Unfortunately we can’t distribute those.

LowSIRIS-REx Trajectory Plot



LowSIRIS-REx Systems Plot

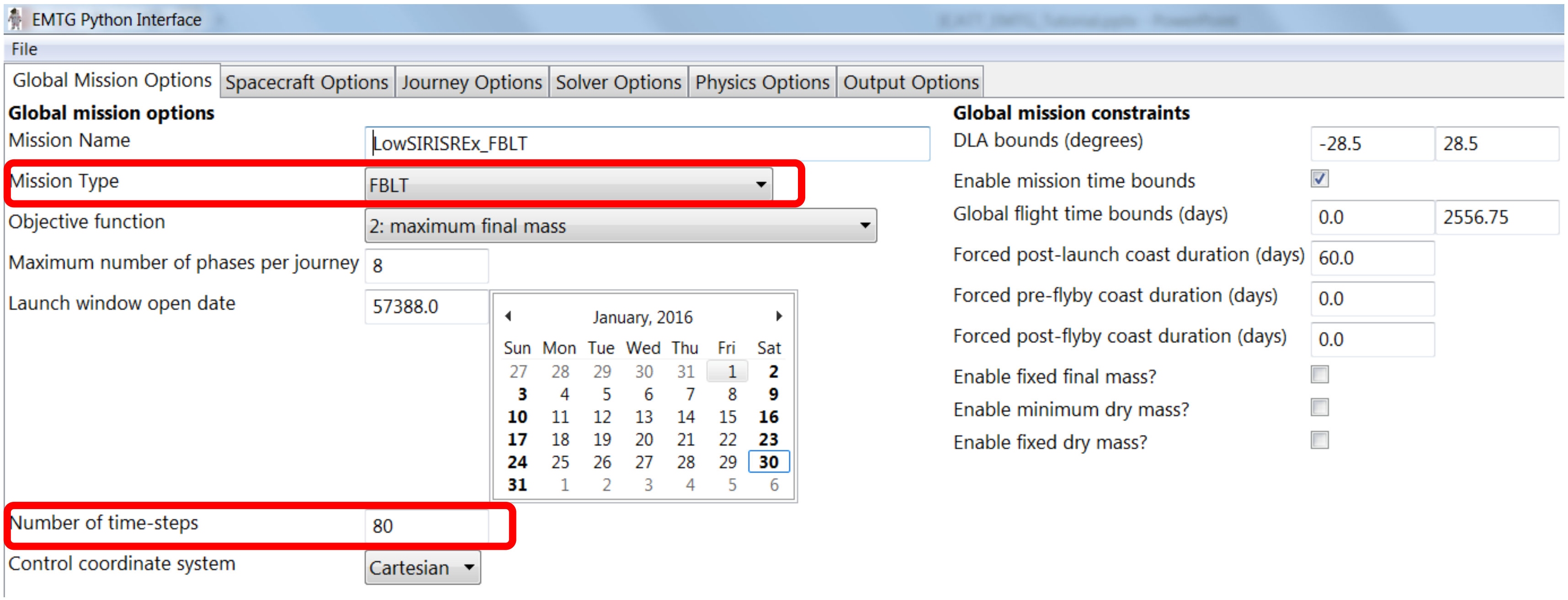


LowSIRIS-REx in Medium-High Fidelity

- The MGALT transcription is not adequate for detailed design work because it approximates the true low-thrust trajectory with a sequence of conic arcs and bounded impulses
- For a more accurate representation of the trajectory, we migrate to the “Finite-Burn Low-Thrust” (FBLT) transcription which uses a numerical integrator and the true low-thrust equations of motion
- FBLT can support perturbing terms like third body gravity and solar radiation pressure (SRP)
- However the body encounters are still patched-conic
- FBLT is a good intermediate step between the speed of MGALT and the accuracy of GMAT
- Integrated trajectory, accurate force model but flybys are still patched-conic

