GMAT User Guide

GMAT User Guide

Table of Contents

I. Introduction 1
1. Introducing GMAT 3
..... 3
2. GMAT Interface Design/Philosophy 4
..... 4
3. System Requirements 5
..... 5
4. Installation 6
..... 6
5. Support and Resources 7
..... 7
II. How To 8
6. Reporting mission parameters 10
..... 10
7. Running scripts from MATLAB 11
..... 11
8. Creating ephemeris files 12
..... 12
9. Visualizing a trajectory 13
..... 13
III. Samples and Tutorials 14
10. Creating and Propagating a Spacecraft 18
Objective and Overview 18
Configuring Resources 18
Creating and Configuring a Spacecraft 18
Creating and Configuring a Propagator 21
Configuring the Default Orbit View Plot 23
Configuring the Mission Tree 26
Creating and Configuring a Propagate Command 26
Running the Mission 29
11. Designing a Hohmann Transfer 32
Objective and Overview 32
Creating and Configuring the Resource Tree 32
Creating the Differential Corrector 32
Modifying the default Orbit View 32
Creating and Configuring the Mission Tree 34
Initial Propagate Sequence 34
Running the Mission 38
12. Creating a Report 42
Objective and Overview 42
Prerequisites 42
Mission Description 42
Resource, Mission, and Output trees 43
Creating and Configuring the Resource Tree 43
Objects Required 43
Creating and Modifying Objects 44
Creating and Configuring the Mission Tree 47
Running the Mission 49
13. Running GMAT Scripts from MATLAB 53
Overview 53
Files and Folders to be used 53

Procedure 53
Additional Instructions 55
14. Force Models 56
Objective and Overview 56
Prerequisites 56
Mission Description 56
Creating and Configuring the Resource Tree 56
Objects Required 56
Creating and Modifying Objects 57
Creating and Configuring the Mission Sequence 63
The resulting view of Resource, Mission, and Output trees 67
Running the Mission 67
15. Algebraic Optimization 71
Objective and Overview 71
Prerequisites 71
Mission Description 71
Creating and Configuring the Resource Tree 71
Objects Required 71
Creating and Modifying Objects 71
Creating and Configuring the Mission Tree 73
Resource Mission and Output trees 73
Creating and Modifying Command Sequences 73
The resulting view of Resource, Mission, and Output trees 77
Running the Mission 77
16. Mars B-Plane Targeting 79
Objective and Overview 79
Prerequisites 79
Mission Description 79
Creating and Configuring the Resource Tree 79
Objects Required 80
Creating and Modifying Objects 80
Creating and Configuring the Mission Sequence 86
Creating and Modifying Sequences 87
The resulting view of Resource, Mission, and Output Trees 93
Running the Mission 93
17. L2 Transfer with Swingby 95
Objective and Overview 95
Prerequisites 95
Mission Description 95
Creating and Configuring the Resource Tree 95
Objects Required 96
Creating and Modifying Objects 96
Creating and Configuring the Mission Sequence 108
Resource Mission and Output trees 108
Creating and Modifying Command Sequences 109
The resulting view of Resource, Mission, and Output trees 122
Running the Mission 125
18. ACE Station Keeping 132
Objective and Overview 132
Prerequisites 132
Mission Description 133
Resource, Mission, and Output trees 134
Creating and Configuring the Resource Tree 135
Objects Required 135

Creating and Modifying Objects 135
Creating and Configuring the Mission Sequence 143
Running the Mission 149
19. Lunar Transfer 152
Objective and Overview 152
Prerequisites 152
Mission Description 152
Resource, Mission, and Output trees 153
Creating and Configuring the Resource Tree 153
Objects Required 154
Creating and Modifying Objects 154
Creating and Configuring the Mission Sequence 161
Creating and Modifying Sequences 161
Running the Mission 167
20. Formation Rendezvous 169
Objective and Overview 169
Prerequisites 169
Mission Description 169
Resource, Mission, and Output trees 170
Creating and Configuring the Resource Tree 171
Objects Required 171
Creating and Modifying Objects 171
Creating and Configuring the Mission Sequence 182
Creating and Modifying Sequences 183
Running the Mission 192
21. Minimum Fuel for Orbit Transfer 194
Objective and Overview 194
Prerequisites 194
Mission Description 194
Resource, Mission, and Output trees 196
Creating and Configuring the Resource Tree 197
Objects Required 197
Creating and Modifying Objects 197
Creating and Configuring the Mission Tree 207
Creating and Modifying Command Sequences 208
Running the Mission 218
22. Minimum Fuel for Lunar Transfer 223
Objective and Overview 223
Prerequisites 223
Mission Description 223
Resource, Mission, and Output trees 225
Creating and Configuring the Resource Tree 226
Objects Required 226
Creating and Modifying Objects 226
Creating and Configuring the Mission Tree 240
Creating and Modifying Command Sequences 241
Running the Mission 247
23. LEO Station Keeping 250
Objective and Overview 250
Prerequisites 250
Mission Description 250
Resource, Mission, and Output trees 251
Creating and Configuring the Resource Tree 252
Objects Required 252

Creating and Modifying Objects 252
Creating and Configuring the Mission Sequence 258
Running the Mission 263
IV. Reference Material 267
24. Resources 272
Spacecraft 272
Spacecraft Overview 272
Orbit Tab 273
Attitude Tab 283
Ballistic Mass Tab 289
Propagator 291
Overview 291
Interfaces 292
Reference Table 294
Object and Command Interactions 299
ReportFile 299
Overview 299
Interfaces 300
Reference Table 300
Objects and Commands Interactions 302
Fuel Tank 302
Overview 302
Using the GUI 303
Using the script 304
Reference Table 304
Thruster 305
Overview 305
Using the GUI 306
Using the script 307
Reference Table 308
SQP(fmincon) 313
Object Overview 313
Interface Overview 314
Field Descriptions 314
Using an fminconOptimizer 317
Solar System 317
Overview 317
Using the GUI 317
Reference Table 318
DifferentialCorrector 319
Overview 319
Interfaces 320
Reference Table 320
Object and Command Interactions 322
Celestial Bodies 322
Overview 322
Using the GUI 323
Using the script 323
Reference Table 324
OpenGL Plot 327
Overview 327
Using the GUI 327
Using the Script 331
Reference Tables 332

XYPlot 338
Overview 338
Using the GUI 338
Using the script 339
Reference Table 339
Coordinate Systems 341
Overview 341
Using the GUI 341
Using the script 343
Finite Burn 343
Overview 344
Reference Table 344
Examples 345
Objects and Commands Interactions 345
Variable 345
Overview 346
Interfaces 346
Object and Commands Interactions 346
String 347
Overview 347
Using the GUI 347
Using the script 348
Array 349
Overview 349
Using the GUI 349
Using the script 350
Reference Table 350
Barycenter 350
Overview 351
Using the GUI 351
Using the Script 352
Reference Table 352
Libration Point 353
Overview 353
Using the GUI 353
Using the Script 354
Reference Table 354
Impulsive Burn 355
Overview 355
Using the GUI 355
Using the Script 357
Reference Table 357
25. Commands 359
Propagate 359
Overview of the Propagate Command 359
Using the Propagate Command in the GUI 359
Using the Propagate Command in the script 361
Reference Table 362
While 364
Overview 364
Using GUI 364
Using the Script 365
Reference Table 366
If 366

Overview 366
Using the GUI 367
Using the script 368
Reference Table 368
Target 369
Overview 369
Using the GUI 369
Using the script 370
Reference Table 371
Report 372
Overview 372
Using the GUI 372
Using the script 373
Reference Table 373
For 374
Overview 374
Interfaces 374
Reference Table 375
Objects and Command Interactions 376
For Loop Behavior 376
Maneuver 376
Overview 376
Using the GUI 376
Using the script 377
Reference Table 377
Equation 378
Overview 378
Using the GUI 378
Using the script 379
Reference Table 379
Toggle 379
Overview 380
Using the GUI 380
Using the script 380
Reference Table 381
Vary 381
Overview 381
Using the GUI 381
Using the script 382
Reference Table 382
Begin Finite Burn 384
Overview 384
Using the GUI 384
Using the script 385
Reference Table 385
Achieve 385
Overview 386
Using the GUI 386
Using the script 387
Reference Table 387
Stop 388
Overview 388
Using the GUI 388
Using the script 389

ScriptEvent 389
Overview 389
Using the GUI 389
Using the script 390
Reference Table 391
Else 391
Overview 391
Using the Else Command in the GUI 391
Using the Else Command in the script 391
NonLinearConstraint 392
Overview 392
Using the GUI 392
Using the Script 393
Reference Table 394
Minimize 394
Overview 394
Using the GUI 395
Using the Script 395
Reference Table 395
Optimize 396
Overview 396
Using the GUI 396
Using the Script 397
Reference Table 397
Call Function 397
Overview 397
Using the Call Function Command in the GUI 398
Using the Call Function Command in the script 399
Reference Table 399
26. Miscellaneous 401
Math Operators in Scripting 401
Objective and Overview 401
Addition 401
Subtraction 401
Division 401
Multiplication 401
Exponentiation 401
Trigonometry 401
Square Root 402
norm 402
Logarithm 402
Matrix Manipulation 402
Radians and Degrees 402
Script Editor Shortcut Keys 402
Reference: Script Editor Shortcut Keys 402
Process Resources 404
Overview 404
General Process References 404
Configuration Management 404
Reviews 404
GMAT Commands in MATLAB 404
Overview 404
Commands 405
GMAT Console Instructions 406

Part I. Introduction

Table of Contents

1. Introducing GMAT 3
..... 3
2. GMAT Interface Design/Philosophy 4
..... 4
3. System Requirements 5
..... 5
4. Installation 6
..... 6
5. Support and Resources 7
..... 7

Chapter 1. Introducing GMAT

Chapter 2. GMAT Interface Design/ Philosophy

Chapter 3. System Requirements

Chapter 4. Installation

Chapter 5. Support and Resources

Part II. How To

Table of Contents

6. Reporting mission parameters 10
..... 10
7. Running scripts from MATLAB 11
..... 11
8. Creating ephemeris files 12
..... 12
9. Visualizing a trajectory 13
..... 13

Chapter 6. Reporting mission parameters

Chapter 7. Running scripts from MATLAB

Chapter 8. Creating ephemeris files

Chapter 9. Visualizing a trajectory

Part III. Samples and Tutorials

Table of Contents

10. Creating and Propagating a Spacecraft 18
    Objective and Overview 18
    Configuring Resources 18
        Creating and Configuring a Spacecraft 18
        Creating and Configuring a Propagator 21
        Configuring the Default Orbit View Plot 23
    Configuring the Mission Tree 26
        Creating and Configuring a Propagate Command 26
    Running the Mission 29
11. Designing a Hohmann Transfer 32
    Objective and Overview 32
    Creating and Configuring the Resource Tree 32
        Creating the Differential Corrector 32
        Modifying the default Orbit View 32
    Creating and Configuring the Mission Tree 34
        Initial Propagate Sequence 34
    Running the Mission 38
12. Creating a Report 42
    Objective and Overview 42
        Prerequisites 42
        Mission Description 42
        Resource, Mission, and Output trees 43
    Creating and Configuring the Resource Tree 43
        Objects Required 43
        Creating and Modifying Objects 44
        Creating and Configuring the Mission Tree 47
        Running the Mission 49
13. Running GMAT Scripts from MATLAB 53
    Overview 53
        Files and Folders to be used 53
    Procedure 53
        Additional Instructions 55
14. Force Models 56
    Objective and Overview 56
        Prerequisites 56
        Mission Description 56
    Creating and Configuring the Resource Tree 56
        Objects Required 56
        Creating and Modifying Objects 57
    Creating and Configuring the Mission Sequence 63
    The resulting view of Resource, Mission, and Output trees 67
    Running the Mission 67
15. Algebraic Optimization 71
    Objective and Overview 71
        Prerequisites 71
        Mission Description 71
    Creating and Configuring the Resource Tree 71
        Objects Required 71
        Creating and Modifying Objects 71
    Creating and Configuring the Mission Tree 73
        Resource Mission and Output trees 73

Creating and Modifying Command Sequences ... 73
The resulting view of Resource, Mission, and Output trees ... 77
Running the Mission ... 77
16. Mars B-Plane Targeting ... 79
Objective and Overview ... 79
Prerequisites ... 79
Mission Description ... 79
Creating and Configuring the Resource Tree ... 79
Objects Required ... 80
Creating and Modifying Objects ... 80
Creating and Configuring the Mission Sequence ... 86
Creating and Modifying Sequences ... 87
The resulting view of Resource, Mission, and Output Trees ... 93
Running the Mission ... 93
17. L2 Transfer with Swingby ... 95
Objective and Overview ... 95
Prerequisites ... 95
Mission Description ... 95
Creating and Configuring the Resource Tree ... 95
Objects Required ... 96
Creating and Modifying Objects ... 96
Creating and Configuring the Mission Sequence ... 108
Resource Mission and Output trees ... 108
Creating and Modifying Command Sequences ... 109
The resulting view of Resource, Mission, and Output trees ... 122
Running the Mission ... 125
18. ACE Station Keeping ... 132
Objective and Overview ... 132
Prerequisites ... 132
Mission Description ... 133
Resource, Mission, and Output trees ... 134
Creating and Configuring the Resource Tree ... 135
Objects Required ... 135
Creating and Modifying Objects ... 135
Creating and Configuring the Mission Sequence ... 143
Running the Mission ... 149
19. Lunar Transfer ... 152
Objective and Overview ... 152
Prerequisites ... 152
Mission Description ... 152
Resource, Mission, and Output trees ... 153
Creating and Configuring the Resource Tree ... 153
Objects Required ... 154
Creating and Modifying Objects ... 154
Creating and Configuring the Mission Sequence ... 161
Creating and Modifying Sequences ... 161
Running the Mission ... 167
20. Formation Rendezvous ... 169
Objective and Overview ... 169
Prerequisites ... 169
Mission Description ... 169
Resource, Mission, and Output trees ... 170
Creating and Configuring the Resource Tree ... 171
Objects Required ... 171

Creating and Modifying Objects 171
Creating and Configuring the Mission Sequence 182
Creating and Modifying Sequences 183
Running the Mission 192
21. Minimum Fuel for Orbit Transfer 194
Objective and Overview 194
Prerequisites 194
Mission Description 194
Resource, Mission, and Output trees 196
Creating and Configuring the Resource Tree 197
Objects Required 197
Creating and Modifying Objects 197
Creating and Configuring the Mission Tree 207
Creating and Modifying Command Sequences 208
Running the Mission 218
22. Minimum Fuel for Lunar Transfer 223
Objective and Overview 223
Prerequisites 223
Mission Description 223
Resource, Mission, and Output trees 225
Creating and Configuring the Resource Tree 226
Objects Required 226
Creating and Modifying Objects 226
Creating and Configuring the Mission Tree 240
Creating and Modifying Command Sequences 241
Running the Mission 247
23. LEO Station Keeping 250
Objective and Overview 250
Prerequisites 250
Mission Description 250
Resource, Mission, and Output trees 251
Creating and Configuring the Resource Tree 252
Objects Required 252
Creating and Modifying Objects 252
Creating and Configuring the Mission Sequence 258
Running the Mission 263

Chapter 10. Creating and Propagating a Spacecraft

This tutorial was written for GMAT 2011a.

Objective and Overview

The objective of this tutorial is to teach you how to create a spacecraft and a propagator, and then propagate the spacecraft to orbit perigee by following these basic steps:

1. Create a spacecraft and set its epoch and orbital elements.
2. Create and configure a propagator.
3. Modify the default Orbit View to visualize the trajectory.
4. Configure the mission sequence to propagate the spacecraft to periapsis.

Configuring Resources

Creating and Configuring a Spacecraft

In this section, you'll learn how to set a spacecraft's initial epoch and classical orbital elements. You'll need GMAT open with the default mission loaded. The default mission is loaded when a new session of the GMAT executable is started or when the New Mission button in the Toolbar is clicked.

Creating a Spacecraft

Working from the GUI, you can create a new spacecraft by starting at the Resource Tree.

1. Right click on the Spacecraft folder, and select Add Spacecraft
2. Rename the spacecraft by right-clicking on the new spacecraft and selecting Rename from the drop down menu. For this tutorial, name the Spacecraft "Sat".
3. See the resource tree figure for a view of the tree after creating a new spacecraft and naming it Sat.

Setting a Spacecraft's Epoch

1. Double left click on the Spacecraft icon for the new spacecraft Sat in the Resource Tree to open the Spacecraft's dialog box. If it is not already selected, click on the Orbit Tab.
2. Left-click the Epoch Format drop-down menu and select UTCGregorian. You'll see the value in the Epoch field change to the UTC Gregorian epoch format.
3. Left-click in the Epoch field, and type the desired value of 22 Jul 2014 11:29:10.811 (or you can cut and paste from the text in this tutorial).
4. Save the changes by left clicking the Apply button at the bottom of the window. Below you see the orbit tab after correctly setting the epoch to the desired value.

You can also set the spacecraft epoch in the script. In the script, you set this epoch using the following two lines:

```
Sat.DateFormat = UTCGregorian;
Sat.Epoch = 22 Jul 2014 11:29:10.811;
```

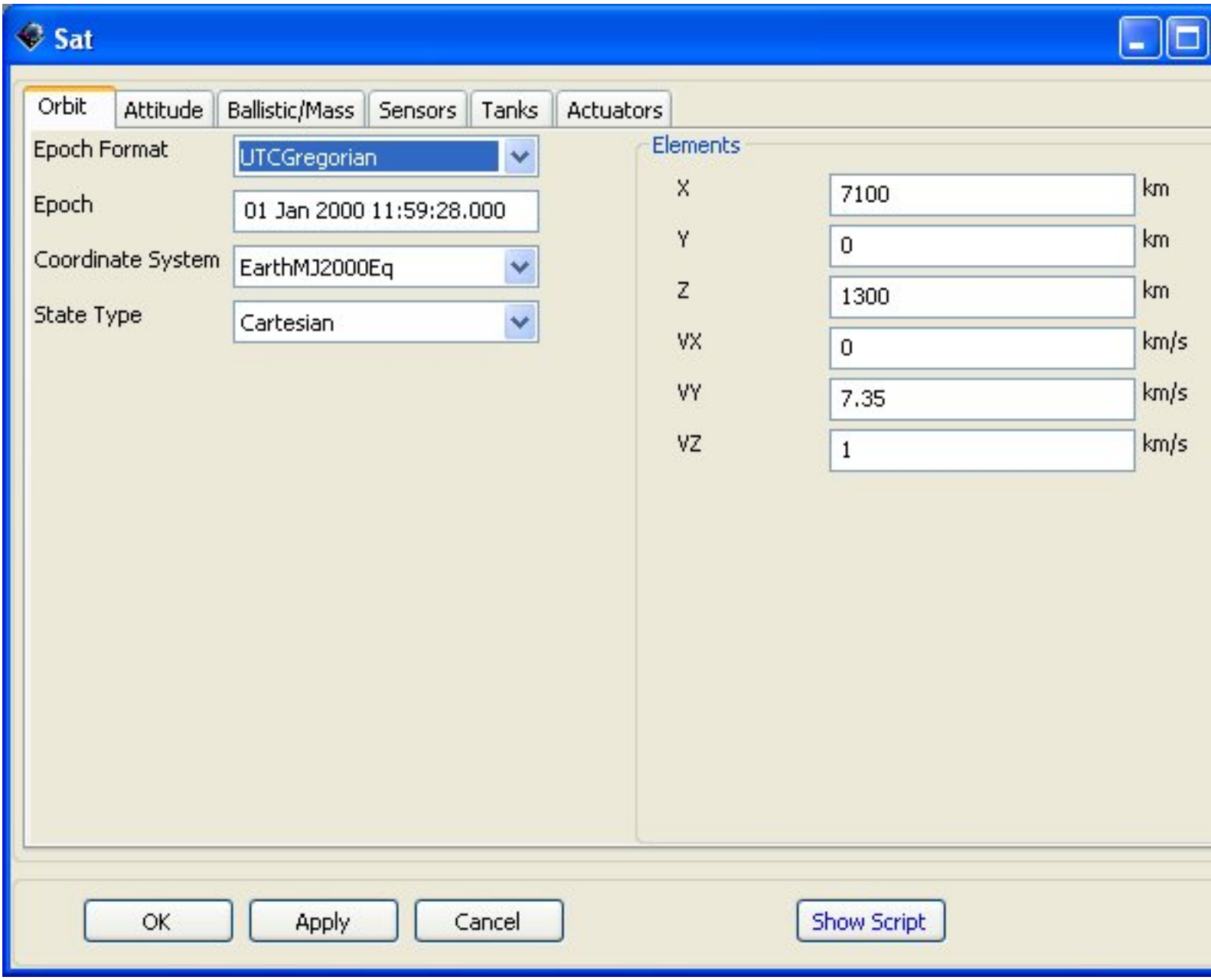


Figure: Spacecraft Epoch Setup

Setting a Spacecraft's State

We'll use the Keplerian orbital elements for this tutorial and we'll enter them with respect to Earth's MJ2000 Equator system.

1. Open the spacecraft dialog box for the spacecraft named Sat if it is not already open. (See section named Setting a Spacecraft's Epoch just above if you don't remember how to do this).
2. Change the state type to Keplerian by left-clicking on the State Type drop down menu and selecting "Keplerian". You will see the GUI reconfigure to the Keplerian representation.
3. Set the values in the GUI to those seen in the image below.
4. Compare the GUI to the screen capture below to ensure you have completed the steps correctly.
5. Save the changes and close the dialog box by pressing the OK button.
6. Save your work to the hard disk, press the save button located in the toolbar. If this is the first time you've saved the mission, you'll be prompted to provide a name and location for the file.

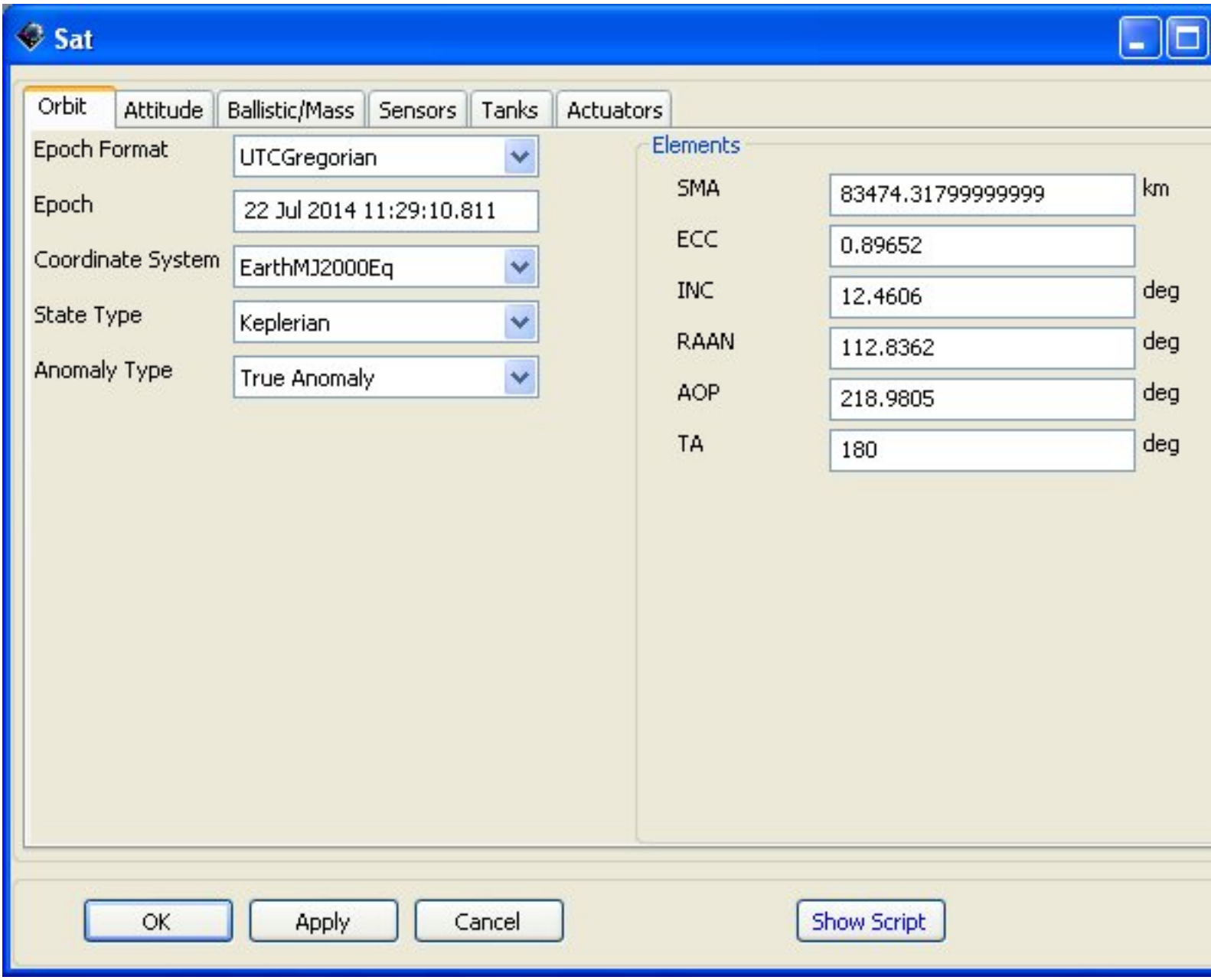


Figure: Spacecraft State Setup

Creating and Configuring a Propagator

In this section you'll learn how to create a new propagator from the GUI. This involves selecting a numerical integrator and a force model. Let's start by creating a new propagator from the GUI.

Creating a Propagator

1. Locate the Propagators folder in the Resource Tree.

2. Right-click on the Propagators folder and select Add Propagator from the drop-down menu.

3. Rename Propagator1 as "LowEarthProp". To do this, right click on the newly created Propagator1 and select Rename. In the dialog box that appears, type LowEarthProp and hit OK.

Look at the dialog box for LowEarthProp by double left clicking on its icon under the Propagators folder. On the left side of the propagator dialog box you see where you can select the desired numerical integrator and configure it for your application. On the right hand side of the panel are combo boxes and lists that allow the user to set up the force model. Now let's look at how to configure a force model.

Configuring a Force Model

For this tutorial we will use an Earth 10x10 non-spherical gravity model, Jacchia-Roberts atmospheric model, and point mass perturbations from the Sun and Moon.

1. Open LowEarthProp from the Propagators folder in the Resource tree

2. Locate the Primary Bodies group on the Propagator dialog box. In the Gravity group box, change the degree and order to 10 by left clicking in the text field and typing in the values.

3. Locate the Atmosphere Model pull-down menu in the Drag group.

4. Left click on the pull-down menu and select JacchiaRoberts. (For now we will leave the default options for Jacchia-Roberts model)

5. Left click on the Select button to the right of the Point Masses text box input field, which brings up a list box containing all gravitational bodies that have not been included in the Primary Bodies list (See Figure below).

6. Left click on "Sun" under the available bodies list, and then left click the right arrow found between the available bodies and Selected Bodies list boxes.

7. Add the Moon (named Luna in GMAT) following the same procedure as the preceding step.

8. Left click on the OK button to save changes.

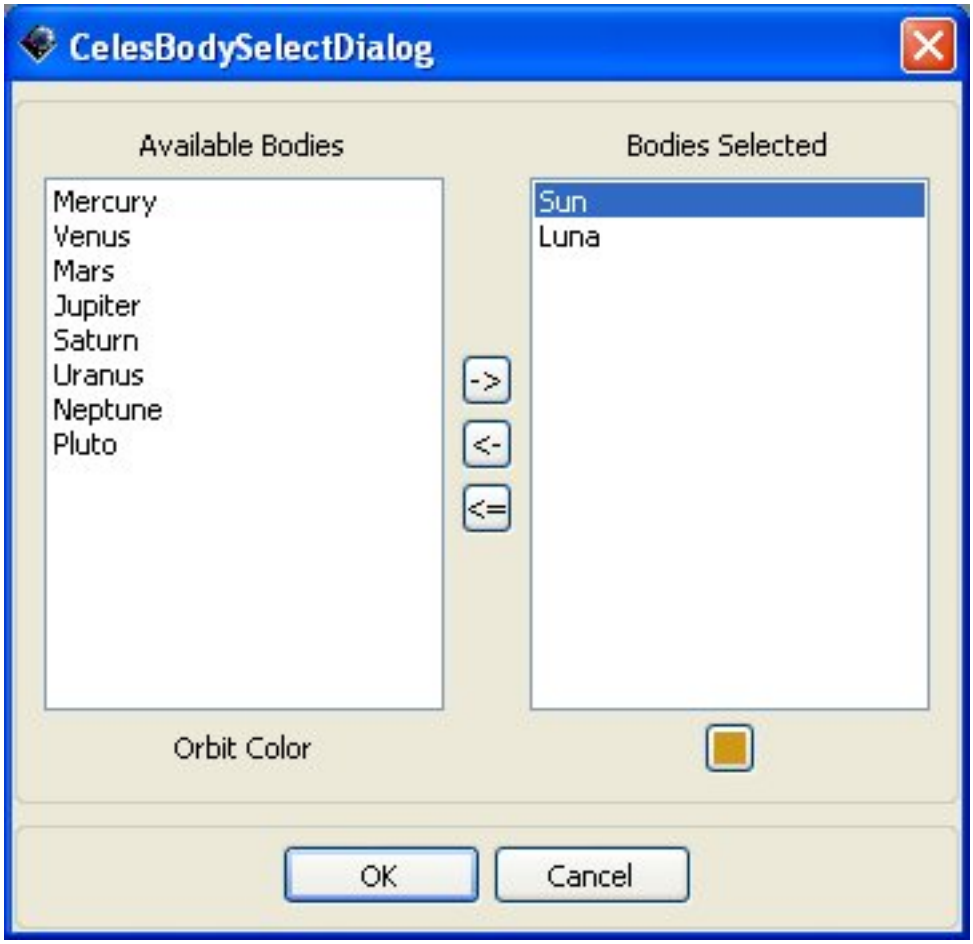


Figure: Force Model Point Mass Configuration

Below is an illustration after correctly configuring the force model according to the instructions above.

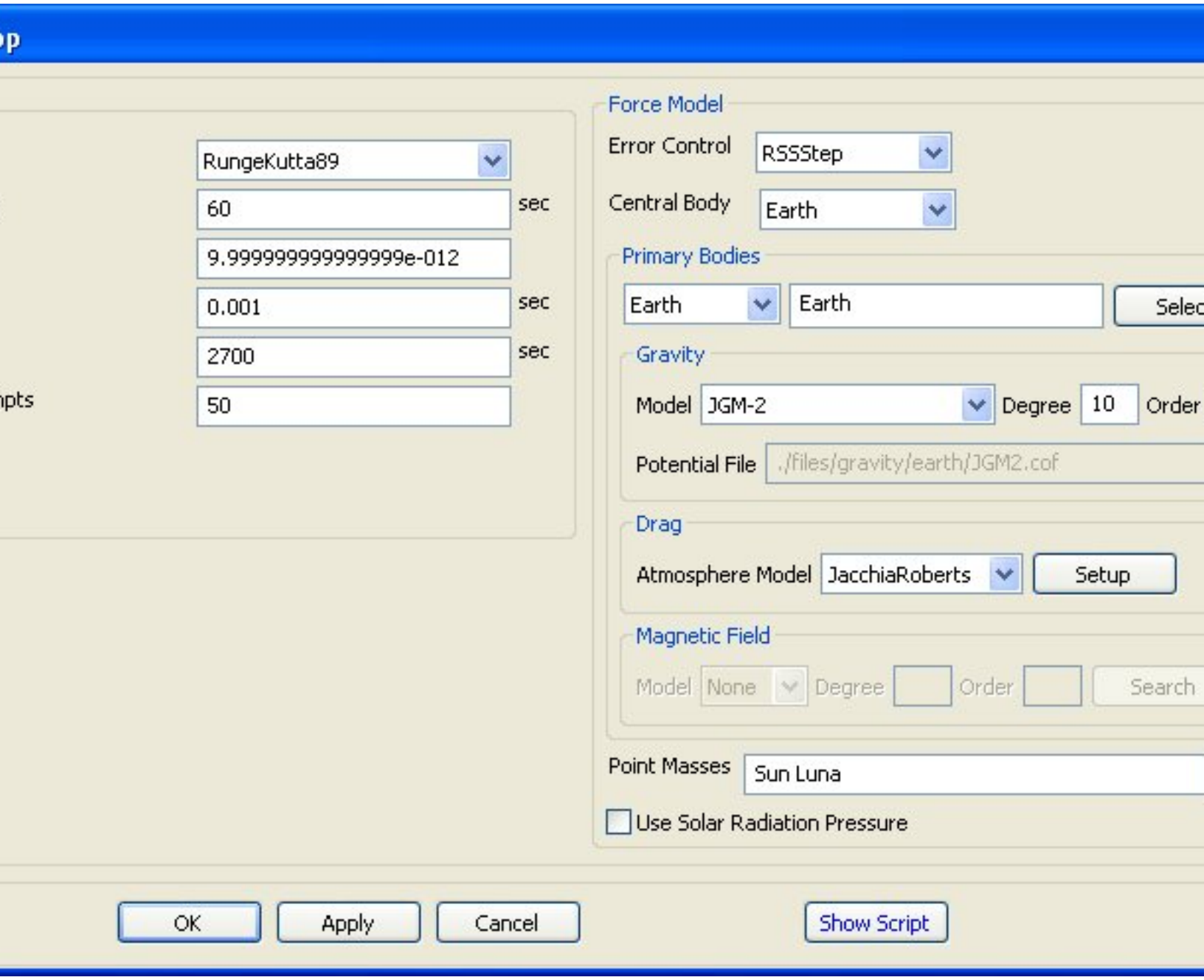


Figure: Force Model Configuration

Configuring the Default Orbit View Plot

In this section, we'll configure the default Orbit View plot to show the spacecraft we've created above. We'll remove DefaultSC from the list of objects to appear in the plot, add Sat, and change the view point so we can see the entire orbit when we propagate the spacecraft.

1. Locate the Output folder under the Resource Tree and double-left-click on DefaultOrbitView to open its dialog box.

2. Locate the View Object group, and find the Selected Spacecraft list.

3. Left click on DefaultSC under Selected Spacecraft and then click the left arrow button that appears to the left of the Selected Spacecraft list. This removes DefaultSC from the plot.

4. Locate the Spacecraft list in the View Object group.

5. Left click on Sat, and then left click on the right-pointing arrow button that appears to the right of the Spacecraft list.

The orbit for Sat is a highly eccentric orbit, and to view the entire orbit, we need to change the settings in the View Definition group.

1. Locate the View Point Vector settings in the View Definition group.

2. In the text boxes to the right of the ViewPointVector pull-down menu, enter 30000, -5000, and 5000 respectively as shown in illustration below

3. Uncheck the DrawXY Plane box located in the Drawing Options group.

See the figure below for the correct configuration of DefaultOrbitView after completing the steps above. When you are finished click Ok to save the changes.

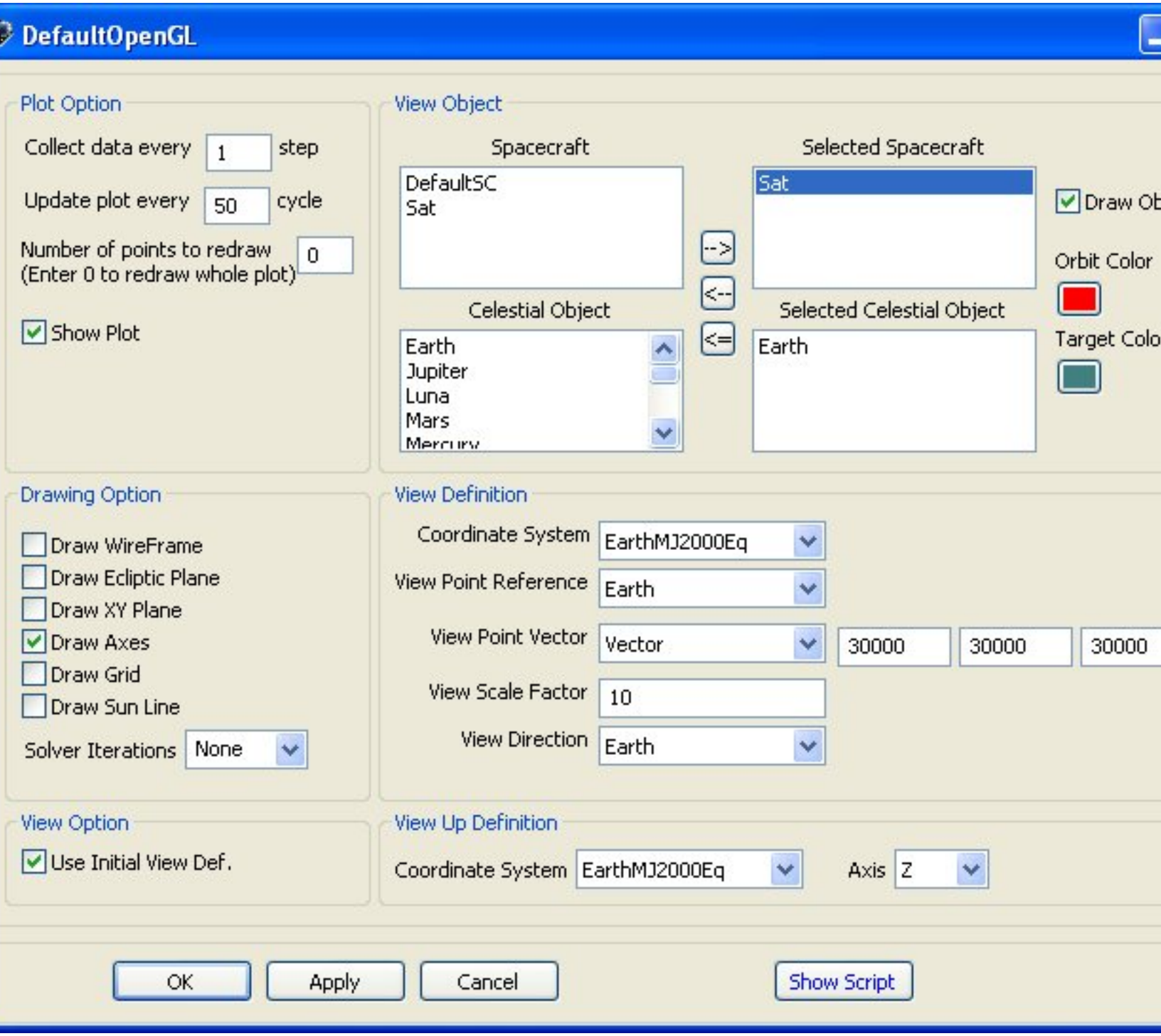


Figure: DefaultOrbitView Configuration

Configuring the Mission Tree

Creating and Configuring a Propagate Command

Creating a Propagate Command

The last step in this tutorial is to propagate the spacecraft. To do this, we'll configure a propagate command in the mission and configure it to propagate until the spacecraft reaches orbit periapis.

1. Locate the Mission tab and left click on it to bring up the mission sequence. The Mission tab is located next to the Resource tab located near the upper left hand corner of the GMAT GUI.

2. Double left click on the event named Propagate1. You will see a dialog box that allows you to configure the command.

Selecting a Propagator

1. Locate the propagator list in the Propagators and Spacecraft group box.

2. Right click on the field where you see DefaultProp.

3. Left click on LowEarthProp and hit Ok.

Figure: Propagate Command Propagator Select Dialog Configuration

Selecting a Spacecraft

To propagate the new spacecraft named Sat, we need to add it to the Spacecraft List group where originally you see DefaultSC. To propagate the Spacecraft named Sat:

1. Right-click in the field where you see DefaultSC.

2. Remove DefaultSC from the list by left clicking on it and then left-click on the left arrow button.

3. Left click on Sat and then left click on the right arrow button to add it to the list of spacecraft to propagate.

4. Left click on the OK button.

Below you see the dialog box after correctly configuring it to propagate the spacecraft named Sat.

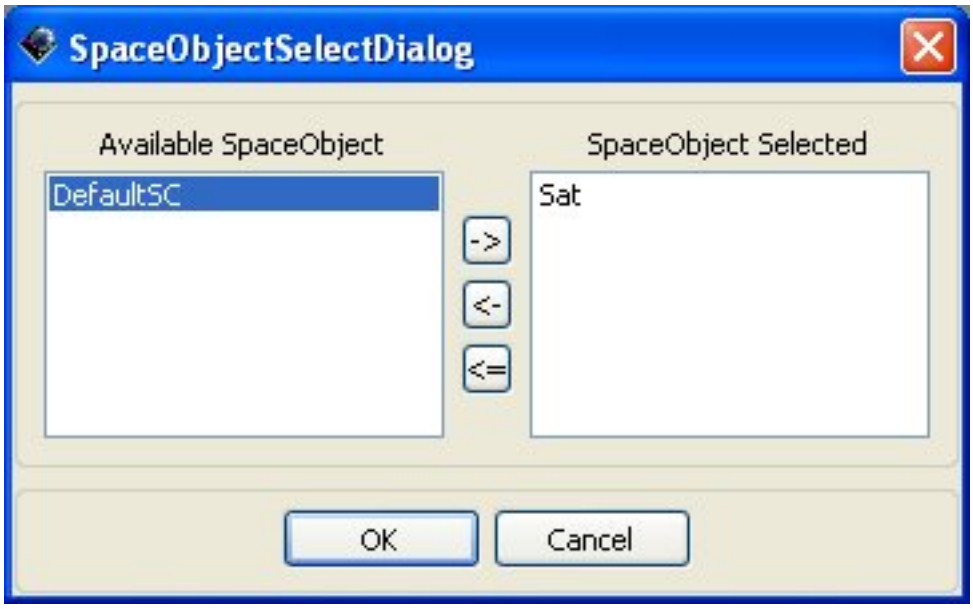


Figure: Propagate Command Space Object Select Dialog Configuration

Selecting a Stopping Condition

The only remaining thing to do in configuring the propagate event, is to tell the propagator when to stop the propagation.

1. Locate the Stopping Conditions group, and right click in the field that currently says DefaultSC.ElapsedSecs located in the Parameter column.

2. Find the Object List and choose Sat instead of Default Sat. (This tells GMAT the stopping condition will be associated with the spacecraft named Sat.)

3. Locate the Property list in the middle of the dialog box and scroll down to locate Periapsis.

4. Left click on the word Periapsis, and then left click on the right-pointing arrow found in the center column of the dialog box.

The dialog box below shows an example after correctly performing these instructions. Once you are finished, click Ok to save your changes.

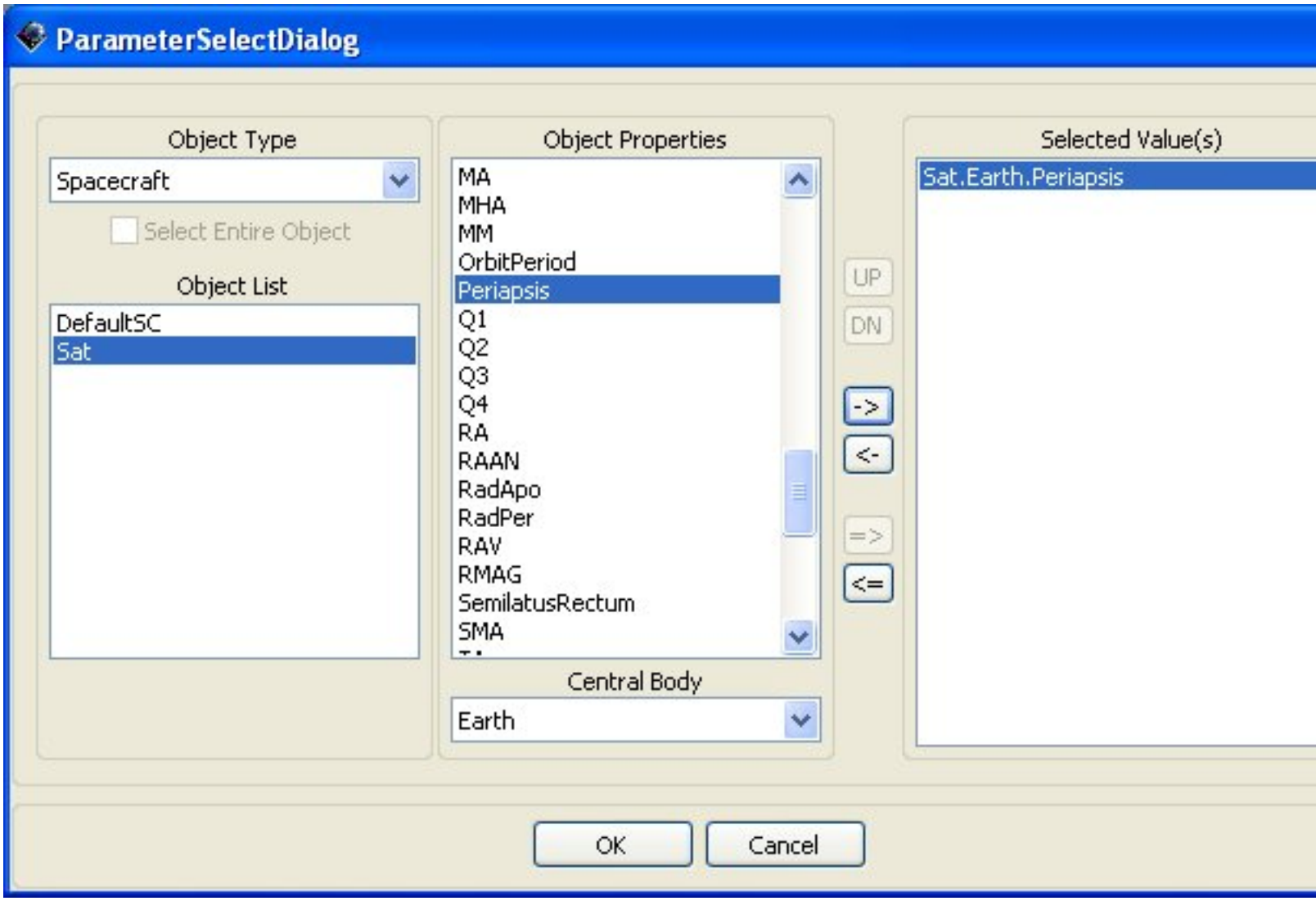


Figure: Propagate Command Parameter Select Dialog Configuration

Having set the propagator, spacecraft, and stopping conditions above, the Propagate1 dialog box should appear as the screen capture below. When you are finished click Ok to save the changes you've made to Propagate1.

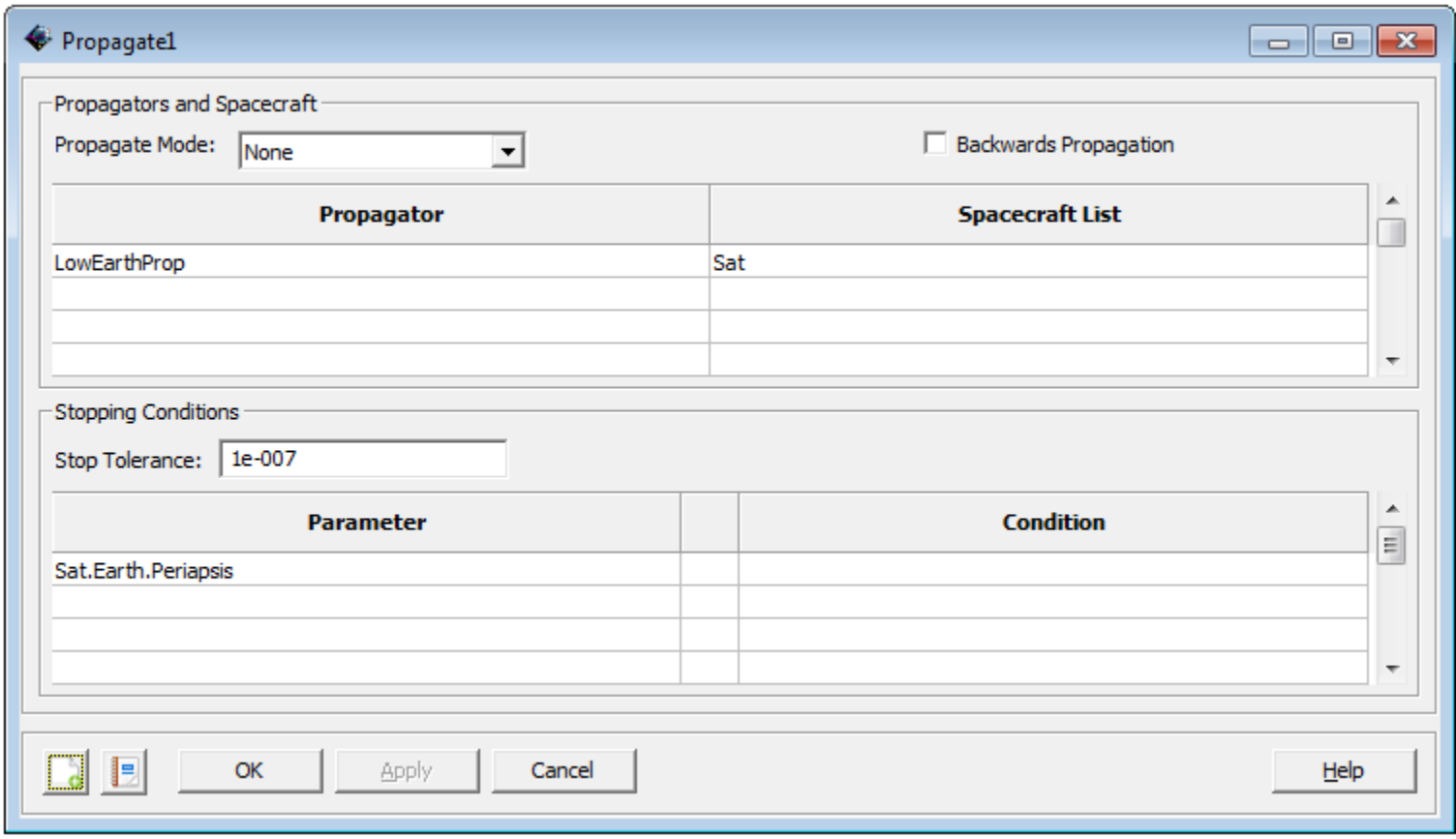


Figure: Propagate Command Configuration

Running the Mission

You have now configured the mission and it is time to run it and see the results.

1. Left-click on the Save button in the toolbar.

2. Left-click on the Run button in the toolbar.

You should see GMAT propagate the orbit and stop at orbit periapsis. The screen capture below summarizes what you should see after correctly completing this tutorial. You can manipulate the Orbit View plot using your mouse to zoom in and rotate the plot.

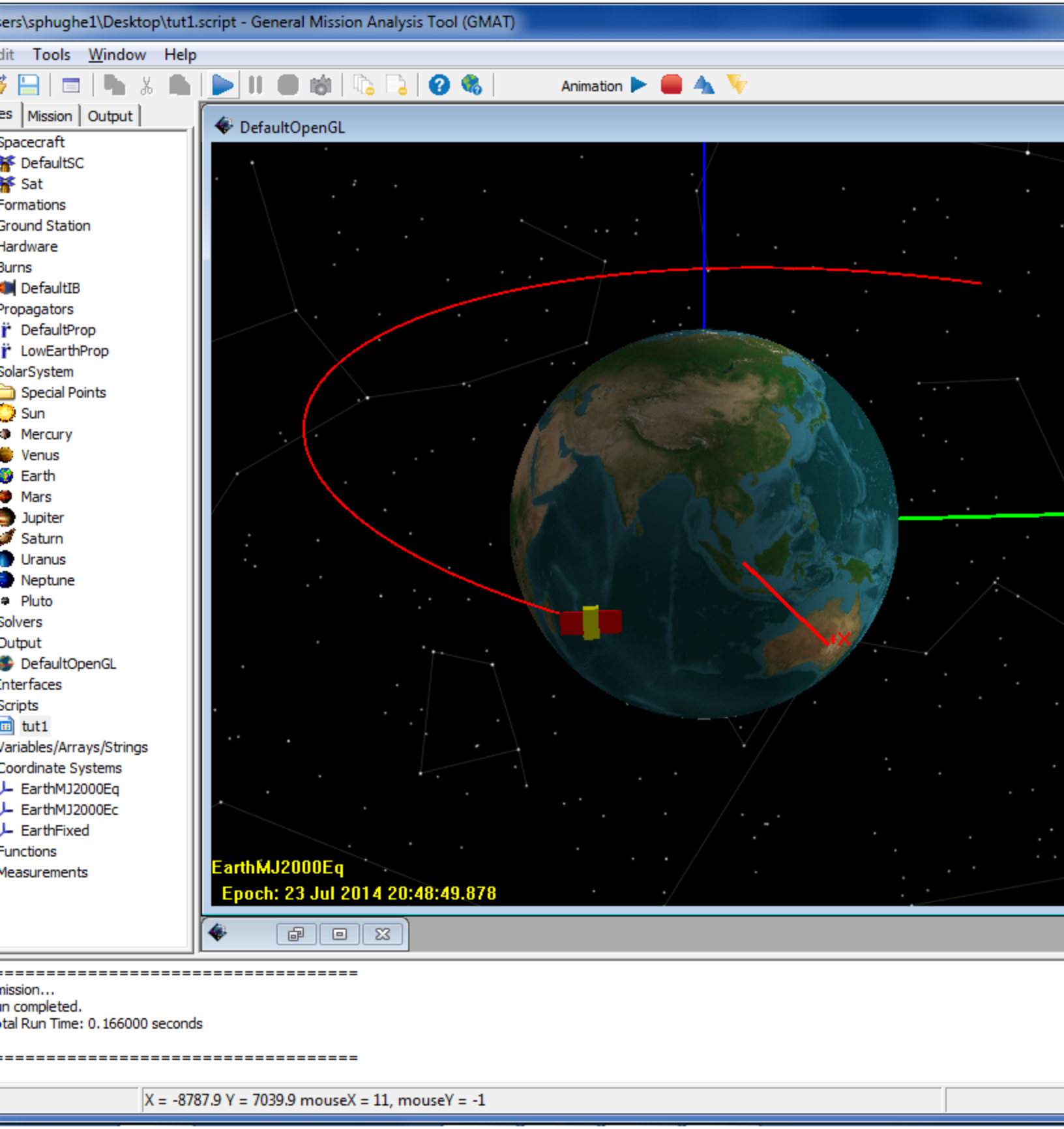

ers\sphughe1\Desktop\tut1.script - General Mission Analysis Tool (GMAT)
dit Tools Window Help
Animation
es Mission Output
Spacecraft
DefaultSC
Sat
Formations
Ground Station
Hardware
Burns
DefaultIB
Propagators
DefaultProp
LowEarthProp
SolarSystem
Special Points
Sun
Mercury
Venus
Earth
Mars
Jupiter
Saturn
Uranus
Neptune
Pluto
Solvers
Output
DefaultOpenGL
Interfaces
Scripts
tut1
Variables/Arrays/Strings
Coordinate Systems
EarthMJ2000Eq
EarthMJ2000Ec
EarthFixed
Functions
Measurements
DefaultOpenGL
+X
EarthMJ2000Eq
Epoch: 23 Jul 2014 20:48:49.878
==========
mission...
un completed.
otal Run Time: 0.166000 seconds
==========
X = -8787.9 Y = 7039.9 mouseX = 11, mouseY = -1

Figure: Orbit View Plot after Mission Run

Chapter 11. Designing a Hohmann Transfer

This tutorial created using the July 2, 2008 version of GMAT

Objective and Overview

The objective of this tutorial is to use differential correction to perform a Hohmann Transfer. To focus on the configuration of the target sequence, we make extensive use of the default configurations for spacecraft, propagators, and maneuvers. The basic steps are.

1. Create and configure a differential corrector.
2. Modify the default Orbit View to visualize the trajectory.
3. Add a target sequence to the mission to raise apogee to GEO altitude
4. Add a second target sequence to circularize the orbit.
5. Run the mission, save the solution, and rerun the mission using the converged solution.

You can download the script file and run it beforehand to see the final results of this tutorial: HohmannTransferDesign.script [scripts/HohmannTransferDesign.script]

Creating and Configuring the Resource Tree

Begin by loading the default mission (click the new mission button in the toolbar) or starting a new GMAT session. For this tutorial, we will use the default configurations for DefaultSC, DefaultProp, and the default impulsive burn. DefaultSC is configured to a near circular orbit and DefaultProp is configured to use Earth as the central body with Earth 10x10 gravity field and perturbations from the Sun and Moon. The default impulsive burn is in the VNB coordinate system. You may want to open the dialog boxes for these objects and look more closely at them as we will leave the settings of those objects at their default values.

Creating the Differential Corrector

To create a differential for use in the mission sequence:

1. Expand the Solvers folder in the Resources tree by clicking on the box with a plus sign in it.
2. Right click the Boundary Value Solvers folder and select Add --> DifferentialCorrector to create DC1.

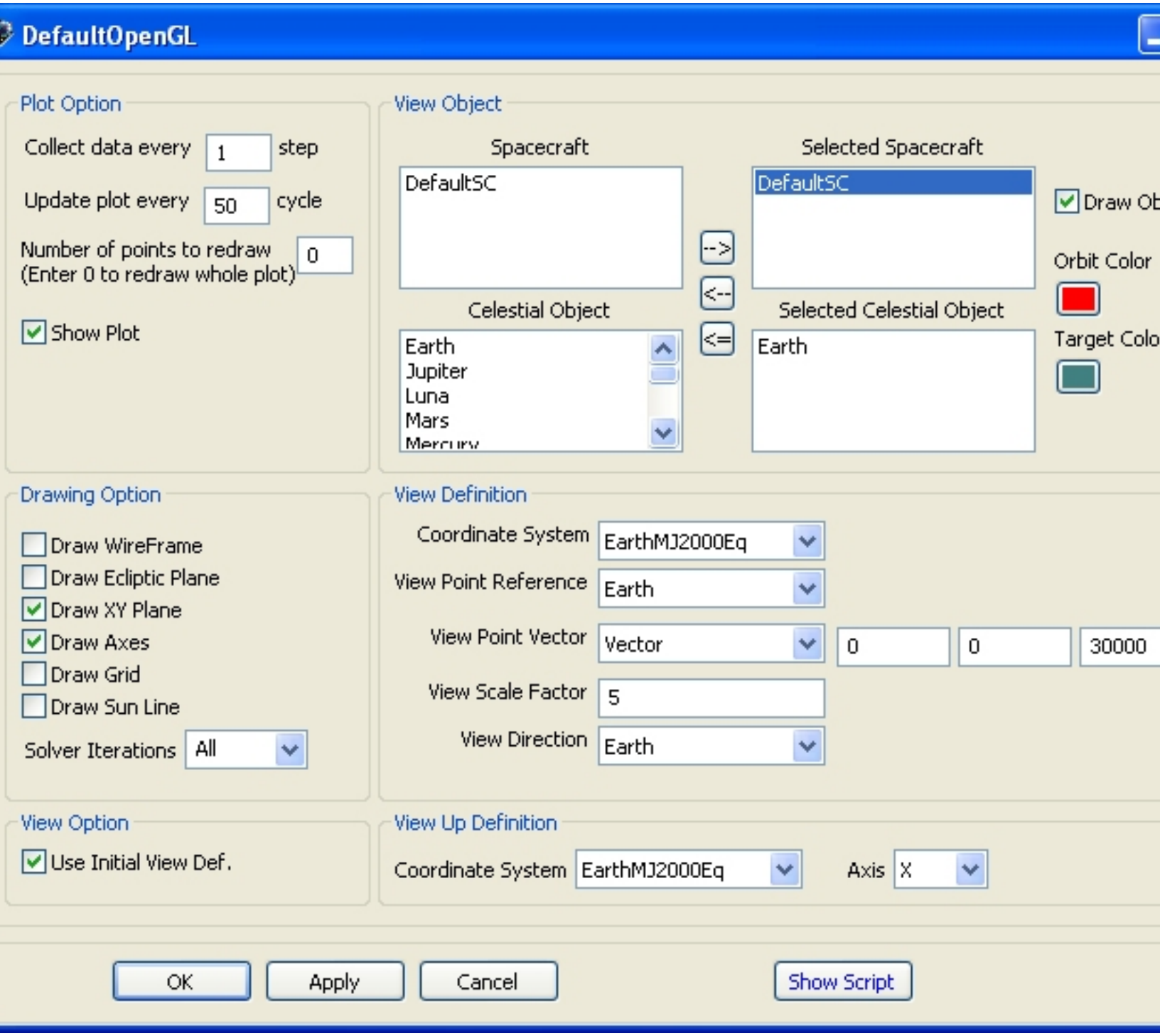
We'll leave the settings for the differential corrector at the default values.

Modifying the default Orbit View

We need to make minor modifications to Default Orbit View so that the entire final orbit will fit in the graphics window.

1. Open the dialog box for the DefaultOrbitView

2. Change the Solver Iterations input field, located in the Drawing Option group box, to the value All
3. Change the View Scale Factor input field to the value 5
4. Change the View Up Definition Axis to X
5. Click the Ok button to save changes and close the dialog box.



Creating and Configuring the Mission Tree

In this section you will create two targeting segments. The first targeting segment will raise orbit apogee to GEOsynchronous altitude (42165km). The second targeting sequence will circularize the orbit. Each targeting segment we will employ a Target command, a Vary command, a Maneuver, and Achieve command inside of the Target branch command.

Initial Propagate Sequence

- Propagate1
  - Set the stopping condition parameter as the Earth's periapsis. It can be found be right-clicking on DefaultSC.ElapsedSecs, under the Parameter heading.
  - The previous step will bring up the Parameter Select Dialog Box. Select Periapsis under the Object Properties list and click the Add Selected Item(s) button.

Script Syntax:

```
Propagate DefaultProp(DefaultSC) {DefaultSC.Earth.Periapsis};
```

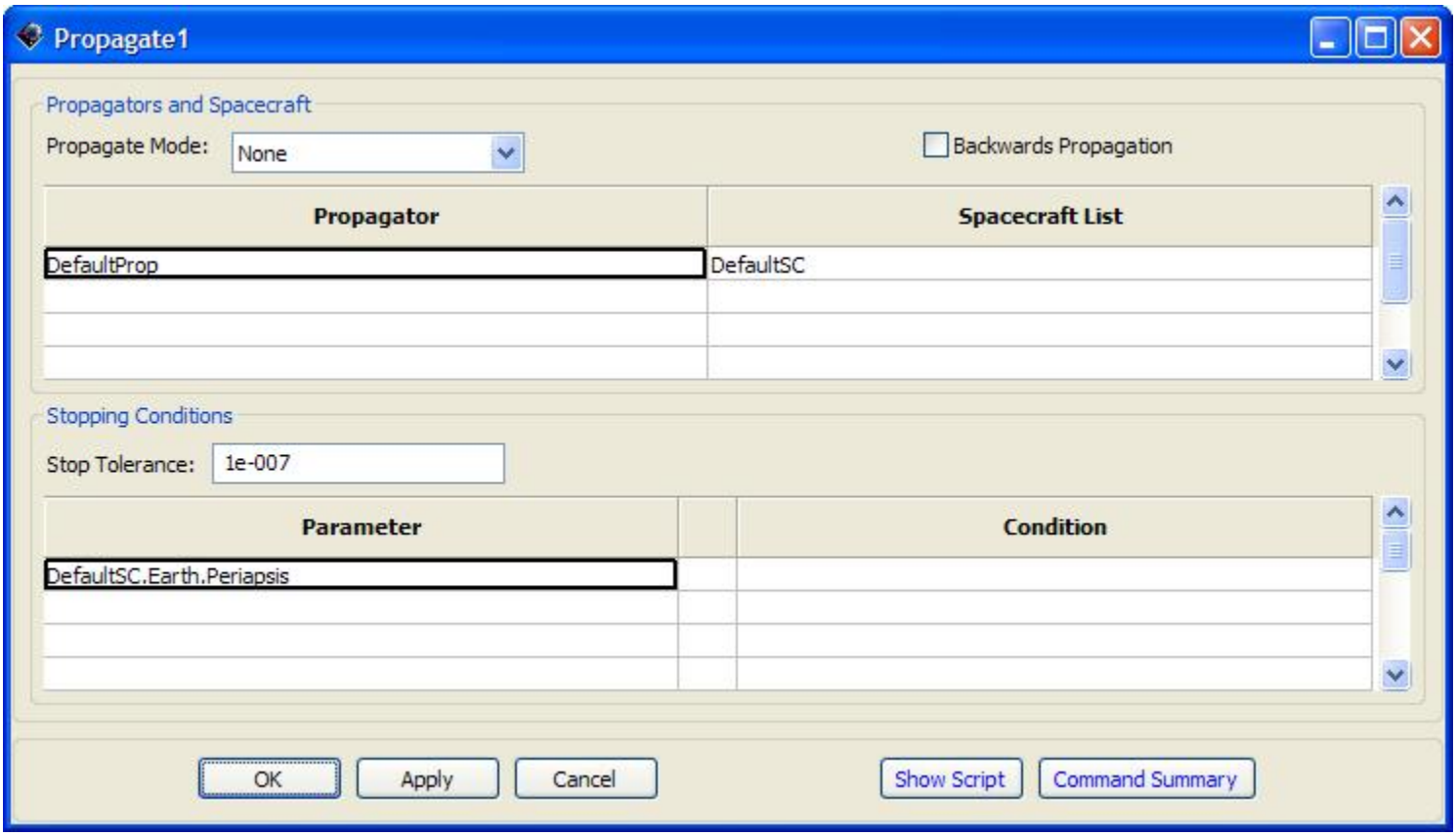


Figure: Propagate1 Propagate Command

First Targeting Sequence

- Target1: Insert this after Propagate1

- Vary1: Append this Vary to Target1
- Maneuver1: Insert this Maneuver after Vary
- Propagate2
  - This propagate should be inserted after the Maneuver
  - Set the stopping parameter as Earth's apoapsis. This process is the same as before when setting the first propagate command's stopping parameter as Earth's periapsis
- Achieve1:
  - This will go after Propagate2.
  - Note: From here on it will be assumed that you will follow the order of commands shown in the mission tree.

Script Syntax:

```
Target DC1;
   Vary DC1(DefaultIB.V = 0.5, {Perturbation = 0.0001, MaxStep = 0.2, Lower = 0.0,
   Maneuver DefaultIB(DefaultSC);
   Propagate DefaultProp(DefaultSC) {DefaultSC.Earth.Apoapsis};
   Achieve DC1(DefaultSC.Earth.RMAG = 42165.0, {Tolerance = 0.1});
EndTarget;  % For targeter DC1
```

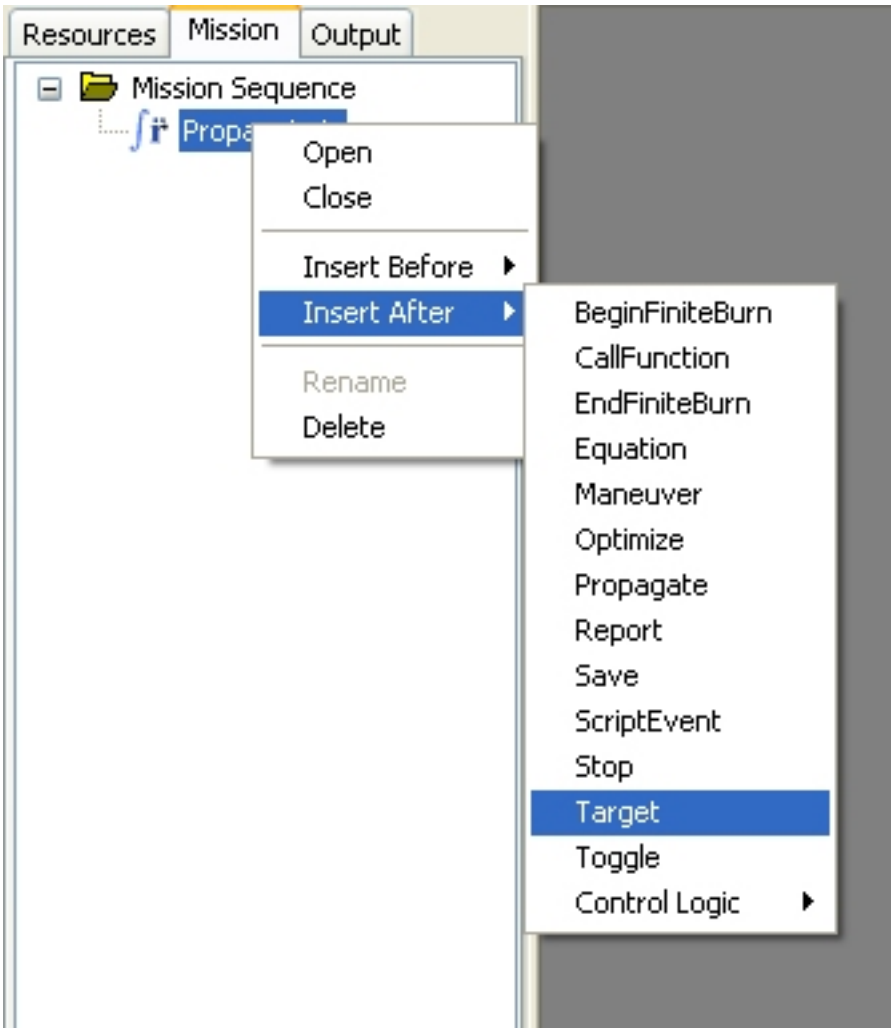


Figure: Appending a Targeting Sequence

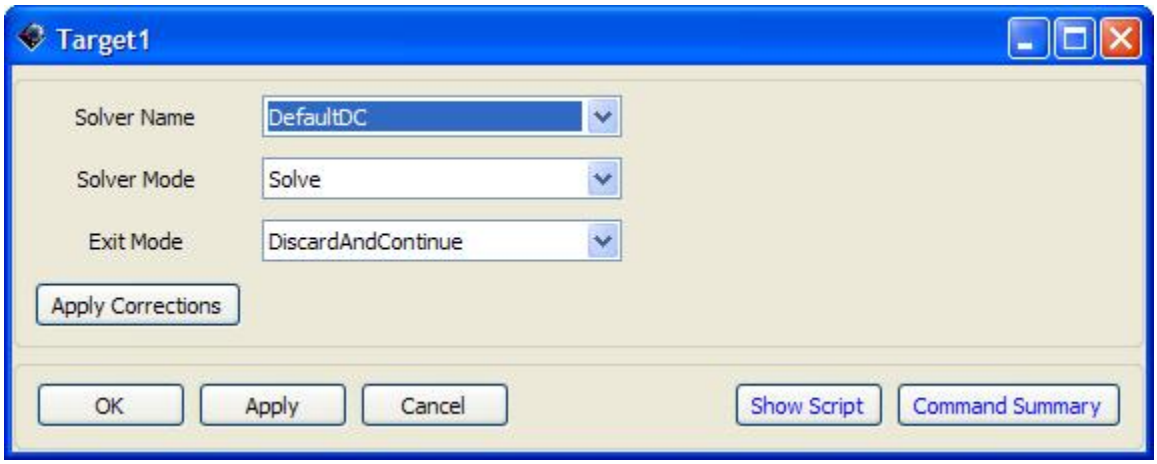


Figure: Target1 Target Command

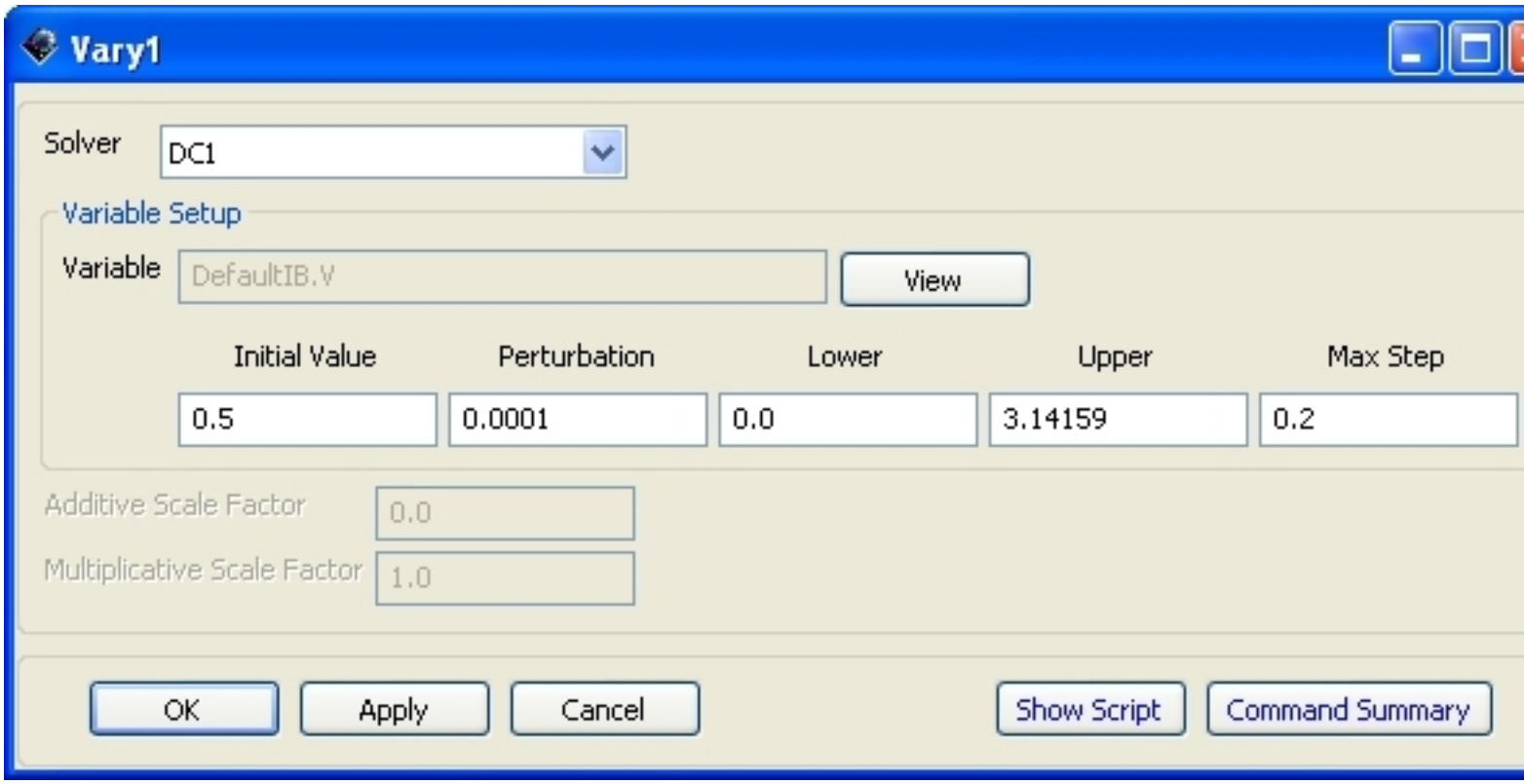


Figure: Vary1 Vary Command

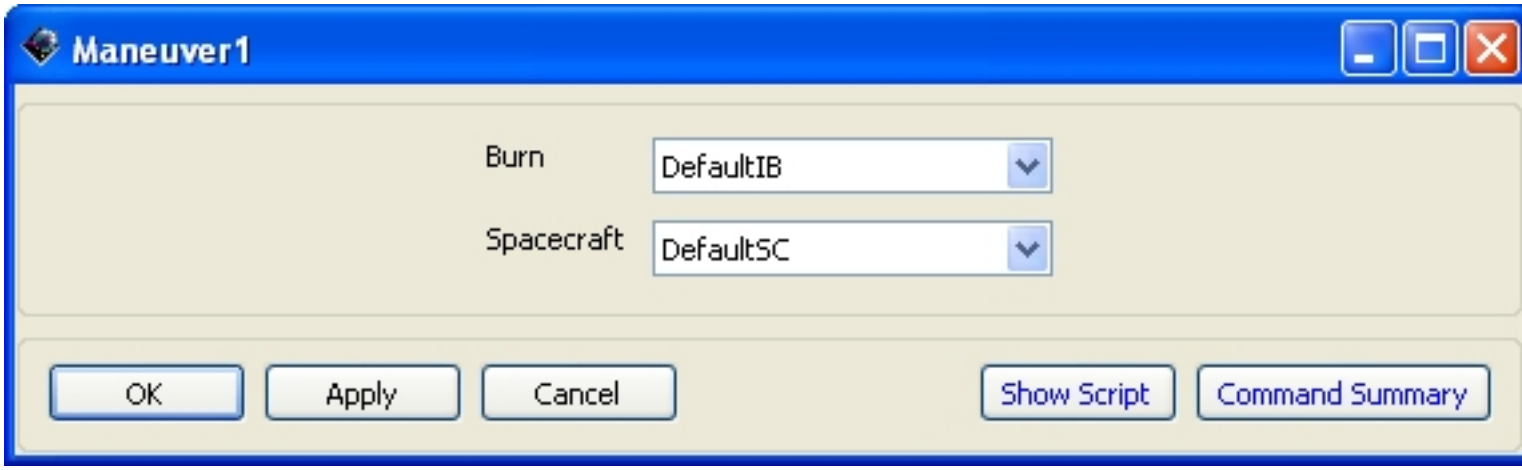


Figure: Maneuver1 Maneuver Command

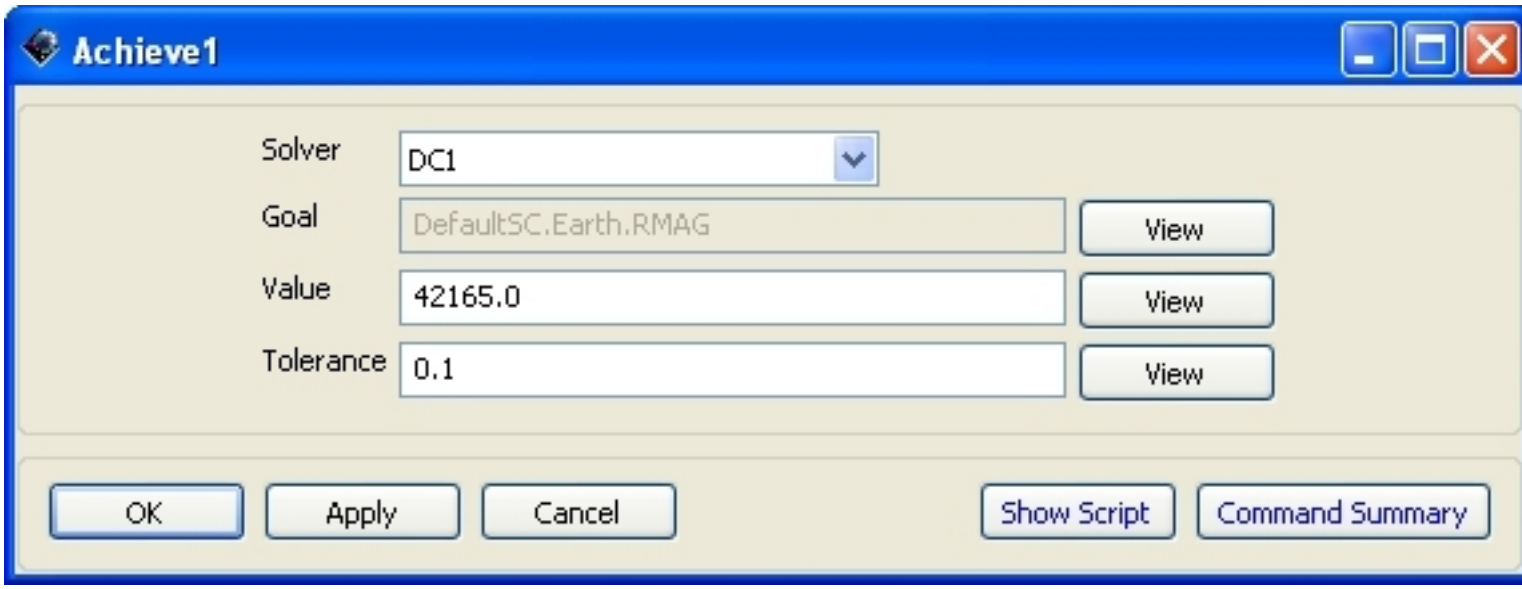


Figure: Achieve1 Achieve Command

Second Targeting Sequence

- Target2:
  - Vary2
  - Maneuver2
  - Achieve2: set the Goal field to be earth's SMA

Script Syntax:

```
Target DefaultDC;
   Vary DefaultDC(DefaultIB.V = 0.5, {Perturbation = 0.0001, MaxStep = 0.2, Lower
   Maneuver DefaultIB(DefaultSC);
   Achieve DefaultDC(DefaultSC.Earth.SMA = 42165.0, {Tolerance = 0.1});
EndTarget;
```

Final Propagate Sequence

- Propagate3: set the stopping condition parameter to be one elapsed day (86400 seconds).

Script Syntax:

```
Propagate DefaultProp(DefaultSC) {DefaultSC.ElapsedSecs = 86400.0};
```

Running the Mission

Resource Mission and Output Trees

The openGL image below was stopped after the first propagate command. The spacecraft started at its initial conditions and propagated forward to the periapsis point of the first orbit. The next openGL image

leaves off where the last image stopped. The transfer orbit is targeted and the spacecraft propagates from the initial orbit periapsis point to the transfer orbit apoapsis point. This image has the solver/targeter iterations turned off. The final image shown below shows all the orbits. The teal-ish lines represent the solver/targeter iterations for the transfer orbit. You don't see these kind of line for the target sequence from the transfer to final orbit because there is no propagation involved. The targeting only occurs at one spot.

