EMTG Tutorial: Constraint Scripting

Tim Sullivan [*]

August 4, 2023

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

[*]Aerospace Engineer, The Aerospace Corporation

List of Known Issues

List of Acronyms

EMTG Evolutionary Mission Trajectory Generator

JD Julian Date

MJD Modified Julian Date

SPICE Spacecraft Planet Instrument Camera-matrix Events

MBH Monotonic Basin Hopping

GMAT General Mission Analysis Toolkit

DSM deep-space maneuver

1 Introduction

Evolutionary Mission Trajectory Generator (EMTG) offers the capability to handle a wide range of constraints that go beyond the limitations of PyEMTG. These constraints can be implemented using scripted constraints in the `.emtgopt` file. There are three classes of constraints available: boundary constraints, maneuver constraints, and phase distance constraints (also known as path constraints). Moreover, users have the option to create new boundary constraints by developing custom EMTG C++ code. An advanced tutorial will cover this process in detail. For more information on constraint scripting, please refer to the "EMTGv9 Constraint Scripting" reference provided in the EMTG documentation located at `docs/0_Users/constraint_scripting`.

2 Constraint Blocks

The EMTG's `.emtgopt` input file consists of a distinct constraint scripting block for each Journey within the mission. These blocks are present at the conclusion of each Journey and encompass all three classes of constraints, as illustrated below:

```
#Maneuver constraint code
#Works for absolute and relative epochs and also magnitudes
BEGIN_MANEUVER_CONSTRAINT_BLOCK
END_MANEUVER_CONSTRAINT_BLOCK
```

```
#Boundary constraint code
BEGIN_BOUNDARY_CONSTRAINT_BLOCK
END_BOUNDARY_CONSTRAINT_BLOCK
```

```
#Phase distance constraint code
BEGIN_PHASE_DISTANCE_CONSTRAINT_BLOCK
END_PHASE_DISTANCE_CONSTRAINT_BLOCK
```

When examining an .emtgopt file, you might come across mentions of "Phases." EMTG divides Journeys and Journey Decision Vectors into individual Phases. In the `#trial decision vector` section of an `.emtgopt` file, you can identify the Journey phases by the prefix `p#` where the `#` represents the Phase number starting from 0. For instance:

```
p0MGAnDSMs: phase flight time,200.28540267242266281755
p1MGAnDSMsForwardSubPhase0: burn index,0.77365173343362281244
```

Constraints are always tagged with a prefix that corresponds to the Phase they constrain. For example, constraints on the first phase of the Journey are prefixed with `p0`, while constraints on the fifth phase are prefixed with `p4`. Furthermore, users can apply constraints specifically to the *last* phase of the journey using the prefix `pEnd`.

2.1 Boundary Constraints

Boundary constraints can be applied to any boundary condition class. To apply a constraint to the left boundary of a specific phase/journey, tag the constraint as a "departure" constraint (e.g. p2_departure_orbitperiod_1yr_1.1yr). Similarly, for a right boundary, it should be tagged as an "arrival" constraint.

Constraints can be bounded by specifying the left and right bounds, along with their units, separated by an underscore. For constraints that require a specific reference frame, the frame is indicated after the upper bound, also separated by an underscore. The general template for a boundary constraint is as follows:

```
BEGIN_BOUNDARY_CONSTRAINT_BLOCK
p<phase number>_<boundary>_<bounded parameter name>_
<lower bound><lower bound units>_<upper bound><upper bound units>_<frame>
END_BOUNDARY_CONSTRAINT_BLOCK
```

where

- `<phase number>` is the integer number of the phase on which the constraint is applied.
- `<boundary>` is `departure` or `arrival`.
- `<lower bound>` The numerical lower bound for the constraint.
- `<lower bound units>` The units of the lower bound when applicable.
- `<upper bound>` The numerical upper bound for the constraint.
- `<upper bound units>` The units of the upper bound when applicable.
- `<frame>` The reference frame (when applicable) with respect to which the bounded parameter is calculated, defined by `body ID`. Valid choices are `ICRF`, `J2000BCI`, `J2000BCF`, `TrueOfDateBCI`, and `TrueOfDateBCF`.

2.2 Maneuver Constraints

In certain situations, there might be a need to override or add additional constraints to the maneuvers calculated by EMTG's optimization routine. This requirement can arise in various scenarios, including but not limited to:

1. Navigation restrictions: no maneuver pre/post another mission critical event
2. Engine type: restrict a certain maneuver to be executed with a particular propulsion system (bipropellant or monopropellant thruster sets)
3. Magnitude: constrain the size of a particular maneuver due to hardware limitations or practical navigation concerns

Maneuver constraints, similar to boundary constraints, are defined in the `.emtgopt` file within the `MANEUVER_CONSTRAINT_BLOCK`. Each maneuver constraint begins with syntax that identifies the corresponding Journey and Phase, followed by the type of constraint and its bounds. For instance, consider the following example: a constraint that enforces burn 0 in Phase 0 to occur between epoch 51544.0 and 51544.5.

```
BEGIN_MANEUVER_CONSTRAINT_BLOCK
p0b0_epoch_abs_51544.0_51544.5
END_MANEUVER_CONSTRAINT_BLOCK
```

For more detailed information on the various types of maneuver constraints and their syntax, please refer to the EMTG Constraint Scripting Guide included in the EMTG documentation.