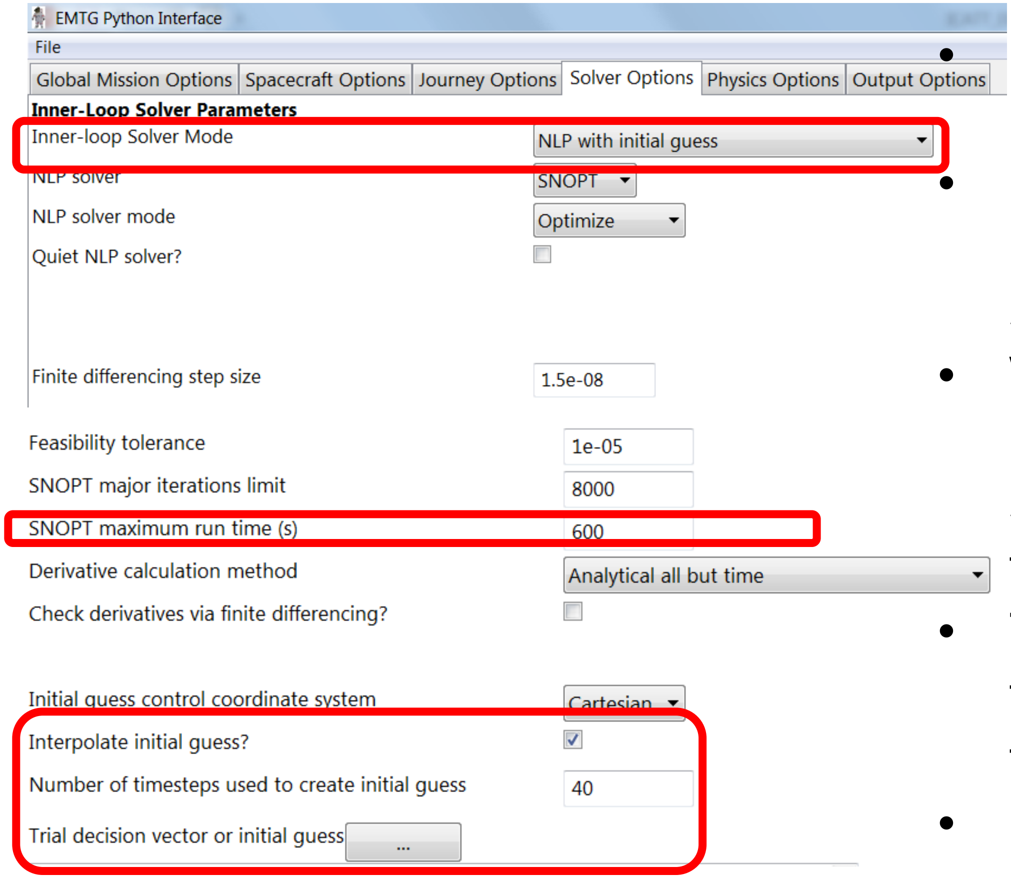
J. Englander, D. Ellison, and B.Conway, “Global Optimization of Low-Thrust, Multiple-Flyby Trajectories at Medium and Medium-High Fidelity,” AAS Space Flight Mechanics Meeting, Santa Fe, NM, January 2014.