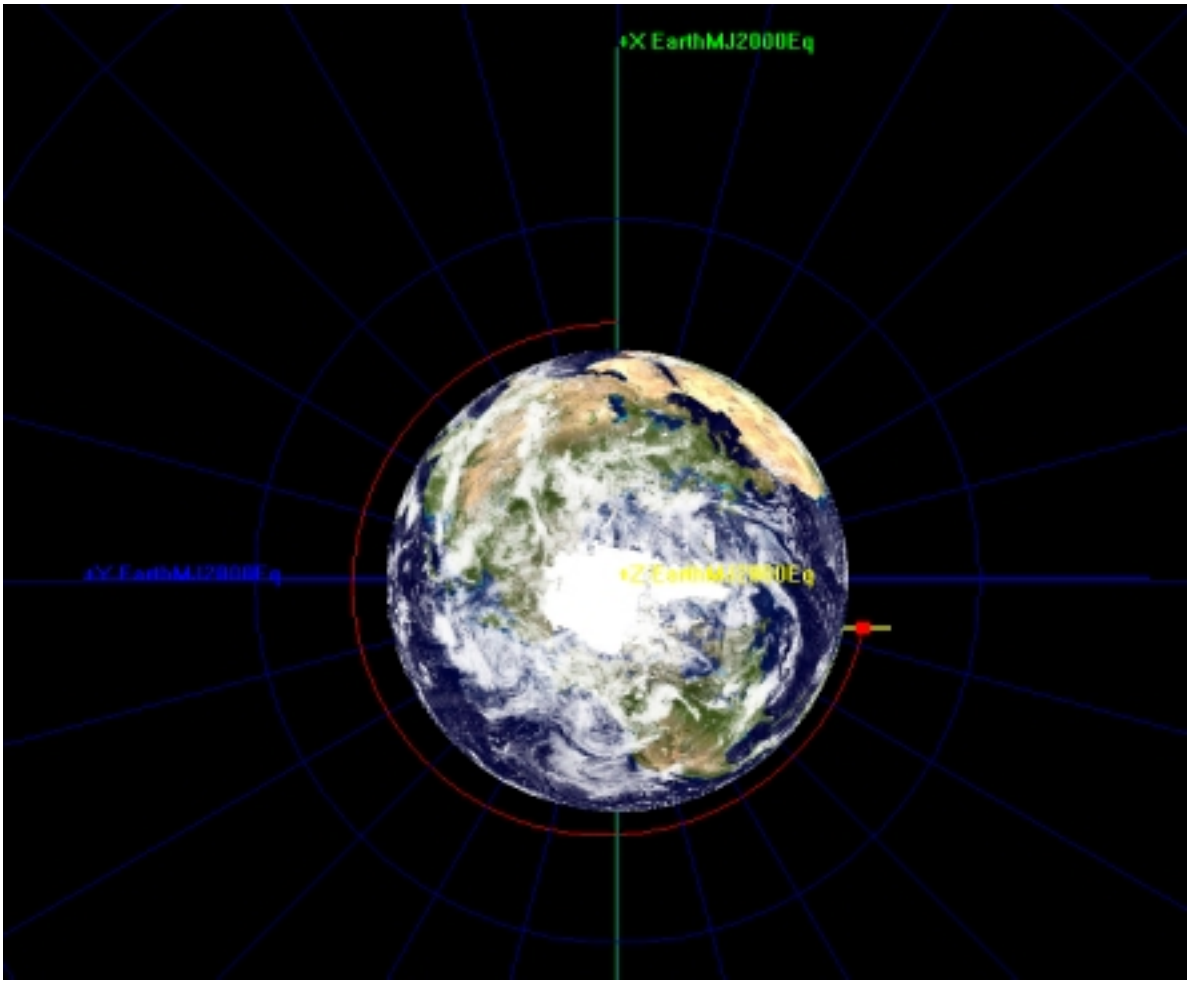


Figure: Output After Initial Propagate Sequence

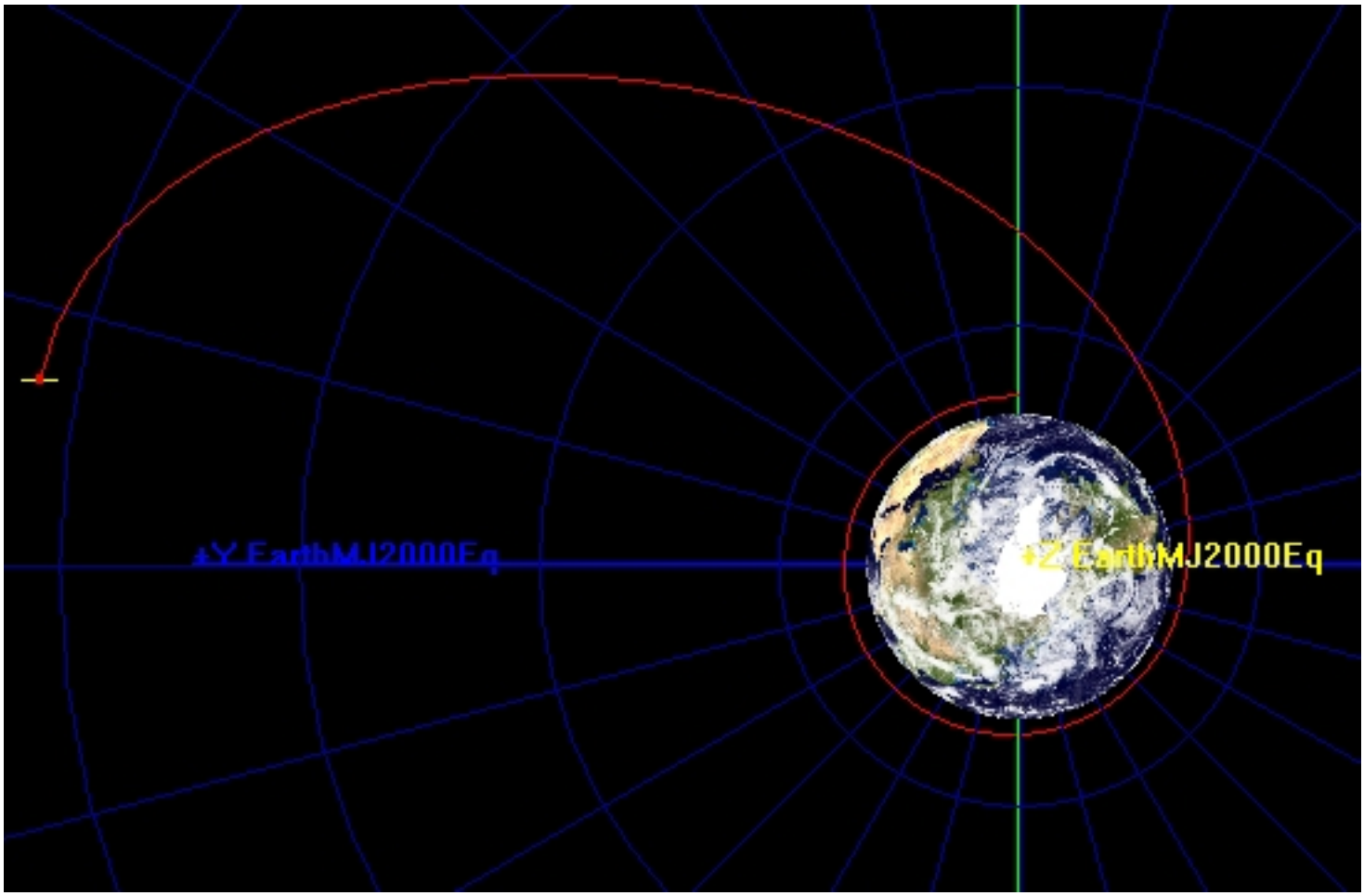


Figure: Output After First and Second Targeting Sequences

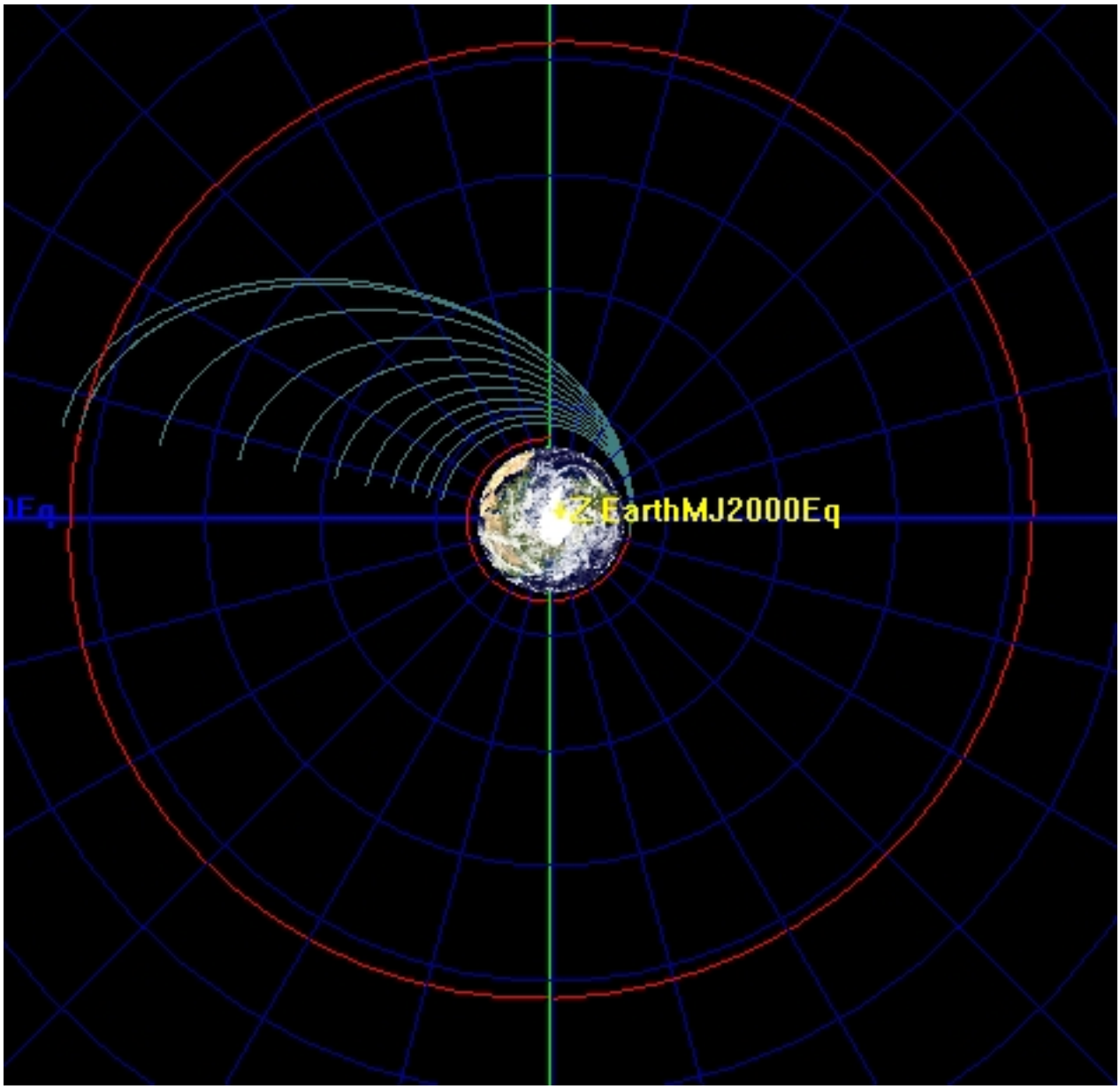


Figure: Output After Final Propagate Sequence

Chapter 12. Creating a Report

This tutorial was started from the default settings in the September 30, 2008 build of GMAT.

Objective and Overview

The objective of this tutorial is to demonstrate how to report parameters out to a file using two different reporting techniques, as well as how to use strings to improve the readability of a report file.

Download the script file: ReportFile.script [scripts/ReportTutorial.script]

Prerequisites

- Objects modified (use if you have trouble with this tutorial):
  - ReportFile
- Commands modified (use if you have trouble with this tutorial):
  - Report Command
  - ScriptEvent Command

Mission Description

- Objective: Use the Report Object to output parameters to an ascii file
- Assume: N/A
- Find: N/A

Resource, Mission, and Output trees

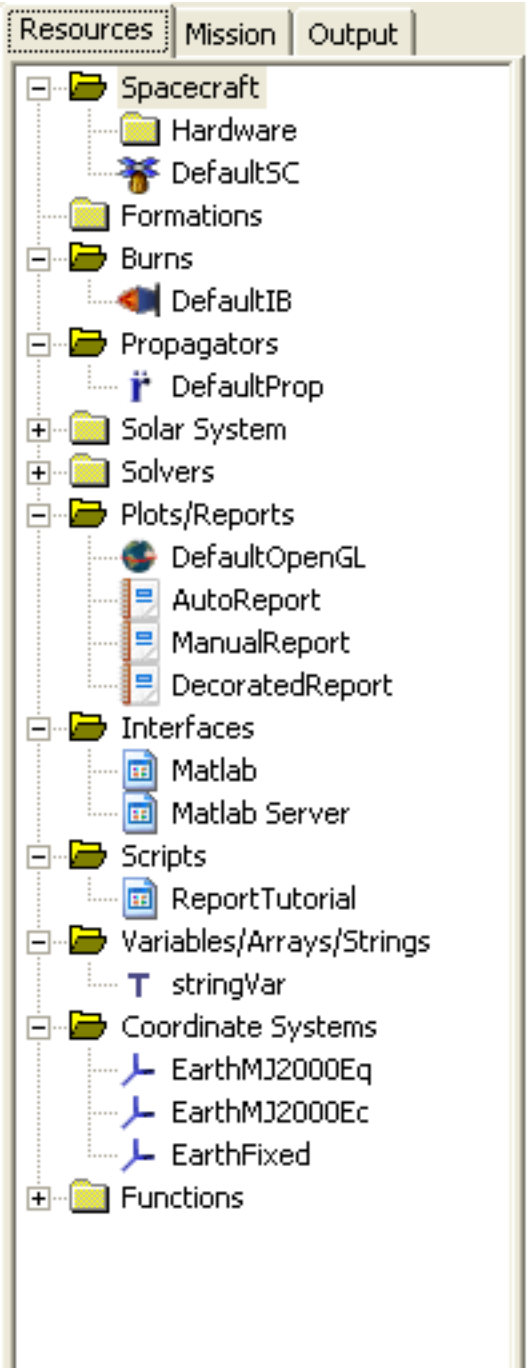


Figure: Creating a Report Resource, Mission, and Output trees

Creating and Configuring the Resource Tree

Objects Required

- Plots/Reports:

  - ReportFile - AutoReport, ManualReport, DecoratedReport

- Variables/Arrays/Strings:

  - String Variable - StringVar

Creating and Modifying Objects

Add ReportFile Objects

The ReportFile has two features that both allow a user to output data to an ASCII text file. One feature of a ReportFile is to output data at every integrator step and the other is to output data at the user's discretion using Report commands in the Mission Sequence.

- Right click on the Plots/Reports folder and add a ReportFile.

- Add two more ReportFiles for a total of three ReportFiles.

- Rename the three ReportFiles by right clicking the ReportFile and selecting Rename. The new ReportFile names are as followed: AutoReport, ManualReport, and DecoratedReport.

- Verify that your Plots/Reports Resource Tree settings are identical to the image below.

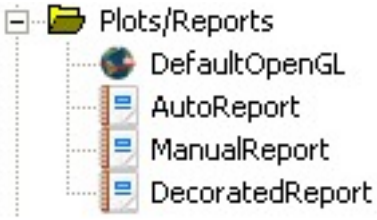


Figure: Plots/Reports Resource Tree Folder

Configure AutoReport ReportFile

The AutoReport ReportFile is used to report data every integrator step size once the mission is run.

- Open the properties panel of the AutoReport ReportFile and remove all parameters in the Selected input field by clicking the Remove All button.

- Add the following parameters to the Selected input field in the order specified: DefaultSC.UTCGregorian, DefaultSC.EarthMJ2000Eq.X, DefaultSC.EarthMJ2000Eq.Y, DefaultSC.EarthMJ2000Eq.Z, DefaultSC.EarthMJ2000Eq.VX, DefaultSC.EarthMJ2000Eq.VY, DefaultSC.EarthMJ2000Eq.VZ

- Change the File input field to ./output/AutoReport.txt .

- Verify that your settings are identical to the image below.

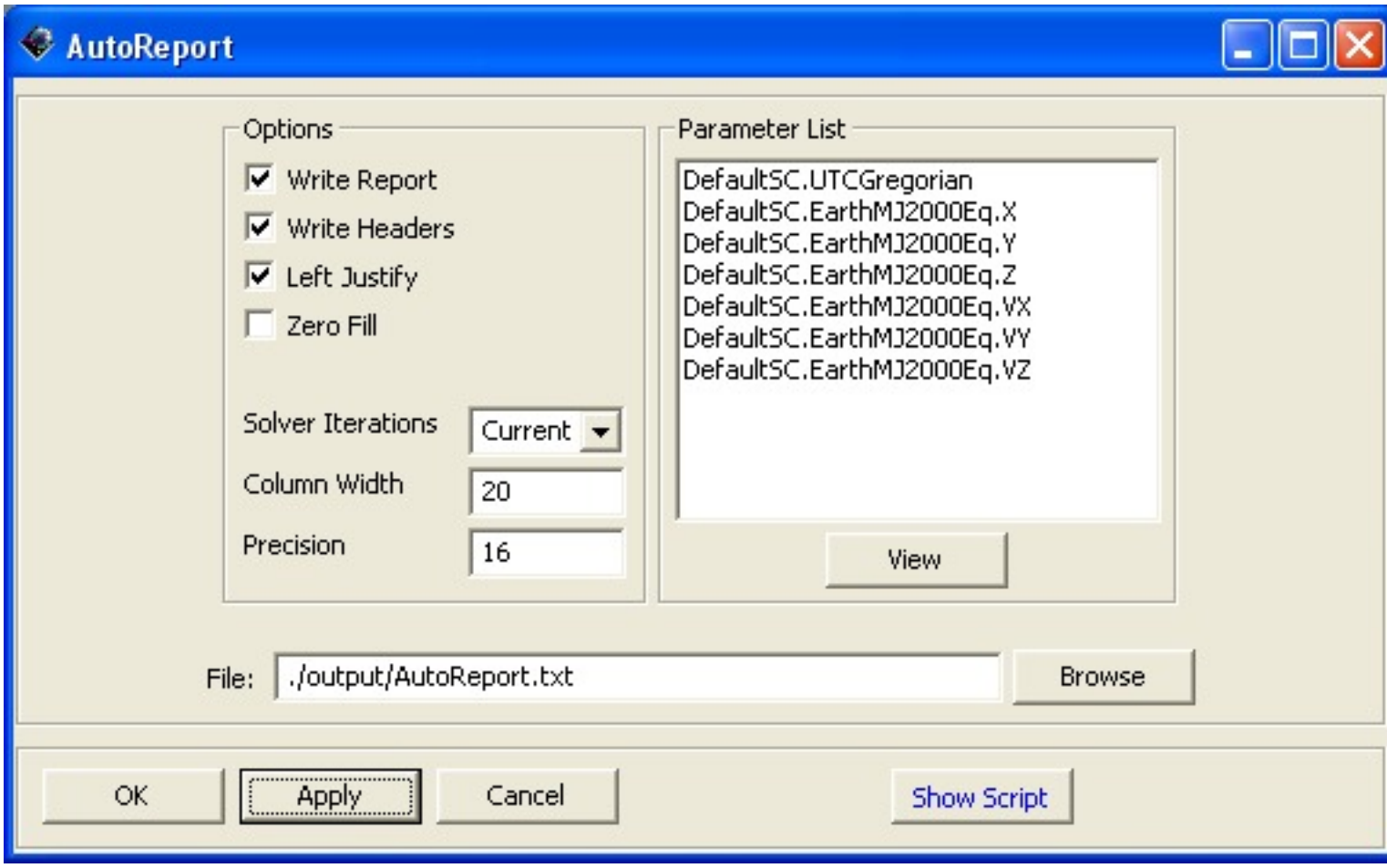


Figure: AutoReport ReportFile Settings

Script Syntax:

```
Create ReportFile AutoReport;
GMAT AutoReport.SolverIterations = Current;
GMAT AutoReport.Filename = './output/AutoReport.txt';
GMAT AutoReport.Precision = 16;
GMAT AutoReport.Add = {DefaultSC.UTCGregorian, DefaultSC.EarthMJ2000Eq.X, DefaultS
GMAT AutoReport.WriteHeaders = On;
GMAT AutoReport.LeftJustify = On;
GMAT AutoReport.ZeroFill = Off;
GMAT AutoReport.ColumnWidth = 20;
```

Configure ManualReport ReportFile

- Open the properties panel of the ManualReport ReportFile
- Remove all parameters in the Selected input field by clicking the Remove All button.
- Change the File input field to ./output/ManualReport.txt .
- Verify that your settings are identical to the image below.

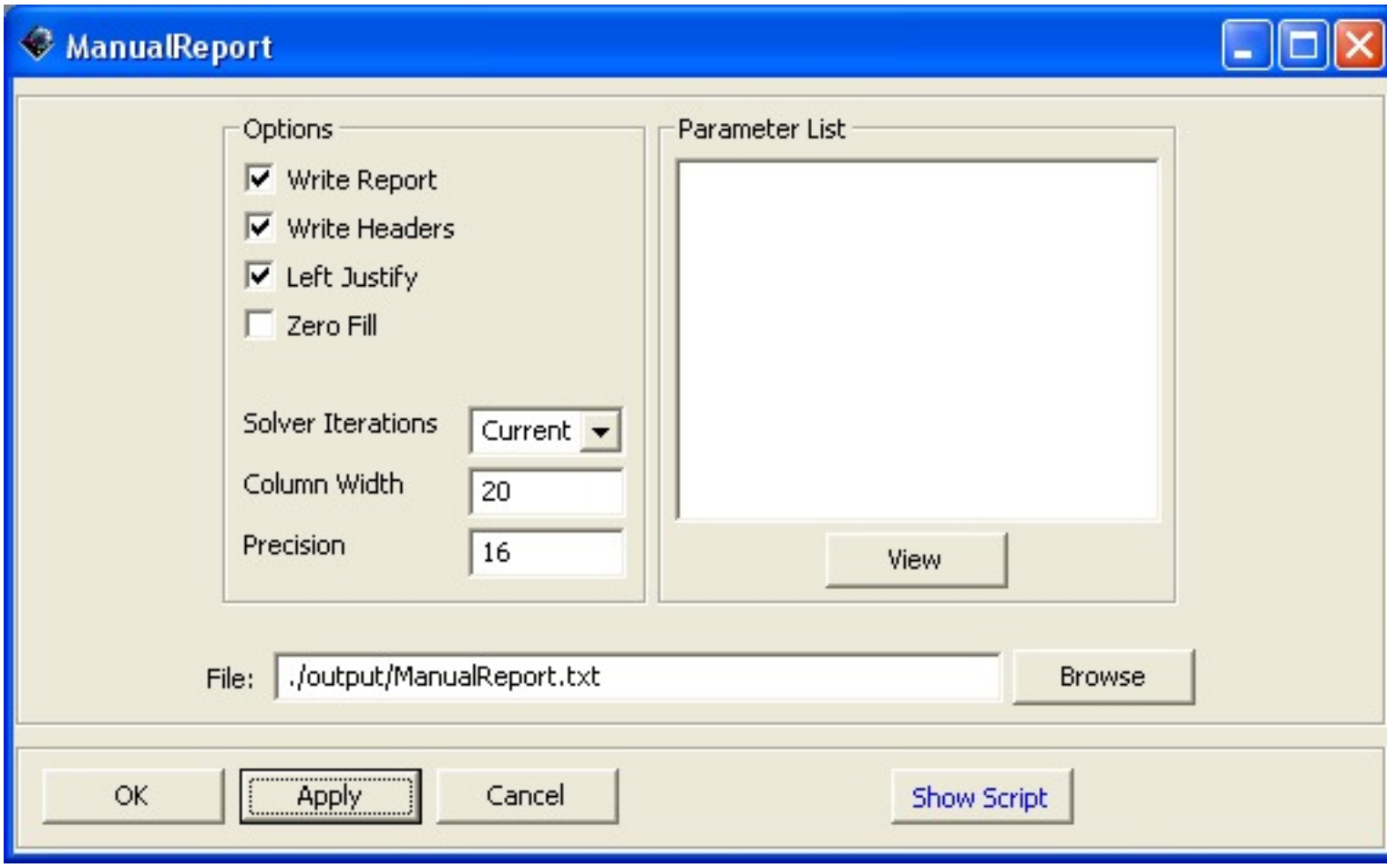


Figure: ManualReport ReportFile Settings

Script Syntax:

```
Create ReportFile ManualReport;
GMAT ManualReport.SolverIterations = Current;
GMAT ManualReport.Filename = './output/ManualReport.txt';
GMAT ManualReport.Precision = 16;
GMAT ManualReport.WriteHeaders = On;
GMAT ManualReport.LeftJustify = On;
GMAT ManualReport.ZeroFill = Off;
GMAT ManualReport.ColumnWidth = 20;
```

Configure DecoratedReport ReportFile

The DecoratedReport ReportFile is identical to AutoReport ReportFile, besides the file path of the report.

- Open the properties panel of the DecoratedReport ReportFile
- Remove all parameters in the Selected input field by clicking the Remove All button.
- Add the following parameters to the Selected input field in the order specified: DefaultSC.UTCGregorian, DefaultSC.EarthMJ2000Eq.X, DefaultSC.EarthMJ2000Eq.Y,

DefaultSC.EarthMJ2000Eq.Z, DefaultSC.EarthMJ2000Eq.VX, DefaultSC.EarthMJ2000Eq.VY, DefaultSC.EarthMJ2000Eq.VZ

- Change the File input field to ./output/DecoratedReport.txt .

Script Syntax:

```
Create ReportFile DecoratedReport;
GMAT DecoratedReport.SolverIterations = Current;
GMAT DecoratedReport.Filename = './output/DecoratedReport.txt';
GMAT DecoratedReport.Precision = 16;
GMAT DecoratedReport.Add = {DefaultSC.UTCGregorian, DefaultSC.EarthMJ2000Eq.X, Def
GMAT DecoratedReport.WriteHeaders = On;
GMAT DecoratedReport.LeftJustify = On;
GMAT DecoratedReport.ZeroFill = Off;
GMAT DecoratedReport.ColumnWidth = 20;
```

Add a String Variable for Decorated Report

- Right click the Variables/Arrays folder and select Add --> String.
- Navigate to the String group box and enter a string Name of stringVar.
- Leave the Value input field blank.
- Click the Create button and click OK to exit out of that panel.
- Verify that your settings are identical to the image below.

Figure: String Variable Creation

Script Syntax:

```
Create String stringVar;
```

Creating and Configuring the Mission Tree

Add Misc. Commands

- Select the Mission tab in the left pane to switch over to the Mission Sequence tree.
- Right click on the Propagate1 command object and Insert After a ScriptEvent.
- Right click on the newly created ScriptEvent(BeginScript) and Insert After a Propagate.
- Verify that your settings are identical to the image below.

(Folder icons and command numbers may not be the same)

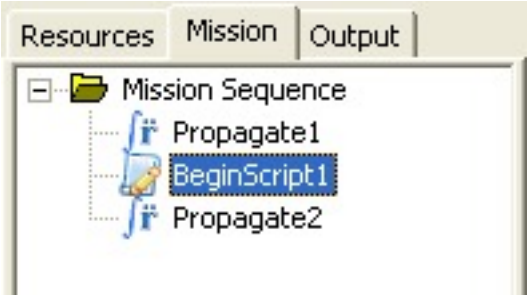


Figure: Mission Tree Configuration 1

Add Report Commands for Manual Report

For the ManualReport ReportFile, we will add a Report command before and after the entire mission sequence to see what the initial state and final state is.

- Right click the first Propagate command and Insert Before a Report command.
- Add the same parameters as the (Configure AutoReport ReportFile):

  DefaultSC.UTCGregorian, DefaultSC.EarthMJ2000Eq.X, DefaultSC.EarthMJ2000Eq.Y, DefaultSC.EarthMJ2000Eq.Z, DefaultSC.EarthMJ2000Eq.VX, DefaultSC.EarthMJ2000Eq.VY, DefaultSC.EarthMJ2000Eq.VZ
- Select the ManualReport ReportFile before closing the Report command properties panel.
- Right click the last Propagate command and Insert After a Report command.
- Add the same parameters as above and select ManualReport for the ReportFile.
- Verify that your settings are identical to the image below.

  (Folder icons and command numbers may not be the same)

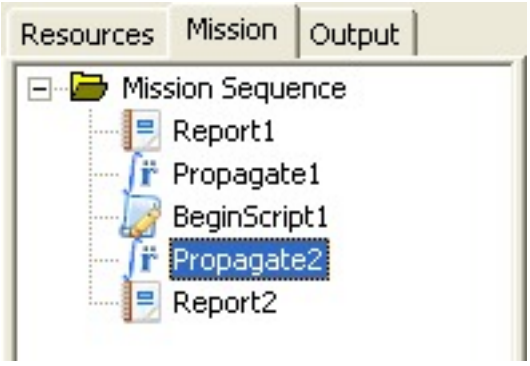


Figure: Mission Tree Configuration 2

Output String Values for Decorated Report

- Right click the first Propagate command and Insert Before a ScriptEvent command.

- Open the properties panel for the first ScriptEvent
- Enter the following between the BeginScript/EndScript text and click OK to exit the panel:

```
GMAT DecoratedReport.WriteHeaders = Off; % Turn report headers off
stringVar = '==================================================='; % Heading
Report DecoratedReport stringVar; % Outputs the string variable to the report
stringVar = GMAT Report File in Time X Y Z VX VY VZ format; % Heading
Report DecoratedReport stringVar; % Outputs the string variable to the report
stringVar = '==================================================='; % Heading
Report DecoratedReport stringVar; % Outputs the string variable to the report
GMAT DecoratedReport.WriteHeaders = On; % Turn report headers back on
```

  Observe how WriteHeaders was turned off while outputting string data. If you don't do this, you will get the word stringVar showing up above each report output line.

- Verify that your settings are identical to the image below.

  (Folder icons and command numbers may not be the same)

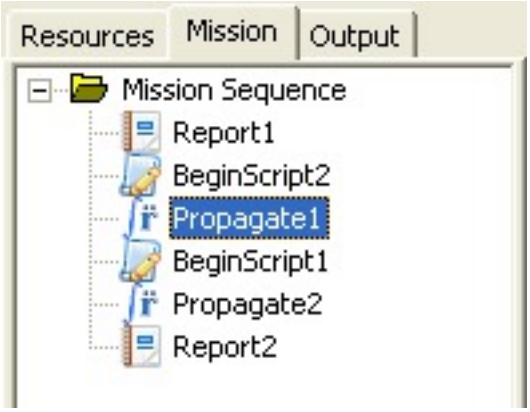


Figure: Mission Tree Configuration 3

This concludes the configuration of the commands needed for the Mission Sequence.

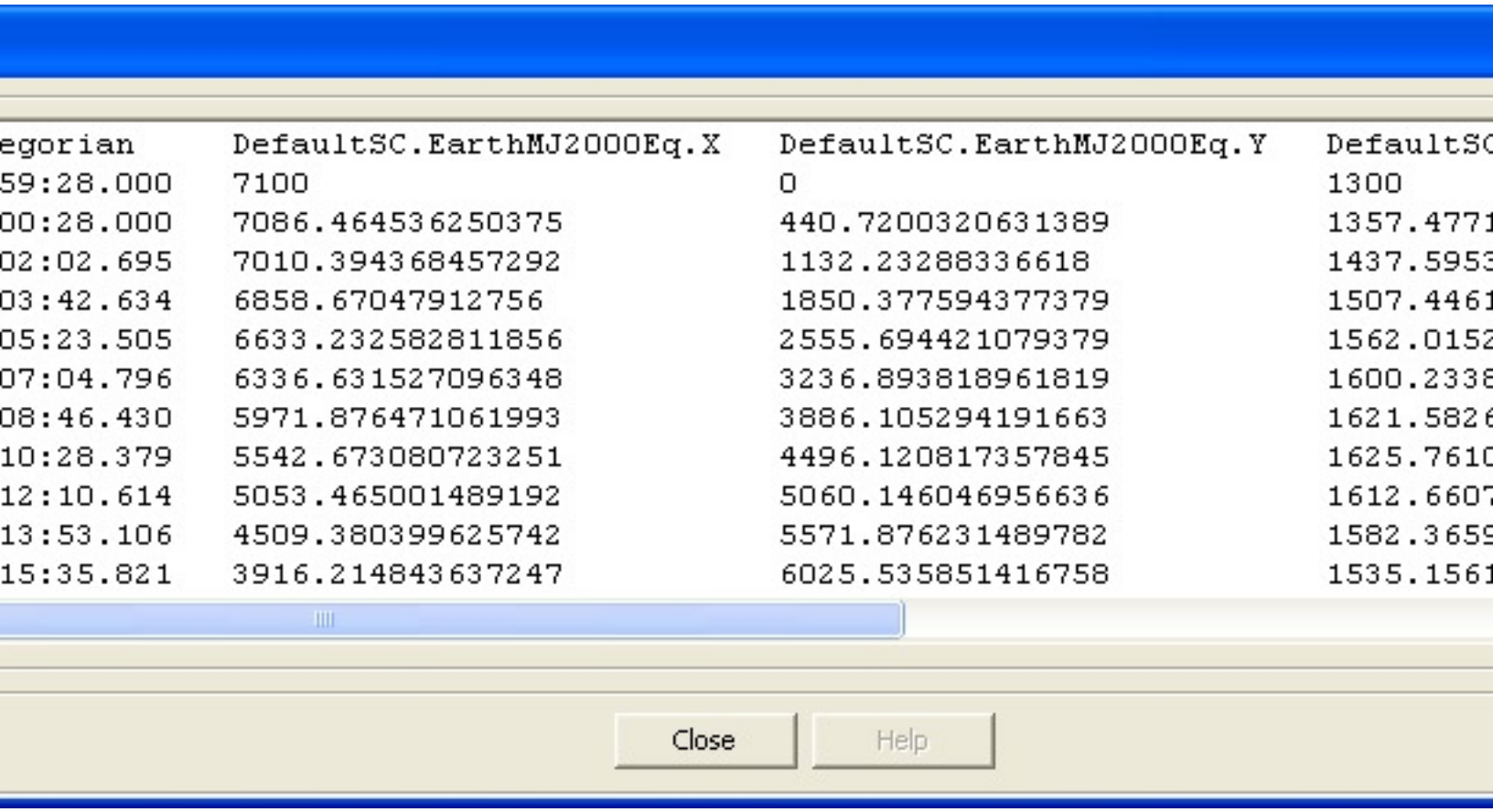
Running the Mission

Run the mission and navigate to the Output Tab.

AutoReport Output

Open the AutoReport file. The contents are a heading and data associated with the parameters in the heading at each propagator integration step.

The AutoReport output should look like the image below.



egorian	DefaultSC.EarthMJ2000Eq.X	DefaultSC.EarthMJ2000Eq.Y	DefaultSC
59:28.000	7100	0	1300
00:28.000	7086.464536250375	440.7200320631389	1357.4771
02:02.695	7010.394368457292	1132.23288336618	1437.5953
03:42.634	6858.67047912756	1850.377594377379	1507.4461
05:23.505	6633.232582811856	2555.694421079379	1562.0152
07:04.796	6336.631527096348	3236.893818961819	1600.2338
08:46.430	5971.876471061993	3886.105294191663	1621.5826
10:28.379	5542.673080723251	4496.120817357845	1625.7610
12:10.614	5053.465001489192	5060.146046956636	1612.6607
13:53.106	4509.380399625742	5571.876231489782	1582.3659
15:35.821	3916.214843637247	6025.535851416758	1535.1561

Figure: AutoReport Output Results

ManualReport Output

Open the ManualReport file. The contents are a heading followed by the initial spacecraft state, a heading, and the final spacecraft state.

The ManualReport output should look like the image below.

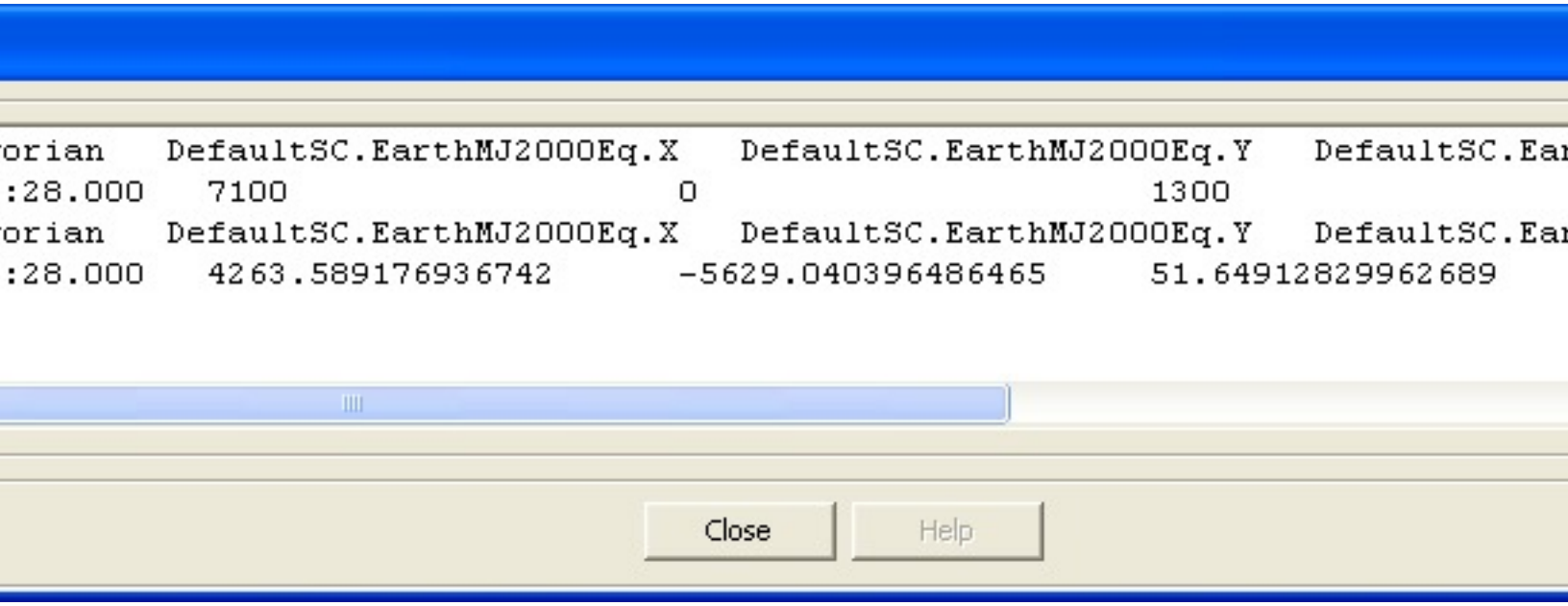
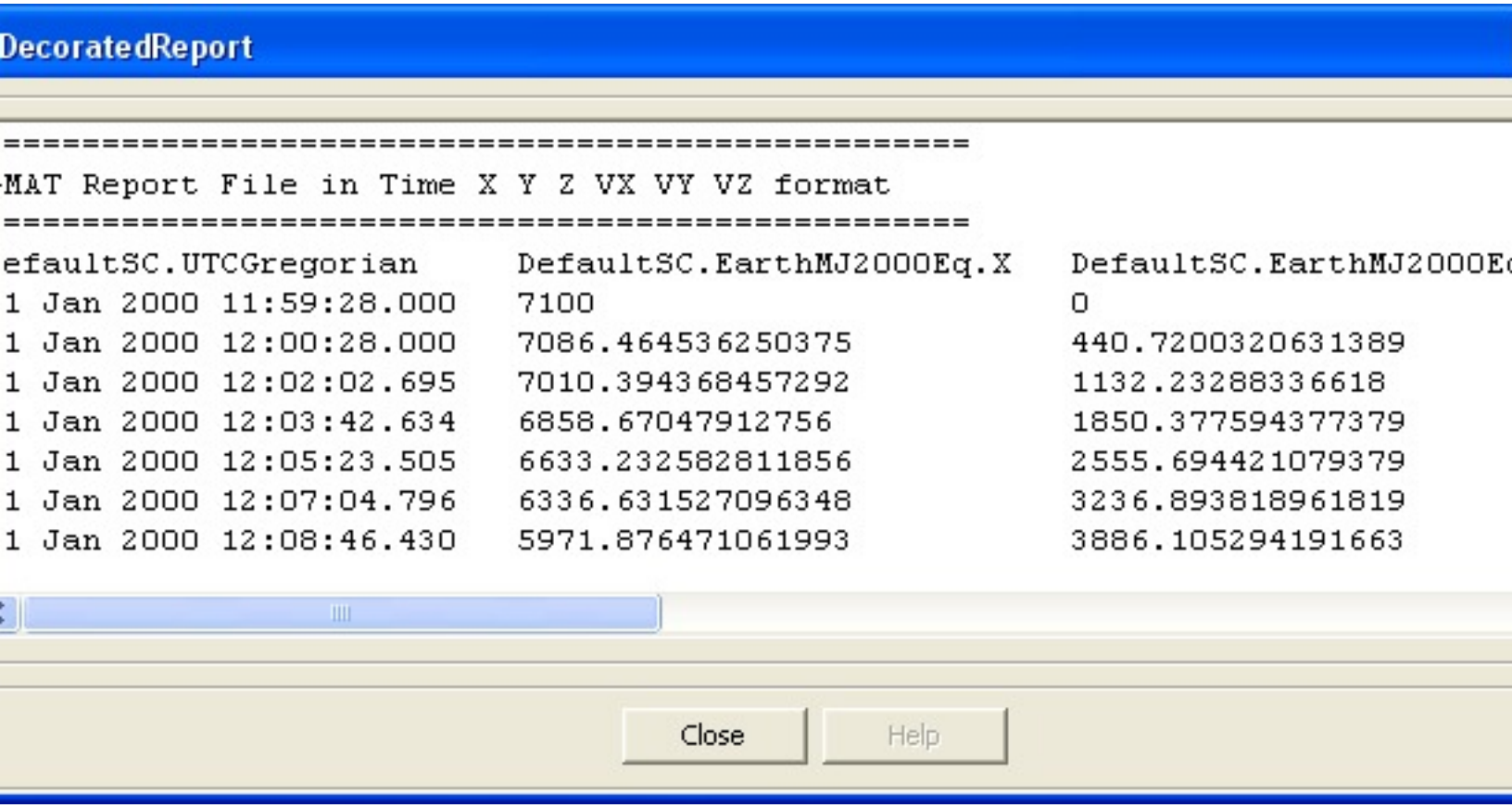


Figure: ManualReport Output Results

DecoratedReport Output

Open the DecoratedReport file. The contents are the custom heading using string variables, followed by the default heading, and data associated with the parameters in the default heading at each propagator integration step.

The DecoratedReport output should look like the image below.



DecoratedReport

```
==================================================
-MAT Report File in Time X Y Z VX VY VZ format
==================================================
efaultSC.UTCGregorian      DefaultSC.EarthMJ2000Eq.X   DefaultSC.EarthMJ2000E
1 Jan 2000 11:59:28.000    7100                        0
1 Jan 2000 12:00:28.000    7086.464536250375           440.7200320631389
1 Jan 2000 12:02:02.695    7010.394368457292           1132.23288336618
1 Jan 2000 12:03:42.634    6858.67047912756            1850.377594377379
1 Jan 2000 12:05:23.505    6633.232582811856           2555.694421079379
1 Jan 2000 12:07:04.796    6336.631527096348           3236.893818961819
1 Jan 2000 12:08:46.430    5971.876471061993           3886.105294191663
```

Close Help

Figure: DecoratedReport Output Results

Chapter 13. Running GMAT Scripts from MATLAB

This tutorial was written for GMAT 2011a.

Overview

GMAT was designed to allow users to run GMAT scripts through MATLAB®. This feature gives the user greater control and flexibility of GMAT that cannot be done with just the GMAT to MATLAB® interface or GMAT alone. For example, if a user would like to dynamically change scripts and run them, that can currently only be done using the MATLAB® to GMAT interface. A MATLAB® script can also be generated to run GMAT scripts that are located in mutliple folders.

If, after running through this tutorial, you still have difficulties with the interfaces between MATLAB® and GMAT working, visit our ???MATLAB Help Forum [http://gmat.ed-pages.com/forum/viewforum.php?f=14] or our MATLAB<->GMAT Interface FAQ [http://gmat.ed-pages.com/forum/viewtopic.php?f=14&t=65] forum topic.

Files and Folders to be used

- [rootGMATpath] - folder with the GMAT executable
- Ex_TargetHohmannTransfer.script - Sample Mission script used for running GMAT from Matlab
- runMatlabToGMAT.m [scripts/runMatlabToGMAT.m] - MATLAB® script that runs the GMAT script
- runMatlabToGMATsimple.m [scripts/runMatlabToGMATsimple.m] - simplified MATLAB® script that runs the GMAT script

Procedure

1. Place the downloaded script runMatlabToGMAT.m script into the [rootGMATpath]\matlab\ directory

   WARNING! This script clears the MATLAB® workspace. Save your current MATLAB® data if it's needed

2. Make sure the script Ex_TargetHohmannTransfer.script is in the [rootGMATpath]\input\SampleMissions\ directory

3. Unless it has been done already, open MATLAB® and add [rootGMATpath]\matlab\ directory and the sub-directories to the MATLAB® path.

4. Open GMAT and start the Matlab Server.

   *The server can be started by navigating in the Resources tree to the Interfaces folder. Right click the Matlab Server object and select Start.

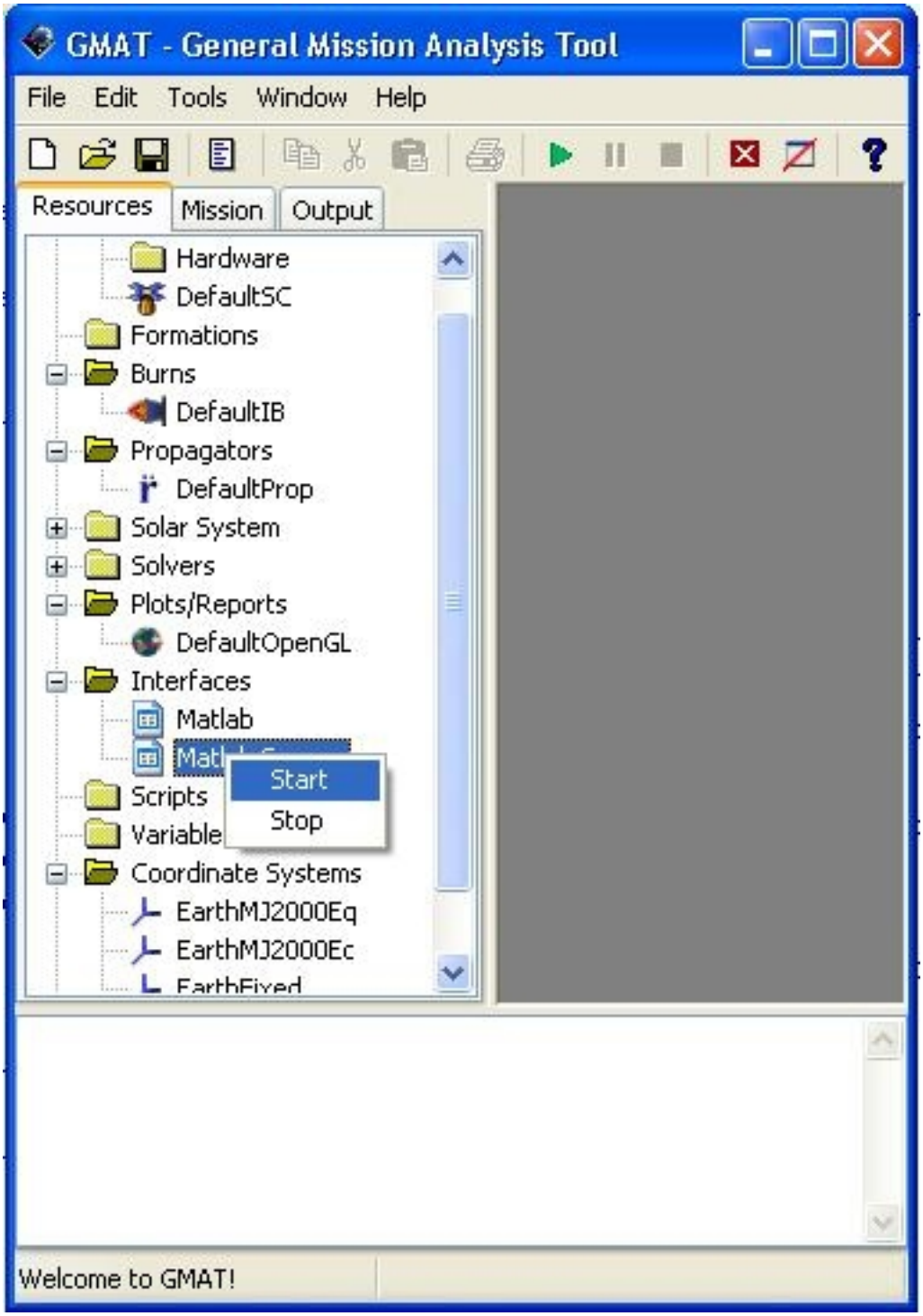


Figure: Opening Matlab Server in GMAT Resource Tree

5. Run runMatlabToGMAT.m in MATLAB®

6. Open a file browser to the [rootGMATpath]\output\SampleMissions\ directory and open the ASCII Ex_TargetHohmannTransfer.report script in a text editor.

7. Scroll down to the last few lines, and notice the extra lines indicating that this script was run from MATLAB® . If the Ex_TargetHohmannTransfer.script file is run from GMAT, these lines will not be present.

8. Congratulations - you have finished the main section of this tutorial. Be sure to open the runMatlabToGMAT.m script, to understand how GMAT is controlled by MATLAB® . This MATLAB® script is heavily commented to explain what is being done. If you still have questions, feel free to post them at our GMAT Source Forge forums (http://sourceforge.net/projects/gmat/)

Additional Instructions

The runMatlabToGMAT.m Matlab script might be too complicated for novice MATLAB® users. If so, use a simplified version of the script, runMatlabToGMATsimple.m. For this script a .m file is needed, so copy the Ex_TargetHohmannTransfer.script and rename the copy as Ex_TargetHohmannTransfer.m . Follow the above instructions but first rename the runMatlabToGMAT.m file to runMatlabToGMATsimple.m.

Chapter 14. Force Models

This tutorial was started from the default settings in the June 9, 2008 build of GMAT.

Objective and Overview

This tutorial demonstrates how to set up different force models and then propagate a spacecraft using those models. Modifying the types of force models affecting a spacecraft at certain points in a mission can be advantageous, for example, in simplifying a mission sequence, increasing accuracy, or reducing computing time.

You can download the script file and run it beforehand to see the final results of this tutorial: ForceModelsTutorial.script [scripts/ForceModelsTutorial.script]

Prerequisites

- Basic Understanding of how to create and propagate a spacecraft, as in Tutorial Creating and Propagating a Spacecraft

Mission Description

- Objective: Propagate a spacecraft using several different force models. Compare the results from different models and differences in computing times.
- Assume:
  - A given spacecraft in low earth orbit (LEO) with
    - A UTCGregorian Epoch of 04 Jan 2003 00:00:00.000
    - An SMA of 11295.67681418613
    - An Eccentricity of 0.2032349944098091
- Find: (Observation only)

Creating and Configuring the Resource Tree

This mission requires only a simple resource tree. There is a single spacecraft, one OpenGL Plot, one report, and a series of 6 propagators to accomplish this mission. Everything else should be default. The steps on how to setup the resource tree is presented below.

Objects Required

- Spacecraft: Sat1
- Propagators: RKV89PointMass, RKV89ThirdBodies, RKV8912x12, RKV89MSISE90, RKV89JR, RKV89SRP
- Plots/Reports: OpenGLSat1, ForceModelRpt

Creating and Modifying Objects

- Spacecraft (Note: Figure below shows how the resulting Sat1 properties window should look like. Similar for other steps)
  - Rename DefaultSC to Sat1
  - Change the Epoch Format to UTCGregorian
  - Set the Epoch to 04 Jan 2003 00:00:00.000
  - Change the State Type to Keplerian
  - Set SMA to 11295.67681418613km
  - Set the Eccentricity to 0.2032349944098091
  - Set the remaining orbital elements to zero

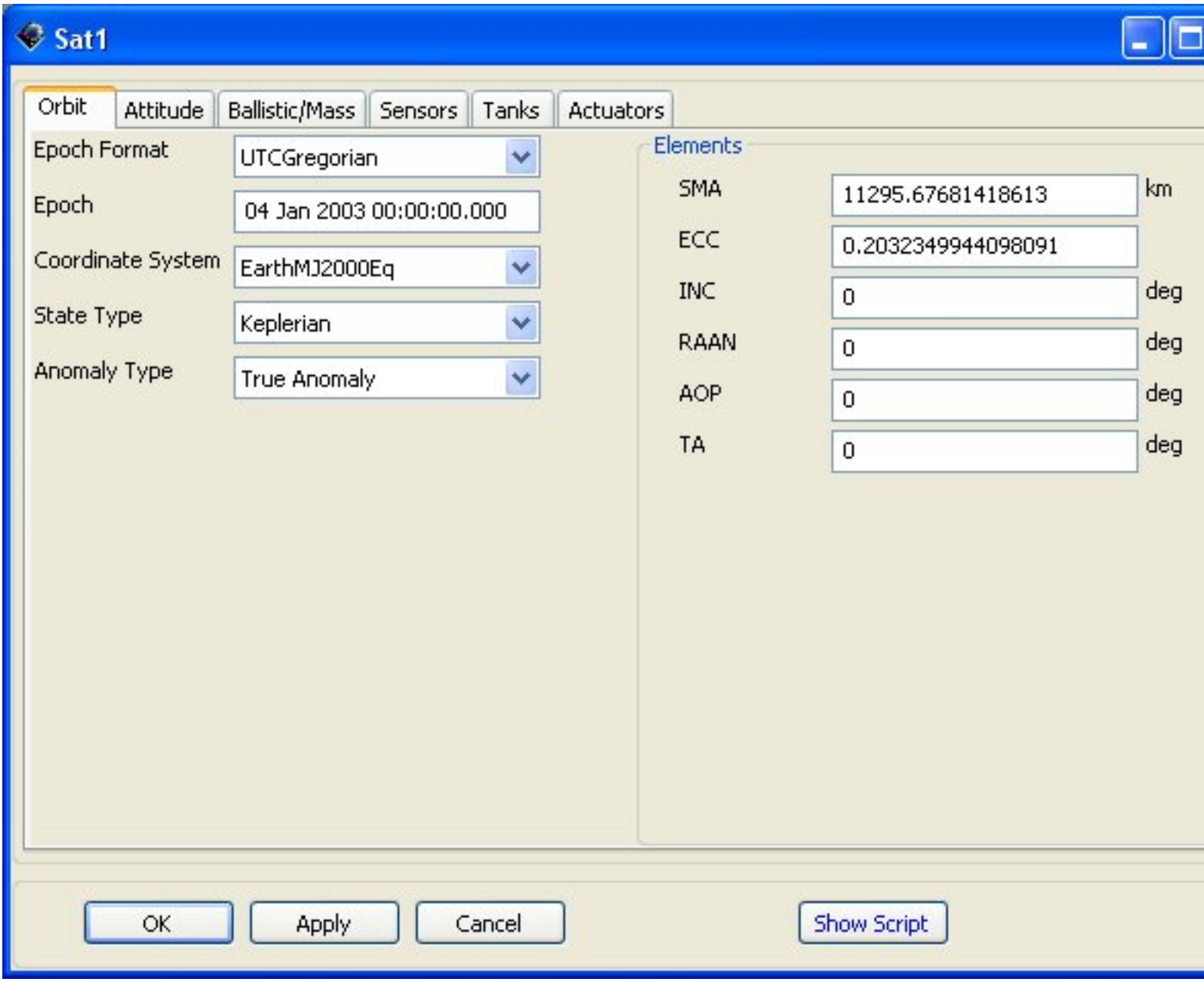


Figure: Sat1 Spacecraft Object

- Propagators
  - RKV89PointMass

    This propagator simply sets the Earth as a point mass and uses it as the Force Model. All other propagators below, except for one (RKV8912x12), use Earth as a point mass as well. Figure below shows this propagator properties' window; windows for other propagators will look similarly.

    - Rename DefaultProp to RKV89PointMass
    - Set the Initial Step Size to 30

- Set the Accuracy at 1e-012
- Set the Min Step Size to 30
- Set the Max Step Size to 30
- Set the Degree in the Gravity Group box to 0
- Set the Order in the Gravity Group box to 0

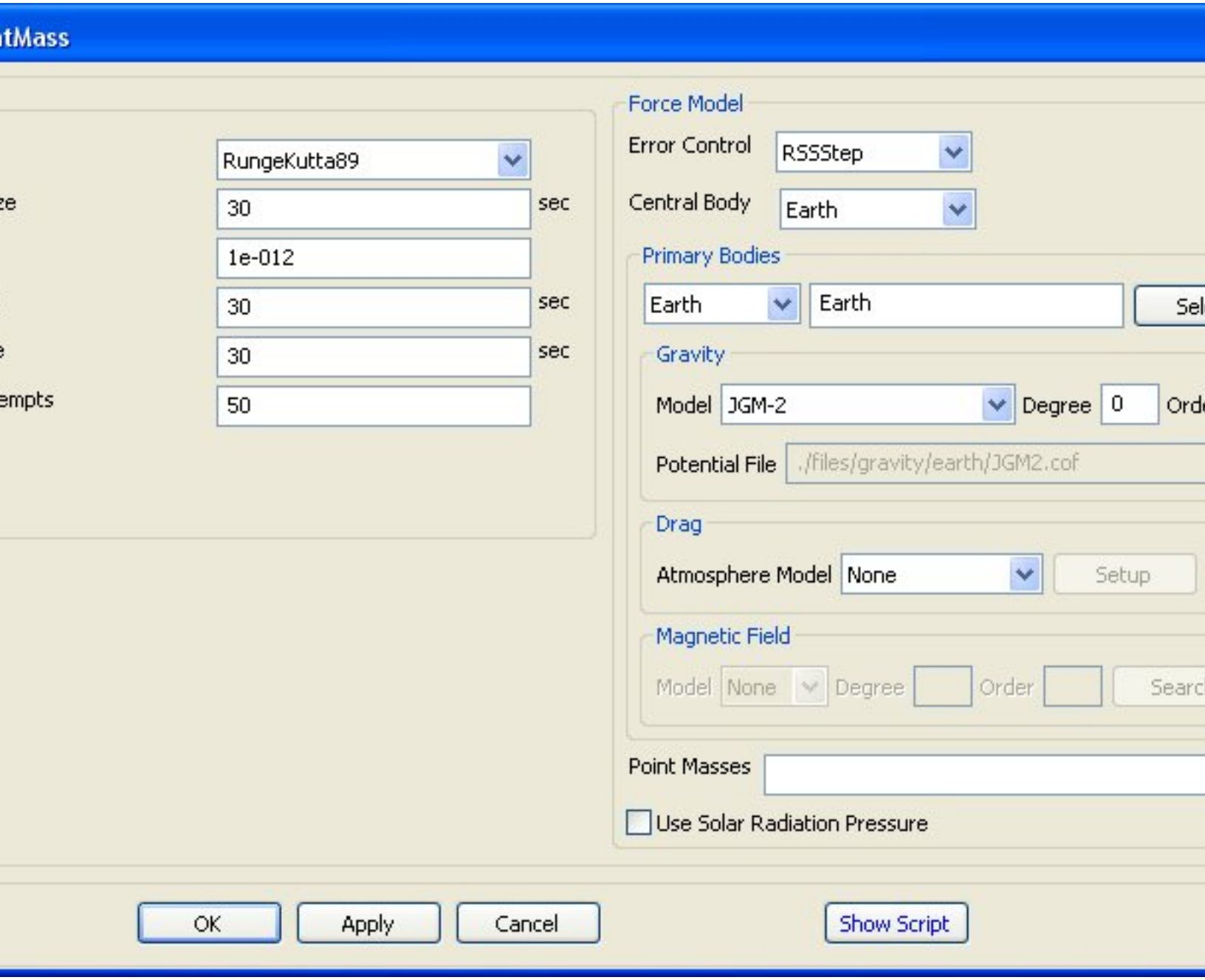


Figure: RKV89PointMass Propagator Object

- RKV89ThirdBodies

  This propagator adds all the planets except Mercury and Pluto as point masses to the Force Model.

  - Add a Propagator and rename it RKV89ThirdBodies
  - Set the Initial Step Size to 30
  - Set the Accuracy at 1e-012
  - Set the Min Step Size to 30
  - Set the Max Step Size to 30 as well
  - Set the Degree in the Gravity Group box to 0
  - Set the Order in the Gravity Group box to 0
  - Add Sun, Luna, and all the planets except Mercury and Pluto to the Point Masses field

- RKV8912x12

  Unlike other propagators, this one does not set Earth as a point mass in the Force Model; it uses instead a 12x12 JGM-2 Gravity Model.

  - Add a Propagator and rename it RKV8912x12
  - Set the Initial Step Size to 30
  - Set the Accuracy at 1e-012
  - Set the Min Step Size to 30
  - Set the Max Step Size to 30 as well
  - Raise the Degree in the Gravity Group box to 12
  - Raise the Order in the Gravity Group box to 12

- RKV89MSISE90

  The Force Model in this Propagator calculates drag on the spacecraft based on the MSISE90 Atmospheric Model.

  - Add a Propagator and rename it RKV89MSISE90
  - Set the Accuracy at 1e-012
  - Set the Min Step Size to 30
  - Set the Max Step Size to 30 as well
  - Lower the Degree in the Gravity Group box to 0
  - Lower the Order in the Gravity Group box down to 0
  - Set the Atmosphere Model under the Drag Group Box to MSISE90

- RKV89JR

  The Force Model in this Propagator calculates drag on the spacecraft based on the Jacchia-Roberts Atmosphere Model.

  - Add a Propagator and rename it RKV89JR
  - Set the Initial Step Size to 30
  - Set the Accuracy at 1e-012
  - Set the Min Step Size to 30
  - Set the Max Step Size to 30 as well
  - Lower the Degree in the Gravity Group box to 0
  - Lower the Order in the Gravity Group box down to 0
  - Set the Atmosphere Model under the Drag Group Box to JacchiaRoberts

- RKV89SRP

  This propagator is like the first one (RKV89PointMass), with the only difference that it accounts for the Solar Radiation Pressure.

  - Add a Propagator and rename it RKV89SRP
  - Set the Initial Step Size to 30
  - Set the Accuracy at 1e-012
  - Set the Min Step Size to 30
  - Set the Max Step Size to 30 as well
  - Lower the Degree in the Gravity Group box to 0
  - Lower the Order in the Gravity Group box down to 0
  - Check the Use Solar Radiation Pressure Check box

- Burns
  - Delete DefaultIB in the Resource tree
- Plots/Reports
  - OpenGLSat1 OpenGL Plot
    - Rename DefaultOpenGL to OpenGLSat1
    - Set the Axis under the View Up Definition Group box to X

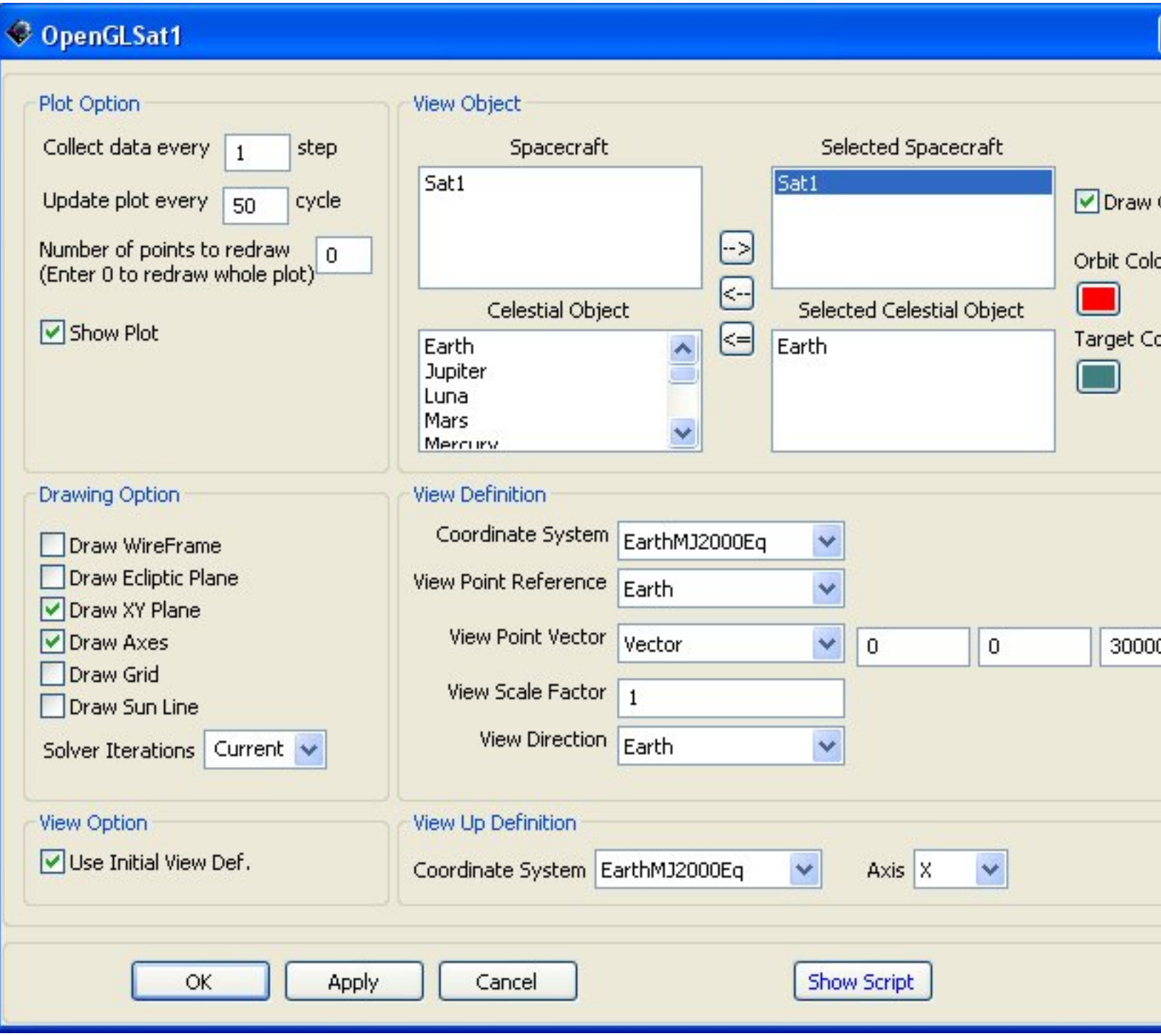


Figure: OpenGLSat1 OpenGL Plot Object

- ForceModelRpt Report
  - Create a Report Named ForceModelRpt
  - Remove all Parameters from the Parameter List

- Select the File as being ./output/SampleMissions/Ex_ForceModels.report

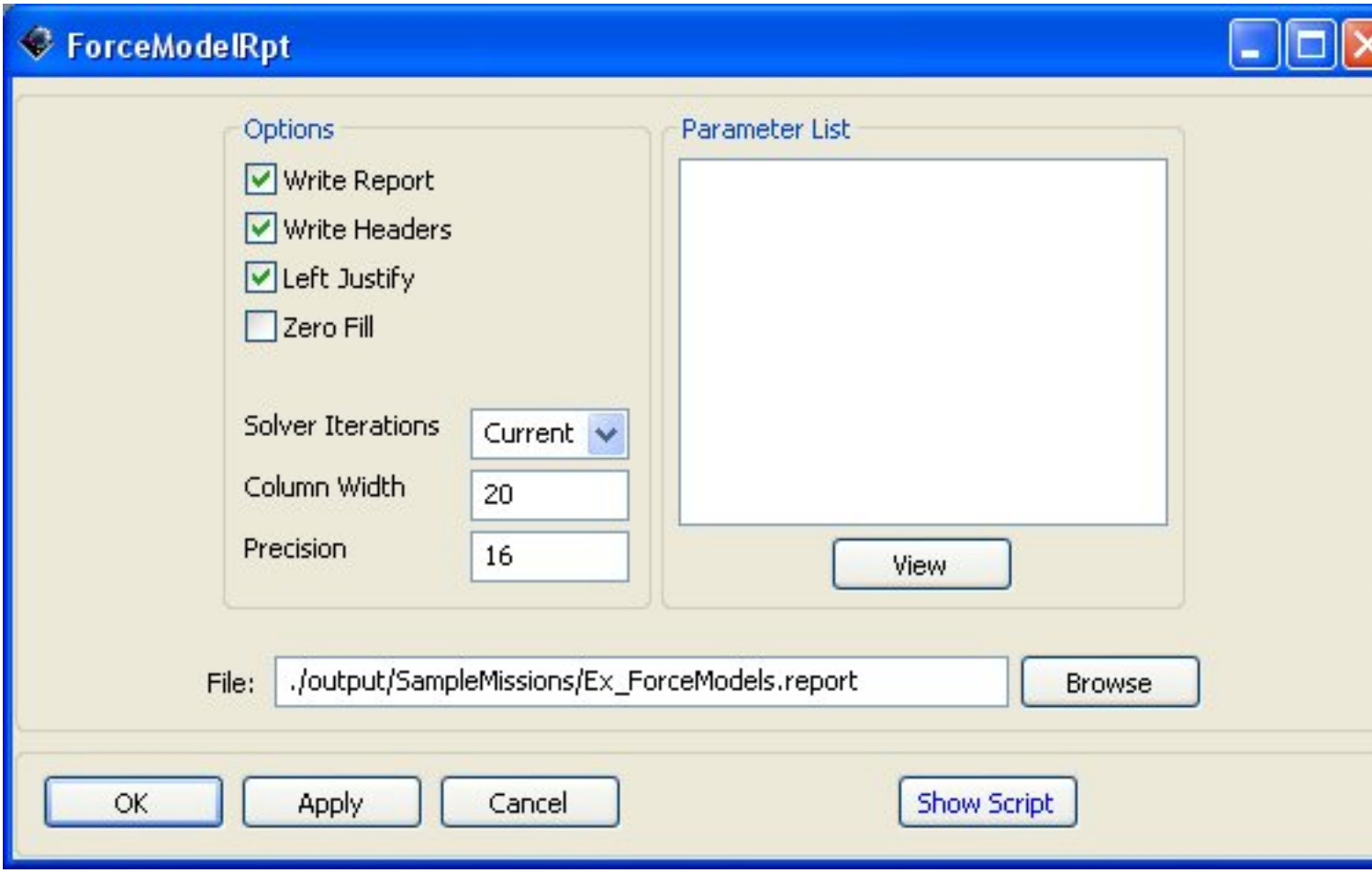


Figure: ForceModelRpt Report Object

Creating and Configuring the Mission Sequence

Image of Mission Tree

In addition to the default layout, this mission sequence includes an initial Report and Equation, followed by a series of alternating Propagates and Reports. Consequently, the mission runs several different Propagators, with their progress reported back to the user via the generated report file.

- Delete the initial Propagate
- Append the Mission Sequence Folder with a Report: Add Sat1.X, Y, Z, VX, VY, and VZ to the Parameter List
- Insert an Equation after Report 1. Set it to ForceModelRpt.WriteHeaders = Off

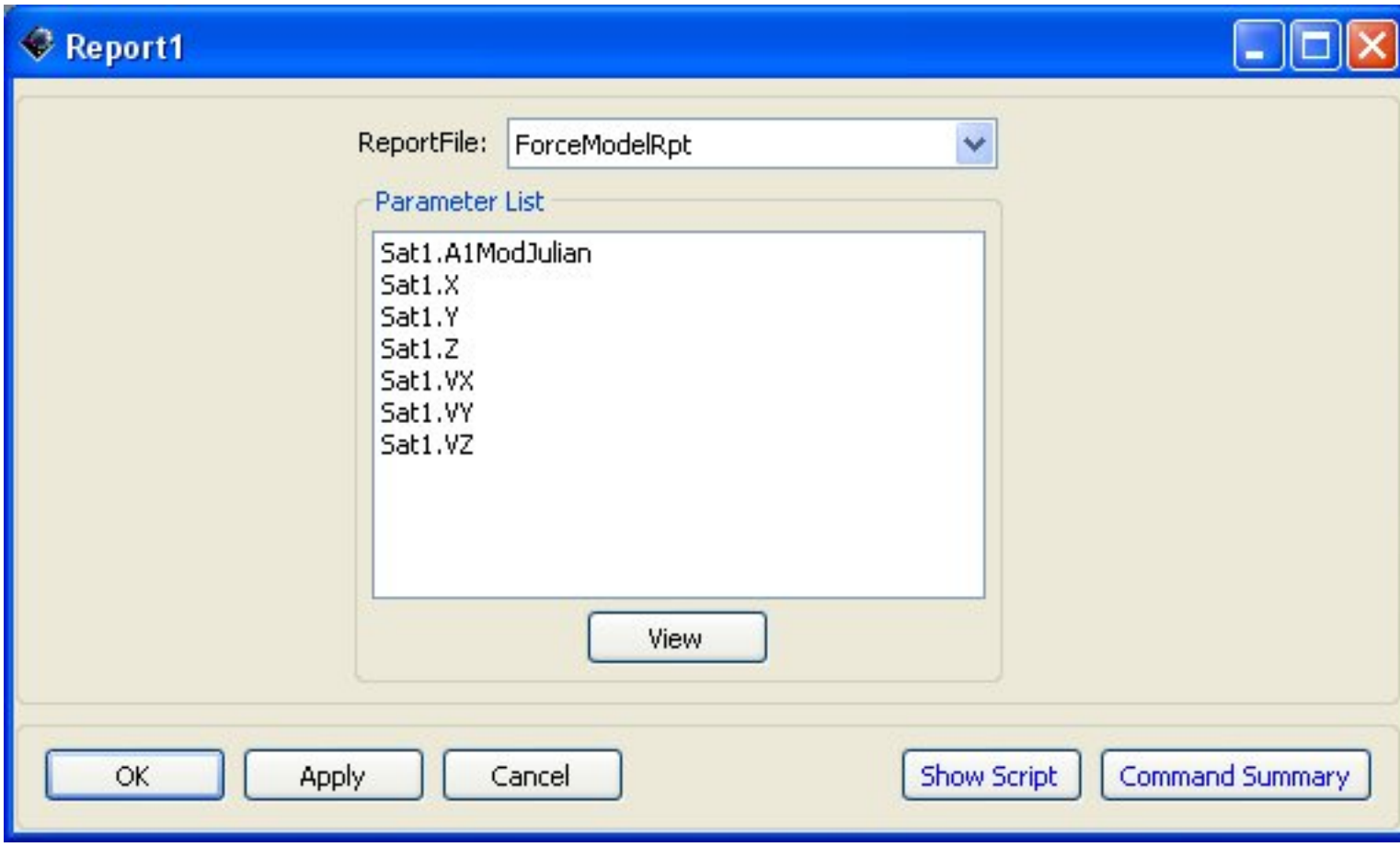


Figure: Report1 Report Command

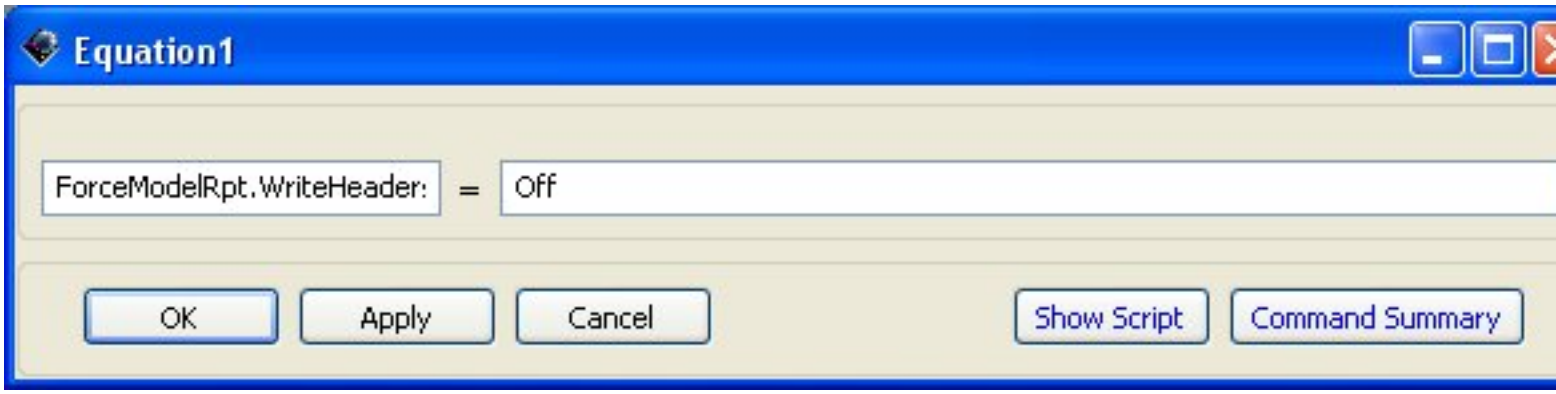


Figure: Equation1 Equation Command

- Insert a Propagate after Equation1
  - Set the Parameter Field to Sat1.ElapsedDays
  - Set the Condition Field to 0.1
- Insert a Report after Propagate 1. Set it exactly as Report1

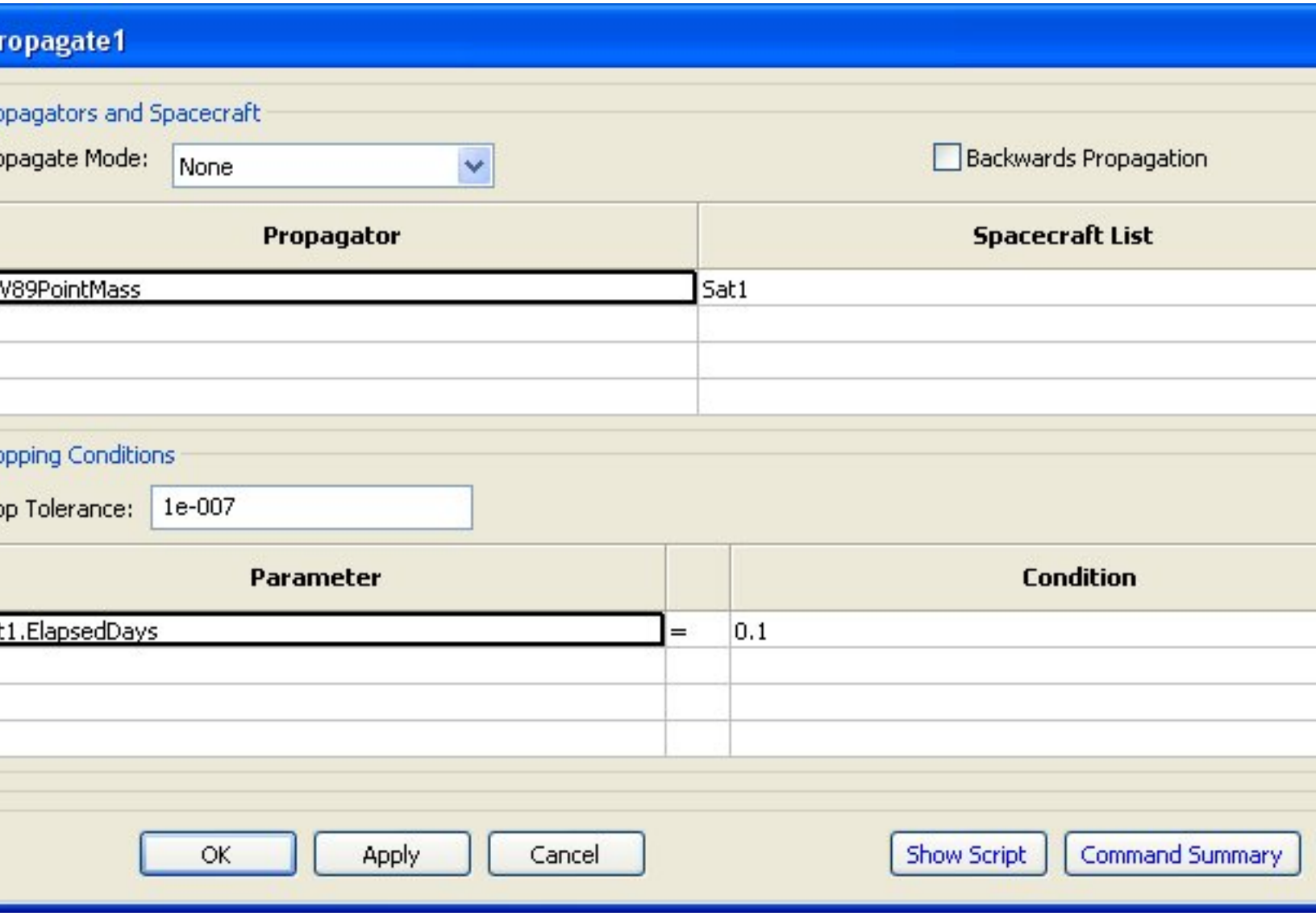


Figure: Propagate1 Propagate Command

- Insert a Propagate after Report2
  - Set the Propagator Field to RKV89ThirdBodies
  - Set the Stopping Conditions Group Box to match Propagate1
- Insert a Report after Propagate 2. Set it exactly as Report1
- Insert a Propagate after Report3
  - Set the Propagator Field to RKV8912x12
  - Set the Stopping Conditions Group Box to match Propagate1
- Insert a Report after Propagate 3. Set it exactly as Report1
- Insert a Propagate after Report4

    - Set the Propagator Field to RKV89MSISE90
    - Set the Stopping Conditions Group Box to match Propagate1
- Insert a Report after Propagate 4. Set it exactly as Report1
- Insert a Propagate after Report5
    - Set the Propagator Field to RKV89JR
    - Set the Stopping Conditions Group Box to match Propagate1
- Insert a Report after Propagate 5. Set it exactly as Report1
- Insert a Propagate after Report6
    - Set the Propagator Field to RKV89SRP
    - Set the Stopping Conditions Group Box to match Propagate1
- Insert a Report after Propagate 6: Set it exactly as Report1

The resulting view of Resource, Mission, and Output trees

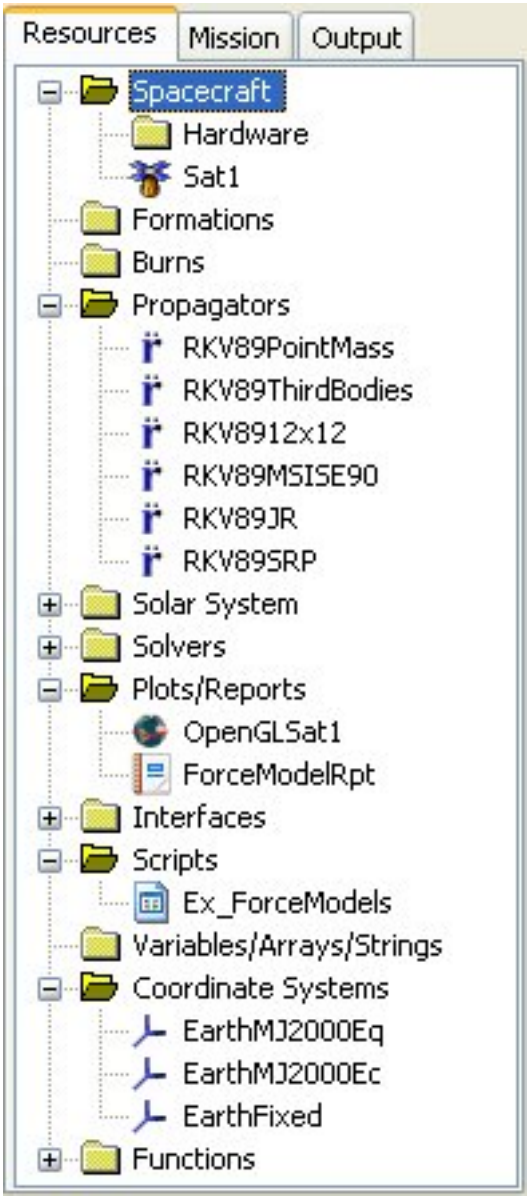


Figure: Force Models' Resource, Mission, and Output trees

Running the Mission

Image of Output Tree

When running the mission, you will see the spacecraft Sat1 orbit the earth. Every one tenth of a day there is a switch to the Propagator that governs the Sat1 dynamics at the related stage. The requested mission report (in the directory specified in the ForceModelRpt Report Object) shows the data that came out of the mission run. If these data seem difficult to understand, you may try to change the output parameters in the Report commands to Keplerian orbital elements, or to some other method of tracking. You will

notice slight variations in the data received from each report, due to the differences in governing forces in each Propagator.

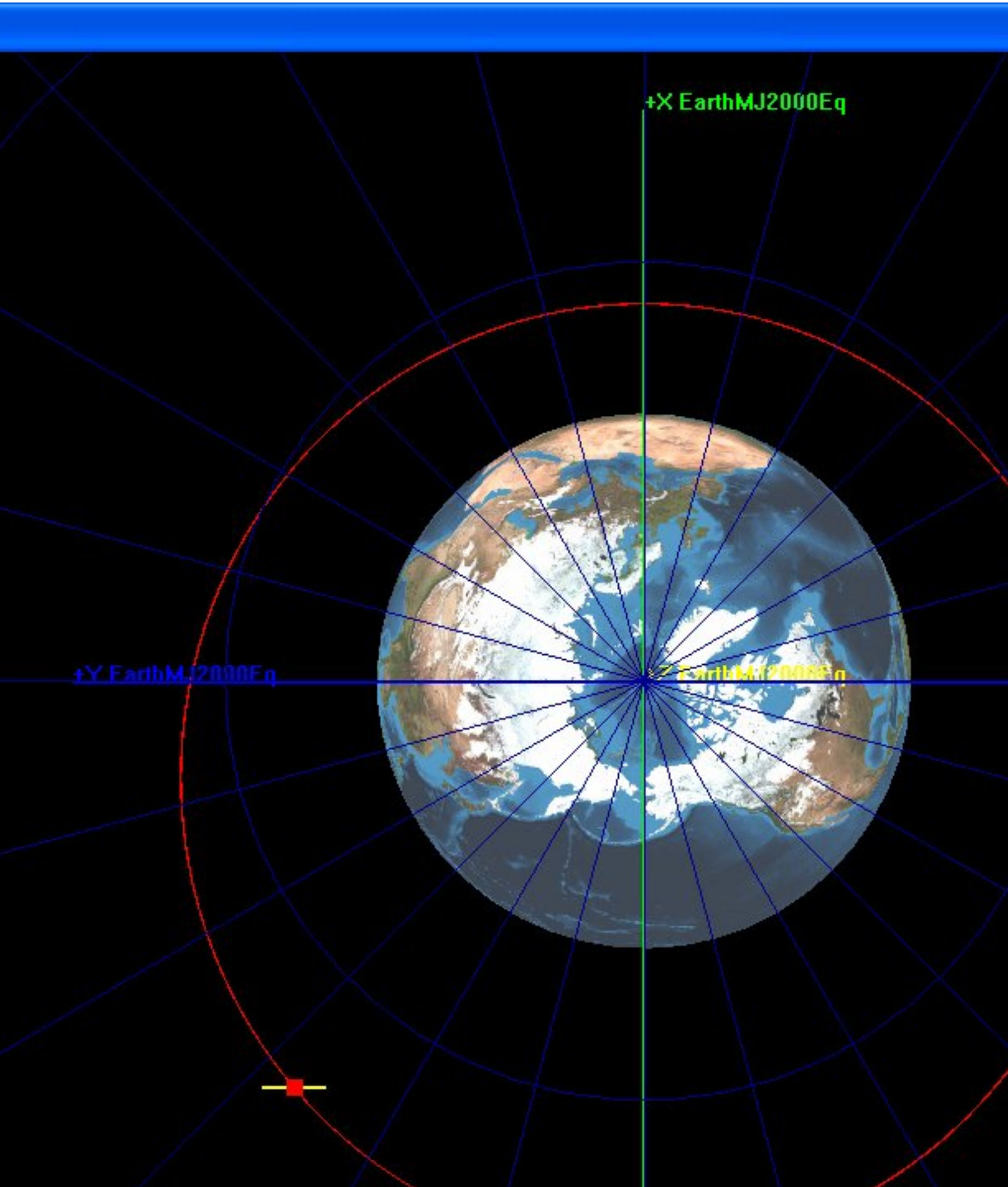

+X EarthMJ2000Eq
+Y EarthMJ2000Eq

Figure: Output

Chapter 15. Algebraic Optimization

This tutorial was started from the default settings in the July 2,2008 build of GMAT.

Objective and Overview

This tutorial finds the minimum value to satisfy a function. This tutorial is intended to show how GMAT's optimizer works. Uses of optimization in a true mission include minimizing the amount of fuel or minimum flight time required to achieve certain characteristics. Learning how to optimize a mission sequence also involves learning about optimizers, nonlinear constraints, and the minimize command.