3 Applying Constraints

The next two sections will walk through adding a boundary and maneuver constraint from the previous tutorial on Boundaries.

3.1 Adding a Boundary Constraint

For this tutorial, let's apply a boundary and a maneuver constraints to our EVM mission. From the Journey Boundaries tutorial make a copy of your `EVM_freepoint.emtgopt` file and name it `EVM_freepoint_boundary_constraint.emtgopt`. You can choose to either keep this EMTG options file in the original Journey Boundaries tutorial folder or make a new one.

Change the mission name to "EVM_freepoint_boundary_constraint" in the Global Options tab in PyEMTG or in the text file. We will use the same Universe and Hardware so those paths may be left as is. If you copied the EMTG options file to a new directory, you may want to change the results file path in the Options tab. We will refer to EMTG's solution with and without additional scripted constraints so run the mission to produce a feasible solution for later comparison or use the solution from the previous Journey Boundaries tutorial. The objective function value for the provided solution is 8.417.

One constraint that can only be added though constraint scripting is Detic Latitude. Add a geodetic latitude constraint to the Mars Free-point arrival phase by modifiying the `#Boundary constraint code` block to read:

```
BEGIN_BOUNDARY_CONSTRAINT_BLOCK
p0_arrival_DeticLatitude_25.0_26.0_J2000BCF
END_BOUNDARY_CONSTRAINT_BLOCK
```

This will cause EMTG to constrain the Mars free point arrival state to lie between 25-26 degrees (the units do not need to be specified with DeticLatitude) in the J2000 body fixed frame.

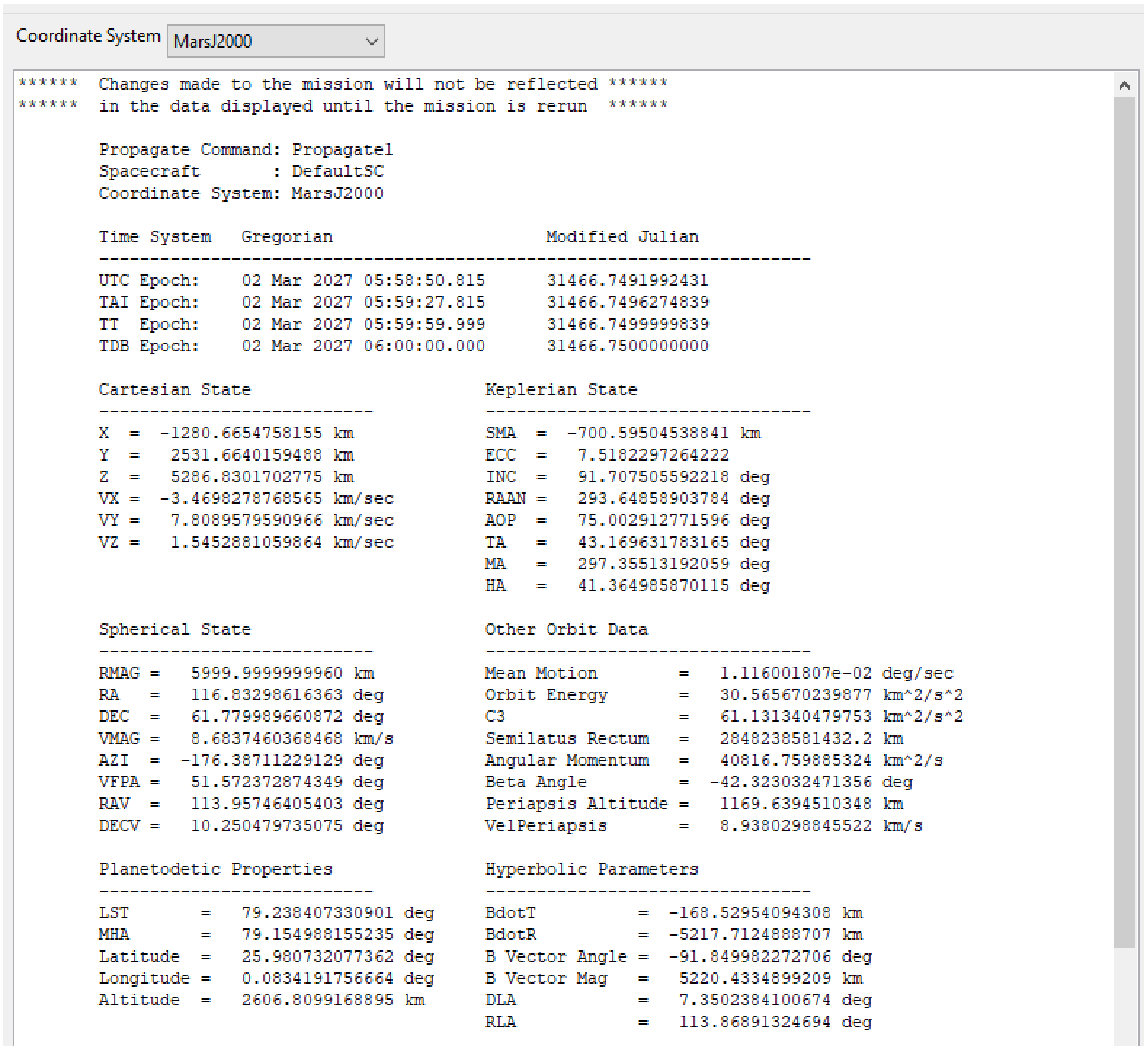
Optionally, you can check that the detic latitude constraint solution is satisfied in another tool like General Mission Analysis Toolkit (GMAT). Make the following Output Options changes to do this: check the "generate forward integrated ephemeris" option in the Output Options tab and use Mars-centered J2000BCF as the output frame by selecting "J2000_BCF" in the "Output file frame" setting and "4" as the SPICE ID of the central body.