LowSIRIS-REx in Medium-High Fidelity Step 1: The Global Options Panel



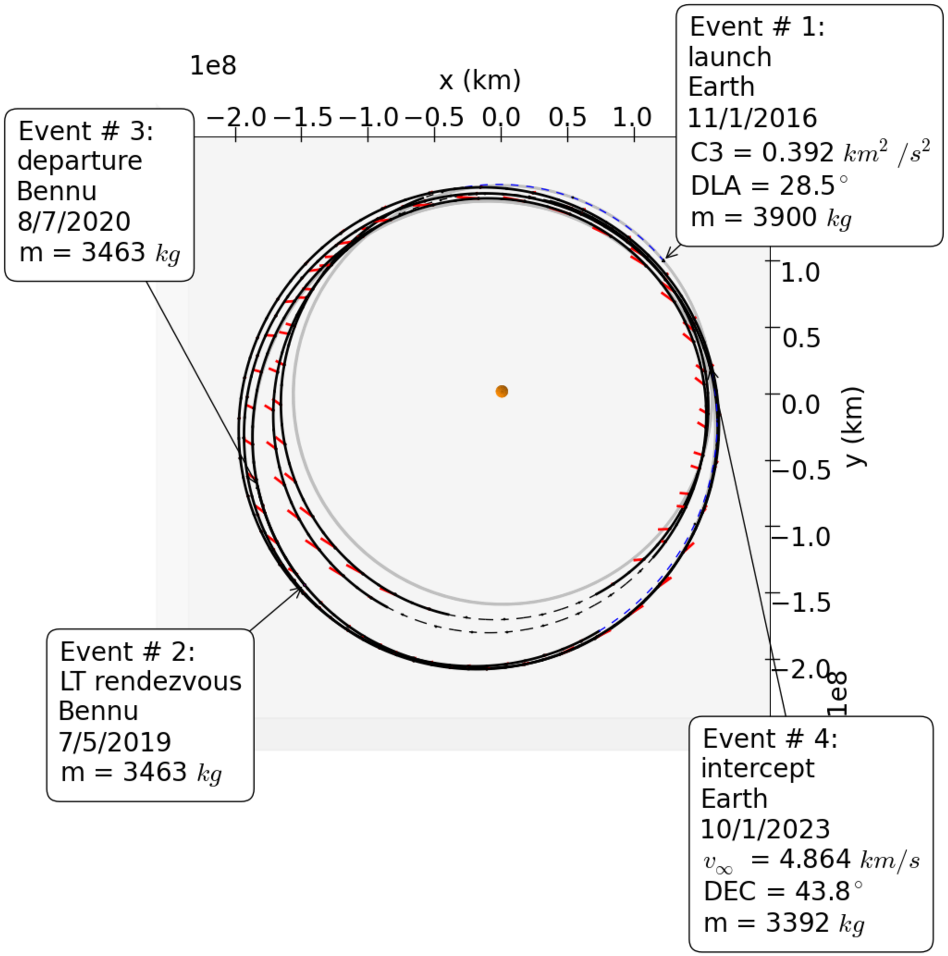
- Change the transcription to FBLT
- We want our detailed trajectory in higher resolution, *i.e.* more opportunities to change the control, so increase the number of time-steps to 80.