You can download the script file and run it beforehand to see the final results of this tutorial: Ex_AlgebraicOptimization.script [scripts/Ex_AlgebraicOptimization.script]

Prerequisites

- Basic Understanding of how to create and propagate a spacecraft, as in Tutorial Creating and Propagating a Spacecraft

Mission Description

- Objective: The goal here is to find values of variables X1 and X2 that minimize a criterion function F, with a constraint that X1+X2=8, as follows:

  $F = (X1 - 2)^2 + (X2 - 2)^2$

  $G = X1 + X2$

  $G = 8$

- Find: Values X1 and X2

Creating and Configuring the Resource Tree

Objects Required

- Optimizer: SQPfmincon
- Plots/Reports: Report Data
- Variables: X1, X2, F, G

Creating and Modifying Objects

- Variables/Strings/Arrays
  - Variable X1 = 0
  - Variable X2 = 0
  - Variable F = 0

    - Variable G = 0
- Solvers
    - SQPfmincon Optimizer (Figure below shows the resulting SQPfmincon window)
        - Tol Fun = .0001
        - Tol Con = .0001
        - Tol X = .01
        - Max Fun Evals = 1000
        - Max Iter = 250
        - Diff Min Change = .000001
        - Diff Max Change = .01
        - Display : Iter
        - Diagnostics

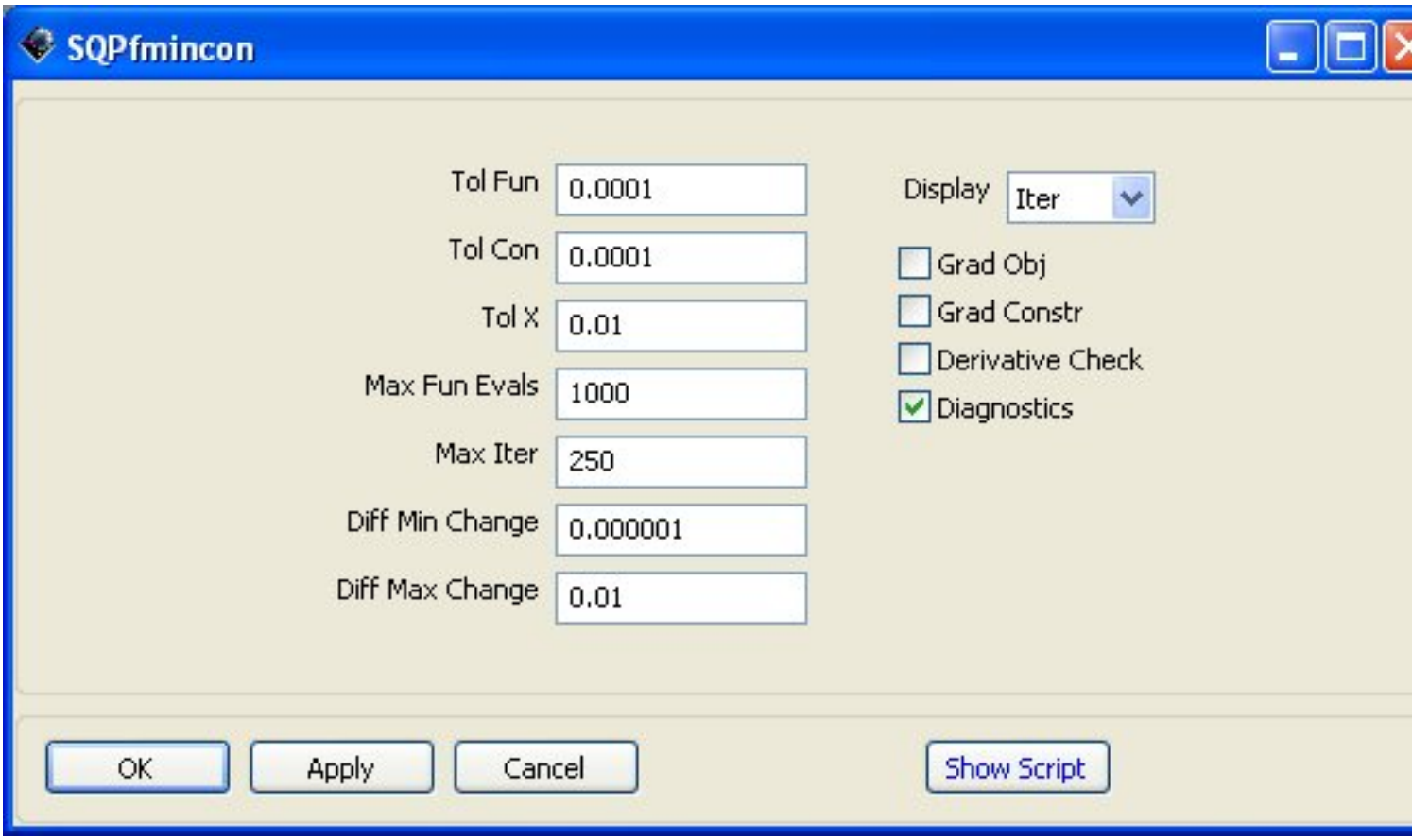


Figure: SQPfmincon Boundary Value Solver Object

- Plots/Reports

- Data ReportFile
  - Clear Parameter List
  - Saved at ./output/SampleMissions/Ex_AlgebraicOptimization.report

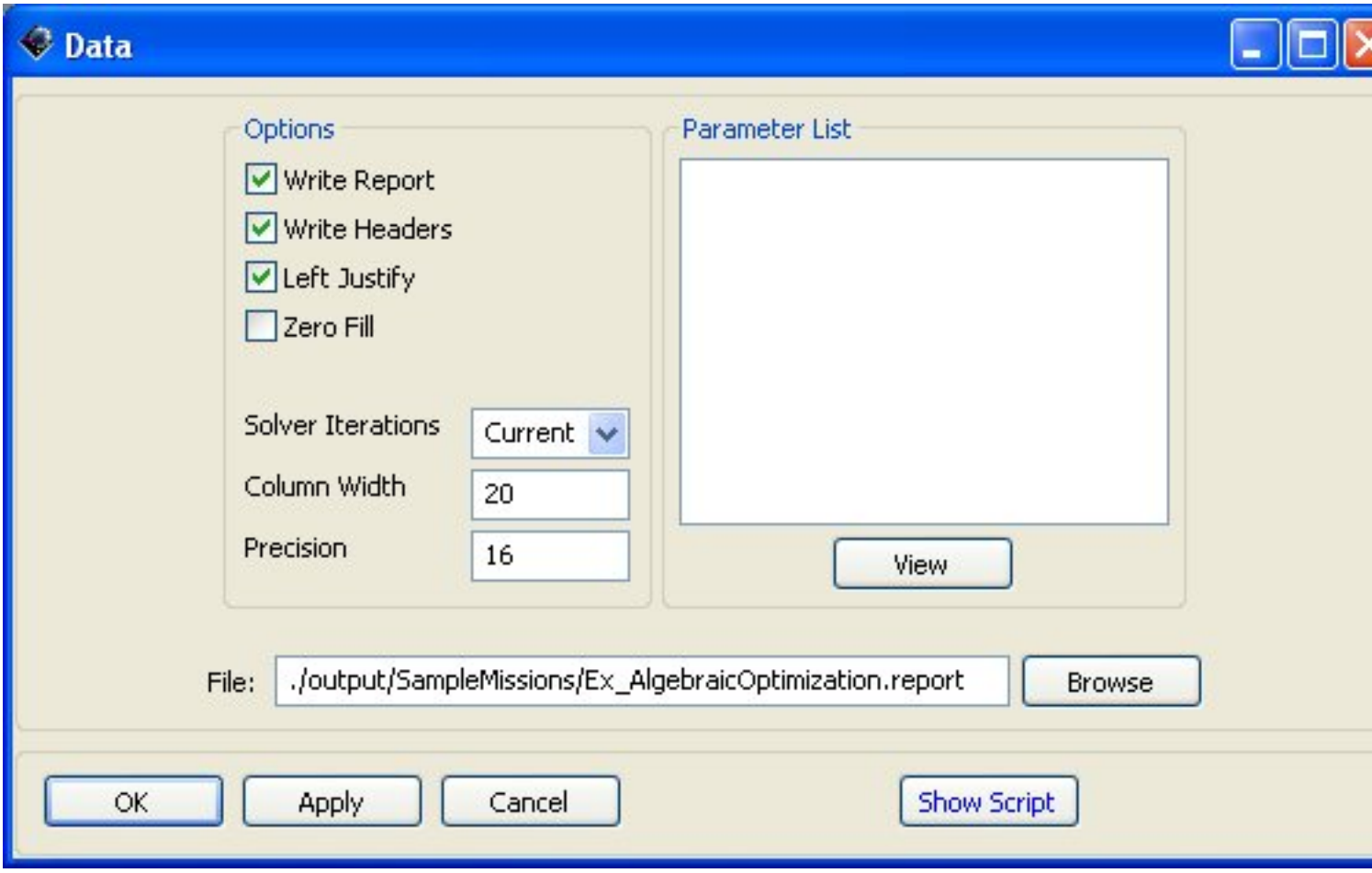


Figure: Data ReportFile System Object

Creating and Configuring the Mission Tree

Resource Mission and Output trees

As said above, values X1 and X2 are variables of function F that is to be optimized. The sum of these values form a constraint G, G=X1+X2=8. The final values of F, X1, X2 are sent to a report file.

Creating and Modifying Command Sequences

Set up the command sequence as in the Optimize sequence below. Optimization is done in Equation1, via the optimization criterion function F, using the Matlab routine SQPfmincon. For the constraint G, since a function cannot be a constraint in itself, the value G is used. This is done in Equation2.

- Optimize

- Vary X1 using SQPfmincon
  - Upper = 10, Lower = -10
- Vary X2 using SQPfmincon
  - Upper = 10, Lower = -10
- Equation1 F = $(X1 - 2)^2 + (X2 - 2)^2$
- Minimize the value of F using SQPfmincon
- Equation2 G = X2 + X1
- Nonlinear Constraint G = 8 using SQPfmincon

- Report to file Data the values of variables F, G, X1, X2

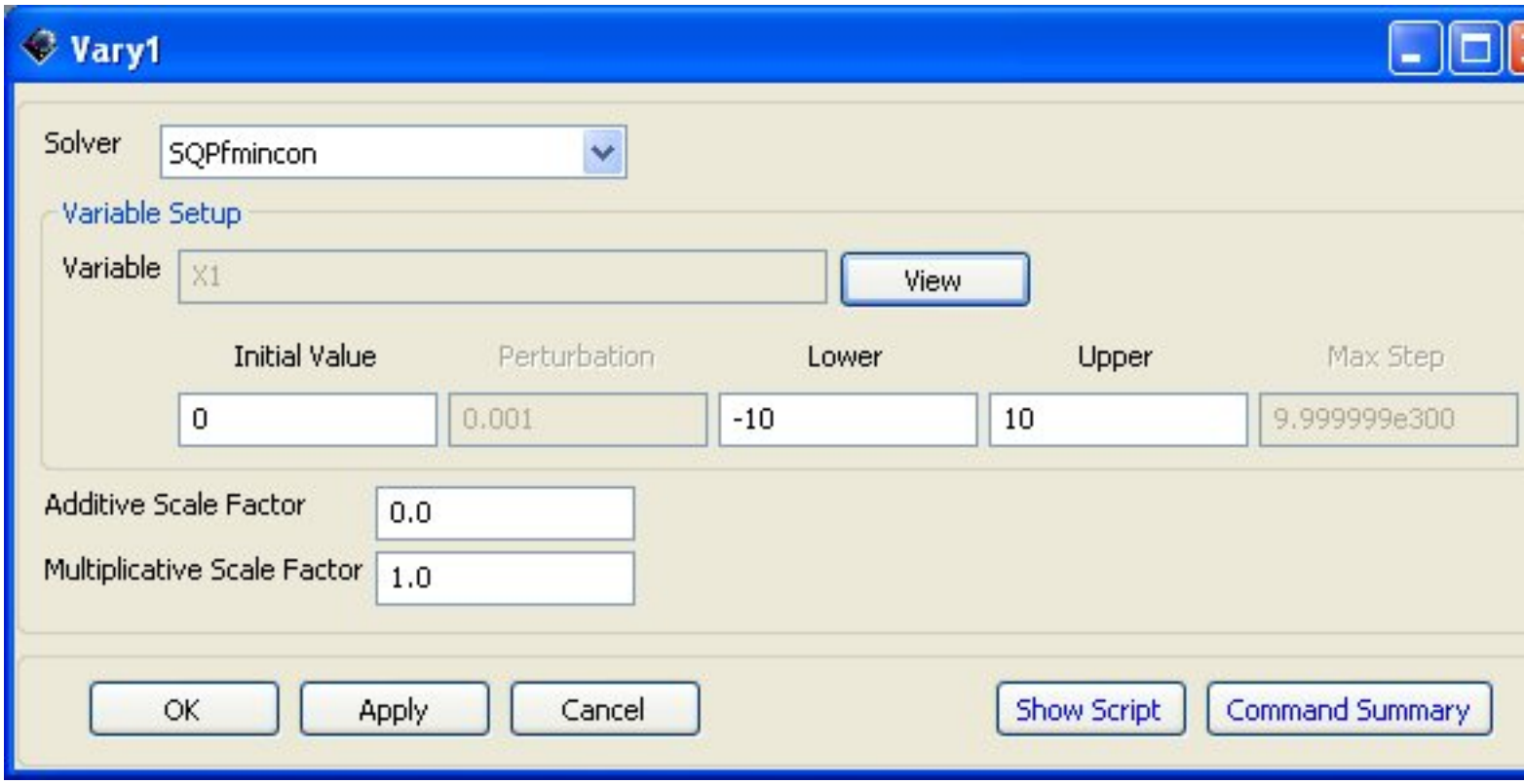


Figure: Vary X1 Command

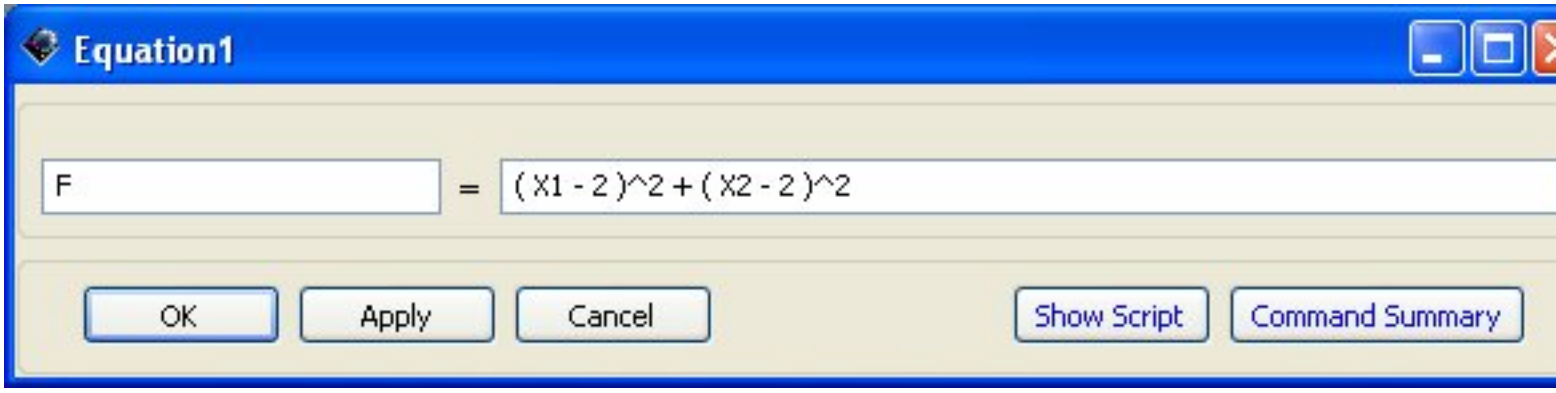


Figure: Equation1 Command

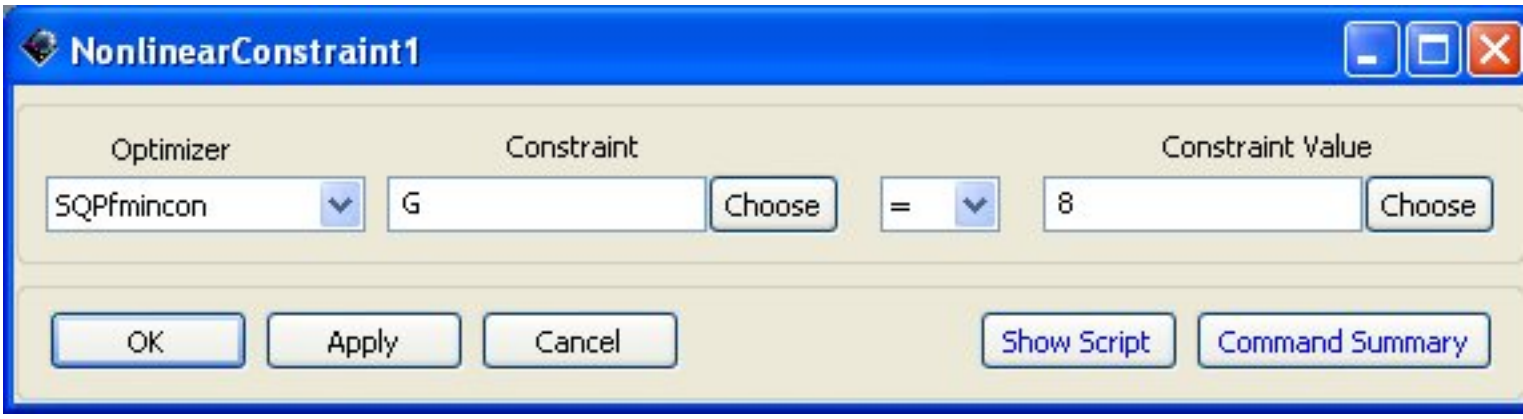


Figure: Nonlinear Constraint Command

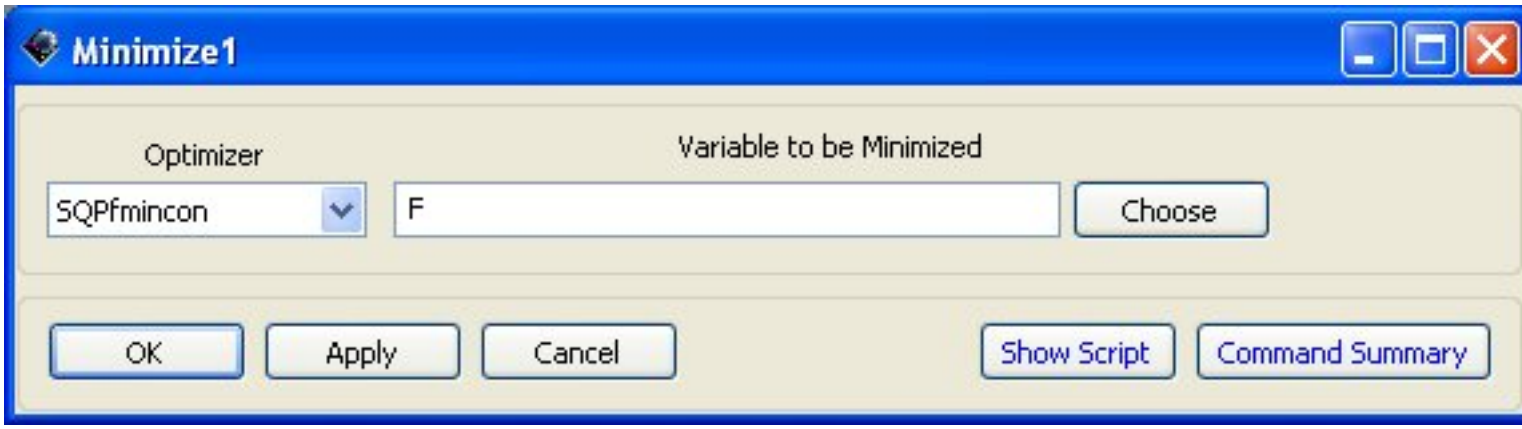


Figure: Minimize F Command

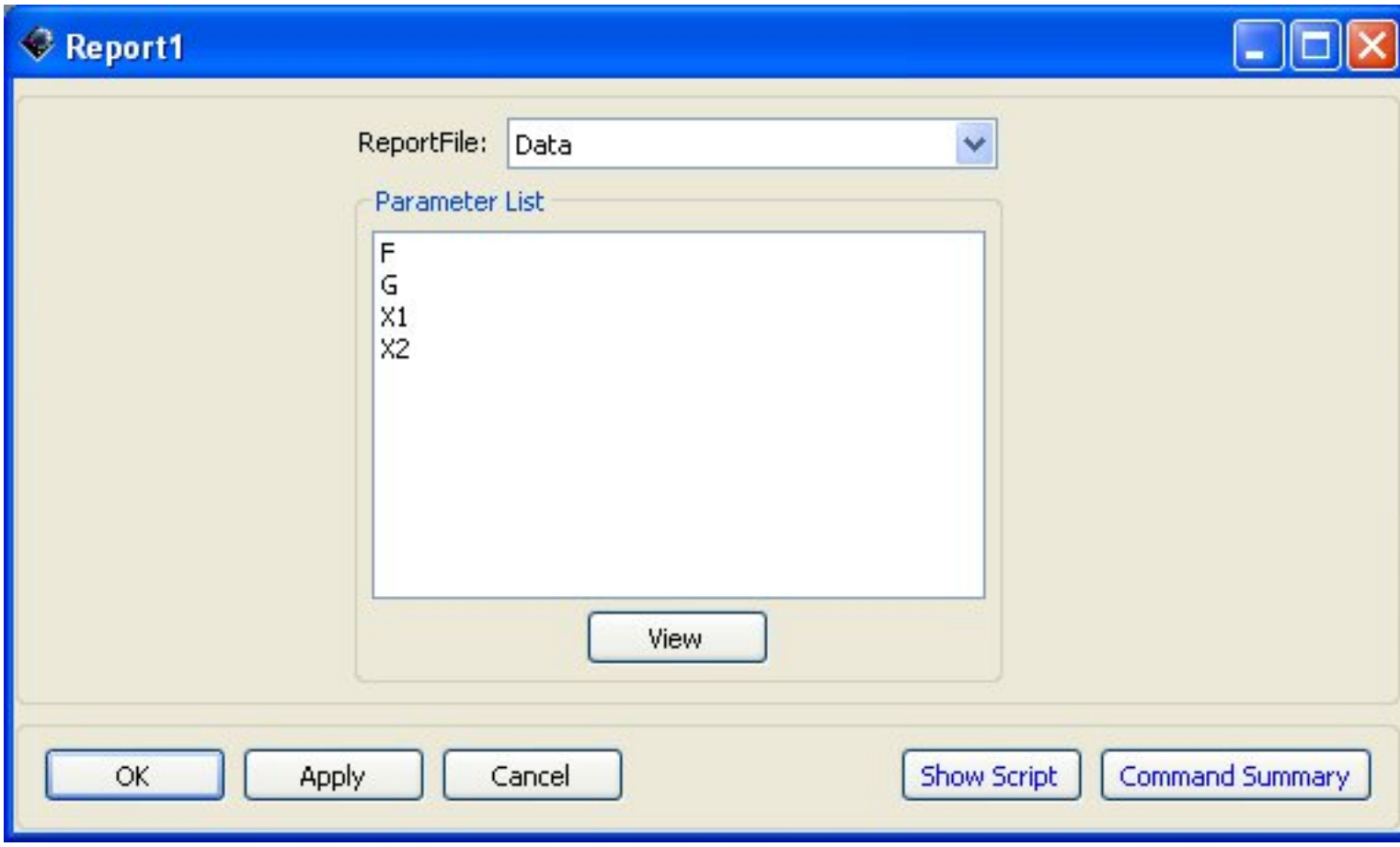


Figure: Report Command

The resulting view of Resource, Mission, and Output trees

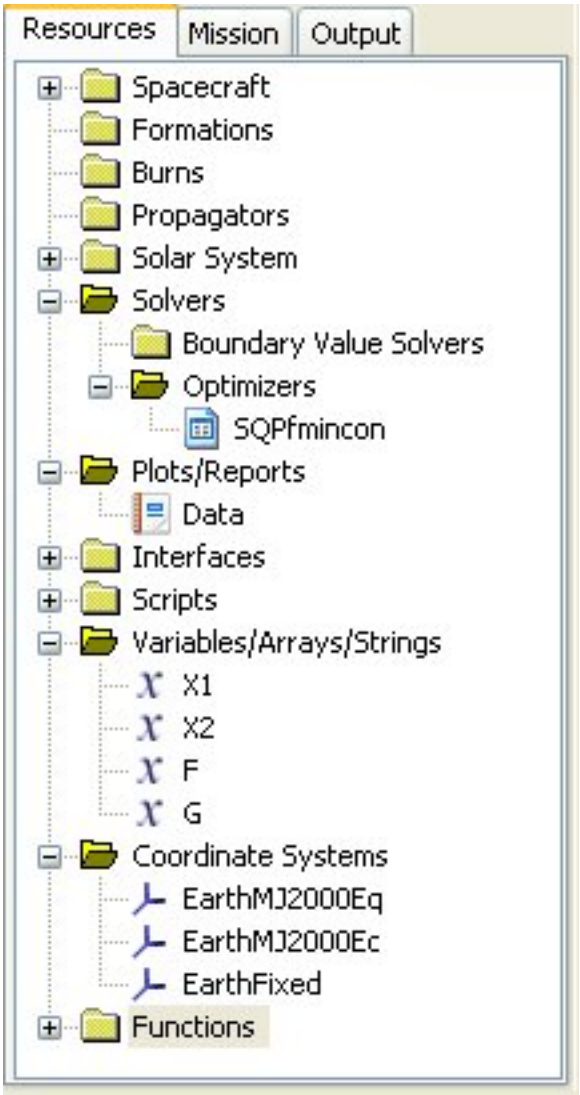
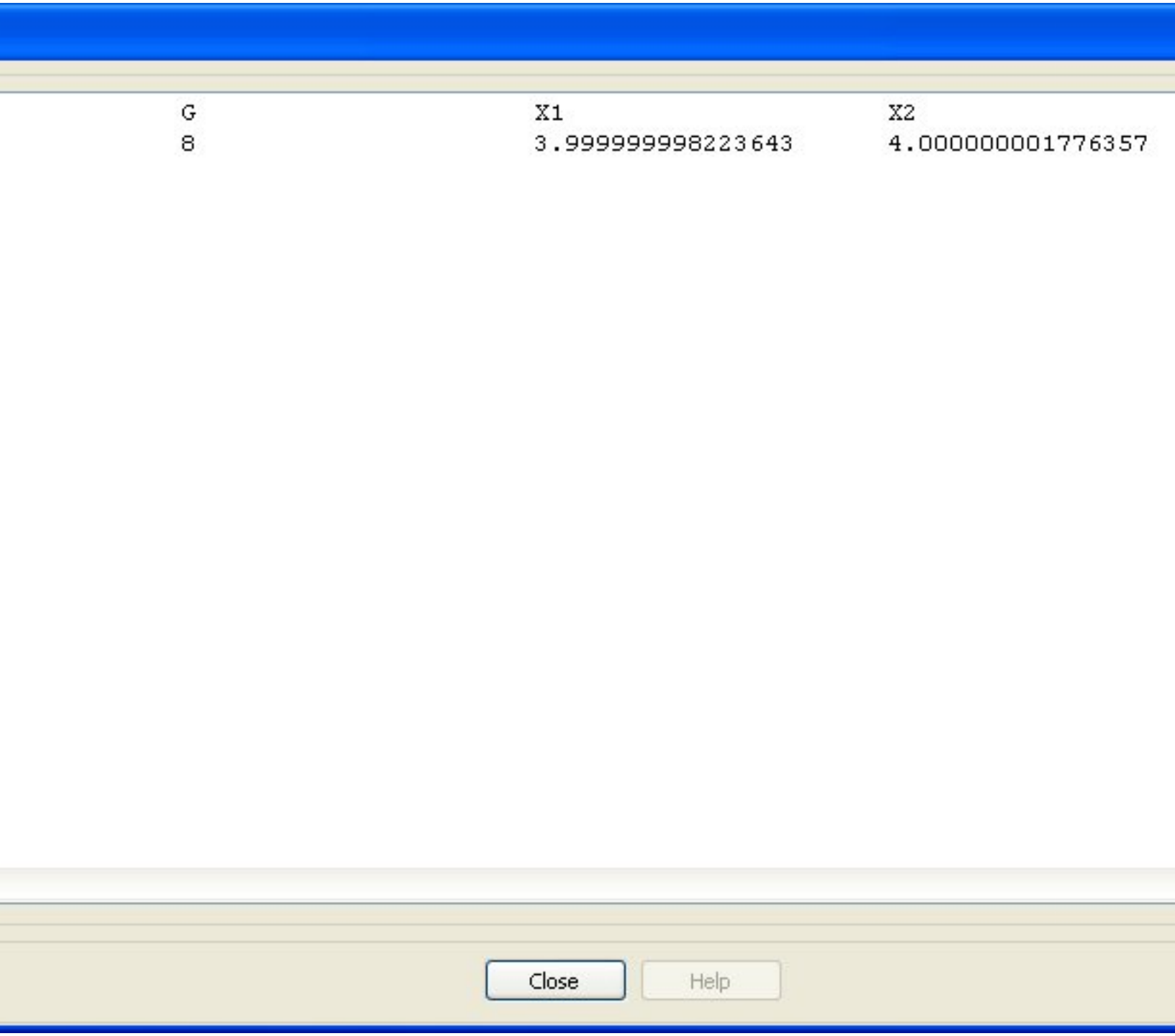


Figure: Algebraic Optimization Resource, Mission, and Output trees

Running the Mission

Resource Mission and Output Trees

There is no plot for this mission sequence. Its results - values X1, X2, and the minimum value of the optimized function F - can be found by calling ReportFile Data.



G	X1	X2
8	3.999999998223643	4.000000001776357

Figure: Data ReportFile

Chapter 16. Mars B-Plane Targeting

This tutorial was started from the default settings in the June 9, 2008 build of GMAT.

Objective and Overview

The goal of this mission is to show how to target a transfer trajectory from an Earth parking orbit to a Mars orbit via B-Plane targeting. Given a specified inclination for the final orbit, we solve for the velocity of the spacecraft in certain directions in order to determine a transfer trajectory.

You can download the script file and run it beforehand to see the final results of this tutorial: MarsBPlaneTutorial.script [scripts/MarsBPlaneTutorial.script]

Prerequisites

- Basic understanding of B-Planes and their usage in targeting a position.
- Tutorials:
  - Creating and Propagating a Spacecraft
  - Designing a Hohmann Transfer

Mission Description

- Objective: Target a Martian transfer trajectory using B-Plane Targeting, to achieve a final inclination of 24.677°.
- Assume:
  - Parking orbit SMA = 10000 km
  - Parking orbit ECC = 0.1
  - INC = 25.0°
  - RAAN = 332.5°
  - AOP = 90.0°
  - TA = 360°
- Find:
  - INC
  - BdotT
  - BdotR

Creating and Configuring the Resource Tree

For this tutorial, four impulsive burns are used to maneuver a single spacecraft, and three propagators are used for different portions of the spacecraft's propagation. The propagators use different point masses to

account for the dominant gravity fields during the spacecraft's trajectory. Two OpenGL plots are used to track the path at the starting point and ending point of the mission. Finally, four coordinate systems are used to define the spacecraft's position - based, accordingly, on Earth, Mars, the ecliptic, and the equatorial plane of the Earth.

Objects Required

- Spacecraft: MarsSupply
- Impulsive Burns: TOI, MarsMCC, MarsBPlane, and MarsOI
- Propagators: EarthProp, SunProp, and MarsProp
- Differential Corrector: MarsTOIDC
- Plots/Reports: EarthView and MarsView and a data report
- Coordinate Systems: MarsMJ2000Eq, SunMJ2kEc, SunMJ2kEq, and EarthSunRot

Creating and Modifying Objects

- Spacecraft
  - Add a spacecraft to the resource tree and rename it MarsSupply
  - Set the epoch format to TAIModJulian
  - Use an Epoch of 21160
  - Set the State Type to Keplerian
  - See Assumptions to modify spacecraft element properties

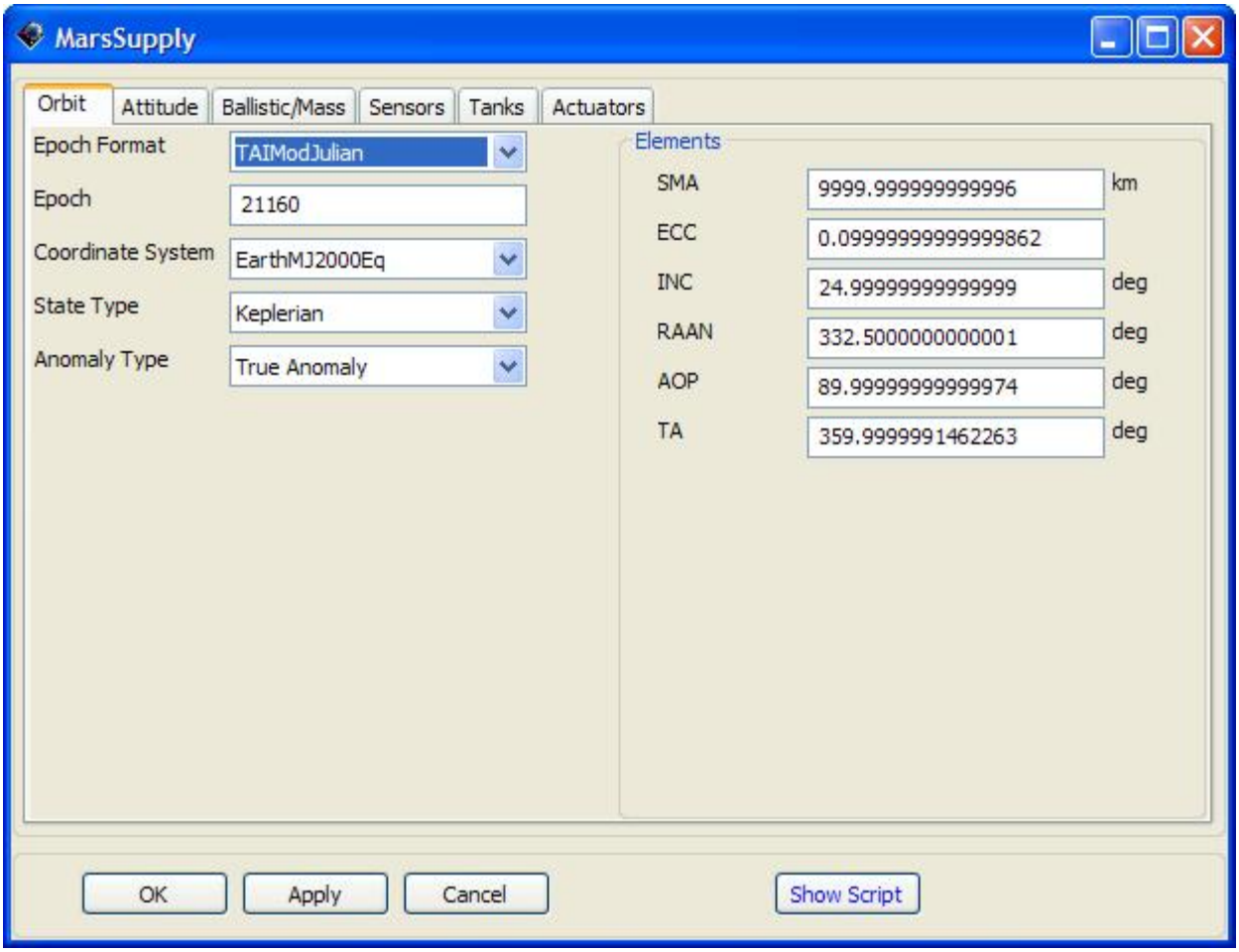


Figure: MarsSupply Spacecraft Object

- Propagators
  - Remove all primary bodies and gravity models from all propagators
  - EarthProp
    - Use an Integrator max step size of 86400 sec
    - Include only a Geocentric point mass

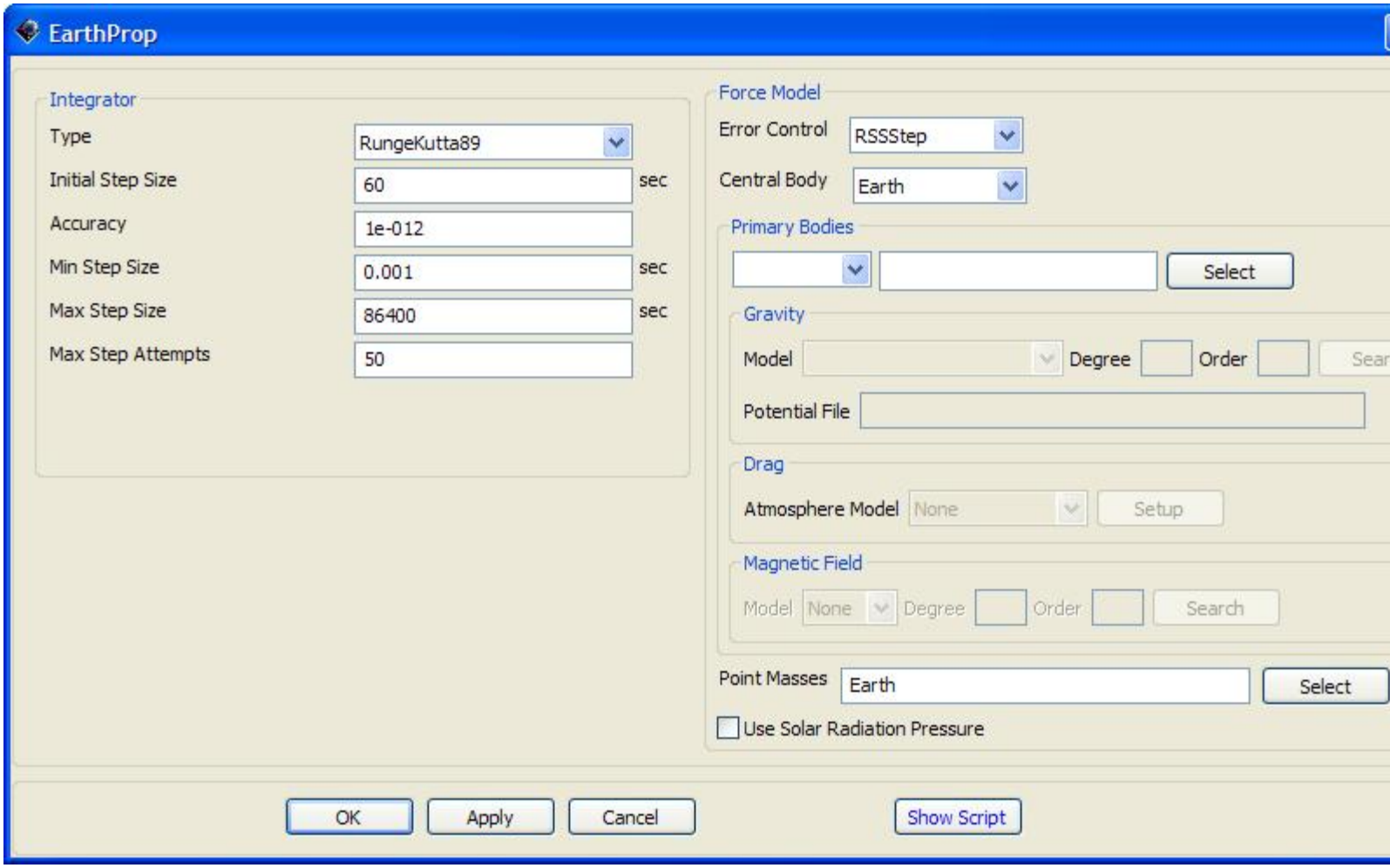


Figure: EarthProp Propagator Object

- SunProp
  - Set the Integrator max step size to 160000 sec
  - Set the Integrator max step attempts to 500
  - Include heliocentric, geocentric, and lunar point masses
- MarsProp includes only a martian point mass
  - Set the Integrator max step size to 86400 sec
  - Set the Integrator max step attempts to 100

- Burns
  - Change the TOI vector V to 2.85 km/s

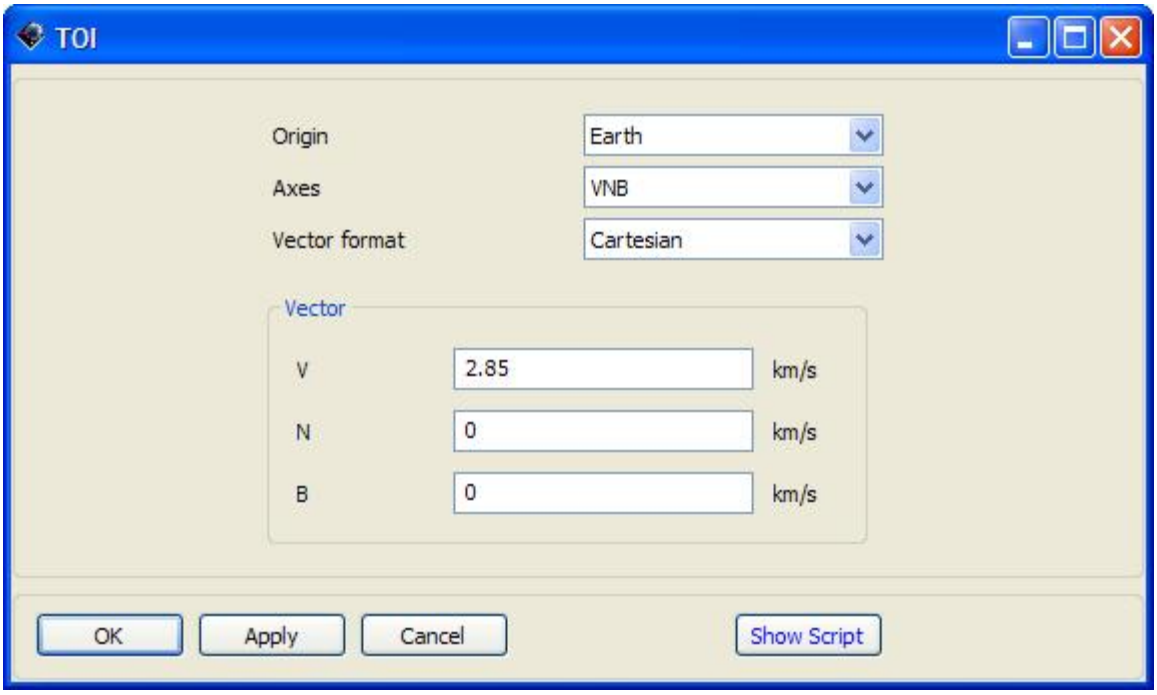


Figure: TOI Impulsive Burn Object

- Change the MarsMCC vector N to - 0.8593 km/s with a heliocentric origin
- Change the MarsBPlane vector V to - 0.3309 km/s and B to 0.6917 km/s with a heliocentric origin
- Change the MarsOI vector V to - 4 km/s with a martian origin

- Coordinate Systems
  - EarthSunRot
    - Set the Type of Axes to ObjectReferenced
    - Make Sun the Primary body and Earth the Secondary body

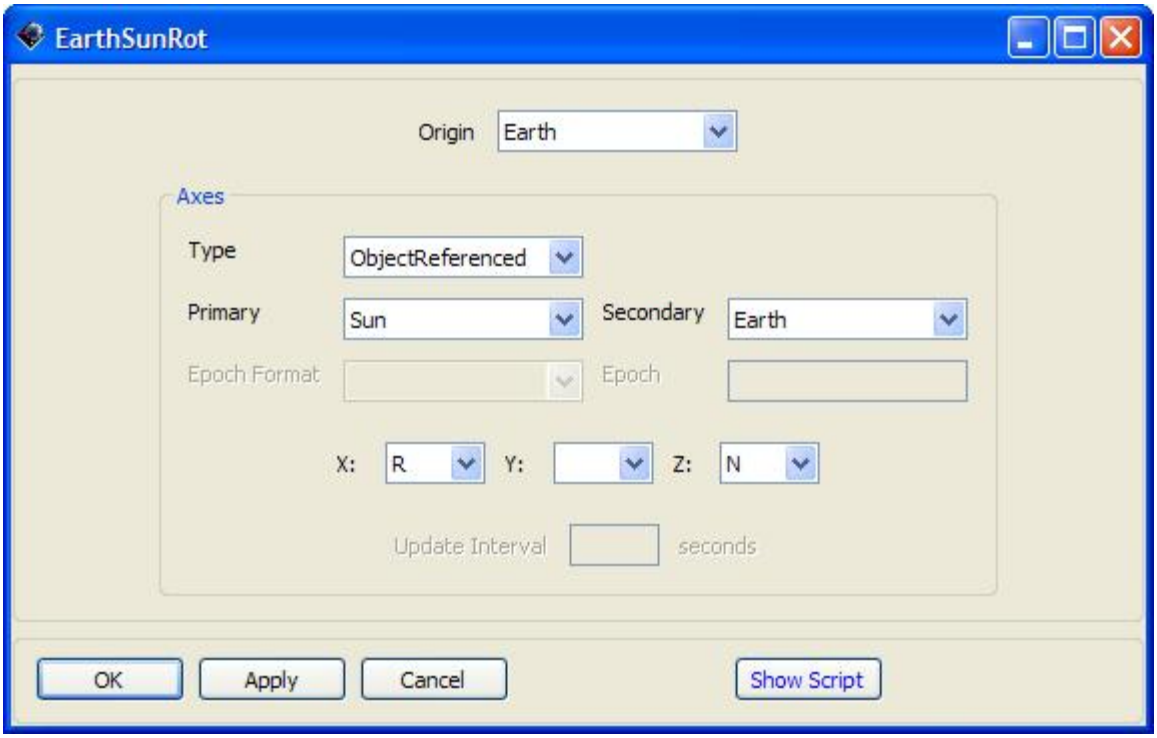


Figure: EarthSunCoord Coordinate System Object

- SunMJ2kEc
  - Apply a heliocentric origin to the system
  - Use MJ2000Ec as the Type of Axes
- SunMJ2kEq
  - Apply a heliocentric origin to the system
  - Use MJ2000Eq as the Type of Axes
- MarsMJ2000Eq
  - Apply a martian origin to the system
  - Use MJ2000Eq as the Type of Axes

- Solvers
  - Add a differential corrector to the Boundary Value Solvers folder in the Solvers folder under the resource tree and rename it MarsTOIDC
  - Use the Targeter Text File targeter_MarsTOIDC.data

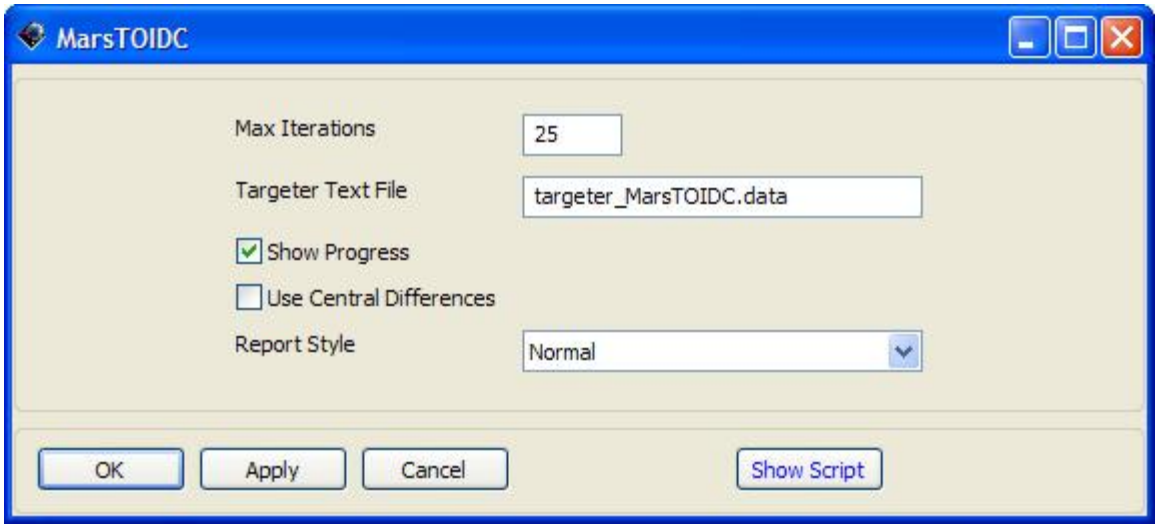


Figure: MarsTOIDC Differential Corrector Object

- Plots/Reports
  - Add a Report file to the Plots/Reports folder under the resource tree

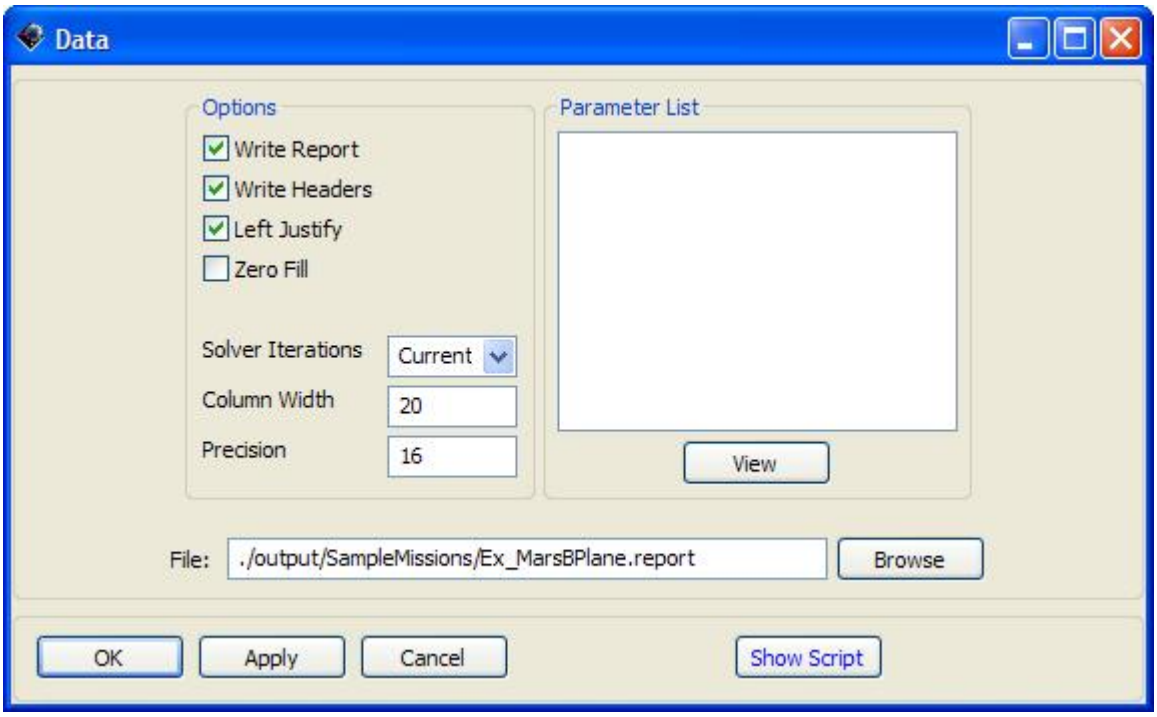


Figure: Data Data Report Object

  - EarthView
    - Add the Sun to the Selected Celestial Objects field
    - Check the Draw Grid and Draw Sun Line boxes in the Drawing Option group box

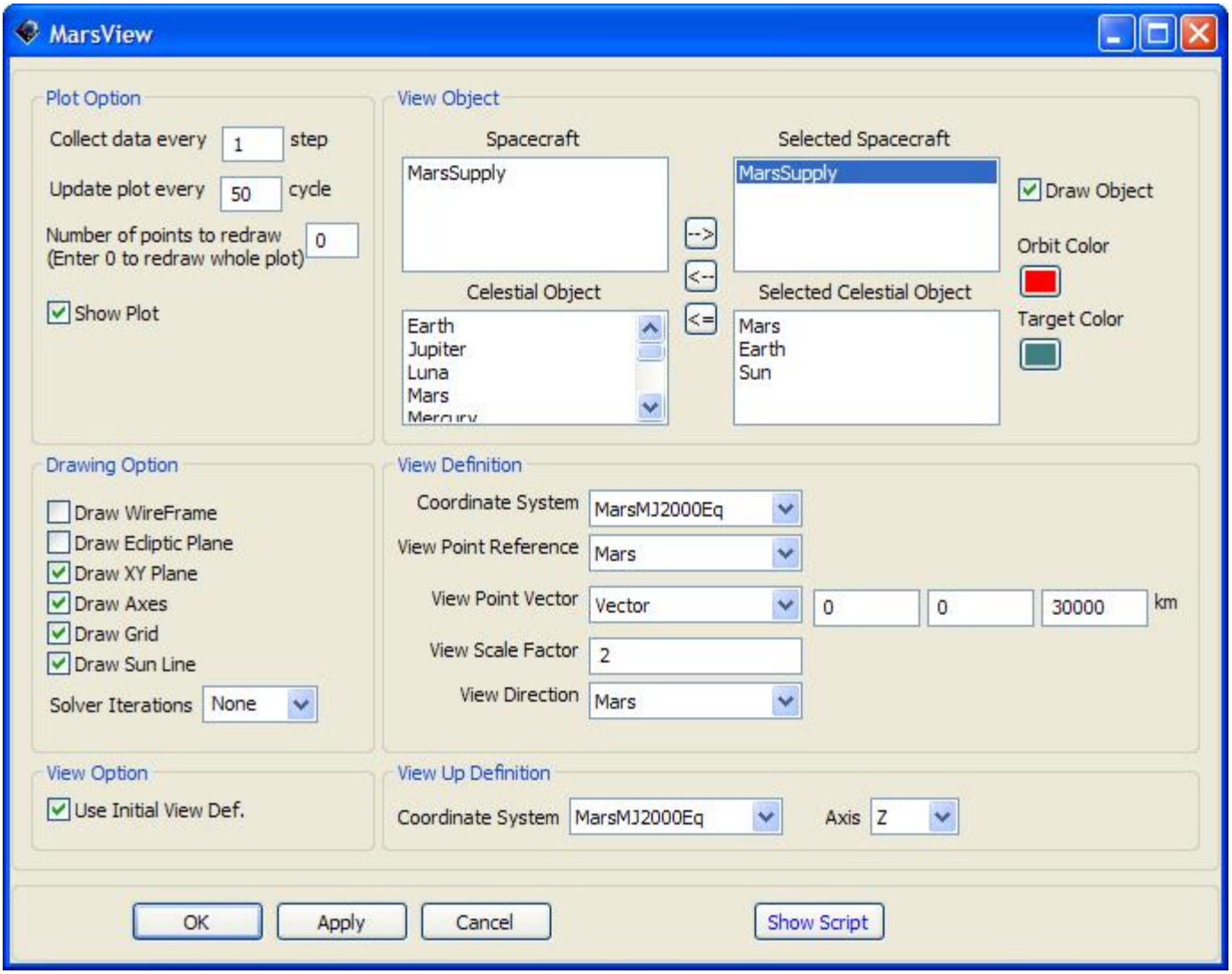


Figure: MarsView OpenGL Plot Object

- MarsView
  - Add the Sun and Mars to the Selected Celestial Objects field
  - Use MarsMJ2000Eq coordinate system in the View Definition and View Up Definition group boxes
  - Use a View Scale Factor of 2
  - Check the Draw Grid and Draw Sun Line boxes in the Drawing Option group box

Creating and Configuring the Mission Sequence

Resource Mission and Output Trees

This mission begins at the spacecraft orbit's Earth periapsis point, then moves to a Mars transfer trajectory, and ends in an elliptical orbit about Mars. The transfer trajectory calls two targeting sequences, one to obtain an inclination, and the other to obtain a B-Plane position. In the first targeting sequence, the inclination is achieved, and is then used when the second sequence targets a distinct B-Plane parameter. After the second targeting sequence is accomplished, the spacecraft is slowed by a final maneuver that puts it into the elliptical orbit and continues to propagate for another half day. During each Propagate and Maneuver command before and after the targeting sequences, reports are added for tracking and analysis purposes, to monitor the spacecraft's Cartesian coordinates and velocity vectors.

- Initial Propagate and Maneuver Sequence
  - For Equation 1, adding a report automatically turns the headers on, so be sure the setting is off to avoid including unnecessary text in the report.
  - Propagate 1 simply propagates the spacecraft to Earth's periapsis.
  - Maneuver 1 applies a #V in the V direction, to propel the spacecraft for 160 days through Propagate commands 2 and 3. When the propagators switch, the dominant point mass gravity field changes from Earth to Sun.
- Targeting Sequences
  - Target 1 includes Vary, Maneuver, and Achieve commands, to prompt GMAT into achieving the desired inclination. The maneuver applies the MarsMCC impulsive burn to achieve a #V in the N direction, so the spacecraft can reach 24.677° and begin its final maneuver.
  - Target 2 includes two Vary, two Propagate, two Achieve, and one Maneuver commands, to target a BdotT of 10,000 km and BdotR of – 10,000 km about Mars' periapsis. The maneuver applies the MarsBPlane impulsive burn, to achieve a #V in both the V and B velocity directions.
- Final Propagate and Maneuver Sequence
  - Maneuver 4 applies a #V in the V direction, to slow the spacecraft to the velocity that is necessary to obtain an elliptical orbit about Mars.
  - The final command, Propagate 6, propagates the spacecraft for an additional half day, so the plot can show the entire elliptical trajectory around Mars.

Creating and Modifying Sequences

Initial Propagate and Maneuver Sequence

- Report 1:
  - Add the Cartesian coordinates X, Y, and Z, and all velocity vectors, VX, VY, and VZ, to the parameters list, to report them for the MarsSupply spacecraft
  - Do not remove the A1ModJulian parameter
- Equation 1: set Data.WriteHeaders = Off
- Propagate 1:
  - Use the EarthProp propagator
  - Set the stopping condition parameter to be Earth's periapsis

- Report 2: refer to Report 1 for parameters list
- Maneuver 1: utilize the TOI burn
- Propagate 2:
  - Use the EarthProp propagator
  - Set the stopping condition to 25 elapsed days
- Report 4: refer to Report 1 for parameters list
- Propagate 3:
  - Use the SunProp propagator
  - Set the stopping condition to 135 elapsed days
- Report 5: refer to Report 1 for parameters list

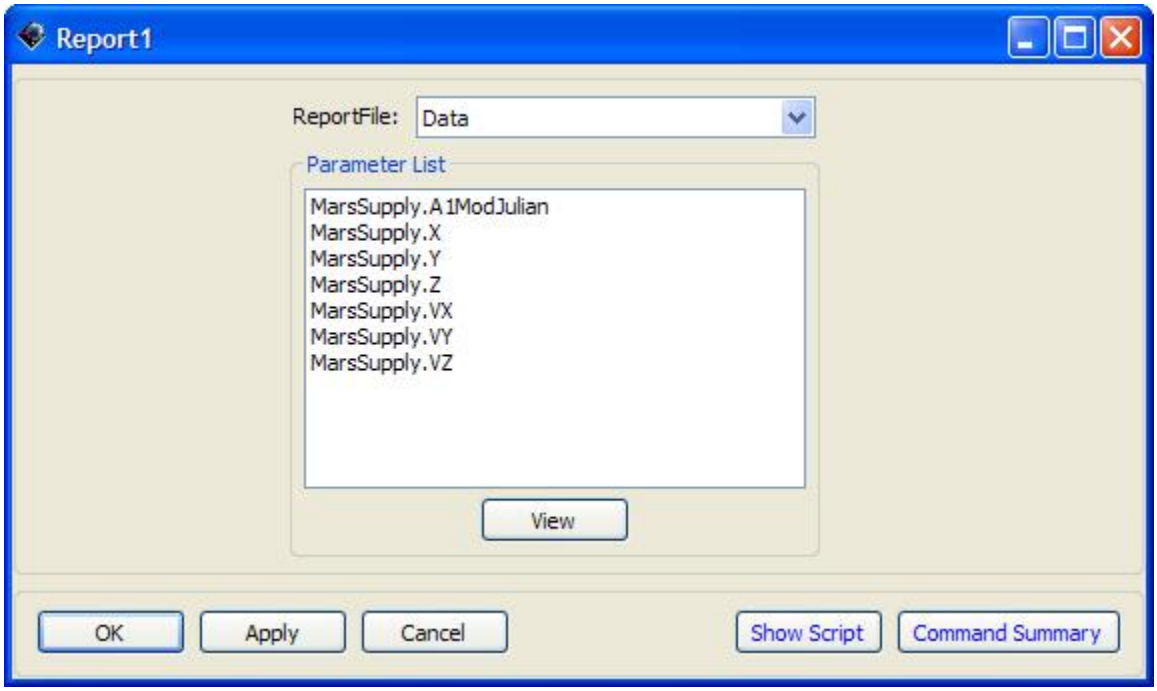


Figure: Report1 Report Command

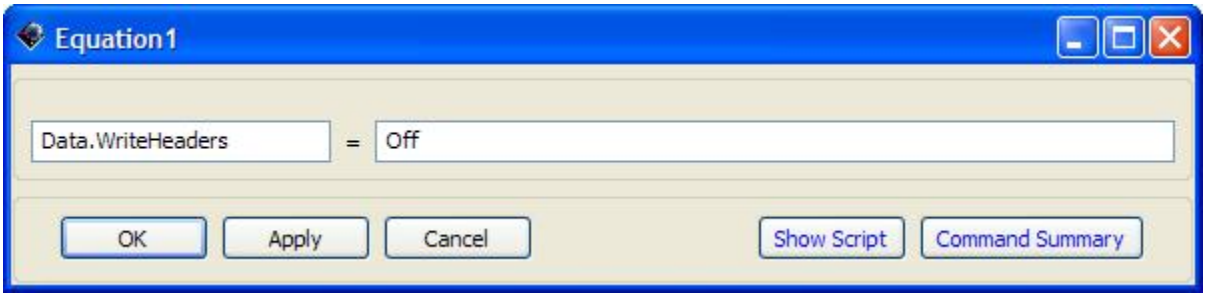


Figure: Equation1 Equation Command

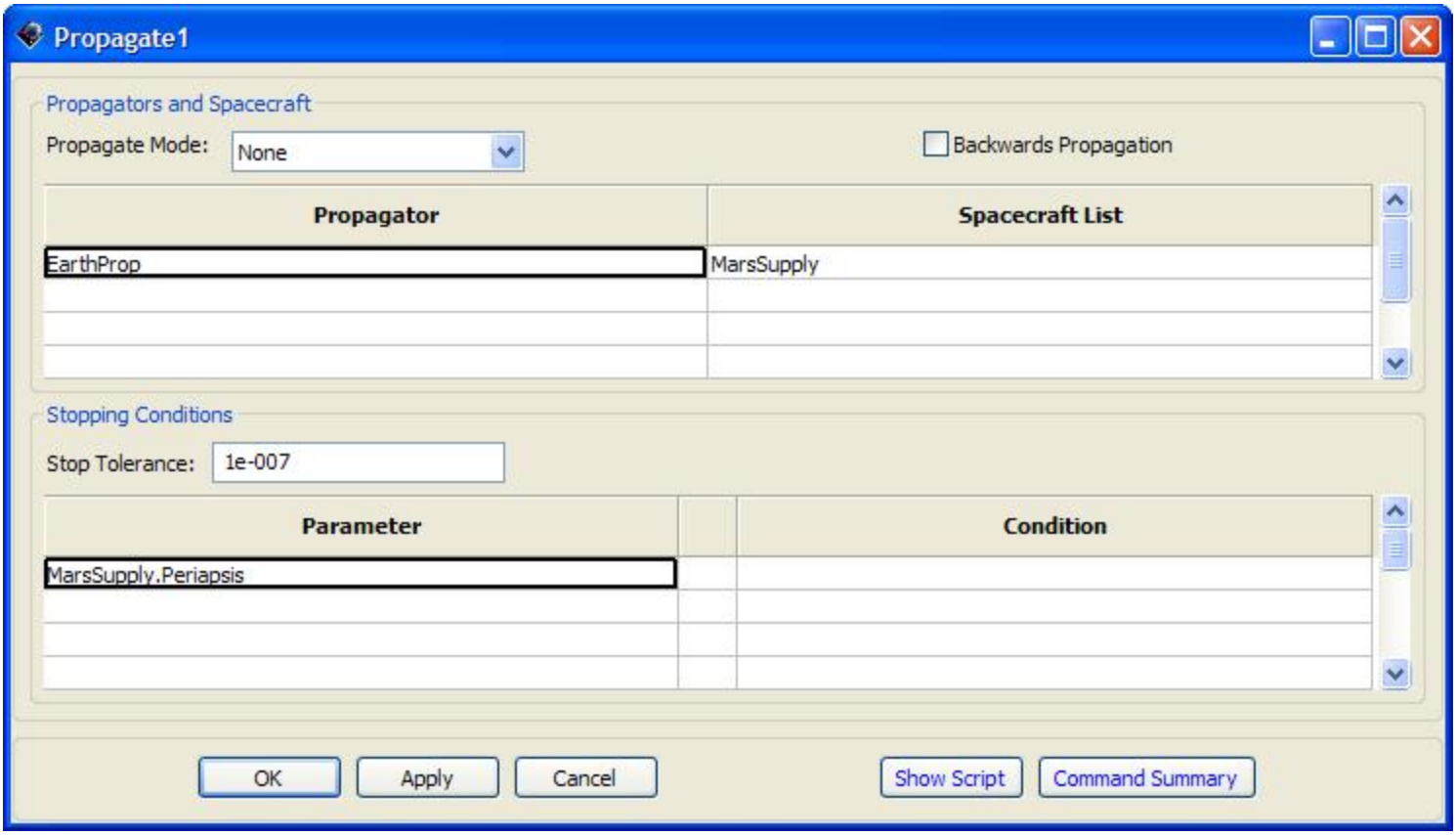


Figure: Propagate1 Propagate Command

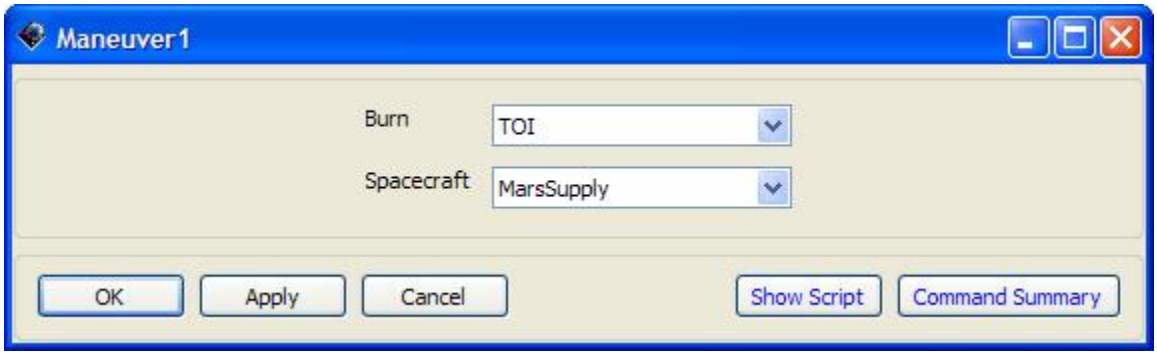


Figure: Maneuver1 Maneuver Command

Targeting Sequences

- Target 1:
  - Vary 1:
    - Set the variable to N using the MarsMCC impulsive burn
    - Set the Initial Value to 0 km/s
    - Set the Perturbation to 0.1

    - Define the Lower (Limit) to be - 8 km/s
    - Define the Upper (Limit) to be 8 km/s
    - Use a Max Step of 0.5 km/s
- Maneuver 2: utilize the MarsMCC impulsive burn
- Achieve 1:
    - Goal:
        - Highlight MarsSupply in the Object List menu
        - Select the SunMJ2kEq coordinate system from the drop down menu
        - Add INC to the Selected Value(s) field
    - Set the Value to 24.677°
    - Use a Tolerance of 0.001°
- Report 6: refer to Report 1 for parameters list

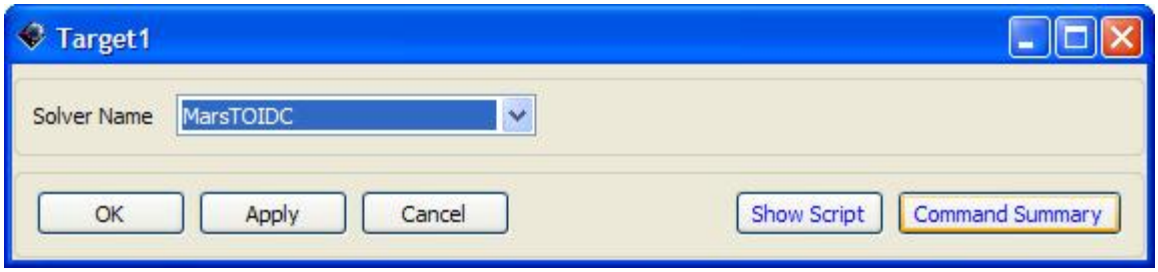


Figure: Target1 Target Command

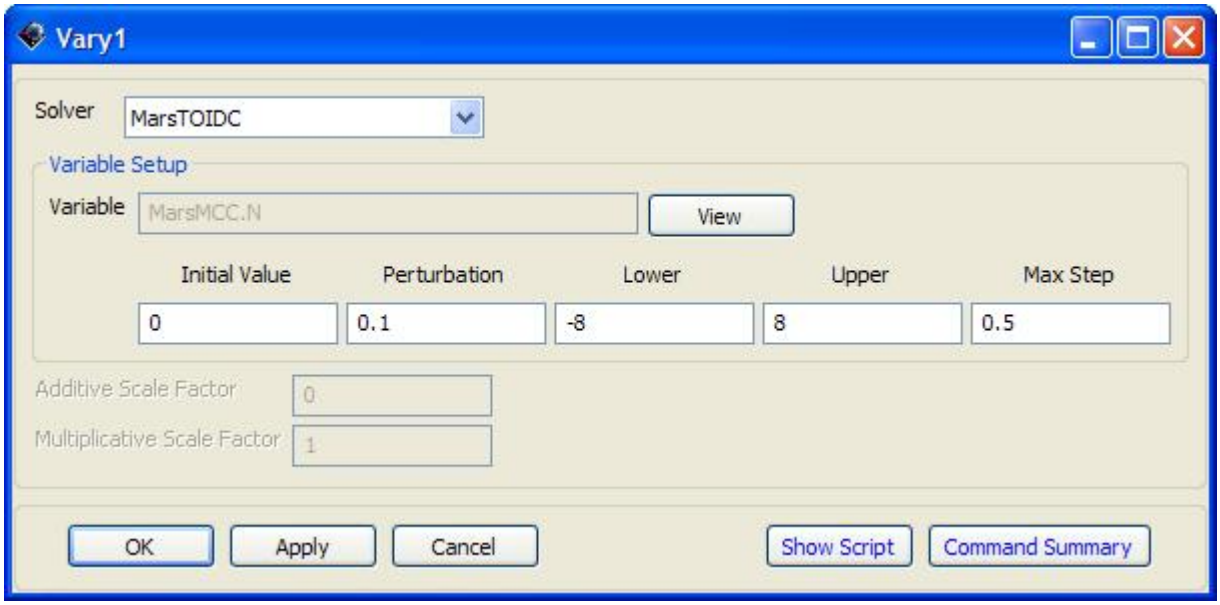


Figure: Vary1 Vary Command

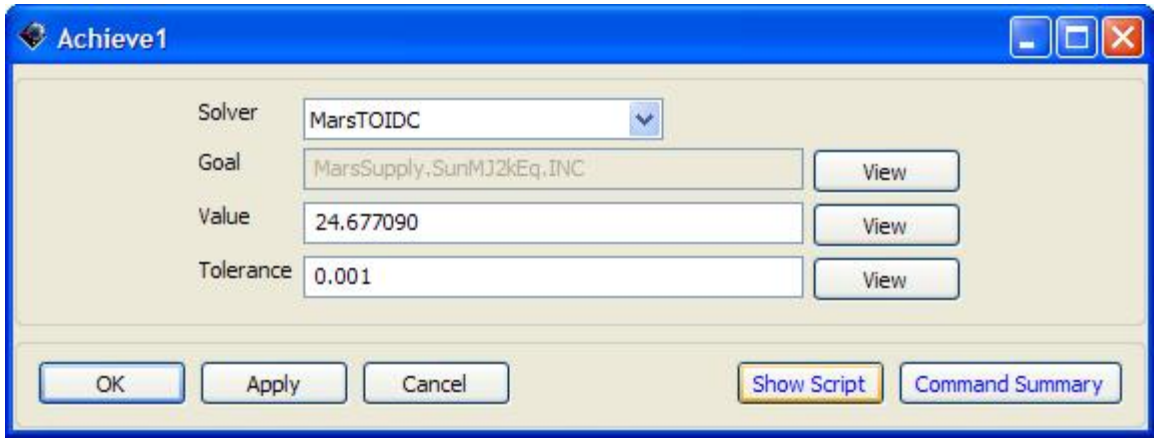


Figure: Achieve1 Achieve Command

- Target 2:
  - Vary 2:
    - Set the second Variable to V using the MarBPlane impulsive burn
    - Use an Initial Value to - 0.2338 km/s
    - Change the Perturbation to 0.001
    - Define the Lower (Limit) to be - 3 km/s
    - Define the Upper (Limit) to be 3 km/s
    - Use a Max Step of 0.1 km/s
  - Vary 3:
    - Set the third Variable to B using the MarsBPlane impulsive burn
    - Use an Initial Value of 0.6683 km/s
    - Change the Perturbation to 0.001
    - Define the Lower (Limit) to be - 3 km/s
    - Define the Upper (Limit) to be 3 km/s
    - Use a Max Step of 0.12 km/s
  - Maneuver 3: Utilizes the MarsBPlane impulsive burn
  - Propagate 4:
    - Use the SunProp propagator
    - Set the stopping condition parameter to 50 elapsed days
  - Propagate 5:

    - Use the MarsProp propagator
    - Set the stopping condition parameter to be Mars' periapsis
  - Achieve 2:
    - Goal:
      - Highlight BdotT
      - Select the MarsMJ2000Eq coordinate system from the drop down menu
      - Add BdotT to the Selected Value(s) field
    - Value = 10000 km
    - Tolerance = 50 km
  - Achieve 3:
    - Goal:
      - Highlight BdotR
      - Select the MarsMJ2000Eq coordinate system from the drop down menu
      - Add BdotR to the Selected Value(s) field
    - Value = -10000 km
    - Tolerance = 50 km

Final Propagate and Maneuver Sequence

- Report 7: refer to Report 1 for parameters list
- Maneuver 4: utilize the MarsOI impulsive burn
- Report 8: refer to Report 1 for parameters list
- Propagate 6:
  - Use the MarsProp propagator
  - Set the stopping condition to 25 elapsed days
- Report 9: refer to Report 1 for the parameter list

The resulting view of Resource, Mission, and Output Trees

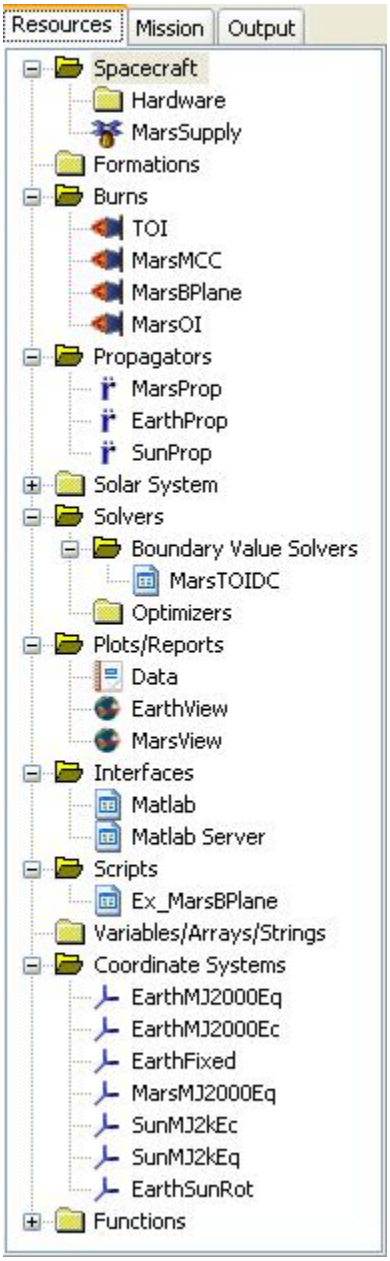


Figure: Resource, Mission, and Output Trees

Running the Mission

Resource Mission and Output Trees

You have now configured the mission, and it is time to run it and see the results. First, save the changes you've made and then click the Run button (green arrow on the tool bar) to run the mission. The screen capture below summarizes what you should see after correctly completing this tutorial.

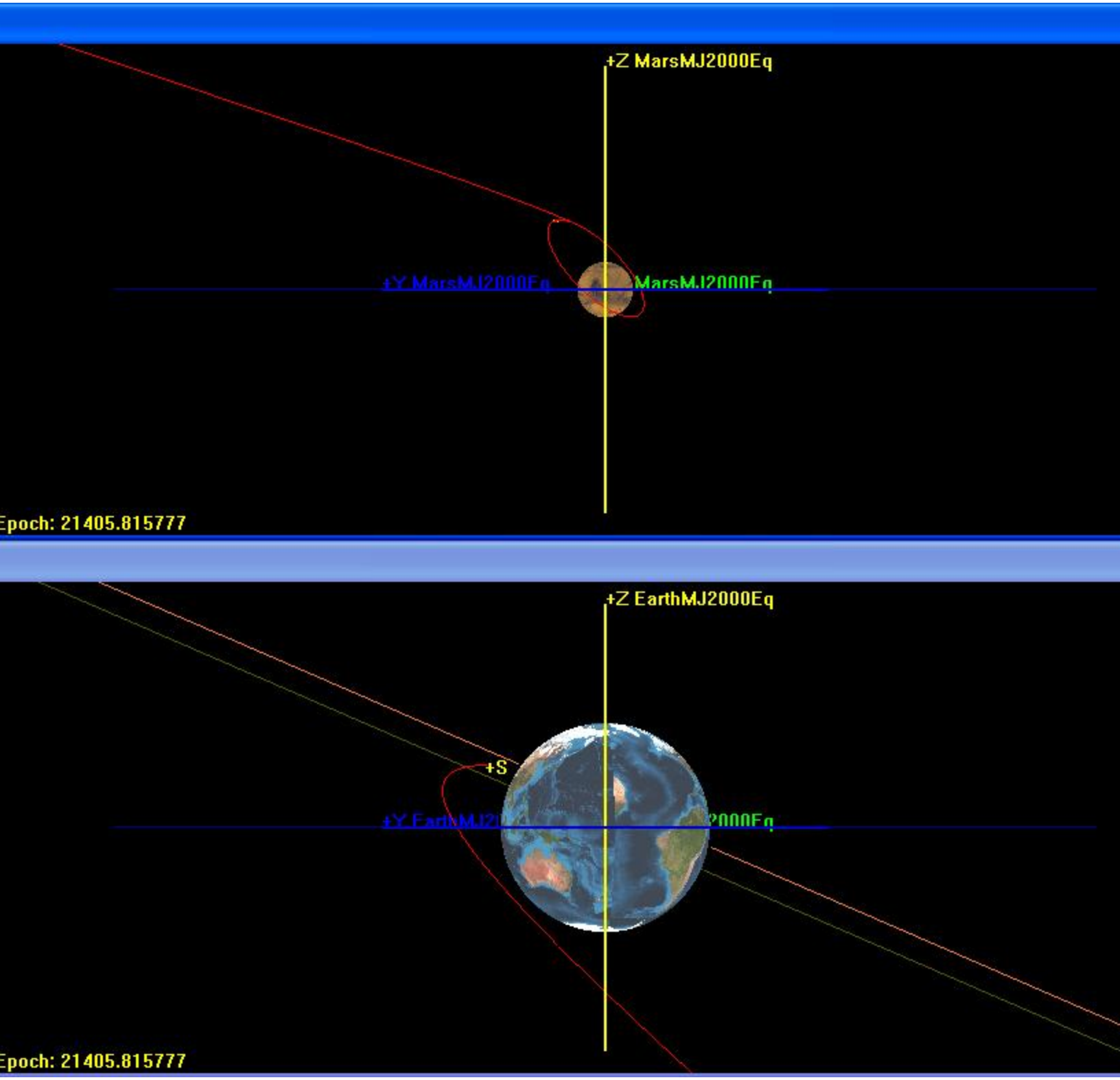


Figure: Output

Chapter 17. L2 Transfer with Swingby

This tutorial was started from the default settings in the June 9, 2008 build of GMAT.

Objective and Overview

The goal of this tutorial is to teach one to create and propagate a transfer trajectory from a LEO (Low Earth Orbit) to the Sun-Earth Lagrangian point L2. To this end, GMAT's tools called Control Logic and the Bisection method will be used. You will learn how Equations and Control Logic statements - namely, If, If Else, For, and While - can be used to find the solution to this relatively complicated problem. Also used will be the Toggle Action, the Report command, and multiple viewpoints with different coordinate systems.

You can download the script file and run it beforehand to see the final results of this tutorial: LunarTransferToL2-tutorial.script [scripts/LunarTransferToL2-tutorial.script]

Prerequisites

Mission Description

- Objective: Target a lunar transfer trajectory with several initially undetermined variables, ending in an orbit about the Sun-Earth Lagrangian point L2.
- Assume:
  - Epoch = 01 Dec 2017 01:07:06.978
  - SMA = 6563.421890530317
  - ECC = 0.001999999999998943
  - INC = 28.68369284287025
  - RAAN = 263.0110331380764
  - AOP = 138.5702976742882
  - TA = 349.1878928421385
- Find:
  - RAAN
  - AOP
  - value #V from LEO to achieve a given energy; value B·T with respect to the Moon, required to achieve the given energy; and value #V to place the spacecraft into an orbit about the Sun-Earth Lagrangian point L2

Creating and Configuring the Resource Tree

For this tutorial, two spacecraft objects, two impulsive burns, and 24 variables are used to achieve an orbit about point L2. To realistically account for the varying gravitational fields of the Sun, Earth, and Moon, we

will use two propagators. Six OpenGL plots will provide good views of the spacecraft and its trajectory. Finally, three coordinate systems will be used for an adequate representation of the mission events.

Objects Required

- Spacecraft: InitSat and Sat
- Impulsive Burns: TOI, P1, and P2
- Solar System Special Points: EarthMoonBary and ESML2
- Propagators NearEarthProp and NearMoonProp
- Plots/Reports: MoonView, EarthInertial, EarthView, EarthView, AboveL2, and L2View
- Variables/Arrays/Strings: YAmp, I, SMA, ECC, INC, TA, RAAN, AOP, Epoch, EnergyError, Flow, FHigh, Prod, c, DesiredEnergy, Error, B·T, LowerBound, UpperBound, LB, UB, BoxSize, LowerBoxSize, and UpperBoxSize
- Coordinate Systems: EarthMoonRot, EarthSunRotCS, ESML2Centered, MoonInertial

Creating and Modifying Objects

- Spacecraft
  - Add spacecraft Sat
  - Use the UTCGregorian epoch format
  - Use the Keplerian State Type
  - Under the Ballistic/Mass tab within the Spacecraft
    - DryMass = 831;
    - Cd = 2.2;
    - Cr = 1.3;
    - DragArea = 20.438;
    - SRPArea = 20.438;
  - Clone Sat and rename the new spacecraft InitSat. (Two figures below show how the resulting windows of Sat properties should look like.)