Run EMTG using the previous solution without the latitude constraint as an initial guess by checking "Seed MBH?" on the Solver Options and selecting the `.emtg` file created when you ran EMTG earlier. When EMTG finishes running, if it found a solution, you should see a new constraint block in the `.emtg` file like the example below showing your detic latitude constraint is met. You will likely also observe that the objective function value is higher than the solution without this constraint (9.571 vs 8.417 when this tutorial was created).

```
BEGIN_BOUNDARY_CONSTRAINT_BLOCK
j1p0CoastPhaseFreePointChemRendezvous detic latitude (degrees, Mars J2000_BCF): 26.00000008
END_BOUNDARY_CONSTRAINT_BLOCK
```

If we create a Mars body fixed frame in GMAT and enter the final state from the epehemeris file as the spacecraft state, we can check that GMAT also reports a Mars detic latitude of about 26 degrees. For this to match, you must ensure that the flattening coefficient for Mars in the EMTG universe file (see the EMTG User Guide chapter on Universe Files) matches the one in the GMAT. You can find the GMAT flattening coefficient for Mars under Solar System/Mars in the Resources panel on the left side of GMAT.

NOTE: The final state in the integrated ephemeris may not have the capture burn at Mars included so the Mars state is still hyperbolic. More information on this issue can be found in the EMTG User Guide in the Ephemeris Output Options section.



```
******  Changes made to the mission will not be reflected ******
******  in the data displayed until the mission is rerun  ******

        Propagate Command: Propagate1
        Spacecraft       : DefaultSC
        Coordinate System: MarsJ2000

        Time System   Gregorian                    Modified Julian
        ----------------------------------------------------------------------
        UTC Epoch:    02 Mar 2027 05:58:50.815      31466.7491992431
        TAI Epoch:    02 Mar 2027 05:59:27.815      31466.7496274839
        TT  Epoch:    02 Mar 2027 05:59:59.999      31466.7499999839
        TDB Epoch:    02 Mar 2027 06:00:00.000      31466.7500000000

        Cartesian State                       Keplerian State
        ---------------------------           --------------------------------
        X  =  -1280.6654758155 km             SMA  =  -700.59504538841 km
        Y  =   2531.6640159488 km             ECC  =   7.5182297264222
        Z  =   5286.8301702775 km             INC  =   91.707505592218 deg
        VX =  -3.4698278768565 km/sec         RAAN =   293.64858903784 deg
        VY =   7.8089579590966 km/sec         AOP  =   75.002912771596 deg
        VZ =   1.5452881059864 km/sec         TA   =   43.169631783165 deg
                                              MA   =   297.35513192059 deg
                                              HA   =   41.364985870115 deg

        Spherical State                       Other Orbit Data
        ---------------------------           --------------------------------
        RMAG =   5999.9999999960 km           Mean Motion        =   1.116001807e-02 deg/sec
        RA   =   116.83298616363 deg          Orbit Energy       =   30.565670239877 km^2/s^2
        DEC  =   61.779989660872 deg          C3                 =   61.131340479753 km^2/s^2
        VMAG =   8.6837460368468 km/s         Semilatus Rectum   =   2848238581432.2 km
        AZI  =  -176.38711229129 deg          Angular Momentum   =   40816.759885324 km^2/s
        VFPA =   51.572372874349 deg          Beta Angle         =  -42.323032471356 deg
        RAV  =   113.95746405403 deg          Periapsis Altitude =   1169.6394510348 km
        DECV =   10.250479735075 deg          VelPeriapsis       =   8.9380298845522 km/s

        Planetodetic Properties               Hyperbolic Parameters
        ---------------------------           --------------------------------
        LST       =   79.238407330901 deg     BdotT          =  -168.52954094308 km
        MHA       =   79.154988155235 deg     BdotR          =  -5217.7124888707 km
        Latitude  =   25.980732077362 deg     B Vector Angle =  -91.849982272706 deg
        Longitude =   0.0834191756664 deg     B Vector Mag   =   5220.4334899209 km
        Altitude  =   2606.8099168895 km      DLA            =   7.3502384100674 deg
                                              RLA            =   113.86891324694 deg
```