LowSIRIS-REx in Medium-High Fidelity
Step 2: The Solver Options Panel

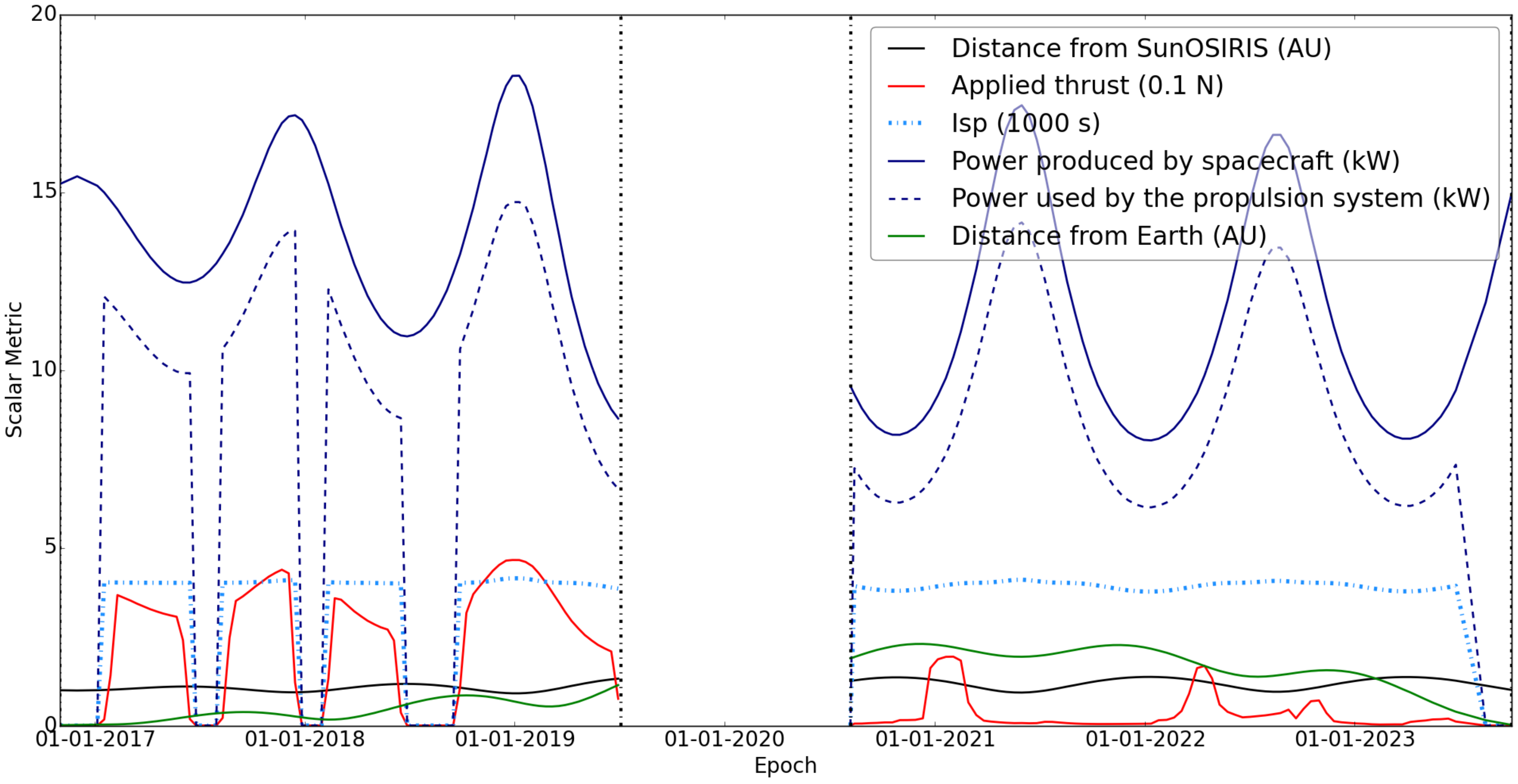


- Since we already have a solution in MGALT, we do not need to run MBH.
- Instead we will run our NLP solver (SNOPT) directly with the previous solution as an initial guess.
- We can tell EMTG to seed from a previous solution by clicking "…" and selecting the previous .emtg results file.
- The previous solution had only 40 time steps so we have to tell EMTG to interpolate it.
- FBLT is very slow. Let's give SNOPT 10 minutes to chew on the problem.

LowSIRIS-REx in Medium-High Fidelity Trajectory Plot



LowSIRIS-REx in Medium-High Fidelity Systems Plot



Some of the many things that we did not cover…

- EMTG's outer-loop solver
  - Can choose planetary flyby sequence
  - For small-bodies missions, can choose both the number and the identity of the targets
  - Can design the propulsion and power system for the spacecraft
  - Finds the Pareto-optimal non-dominated front between mission and systems objective functions of the user's choice
- Operational constraints
  - Distance from the sun and other bodies
  - Visibility angles at arrival and departure
- Powered gravity assist
- Low-thrust departure and arrival spirals
- Non-body boundary conditions
- Central bodies other than the sun (EMTG is central body agnostic)
- Automated export to GMAT

Conclusion

- EMTG is a flexible, automated tool for preliminary design of interplanetary trajectories that can find optimal solutions without requiring an initial guess.
- EMTG can design both missions with chemical propulsion and/or low-thrust electric propulsion.
- When operated in hybrid optimal control mode, EMTG can autonomously explore the design space for a mission and can be left to run independently for days while analysts do other work.
- Mission design mathematics may easily be automated. Communication and understanding cannot be. EMTG's automation allows analysts to focus their attention on understanding the needs of the customers (scientists) and the capabilities of the spacecraft while leaving the repetitive work to the computer.

Thank You

EMTG is available open-source at

https://sourceforge.net/projects/emtg/