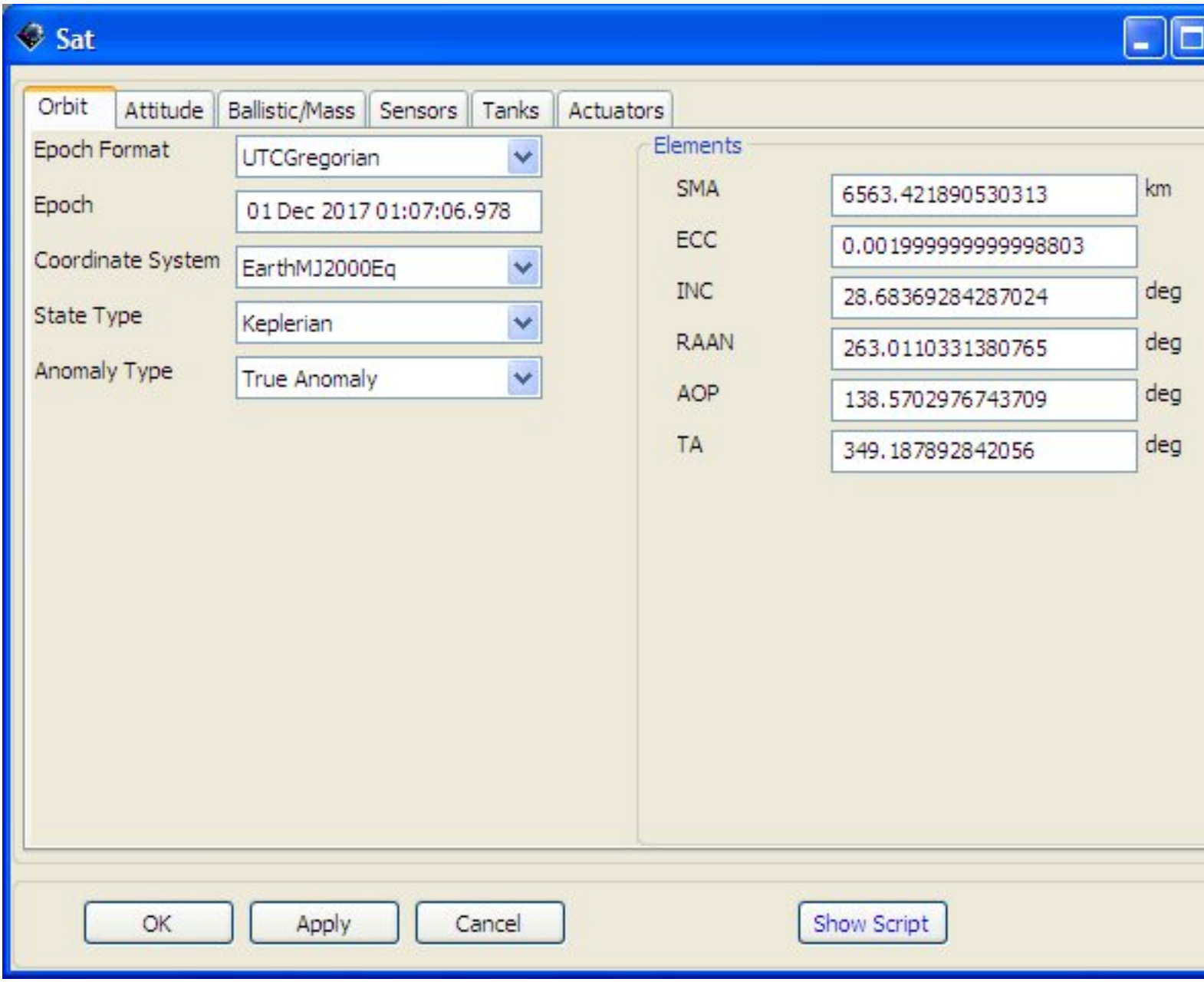


Figure: Sat Spacecraft Object-Orbit Tab

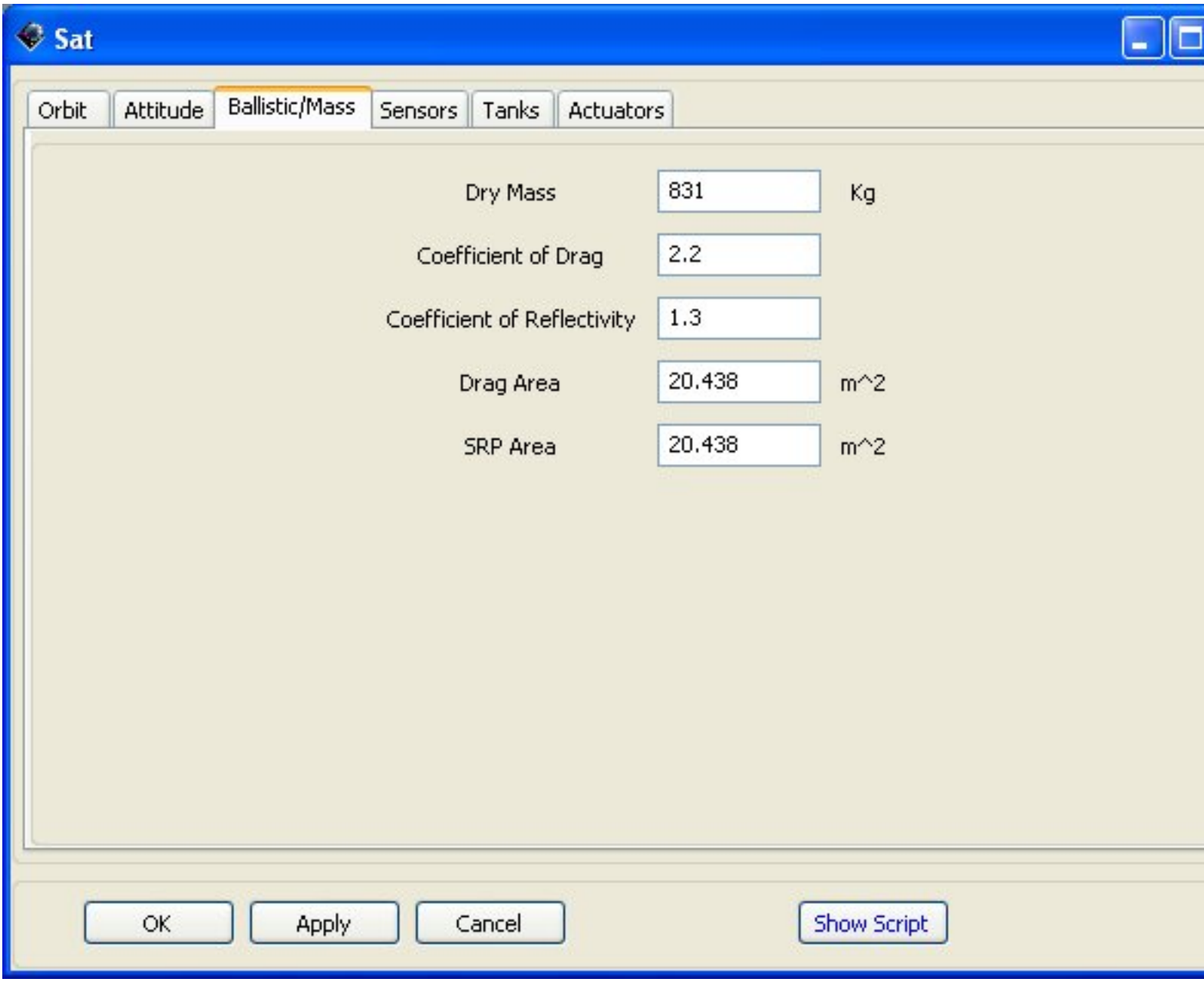


Figure: Sat Spacecraft Object-Ballistic/Mass Tab

- Burns
  - TOI:
    - V=3.1 - this is roughly the amount of burn required to go from LEO to near the Moon

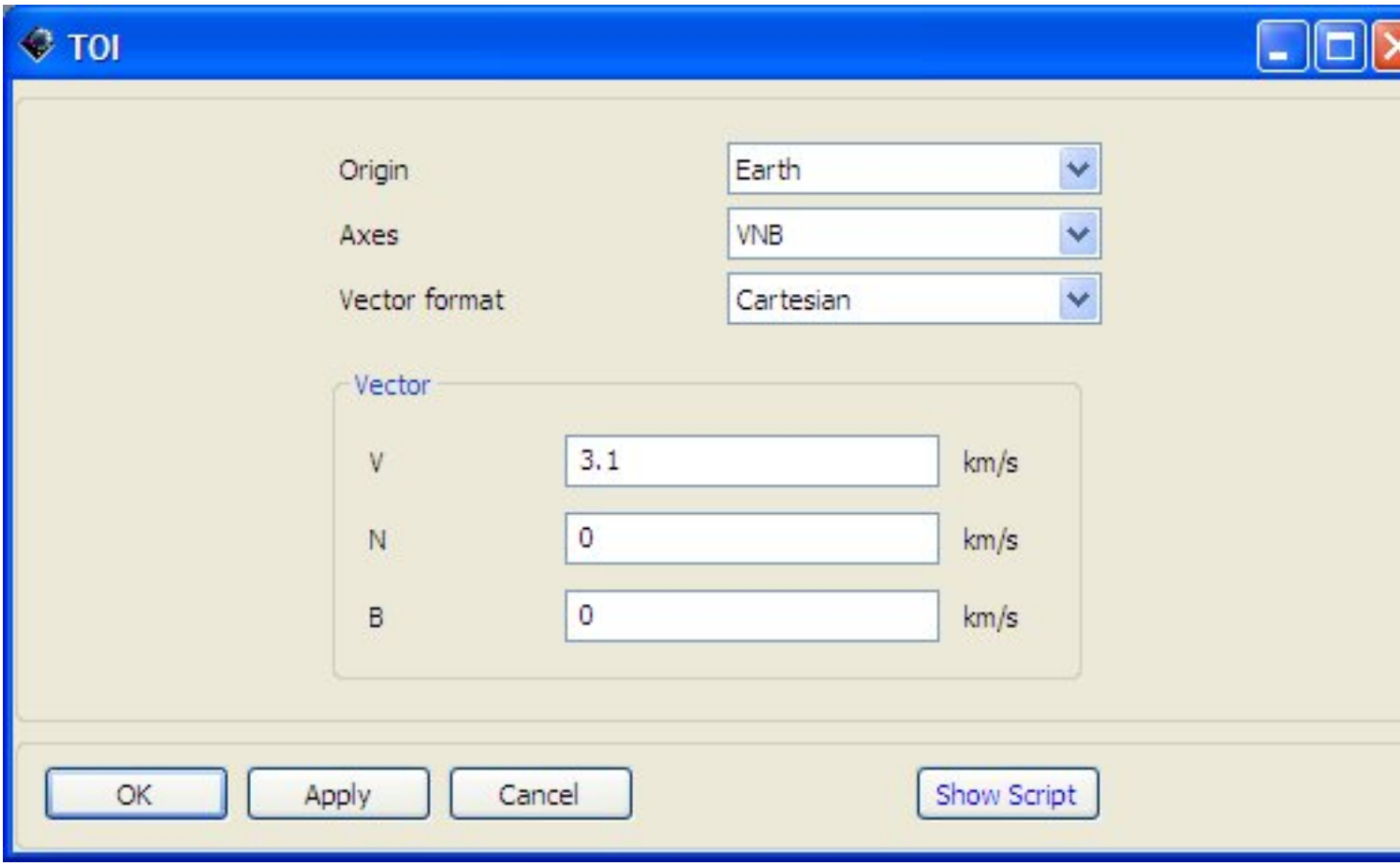


Figure: Burn Object

- P1 and P2: leave the default settings

- Solar System Special Points
  - ESML2:
    - Add a Libration Point.
    - Sun should be the Primary Body, Earth - the Secondary body; select L2

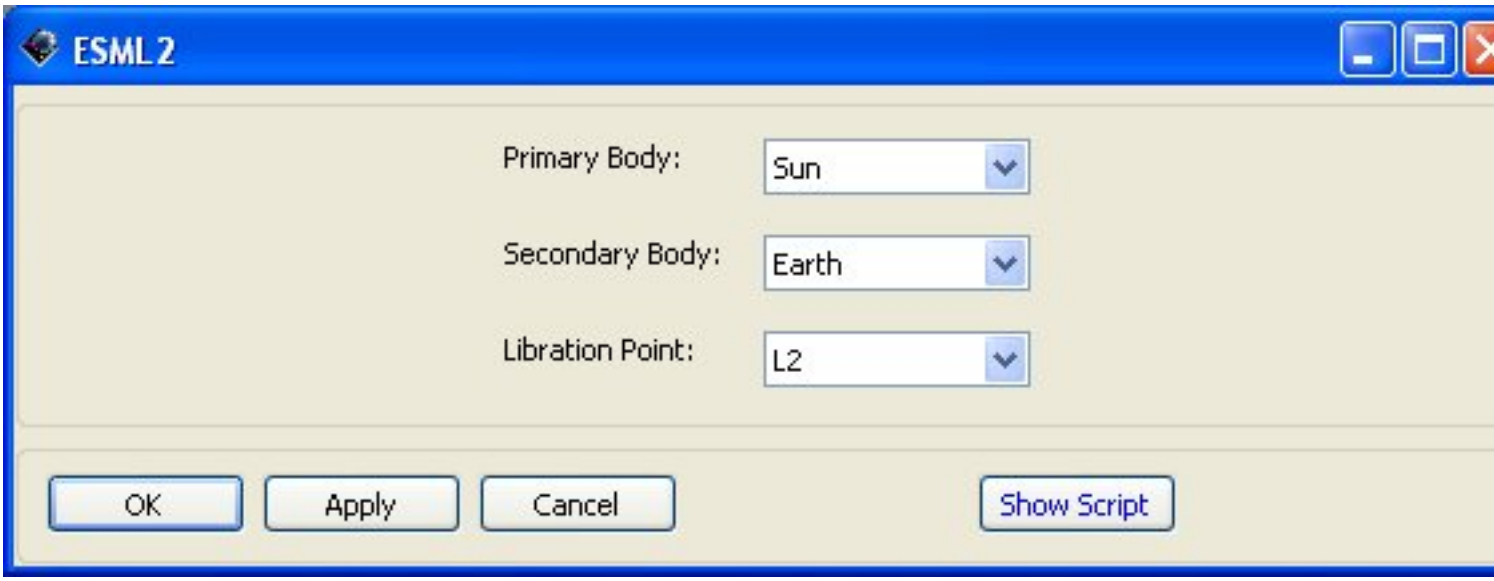


Figure: ESML2 Libration Point Special Point Object

- EarthMoonBary:
  - Add a BaryCenter.
  - Select Earth and Luna

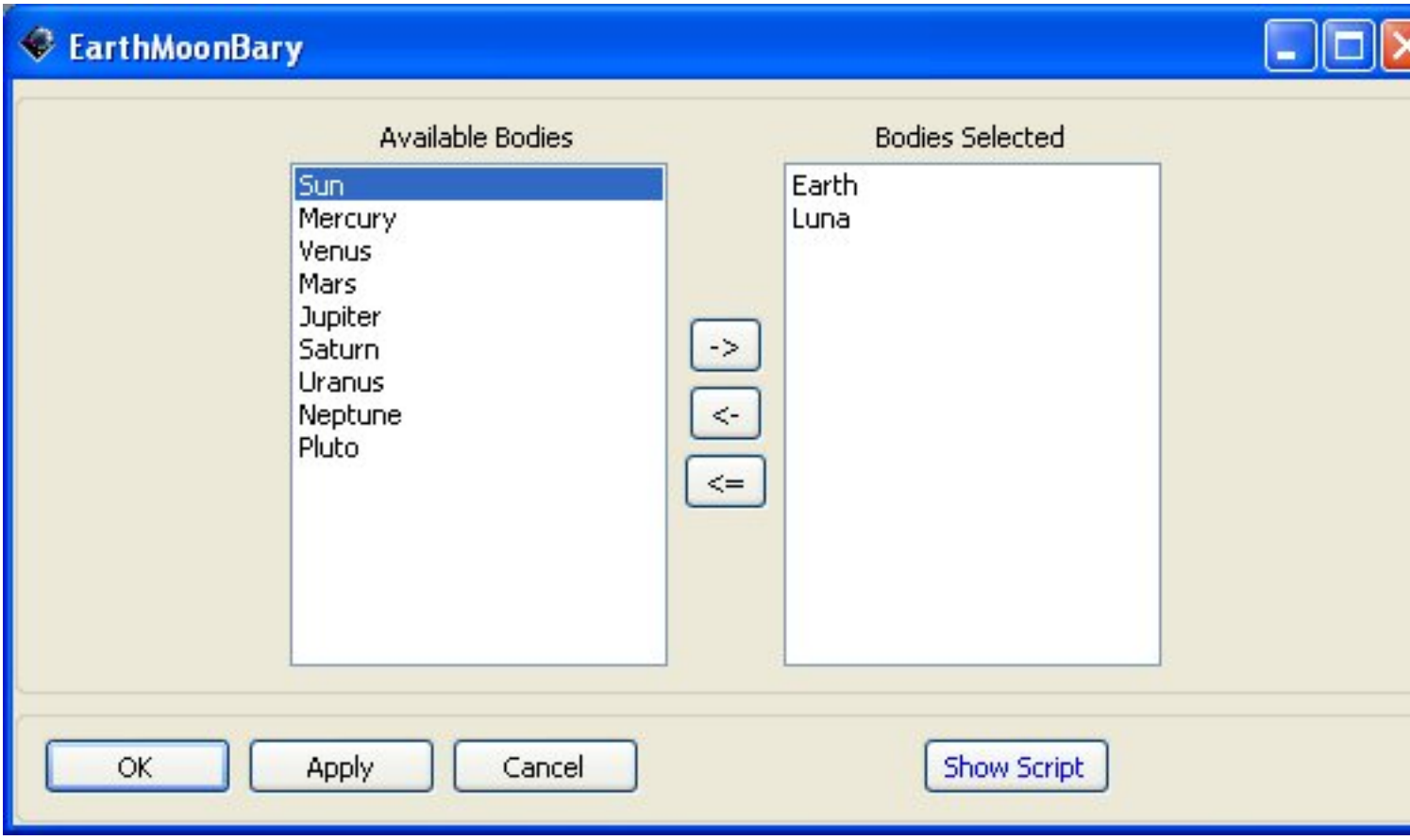


Figure: Barycenter Special Point Object

- Propagators
  - NearMoonProp:
    - Use Luna as the Primary body
    - LP-165 gravity model with degree and order of 10
    - Add Earth, Sun, and Jupiter as point masses
    - Include SRP flux

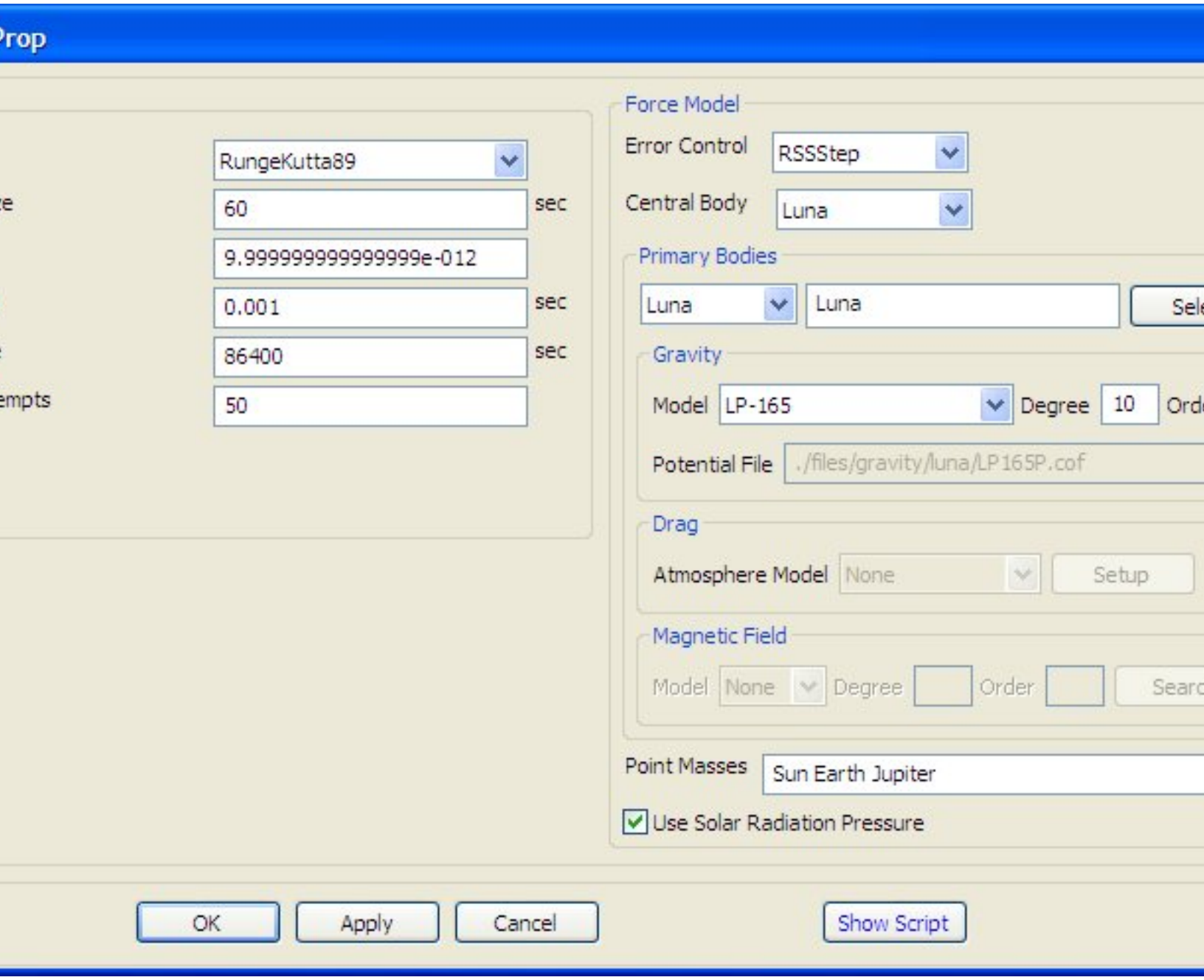


Figure: Propagator Object

- NearEarthProp:
  - Use Earth as the Primary body
  - Includes heliocentric and lunar point masses
  - Include SRP flux
- Coordinate System

- EarthMoonRot:
  - Use Earth as the origin
  - Axes Type is ObjectReferenced
  - The Primary body is Luna and the Secondary body is Earth

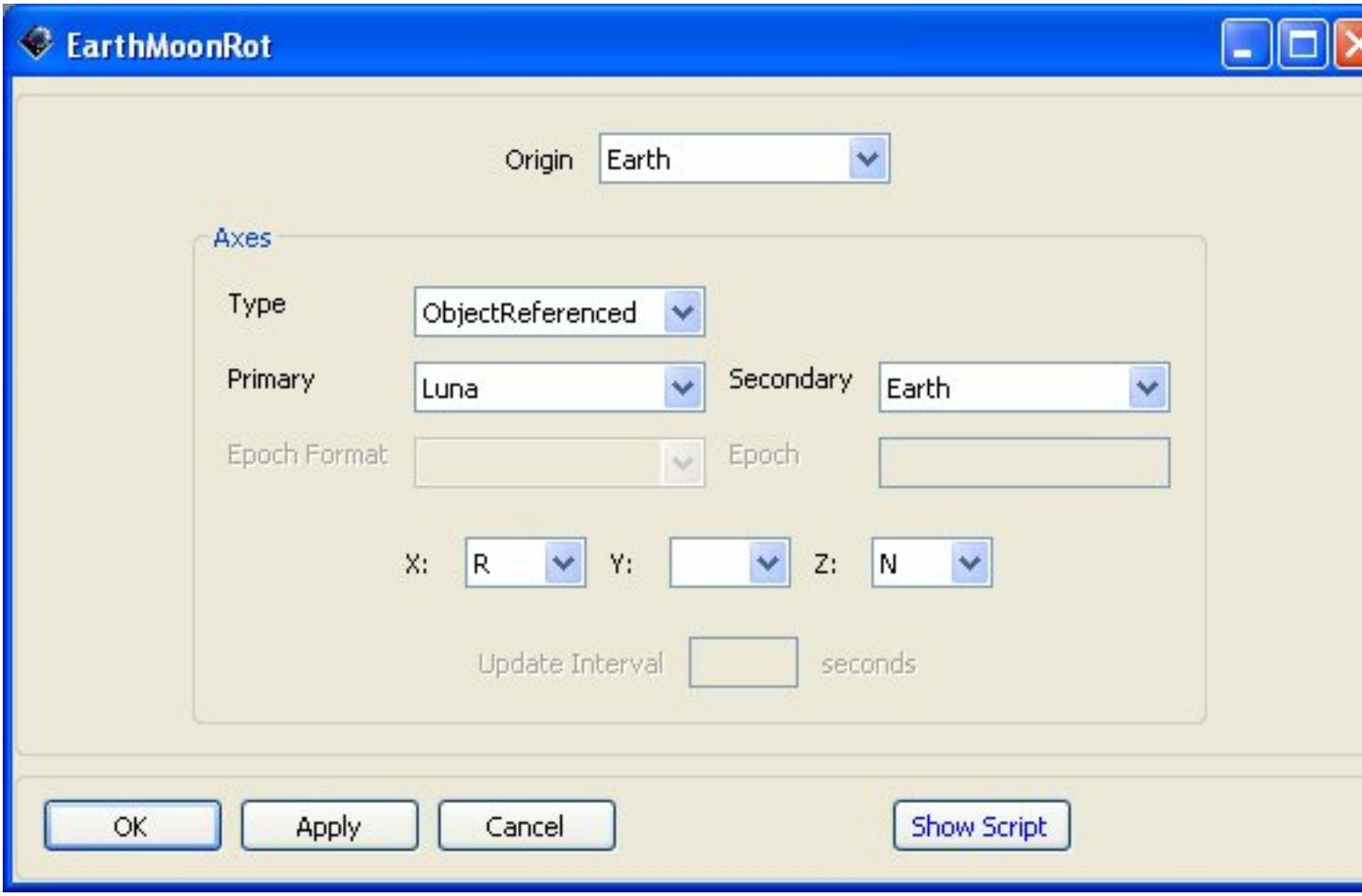


Figure: EarthMoonRot Coordinate System Object

- EarthSunRotCS:
  - Use Earth as the origin
  - Axes Type is ObjectReferenced
  - Primary body is Sun, Secondary body is Earth
- ESML2Centered:
  - Use ESML2 as the origin
  - Axes Type is ObjectReferenced

    - Primary body is Sun, Secondary body is Earth
  - MoonInertial:
    - Use Luna as origin
    - Axes Type is BodyInertial
    - Epoch is 21544.99962789778
- Plots/Reports
  - SystemView:
    - Add Luna to the Selected Celestial Objects field
    - Use EarthMoonRot
    - ViewPointVector <10000, 10000, 10000>
    - Use a View Scale Factor of 40
    - Select Z as View Up Definition Axis with the Coordinate System EarthMoonRot
    - Collect Data every 3 steps, Update Every 50 cycles, and redraw 100 points
    - For Drawing options, show All iterations and Draw only the Axes

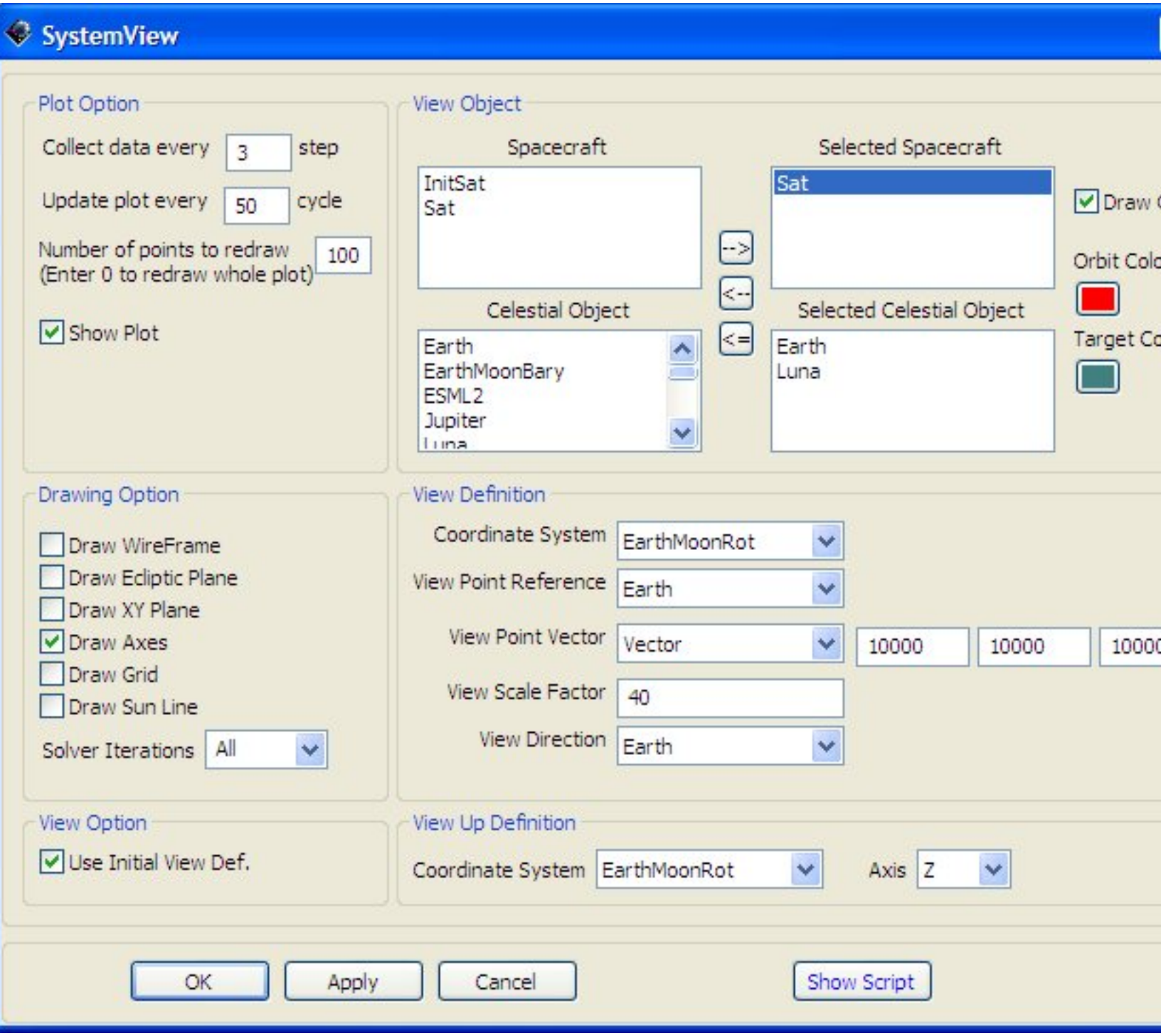


Figure: SystemView OpenGL Plot Object

- EarthView:
  - Add Luna and ESML2 to the Selected Celestial Objects field

    - Use the EarthSunRotCS coordinate system in the View Definition and View Up Definition group boxes
    - The ViewUp definition should be along the Y-Axis
    - Have no Drawing options selected, and show none of the solver iterations
    - Collect data every 5 steps, update every 50 cycles
- EarthView2:
    - Same as EarthView1
- MoonView:
    - Add Luna to the Selected Celestial Objects field
    - Use the MoonInertial coordinate system in the View Definition and View Up Definition group boxes
    - Vector should be <30000 30000 30000>, with the scale factor of 1
    - The ViewUp definition should be along the Z-Axis
    - Draw only Axes, no iterations
    - Collect data every 5 steps, update every 50 cycles, redraw 150 points
- EarthInertial:
    - Add Luna, ESML2, Earth, and Sun to the Selected Celestial Objects field
    - Use EarthMJ2000EQ coordinate system in the View Definition and View Up Definition group boxes
    - Vector should be <0 0 30000>, with the scale factor of 100
    - The ViewUp definition should be along the Y-Axis
    - Draw only Sun Line, no iterations
    - Collect data every 5 steps, update every 50 cycles, redraw 150 points
- AboveL2:
    - Add Luna, Earth, and ESML2 to the Selected Celestial Objects field
    - Use the ESML2Centered coordinate system in the View Definition and View Up Definition group boxes
    - View Point Reference at ESML2
    - Vector should be <0 0 30000>, with the scale factor of 75
    - The ViewUp definition should be along the Y-Axis
    - Draw only Axes, no iterations

    - Collect data every 5 steps, update every 50 cycles, redraw 150 points

- Report:
    - Add a ReportFile called Data
    - Left Justify, Precision = 16, ZeroFill off, ColumnWidth = 20, WriteHeaders = On
    - Choose the folder for the file. (File name here is provided only as an example).

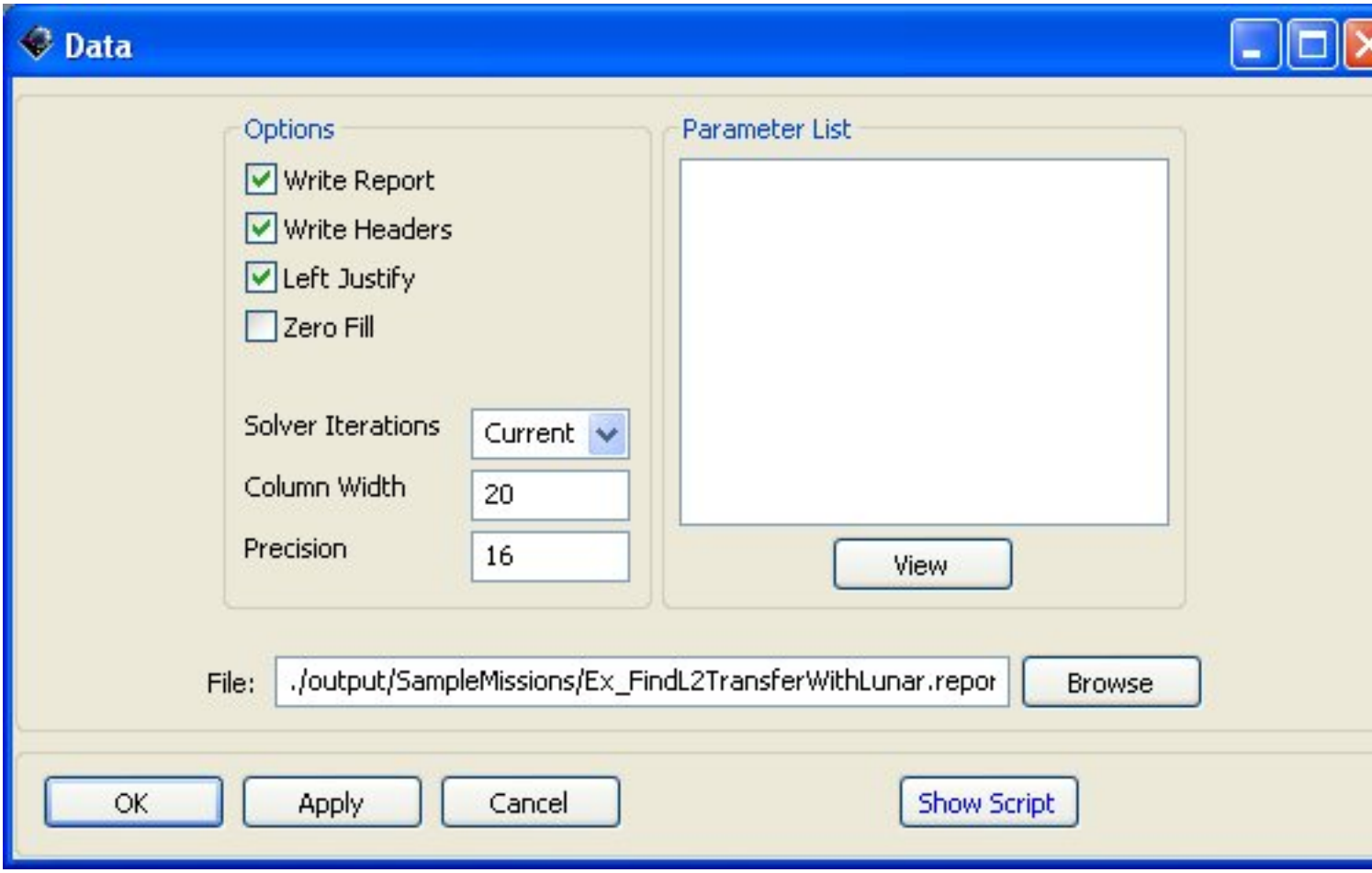


Figure: Data Data Report Object

- Variables, Arrays, Strings
    - Add variables YAmp, I, SMA, ECC, INC, TA, RAAN, AOP, Epoch, EnergyError, FLow, FHigh, Prod, c, DesiredEnergy, Error, B·T, LowerBound, UpperBound, LB, UB, BoxSize, LowerBoxSize, UpperBoxSize
    - Set all variables equal to zero and to the Sat spacecraft (as shown in figure below for variable SMA)

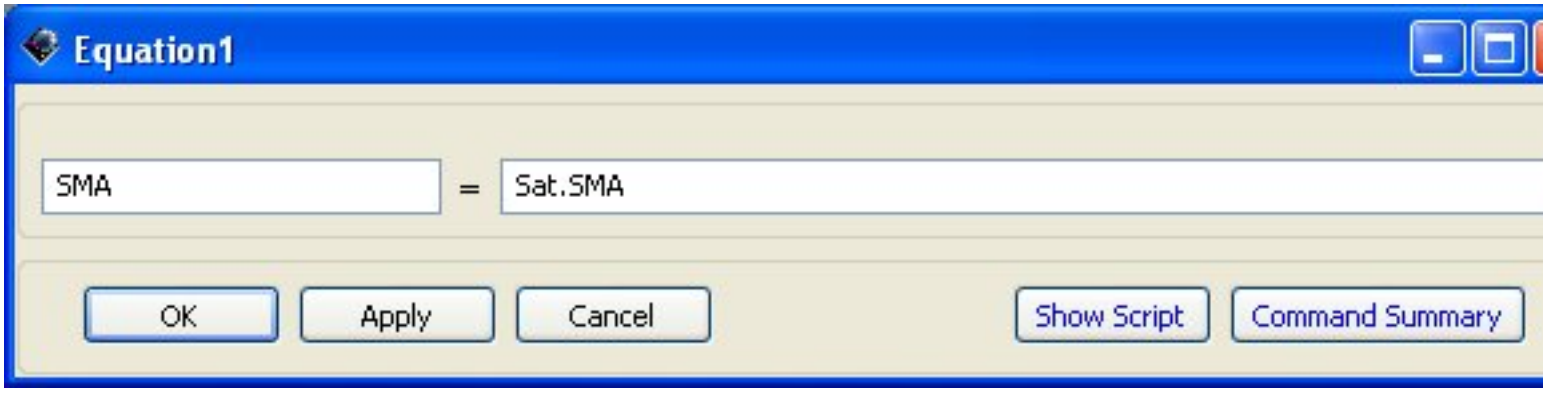


Figure: Equation 1 Variable Object

Creating and Configuring the Mission Sequence

Resource Mission and Output trees

To accomplish the mission, three targeting sequences are used, together with a relatively complicated set of equations and control logic statements. The goal is to calculate the thrust that is required to take the spacecraft from LEO, perform a flyby of the Moon, and place it in an orbit about the Earth-Sun Lagrange point L2. (For those interested, Wikipedia has a good explanatory text on Lagrange points.) Also, B-plane targeting is used here, similar to that in the Lunar Transfer tutorial.

- The First Targeting Sequence aligns the estimated trajectory with the Earth-Moon line. To accomplish this, AOP and RAAN values are varied. The thrust maneuver is applied before the targeting sequence, with an estimated 3.1 km/s, a good but intentionally low guess that will be modified in the following sections.
- The Sat Equations take the Sat spacecraft through a series of operations that put it in the correct location. The equations use the initial conditions of Sat as well as the values of AOP and RAAN found in the previous sequence; they also set values for some variables.
- The First Bisection Method section contains the counter equation, as well as a setup for the first and second iteration which deal with the lower and upper bound respectively.
- While Loop
  - The first targeting sequence within the While loop varies P1.V and P1.B, to attain a given value of BR. The spacecraft is propagated from its initial condition to the Moon's periapsis, utilizing two separate propagators that differ by their primary bodies.
  - The second targeting section propagates the spacecraft past the Moon until it enters a required box (region), and then propagates it until it leaves a larger box.
  - The third targeting section is where the Bisection method is really set up. The difference between the goal and the resultant is found for the resultant. The first iteration resultant is stored into a variable for the lower boundary, and the second iteration resultant is stored into a variable for the upper boundary. The sign of the product of the two resultants is checked to ensure that the two boundaries lie on both sides of the expected answer. If this is not the case, the sequence is stopped.

  - The fourth section is the core of the Bisection method. The sign of the difference between the iteration's result and the goal is checked against the boundaries. The boundary with which it has the same sign as is replaced and the range that the answer can be contained in is cut in half.

- The Intermediate Stage places the spacecraft back at its initial conditions immediately following the first burn. The adjusted burn, P1, is added to the original burn. These conditions for the spacecraft are then stored again into the variable InitSat. Additional plots are turned on; some values are stored into variables for the forthcoming targeting sequence.

- The Final Target Sequence finds an additional fine tuning burn, P2, such as to provide the spacecraft with only a very small velocity when away from the Earth. The spacecraft is propagated from its original position with the P1 burn added in. It is propagated past the Moon and near L2. This is done in a similar manner as before. A For loop is used for this fine tuning, to ensure the spacecraft stays withing a certain region of space.

Creating and Modifying Command Sequences

First Targeting Sequence

The purpose of the first targeting sequence is to find values for AOP and RAAN that will position the spacecraft, after the maneuver, near the Moon. This is very similar to what was done in the Lunar Transfer Sample Mission. Instead of aiming for a particular set of coordinates, we are aiming here for a DEC of zero and RA of 180 degrees within the EarthMoonRot coordinate system. This means that the spacecraft is expected to be put on the line from Earth to the Moon. (For more information on the values of DEC and RA, refer to the relevant Wikipedia articles.)

This target sequence does several things that need more explaining. The role of the equations is to store the values AOP and RAAN before the spacecraft is propagated. This enables us to place the spacecraft back at those particular locations after the propagation is complete. This is similar to what was done in the Lunar Transfer tutorial.

- Command Toggle1 maintains clean plots until they are needed for later targeting. This prevents multiple targeting sequences from overlapping and, since it is not blocked by the earlier sequences, allows for a better understanding of the later targeting sequence. This is a very appropriate time to use the Toggle Action.

- Maneuver1 applies the rough maneuver of TOI to Sat, so as to bring it near the Moon, for the first targeting sequence. At this point there is no intent for Sat to reach the exact desired location. Rather, the value #V=3.1 km/s is an estimate of what will be required.

- Equation1 through Equation7 do a few things. Equations 1-4 store the initial values of the elements of Sat into variables so that they can be accessed later. Equation5 sets the required Epoch, and Equation6 uses that variable to set the Sat's timing. Equation7 sets InitSat to the same elements that Sat has, so that later, after Sat is propagated, InitSat will be storing the Sat's initial elements.

- The conditions on the Propagate1 are unusual. The first two conditions make sure that if the current iteration fails to bring Sat near the Moon, it will stop, so that the next iteration can be tried. The third condition is that if Sat gets close enough to the Moon, the iteration will stop, and will check if the Achieve conditions are fulfilled.

- Toggle 1:

  - Add a toggle command to the mission sequence

  - Uncheck EarthView2 and AboveL2 to turn those subscribers off

- Equation1: SMA = Sat. SMA
- Equation2: ECC = Sat.ECC
- Equation3: INC = Sat.INC
- Equation4: TA = Sat.TA
- Equation5: Epoch = 28025
- Equation6: Sat.A1Epoch = Epoch
- Equation7: InitSat = Sat;
- Target 1:
  - Vary AOP (range of -360 to 360)
  - Vary RAAN (range of -360 to 360)
  - Equation8: RAAN = Sat.RAAN
  - Equation9: AOP = Sat.AOP
  - Propagate using NearEarthProp til:
    - RMAG from Earth is 34000
    - ElapsedDays = 4
    - RMAG from Moon is 1000
  - Achieve RA = 180, Tolerance = 1
  - Achieve DEC = 0, Tolerance = 1

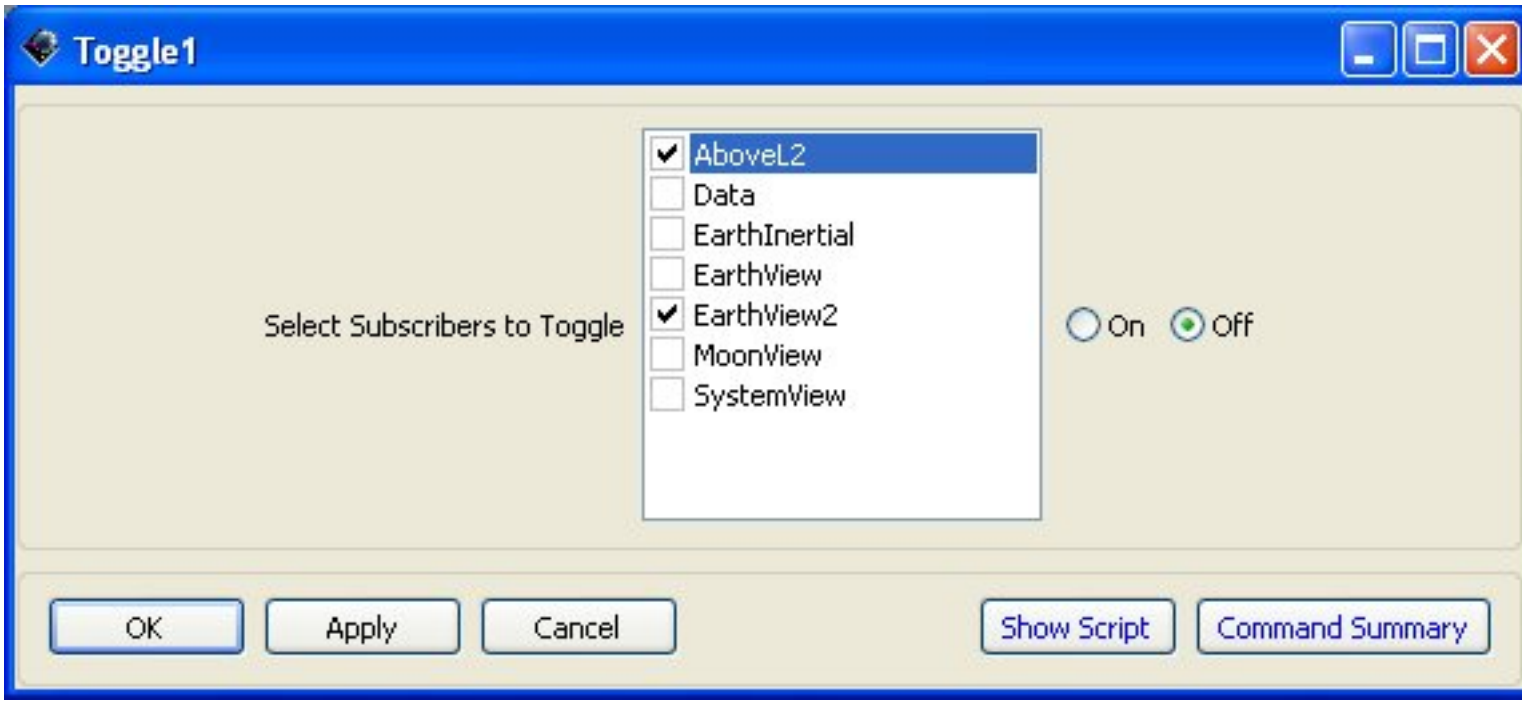


Figure: Toggle1 Toggle Command

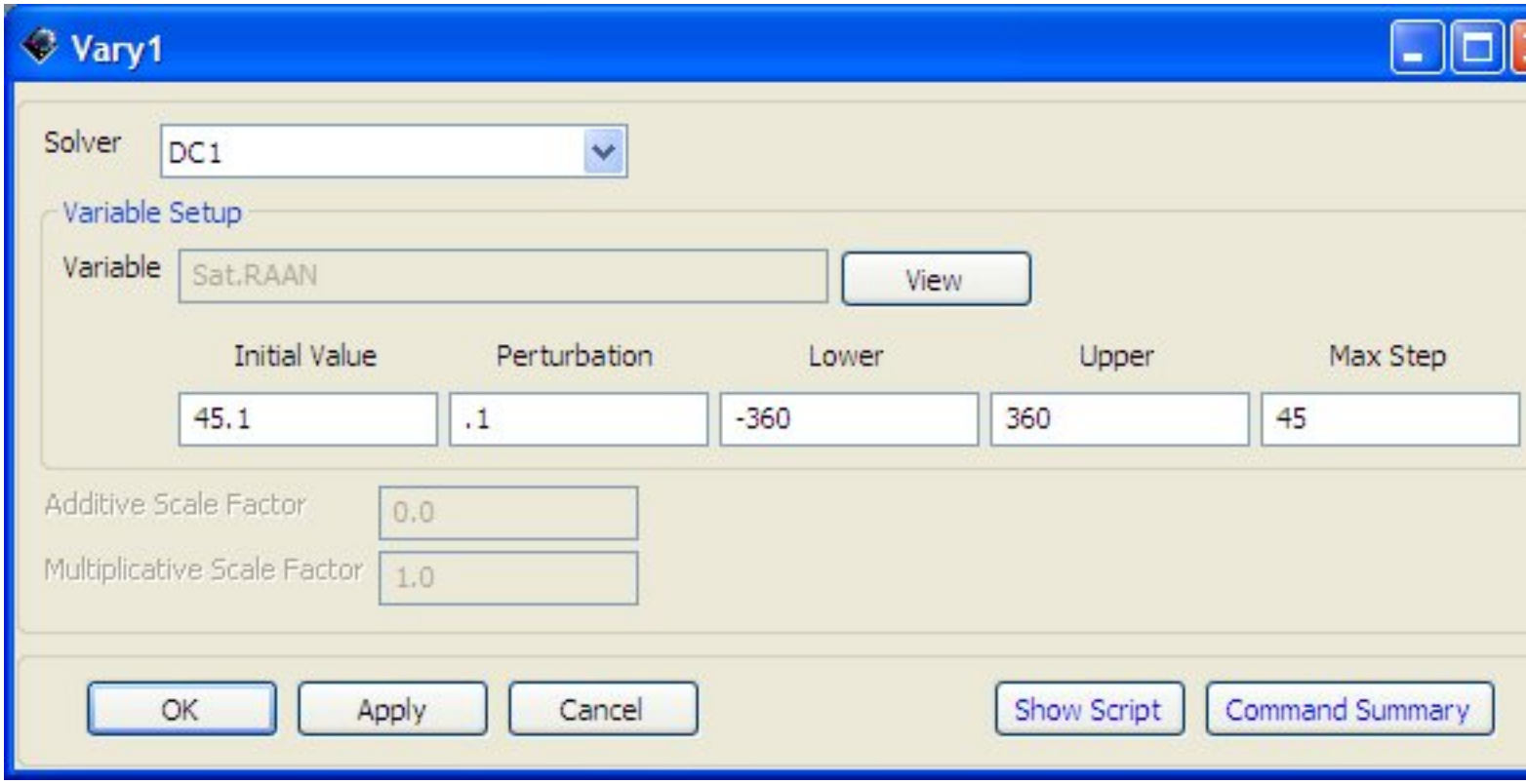


Figure: Vary1 Vary Command

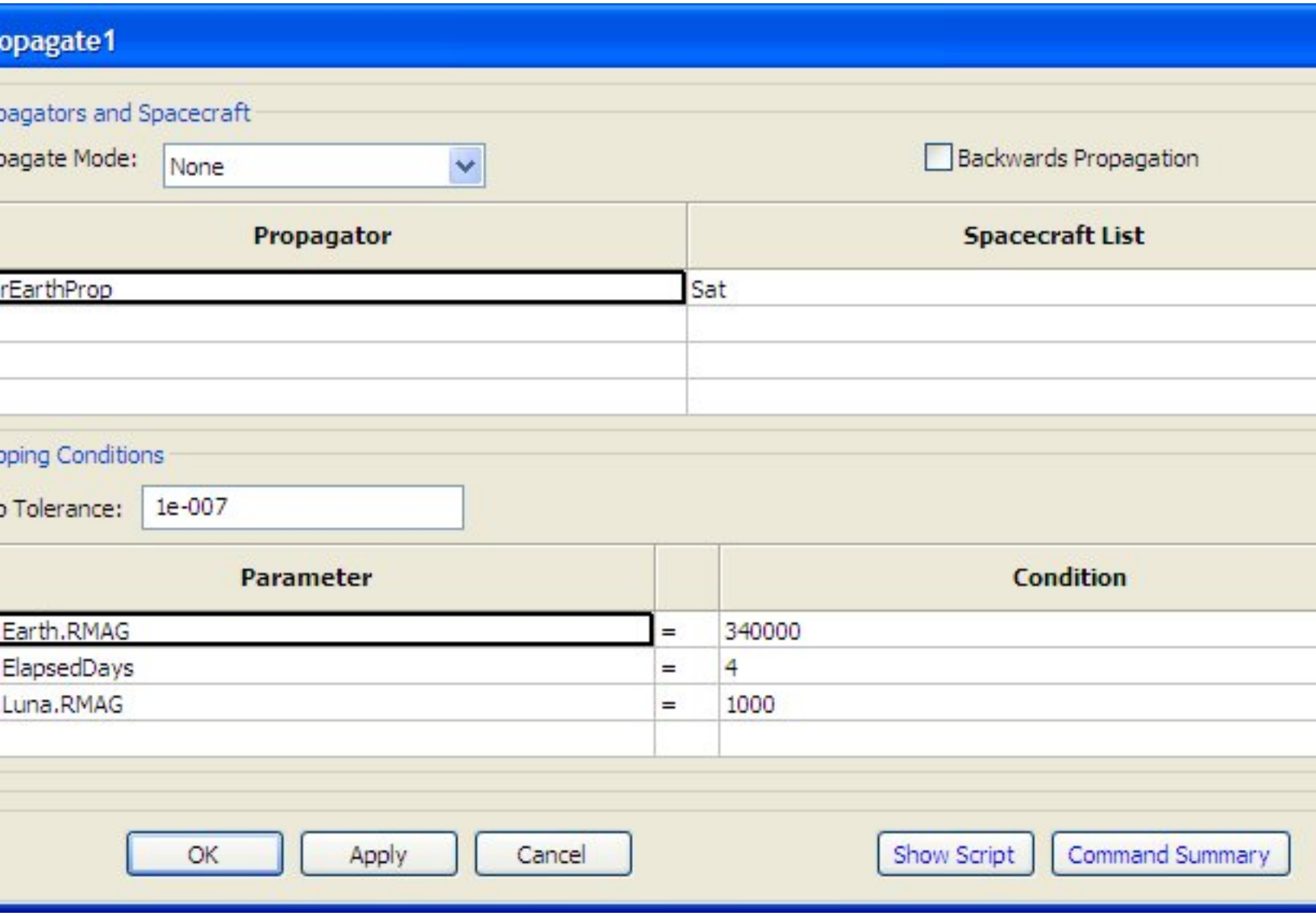


Figure: Propagate1 Propagate Command

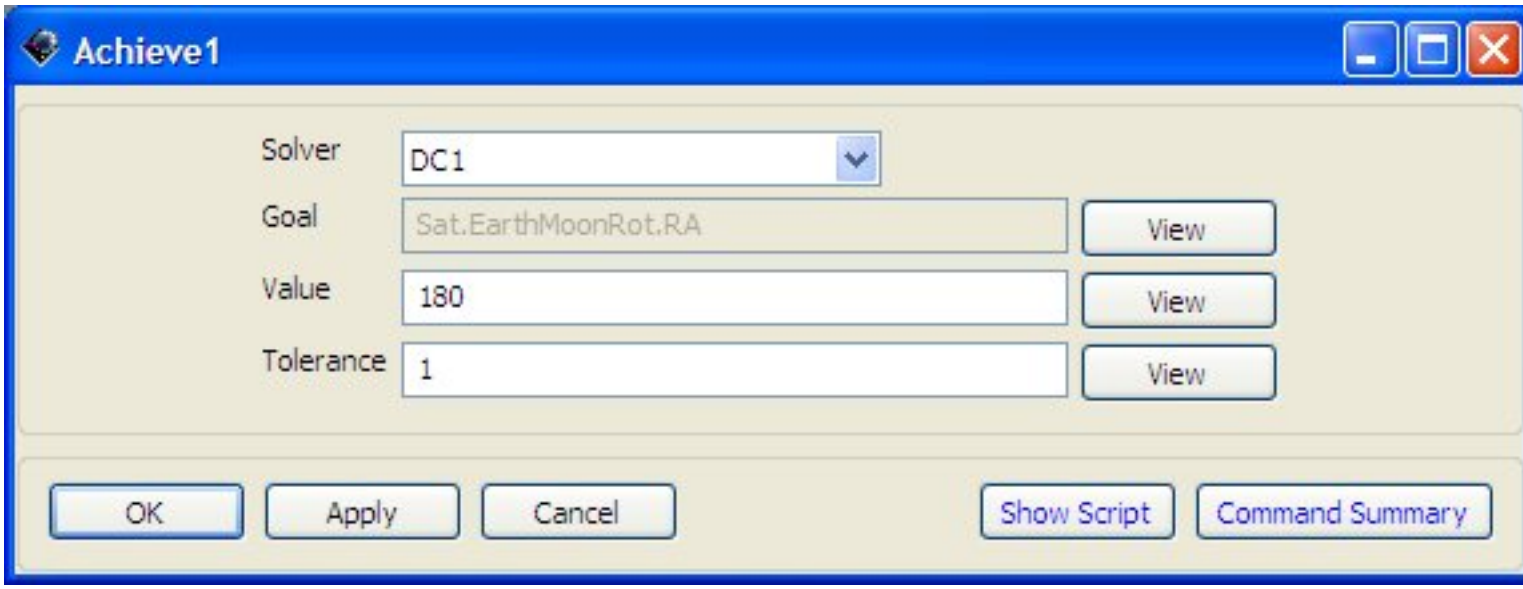


Figure: Achieve1 Achieve Command

Sat Equations

- Equation10 passes back the Sat its initial conditions (which had been stored in InitSat).
- Equation11 through Equation13 give to Sat the RAAN and AOP values found in the first targeting sequence (i.e. in Equation8 and Equation9), and give it the originally given Epoch.
- Equation14 is used to store all these variables, including those from the first targeting sequence (in InitSat, for later use).
- Equation10: Sat=InitSat
- Equation11: Sat.RAAN= RAAN
- Equation12: Sat.AOP = AOP
- Equation13: Sat.A1Epoch = Epoch
- Equation14: InitSat = Sat

Other Equations

- Equation15 in this next sequence sets the desired energy (variable DesiredEnergy) to the value necessary to take Sat near the Lagrange Point L2. This is not precise and is not the final value of the energy of the spacecraft in orbit. (This energy was calculated using hand calculations not shown here.)
- Equation16 is an initial guess of the B·T of the Moon required to achieve this energy.
- Equation17 is setting the counter variable, c, equal to 0.
- Equation18 is the lower bound on the guess of B·T needed.
- Equation19 similarly is the upper bound guess.

- Equation18 and Equation19 combine for a result in that you are saying that the answer to what B·T is needed can be found between these two values. If the value is not contained between these two, there is no possible way GMAT will find the correct answer.
- Equation20 sets an upper bound on the amount of error tolerance in the calculations.
- Equation15: DesiredEnergy = -0.2076893509448376
- Equation16: B·T = 10000
- Equation17: c = 0
- Equation18: LB = 10000
- Equation19: UB = 11000
- Equation20: Error = 1e10

Section 1: Bisection Method

- This section begins with the While statement. The While conditions on Error cause it to leave the While loop once a certain accuracy has been achieved. Variable c handles the maximum amount of iterations desired (equal to 20).
- Next is the counter condition as value c passes through the loop.
- In the next statement, on the first iteration the value LB (Lower Bound) is set first equal to B·T. Similarly for the value UB (Upper bound) in the following statement.
- The last equation in this sequence puts Sat back at its initial conditions, by making its elements equal to those of InitSat. While this isn't necessary on the first iteration since it's already been done, the following iterations will need this in order to reset the location of Sat.
- Create While Error >= .000001 & c < 20
  - Create Equation c = c + 1
  - Create If c==1
    - Create Equation B·T = LB
  - Create If c==2
    - Create Equation B·T = UB
  - Create Equation Sat = InitSat

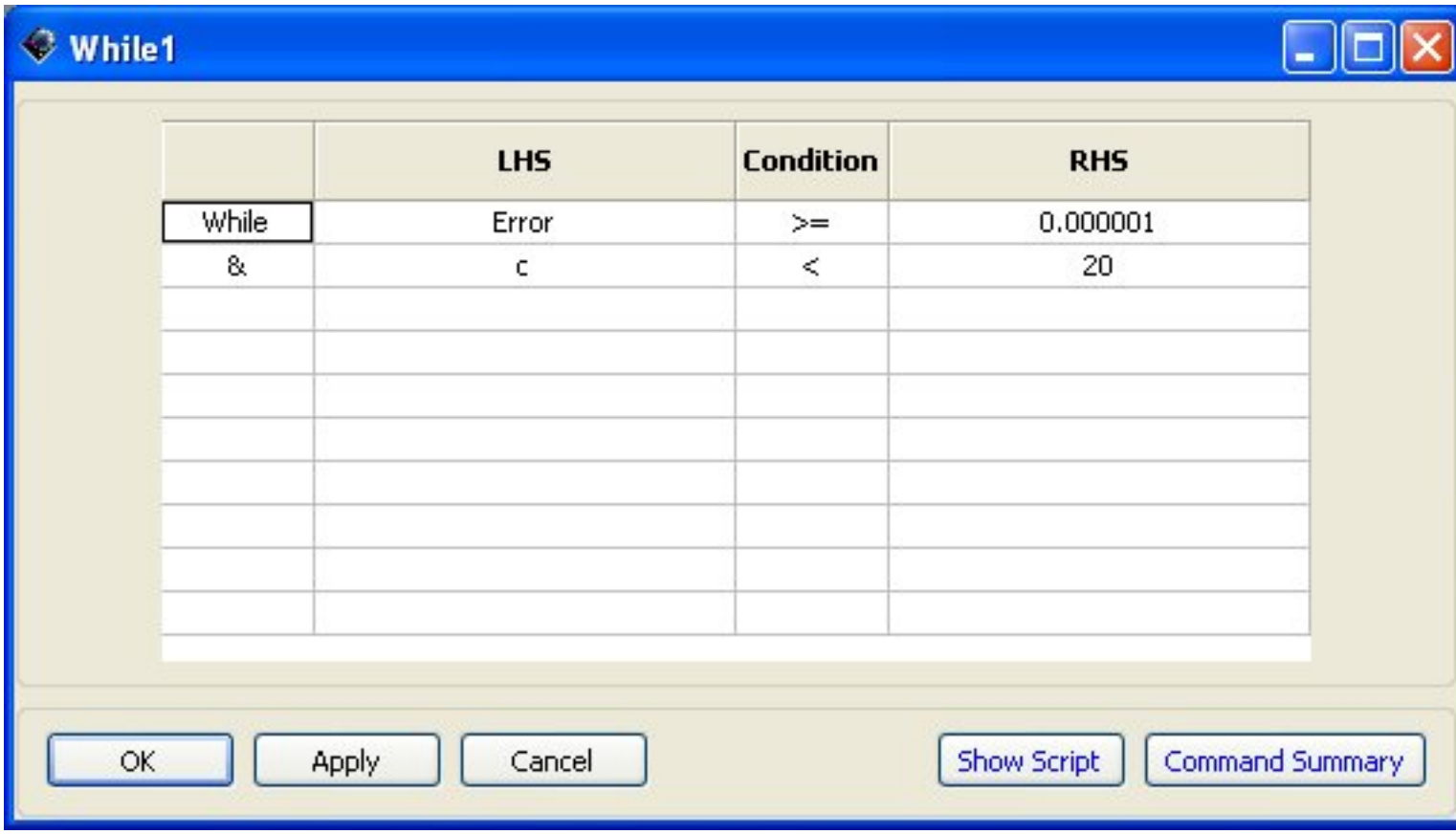


Figure: While

While Loop: Target 2

- This target sequence starts with the spacecraft in the initial conditions, in orbit about the Earth. Similar to the second target sequence in the Lunar Transfer Sample Mission, this sequence is intended to take data from the first target sequence and use thrust in two directions in order to more precisely target a desired point in the B-plane.
- Recall that a #V=3.1 km/s has already been applied. Value P1 is intended to refine the TOI burn; for this, #V is set to #V=P1+TOI.
- The value NearEarthProp is used first since its calculation is more accurate and quick; here Sat is near the Earth, with Earth being the Primary body. At a point between the Earth and Moon, sometime after a day in transit, the Moon is chosen as the Primary body (using the value NearMoonProp).
- Sat is thus propagated until Moon's periapsis. The procedure then checks if the Achieves conditions were reached. The purpose of expression Sat.MoonInertial.B·T=B·T is see what burn would be required to reach the guessed value of B·T.
- Vary P1.V
  - Initial Guess = .01, Perturbation=0.00001, MaxStep=.05, Lower =-360, Upper=360
- Vary P1.B
  - Initial Guess = .001, Perturbation=0.00001, MaxStep=.05, Lower =-360, Upper=360

- Propagate using NearEarthProp for 1 day
- Propagate using NearMoonProp until Periapsis with the Moon
- Achieve Sat.MoonInertial.B·T=B·T (the variable) with tolerance of 0.0001
- Achieve Sat.MoonInertial.B·R= 0 with tolerance of 0.0001.

While Loop: Propagate 4 and 5

- Of the two propagators, the first takes the satellite from the Moon's periapsis first to the boundaries of a predefined box (region) centered around the L2 point. The outer boundaries of this box are 50,000 km in both directions on the X-axis.
- The second propagator takes the satellite from the outer boundaries of the box above, and propagates it until it leaves a second (larger) box, whose outer boundaries are 1,000,000 km in both directions on the X-axis.
- The stop tolerance is chosen so that if the satellite moves too quickly, it will go past the stopping condition and cause the program to stop.
- Propagate using NearMoonProp with following conditions
  - ESML2Centered.X = -50000
  - ESML2Centered.X = 50000
  - StopTolerance = 0.0001
- Propagate using NearMoonProp with following conditions
  - ESML2Centered.X = -1000000
  - ESML2Centered.X = 1000000
  - ElapsedDays = 160
  - StopTolerance=0.0001

While Loop, Section 2: Bisection Method

- This section starts the Bisection method proper. The first equation here calculates the difference between the energy defined by the (guessed) B·T and the energy estimated so as to place the satellite near an L2 orbit. The first If statement, used for the first iteration, sets FLow equal to the value EnergyError that corresponds to the Lower Bound try for B·T. Similarly, the second If statement sets FHigh to the EnergyError value that corresponds to the Upper Bound try for B·T.
- The third If statement is for the third iteration - it checks if the initial boundaries of the problem do indeed bound the problem's solution. In this statement, variable Prod is set equal to the product of FHigh and FLow. Another If statement stops the program when Prod is positive. This is because Prod being positive means that both FHigh and FLow are positive or both are negative - either case means that they do not bound the solution.
- The solution exist when FLow and FHigh are both equal to 0, meaning that the EnergyError was 0. For the Bisection method to work (it should work every time, no matter how large the given range box is), the only requirement on the initial guesses is that the expected solution is between them. To check your guesses, especially if you don't get it to work the first time, it is useful to use an OpenGL Plot. This

is the purpose of the EarthView plot in this tutorial. For the first two iterations (which are your initial guesses), the one with the lower energy should take the satellite back towards the Earth after going to L2, whereas the iteration with the higher energy should take the satellite outward, away from the Earth. If you see this, it means your Bisection method should converge. The plot will also help to determine which of the energy guesses, the high or the low, needs to change. If both take the satellite back towards the Earth, your lower bound guess was too high. If both take the satellite away from the Earth, it means that your upper bound guess was too low.

- Create Equation EnergyError = Sat.Energy – DesiredEnergy
- Create If c==1
  - FLow = EnergyError
- Create If c==2
  - FHigh = EnergyError
- Create If c==3
  - Create Equation Prod = Flow * FHigh
  - Create If Prod > 0
    - Create Stop

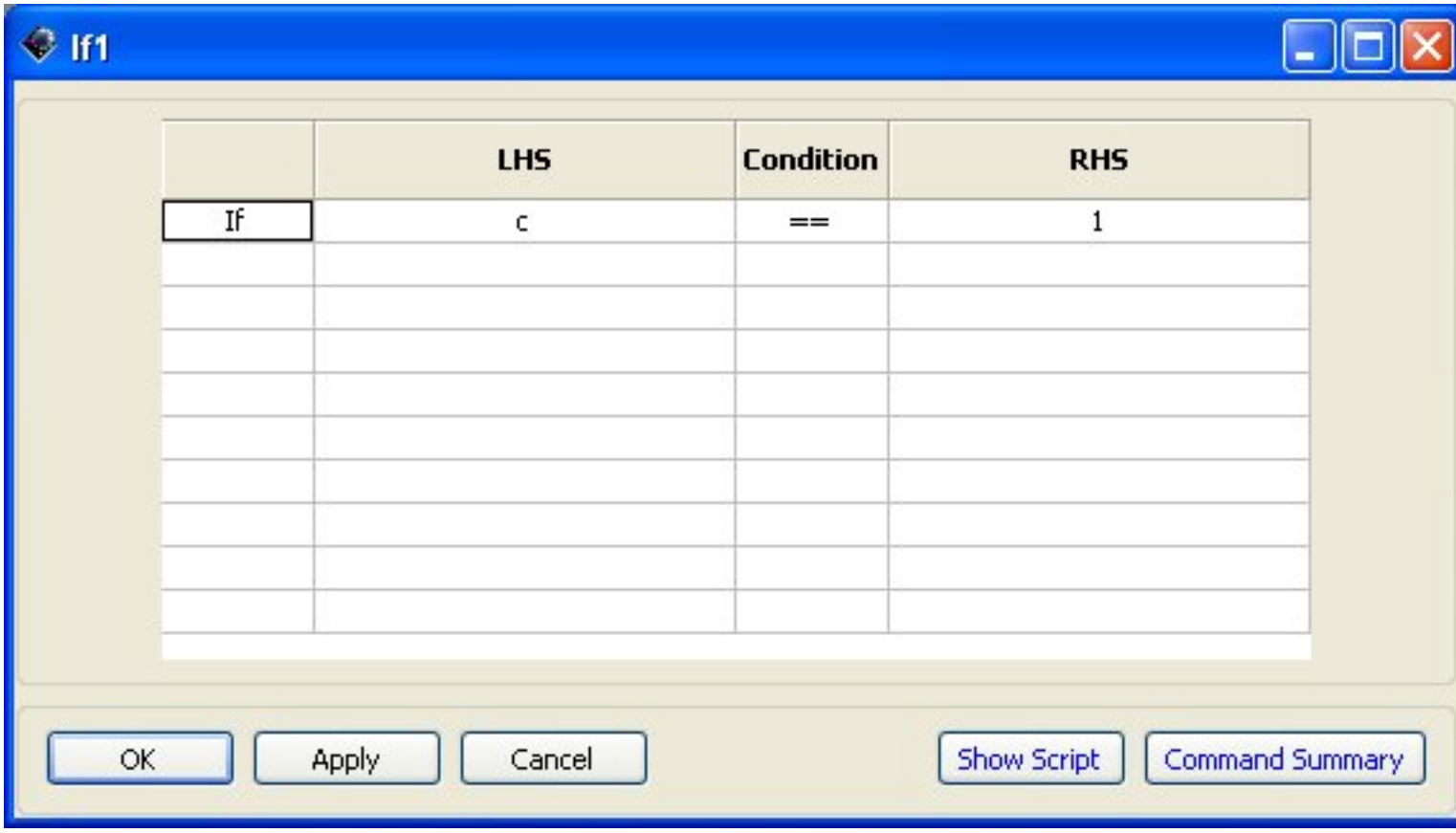


Figure: If statement

While Loop, Section 3: Bisection Method, cont.

- This section finishes the While sequence. The first If statement here is the same for the iterations that follow the first two iterations (for the lower and the upper energy bounds). Again, Prod is the product of FLow and EnergyError. If the energy error has the same sign as FLow, it means it is erring in the same manner as FLow. That is, if Prod is positive, the lower bound becomes B·T. This cuts the range that contains the answer in half. If Prod is negative - FLow and EnergyError have different signs - then it is erring on the same side as FHigh, and then B·T becomes the new upper bound.

- The next equation takes the absolute value of EnergyError.

- The last equation takes the average of the latest upper and lower bound values found in this iteration. This average is equated to B·T and will become the initial guess for the next iteration. As the iterations progress, the space box (range) is cut in half repeatedly until the solution is contained within a range so small that the required thrust to reach that B·T (and thus that energy) is known to a fair extent.

- Create If c >2
  - Create Equation Prod = FLow * EnergyError
  - Create If Else
    - If Prod < 0
      - Create Equation UB = B·T
    - Else
      - Create Equation LB = B·T
  - Create Report Data with following variable values shown: Error, UB, B·T, LB, EnergyError, Sat.Energy, and DesiredEnergy
  - Create Equation, and make the absolute value of EnergyError equal to Error
  - Create Equation B·T = (UB + LB) / 2

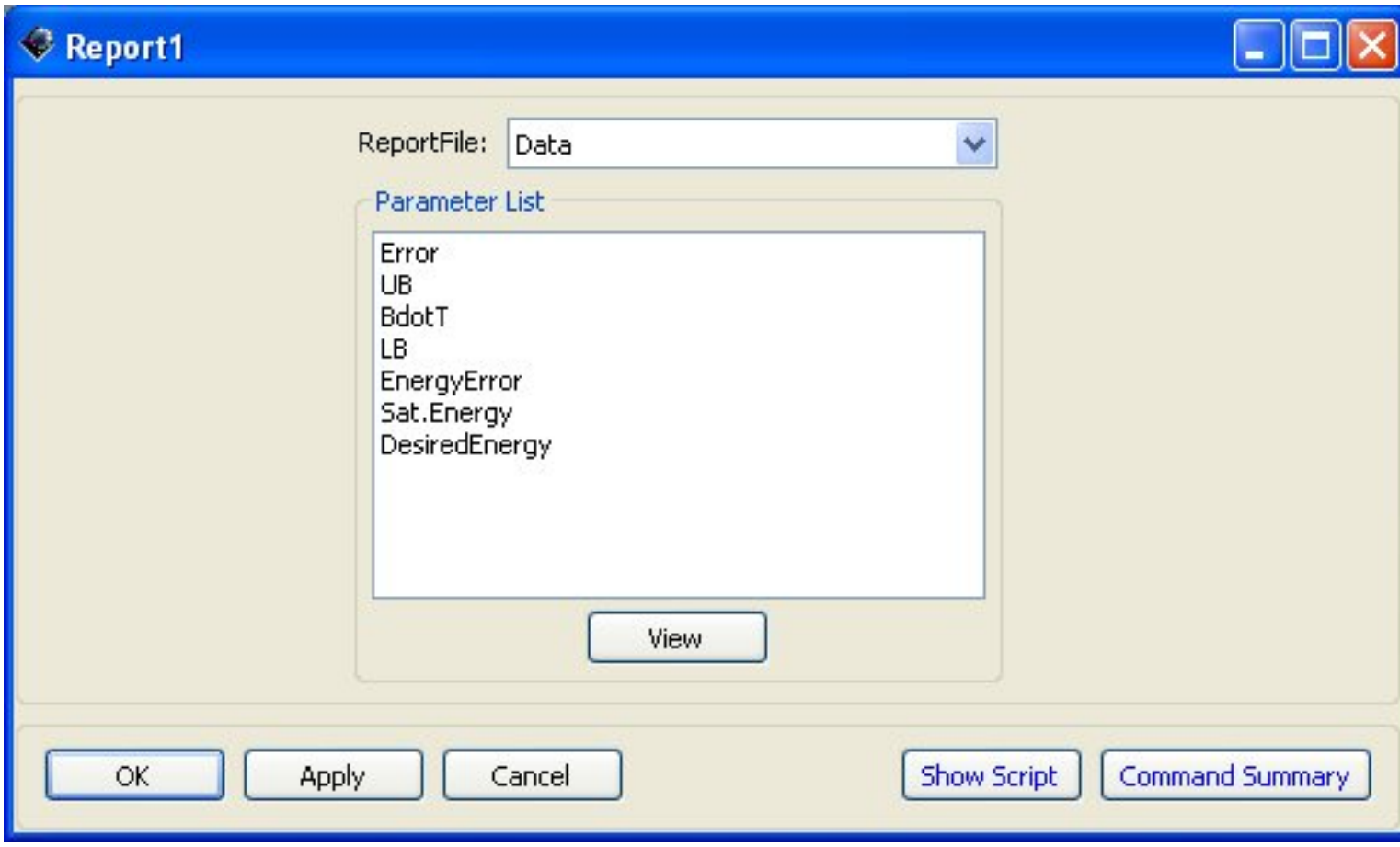


Figure: Report Command

Intermediate Stage

- This sequence does the following: The first equation sets the spacecraft back to its initial condition. The next item adds to the TOI burn P1, thus placing the spacecraft where it needs to be. These conditions are then stored again into InitSat. The fourth line of the sequence turns on EarthView2 and AboveL2. (These were turned off in previous sequences, so that the finer changes in orbit about the L2 could be seen unobstructed.)
- The next two equations set up the space box (range) size desired about L2.
- Equation34: Sat = InitSat
- Maneuver P1
- Equation35: InitSat = Sat
- Toggle2: turn on EarthView2 and AboveL2
- Equation36: LowerBoxSize = -500000
- Equation37: UpperBoxSize = 500000

Final Targeting Sequence

- This targeting sequence further adjusts the initial burn while the spacecraft is still in a Low Earth Orbit. This burn is to achieve a small negative velocity away from the Earth at the L2 point. After the initial burn is varied, the satellite is propagated. It is propagated again in a similar manner to the last sequence: first, for a day using the Earth propagator, then toward the lunar periapsis, and then for several more days. Lastly, it is propagated again for 160 more days, to produce enough information to estimate the orbit's quality.
- In the For loop, the first two equations make the box (range) continually smaller. (Note: in testing, only one iteration was necessary.) If the satellite is found to be within the boundaries of the desired box, the spacecraft is propagated until it leaves the box - or, as a stop limit, if it stays there for 220 days.
- Create Target sequence
  - Vary P2.V
    - Perturbation of -.00000001
    - Max Step of .00000001
    - Lower bound=-.000005
    - Upper bound= .0000001
  - Maneuver – apply burn P2 to Sat
  - Propagate using NearEarthProp for 1 day
  - Propagate using NearMoonProp until Luna Periapsis
  - Propagate using NearMoonProp for 8 days
  - Propagate using NearMoonProp for 160 days
  - Send Sat.ESML2Centered.X position and Y position to Report Data
  - Create For loop, with counter's values from 1 to 1 in increments of 1
    - Create Equation LowerBoxSize = LowerBoxSize - 50
    - Create Equation UpperBoxSize = UpperBoxSize +50
    - Create If statement
      - If Sat.ESML2Centered.X > LowerBoxSize
      - & Sat.ESML2Centered.X < UpperBoxSize
      - & Sat.ESML2Centered.Y < UpperBoxSize
      - & Sat.ESML2Centered.Y > LowerBoxSize
    - Propagate using NearMoonProp
      - For 220 days
      - ESML2Centered.Y=0

    - ESML2Centered.Y = LowerBoxSize
    - ESML2Centered.X = LowerBoxSize
    - ESML2Centered.X = UpperBoxSize
    - ESML2Centered.Y = UpperBoxSize
    - StopTolerance of .0001
- Achieve ESML2Centered.VX = -.001 with a tolerance of .001

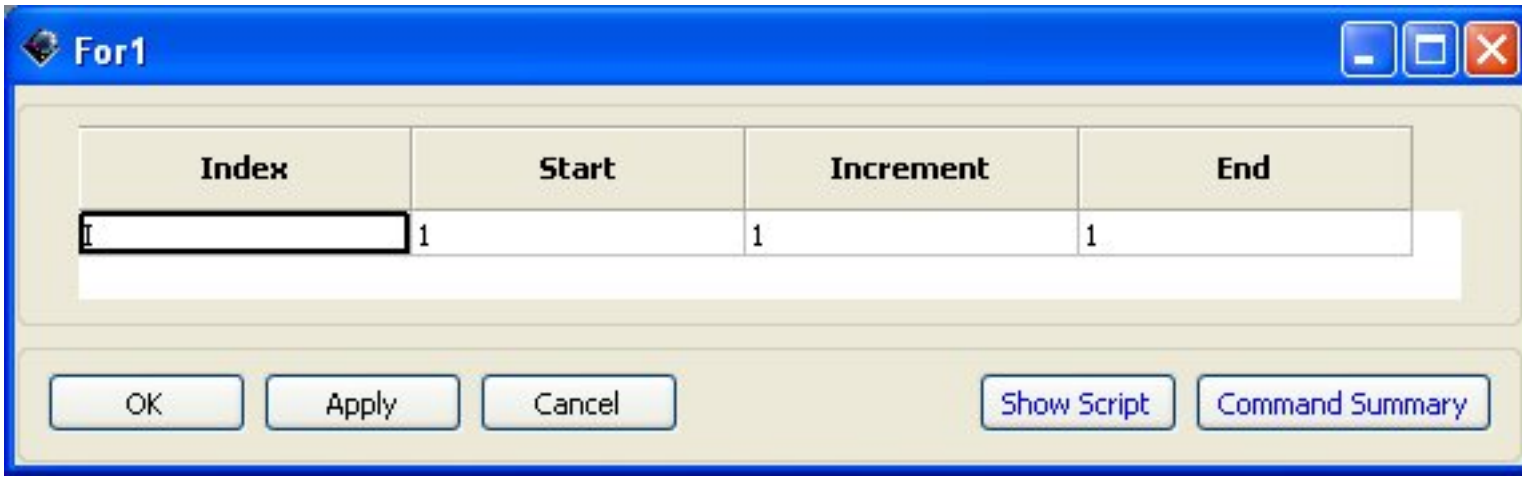


Figure: For loop

The resulting view of Resource, Mission, and Output trees

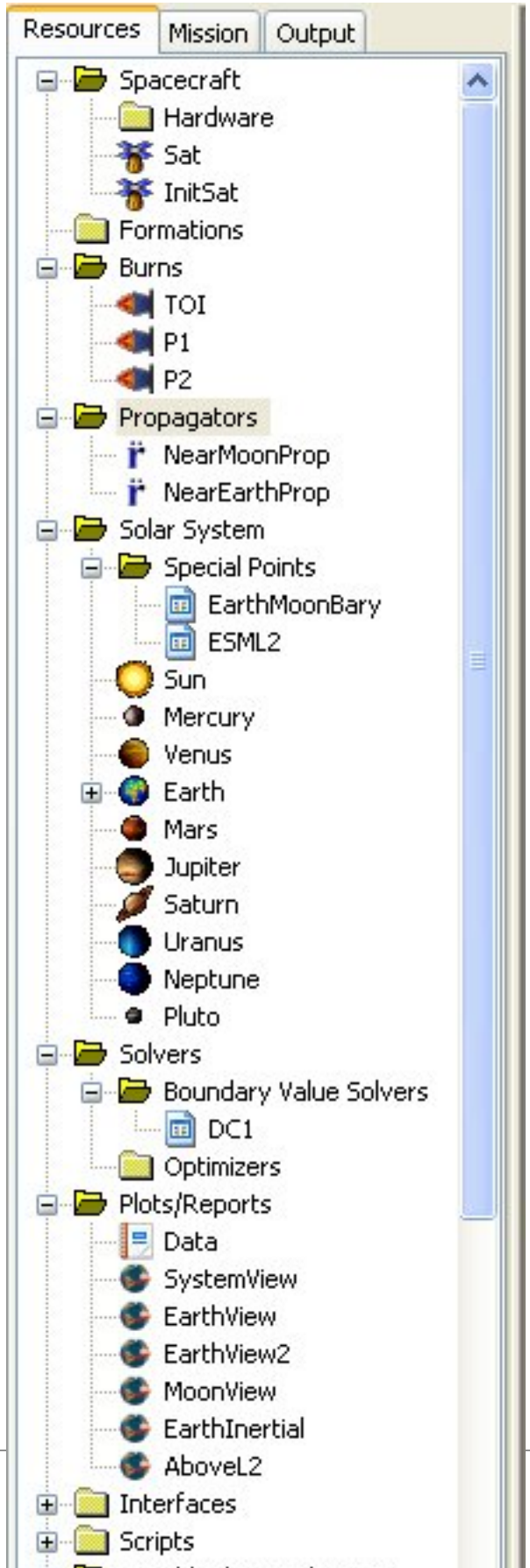


Figure: Resource tree

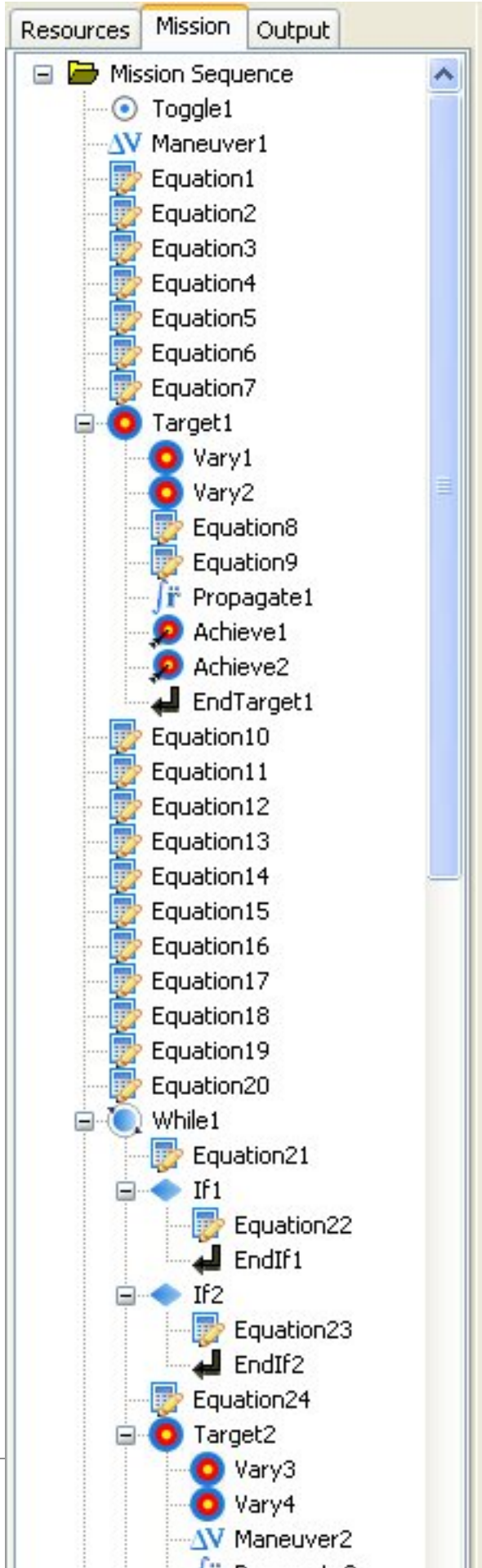
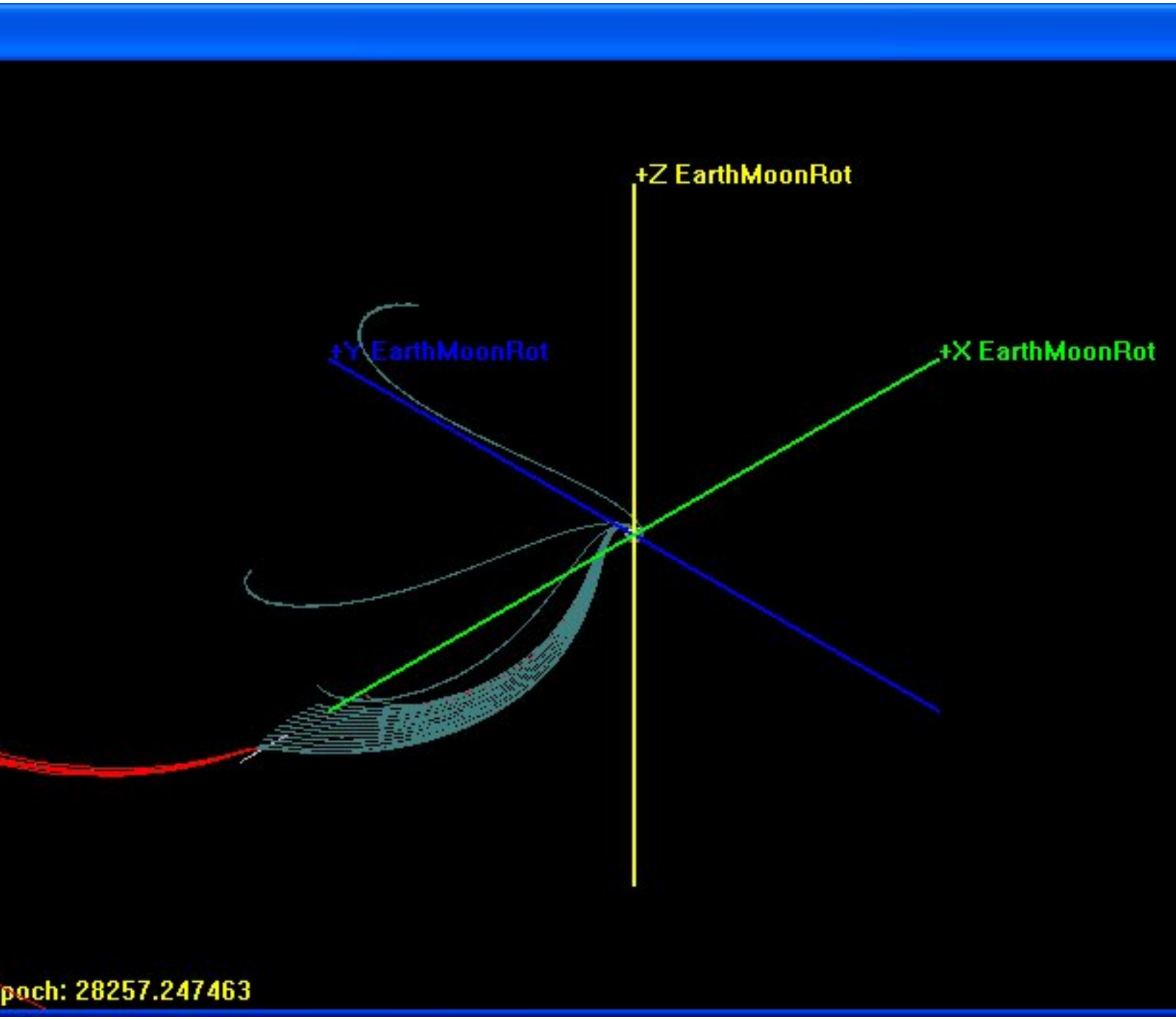