Figure 1: Mars Arrival State in GMAT

3.2 Adding a Maneuver Constraint

Let's add a maneuver constraint in our `EVM_freepoint` mission. Create a new `.emtgopt` file using either your `EVM_freepoint.emtgopt` or `EVM_freepoint_boundary_constraint.emtgopt` file and name it `EVM_freepoint_maneuver_constraint.emtgopt`. Change the mission name to "EVM_freepoint_maneuver_constraint" in the Global Options tab in PyEMTG. Open the "Solver Options" tab in PyEMTG and change the "Inner-loop Solver Mode" to "Evaluate TrialX" and select a previous `.emtg` solution file from either the `EVM_freepoint` or `EVM_freepoint_boundary_constraint` results as the "Trial decision vector". Save the file in PyEMTG. Open both the `EVM_freepoint_maneuver_constraint.emtgopt` file and the `.emtg` solution file you just used as a "Trial decision vector" in text editors.

Examine Journey 0 in the `.emtg` file and find the date of the first `chem burn` entry. We're going to apply a maneuver constraint which forces this deep-space maneuver (DSM) to be on a different day. Convert the date of the DSM to an Modified Julian Date (MJD) epoch using one of the calendar tools in PyEMTG such as the one in the Global Options tab. For example 5 May 2026 would convert to 61165.0. In the Journey 0 section of your `EVM_freepoint_maneuver_constraint.emtgopt` file change the `BEGIN_MANEUVER_CONSTRAINT_BLOCK` by adding a constraint like the one below. This syntax indicates that the Phase 0 (`p0`) burn 0 (`b0`) absolute epoch `epoch_abs` should fall between MJD 61160.0-61162.0. Change the epochs as appropriate so that the constraint is forcing the maneuver to occur in a window that *does not* include the maneuver epoch in the `.emtg` file. Save the `.emtgopt` file.

```
BEGIN_MANEUVER_CONSTRAINT_BLOCK
p0b0_epoch_abs_61160.0_61162.0
END_MANEUVER_CONSTRAINT_BLOCK
```

Run the file using either the command line or PyEMTG (if you use PyEMTG, ensure that you *re-open* the `.emtgopt` file as updates in your text editor do not sync to PyEMTG if the file is still open in PyEMTG). You should see terminal output similar to what is shown below indicating that the maneuver constraint is violated. Additionally, in the run folder created inside your results directory your `.emtg` file should be named `FAILURE_EVM_freepoint_maneuver_constraint.emtg` indicating that the solution was not feasible. Note that the somewhat confusing `Decision vector is feasible` terminal output simply means that the trial decision vector is within it's upper and lower bounds *not* that it meets all the constraints.

```
Acquired infeasible point with feasibility 0.000460129
Worst constraint is F[1]: j0p0MGAnDSMsForwardSubPhase0: maneuver epoch absolute
constraint with violation 0.000460129
Decision vector is feasible.
EMTG run complete.
Press enter to close window.
```

Now try changing the solver mode to "Monotonic Basin Hopping" in the Solver Options tab of PyEMTG or by changing the text file `run_inner_loop` option to 1 and running EMTG. Open the resulting `.emtg` file and verify that the first DSM in Journey 0, Phase 0 occurs between the dates you set in the maneuver constraint block. For example, the chemical burn shown below on 5/2/2026 (Julian Date (JD) 2461162.50034601) converts to a MJD of 61162.00034601 (MJD = JD - 2,400,000.5) which meets the constraint within EMTG's tolerance.

```
10 | 2461157.44477123 |   4/26/2026 |        coast |
11 | 2461162.50034601 |    5/2/2026 |    chem_burn |
12 | 2461162.50034601 |    5/2/2026 |  match_point |
```