Resources
Mission
Output
Mission Sequence
Toggle1
Maneuver1
Equation1
Equation2
Equation3
Equation4
Equation5
Equation6
Equation7
Target1
Vary1
Vary2
Equation8
Equation9
Propagate1
Achieve1
Achieve2
EndTarget1
Equation10
Equation11
Equation12
Equation13
Equation14
Equation15
Equation16
Equation17
Equation18
Equation19
Equation20
While1
Equation21
If1
Equation22
EndIf1
If2
Equation23
EndIf2
Equation24
Target2
Vary3
Vary4
Maneuver2

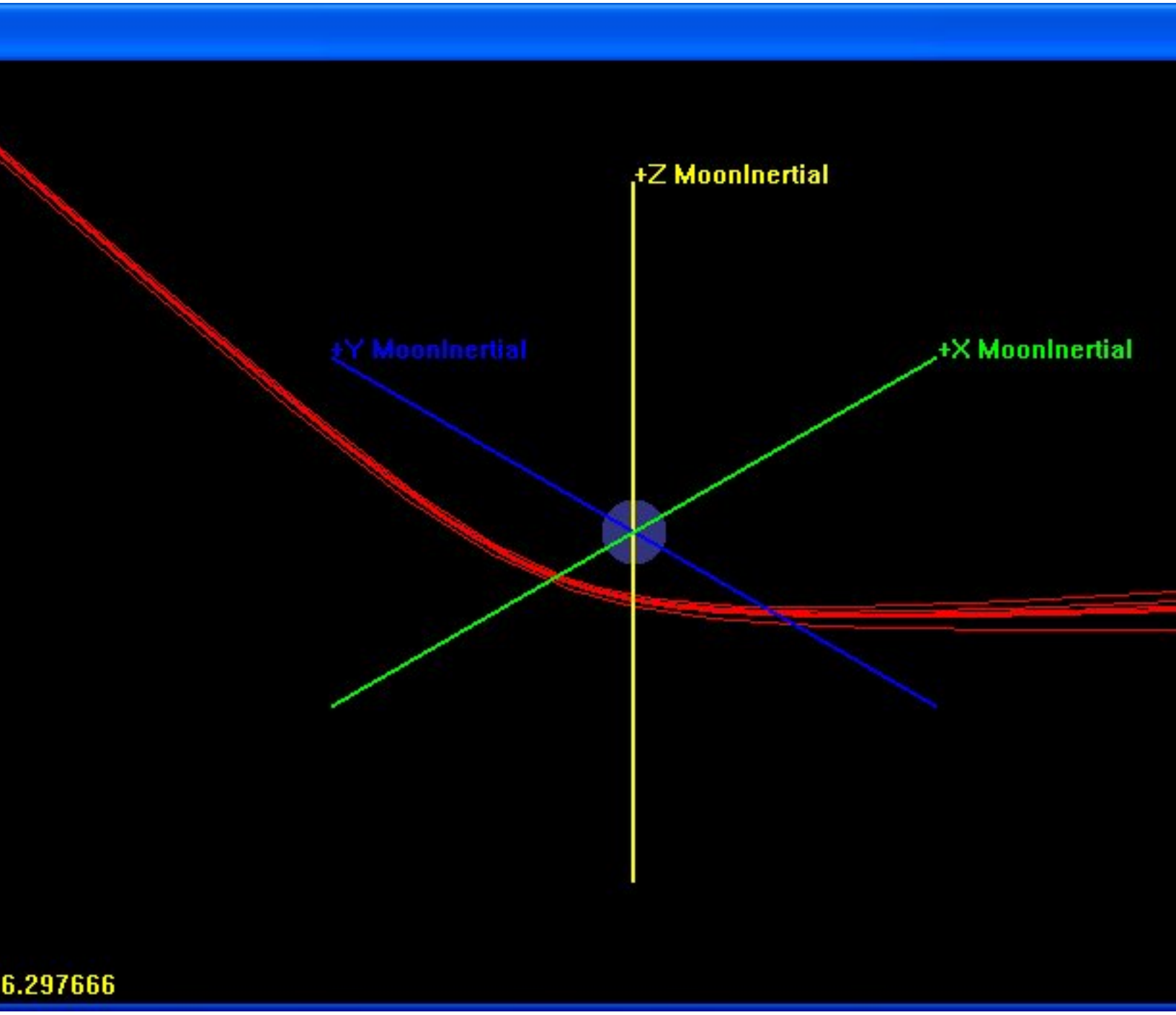
Figure: Mission and Output trees

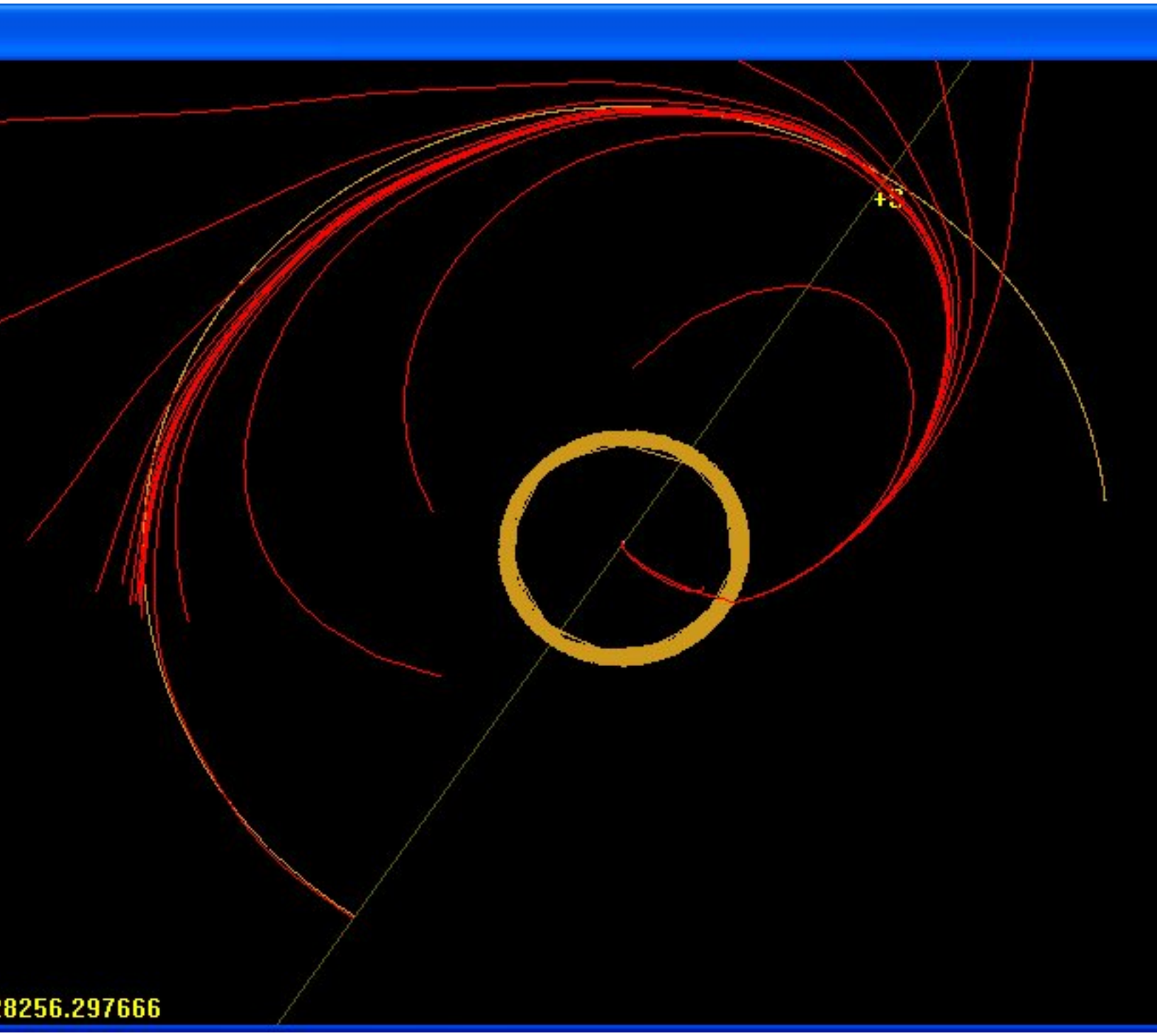
Running the Mission

Resource Mission and Output Trees

You have now configured the mission - it is time to run it and see the results. First, make sure you save the changes you've made. Then click the green arrow on the tool bar to run the mission. The screen capture below summarizes what you should see if you completed the mission correctly. In the plot SystemView you should see the first targeting sequence tracking. Plot MoonView shows the Moon swingbys. Plot EarthInertial shows the satellite getting closer and closer to tracing out the line that L2 follows. Plot EarthView1 shows the iterations getting closer and closer to wrapping around point L2 and orbiting it. Plots EarthView2 and AboveL2 show the iterations further refining the calculations. You can zoom in on point L2, to better see the events around this area. The Report presents a variety of important data at crucial points during the mission,


+Z EarthMoonRot
+Y EarthMoonRot
+X EarthMoonRot
poch: 28257.247463


+Z MoonInertial
+Y MoonInertial
+X MoonInertial
6.297666


8256.297666

: 28256.297666

7.247463

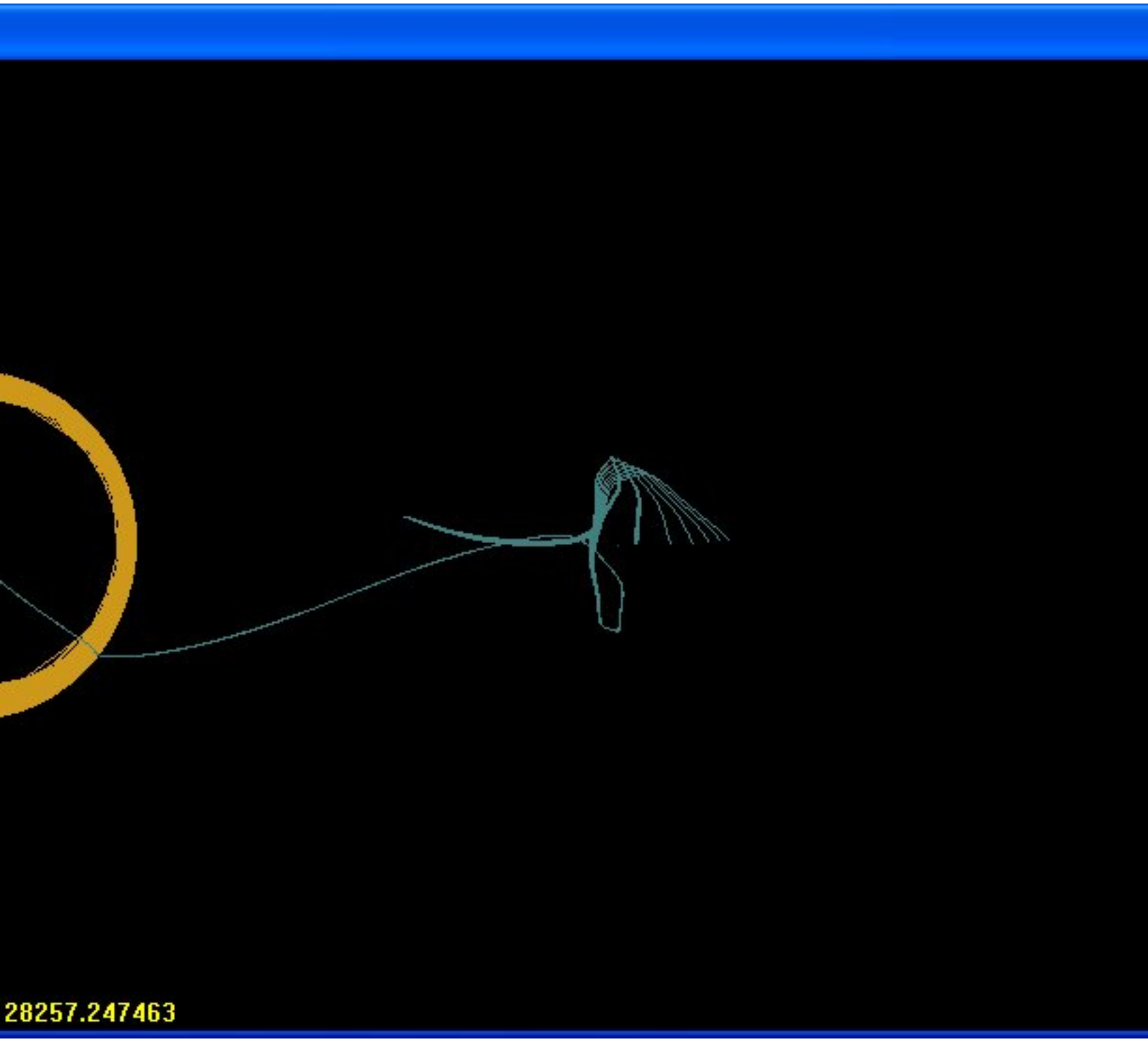


Figure: Output

Chapter 18. ACE Station Keeping

This tutorial was started from the default settings in the June 9, 2008 build of GMAT.

Objective and Overview

ACE, or Advanced Composition Explorer, studies the Sun and is one of only a few spacecraft that resides at the Sun-Earth L_1 Libration Point in a Lissajous Orbit that is quasi-periodic. This tutorial will show the method in which a spacecraft may be station kept at this point in space. Put simply, a number of orbits about the Libration point will be simulated, with each being corrected slightly by small maneuvers due to the unstable nature of Sun-Earth L_1.

Download the script file: ACEStationKeepingTutorial.script [scripts/ACEStationKeepingTutorial.script]

Prerequisites

- Basic Understanding of Libration Point Station Keeping
- Tutorials:
  - Creating and Propagating a Spacecraft
  - Designing a Hohman Transfer
- Objects:
  - Spacecraft Object
  - Impulsive Burn Object
  - Propagator Object
  - Barycenter Object
  - Libration Point Object
  - Differential Corrector Object
  - OpenGL Plot Object
  - ReportFile
  - Variable Object
  - Coordinate System Objects
- Commands:
  - Report Command
  - Equation Command
  - For Command
  - Target Command

- Vary Command
- Maneuver Command
- Propagate Command
- Achieve Command

Mission Description

- Objective: Maintain an orbit about the Sun-Earth L_1 Libration point by performing a burn perpendicular to the direction of travel so that as the spacecraft passes through the ecliptic, its velocity relative to the Sun and Earth is zero.
- Assume:
  - A preexisting orbit with given initial conditions
  - A UTCGregorian Epoch of 09 Dec 2005 12:59:60.000
- Find: The deltaV required perpendicular to the direction of travel at each pass of the ecliptic to meet the above objective

Resource, Mission, and Output trees

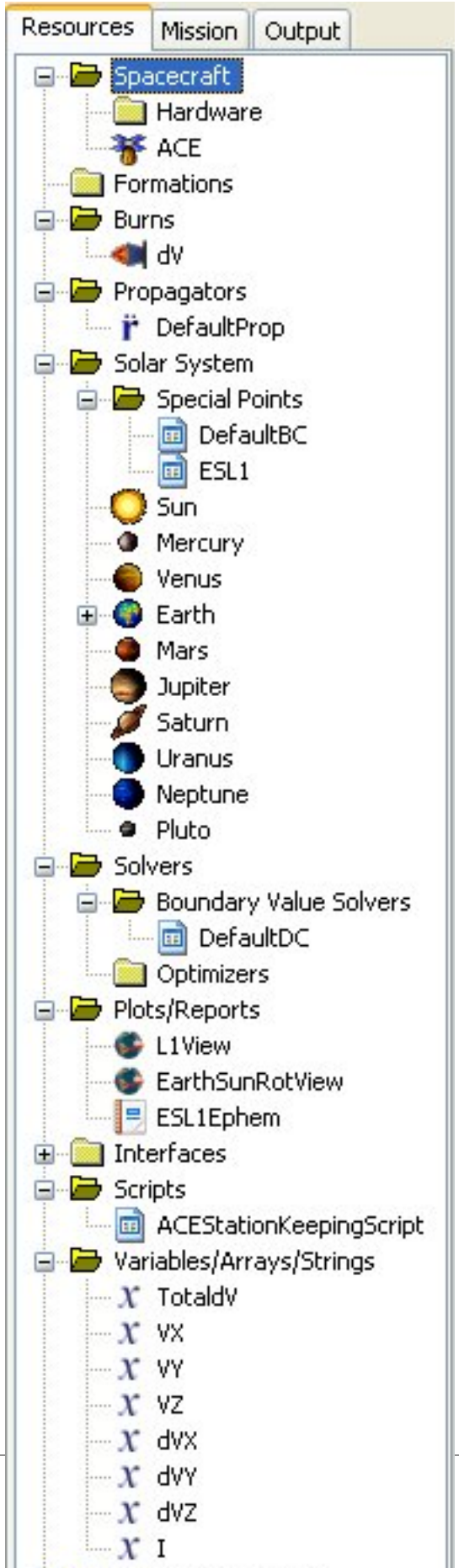


Figure: ACE Resource, Mission, and Output Trees

Creating and Configuring the Resource Tree

This section will outline exactly what resources are required to complete the mission outlined within this tutorial. One spacecraft, a single burn to control that spacecraft, and a simple propagator, will be needed. To better see and analyze the spacecraft's motions, we will also need several Plots/Reports, Variable Object, and Coordinate Systems.

Objects Required

- Spacecraft: ACE
- Burns: dV
- Propagators: defaultProp (modified)
- Solar System: DefaultBC, ESL1
- Solvers: DefaultDC
- Plots/Reports: L1View, EarthSunRotView, ESL1Ephem
- Variables: TotaldV, VX, VY, VZ, dVX, dVY, dVZ, I
- Coordinate Systems: EarthSunL1CS, EarthSunRotCS

Creating and Modifying Objects

- Spacecraft
  - Rename DefaultSC to ACE
  - Change the Epoch Format to UTCGregorian
  - Set the Epoch to 09 Dec 2005 12:59:60.000
  - Set X = -528665.5773210001, Y = -1325445.164527, Z = -525969.843578, VX = 0.240435056, VY = -0.068342458, and VZ = 0.0315715361

  Script Syntax:

```
Create ((Spacecraft Object)) ACE;
GMAT ACE.DateFormat = UTCGregorian;
GMAT ACE.Epoch = '09 Dec 2005 12:59:60.000';
GMAT ACE.X = -528665.5773210001;
GMAT ACE.Y = -1325445.164527;
GMAT ACE.Z = -525969.843578;
GMAT ACE.VX = 0.240435056;
GMAT ACE.VY = -0.068342458;
GMAT ACE.VZ = 0.0315715361;
```

Figure: ACE Spacecraft Object Object

- Propagators
  - Change the Min Step Size to 9.999999999999999e-022
  - Change the Max Step Size to 86400
  - Change the Max Step Attempts to 100
  - Remove any Primary Bodies
  - Add Sun, Earth, and Luna as Point Masses

  Script Syntax:

```
Create ForceModel DefaultProp_ForceModel;
GMAT DefaultProp_ForceModel.CentralBody = Earth;
GMAT DefaultProp_ForceModel.PointMasses = {Sun, Earth, Luna};
Create Propagator DefaultProp;
GMAT DefaultProp.FM = DefaultProp_ForceModel;
GMAT DefaultProp.MinStep = 9.999999999999999e-022;
GMAT DefaultProp.MaxStep = 86400;
GMAT DefaultProp.MaxStepAttempts = 100;
```

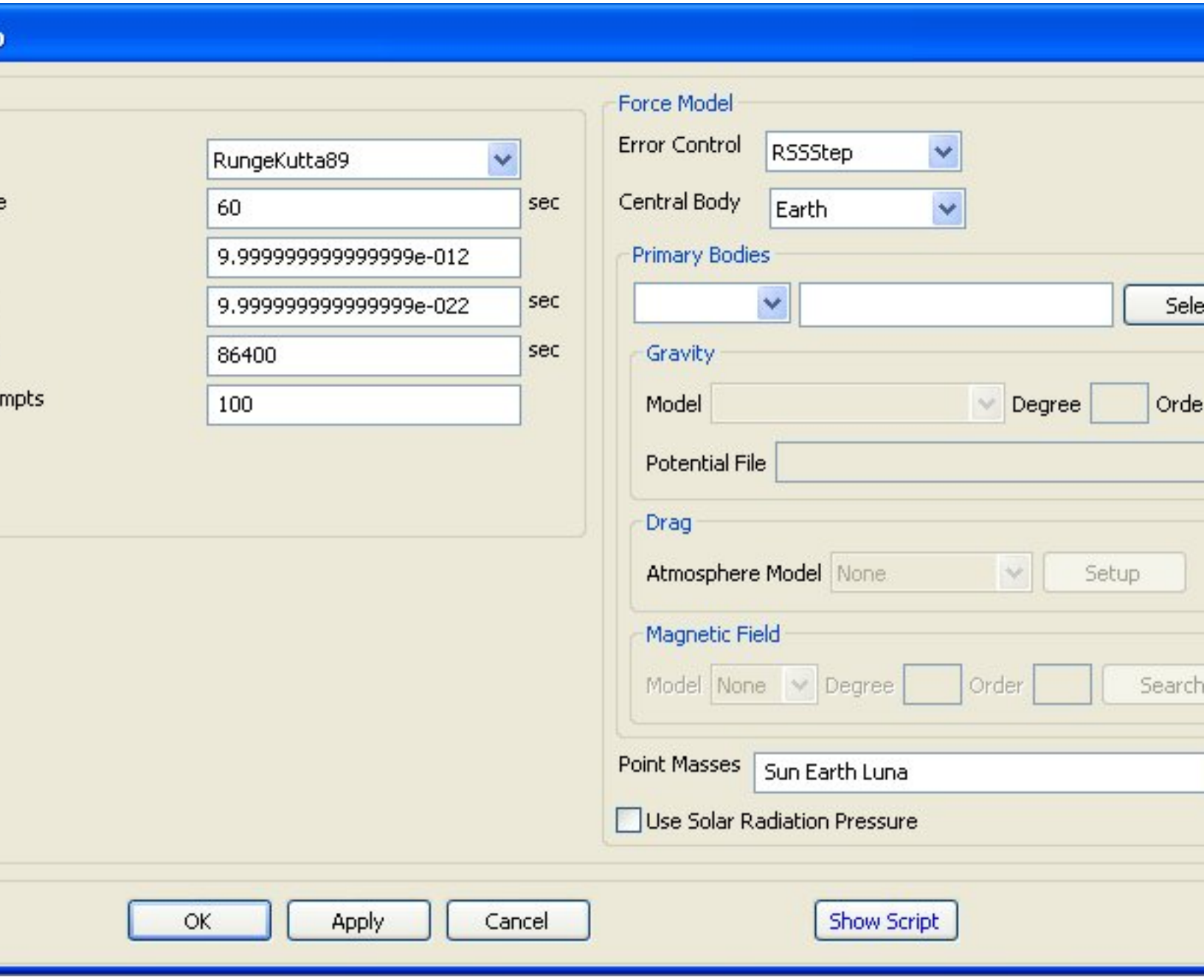


Figure: DefaultProp Propagator Object

- Solar System
  - Add a Barycenter under Special Points and rename it DefaultBC
  - Add a Libration Point under Special Points and rename it ESL1

  Script Syntax:

  ```
  Create Barycenter DefaultBC;
  ```

```
Create LibrationPoint ESL1;
```

Figure: ESL1 Special Point Object

- Burns
  - Rename DefaultIB to dV

  Script Syntax:

  ```
  Create ImpulsiveBurn dV;
  ```

- Variables/Arrays/Strings
  - To create the necessary variables, insert the line of code below into the script just after the setup of the Coordinate System Objects

  Script Syntax:

  ```
  Create Variable TotaldV VX VY VZ dVX dVY dVZ I;
  ```

- Coordinate Systems
  - EarthSunL1CS
    - Add a new Coordinate System and name it EarthSunL1CS
    - Set the Origin to ESL1
    - Change the Axes type to Object Referenced
    - Change the Primary and Secondary Values to Sun and Earth, respectively

    Script Syntax:

    ```
    Create CoordinateSystem EarthSunL1CS;
    GMAT EarthSunL1CS.Origin = ESL1;
    GMAT EarthSunL1CS.Axes = ObjectReferenced;
    ```

```
GMAT EarthSunL1CS.Primary = Sun;
GMAT EarthSunL1CS.Secondary = Earth;
```

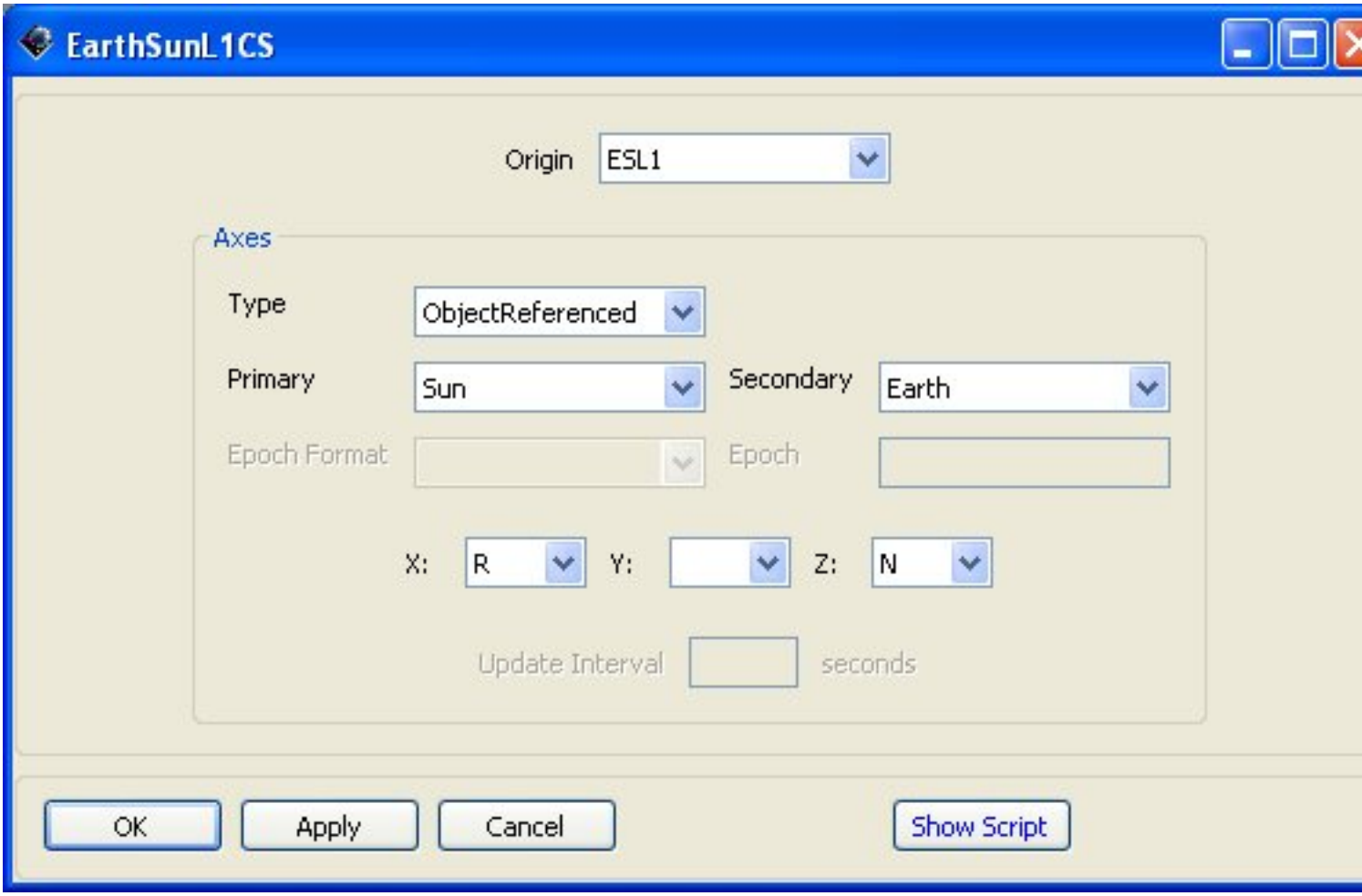


Figure: EarthSunL1CS Coordinate System Object

- EarthSunRotCS
  - Add a new Coordinate System and name it EarthSunRotCS
  - Change the Axes type to Object Referenced
  - Change the Primary and Secondary Values to Sun and Earth, respectively

  Script Syntax:

  ```
  Create CoordinateSystem EarthSunRotCS;
  GMAT EarthSunRotCS.Origin = Earth;
  GMAT EarthSunRotCS.Axes = ObjectReferenced;
  GMAT EarthSunRotCS.Primary = Sun;
  ```

- Solvers

- Create a Differential Corrector Object under Solvers > Boundary Value Solvers and name it DefaultDC
- Add the code lines below to the script under the DefaultDC section

Script Syntax:

```
Create DifferentialCorrector DefaultDC;
GMAT DefaultDC.ShowProgress = true;
GMAT DefaultDC.ReportStyle = Normal;
GMAT DefaultDC.TargeterTextFile = targeter_DefaultDC.data;
GMAT DefaultDC.Variables = {dV.V, dV.N};
GMAT DefaultDC.MaximumIterations = 25;
GMAT DefaultDC.UseCentralDifferences = false;
```

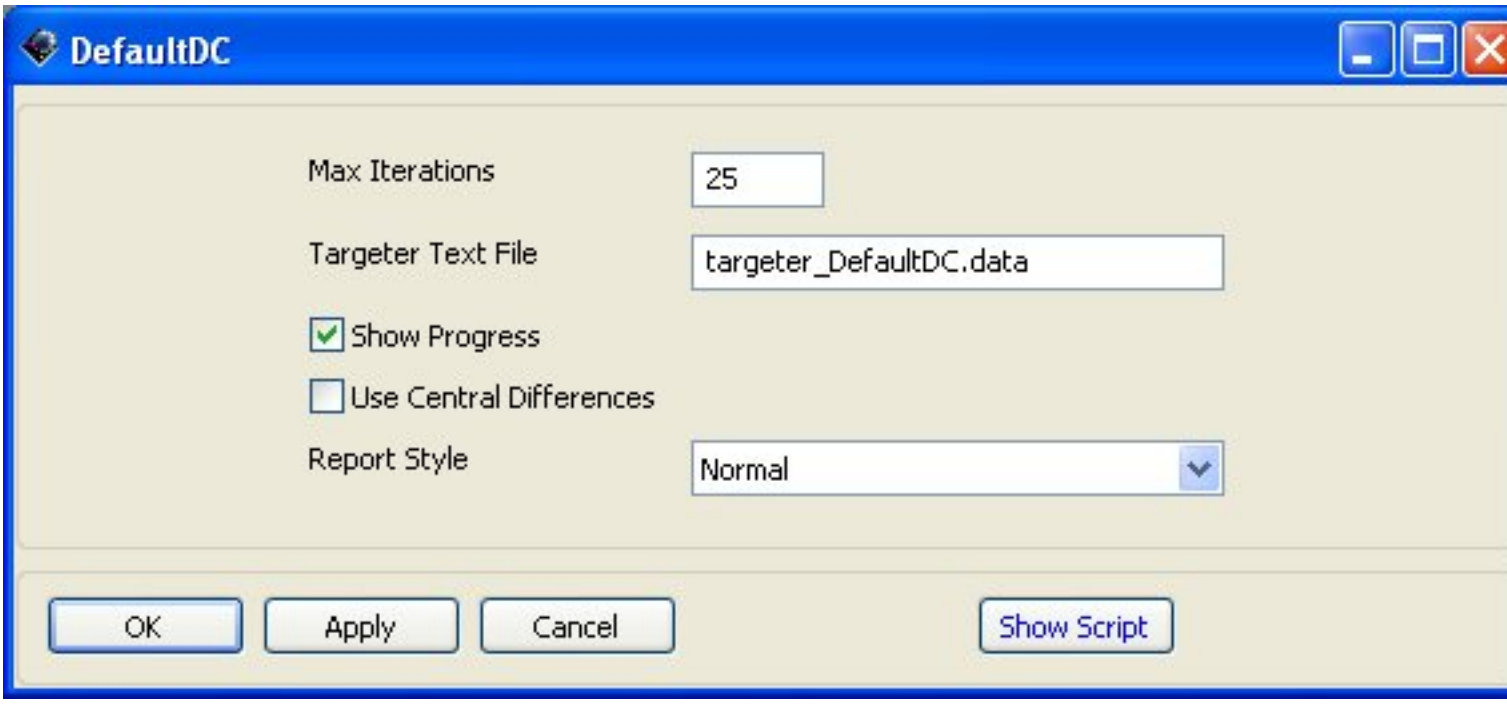


Figure: DefaultDC Boundary Value Solver Object

- Plots/Reports
  - Rename DefaultOpenGL to L1View
  - Create an OpenGL Plot Named EarthSunRotView
    - Set EarthSunL1CS as the both the View Direction and View Up Definition Coordinate System (Note: you will have to create the coordinate system outlined below before taking this step)
    - Change both the View Point Reference and View Direction to ESL1 (Note: again, this will need to be created first)
    - Change the View Scale Factor to 12
    - Uncheck DrawXY Plane and set Solver Iterations to None

Script Syntax:

```
Create OpenGLPlot L1View;
GMAT L1View.SolverIterations = None;
GMAT EarthSunRotView.Add = {ACE, Earth};
GMAT L1View.TargetColor = [ 8421440 0 ];
GMAT L1View.CoordinateSystem = EarthSunL1CS;
GMAT L1View.ViewPointReference = ESL1;
GMAT L1View.ViewDirection = ESL1;
GMAT L1View.ViewScaleFactor = 12;
GMAT L1View.ViewUpCoordinateSystem = EarthSunL1CS;
GMAT L1View.XYPlane = Off;
```

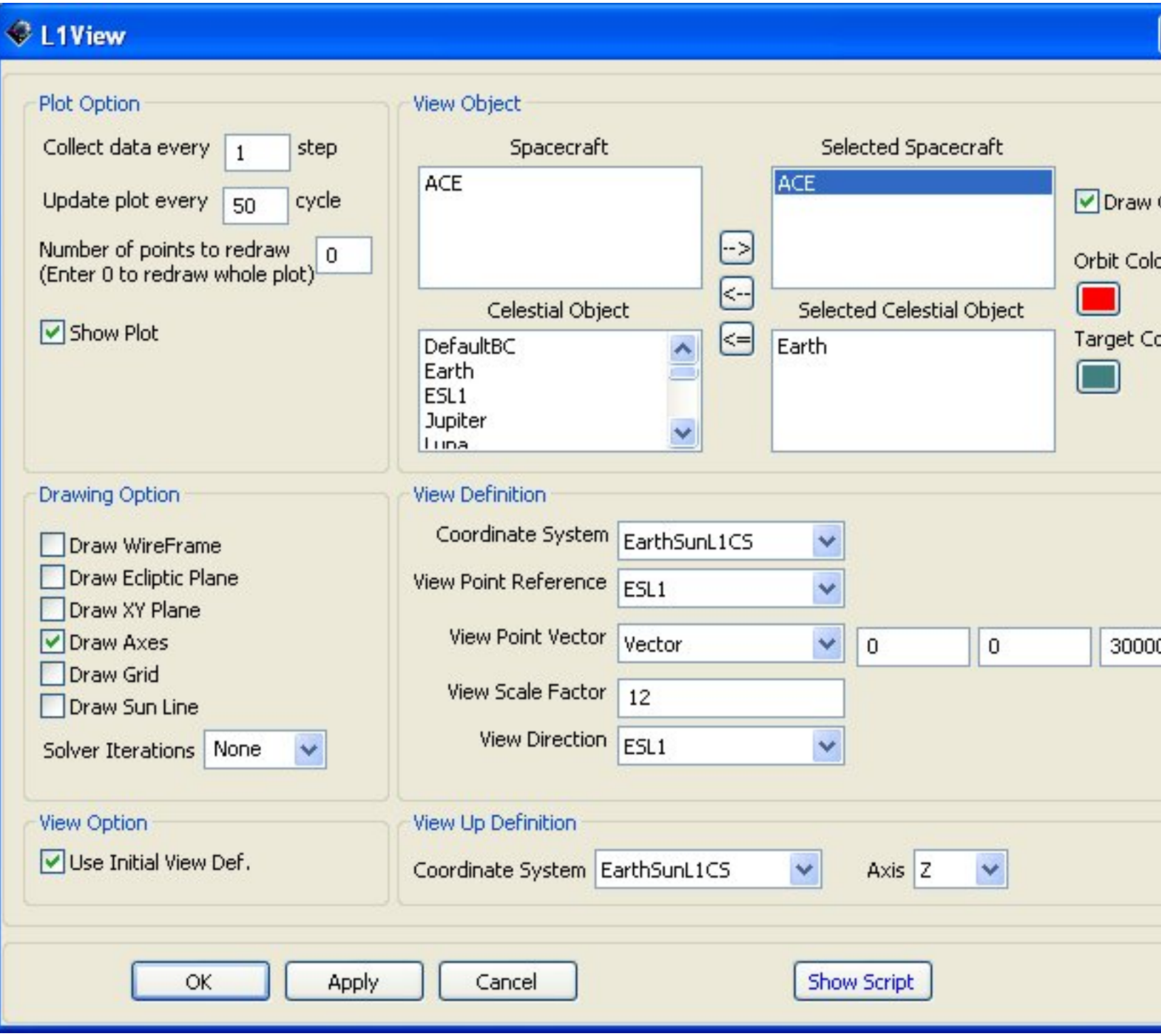


Figure: L1View OpenGL Plot Object

- Create an OpenGL Plot Named EarthSunRotView
  - Add Luna as a Celestial Object and Change its orbit color to Grey

- Switch both the View Definition and the View Up Direction Coordinate System Objects to EarthSunRotCS (Note: Don't forget to create this System first as well)
- Set the View Scale Factor to 100
- Switch the View Up Definition Axis to Y
- Uncheck DrawXY Plane and set Solver Iterations to None

Script Syntax:

```
Create OpenGLPlot EarthSunRotView;
GMAT EarthSunRotView.SolverIterations = None;
GMAT EarthSunRotView.Add = {ACE, Earth, Luna};
GMAT EarthSunRotView.OrbitColor = [ 255 32768 12632256 ];
GMAT EarthSunRotView.TargetColor = [ 8421440 0 0 ];
GMAT EarthSunRotView.CoordinateSystem = EarthSunRotCS;
GMAT EarthSunRotView.ViewPointVector = [ 0 0 30000 ];
GMAT EarthSunRotView.ViewScaleFactor = 100;
GMAT EarthSunRotView.ViewUpCoordinateSystem = EarthSunRotCS;
GMAT EarthSunRotView.ViewUpAxis = Y;
GMAT EarthSunRotView.XYPlane = Off;
```

- Create a Report Named ESL1Ephem
  - Remove all Parameters from the Parameter List
  - Select the File as being ./output/SampleMissions/Ex_ACEStationKeeping.report

  Script Syntax:

```
Create ReportFile ESL1Ephem;
GMAT ESL1Ephem.SolverIterations = Current;
GMAT ESL1Ephem.Filename = './output/SampleMissions/Ex_ACEStationKeeping.report'
GMAT ESL1Ephem.Precision = 16;
GMAT ESL1Ephem.WriteHeaders = On;
GMAT ESL1Ephem.LeftJustify = On;
GMAT ESL1Ephem.ZeroFill = Off;
GMAT ESL1Ephem.ColumnWidth = 20;
```

Creating and Configuring the Mission Sequence

Image of Mission Tree

Now that we have all the resources at out disposal to complete this mission, we will go ahead and start building the mission tree. ACE's mission if fairly straight forward at the level we will be dealing with. As stated before, the primary goal here is to allow the spacecraft seven iterations, or half orbits, and at the beginning of each one, calculate the deltaV required in the perpendicular-to-travel direction required at the ecliptic to ensure no velocity towards or away from the Sun or Earth. Basically, the spacecraft must not have any movement between these two bodies as it passes through the plane on which the Sun and Earth both lie. The entire Mission Sequence is relatively short and is given below in its entirety.

This particular mission tree may be a bit more complex then those you have previously dealt with but it is only due to the several equations and reports that it looks more complicated than it actually is. This part of the tutorial will go through the mission sequence step by step and at the end you will have a functional setup.

- Delete the initial Propagate
- Report1:
  - Append this to the Mission Sequence Folder
  - Remove ACE.A1ModJulian from the parameters list
  - Add ACE.EarthSunL1CS.X, Y, Z, VZ, VY, and VZ to the parameters list
- Equation1: Set to TotaldV = 0
- For1: Change the End parameter to 7
  - Target1: (Note: In the 6-16-2008 build of GMAT, and for several builds thereafter, the targeter requires an extra parameter. If you are working with a version soon after 6-09-2008, use the line "Target Command DefaultDC {ExitMode = SaveAndContinue};" instead of the normal target line in the script)
    - Vary1
      - Set the variable to dV.B
      - Set the initial value to 0
      - Set the perturbation to 1e-5
      - Set the Lower to -0.1
      - Set the Upper to 0.1
      - Set the Max Step to 0.01
    - Equation2: Set as VX = ACE.VX
    - Equation3: Set as VY = ACE.VY
    - Equation4: Set as VZ = ACE.VZ
    - Maneuver1
    - Equation5: Set as ACE.VX - VX
    - Equation6: Set as ACE.VY - VY
    - Equation7: Set as ACE.VZ - VZ
    - Equation8: Set as TotaldV = TotaldV + sqrt(dVZ^2 + dVY^2 + dVZ^2)
    - Propagate1: Set stopping condition as ACE.EarthSunL1CS.Y = 0
    - Achieve1: Set the goal as ACE.EarthSunL1CS.VX = 0 with a tolerance of 1e-005
  - Propagate2

- Report2
  - Remove ACE.A1ModJulian and put in ACE.UTCGregorian
  - Add ACE.EarthSunL1CS.X, Y, Z, VZ, VY, and VZ to the parameters list

Script Syntax:

```
Report ESL1Ephem ACE.EarthSunL1CS.X ACE.EarthSunL1CS.Y ACE.EarthSunL1CS.Z ACE.Eart

GMAT TotaldV = 0;
For I = 1:7

   Target DefaultDC;

      Vary DefaultDC(dV.B = 0, {Perturbation = 1e-5, MaxStep = 0.01, Lower = -0.1,

      GMAT VX = ACE.VX;
      GMAT VY = ACE.VY;
      GMAT VZ = ACE.VZ;

      Maneuver dV(ACE);

      GMAT dVX = ACE.VX - VX;
      GMAT dVY = ACE.VY - VY;
      GMAT dVZ = ACE.VZ - VZ;

      GMAT TotaldV = TotaldV + sqrt(  dVZ^2 + dVY^2 + dVZ^2  );

      Propagate DefaultProp(ACE, {ACE.EarthSunL1CS.Y = 0});

      Achieve DefaultDC(ACE.EarthSunL1CS.VX = 0, {Tolerance = 1e-005});

   EndTarget;  % For targeter DefaultDC
   Propagate DefaultProp(ACE);

   Report ESL1Ephem ACE.UTCGregorian ACE.EarthSunL1CS.X ACE.EarthSunL1CS.Y ACE.Ear

EndFor;
```

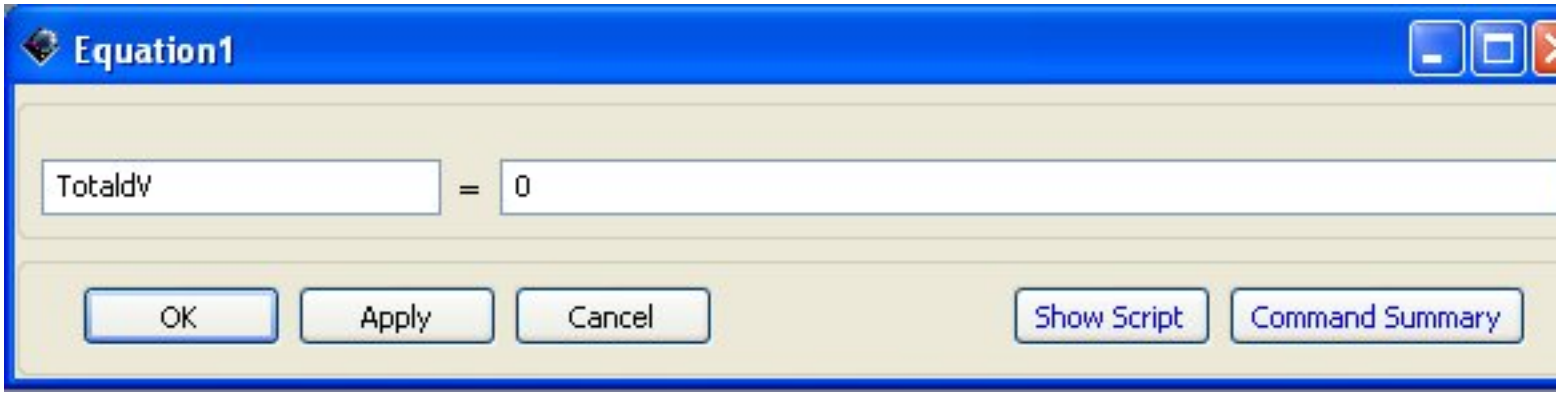


Figure: Equation1 Equation Command

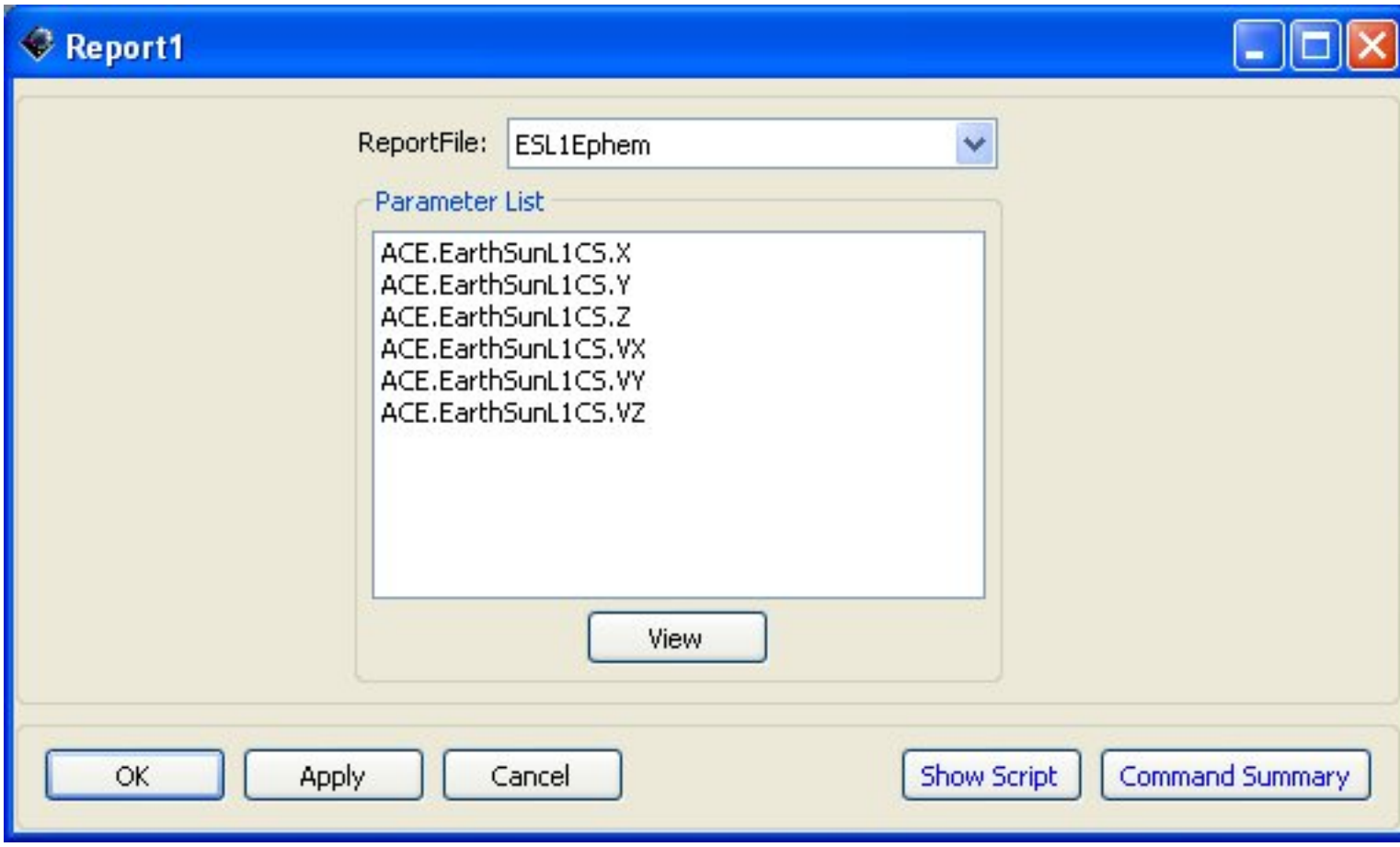


Figure: Report1 Report Command

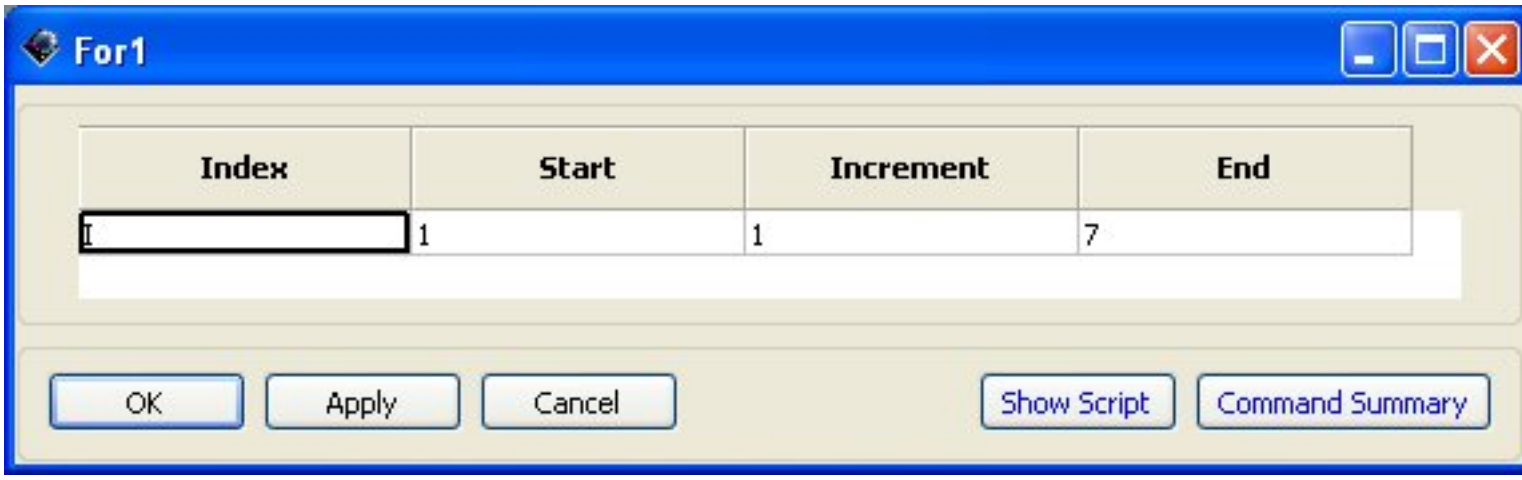


Figure: For1 For Command

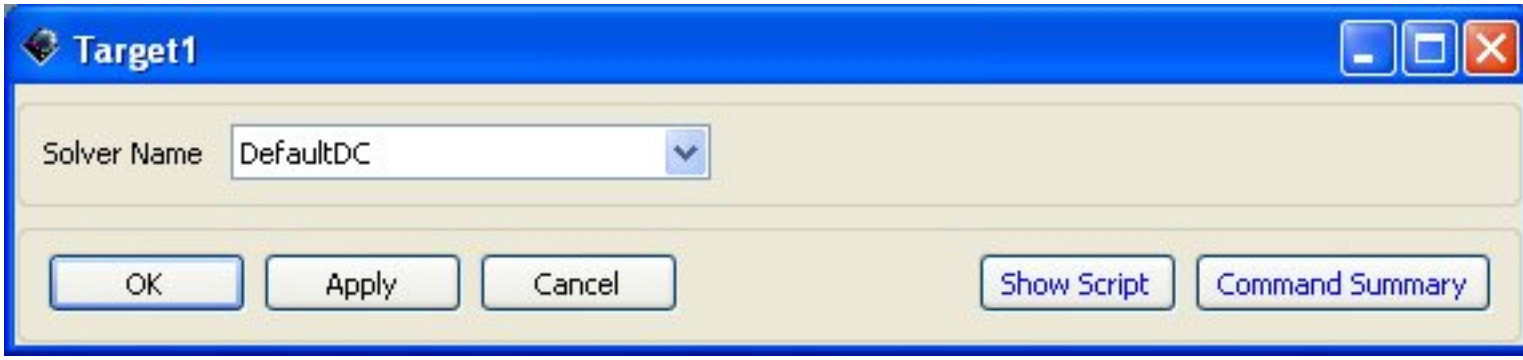


Figure: Target1 Target Command Command

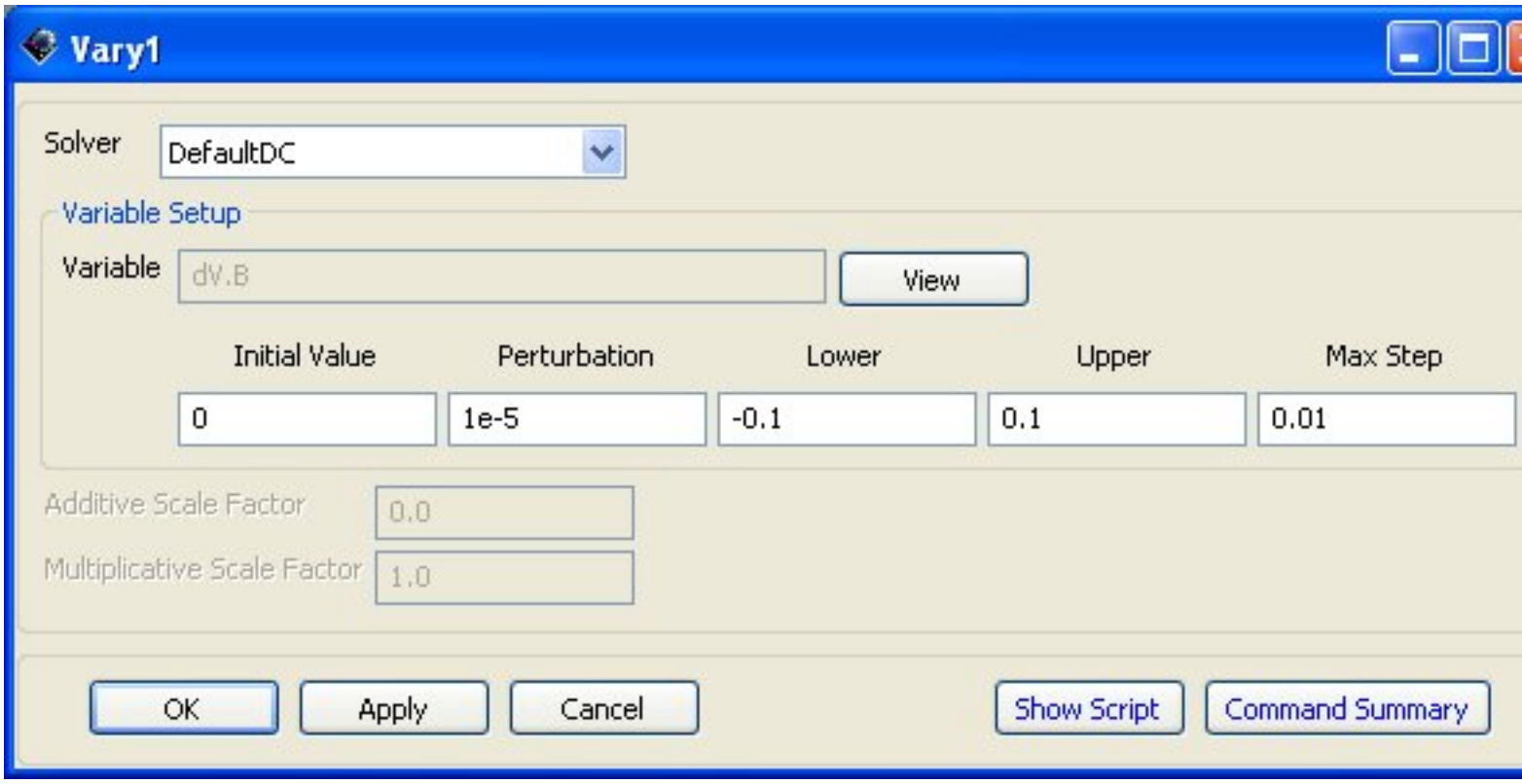


Figure: Vary1 Vary Command

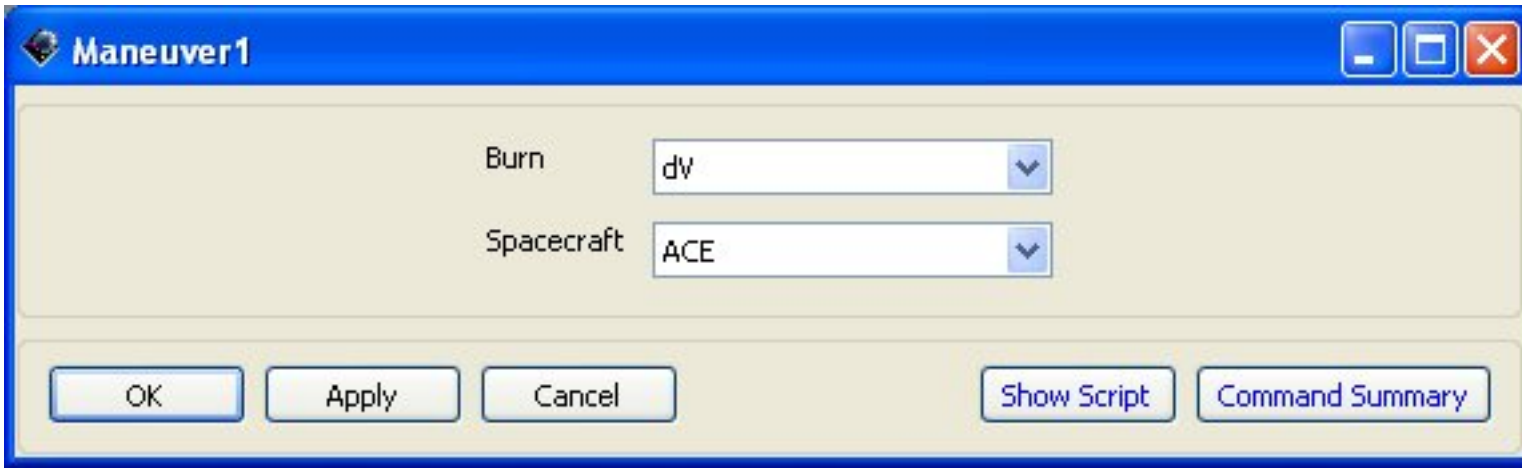


Figure: Maneuver1 Maneuver Command

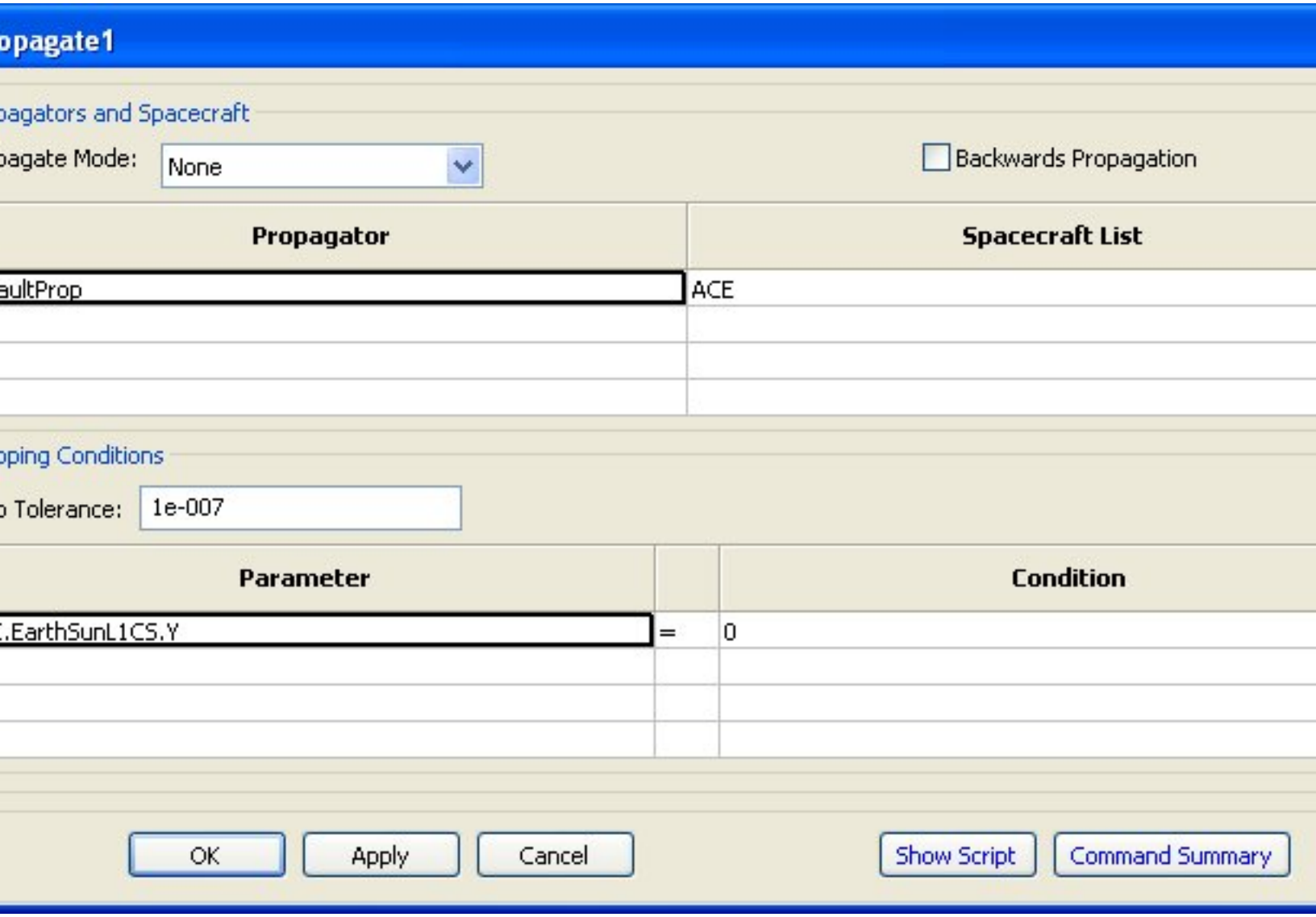


Figure: Propagate1 Propagate Command

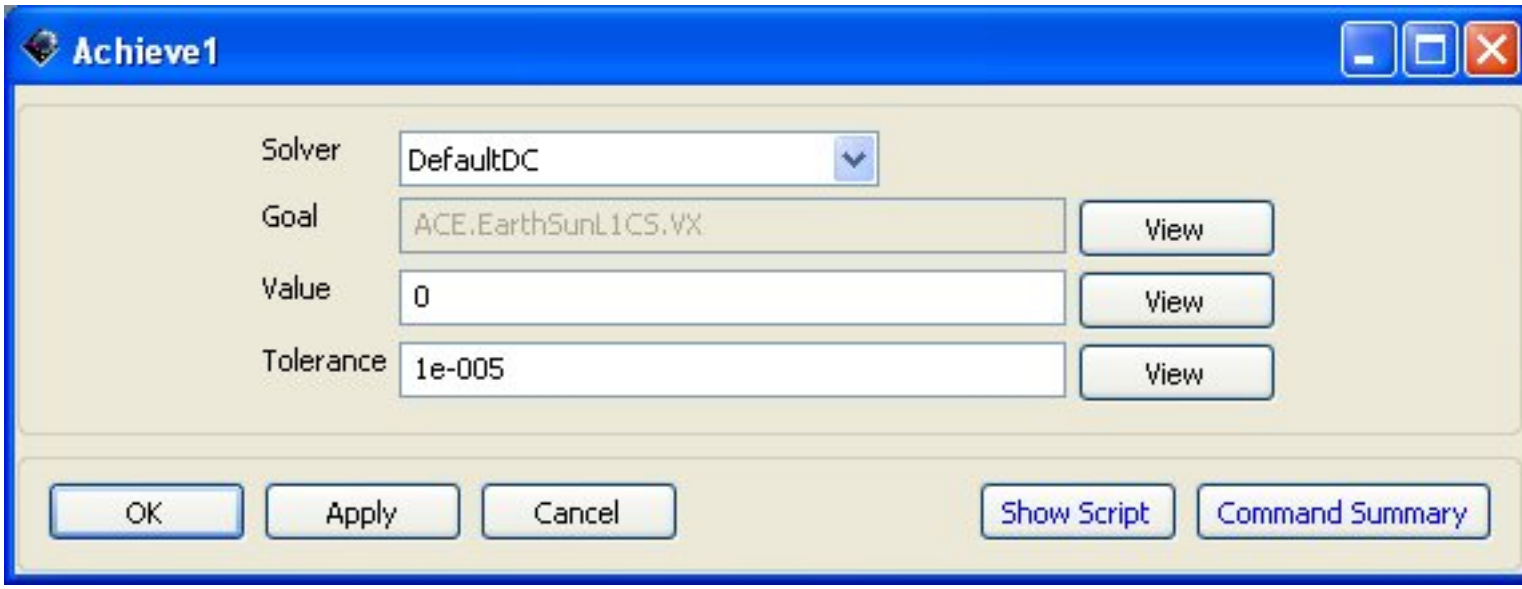


Figure: Achieve1 Achieve Command Command

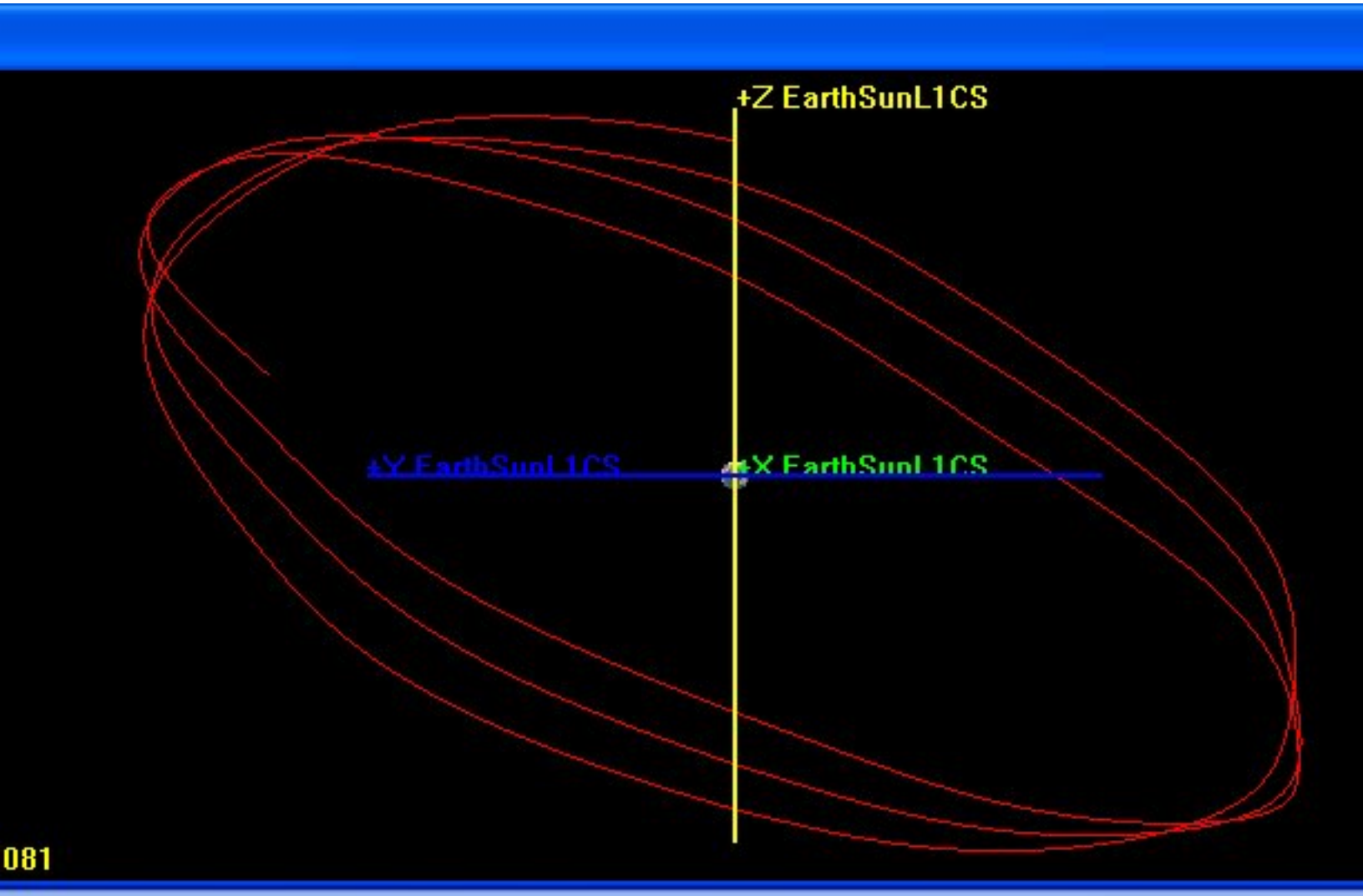
Running the Mission

Image of Output Tree

While running this setup, you should see the spacecraft ACE make seven half-loops around the Sun-Earth L_1 Libration Point. Just before each run you will see a slight pause in drawing of the model as GMAT computes the needed change in velocity. As you may be able to see in the reports, the changes in velocity here are quite small due to this Libration Point's unstable nature.

If you look into the reports stored in the directory specified then you will find the various parameters that were saved. This information provides insight as to the movement of the ACE spacecraft over time, especially if you take to a more text based presentation of information, rather than a visual one.

By now you should understand not only how to setup the required objects for this mission such as the Barycenter, Libration Point, Coordinate System Objects, Reports, Variable Object, and different views for the OpenGL Plot Object, but also how to apply these to create an effective and well documented mission. Below are the screen captures of both the output tree (where you can view the report output information) and the output window itself.


+Z EarthSunL1CS
+Y EarthSunL1CS
+X EarthSunL1CS
081

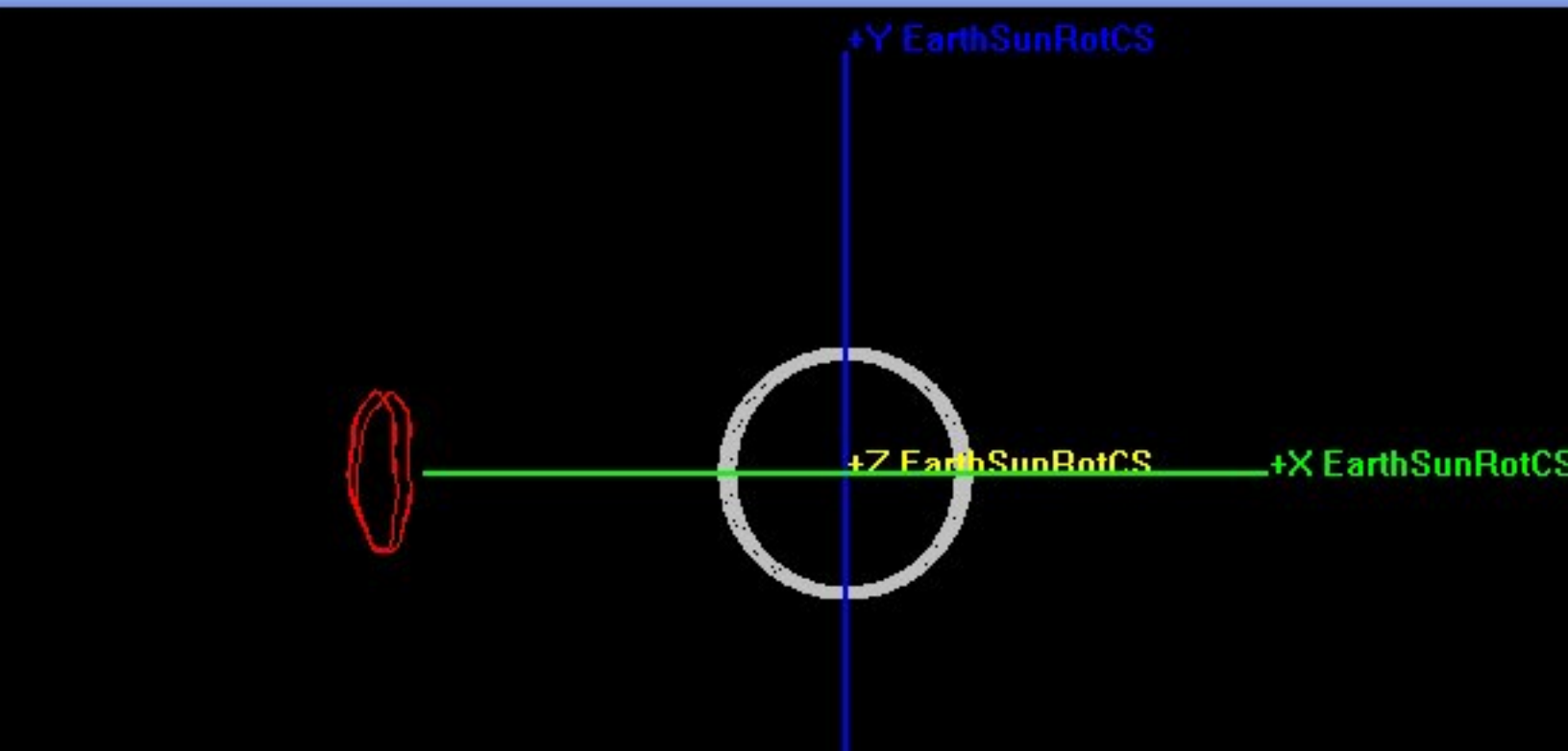

+Y EarthSunRotCS
+Z EarthSunRotCS
+X EarthSunRotCS

Figure: Output

Chapter 19. Lunar Transfer

This tutorial was started from the default settings in the June 9, 2008 build of GMAT.

Objective and Overview

The primary goal of this mission is for you to learn how to create and propagate a transfer trajectory from a LEO (Low Earth Orbit) to a lunar orbit. Given the capabilities of GMAT, you will also learn how to solve for several initially unknown variables and apply them to give your spacecraft a more efficient path to the moon.

Download the script file: LunarTransferTutorial.script [scripts/LunarTransferTutorials.script]

Prerequisites

- Basic understanding of Hohmann Transfer trajectories
- Tutorials:
  - Creating and Propagating a Spacecraft
  - Designing a Hohmann Transfer

Mission Description

- Objective: Target a lunar transfer trajectory with several initially undetermined variables and achieve a given altitude for a lunar propagation.
- Assume:
  - Launch site is Kennedy Space Center (INC = 28.5°)
  - Parking orbit altitude is 185 km (SMA = 6555 km)
  - Parking orbit is nearly circular (ECC = 0.001)
- Find:
  - RAAN
  - AOP
  - #V

Resource, Mission, and Output trees

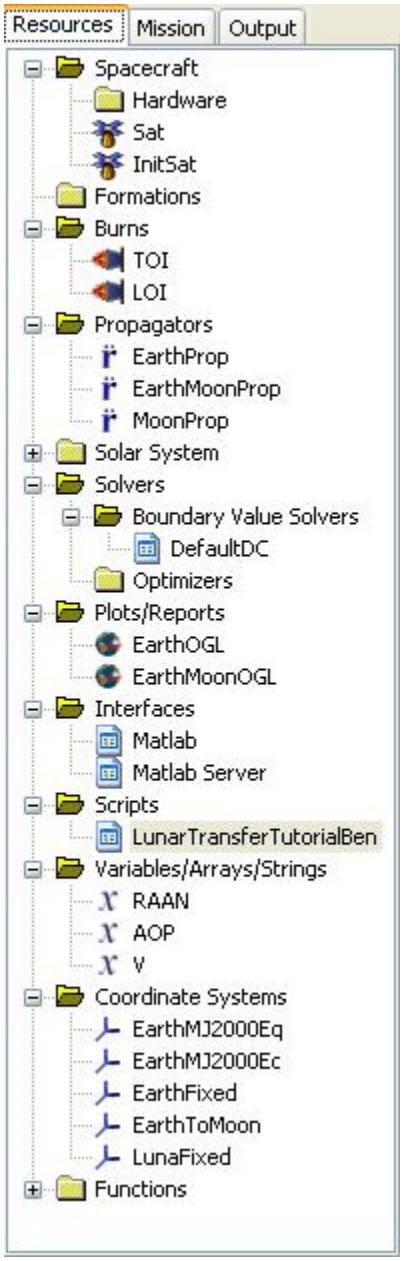


Figure: Lunar Transfer Resource, Mission, and Output trees

Creating and Configuring the Resource Tree

For this tutorial, two spacecraft, two impulsive burns, and three undetermined variables are needed in order to propagate a viable lunar transfer trajectory to a lunar orbit. To realistically account for the varying gravitational fields of the sun, earth, and moon, three propagators are required. Two OpenGL plots are needed to see how the spacecraft and its trajectory would look from an arbitrary point of view in space. Finally, an earth-lunar and lunar coordinate system should be used to allow for an easier interpretation of how the spacecraft is positioned once the moon is the primary body of its propagation.

Objects Required

- Spacecraft: InitSat and Sat
- Impulsive Burns: TOI (Transfer Orbit Insertion) and LOI (Lunar Orbit Insertion)
- Propagators: EarthOnlyProp, NearEarthProp, and NearMoonProp
- Plots/Reports: EarthOGL and EarthMoonOGL
- Variables/Arrays/Strings: RAAN, AOP, and #V
- Coordinate Systems: EarthToMoon and LunaFixed

Creating and Modifying Objects

- Spacecraft
  - Add a spacecraft to the resource tree and name it Sat
  - Set the epoch format to UTCGregorian
  - Use Keplerian as the State Type for the coordinate system
  - See Assumptions to modify spacecraft element properties
  - Clone Sat and rename it InitSat

  Script Syntax:

```
Create Spacecraft Sat;
GMAT Sat.DateFormat = UTCGregorian;
GMAT Sat.DisplayStateType = Keplerian;
GMAT Sat.SMA = 6554.999999999999;
GMAT Sat.ECC = 0.0009999999999997056;
GMAT Sat.INC = 28.50000000000001;
Create Spacecraft InitSat;
```

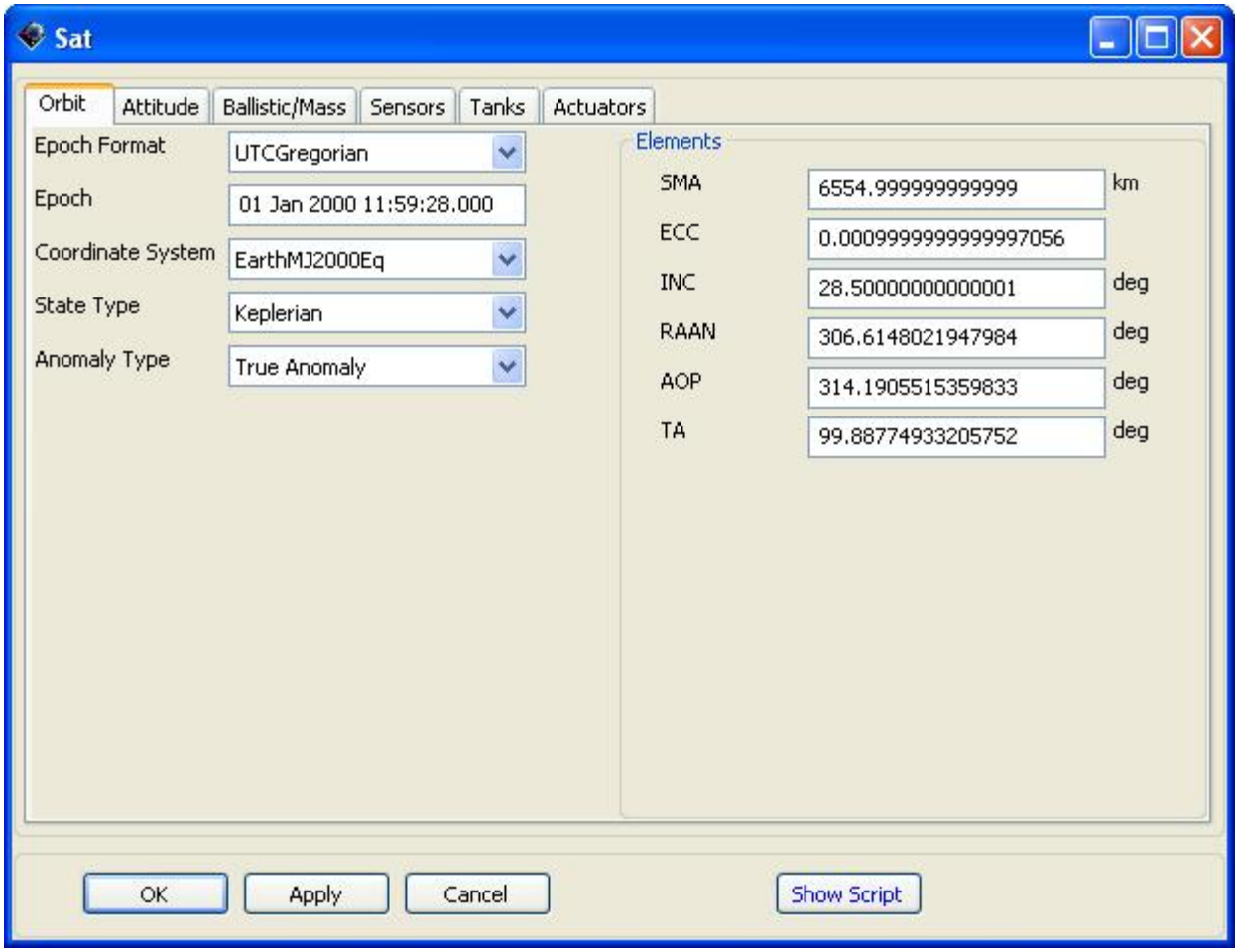


Figure: Sat Spacecraft Object

- Propagators
  - EarthOnly:
    - Include only a heliocentric pointmass

    Script Syntax:

```
Create ForceModel EarthOnly_ForceModel;
GMAT EarthOnly_ForceModel.PointMasses = {Sun};

Create Propagator EarthOnly;
```

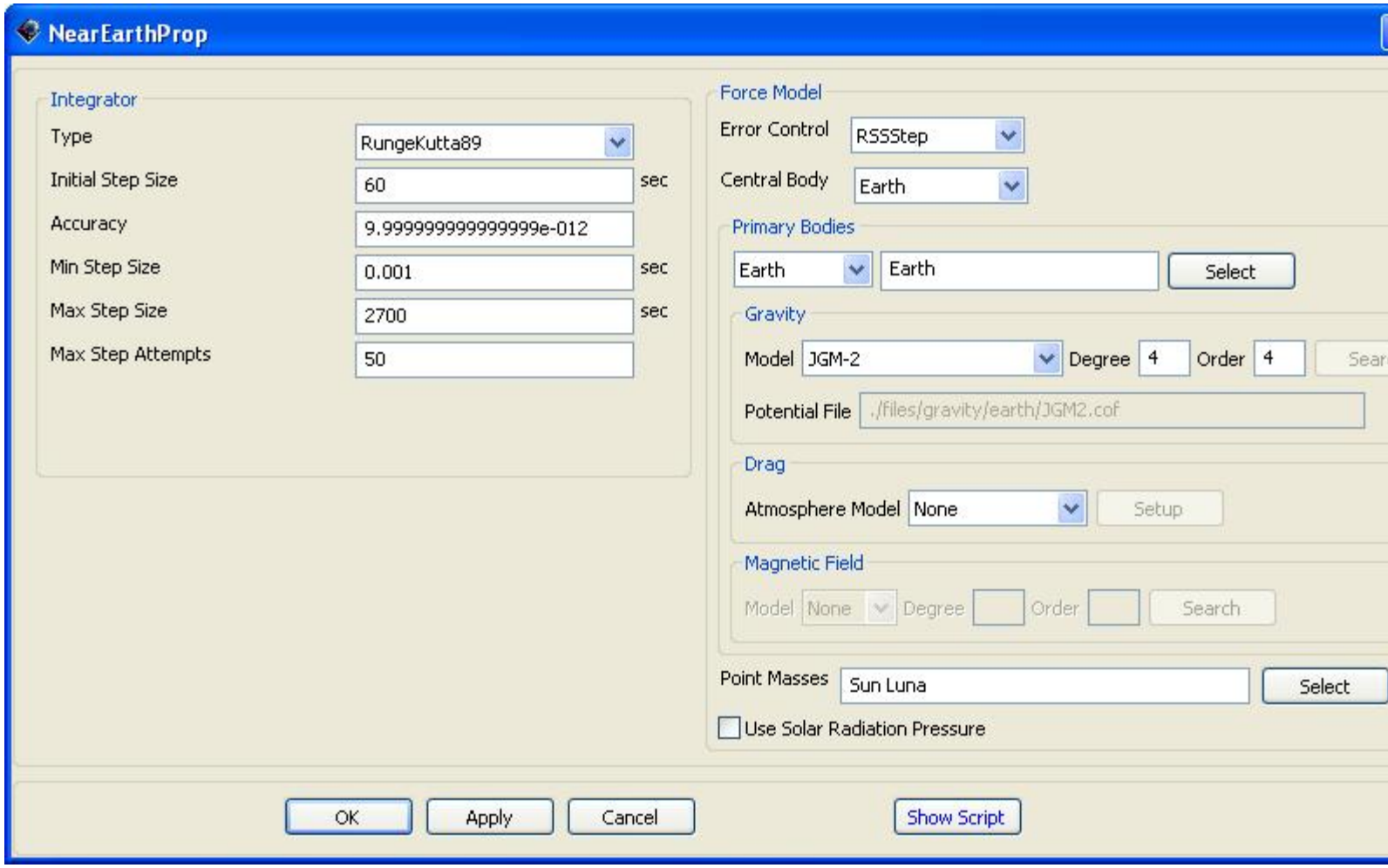


Figure: NearEarthProp Propagator Object

- NearEarthProp:
  - Include heliocentric and lunar point masses

  Script Syntax:

```
Create ForceModel NearEarthProp_ForceModel;
GMAT NearEarthProp_ForceModel.PointMasses = {Sun, Luna};

Create Propagator NearEarthProp;
GMAT NearEarthProp.FM = NearEarthProp_ForceModel;
```

- NearMoonProp:
  - Use the moon as the only Primary Body
  - Change the gravity model to LP-165 with a degree and order of 4
  - Include geocentric and heliocentric point masses

Script Syntax:

```
Create ForceModel NearMoonProp_ForceModel;
GMAT NearMoonProp_ForceModel.CentralBody = Luna;
GMAT NearMoonProp_ForceModel.PrimaryBodies = {Luna};
GMAT NearMoonProp_ForceModel.PointMasses = {Sun, Earth};
GMAT NearMoonProp_ForceModel.GravityField.Luna.PotentialFile =
'LP165P.cof';

Create Propagator NearMoonProp;
GMAT NearMoonProp.FM = NearMoonProp_ForceModel;
```

- Burns
  - TOI:
    - Change V to be 2 km/s

    Script Syntax:

```
Create ImpulsiveBurn TOI;
GMAT TOI.Element1 = 2;
```

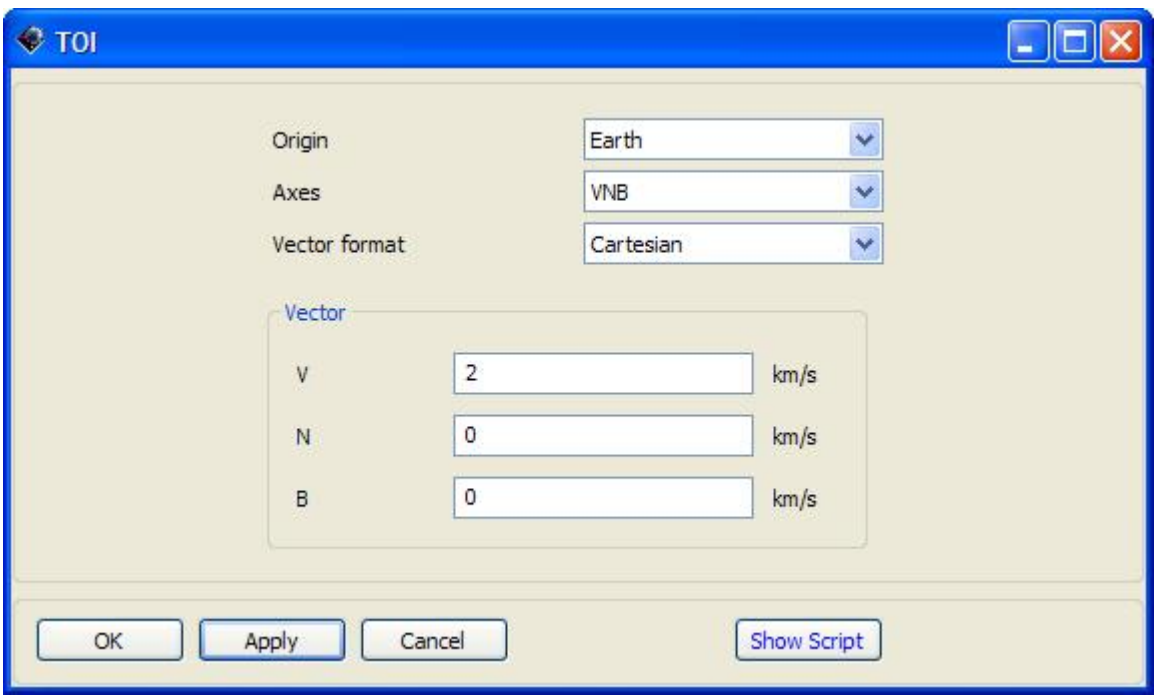


Figure: TOI Impulsive Burn Object

  - LOI:

    Script Syntax:

```
Create ImpulsiveBurn LOI;
```

- Variables/Arrays/Strings

- Add three variables to the resource tree and name them RAAN, AOP, and #V
- Set each equal to zero and to the Sat spacecraft

Script Syntax:

```
Create Variable RAAN AOP V;
```

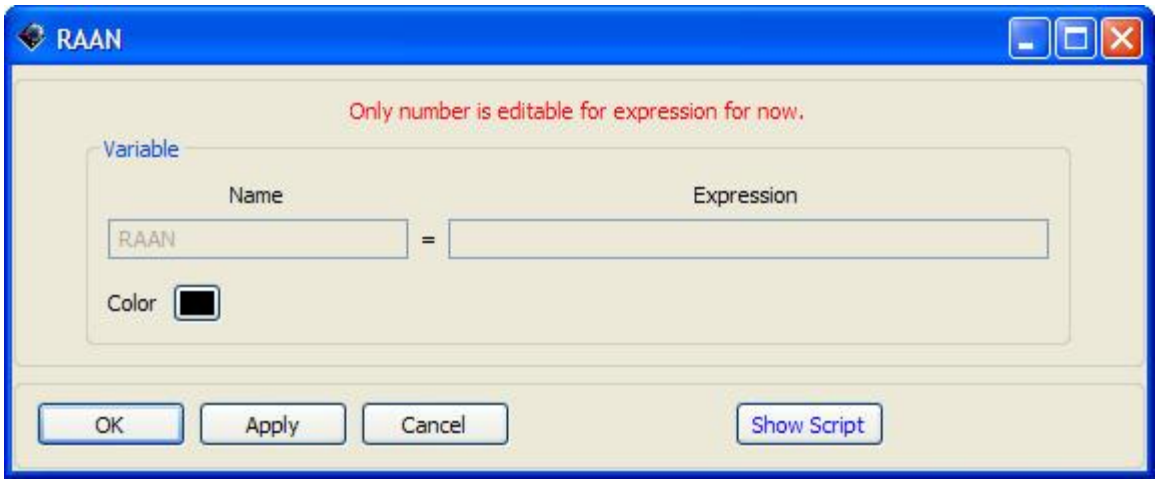


Figure: RAAN Variable Object

- Coordinate Systems
  - Add a new system to the resource tree and rename it EarthToMoon
  - Change the origin to be the moon
  - Set the Axes Type to ObjectReferenced
  - Set the moon to be the primary body and the earth to be the secondary body

  Script Syntax:

```
Create CoordinateSystem EarthMoonRot;
GMAT EarthMoonRot.Origin = Luna;
GMAT EarthMoonRot.Axes = ObjectReferenced;
GMAT EarthMoonRot.Primary = Luna;
GMAT EarthMoonRot.Secondary = Earth;
```

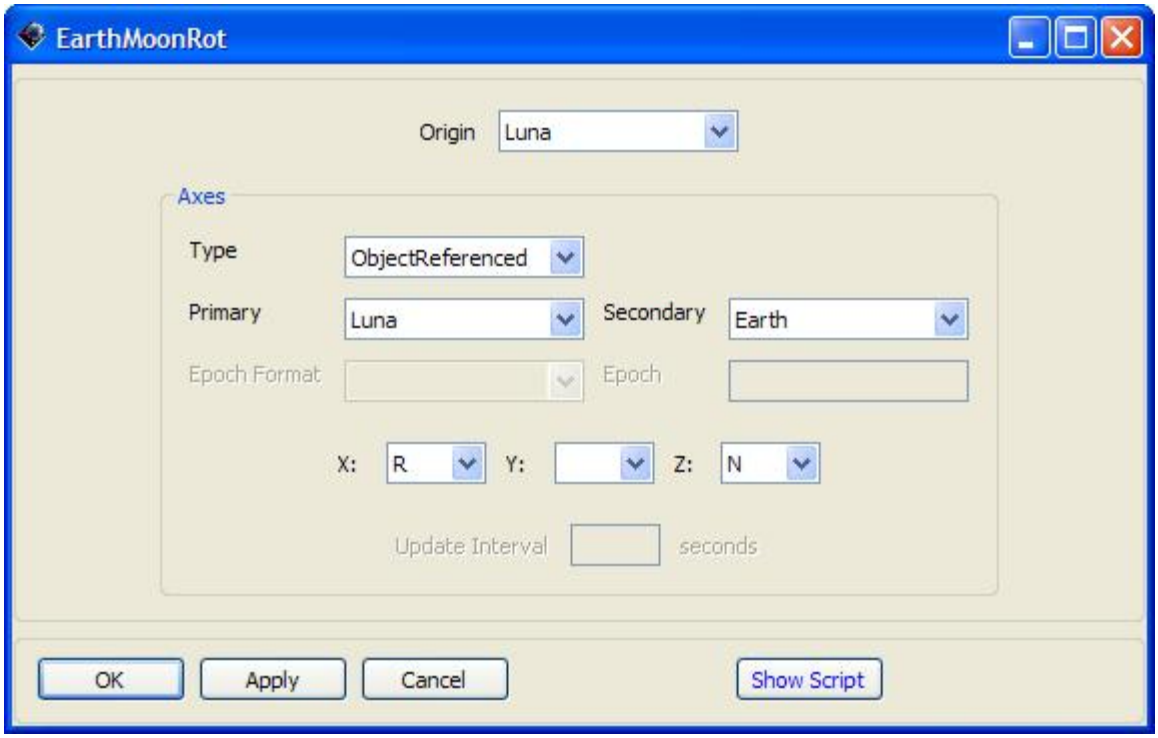


Figure: EarthMoonRot Coordinate System Object

- Plots/Reports
  - EarthOGL:
    - Add Luna to the Selected Celestial Objects field
    - Use a View Scale Factor of 15
    - Select X as View Up Definition Axis

    Script Syntax:

```
Create OpenGLPlot EarthOGL;
GMAT EarthOGL.Add = {Sat, Earth, Luna};
GMAT EarthOGL.ViewScaleFactor = 15;
GMAT EarthOGL.ViewUpAxis = X;
```

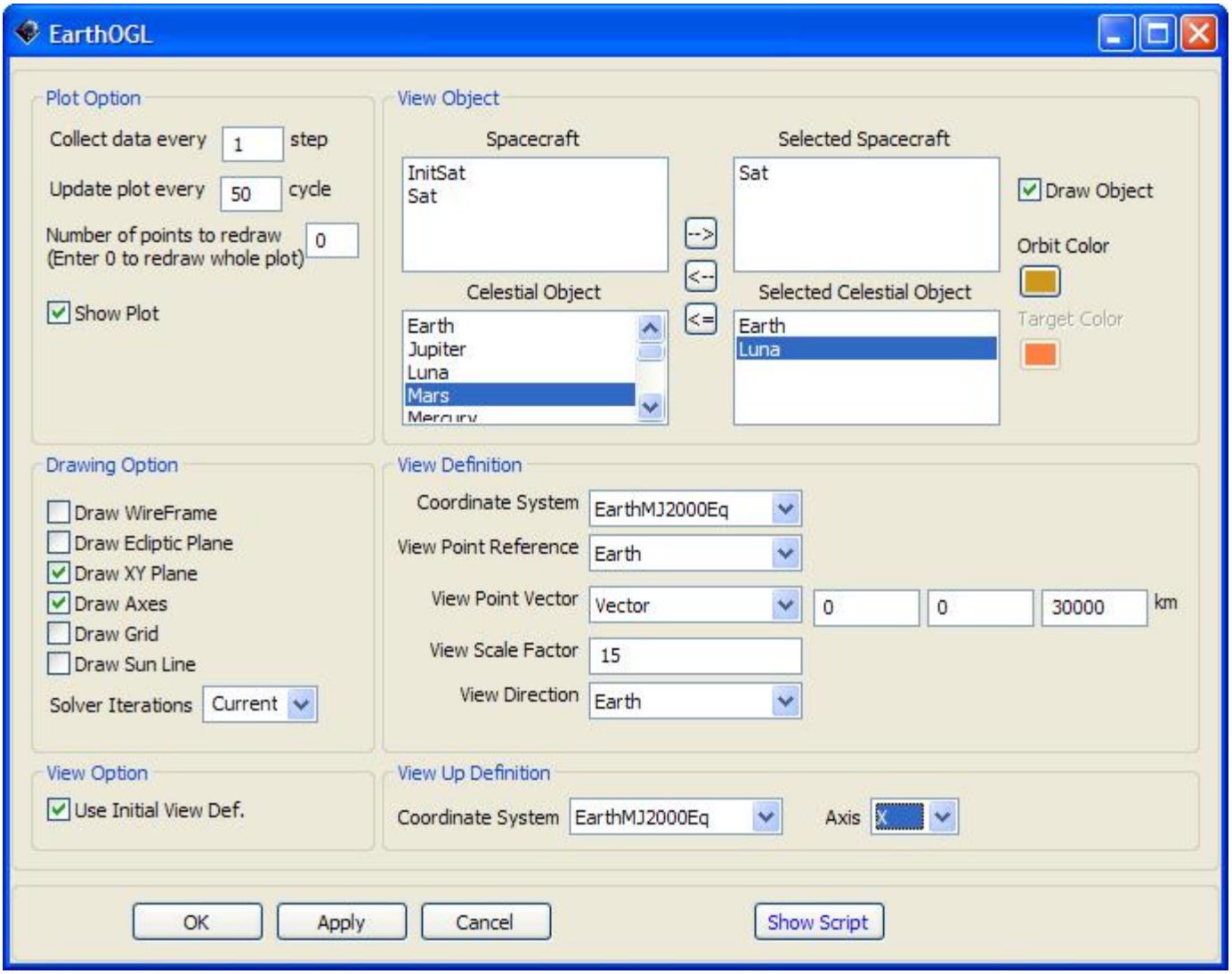


Figure: EarthOGL OpenGL Plot Object

- EarthMoonOGL:
  - Add Luna to the Selected Celestial Objects field
  - Change the coordinate system in the View Definition and View Up Definition group boxes to be EarthToMoon
  - Select Luna as the View Point Reference and View Direction body from those drop down menus

  Script Syntax:

```
Create OpenGLPlot EarthMoonOGL;
GMAT EarthMoonOGL.Add = {Sat, Luna, Earth};
GMAT EarthMoonOGL.ViewDirection = Luna;
GMAT EarthMoonOGL.ViewUpAxis = Z;
GMAT EarthMoonOGL.ViewPointReference = Earth;
```

```
GMAT EarthMoonOGL.CoordinateSystem = EarthMJ2000Eq;
GMAT EarthMoonOGL.ViewUpCoordinateSystem = EarthMJ2000Eq;
```

Creating and Configuring the Mission Sequence

Resource Mission and Output Trees

This mission calls three targeting sequences to accomplish the mission, which are explained below. The goal of the mission is to vary three unknowns, achieve them, and successfully transfer into a lunar, polar, circular orbit. To achieve a polar orbit we will use what is known as B-Plane Targeting to define how we want the orbit to be oriented. Setting the BdotT variable to zero says our spacecraft will orbit over the lunar poles. If the orbit was to be equatorial we would simply replace BdotT with a BdotR of zero. (Note: If the dot product of B and T vectors is unfamiliar to you, Analytical Graphics Inc., makers of STK, has a good website [https://www.analyticalgraphics.com/resources/help/stk613/helpSystem/extfile/gator/eq-bplane.htm|explaining] it).

- The Target 1 Sequence solves for three specified variables (in this case RAAN, AOP, and #V) by appling the first impulsive burn to get the spacecraft to exit its parking orbit and enter a lunar transfer trajectory. The spacecraft is told to propagate from the parking orbit's apoapsis to initiate the sequence.
- The Target 2 Sequence calls the values achieved in Target 1 to use them as a good initial guess to converge onto a viable lunar transfer trajectory, which is why we needed a second spacecraft. InitSat uses those achieved values and sets them equal to three new Vary commands. These new commands wish to achieve a lunar altitude and BdotT, calling for a definite type of lunar orbit. Also, in order to make things more realistic, two Propagates are added to slowly progress from one dominant gravitational field into another when approaching the moon.
- The Target 3 Sequence utilizes the second impulsive burn to slow the spacecraft in its velocity direction to achieve a circular eccentricity, and a polar, lunar orbit. This sequence states occurs at lunar apoapsis.

Creating and Modifying Sequences

Target 1 Sequence

- Target 1:
  - Vary 1:
    - Set the first Variable to Element1 using the TOI impulsive burn
    - Set the Initial Value parameter equal to 3.15 km/s
    - Define the Lower (Limit) to be 3.05 km/s
    - Define the Upper (Limit) to 3.25 km/s
    - Change the Max Step to 0.01 km/s
  - Vary 2:
    - Set the second Variable to AOP
    - Use an Initial Value of 0°

    - Change the Perturbation field to 0.2
    - Change the Max Step to 15°
  - Vary 3:
    - Set the third Variable to RAAN
    - Set the Initial Value equal to 320°
    - Change the Perturbation to 0.2
    - Change the Max Step to 15°
  - Maneuver 1: utilize the TOI impulsive burn
  - Propagate 1: set the Stopping Condition parameter to earth's apoapsis
  - Achieve 1:
    - Goal:
      - Highlight Sat in Object List menu
      - Select the EarthToMoon coordinate system from the drop down menu
      - Add X to the Selected Value(s) field
      - Change the Value field to be 16000 km
  - Achieve 2:
    - Goal:
      - Highlight Sat in Object List menu
      - Select the MarsMJ2000Eq coordinate system from the drop down menu
      - Add Y in to the Selected Value(s) field
      - Change the Value field to be 0 km
  - Achieve 3:
    - Goal:
      - Highlight Sat in Object List menu
      - Select the MarsMJ2000Eq coordinate system from the drop down menu
      - Add Z in to the Selected Value(s) field
      - Change the Value field to be 0 km

Script Syntax:

```
GMAT InitSat = Sat;
Target DC1;
```

```
   Vary DC1(Sat.RAAN = 270, {Perturbation = .01, MaxStep = 15, Lower = -5360, Uppe
   Vary DC1(Sat.AOP = 0, {Perturbation = .01, MaxStep = 15, Lower = -5360, Upper =
   Vary DC1(TOI.Element1 = 3.1, {Perturbation = 0.00001, MaxStep = 0.005, Lower =
   Maneuver TOI(Sat);
   Propagate EarthOnly(Sat) {Sat.Earth.Apoapsis};
   Achieve DC1(Sat.EarthMoonRot.X = 10000, {Tolerance = 1});
   Achieve DC1(Sat.EarthMoonRot.Y = 0, {Tolerance = 1});
   Achieve DC1(Sat.EarthMoonRot.Z = 0, {Tolerance = 1});
EndTarget;
```

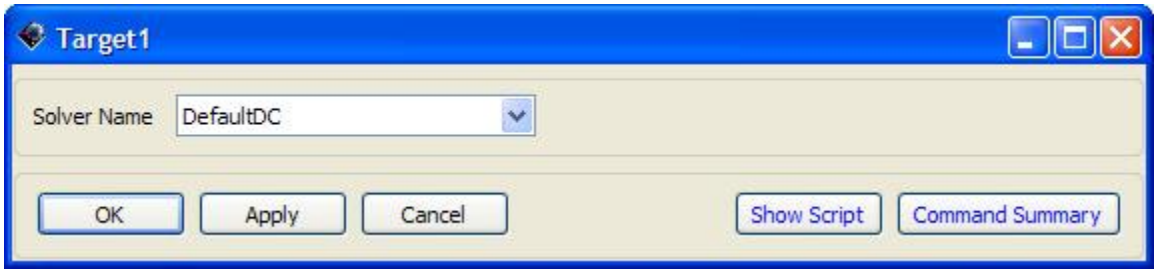


Figure: Target1 Target Command

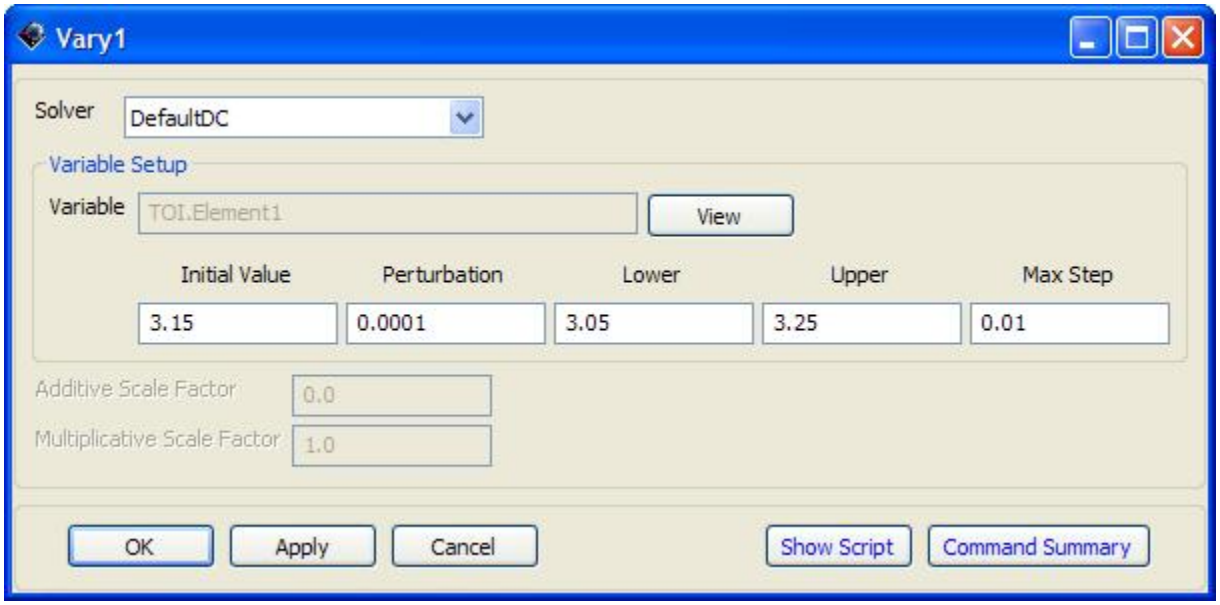


Figure: Vary1 Vary Command

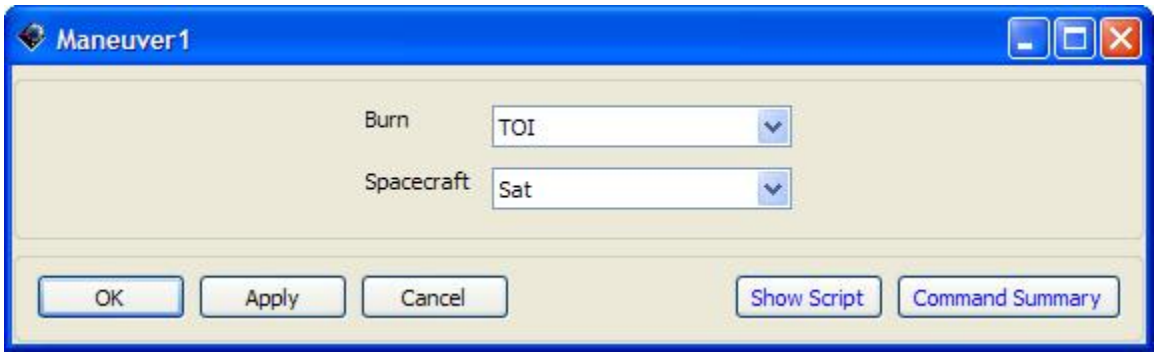


Figure: Maneuver1 Maneuver Command

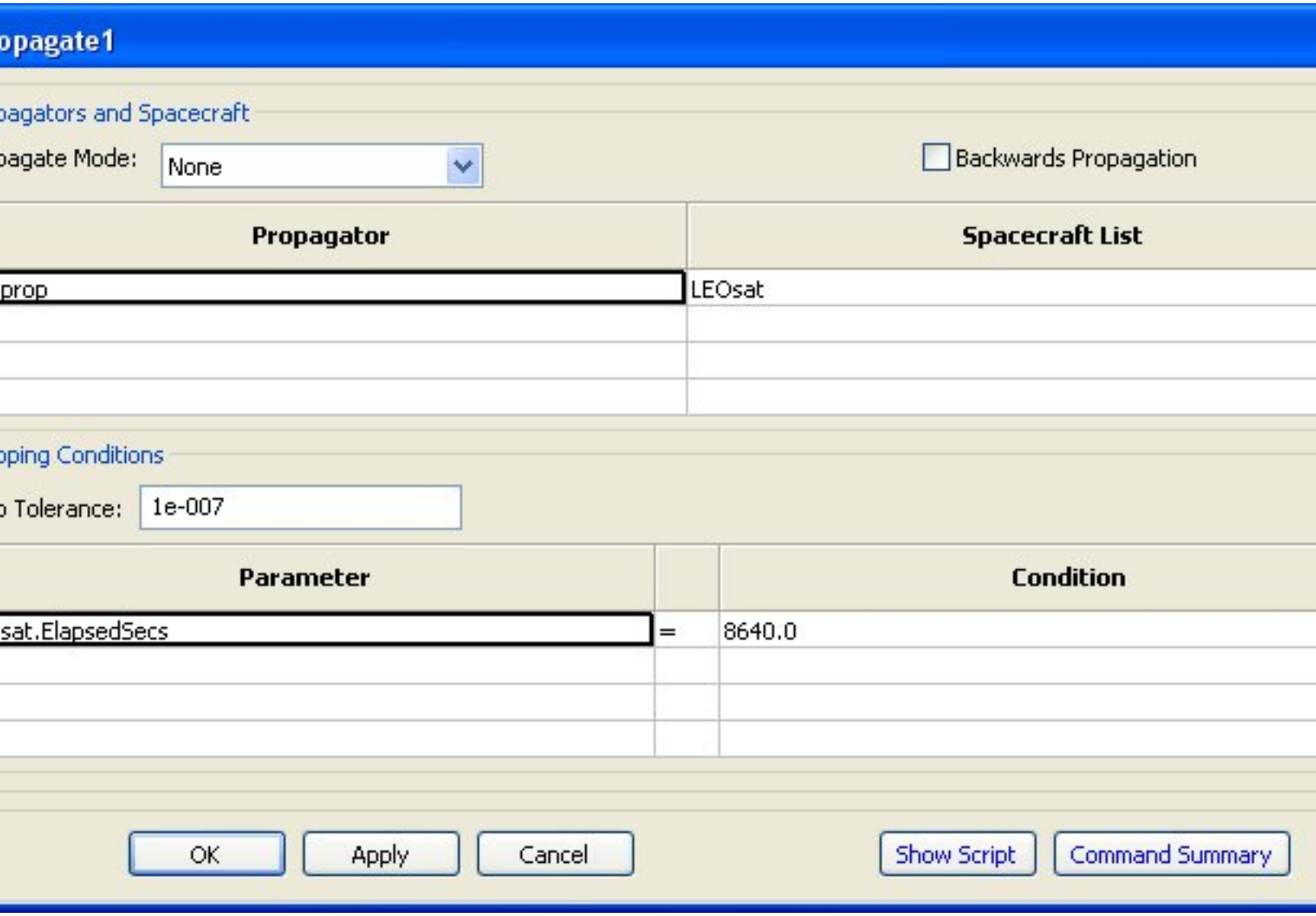


Figure: Propagate1 Propagate Command

Figure: Achieve1 Achieve Command

- Equation 1: set Sat.RAAN = RAAN
- Equation 2: set Sat.AOP = AOP
- Equation 3: set Sat = InitSat

Script Syntax:

```
GMAT Sat.RAAN = RAAN;
GMAT Sat.AOP = AOP;
GMAT Sat = InitSat;
```

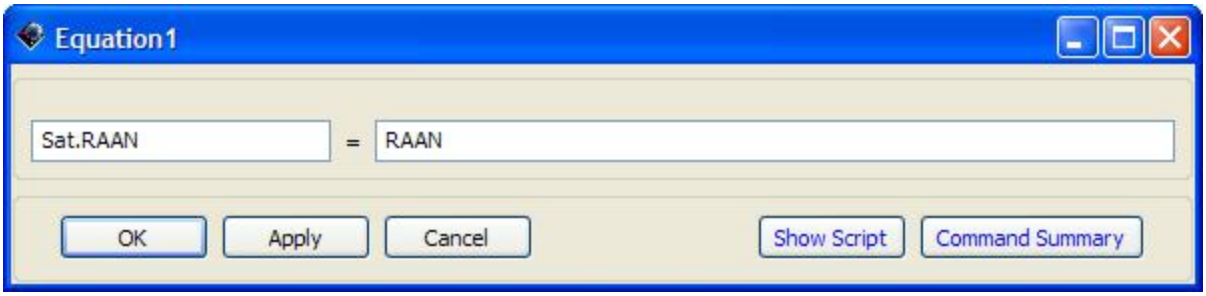


Figure: Equation1 Equation Command

Target 2 Sequence

- Target 2:
  - Vary 4:
    - Set the first Variable to Element 1 using TOI impulsive burn
    - Set the Initial Value to TOI.Element1
  - Vary 5:
    - Set the second Variable to AOP
    - Set the Initial Value to AOP
  - Vary 6:
    - Set the third Variable to RAAN
    - Set the Initial Value to RAAN
  - Maneuver 2: utilize the TOI impulsive burn
  - Set the stopping condition parameter to be 2 days for Sat in Propagate 2
  - Set the stopping condition parameter to be lunar periapsis in Propagate 3
  - Achieve 4:
    - Goal:
      - Use Lunar altitude
      - Set the Altitude to 300 km
      - Use a Tolerance of 1.0 km
  - Achieve 5:
    - Define the Goal as BdotT using the EarthToMoon coordinate system
      - Use a BdotT of 0.0 km
      - Use a Tolerance of 1.0 km

Script Syntax:

```
GMAT Sat = InitSat;
GMAT RAAN = Sat.RAAN;
GMAT AOP = Sat.AOP;

Target DefaultDC;
   Vary DefaultDC(TOI.Element1 = TOI.Element1, {Perturbation = 0.0001, MaxStep = 0
      9.999999e300});
   Vary DefaultDC(Sat.AOP = AOP, {Perturbation = 0.1, MaxStep = 15}

   Vary DefaultDC(Sat.RAAN = RAAN, {Perturbation = 0.1, MaxStep = 15}
   Maneuver TOI(Sat);
   Propagate EarthMoonProp(Sat) {Sat.ElapsedDays = 2.0};
   Propagate MoonProp(Sat) {Sat.Luna.Periapsis};
   Achieve DefaultDC(Sat.Luna.Altitude = 300, {Tolerance = 1.0});
   Achieve DefaultDC(Sat.EarthToMoon.BdotT = 0.0, {Tolerance = 1.0});
EndTarget;
```

Target 3 Sequence

- Target 3:
  - Vary 7:
    - Set the Variable to V using LOI impulsive burn
    - Use an Initial Value of - 0.5 km/s
    - Define the Lower (Limit) to be - 4 km/s
    - Define the (Limit) to be 3.14 km/s
  - Maneuver 3: utilize the LOI impulsive burn
  - Propagate 4: change the Stopping Condition parameter to be lunar apoapsis
  - Achieve 6:
    - Change the Goal to reach a lunar eccentricity
    - Change the eccentricity Value to be equal to zero

Script Syntax:

```
Target DC1;
   Vary DC1(LOI.Element1 = 0.5, {Perturbation = 0.0001, MaxStep = 0.2, Lower = -4,
   Maneuver LOI(Sat);
   Propagate NearMoonProp(Sat) {Sat.Luna.Apoapsis};
   Achieve DC1(Sat.Luna.Altitude = 300, {Tolerance = 1});
   Achieve DC1(Sat.Luna.ECC = 0, {Tolerance = 0.1});
EndTarget;
Propagate NearMoonProp(Sat) {Sat.ElapsedDays = 2};
Propagate MoonProp(Sat) {Sat.ElapsedDays = 2.0};
```

Running the Mission

Resource Mission and Output Trees

You have now configured the mission, and it is time to run it and see the results. First, save the changes you've made and then click the green arrow on the tool bar to run the mission. The screen capture below summarizes what you should see after correctly completing this tutorial.

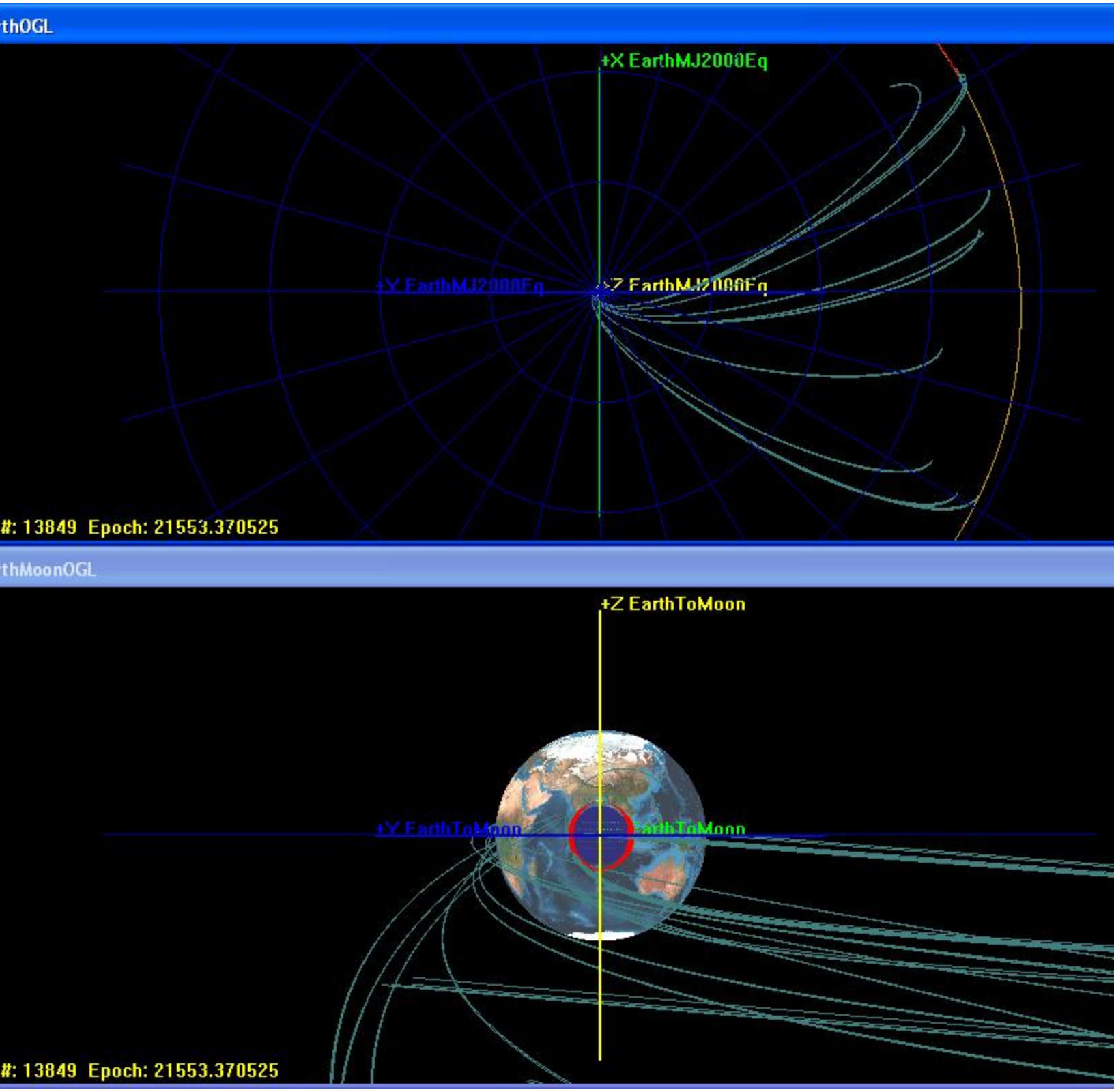


Figure: Output

Chapter 20. Formation Rendezvous

This tutorial was started from the default settings in the June 9, 2008 build of GMAT.

Objective and Overview

The primary goal of this mission is for you to learn how to create a formation flying mission that uses targeting sequences. The mission requires spacecraft dependent coordination systems and has plots in different spacecraft dependent coordinate systems and targets variables that are expressed in spacecraft dependent coordinate systems. Also, this mission includes a rendezvous. This tutorial indicates the method in which the spacecraft can accomplish formation flying and formation rendezvous.

Download the script file: FormationRendezvousTutorial.script [scripts/FormationRendezvousTutorial.script]

Prerequisites

- Basic Understanding of Formation Rendezvous

Mission Description

- Objective: Accomplishing a rendezvous by two certain spacecraft from formation flying of four spacecraft.
- Given:
  - Initial formation states of spacecraft, which indicate orbital elements
- Find:
  - ?Burn1 (?V, ?N, ?B)

Resource, Mission, and Output trees

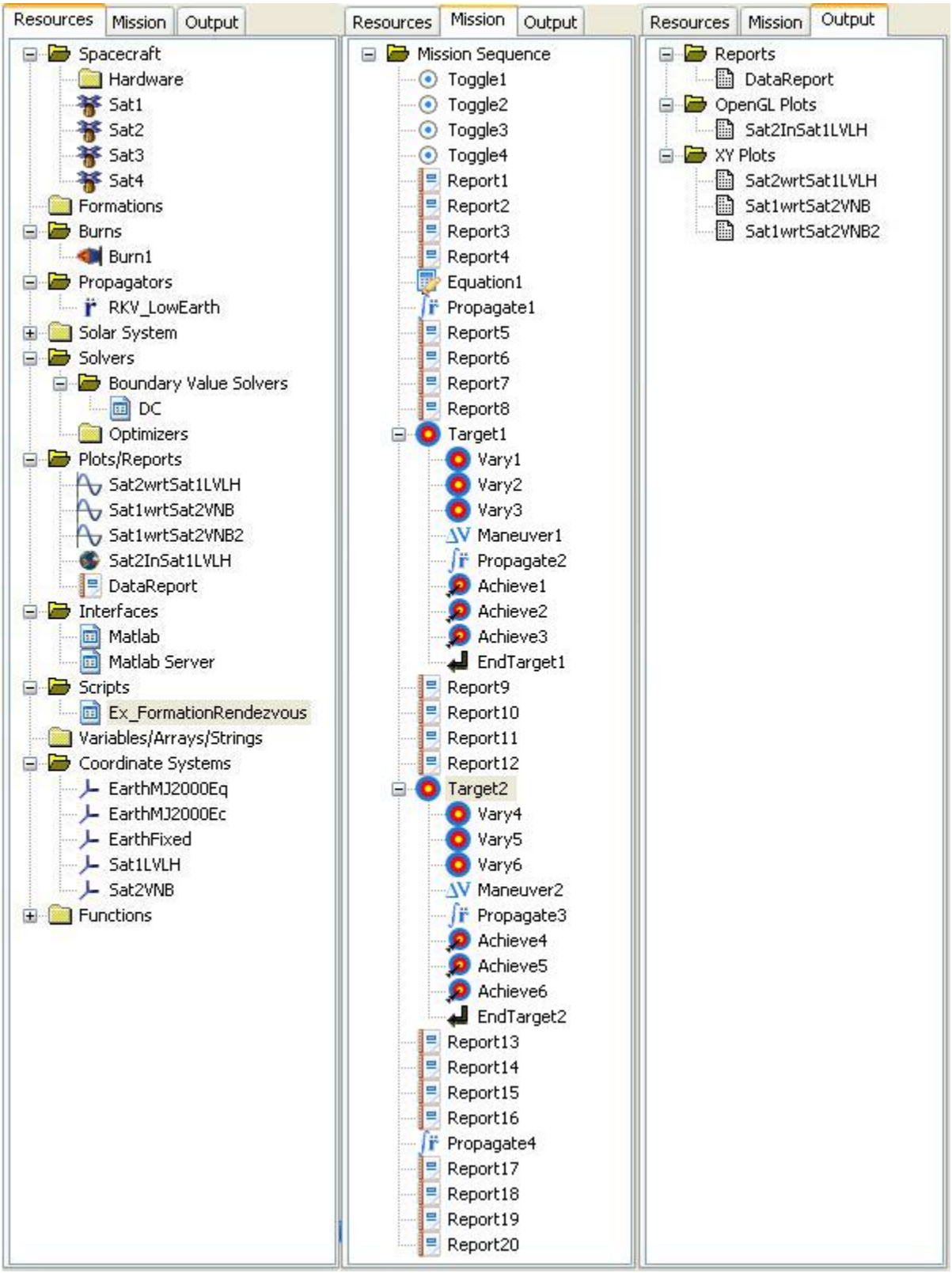


Figure: Formation Rendezvous Resource, Mission, and Output trees

Creating and Configuring the Resource Tree

In this section, some assumptions or conditions are described to accomplish the mission. Four spacecraft are created in this simple mission but only one spacecraft can have its trajectory changed. A single burn and a single propagator are needed as well. Several Plots/Reports and Coordinate Systems are set in order to analyze the details of each spacecraft trajectory.

Objects Required

- Spacecraft: Sat1, Sat2, Sat3, Sat4
- Impulsive Burn: Burn1
- Propagator: RKV_LowEarth
- Differential Corrector: DC
- OpenGL Plots: EarthView, DataReport
- Coordinate systems: Sat1LVLH, Sat2VNB

Creating and Modifying Objects

- Spacecraft
  - Sat1
    - Add spacecraft Sat1 (where, in this tutorial each initial orbital parameter assumes to be given as shown in the following Script Syntax)
    - Use coordination system EarthMJ2000Eq
    - Change the Epoch Format to UTCGregorian
    - Set the Epoch to 01 Jan 2000 11:59:27.966
    - Use Keplerian State Type

    Script Syntax:

```
Create Spacecraft Sat1;
GMAT Sat1.Epoch.UTCGregorian = 01 Jan 2000 11:59:27.966;
GMAT Sat1.CoordinateSystem = EarthMJ2000Eq;
GMAT Sat1.DisplayStateType = Keplerian;
GMAT Sat1.SMA   = 7600;
GMAT Sat1.ECC   = 0;
GMAT Sat1.INC   = 100.51;
GMAT Sat1.AOP  = 0;
GMAT Sat1.RAAN  = 278.85;
GMAT Sat1.TA  = 90;
```

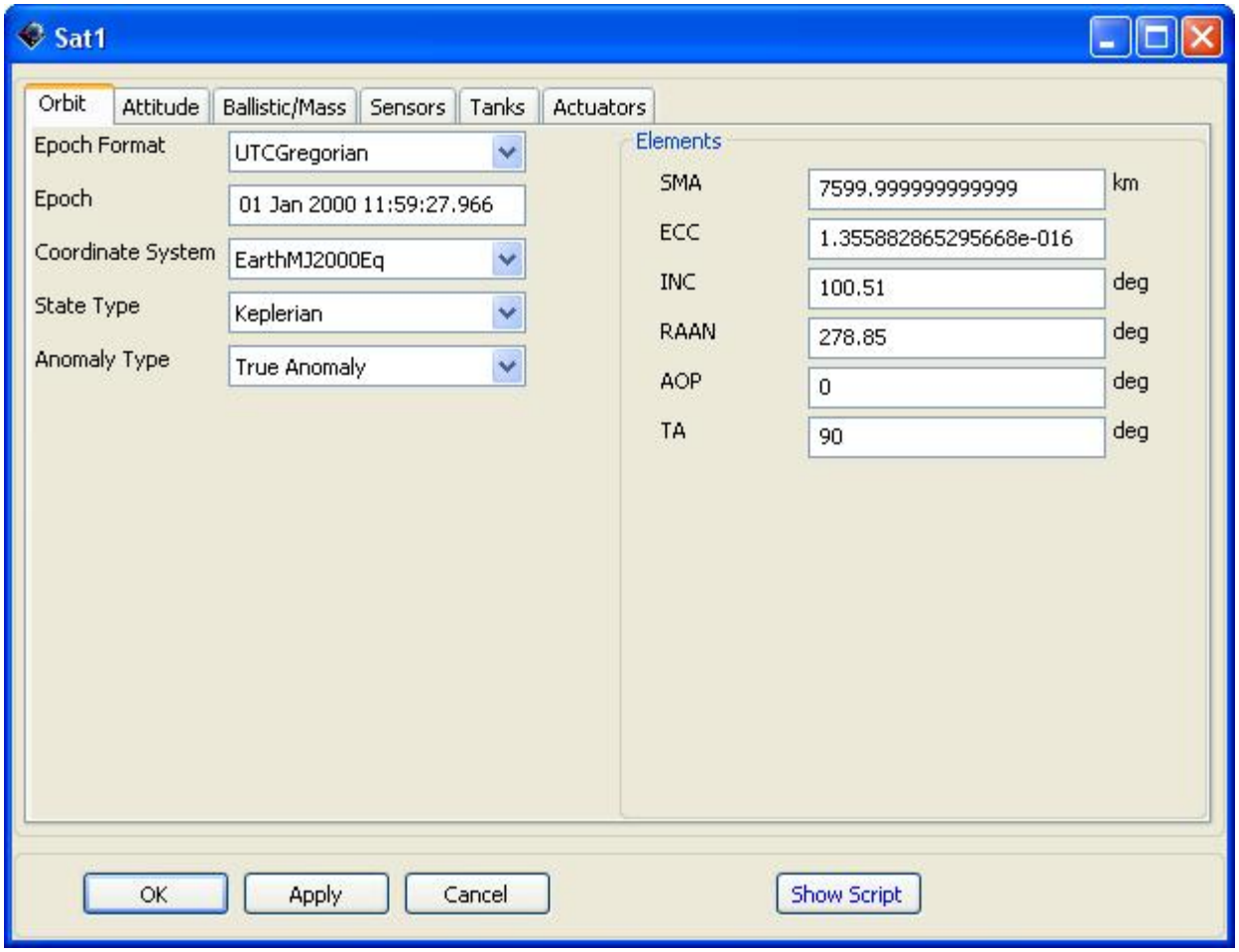


Figure: Sat1 Spacecraft Object

- Sat2
  - Add spacecraft Sat2 (where, in this tutorial each initial orbital parameter assumes to be given as shown in the following Script Syntax)
  - Use coordination system EarthMJ2000Eq
  - Change the Epoch Format to UTCGregorian
  - Set the Epoch to 01 Jan 2000 11:59:27.966
  - Use Keplerian State Type

  Script Syntax:

```
Create Spacecraft Sat2;
GMAT Sat2.Epoch.UTCGregorian = 01 Jan 2000 11:59:27.966;
GMAT Sat2.CoordinateSystem = EarthMJ2000Eq;
GMAT Sat2.DisplayStateType = Keplerian;
GMAT Sat2.SMA   = 7600;
GMAT Sat2.ECC   = 0.045;
```

```
GMAT Sat2.INC   = 100.47;
GMAT Sat2.AOP   = 180.95;
GMAT Sat2.RAAN  = 284.39;
GMAT Sat2.TA    = 264.90;
```

- Sat3
  - Add spacecraft Sat3 (where, in this tutorial each initial orbital parameter assumes to be given as shown in the following Script Syntax)
  - Use coordination system EarthMJ2000Eq
  - Change the Epoch Format to UTCGregorian
  - Set the Epoch to 01 Jan 2000 11:59:27.966
  - Use Keplerian State Type

  Script Syntax:

```
Create Spacecraft Sat3;
GMAT Sat3.Epoch.UTCGregorian = 01 Jan 2000 11:59:27.966;
GMAT Sat3.CoordinateSystem = EarthMJ2000Eq;
GMAT Sat3.DisplayStateType = Keplerian;
GMAT Sat3.SMA   = 7600;
GMAT Sat3.ECC   = 0.0702;
GMAT Sat3.INC   = 100.40;
GMAT Sat3.AOP   = 181.54;
GMAT Sat3.RAAN  = 287.90;
GMAT Sat3.TA    = 261.81;
```

- Sat4
  - Add spacecraft Sat4 (where, in this tutorial each initial orbital parameter assumes to be given as shown in the following Script Syntax)
  - Use coordination system EarthMJ2000Eq
  - Change the Epoch Format to UTCGregorian
  - Set the Epoch to 01 Jan 2000 11:59:27.966
  - Use Keplerian State Type

  Script Syntax:

```
Create Spacecraft Sat4;
GMAT Sat4.Epoch.UTCGregorian = 01 Jan 2000 11:59:27.966;
GMAT Sat4.CoordinateSystem = EarthMJ2000Eq;
GMAT Sat4.DisplayStateType = Keplerian;
GMAT Sat4.SMA   = 7600;
GMAT Sat4.ECC   = 0.101;
GMAT Sat4.INC   = 100.29;
GMAT Sat4.AOP   = 182.141;
GMAT Sat4.RAAN  = 291.82;
GMAT Sat4.TA    = 258.67;
```

- Propagators
  - Add propagator RKV_LowEarth
  - Remove primary bodies or gravity models from all propagators
  - Use RKV_LowEarth propagator
  - Set the integrator Min Step Size to 0.001
  - Set the integrator Max Step Size to 900
  - Set Earth, Sun, Luna and Jupiter as Point Masses

  Script Syntax:

  ```
  Create ForceModel LowEarth;
  GMAT LowEarth.PointMasses = {Earth, Sun, Luna, Jupiter};

  Create Propagator RKV_LowEarth;
  GMAT RKV_LowEarth.Type    = RungeKutta89;
  GMAT RKV_LowEarth.MinStep = .001;
  GMAT RKV_LowEarth.MaxStep = 900;
  GMAT RKV_LowEarth.FM      = LowEarth;
  ```

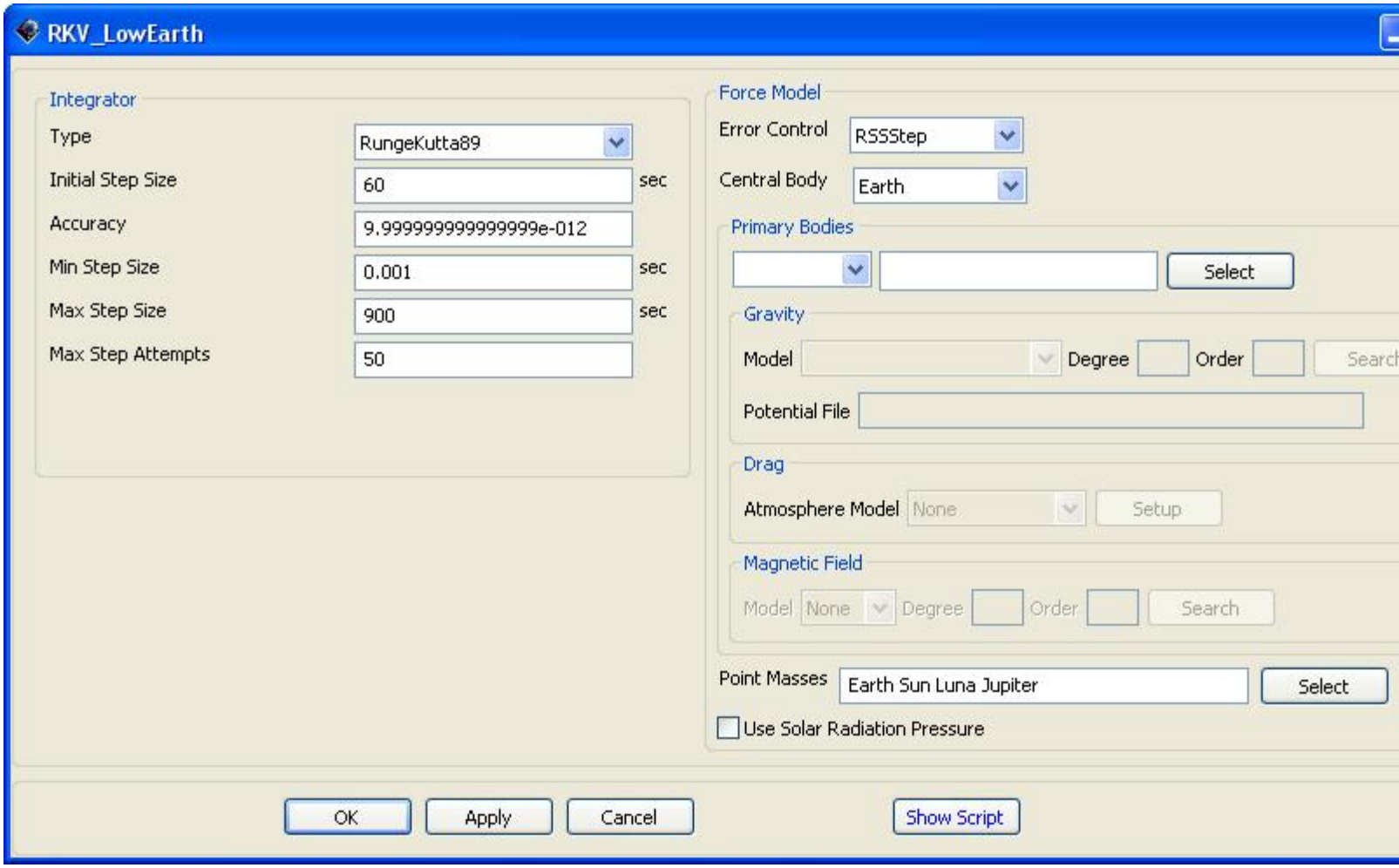


Figure: RKV_LowEarth Propagator Object

- Burns
  - Add impulsive burn Burn1
  - Origin is Earth
  - Set V to 0.0001

  Script Syntax:

```
Create ImpulsiveBurn Burn1;
GMAT Burn1.Origin = Earth;
GMAT Burn1.Axes = VNB;
GMAT Burn1.Element1 = .0001;
```

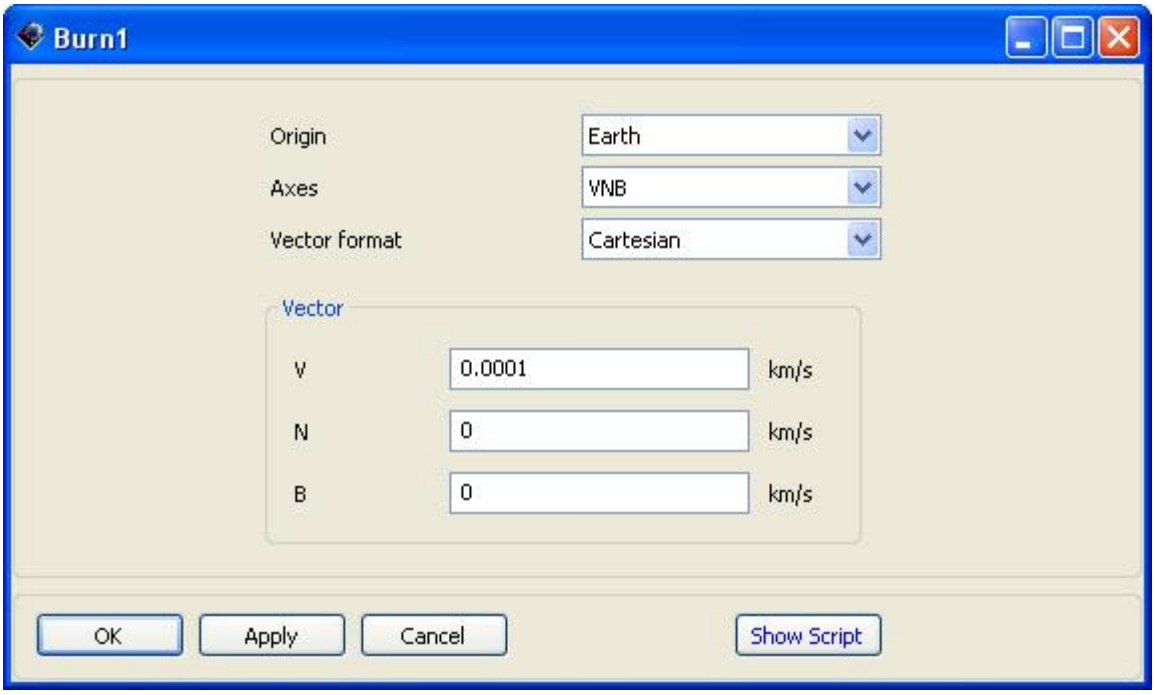


Figure: Burn1 Impulsive Burn Object

- Solvers
  - Set DifferentialCorrector DC
  - Use the target text file DifferentialCorrectorDC.data

  Script Syntax:

```
Create DifferentialCorrector DC;
GMAT DC.ShowProgress = true;
GMAT DC.ReportStyle = 'Normal';
GMAT DC.TargeterTextFile = 'DifferentialCorrectorDC.data';
GMAT DC.MaximumIterations = 25;
```

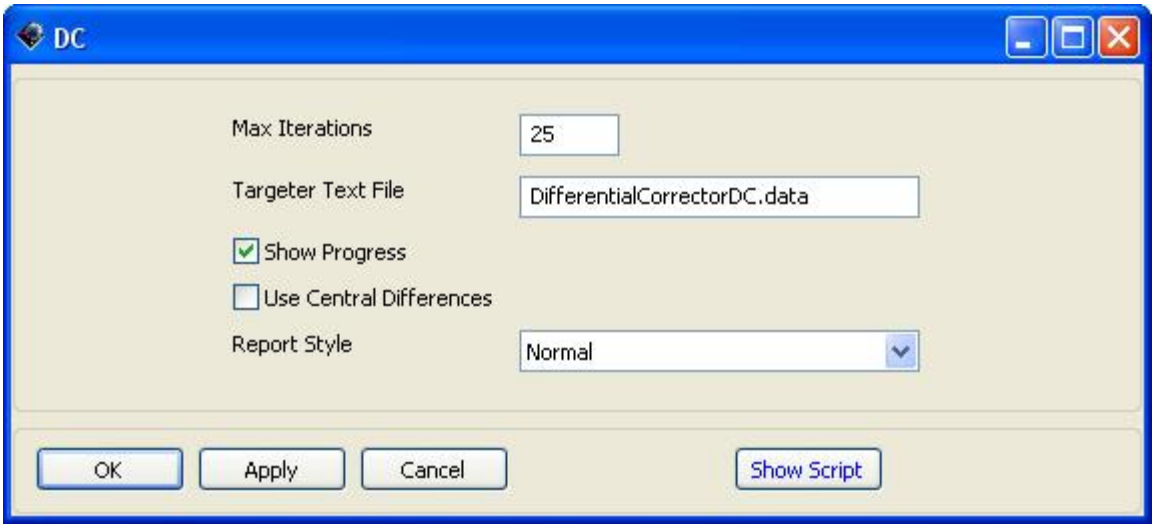


Figure: DC Differential Corrector Object

- Coordinate Systems
  - Add two new systems Sat1LVLH, Sat2VNB
  - Sat1LVLH
    - Type of Axes is ObjectReferenced
    - Origin of system is Sat1
    - Primary is Earth

    Script Syntax:

```
Create CoordinateSystem EarthMoonRot;
Create CoordinateSystem Sat1LVLH;
GMAT Sat1LVLH.Axes    = ObjectReferenced;
GMAT Sat1LVLH.Origin  = Sat1;
GMAT Sat1LVLH.Primary = Earth;
GMAT Sat1LVLH.YAxis   = -N;
GMAT Sat1LVLH.ZAxis   = -R;
```

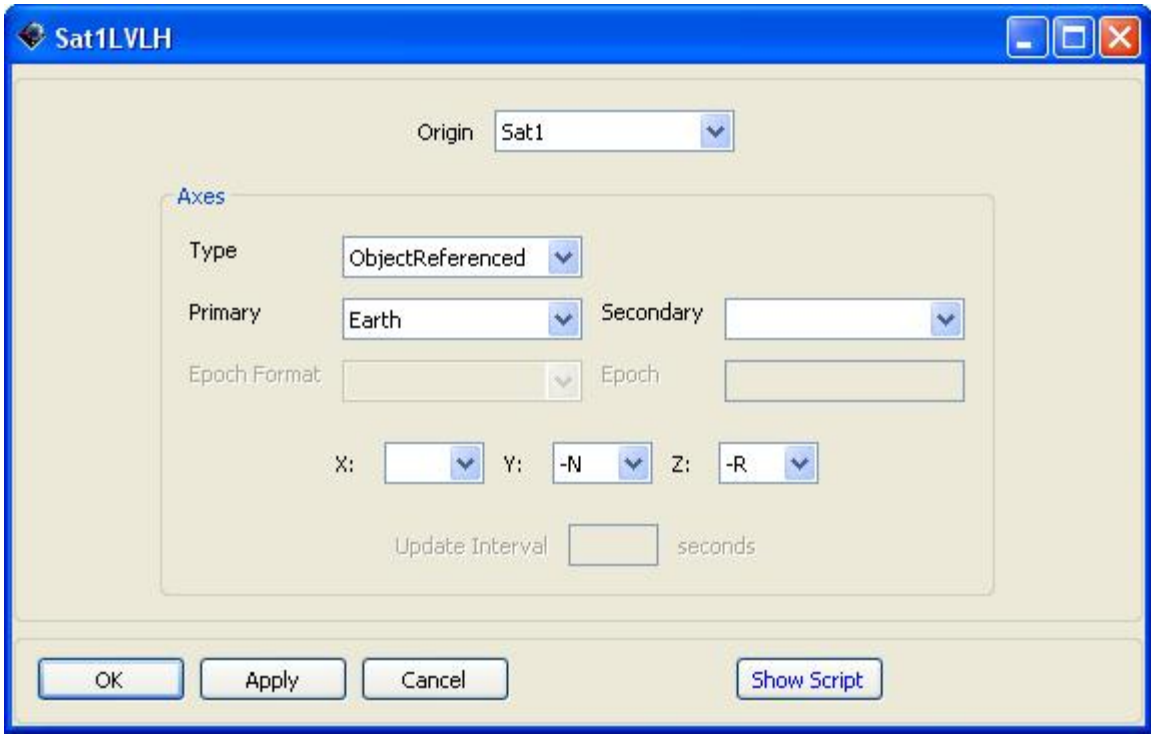


Figure: Sat1LVLH Coordinate System Object

- Sat2VNB
  - Type of Axes is ObjectReferenced
  - Origin of system is Sat2
  - Primary is Earth

  Script Syntax:

```
Create CoordinateSystem Sat2VNB;
GMAT Sat2VNB.Axes    = ObjectReferenced;
GMAT Sat2VNB.Origin  = Sat2;
GMAT Sat2VNB.Primary = Earth;
GMAT Sat2VNB.XAxis   = V;
GMAT Sat2VNB.YAxis   = N;
```

- Plots/Reports
  - Data Report saved to a file

  Script Syntax:

```
Create ReportFile DataReport;
GMAT DataReport.Filename = ./output/SampleMissions/Ex_FormationRendezvous.repor
GMAT DataReport.Precision = 16;
GMAT DataReport.WriteHeaders = On;
GMAT DataReport.ColumnWidth = 20;
```

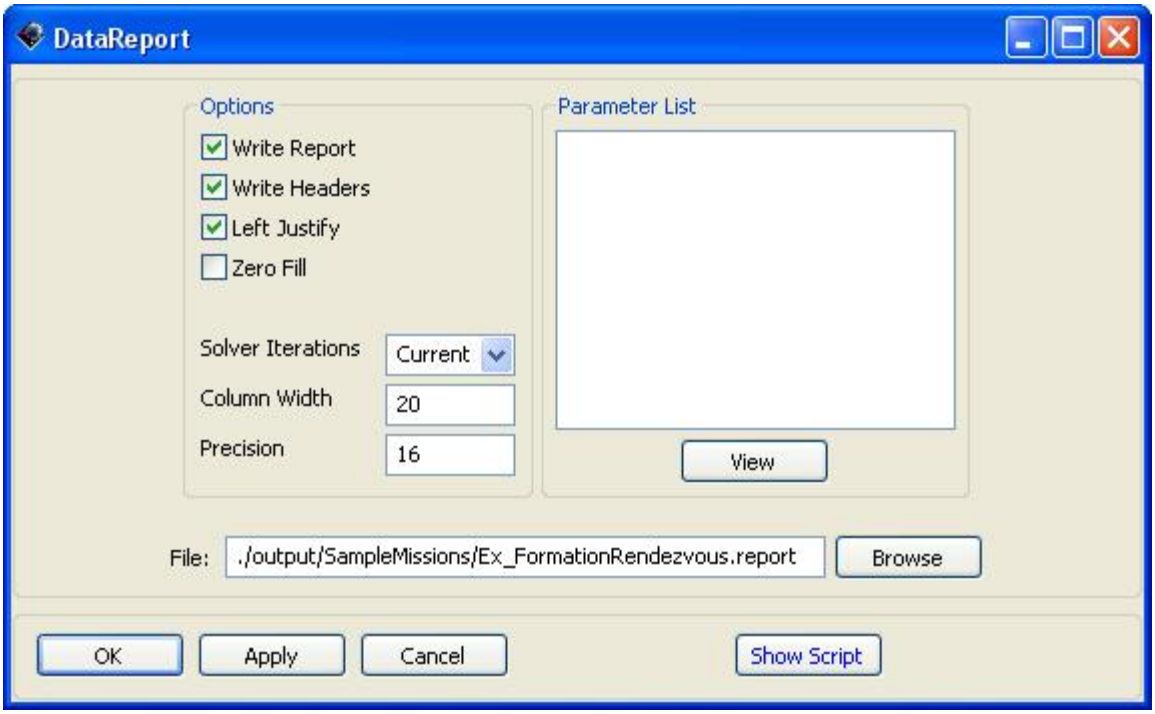


Figure: Data Report Object

- XY Plots
  - Add plots Sat2wrtSat1LVLH, Sat1wrtSat2VNB and Sat2wrtSat1VNB2
    - Sat2wrtSat1LVLH: XZ plots of Sat2 in Sat1VNB

      Script Syntax:

```
Create XYPlot Sat2wrtSat1LVLH;
GMAT Sat2wrtSat1LVLH.SolverIterations = None;
GMAT Sat2wrtSat1LVLH.IndVar   = Sat2.Sat1LVLH.X;
GMAT Sat2wrtSat1LVLH.Add      = Sat2.Sat1LVLH.Z;
GMAT Sat2wrtSat1LVLH.Grid     = On;
```

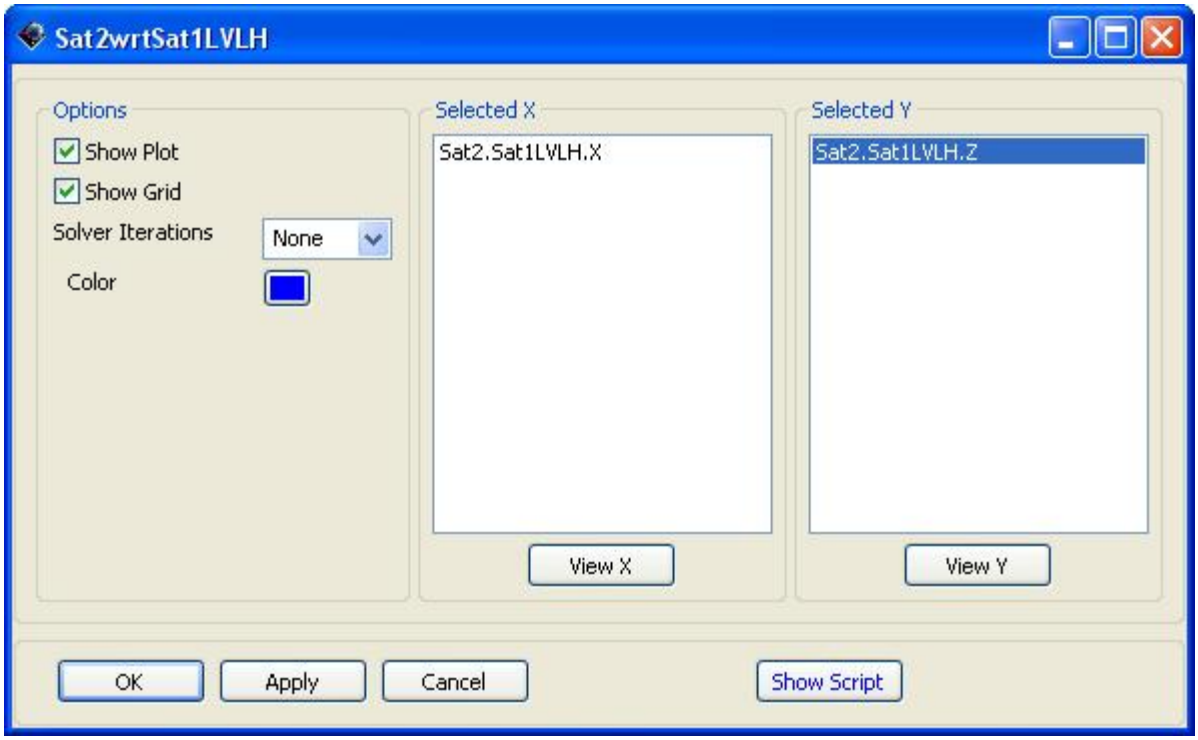


Figure: Sat2wrtSat1LVLH XY Plot Object

- Sat1wrtSat2VNB: XY plots of Sat1 in Sat2VNB

  Script Syntax:

  ```
  Create XYPlot Sat1wrtSat2VNB;
  GMAT Sat1wrtSat2VNB.SolverIterations = None;
  GMAT Sat1wrtSat2VNB.IndVar       = Sat1.Sat2VNB.X;
  GMAT Sat1wrtSat2VNB.Add          = Sat1.Sat2VNB.Y;
  GMAT Sat1wrtSat2VNB.Grid         = On;
  ```

- Sat2wrtSat1VNB2: YZ plots of Sat1 in Sat2VNB

  Script Syntax:

  ```
  Create XYPlot Sat1wrtSat2VNB2;
  GMAT Sat1wrtSat2VNB2.SolverIterations = None;
  GMAT Sat1wrtSat2VNB2.IndVar       = Sat1.Sat2VNB.Y;
  GMAT Sat1wrtSat2VNB2.Add          = Sat1.Sat2VNB.Z;
  GMAT Sat1wrtSat2VNB2.Grid         = On;
  ```

- OpenGL Plots
  - Add plot Sat2InSat1LVLH
  - Sat2InSat1LVLH
    - Add the Earth to the Selected Celestial Objects field

- Use a View Scale Factor of 0.15
- Select coordination system Sat1LVLH
- Select X as View Up Axis
- Select Earth as the View Point Reference and the View Direction body from those drop down menus

Script Syntax:

```
Create OpenGLPlot Sat2InSat1LVLH;
GMAT Sat2InSat1LVLH.Add = {Sat2, Sat3, Sat4};
GMAT Sat2InSat1LVLH.CoordinateSystem = Sat1LVLH;
GMAT Sat2InSat1LVLH.ViewPointReference = Earth;
GMAT Sat2InSat1LVLH.ViewDirection = Earth;
GMAT Sat2InSat1LVLH.ViewScaleFactor = .15;
GMAT Sat2InSat1LVLH.FixedFovAngle = 45;
GMAT Sat2InSat1LVLH.ViewUpCoordinateSystem = Sat1LVLH;
GMAT Sat2InSat1LVLH.ViewUpAxis = X;
GMAT Sat2InSat1LVLH.CelestialPlane = Off;
GMAT Sat2InSat1LVLH.XYPlane = On;
GMAT Sat2InSat1LVLH.WireFrame = Off;
GMAT Sat2InSat1LVLH.SolverIterations = All;
GMAT Sat2InSat1LVLH.Axes = Off;
GMAT Sat2InSat1LVLH.SunLine = Off;
GMAT Sat2InSat1LVLH.PerspectiveMode = Off;
GMAT Sat2InSat1LVLH.UseFixedFov = Off;
GMAT Sat2InSat1LVLH.DataCollectFrequency = 1;
GMAT Sat2InSat1LVLH.UpdatePlotFrequency = 50;
GMAT Sat2InSat1LVLH.NumPointsToRedraw = 0;
```

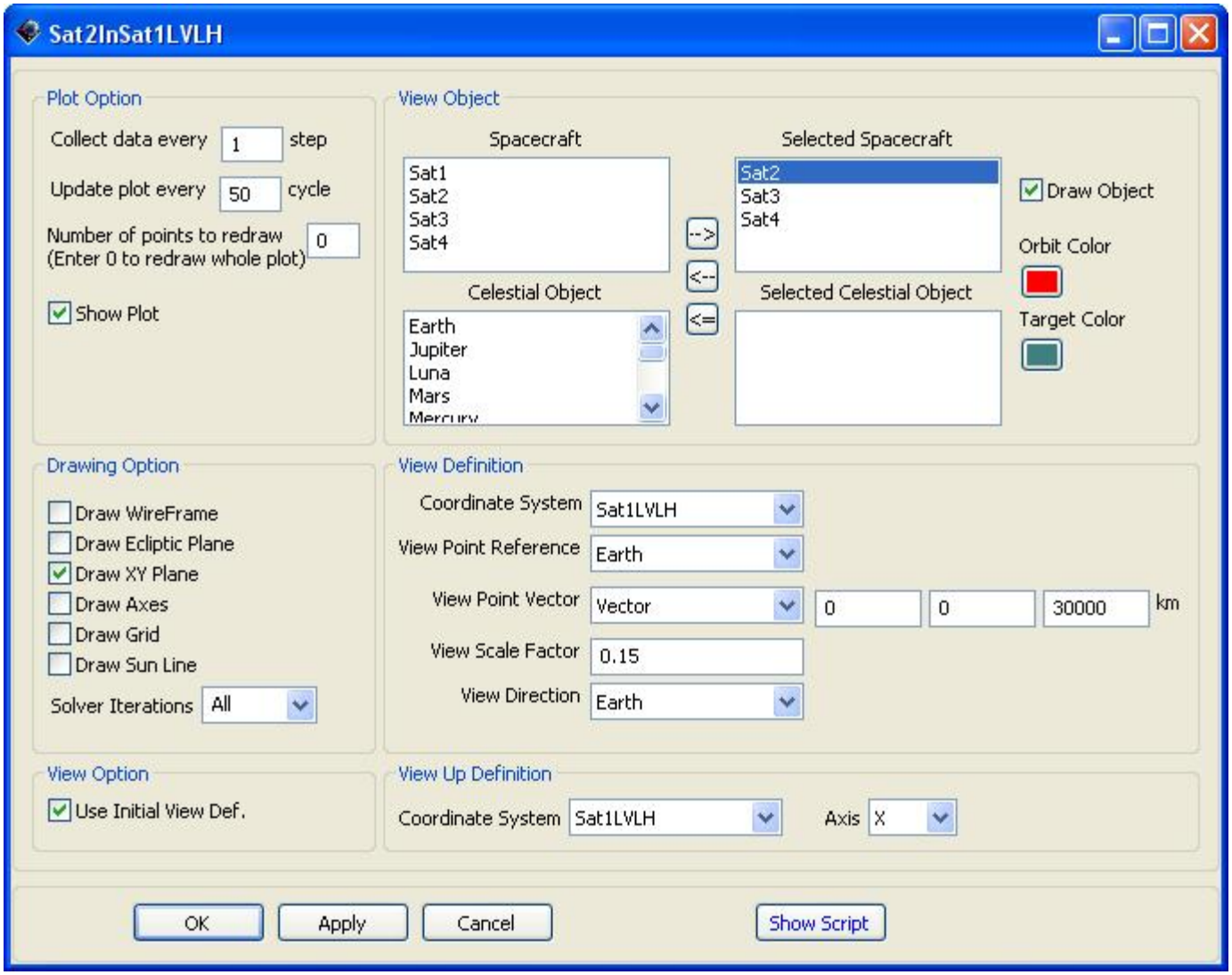


Figure: Sat2InSat1LVLH OpenGL Plot Object

Creating and Configuring the Mission Sequence

Image of Output Tree

This mission consists of first sequence, two targeting sequences. The first sequence includes some reports, an equation and a propagator, and the purpose is observation of the achievement of the formation flying under given initial conditions. Each targeting sequence also includes some reports, propagators, an equation and a maneuver, and the goal is the achievement of two formation rendezvous. The details of them are described below.

- The Target1 Sequence solves for three specified variables (?V, ?N and ?B) by applying Burn1 to Sat2 for accomplishing formation rendezvous which the positions of Sat2 would be same with of Sat1.

- The Target2 Sequence solves for three specified variables (?V, ?N and ?B) by applying Burn1 to Sat2 for accomplishing formation rendezvous which the velocities of Sat2 would be same with of Sat1.

Creating and Modifying Sequences

First Sequence

- Toggle1
  - Select Sat1wrtSat2VNB On
- Toggle2
  - Select Sat2InSat1LVLH On
- Toggle3
  - Select Sat1wrtSat2VNB2 On
- Toggle4
  - Select Sat2wrtSat1LVLHOn
- Report1
  - Add the Cartesian coordinates, X, Y, and Z, and all velocity, VX, VY, and VZ, to the parameters list to report them for Sat1
- Report2
  - Add the Cartesian coordinates, X, Y, and Z, and all velocity, VX, VY, and VZ, to the parameters list to report them for Sat2
- Report3
  - Add the Cartesian coordinates, X, Y, and Z, and all velocity, VX, VY, and VZ, to the parameters list to report them for Sat3
- Report4
  - Add the Cartesian coordinates, X, Y, and Z, and all velocity, VX, VY, and VZ, to the parameters list to report them for Sat4
- Equation1
  - DataReport.WriteHeaders = Off
- Propagator1
  - Use RKV_LowEarth propagator
  - Set the stopping condition parameter to 0.2 elapsed days
- Report5
  - Refer to Report1 for parameters list

- Report6
  - Refer to Report2 for parameters list
- Report7
  - Refer to Report3 for parameters list
- Report8
  - Refer to Report4 for parameters list

Script Syntax:

```
Toggle Sat1wrtSat2VNB On;
Toggle Sat2InSat1LVLH On;
Toggle Sat1wrtSat2VNB2 On;
Toggle Sat2wrtSat1LVLH On;

Report DataReport Sat1.X Sat1.Y Sat1.Z Sat1.VX Sat1.VY Sat1.VZ;
Report DataReport Sat2.X Sat2.Y Sat2.Z Sat2.VX Sat2.VY Sat2.VZ;
Report DataReport Sat3.X Sat3.Y Sat3.Z Sat3.VX Sat3.VY Sat3.VZ;
Report DataReport Sat4.X Sat4.Y Sat4.Z Sat4.VX Sat4.VY Sat4.VZ;

GMAT DataReport.WriteHeaders = Off;

Propagate RKV_LowEarth(Sat1, Sat2, Sat3, Sat4, {Sat1.ElapsedDays = .2});

Report DataReport Sat1.X Sat1.Y Sat1.Z Sat1.VX Sat1.VY Sat1.VZ;
Report DataReport Sat2.X Sat2.Y Sat2.Z Sat2.VX Sat2.VY Sat2.VZ;
Report DataReport Sat3.X Sat3.Y Sat3.Z Sat3.VX Sat3.VY Sat3.VZ;
Report DataReport Sat4.X Sat4.Y Sat4.Z Sat4.VX Sat4.VY Sat4.VZ;
```

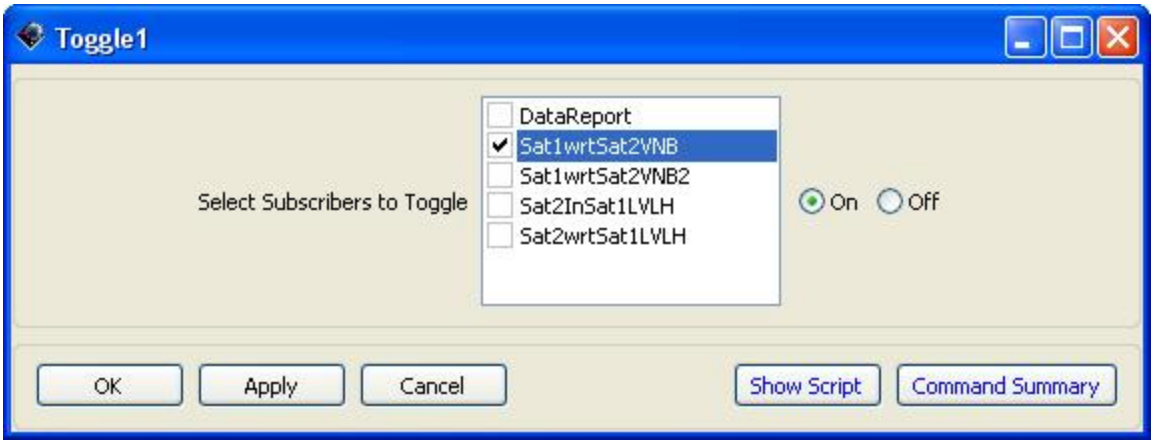


Figure: Toggle1 Toggle Command

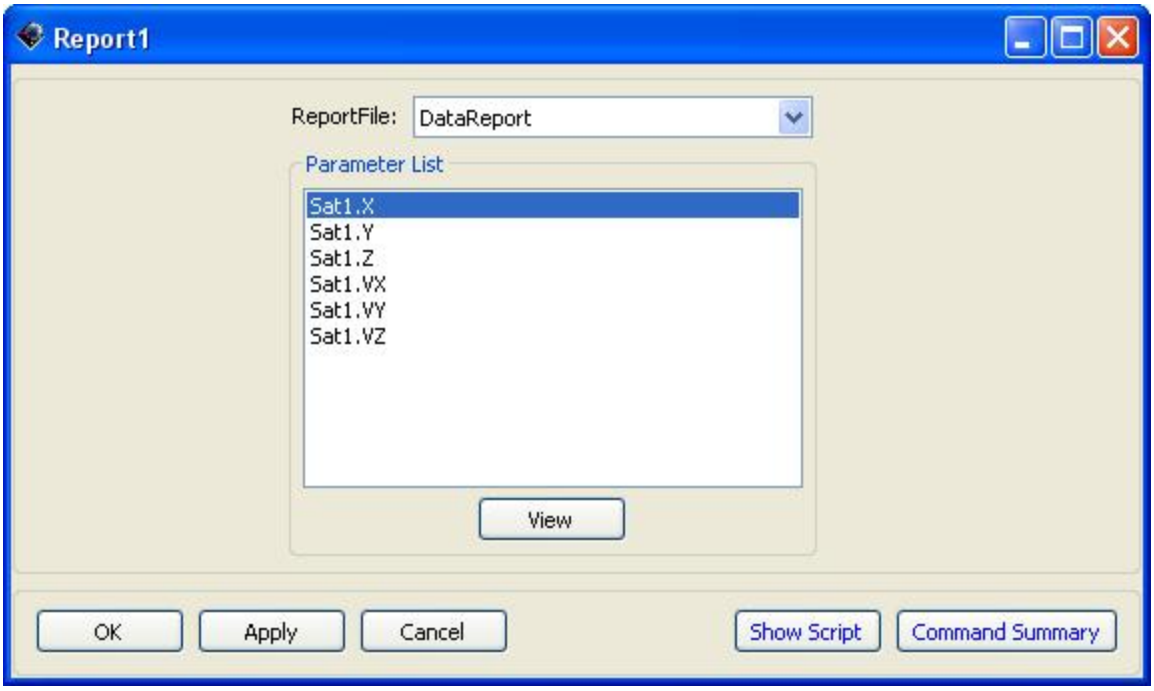


Figure: Report1 Report Command

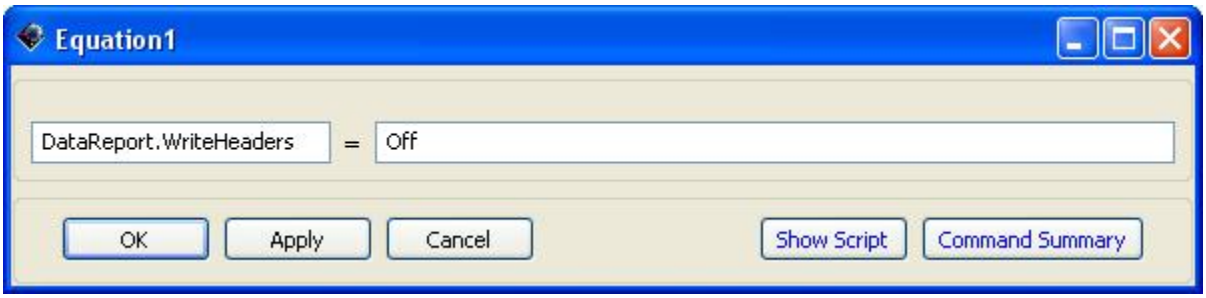


Figure: Equation1 Equation Command

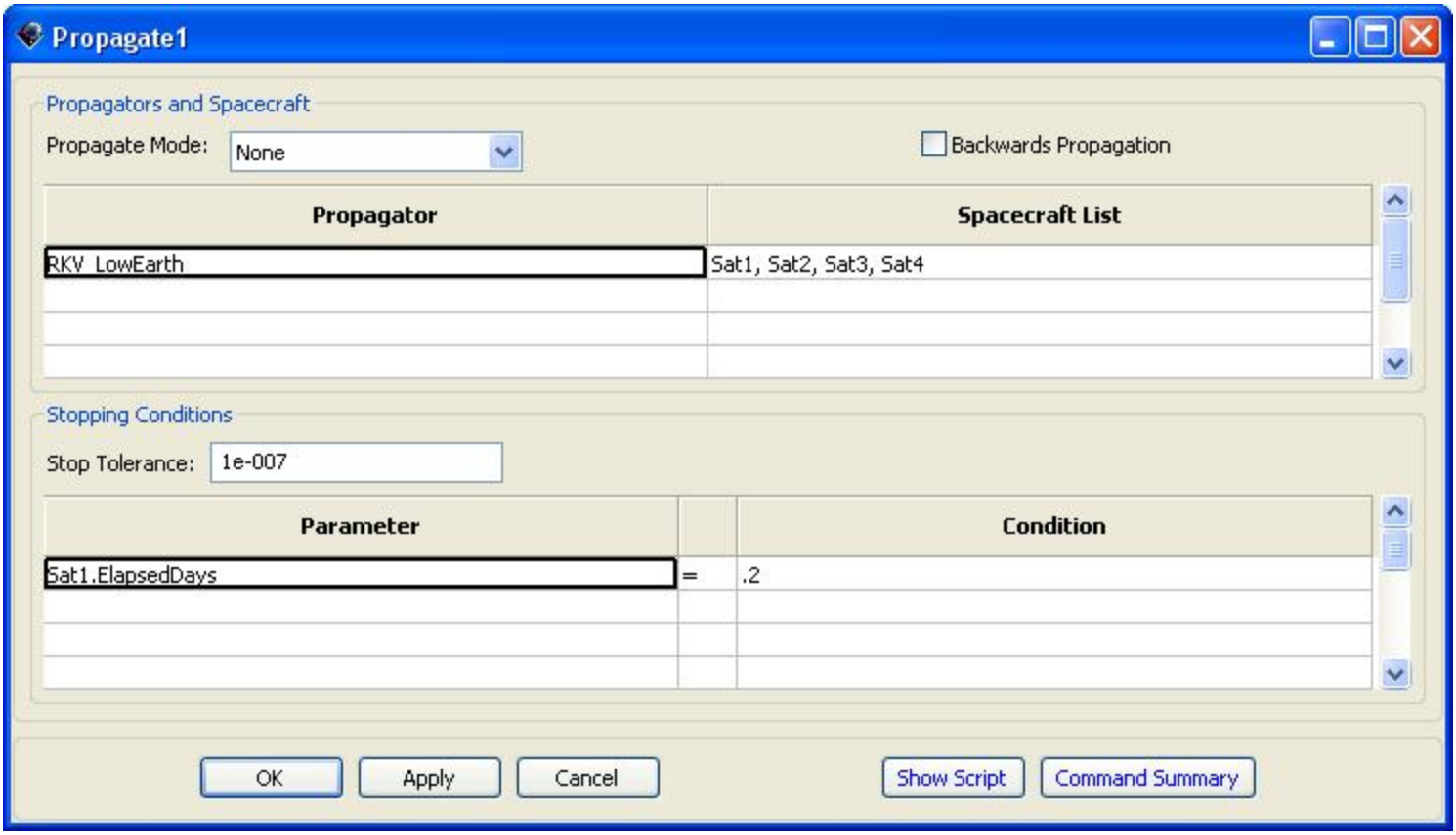


Figure: Propagator1 Propagate Command

Target1 Sequence

- Target1
  - Vary1
    - Set the variable V using Burn1
    - Initial Value = - 0.0001
    - Perturbation = 0.001
    - Lower (Limit) = - 0.3
    - Upper (Limit) = 0.3
    - Max Step = 0.05
  - Vary2
    - Set the variable N using Burn1
    - Initial Value = 0
    - Perturbation = 0.001

    - Lower (Limit) = - 0.3
    - Upper (Limit) = 0.3
    - Max Step = 0.05
- Vary3
    - Set the variable B using Burn1
    - Initial Value = 0
    - Perturbation = 0.001
    - Lower (Limit) = - 0.3
    - Upper (Limit) = 0.3
    - Max Step = 0.05
- Maneuver1
    - Utilizes the Burn1 for Sat2
- Propagate2
    - Use RKV_LowEarth propagator
    - Set the stopping condition parameter to be 5498.65387429 elapsed secs
- Achieve1
    - Goal
        - Select Sat1LVLH coordinate system
        - Sat2.X = 0
        - Tolerance = 0.0001
- Achieve2
    - Goal
        - Select Sat1LVLH coordinate system
        - Sat2.Y =0
        - Tolerance = 0.0001
- Achieve3
    - Goal
        - Select Sat1LVLH coordinate system
        - Sat2.Z = 0

    - Tolerance = 0.0001

- Report9
    - Refer to Report1 for parameters list
- Report10
    - Refer to Report2 for parameters list
- Report11
    - Refer to Report3 for parameters list
- Report12
    - Refer to Report4 for parameters list

Script Syntax:

```
Target DC;

     Vary DC(Burn1.V = 0.0001, {Perturbation = 0.0001, MaxStep = 0.05, Lower = -0
     Vary DC(Burn1.N = 0.000, {Perturbation = 0.0001, MaxStep = 0.05, Lower = -0
     Vary DC(Burn1.B = 0.000, {Perturbation = 0.0001, MaxStep = 0.05, Lower = -0

     Maneuver Burn1(Sat2);
     Propagate  RKV_LowEarth(Sat1, Sat2, Sat3, Sat4, {Sat1.ElapsedSecs = 5498.653

     Achieve DC(Sat2.Sat1LVLH.X = 0, {Tolerance = 0.0001});
     Achieve DC(Sat2.Sat1LVLH.Y = 0, {Tolerance = 0.0001});
     Achieve DC(Sat2.Sat1LVLH.Z = 0, {Tolerance = 0.0001});

EndTarget;

Report DataReport Sat1.X Sat1.Y Sat1.Z Sat1.VX Sat1.VY Sat1.VZ;
Report DataReport Sat2.X Sat2.Y Sat2.Z Sat2.VX Sat2.VY Sat2.VZ;
Report DataReport Sat3.X Sat3.Y Sat3.Z Sat3.VX Sat3.VY Sat3.VZ;
Report DataReport Sat4.X Sat4.Y Sat4.Z Sat4.VX Sat4.VY Sat4.VZ;
```

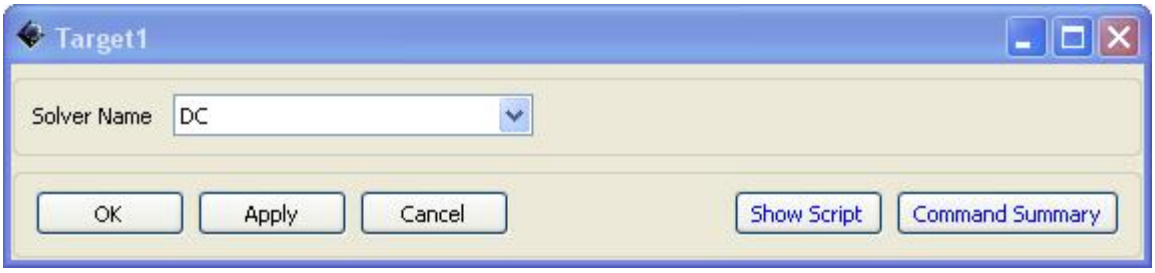


Figure: Target1 Target Command

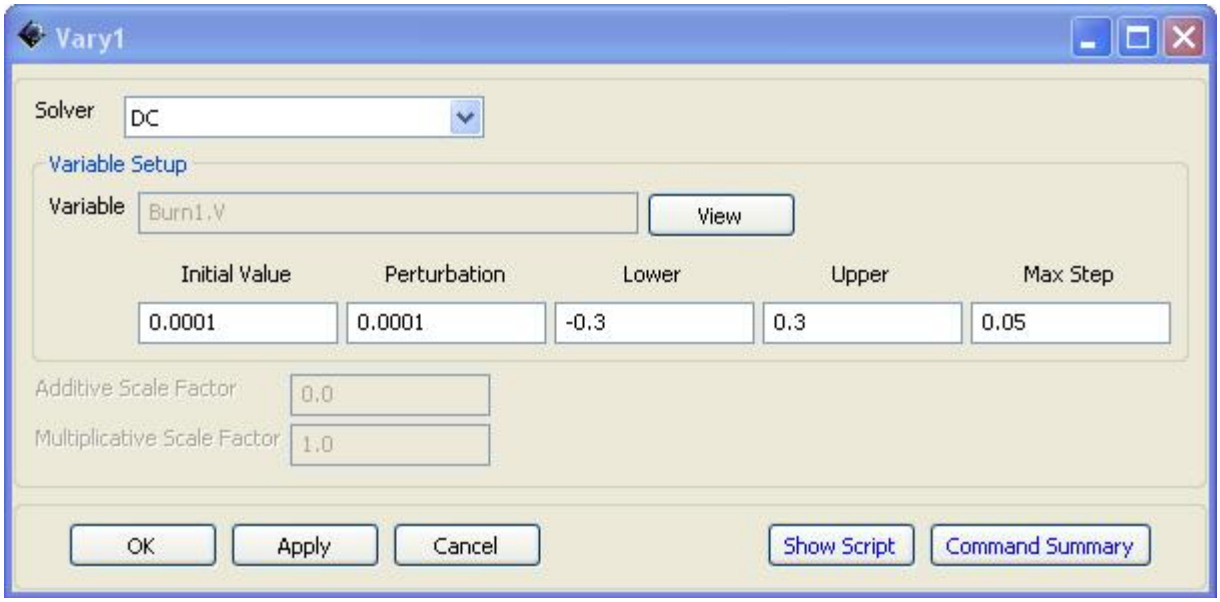


Figure: Vary1 Vary Command

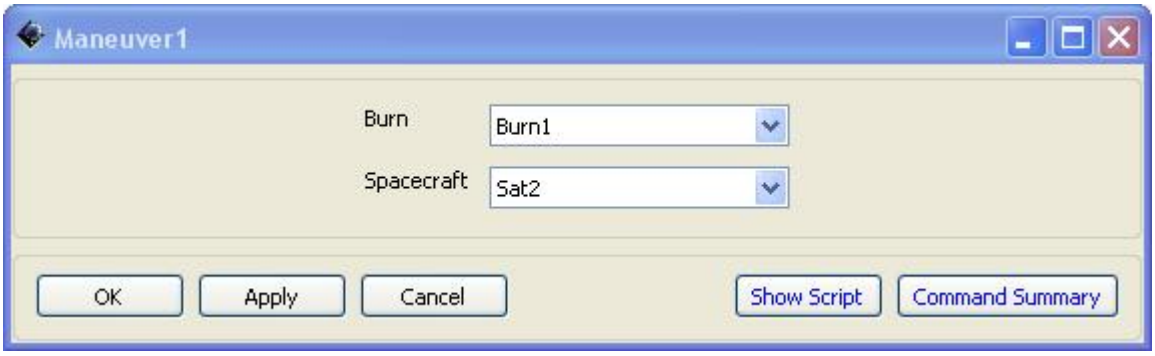


Figure: Maneuver1 Maneuver Command

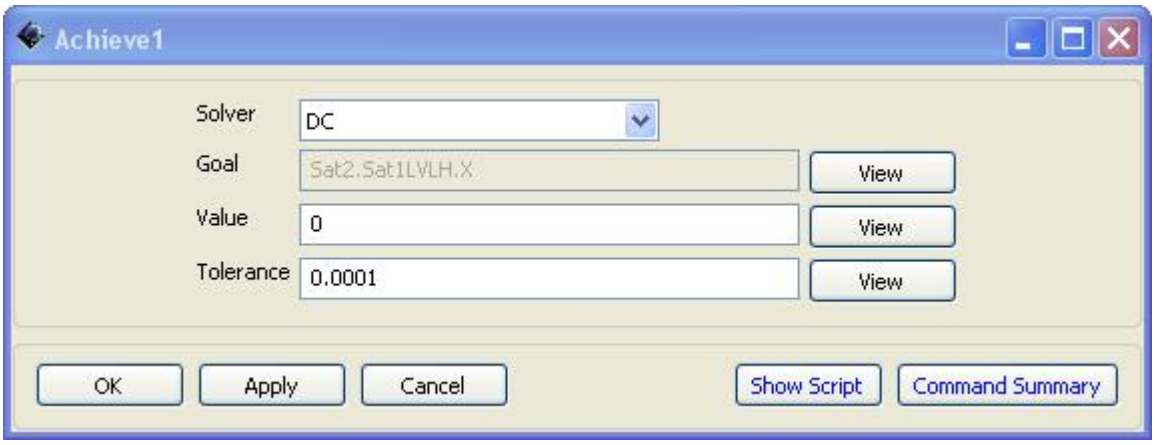


Figure: Achieve1 Achieve Command

Target2 Sequence

- Target2

- Vary4
  - Set the variable V using Burn1
  - Initial Value = - 0.0001
  - Perturbation = 0.001
  - Lower (Limit) = - 1.0
  - Upper (Limit) = 1.0
  - Max Step = 0.5 km/s
- Vary5
  - Set the variable N using Burn1
  - Initial Value = 0
  - Perturbation = 0.001
  - Lower (Limit) = - 1.0
  - Upper (Limit) = 1.0
  - Max Step = 0.5
- Vary6
  - Set the variable B using Burn1
  - Initial Value = 0
  - Perturbation = 0.001
  - Lower (Limit) = - 1.0
  - Upper (Limit) = 1.0
  - Max Step = 0.5
- Maneuver2
  - Utilizes the Burn1 for Sat2
- Propagate3
  - Use RKV_LowEarth propagator
  - Set the stopping condition parameter to 5498.65387429 elapsed secs
- Achieve4
  - Goal
    - Sat2.VX = Sat1.VX

    - Tolerance = 0.0001
- Achieve5
  - Goal
    - Sat2.VY = Sat1.VY
    - Tolerance = 0.0001
- Achieve6
  - Goal
    - Sat2.VZ = Sat1.VZ
    - Tolerance = 0.0001
- Report13
  - Refer to Report1 for parameters list
- Report14
  - Refer to Report2 for parameters list
- Report15
  - Refer to Report3 for parameters list
- Report16
  - Refer to Report4 for parameters list
- Propagate4
  - Use RKV_LowEarth propagator
  - Set the stopping condition parameter to 0.2 elapsed days
- Report17
  - Refer to Report1 for parameters list
- Report18
  - Refer to Report2 for parameters list
- Report19
  - Refer to Report3 for parameters list
- Report20
  - Refer to Report4 for parameters list

```
Target DC;

     Vary DC(Burn1.V = 0.0001, {Perturbation = 0.0001, MaxStep = 0.5, Lower = -1
     Vary DC(Burn1.N = 0.000, {Perturbation = 0.0001, MaxStep = 0.5, Lower = -1.0
     Vary DC(Burn1.B = 0.000, {Perturbation = 0.0001, MaxStep = 0.5, Lower = -1.0

     Maneuver Burn1(Sat2);

     Propagate  RKV_LowEarth(Sat1, Sat2, Sat3, Sat4, {Sat1.ElapsedSecs = 5498.653

     Achieve DC(Sat2.VX = Sat1.VX, {Tolerance = 0.0001});
     Achieve DC(Sat2.VY = Sat1.VY, {Tolerance = 0.0001});
     Achieve DC(Sat2.VZ = Sat1.VZ, {Tolerance = 0.0001});

EndTarget;

Report DataReport Sat1.X Sat1.Y Sat1.Z Sat1.VX Sat1.VY Sat1.VZ;
Report DataReport Sat2.X Sat2.Y Sat2.Z Sat2.VX Sat2.VY Sat2.VZ;
Report DataReport Sat3.X Sat3.Y Sat3.Z Sat3.VX Sat3.VY Sat3.VZ;
Report DataReport Sat4.X Sat4.Y Sat4.Z Sat4.VX Sat4.VY Sat4.VZ;

Propagate RKV_LowEarth(Sat1, Sat2, Sat3, Sat4, {Sat1.ElapsedDays = .2});

Report DataReport Sat1.X Sat1.Y Sat1.Z Sat1.VX Sat1.VY Sat1.VZ;
Report DataReport Sat2.X Sat2.Y Sat2.Z Sat2.VX Sat2.VY Sat2.VZ;
Report DataReport Sat3.X Sat3.Y Sat3.Z Sat3.VX Sat3.VY Sat3.VZ;
Report DataReport Sat4.X Sat4.Y Sat4.Z Sat4.VX Sat4.VY Sat4.VZ;
```

Running the Mission

Image of Output Tree

According to this tutorial, you complete this mission program. Firstly please save all, then click green arrow which indicates running simulation. So, by these steps you can create the following figures as outputs.

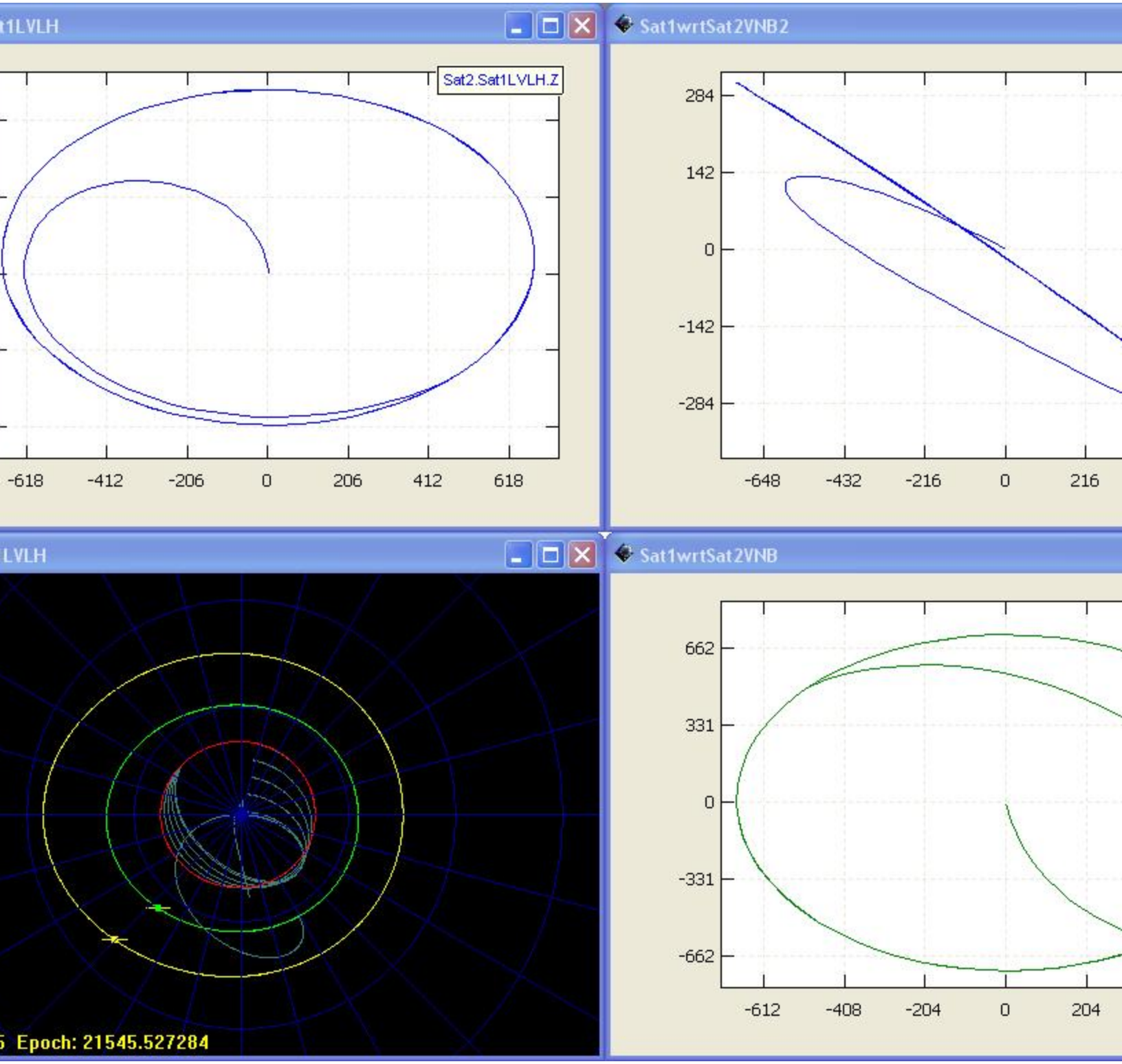


Figure: Output

Chapter 21. Minimum Fuel for Orbit Transfer

This tutorial was started from the default settings in the July 2, 2008 build of GMAT.

Objective and Overview

This tutorial shows how GMAT's optimizer can be used in a mission setting. This mission basically shows how a Hohmann transfer is the most efficient to go from one orbit to the other. While this isn't entirely a Hohmann transfer since it also has a plane change, it generally abides by its rules. Again, this tutorial will show how the optimize, optimizer, vary command using the optimizer, nonlinear constraints, and minimize command. To accomplish this, the transfer orbit insertion burn and the final orbit insertion burn will each vary in all directions. Additionally, the time that the spacecraft flies in the transfer orbit and the time/point that it leaves the initial orbit are both varied as well.

Download the script file: Ex_LowEarthMinFuelTransfer.script [scripts/Ex_LowEarthMinFuelTransfer.script]

Note: This mission is the same as the Optimization sample mission largely, except the Optimization mission has a few errors in it and does not contain XY Plots

Prerequisites

Mission Description

- Objective:
  - SMA=10,000
  - ECC=.05
  - INC=5
  - Minimum Fuel Required to Achieve Above Objectives
- Assume:
  - SMA = 7500
  - ECC = 0
  - INC = 0
  - RAAN = 45
  - AOP = 0
  - TA = 360
- Find:
  - Epoch of Transfer Orbit Insertion Burn

- Epoch of Final Orbit Insertion Burn
- Transfer Orbit Insertion Burn
- Final Orbit Insertion Burn

Resource, Mission, and Output trees

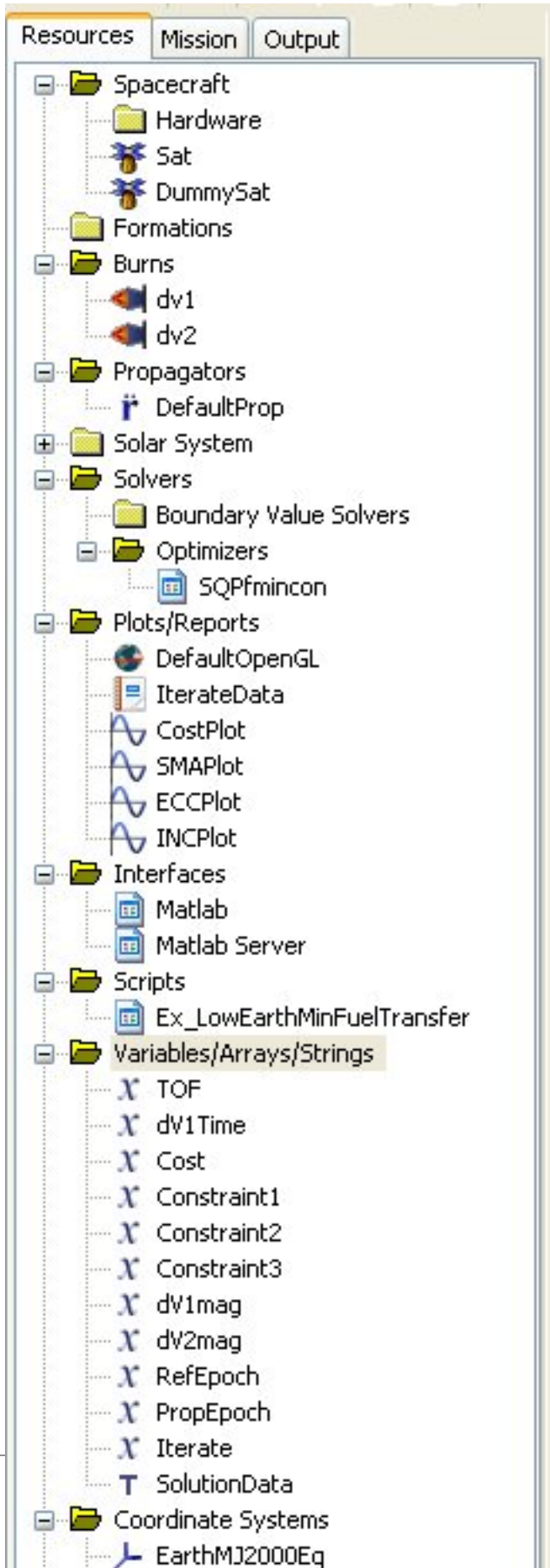


Figure: Minimum Fuel Orbit Transfer Resource, Mission, and Output trees

Creating and Configuring the Resource Tree

Two spacecraft will be required, one for actual propagation and the other is a dummy satellite whose conditions we don't care about but For this mission, two burns will be required. One burn will transfer the spacecraft from its initial orbit to its transfer orbit. The second burn will transfer the spacecraft from its transfer orbit to its final orbit. Only one propagator is needed since the spacecraft is staying close to Earth. Only one OpenGL plot will be used. One report file will be used to store the final numbers of the burns and to see how closely it achieves the desired goals. Four XY plots will be used to observe how quickly convergence is achieved for the objectives of the mission. Also an optimizer will be used instead of a differential corrector. Also a number of variables will be used.

Objects Required

- Spacecraft: Sat, DummySat
- Burns: dV1, dV2
- Propagators: DefaultProp
- Optimizer: SPQfmincon
- Plots/Reports: DefaultOpenGL, IterateData, CostPlot, SMAPlot, ECCPlot, INCPlot
- Coordinate Systems:EarthMJ2000Eq
- Variables: TOF dV1Time Cost Constraint1 Constraint2 Constraint3, dV1mag dV2mag RefEpoch PropEpoch Iterate

Creating and Modifying Objects

- Spacecraft
  - Sat
    - SMA = 7500
    - ECC = 0
    - INC = 0
    - RAAN = 45
    - AOP = 0
    - TA = 360

    Script Syntax:

```
Create Spacecraft Sat;
GMAT Sat.DisplayStateType = Keplerian;
GMAT Sat.SMA = 7500;
GMAT Sat.ECC = 0;
GMAT Sat.INC = 0;
GMAT Sat.RAAN = 45;
```

```
GMAT Sat.AOP = 0;
GMAT Sat.TA = 360;
```

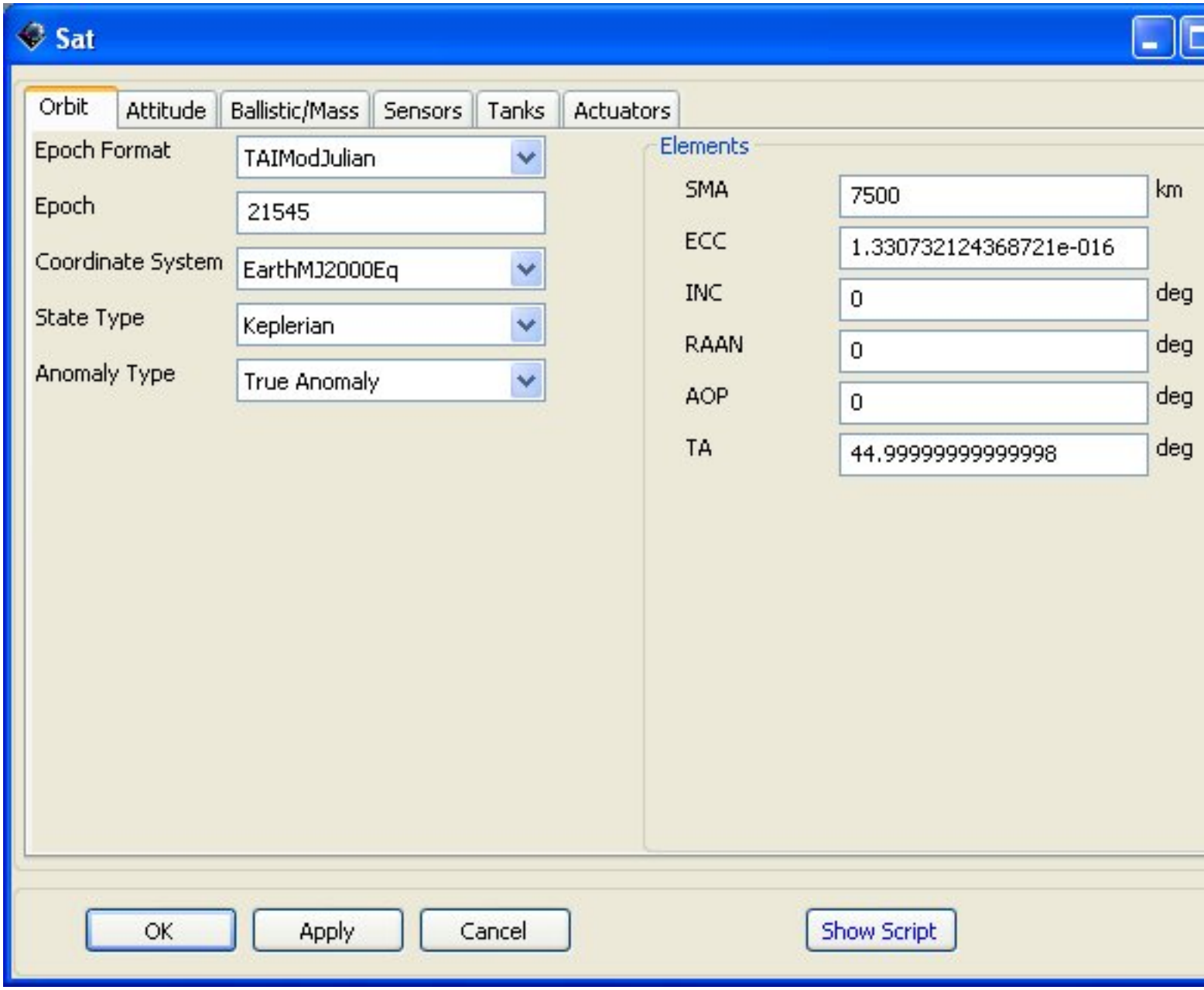


Figure: Sat Spacecraft Object'

- DummySat
  - Default Conditions

  Script Syntax:

```
Create Spacecraft DummySat;
GMAT Sat.DisplayStateType = Cartesian;
```

- Propagators
  - DefaultProp
    - It uses all of the default values except it has no primary body, but rather uses a point force model of the Earth with no drag. If it was desired to make it more accurate, drag, J2, solar radiation, and other celestial bodies could be added to the propagator.

    Script Syntax:

    ```
    Create ForceModel DefaultProp_ForceModel;
    GMAT DefaultProp_ForceModel.CentralBody = Earth;
    GMAT DefaultProp_ForceModel.PointMasses = {Earth};
    GMAT DefaultProp_ForceModel.Drag = None;
    GMAT DefaultProp_ForceModel.SRP = Off;
    GMAT DefaultProp_ForceModel.ErrorControl = RSSStep;

    Create Propagator DefaultProp;
    ```

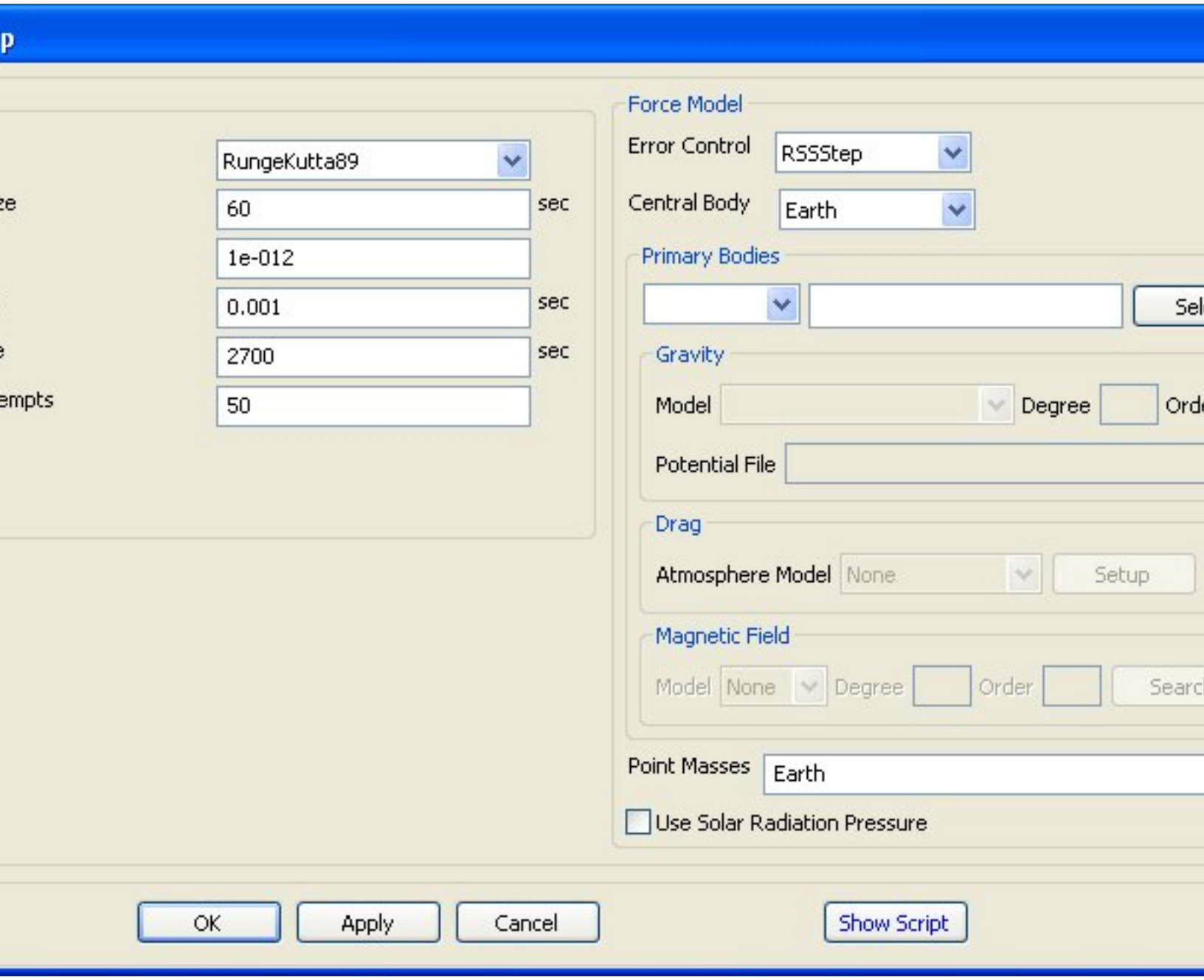


Figure: DefaultProp Propagator Object

- Burns
  - dV1
    - Default Values
  - dV2
    - Default Values

Script Syntax:

```
Create ImpulsiveBurn dv1;
GMAT dv1.Origin = Earth;
GMAT dv1.Axes   = VNB;

Create ImpulsiveBurn dv2;
GMAT dv2.Origin = Earth;
GMAT dv2.Axes   = VNB;
```

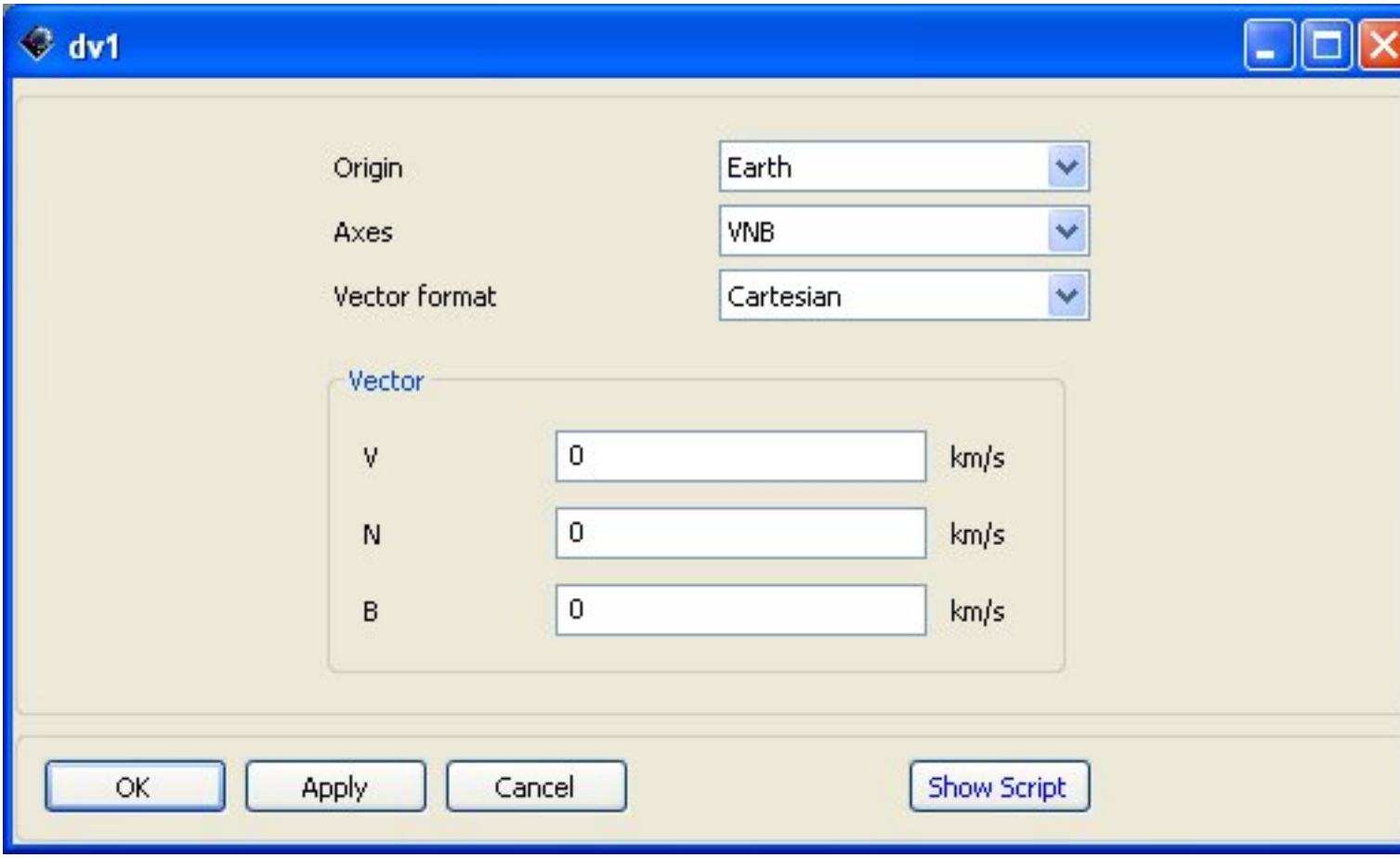


Figure: dv1 Burn Object

- Variables/Strings/Arrays
  - Variable TOF
  - Variable dV1 Time
  - Variable Cost
  - Variable Constraint1
  - Variable Constraint2

- Variable Constraint3
- Variable dV1mag
- Variable dV2mag
- Variable RefEpoch
- Variable PropEpoch
- Variable Iterate

Script Syntax:

```
Create Variable TOF dV1Time Cost Constraint1 Constraint2 Constraint3, dV1mag dV2r
```

- Coordinate Systems
  - Use Default coordinate systems
- Solvers
  - Optimizer fmincon
    - DiffMaxChange = .01
    - DiffMinChange = .00001
    - MaxFunEvals = 1000
    - MaxIter = 100
    - TolX = .001
    - TolFun = .00001
    - TolCon = .00001
    - Display = Iter
    - Diagnostics = On

    Script Syntax:

```
Create FminconOptimizer SQPfmincon
GMAT SQPfmincon.DiffMaxChange = 0.1;    % Real number
GMAT SQPfmincon.DiffMinChange = 0.00001;  % Real number
GMAT SQPfmincon.MaxFunEvals   = 1000;    % Real number
GMAT SQPfmincon.MaxIter       = 100;     % Real number
GMAT SQPfmincon.TolX          = 0.001;    % Real number
GMAT SQPfmincon.TolFun        = 0.00001;  % Real number
GMAT SQPfmincon.TolCon        = 0.00001;  % Real number
GMAT SQPfmincon.DerivativeCheck = Off;   % {On, Off}
GMAT SQPfmincon.Diagnostics     = On;    % {On, Off}
GMAT SQPfmincon.Display         = Iter   % {Iter, Off, Notify, Final}
GMAT SQPfmincon.GradObj         = Off;   % {On, Off}
GMAT SQPfmincon.GradConstr      = Off;   % {On, Off}
```

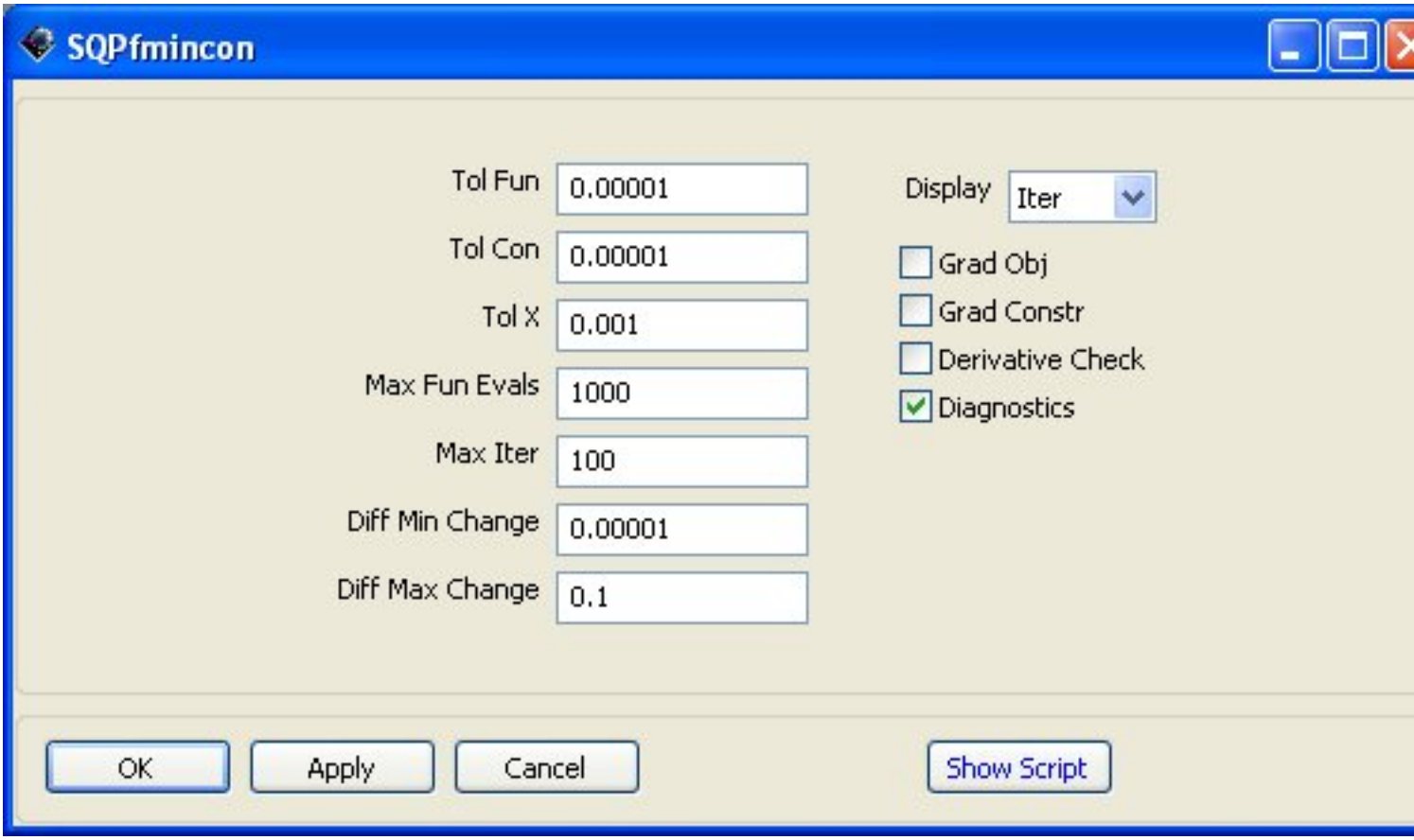


Figure: SQPfmincon Boundary Value Solver Object

- Plots/Reports
  - XY Plot CostPlot
    - Inde = Iterate
    - Add = Cost
  - XY Plot SMAPlot
    - Inde = Iterate
    - Add = SMA
  - XY Plot ECCPlot
    - Inde = Iterate
    - Add = ECC
  - XY Plot INCPlot
    - Inde = Iterate

- Add = INC

Script Syntax:

```
Create XYPlot CostPlot
CostPlot.IndVar = Iterate;
CostPlot.Add    = Cost;

Create XYPlot SMAPlot
SMAPlot.IndVar = Iterate;
SMAPlot.Add    = Sat.SMA;

Create XYPlot ECCPlot
ECCPlot.IndVar = Iterate;
ECCPlot.Add    = Sat.ECC;

Create XYPlot INCPlot
INCPlot.IndVar = Iterate;
INCPlot.Add    = Sat.INC;
```

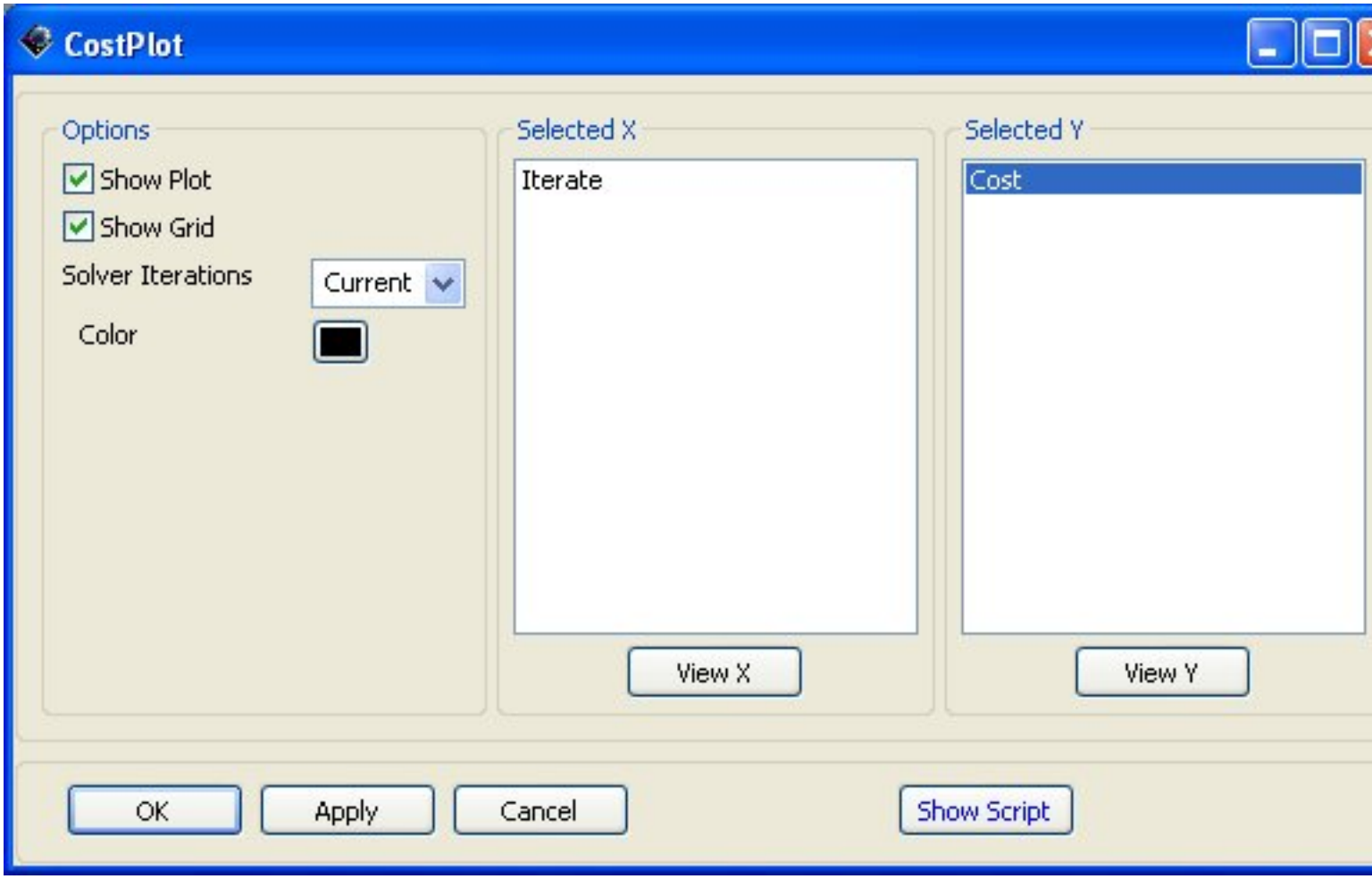


Figure: Cost XY Plot Object

- Report IterateDate
  - Filename = ./output/SampleMissions/Ex_LowEarthMinFuelTransfer.report

  Script Syntax:

```
Create ReportFile IterateData;
GMAT IterateData.Filename = ./output/SampleMissions/Ex_LowEarthMinFuelTransfer.
GMAT IterateData.Precision = 10;
GMAT IterateData.WriteHeaders = On;
GMAT IterateData.ColumnWidth = 16;
```

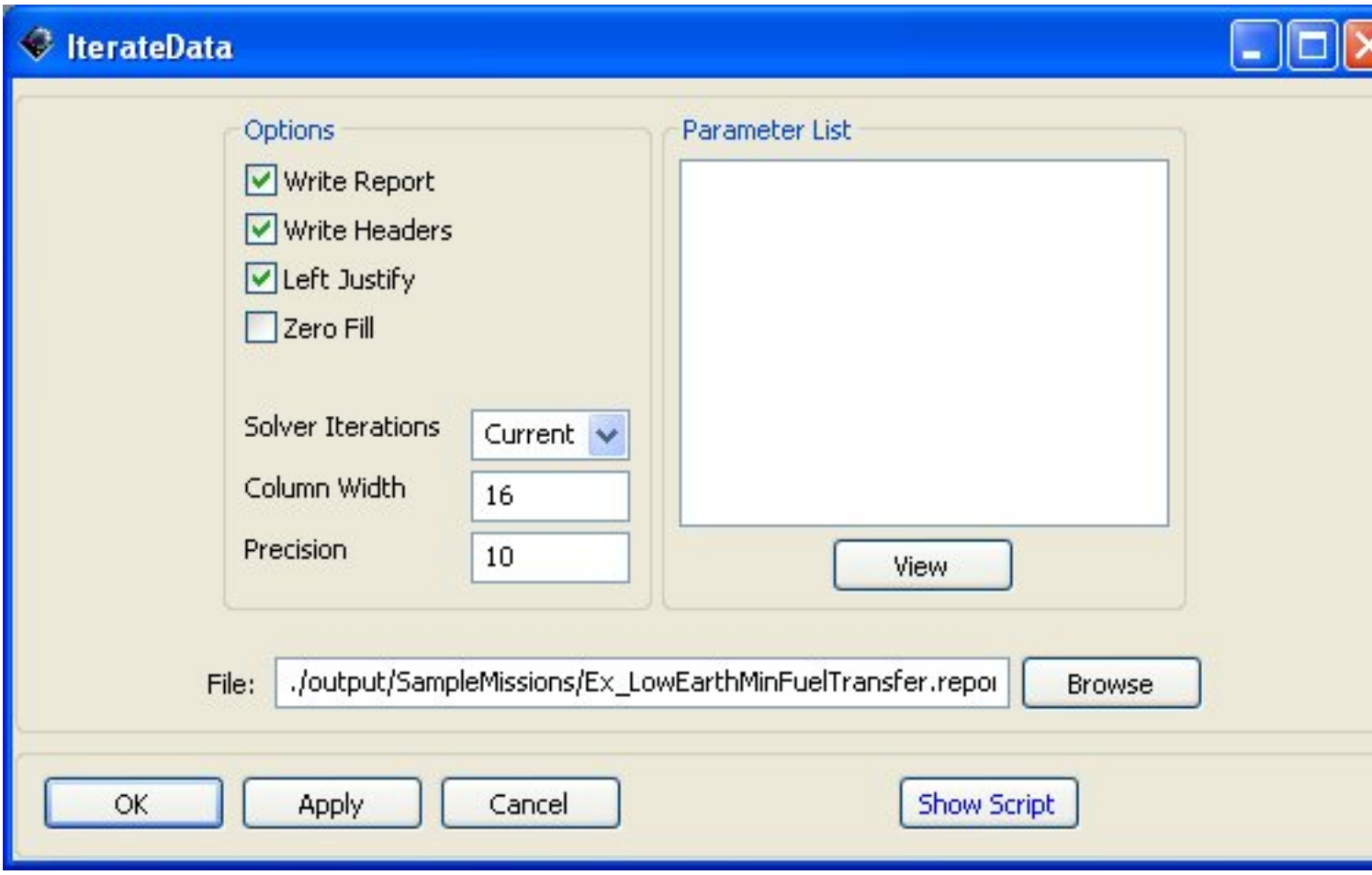


Figure: IterateData Report Object

- DefaultOpenGL
  - Collect Data every 6 steps
  - Number of Points to redraw: 100
  - Selected Celestial Objects: Earth
  - View Up Definition Axis: X

- SolverIterations: All

Script Syntax:

```
Create OpenGLPlot DefaultOpenGL;
GMAT DefaultOpenGL.Add = {Sat, Earth};
GMAT DefaultOpenGL.CoordinateSystem = EarthMJ2000Eq;
GMAT DefaultOpenGL.ViewPointReference = Earth;
GMAT DefaultOpenGL.ViewDirection = Earth;
GMAT DefaultOpenGL.ViewScaleFactor = 1;
GMAT DefaultOpenGL.FixedFovAngle = 45;
GMAT DefaultOpenGL.ViewUpCoordinateSystem = EarthMJ2000Eq;
GMAT DefaultOpenGL.ViewUpAxis = X;
GMAT DefaultOpenGL.CelestialPlane = Off;
GMAT DefaultOpenGL.XYPlane = On;
GMAT DefaultOpenGL.WireFrame = Off;
GMAT DefaultOpenGL.SolverIterations = All;
GMAT DefaultOpenGL.Axes = Off;
GMAT DefaultOpenGL.Grid = Off;
GMAT DefaultOpenGL.SunLine = Off;
GMAT DefaultOpenGL.UseInitialView = On;
GMAT DefaultOpenGL.PerspectiveMode = Off;
GMAT DefaultOpenGL.UseFixedFov = Off;
GMAT DefaultOpenGL.DataCollectFrequency = 6;
GMAT DefaultOpenGL.UpdatePlotFrequency = 50;
GMAT DefaultOpenGL.NumPointsToRedraw = 100;
GMAT DefaultOpenGL.ShowPlot = true;
```

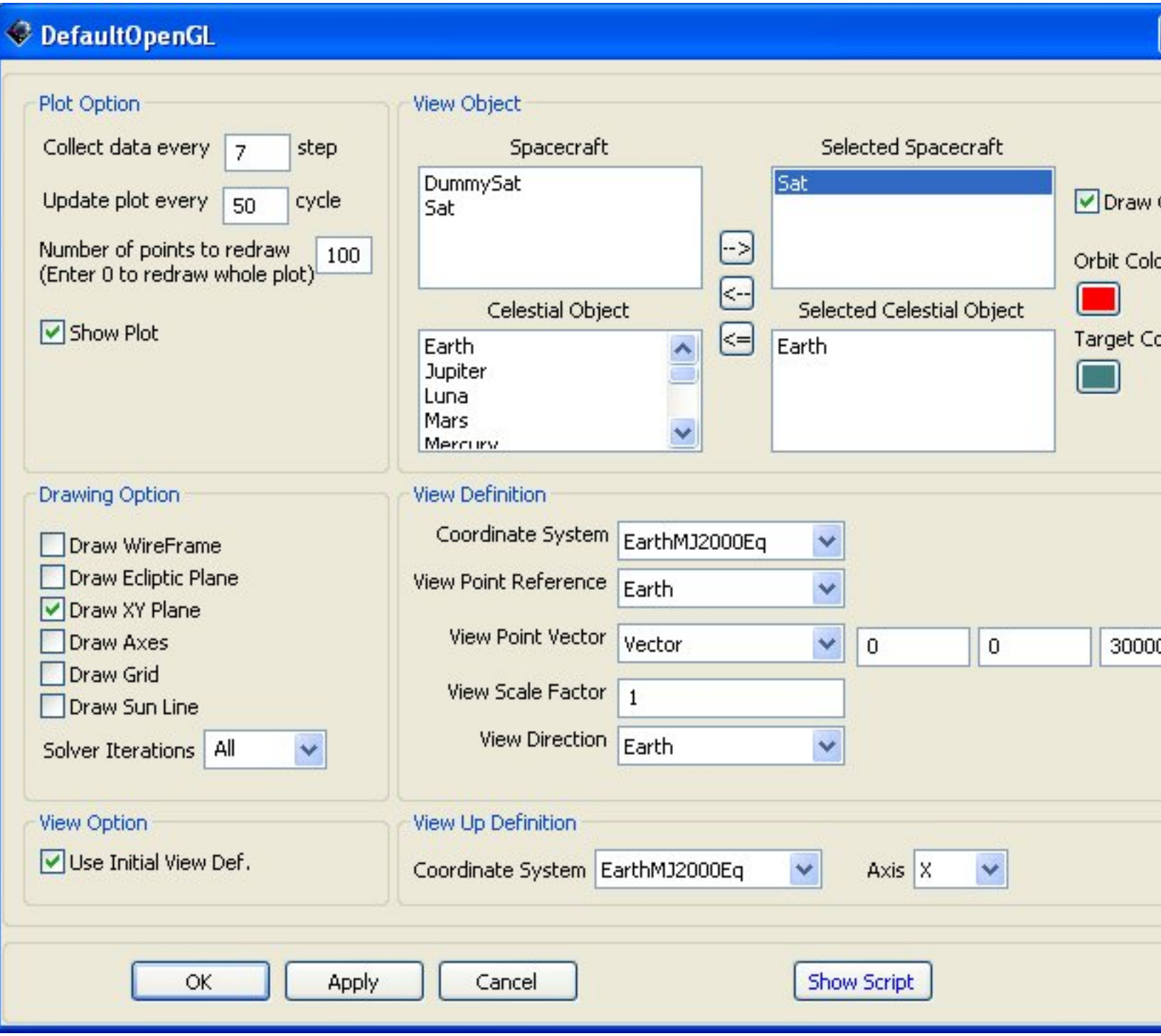


Figure: DefaultOpenGL System Object

Creating and Configuring the Mission Tree

Resource Mission and Output Trees

The spacecraft propagates a single orbit and then the optimizing sequence starts. First the time of the transfer orbit insertion burn is varied to determine the best time to start the orbit insertion. The two if statements depend on if the time is negative or positive. If it is positive, the first if statement allows the spacecraft to propagate forward in time to that point. If it is negative, the second if statement allows the spacecraft to propagate backwards in time to that point. Next the three directions of the first burn are varied in magniture. Then the maneuever is performed and the amount of time the spacecraft is on that transfer orbit is varied. Then the magnitude of the three directions of the second transfer burn

Creating and Modifying Command Sequences

Pre-Optimization Sequence

- Firstly, the four XY plots are turned off so that they can only take data when we want them to be. If they were left on, they would take the wrong data. We want them to take a measurement at the end of the iteration so we know how close each iteration is to the final value.
- The propagate command is then to make the spacecraft orbit one period so that we can see what the original orbit looked like.
- In Equation2, the counter variable is set to 0 before the optimization sequence starts.
- Toggle CostPlot Off;
- Toggle SMAPlot Off;
- Toggle ECCPlot Off;
- Toggle INCPlot Off;
- Propagate Sat for Elapsed Seconds = Sat . Period
- Equation1 RefEpoch = Sat.A1ModJulian
- Equation2 Iterate = 0

Script Syntax:

```
Toggle CostPlot Off;
Toggle SMAPlot Off;
Toggle ECCPlot Off;
Toggle INCPlot Off;
Propagate DefaultProp(Sat) {Sat.ElapsedSecs = Sat.OrbitPeriod};
GMAT RefEpoch = Sat.A1ModJulian;
GMAT Iterate = 0;
```

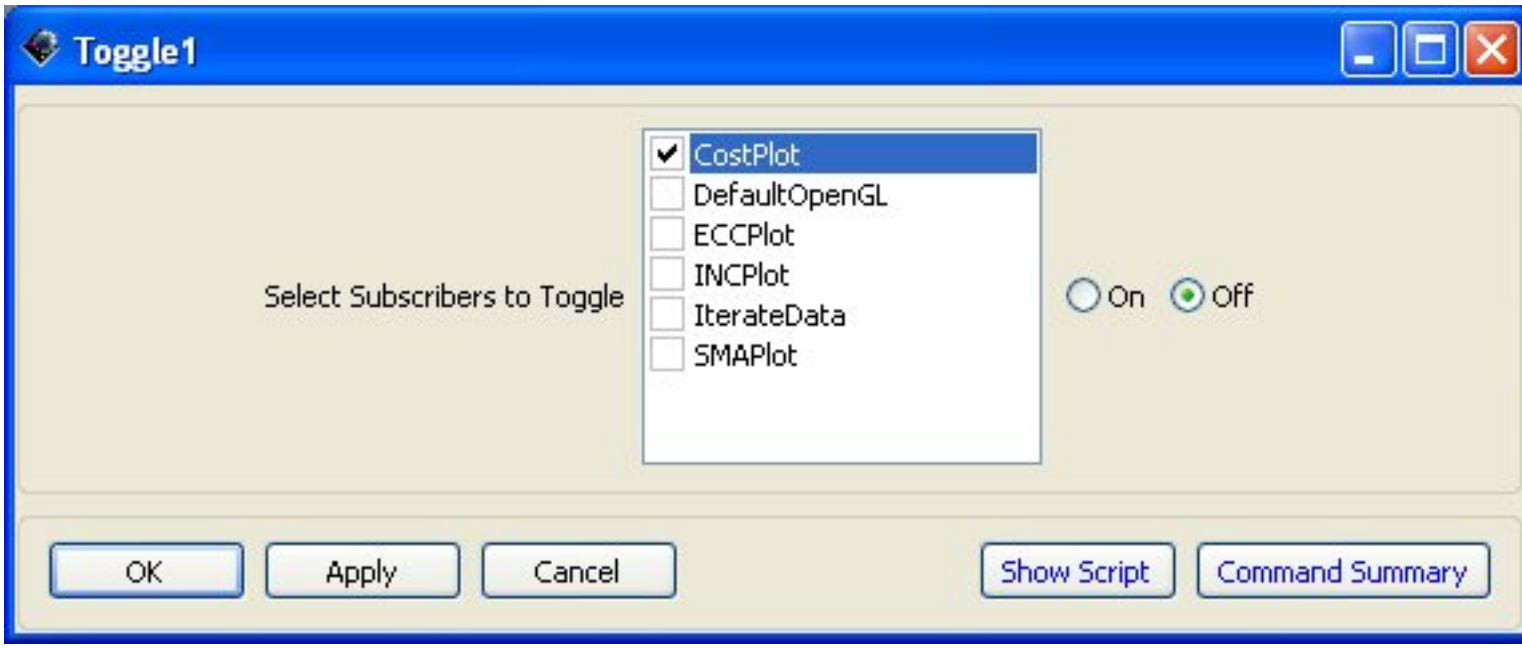


Figure: Toggle1 Command

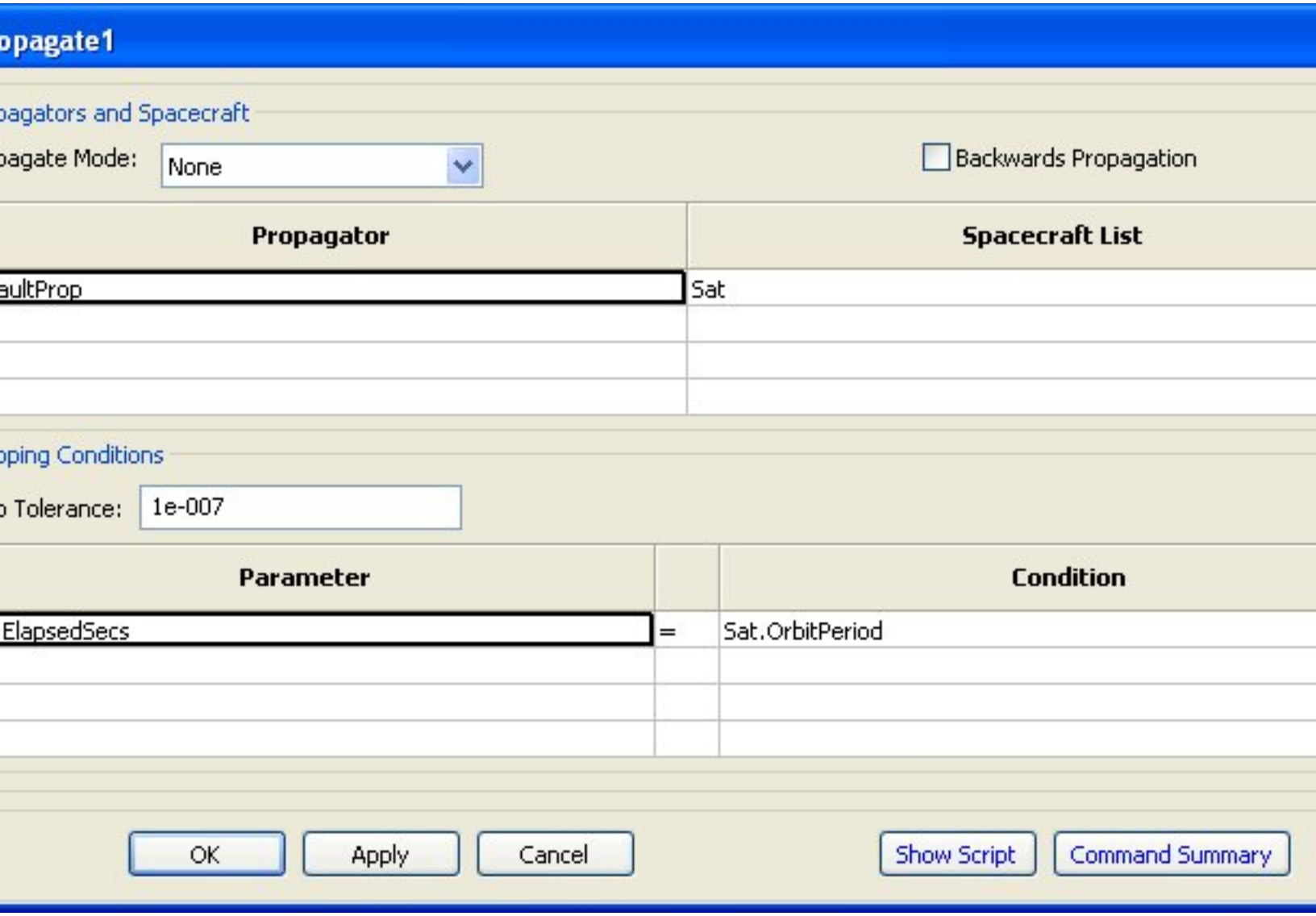


Figure: Propagate Command

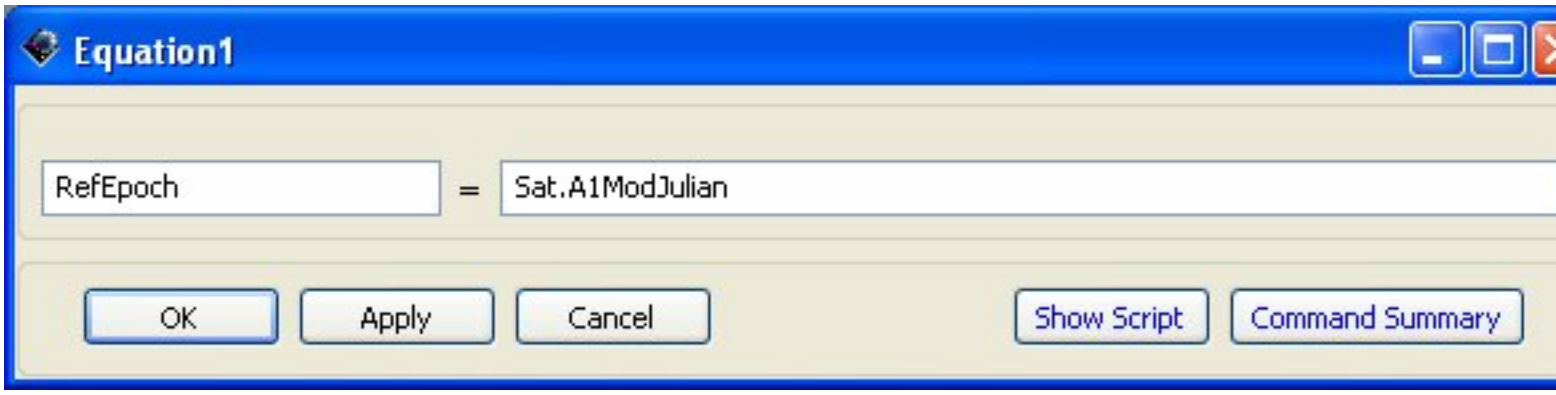


Figure: Equation1 Command

Optimization Sequence - Part 1

- Equation3 counts the number of iterations performed in the optimization sequence. This number will allow tracking of the data in regards to the particular iteration. This is helpful for the ReportFile and the XY Plots.
- The time when the transfer orbit insertion burn is started is varied. Note that the units on dV1Time(as well as A1ModJulian) is days.
- Equation4 is used to change the point to which it propagates to.
- The first If statement allows it to propagate forward while the second If statement allows it to propagate backwards, thus covering the entire orbit so that a burn is possible at any point.
- Equation3 Iterate = Iterate + 1
- Vary dV1 Time
  - Lower = -.5, Upper = .5, MultiplicativeScaleFactor = 10
- Equation4 PropEpoch = RefEpoch + dV1Time
- If dV1 > 0
  - Propagate until Sat.A1ModJulian = PropEpoch
- If dV1 < 0
  - Backwards Propagate until Sat.A1ModJulian = PropEpoch

Script Syntax:

```
Vary SQPfmincon(dV1Time = 0.1, {Lower = -.5, Upper = 0.5, AdditiveScaleFactor = 0.
   GMAT PropEpoch = RefEpoch + dV1Time;
   If dV1Time > 0
      Propagate DefaultProp(Sat) {Sat.A1ModJulian = PropEpoch};
   EndIf;
   If dV1Time < 0
      Propagate BackProp DefaultProp(Sat) {Sat.A1ModJulian = PropEpoch};
   EndIf;
```

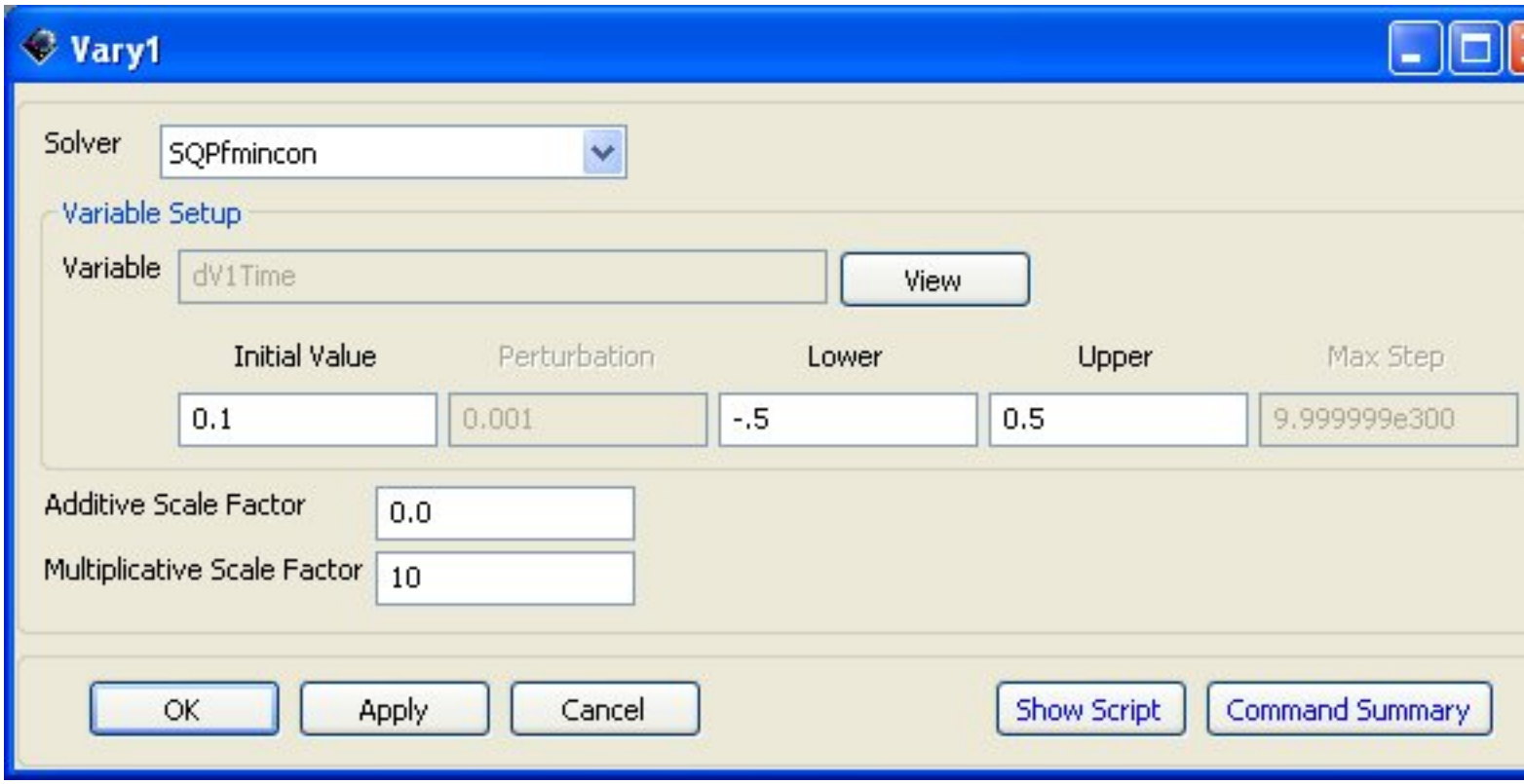


Figure: Vary1 Command

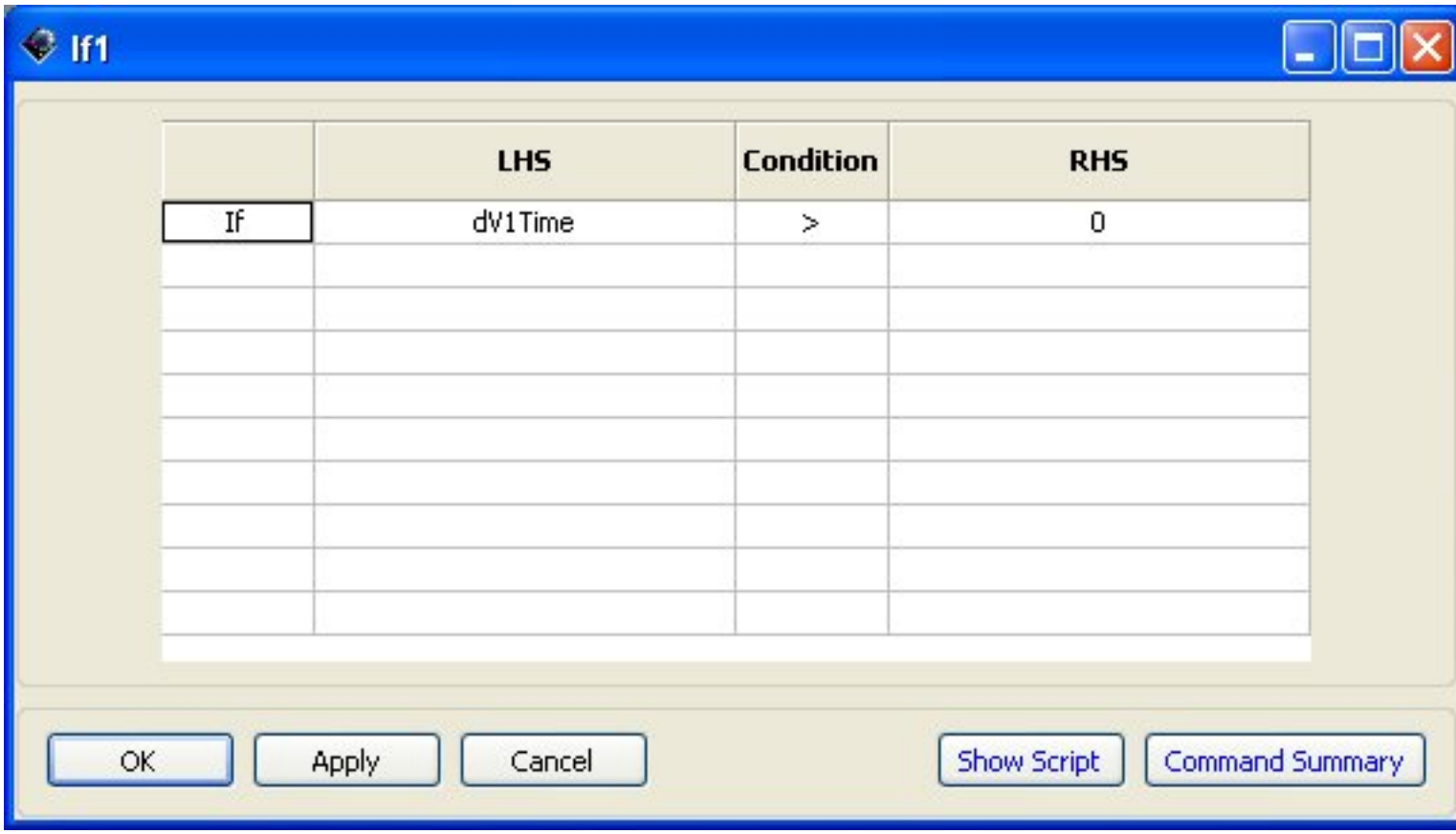


Figure: If Command

Optimization Sequence - Part 2

- First the transfer orbit insertion burn is varied and the maneuver is applied.
- Next the time in which it remains in the transfer orbit (TOF) is varied.
- Now TOF is summed with RefEpoch so that Sat is propagated to the correct location.
- Final orbit insertion burn is varied and the maneuver is applied.
- The final values for the SMA, ECC, and INC are stored in variables. The reason SMA is divided by 10000 is to give it a wider tolerance.
- The goal values for SMA, ECC, and INC are compared to the goal values by using Nonlinear Constraints.
- Equation10 and 11 are used to find the magnitude of the burns.
- It is then made so that the value to be minimized is the Cost
- A ScriptEvent is inserted in the mission sequence for simplicity
  - The four XY plots are turned on so that they are able to record data.
  - The DummySat is propagated a step. In order for any data to be recorded by an XY plot, a propagation of any sort needs to take place, thus a DummySat is used which is not useful or needed beyond this particular need.

  - The four XY plots are turned off, so that they will not record the other values that aren't useful.
- Thus the Optimization Sequence Closes
- Vary2: dv1.V
  - Initial = .1, Lower = -5, Upper = 5
- Vary3: dv1.N
  - Initial = .01, Lower = -5, Upper = 5
- Vary4: dv1.B
  - Initial = .01, Lower = -5, Upper = 5
- Maneuver1: Apply dv1 to Sat
- Vary5: TOF
  - Initial = .3, Lower = 0, Upper = 1, Multiplicative Scale Factor= 10
- Equation5: PropEpoch = RefEpoch + TOF
- Propagate4: Sat until Sat.A1ModJulian = PropEpoch
- Vary6: dv2.V
  - Initial = .1, Lower = -5, Upper = 5
- Vary7: dv2.N
  - Initial = .01, Lower = -5, Upper = 5
- Vary8: dv2.B
  - Initial = .01, Lower = -5, Upper = 5
- Maneuver2: Apply dv2 to Sat
- Equation6: Constraint1 = Sat.SMA/10000
- Equation7: Constraint2 = Sat.ECC
- Equation8: Constraint3 = Sat.INC
- Nonlinear Constraint1 : Constraint1 = 1
- Nonlinear Constraint2 : Constraint2 = .05
- Nonlinear Constraint3 : Constraint3 = 5
- Equation9: dV1mag = sqrt(dv1.Element1^2 + dv1.Element2^2 + dv1.Element3^2)
- Equation10: dV2mag = sqrt(dv2.Element1^2 + dv2.Element2^2 + dv2.Element3^2)
- Equation11: Cost = (dV1mag + dV2mag)
- Minimize1: Cost
- Script Event

- Report IterateData dV1Time TOF Cost Sat.SMA Sat.ECC Sat.INC;
- Toggle CostPlot On;
- Toggle SMAPlot On;
- Toggle ECCPlot On;
- Toggle INCPlot On;
- Propagate DefaultProp(DummySat);
- Toggle CostPlot Off;
- Toggle SMAPlot Off;
- Toggle ECCPlot Off;
- Toggle INCPlot Off;

Script Syntax:

```
Vary SQPfmincon(dv1.Element1 = 0.1, {Lower = -5, Upper = 5, AdditiveScaleFactor =
   Vary SQPfmincon(dv1.Element2 = 0.01, {Lower = -5, Upper = 5, AdditiveScaleFacto
   Vary SQPfmincon(dv1.Element3 = 0.01, {Lower = -5, Upper = 5, AdditiveScaleFacto
   Maneuver dv1(Sat);

   Vary SQPfmincon(TOF = 0.3, {Lower = 0, Upper = 1, AdditiveScaleFactor = 0.0, Mu
   GMAT PropEpoch = RefEpoch + TOF;
   Propagate DefaultProp(Sat) {Sat.A1ModJulian = PropEpoch};

   Vary SQPfmincon(dv2.Element1 = 0.1, {Lower = -5, Upper = 5, AdditiveScaleFactor
   Vary SQPfmincon(dv2.Element2 = 0.01, {Lower = -5, Upper = 5, AdditiveScaleFacto
   Vary SQPfmincon(dv2.Element3 = 0.01, {Lower = -5, Upper = 5, AdditiveScaleFacto
   Maneuver dv2(Sat);

   GMAT Constraint1 = Sat.SMA/10000;
   GMAT Constraint2 = Sat.ECC;
   GMAT Constraint3 = Sat.INC;

   NonlinearConstraint SQPfmincon(Constraint1=1);    % SMA Constraint
   NonlinearConstraint SQPfmincon(Constraint2=0.05);    % ECC Constraint
   NonlinearConstraint SQPfmincon(Constraint3=5);    % INC Constraint

   GMAT dV1mag = sqrt( dv1.Element1^2 + dv1.Element2^2 + dv1.Element3^2 );
   GMAT dV2mag = sqrt( dv2.Element1^2 + dv2.Element2^2 + dv2.Element3^2 );
   GMAT Cost = ( dV1mag + dV2mag );
   Minimize SQPfmincon(Cost);

   BeginScript
      Report IterateData dV1Time TOF Cost Sat.SMA Sat.ECC Sat.INC;
      Toggle CostPlot On;
      Toggle SMAPlot On;
      Toggle ECCPlot On;
      Toggle INCPlot On;
```

```
            Propagate DefaultProp(DummySat);
            Toggle CostPlot Off;
            Toggle SMAPlot Off;
            Toggle ECCPlot Off;
            Toggle INCPlot Off;
         EndScript;

      EndOptimize;  % For optimizer SQPfmincon
```

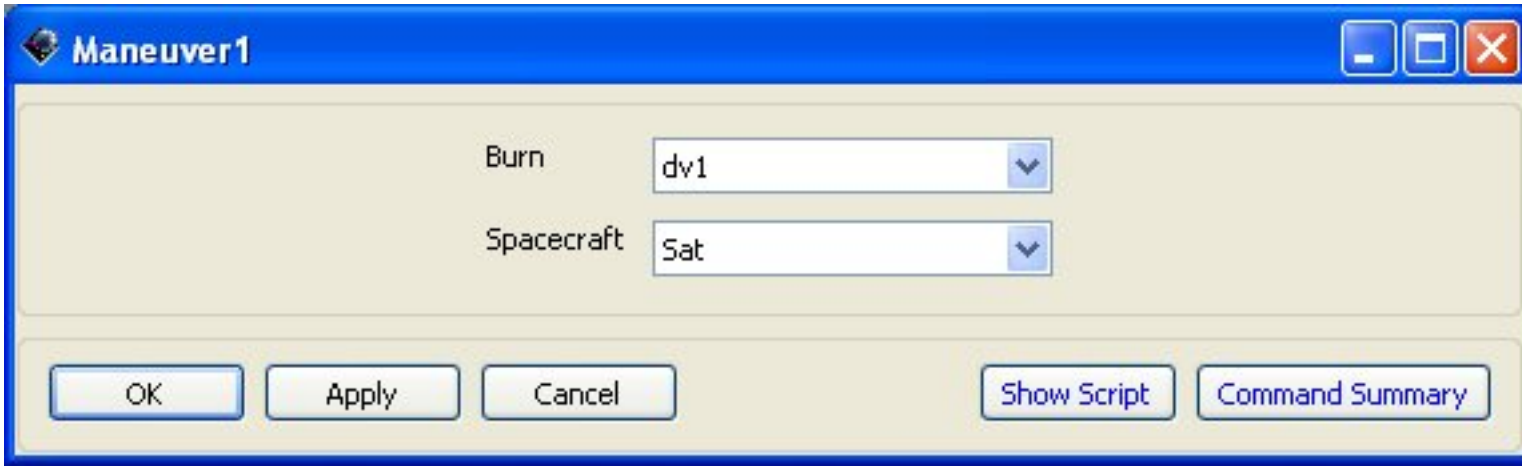


Figure: Maneuver Command

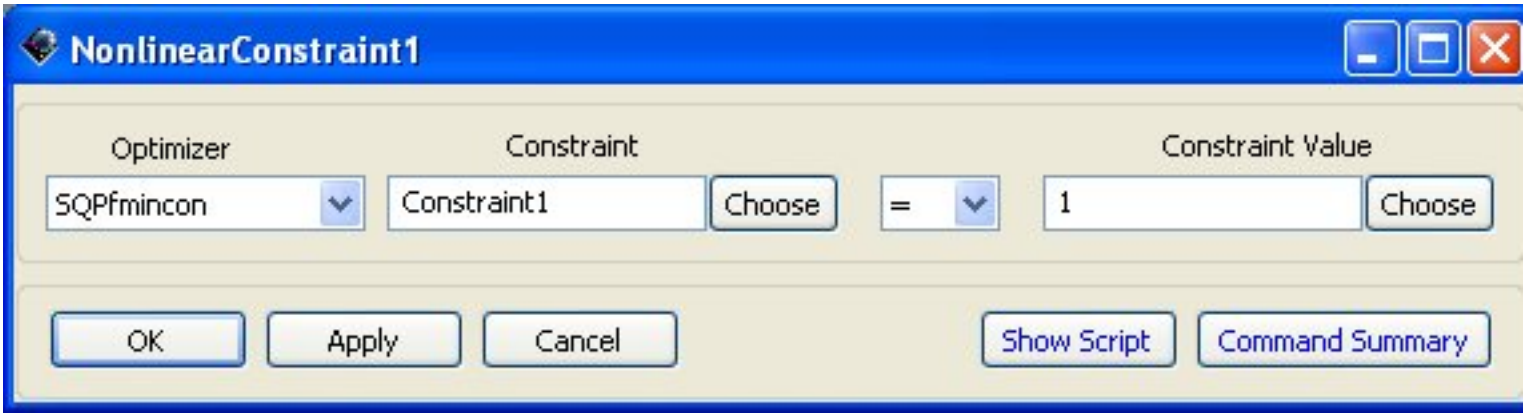


Figure: Nonlinear Constraint Command

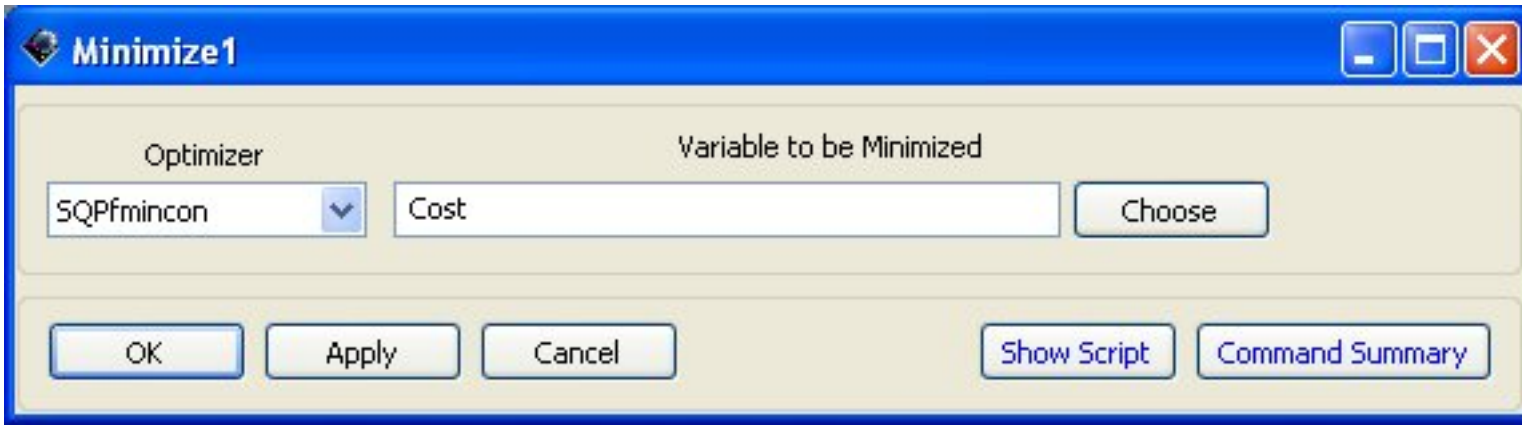


Figure: Minimize Command

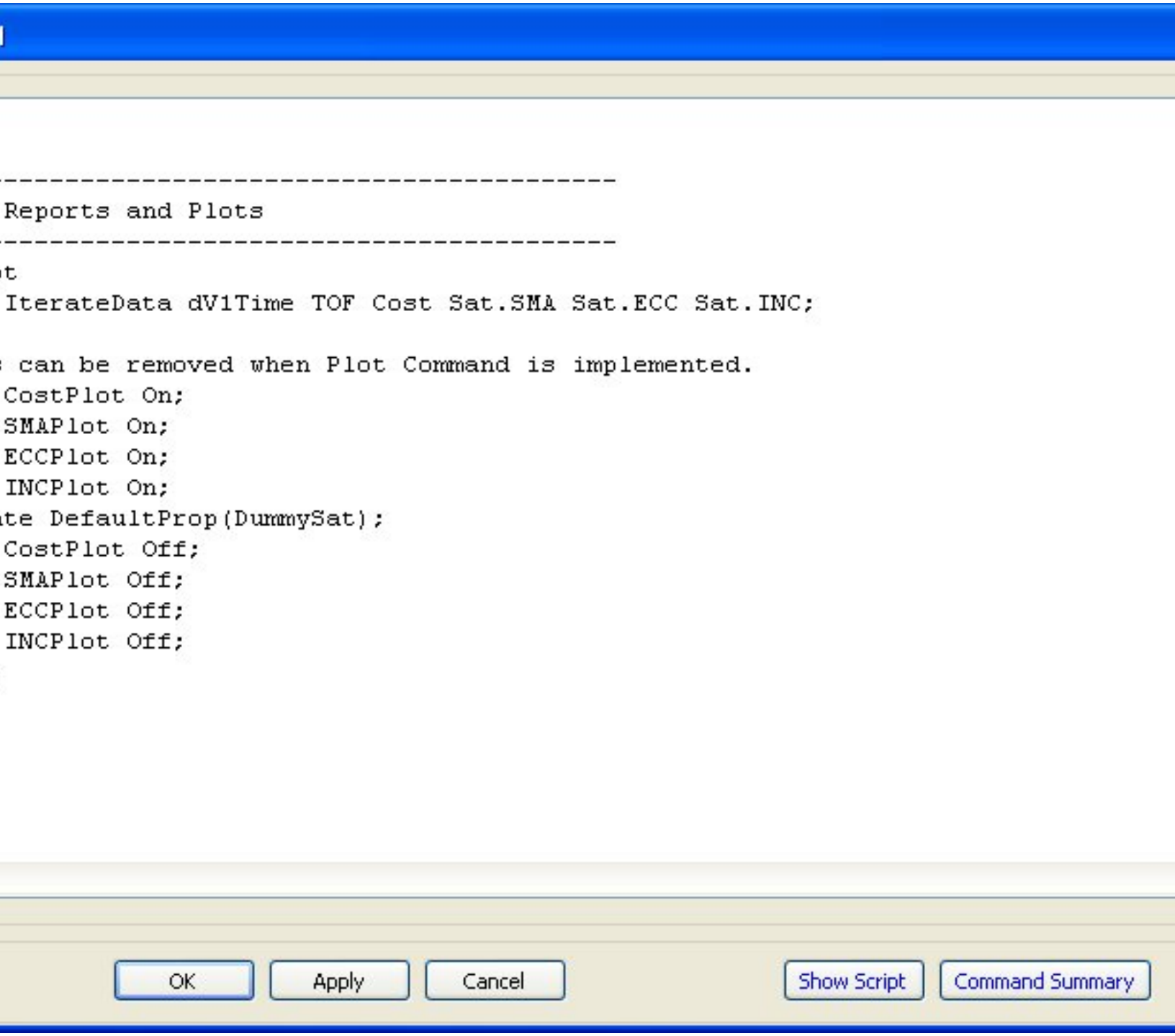


Figure: ScriptEvent Command

Closing Sequence

- SolutionData is reported to IterateData
- TOF and Cost is reported to IterateData

- Sat.SMA, Sat.ECC, Sat.INC are reported to IterateData
- dv1.V, dv1.N, dv1.B are reported to IterateData
- dv2.V, dv2.N, dv2.B are reported to IterateData
- Sat is propagated for one orbital period

Script Syntax:

```
Report IterateData SolutionData;
Report IterateData TOF Cost;
Report IterateData Sat.SMA Sat.ECC Sat.INC;
Report IterateData dv1.Element1 dv1.Element2 dv1.Element3;
Report IterateData dv2.Element1 dv2.Element2 dv2.Element3;

Propagate DefaultProp(Sat) {Sat.ElapsedSecs = Sat.OrbitPeriod};
```

Running the Mission

Resource Mission and Output Trees

The SMAPlot should have a large spike but be largely constant otherwise at about 10000. The INC plot starts increasing then decreasing, is constant for a time, spikes, and then is largely constant at about 5 degrees. The ECC plot is largely zeroed except for a single spike. The Cost plot decreases by a small amount, spikes, and then is constants largely at some value near 1.

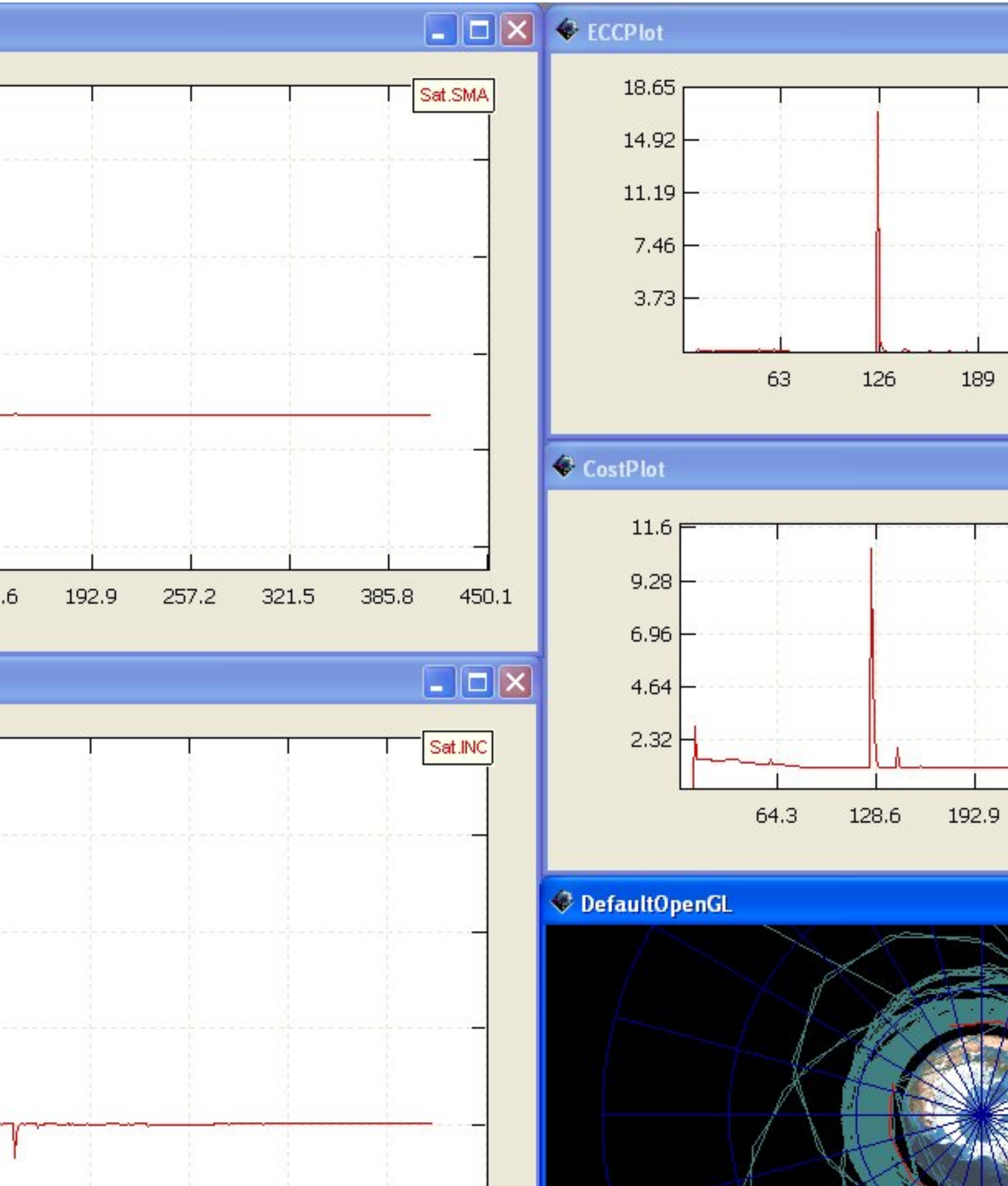

Sat.SMA
.6
192.9
257.2
321.5
385.8
450.1
ECCPlot
18.65
14.92
11.19
7.46
3.73
63
126
189
CostPlot
11.6
9.28
6.96
4.64
2.32
64.3
128.6
192.9
Sat.INC
DefaultOpenGL

Figure: Output

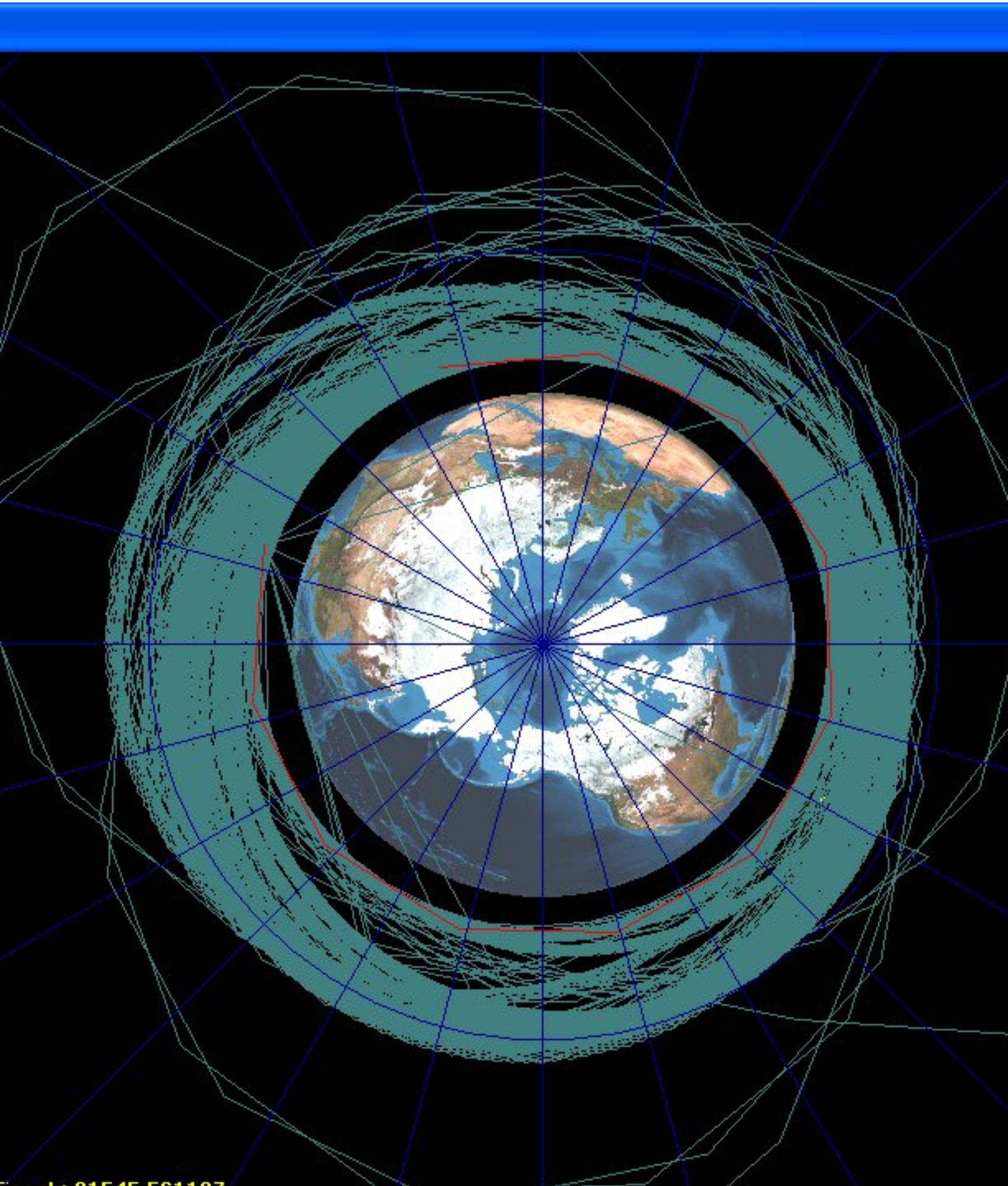


Figure: GL plot enlarged Command

Chapter 22. Minimum Fuel for Lunar Transfer

This tutorial was started from the default settings in the July 2, 2008 build of GMAT.

Objective and Overview

This mission demonstrates further use of optimization sequences. This optimization sequence minimizes the amount of fuel from the Earth to the moon using a standard two-burn trajectory to the moon. Note that this is not a low energy boundary orbit.

Download the script file: Ex_MinFuelLunarCapture.script [scripts/Ex_MinFuelLunarCapture.script]

Prerequisites

Mission Description

- Objective:
  - Minimal Fuel Requirements to go from LEO to Orbit about Moon
  - Luna.SMA = 7600
  - Luna.ECC = 0.2
  - MoonMJ2000Eq.INC = 85
- Assume:
  - Epoch.UTCGregorian = 22 Jul 2014 11:29:10.811;
  - Using the CoordinateSystem = EarthMJ2000Eq;
    - X = -137380.19843385062;
    - Y = 75679.878675370544;
    - Z = 21487.638751878556;
    - VX = -0.23245320142355028;
    - VY = -0.44627539677580192;
    - VZ = 0.085612056628771024;
    - DryMass = 1000;
    - Cd = 2.2000000000000002;
    - Cr = 1.7;

    - DragArea = 15;
    - SRPArea = 1;
- Find:
    - Minimum burn requirements for Transfer Orbit Insertion
    - Minimum burn requirements for Lunar Orbit Insertion

Resource, Mission, and Output trees

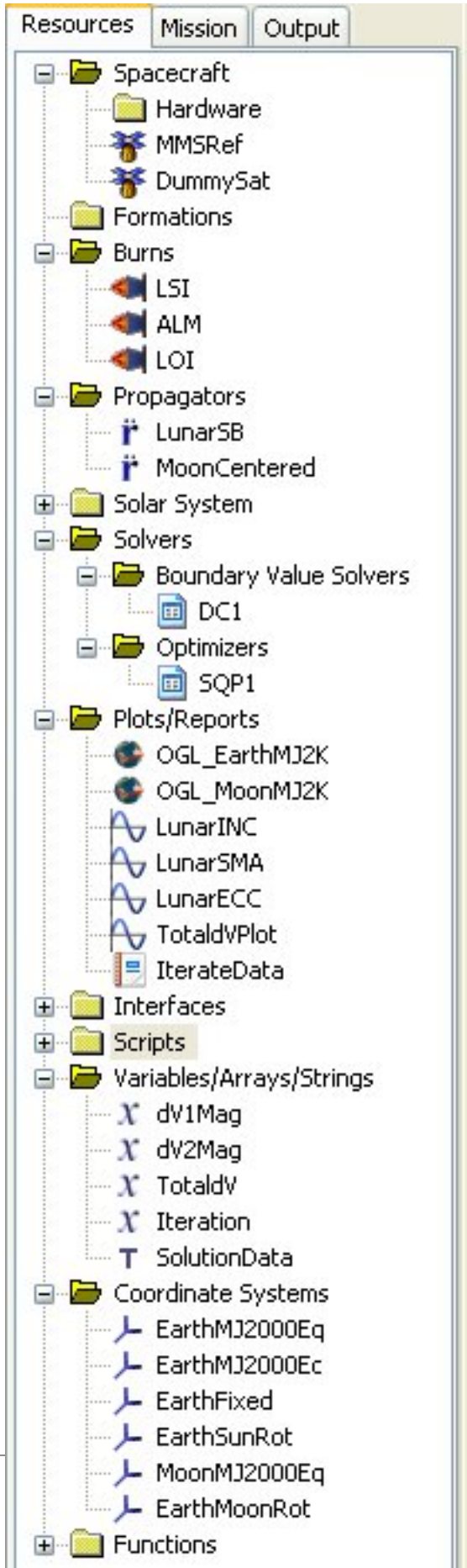


Figure: Minimum Fuel for Lunar Transfer Resource, Mission, and Output trees

Creating and Configuring the Resource Tree

Objects Required

- Spacecraft: MMSRef, DummySat
- Burns: LSI, LOI (Lunar Orbit Insertion)
- Propagators: LunarSB, MoonCentered
- Optimizer: SQP1
- Plots/Reports: OGL_EarthMJ2K, OGL_MoonMJ2K, LunarINC, LunarSMA, LunarECC, TotaldVPlot, IterateData
- Coordinate Systems: EarthMJ2000Eq, EarthMJ2000Ec, EarthSunRot, MoonMJ2000Eq, EarthMoonRot
- Variables: dV1Mag, dV2Mag, TotaldV, Iteration, SolutionData

Creating and Modifying Objects

- Spacecraft
  - MMSRef
    - X = -137380.19843385062;
    - Y = 75679.878675370544;
    - Z = 21487.638751878556;
    - VX = -0.23245320142355028;
    - VY = -0.44627539677580192;
    - VZ = 0.085612056628771024;
  - DummySat
    - Use Default values

Script Syntax:

```
Create Spacecraft MMSRef DummySat;
GMAT MMSRef.Epoch.UTCGregorian = 22 Jul 2014 11:29:10.811;
GMAT MMSRef.CoordinateSystem = EarthMJ2000Eq;
GMAT MMSRef.DisplayStateType = Cartesian;
GMAT MMSRef.AnomalyType = TA;
GMAT MMSRef.X = -137380.19843385062;
GMAT MMSRef.Y = 75679.878675370544;
GMAT MMSRef.Z = 21487.638751878556;
GMAT MMSRef.VX = -0.23245320142355028;
```

```
GMAT MMSRef.VY = -0.44627539677580192;
GMAT MMSRef.VZ = 0.085612056628771024;
```

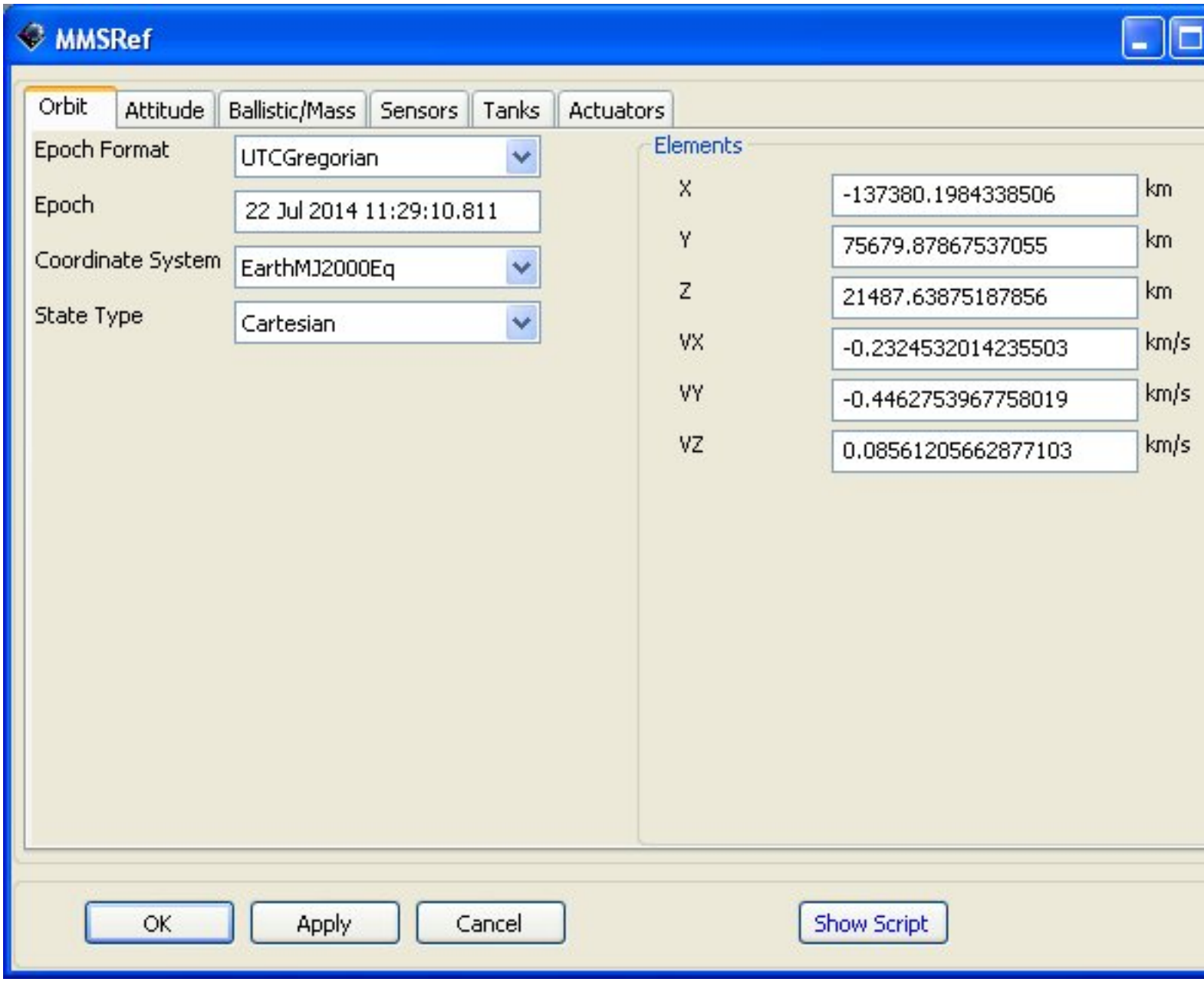


Figure: MMSRef Spacecraft Object'

- Propagators
  - LunarSB
    - Max Step Size = 45000
    - Use Point Masses for Sun, Luna, and Earth
    - No primary bodies, gravity models, atmosphere, or magnetic field models

Script Syntax:

```
Create ForceModel LunarSB_ForceModel;
GMAT LunarSB_ForceModel.CentralBody = Earth;
GMAT LunarSB_ForceModel.PointMasses = {Earth, Sun, Luna};
GMAT LunarSB_ForceModel.Drag = None;
GMAT LunarSB_ForceModel.SRP = Off;
GMAT LunarSB_ForceModel.ErrorControl = RSSStep;

Create Propagator LunarSB;
GMAT LunarSB.FM = LunarSB_ForceModel;
GMAT LunarSB.Type = RungeKutta89;
GMAT LunarSB.InitialStepSize = 60;
GMAT LunarSB.Accuracy = 9.999999999999999e-012;
GMAT LunarSB.MinStep = 0.001;
GMAT LunarSB.MaxStep = 45000;
GMAT LunarSB.MaxStepAttempts = 50;
```

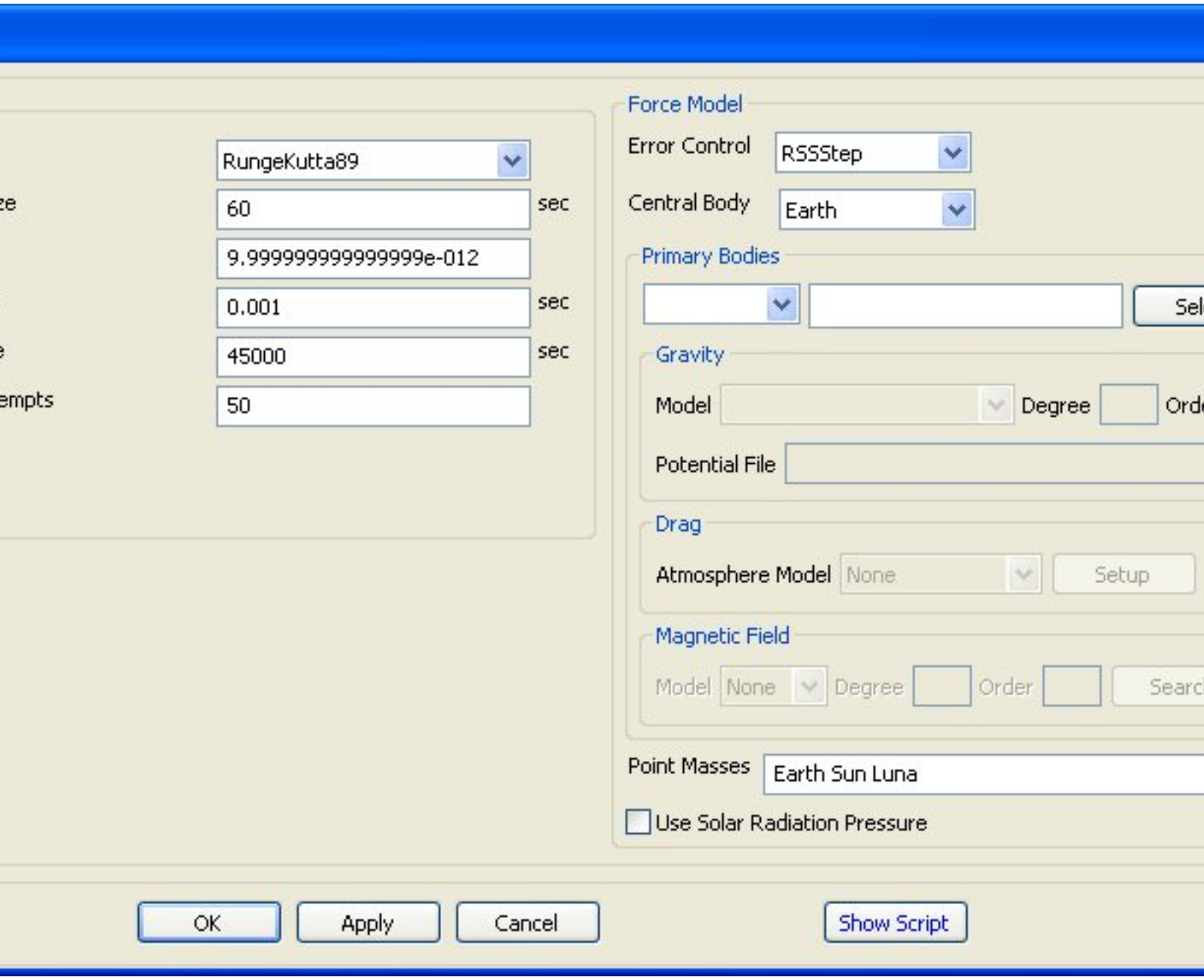


Figure: LunarSB Propagator Object

- MoonCentered
  - Max Step Size = 15000
  - Central Body = Luna
  - Use Point Masses for Sun, Luna, and Earth
  - No primary bodies, gravity models, atmosphere, or magnetic field models

Script Syntax:

```
Create ForceModel MoonCentered_ForceModel;
GMAT MoonCentered_ForceModel.CentralBody = Luna;
GMAT MoonCentered_ForceModel.PointMasses = {Earth, Sun, Luna};
GMAT MoonCentered_ForceModel.Drag = None;
GMAT MoonCentered_ForceModel.SRP = Off;
GMAT MoonCentered_ForceModel.ErrorControl = RSSStep;

Create Propagator MoonCentered;
GMAT MoonCentered.FM = MoonCentered_ForceModel;
GMAT MoonCentered.Type = RungeKutta89;
GMAT MoonCentered.InitialStepSize = 60;
GMAT MoonCentered.Accuracy = 9.999999999999999e-012;
GMAT MoonCentered.MinStep = 0.001;
```

- Burns
  - LSI
    - V = 0.14676929889000001; N = 0.046042675892; B = 0.090223244096999999;
  - LOI
    - Origin: Luna
    - V = -.652

Script Syntax:

```
Create ImpulsiveBurn LSI;
GMAT LSI.Origin = Earth;
GMAT LSI.Axes = VNB;
GMAT LSI.VectorFormat = Cartesian;
GMAT LSI.V = 0.14676929889000001;
GMAT LSI.N = 0.046042675892;
GMAT LSI.B = 0.090223244096999999;

Create ImpulsiveBurn LOI
GMAT LOI.Origin = Luna;
GMAT LOI.Axes   = VNB;
GMAT LOI.V      = -0.652;
```

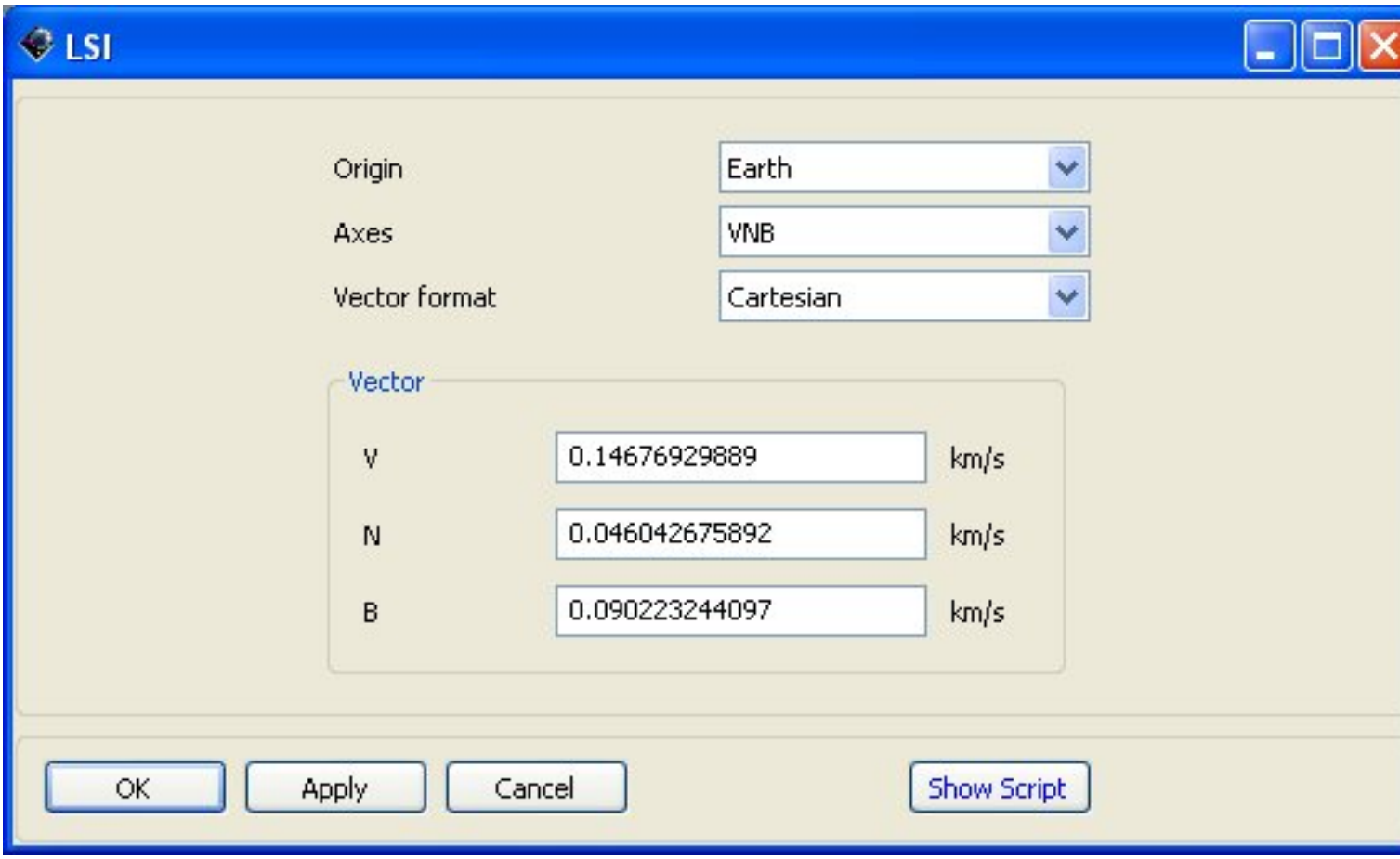


Figure: LSI Burn Object

- Variables/Strings/Arrays
  - Variable dV1Mag
  - Variable dV2Mag
  - Variable TotaldV
  - Variable Iteration
  - String SolutionData
    - SolutionData = ---------- Solution Data ----------

  Script Syntax:

```
Create Variable dV1Mag dV2Mag TotaldV Iteration;
Create String SolutionData
SolutionData = '---------- Solution Data ----------';
```

- Coordinate Systems
  - EarthSunRot

  - ObjectReferenced
  - Primary: Sun, Secondary: Earth
  - XAxis: R
  - ZAxis: N
- MoonMJ2000Eq
  - Origin: Luna
  - Axes: MJ2000Eq

Script Syntax:

```
Create CoordinateSystem EarthSunRot;
GMAT EarthSunRot.Origin = Earth;
GMAT EarthSunRot.J2000Body = Earth;
GMAT EarthSunRot.Axes = ObjectReferenced;
GMAT EarthSunRot.Epoch = 21545.000000397937;
GMAT EarthSunRot.UpdateInterval = 60;
GMAT EarthSunRot.XAxis = R;
GMAT EarthSunRot.ZAxis = N;
GMAT EarthSunRot.Primary = Sun;
GMAT EarthSunRot.Secondary = Earth;

Create CoordinateSystem MoonMJ2000Eq;
GMAT MoonMJ2000Eq.Origin = Luna;
GMAT MoonMJ2000Eq.Axes = MJ2000Eq;

Create CoordinateSystem EarthMoonRot;
GMAT EarthMoonRot.Origin = Luna;
GMAT EarthMoonRot.Axes = ObjectReferenced;
GMAT EarthMoonRot.XAxis = R;
GMAT EarthMoonRot.ZAxis = N;
GMAT EarthMoonRot.Primary = Earth;
GMAT EarthMoonRot.Secondary = Luna;
```

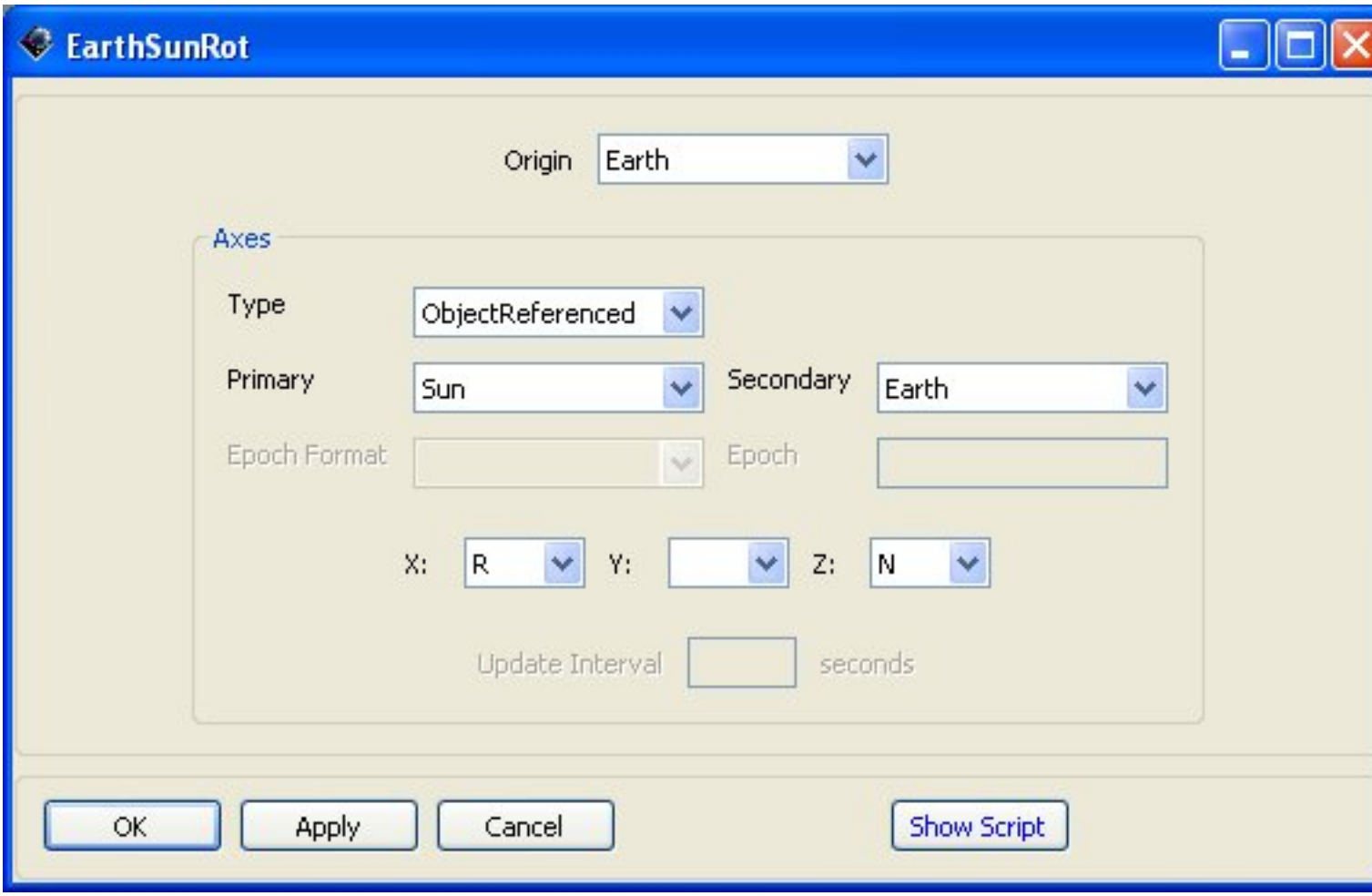


Figure: EarthSunRot Coordinate System Object

- Solvers
  - SQP1
    - Tol Fun: 1e-3
    - Tol Con: 1e-3
    - Tol X: 1e-3
    - Max Iter: 50
    - Diff Min Change: 1e-5

  Script Syntax:

```
Create FminconOptimizer SQP1;
GMAT SQP1.ShowProgress = true;
GMAT SQP1.ReportStyle = Normal;
GMAT SQP1.TargeterTextFile = FminconOptimizerSQP1.data;
GMAT SQP1.MaximumIterations = 25;
```

```
GMAT SQP1.ObjectiveFunction = Objective;
GMAT SQP1.Tolerance = 0;
GMAT SQP1.SourceType = MATLAB;
GMAT SQP1.DiffMaxChange = 0.1000;
GMAT SQP1.DiffMinChange = 1.0000e-5;
GMAT SQP1.MaxFunEvals = 1000;
GMAT SQP1.MaxIter = 50;
GMAT SQP1.TolX = 1.0000e-03;
GMAT SQP1.TolFun = 1.0000e-03;
GMAT SQP1.TolCon = 1.0000e-03;
GMAT SQP1.DerivativeCheck = Off;
GMAT SQP1.Diagnostics = Off;
GMAT SQP1.Display = iter;
GMAT SQP1.GradObj = Off;
GMAT SQP1.GradConstr = Off;
```

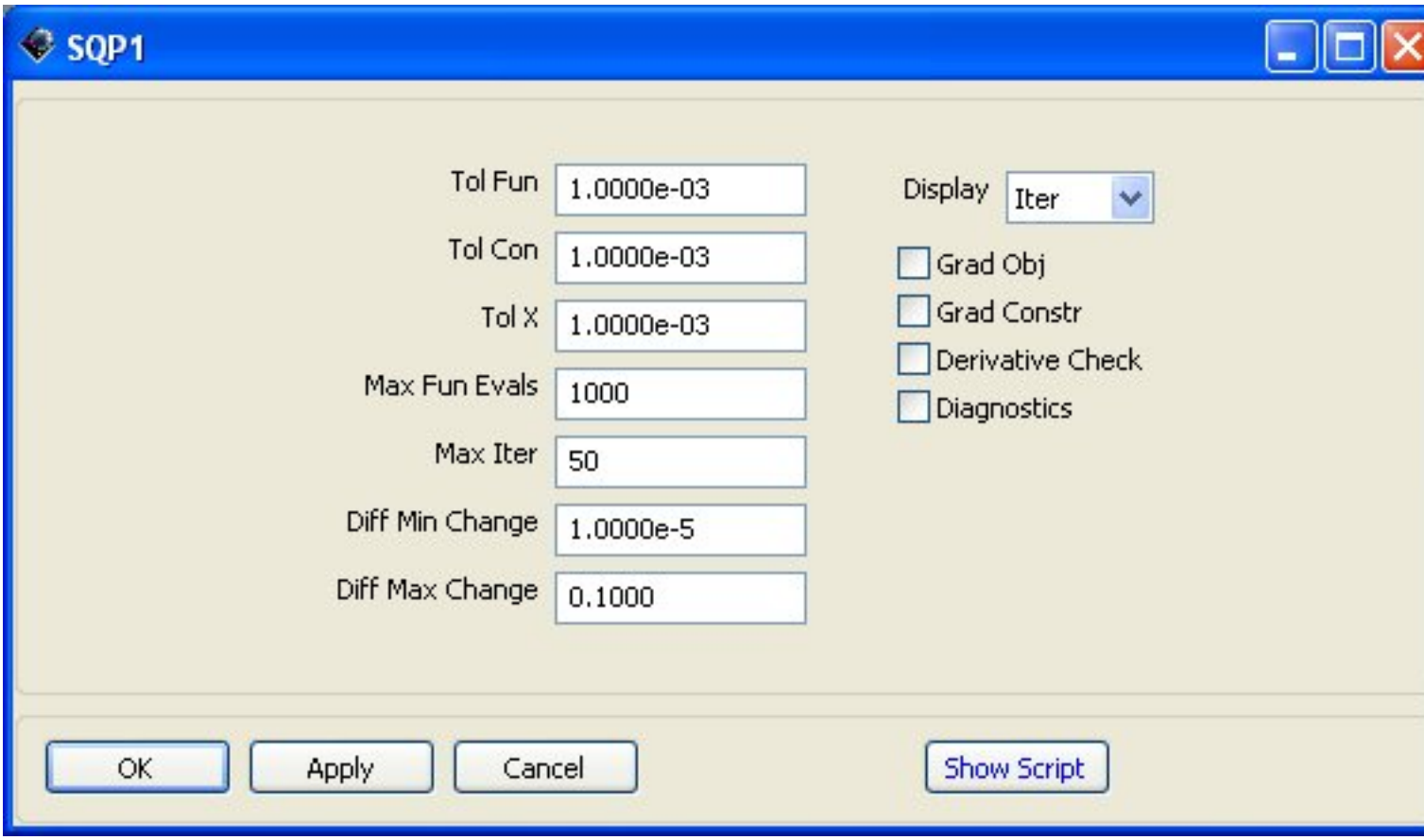


Figure: SQP1 Optimizer Object

- Plots/Reports
  - OpenGLPlot OGL_EarthMJ2K
    - Objects viewed: MMSRef, Earth, Luna
    - Collect data every 5 steps

  - Number of Points to redraw: 150
  - No Solving Iterations displayed
  - View Scale Factor: 30
  - View Up Definition: X Axis
- OpenGLPlot OGL_MoonMJ2K
  - Objects viewed: MMSRef, Earth, Luna
  - Collect data every 5 steps
  - NUmber of points to redraw: 75
  - Show All Solver Iterations
  - CoordinateSystem: MoonMJ200Eq
  - View Reference: Luna
  - View Direction: Luna
  - View Up Definition Coordinate System: MoonMJ2000Eq
  - View Up Definition Axis: X-Axis

Script Syntax:

```
Create OpenGLPlot OGL_EarthMJ2K;
GMAT OGL_EarthMJ2K.Add = {MMSRef, Earth, Luna};
GMAT OGL_EarthMJ2K.CoordinateSystem = EarthMJ2000Eq;
GMAT OGL_EarthMJ2K.ViewPointReference = Earth;
GMAT OGL_EarthMJ2K.ViewDirection = Earth;
GMAT OGL_EarthMJ2K.ViewScaleFactor = 30;
GMAT OGL_EarthMJ2K.FixedFovAngle = 50;
GMAT OGL_EarthMJ2K.ViewUpCoordinateSystem = EarthMJ2000Eq;
GMAT OGL_EarthMJ2K.ViewUpAxis = X;
GMAT OGL_EarthMJ2K.CelestialPlane = Off;
GMAT OGL_EarthMJ2K.XYPlane = On;
GMAT OGL_EarthMJ2K.WireFrame = Off;
GMAT OGL_EarthMJ2K.SolverIterations = None;
GMAT OGL_EarthMJ2K.Axes = On;
GMAT OGL_EarthMJ2K.PerspectiveMode = Off;
GMAT OGL_EarthMJ2K.UseFixedFov = Off;
GMAT OGL_EarthMJ2K.DataCollectFrequency = 5;
GMAT OGL_EarthMJ2K.UpdatePlotFrequency = 50;
GMAT OGL_EarthMJ2K.NumPointsToRedraw = 150;
GMAT OGL_EarthMJ2K.UseInitialView = On;

Create OpenGLPlot OGL_MoonMJ2K;
GMAT OGL_MoonMJ2K.Add = {MMSRef, Earth, Luna};
GMAT OGL_MoonMJ2K.CoordinateSystem = MoonMJ2000Eq;
GMAT OGL_MoonMJ2K.ViewPointReference = Luna;
```

```
GMAT OGL_MoonMJ2K.ViewDirection = Luna;
GMAT OGL_MoonMJ2K.ViewScaleFactor = 1;
GMAT OGL_MoonMJ2K.FixedFovAngle = 45;
GMAT OGL_MoonMJ2K.ViewUpCoordinateSystem = MoonMJ2000Eq;
GMAT OGL_MoonMJ2K.ViewUpAxis = X;
GMAT OGL_MoonMJ2K.CelestialPlane = Off;
GMAT OGL_MoonMJ2K.XYPlane = On;
GMAT OGL_MoonMJ2K.WireFrame = Off;
GMAT OGL_MoonMJ2K.SolverIterations = All;
GMAT OGL_MoonMJ2K.Axes = On;
GMAT OGL_MoonMJ2K.SunLine = Off;
GMAT OGL_MoonMJ2K.PerspectiveMode = Off;
GMAT OGL_MoonMJ2K.UseFixedFov = Off;
GMAT OGL_MoonMJ2K.DataCollectFrequency = 5;
GMAT OGL_MoonMJ2K.UpdatePlotFrequency = 50;
GMAT OGL_MoonMJ2K.NumPointsToRedraw = 75;
```

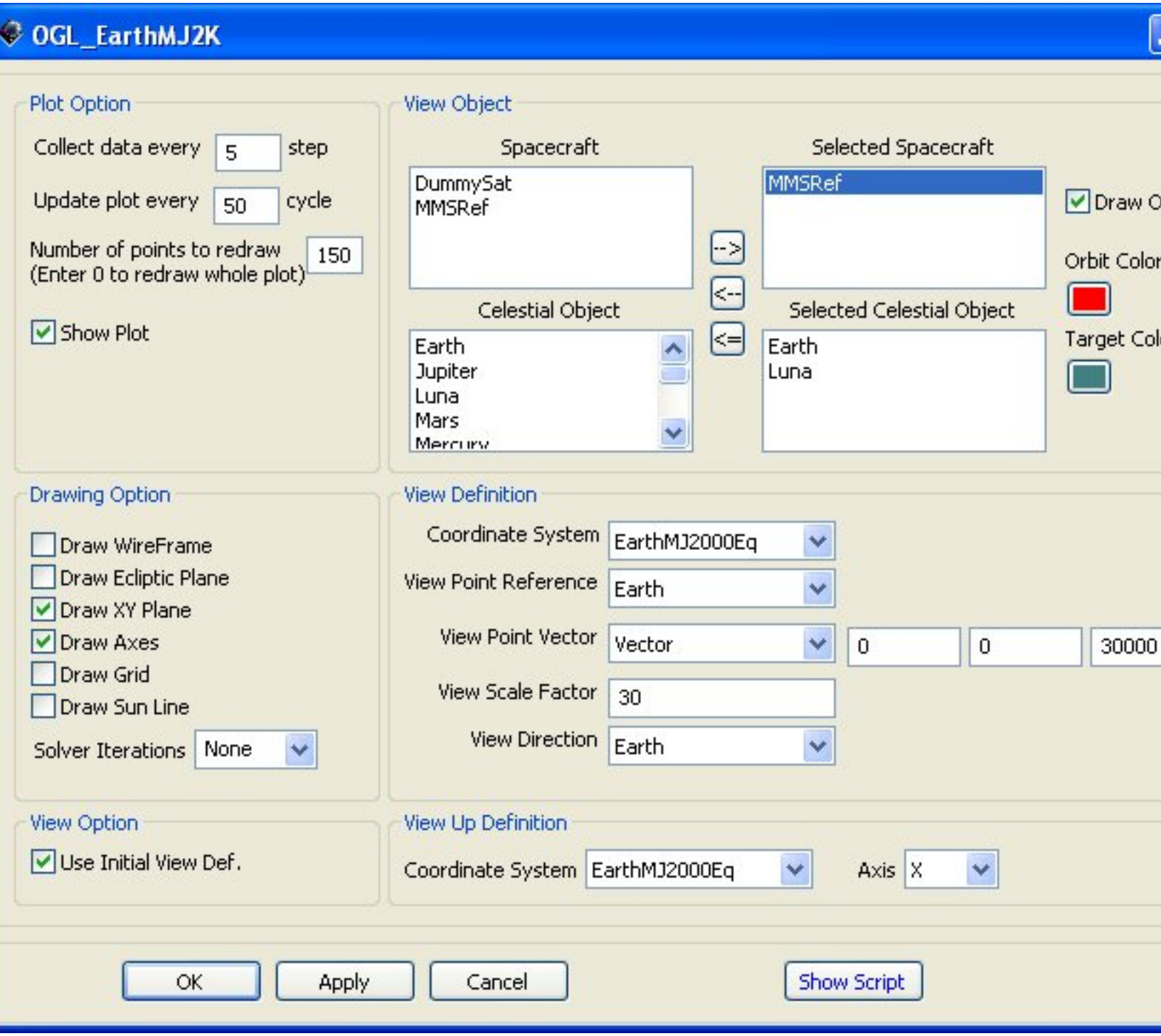


Figure: OGL_EarthMJ2K Open GLPlot Object

- XYPlot LunarINC
  - X: Iteration
  - Y: MoonMJ2000Eq.INC

- XYPlot LunarSMA
  - X: Iteration
  - Y: Luna.SMA
- XYPlot LunarECC
  - X: Iteration
  - Y: Luna.ECC
- XYPlot TotaldVPlot
  - X: Iteration
  - Y: TotaldV

Script Syntax:

```
Create XYPlot LunarINC
GMAT LunarINC.IndVar = Iteration;
GMAT LunarINC.Add    = { MMSRef.MoonMJ2000Eq.INC };
GMAT LunarINC.Grid   = On;
GMAT LunarINC.SolverIterations = All;

Create XYPlot LunarSMA
GMAT LunarSMA.IndVar = Iteration;
GMAT LunarSMA.Add    = { MMSRef.Luna.SMA };
GMAT LunarSMA.Grid   = On;
GMAT LunarSMA.SolverIterations = All;

Create XYPlot LunarECC
GMAT  LunarECC.IndVar = Iteration;
GMAT LunarECC.Add     = { MMSRef.Luna.ECC };
GMAT LunarECC.Grid    = On;
GMAT LunarECC.SolverIterations = All;

Create XYPlot TotaldVPlot;
TotaldVPlot.IndVar = Iteration;
TotaldVPlot.Add    = TotaldV;
TotaldVPlot.SolverIterations = All;
```

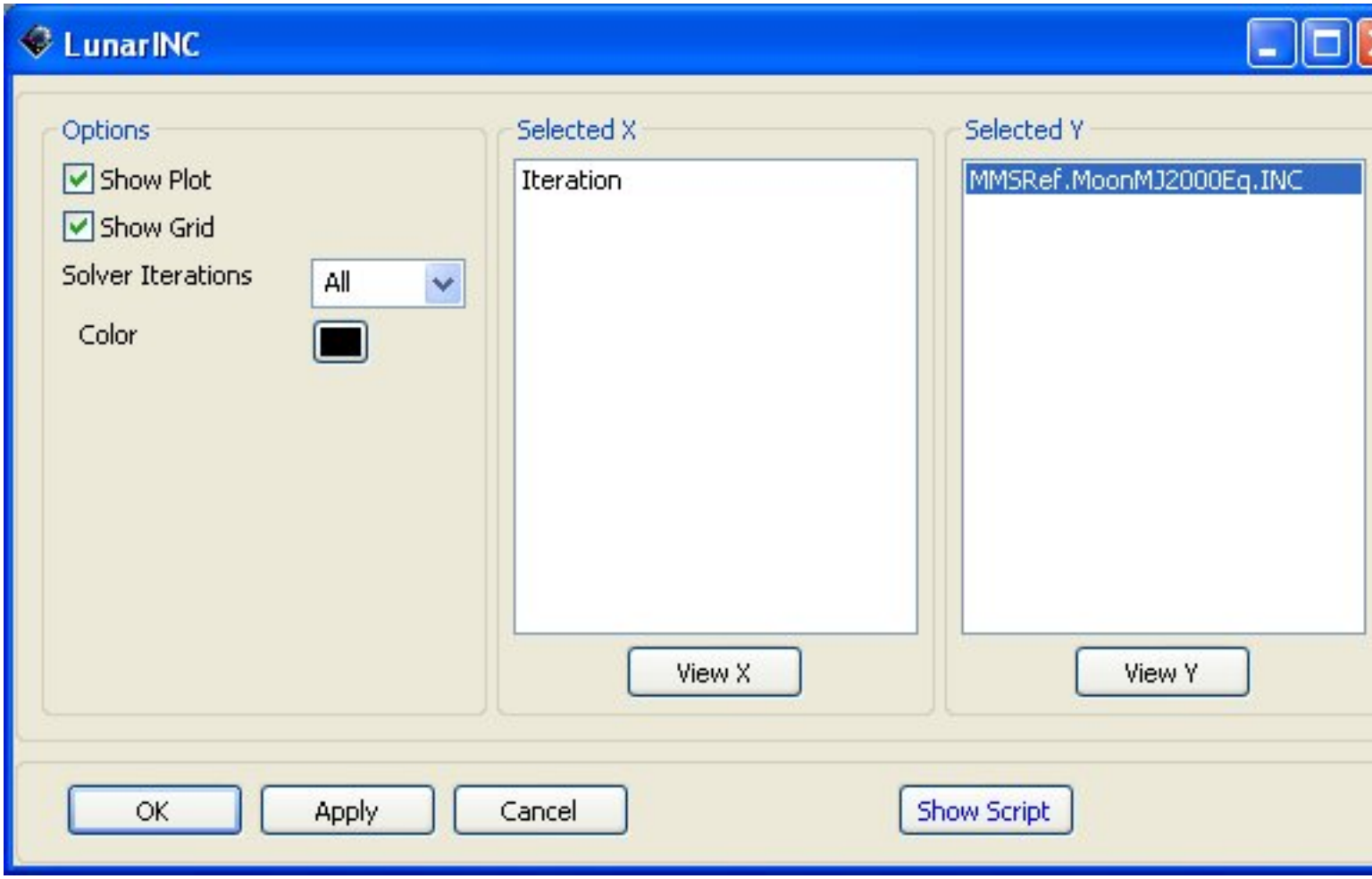


Figure: LunarINC XY Plot Object

- ReportFile IterateData
  - Filename: ./output/SampleMissions/Ex_MinFuelLunarCapture.report

Script Syntax:

```
Create ReportFile IterateData;
IterateData.Filename = ./output/SampleMissions/Ex_MinFuelLunarCapture.report;
```

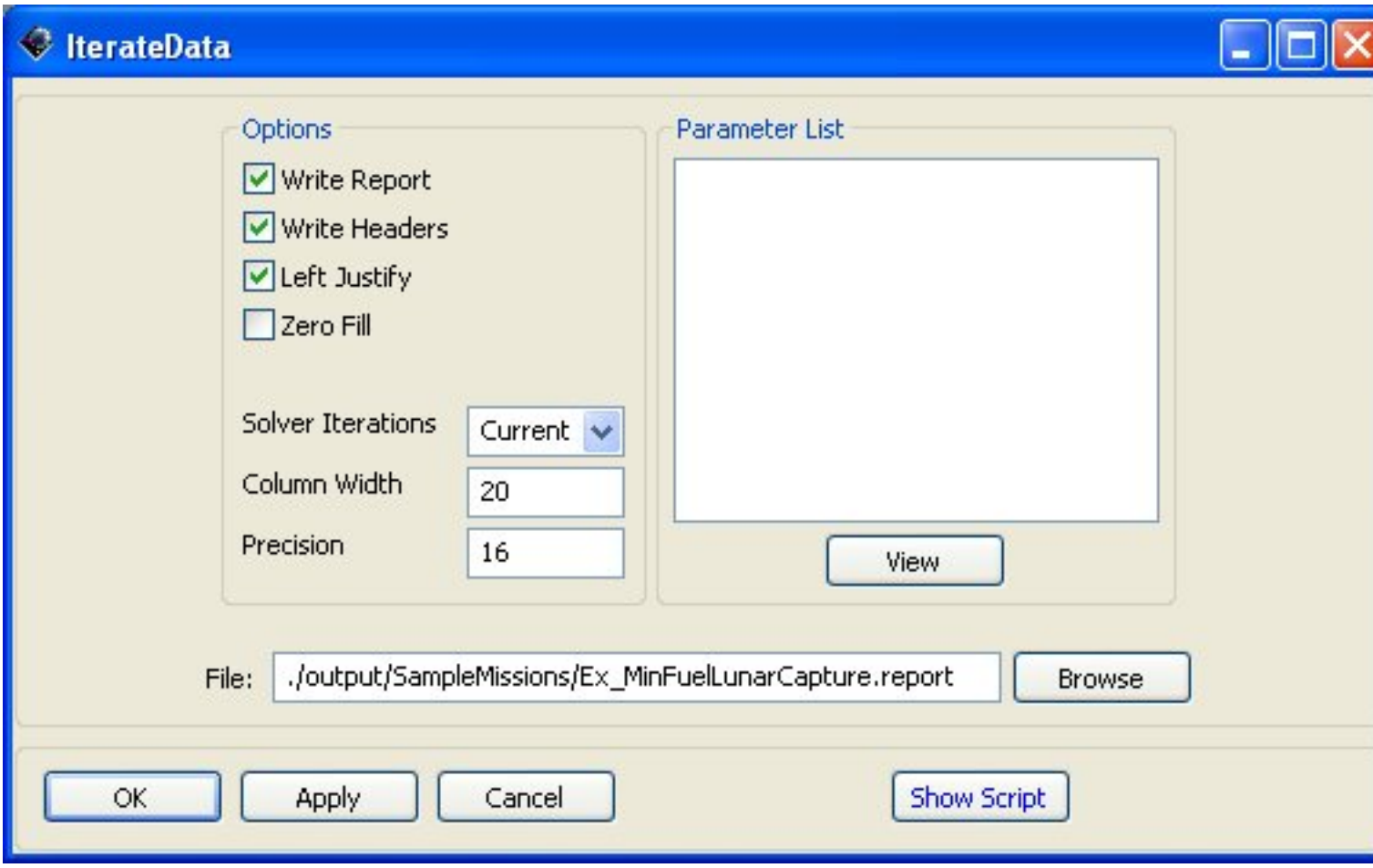


Figure: IterateData FileReport Object

Creating and Configuring the Mission Tree

Resource Mmission and Output Trees

This mission is set-up very similarly to how the Lunar Transfer Tutorial is set-up, except that optimization sequences are used instead of targeting sequences.

- Optimization Sequence Setup
  - Turn off the plots until the proper moment to record data and propagate spacecraft until periapsis
- Optimization Sequence and Final Commands
  - Vary LSI in all directions, perform the burn and propagate to Lunar periapsis, period of time, or until it exits region of space.
  - Nonlinear Constraints on BdotT, BdotR, and Luna.RMag to constrain how far from the moon it can go
  - Vary LOI in all directions and perform burn
  - Nonlinear Constraints dictating goals of mission

- Propagate for period of time or until it exits region of space
- Create Equations to find the magnitude of the burns and take the sum of the two burns
- Minimize the Total amount of dV required by the two burns
- Toggle on plots to record data, propagate dummy spacecraft for data to be recorded, toggle off plots
- Report Parameters to report file

Creating and Modifying Command Sequences

Optimization Sequence Setup

- Toggle off LunarINC, LunarSMA, LunarECC, and TotaldVPlot
- Propagate MMSRef til Periapsis using LunarSB propagator
- Equation1: Iteration = 0

Script Syntax:

```
Toggle LunarINC Off;
Toggle LunarSMA Off;
Toggle LunarECC Off;
Toggle TotaldVPlot Off;
Propagate LunarSB(MMSRef, {MMSRef.Periapsis});
Iteration = 0;
```

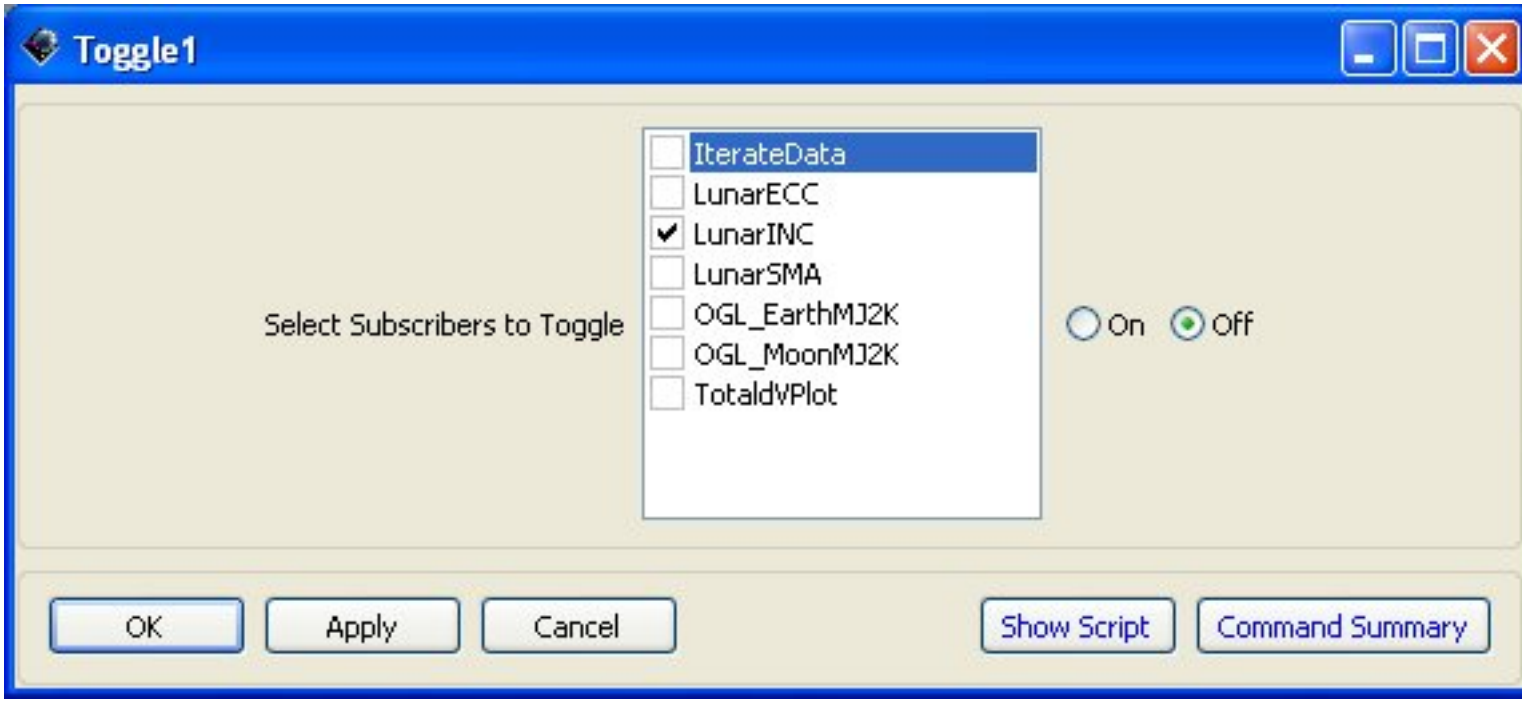


Figure: Toggle Command

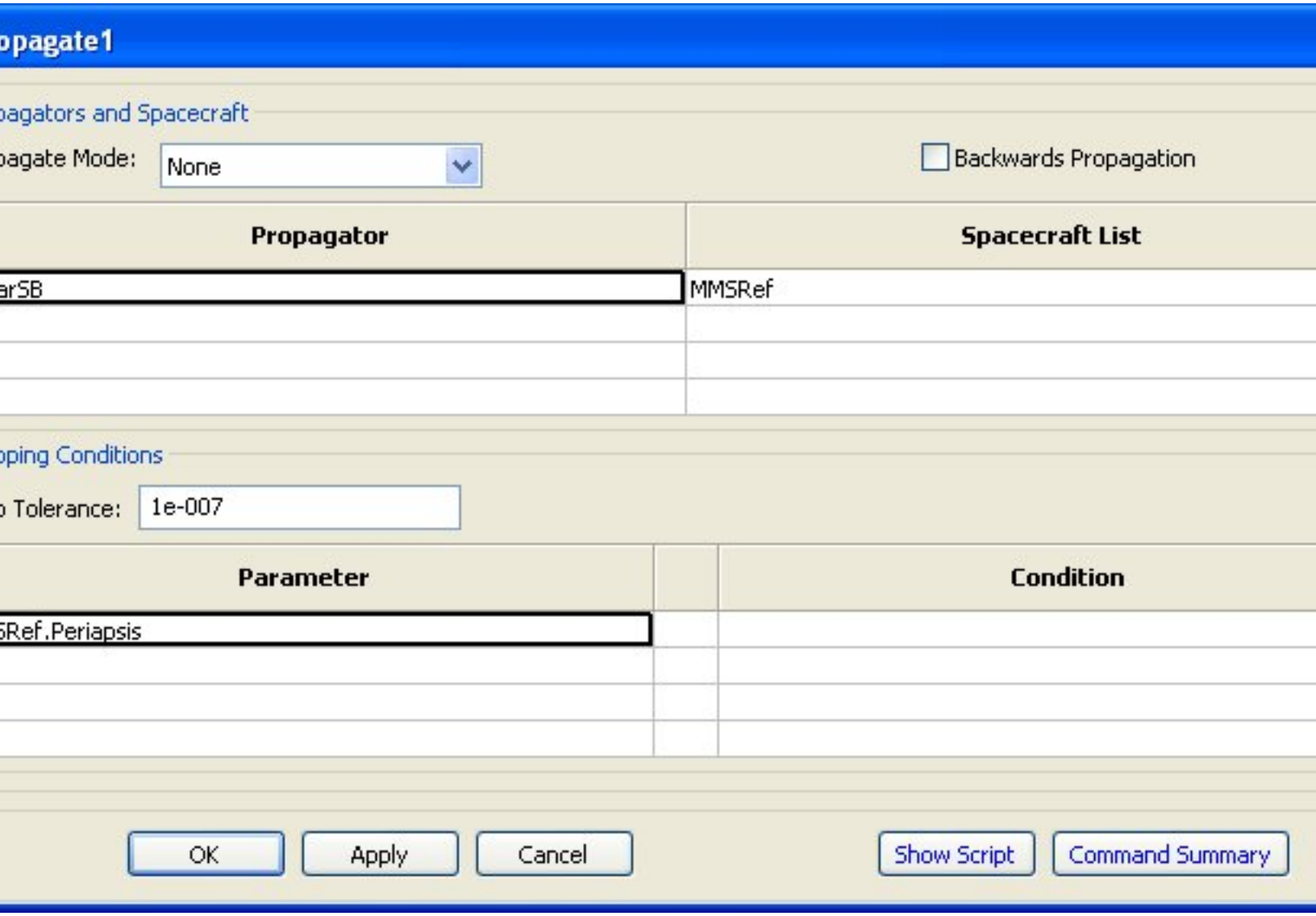


Figure: Propagate Command

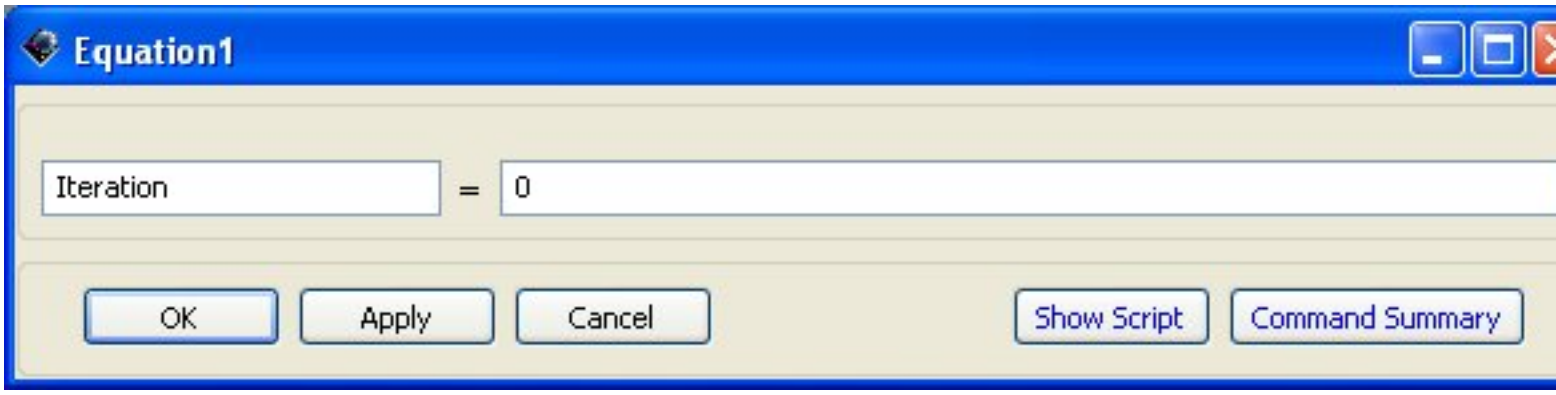


Figure: Equation1 Command

Optimization Sequence and Final Commands

- Equation2: Iteration = Iteration + 1
- Vary LSI.V
  - Initial Guess = .1462, Lower = -1, Upper = 1
- Vary LSI.N
  - Initial Guess = .0460, Lower = -1, Upper = 1
- Vary LSI.B
  - Initial Guess = .1086, Lower = -1, Upper = 1
- Maneuver MMSRef using LSI
- Propagate MMSRef til for 1.5 days using LunarSB propagator
- Propagate MMSRef using MoonCentered propagator til: Luna Periapsis, 8 days have elapsed, Luna.RMAG = 1000, Earth.RMAG = 3000.
- Nonlinear Constraint: MMSRef.MoonMJ2000Eq.BdotT <= 20000.0
- NonlinearConstraint: MMSRef.Luna.RMAG >= 2000
- NonlinearConstraint: MMSRef.MoonMJ2000Eq.BdotR <= 15000.0
- Vary LOI.V
  - Initial Guess = -.752, Lower = -1, Upper = 1
- Vary LOI.N
  - Initial Guess = 0, Lower = -1, Upper = 1
- Vary LOI.B
  - Initial Guess = 0, Lower = -1, Upper = 1
- Maneuver MMSRef using LOI
- NonlinearConstraint: MMSRef.Luna.SMA = 7600
- NonlinearConstraint: MMSRef.Luna.ECC = 0.2
- NonlinearConstraint: MMSRef.MoonMJ2000Eq.INC = 85
- Propagate MMSRef using MoonCentered propagator til: Luna.Periapsis, 5 days have elapsed, Luna.RMAG = 1000, Earth.RMAG = 3000
- Equation3: dV1Mag = sqrt(LSI.Element1^2 + LSI.Element2^2 + LSI.Element3^2)
- Equation4: dV2Mag = sqrt(LOI.Element1^2 + LOI.Element2^2 + LOI.Element3^2)

- Equation5: TotaldV = (dV1Mag + dV2Mag)
- Minimize TotaldV
- Report to IterateData: TotaldV, Luna.SMA, Luna.ECC, MoonMJ2000Eq.INC
- Toggle On LunarINC, LunarSMA, LunarECC, TotaldVPlot
- Propagate DummySat using LunarSB
- Toggle Off LunarINC, LunarSMA, LunarECC, TotaldVPlot
- End Optimization Sequence
- Report to IterateData: TotaldV, dV1Mag, dV2Mag, MMSRef.Luna.SMA, MMSRef.Luna.ECC MMSRef.MoonMJ2000Eq.INC, LSI.Element1, LSI.Element2, LSI.Element3, LOI.Element1, LOI.Element2 LOI.Element3
- Propagate MMSRef using MoonCentered propagator for 15 days

Script Syntax:

```
Optimize SQP1

     Iteration = Iteration + 1;

     Vary SQP1(LSI.V = 0.1462 ,   { Lower = -1, Upper = 1, MultiplicativeScaleFact
     Vary SQP1(LSI.N = 0.0460 ,   { Lower = -1, Upper = 1, MultiplicativeScaleFact
     Vary SQP1(LSI.B = 0.1086 ,   { Lower = -1, Upper = 1, MultiplicativeScaleFact
     Maneuver LSI(MMSRef);

     Propagate LunarSB(MMSRef, {MMSRef.ElapsedDays = 1.5});
     Propagate MoonCentered(MMSRef, {MMSRef.Luna.Periapsis, MMSRef.ElapsedDays = 8

     NonlinearConstraint SQP1(MMSRef.MoonMJ2000Eq.BdotT <= 20000.0);
     NonlinearConstraint SQP1(MMSRef.Luna.RMAG >= 2000);
     NonlinearConstraint SQP1(MMSRef.MoonMJ2000Eq.BdotR <= 15000.0);

     Vary SQP1(LOI.V = -0.75198120104, { Lower = -1, Upper = 1});
     Vary SQP1(LOI.N = 0, {  Lower = -1, Upper = 1});
     Vary SQP1(LOI.B = 0, { Lower = -1, Upper = 1});
     Maneuver LOI(MMSRef);

     NonlinearConstraint SQP1(MMSRef.Luna.SMA = 7600);
     NonlinearConstraint SQP1(MMSRef.Luna.ECC = 0.2);
     NonlinearConstraint SQP1(MMSRef.MoonMJ2000Eq.INC = 85);

     Propagate MoonCentered(MMSRef, {MMSRef.Luna.Periapsis, MMSRef.ElapsedDays = 5

     dV1Mag = sqrt( LSI.Element1^2 + LSI.Element2^2 + LSI.Element3^2 );
     dV2Mag = sqrt( LOI.Element1^2 + LOI.Element2^2 + LOI.Element3^2 );
     TotaldV = ( dV1Mag + dV2Mag );
     Minimize SQP1(TotaldV);
```

```
        Report IterateData TotaldV  MMSRef.Luna.SMA MMSRef.Luna.ECC MMSRef.MoonMJ2000

        Toggle LunarINC On;
        Toggle LunarSMA On;
        Toggle LunarECC On;
        Toggle TotaldVPlot On;
        Propagate LunarSB(DummySat);
        Toggle LunarINC Off;
        Toggle LunarSMA Off;
        Toggle LunarECC Off;
        Toggle TotaldVPlot Off;

    EndOptimize

    Report IterateData TotaldV dV1Mag dV2Mag
    Report IterateData MMSRef.Luna.SMA MMSRef.Luna.ECC MMSRef.MoonMJ2000Eq.INC;
    Report IterateData LSI.Element1 LSI.Element2 LSI.Element3
    Report IterateData LOI.Element1 LOI.Element2 LOI.Element3

    Propagate MoonCentered(MMSRef, {MMSRef.ElapsedDays = 15});
```

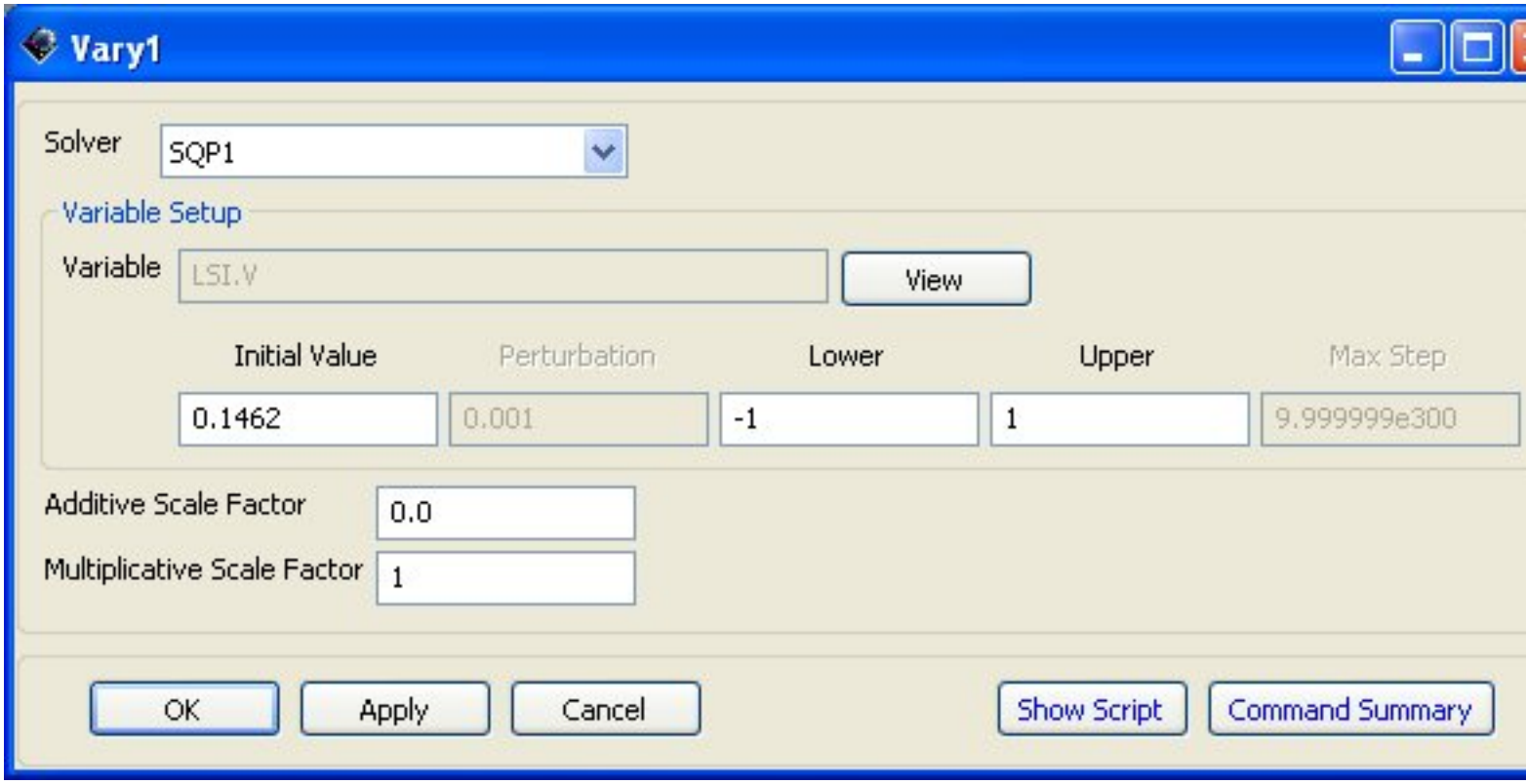


Figure: Vary Command

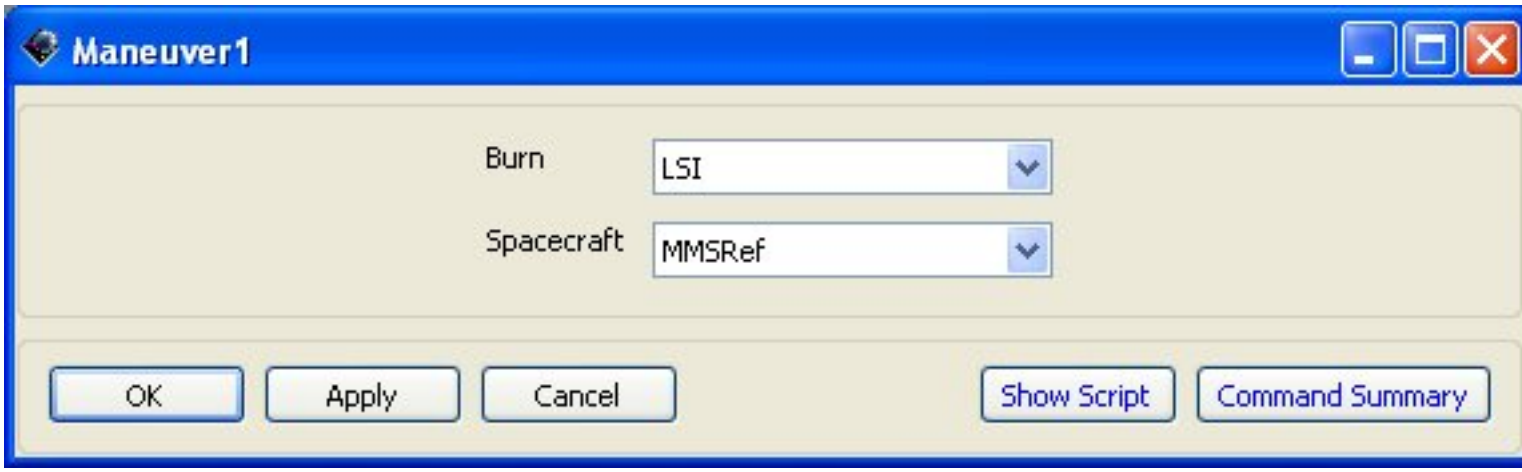


Figure: Maneuver Command

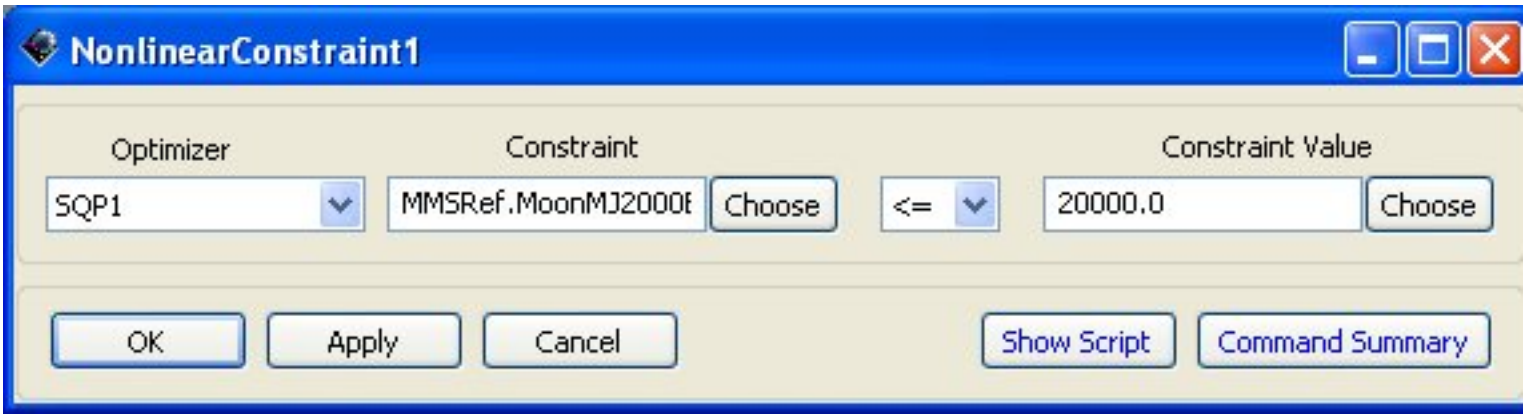


Figure: Nonlinear Constant Command

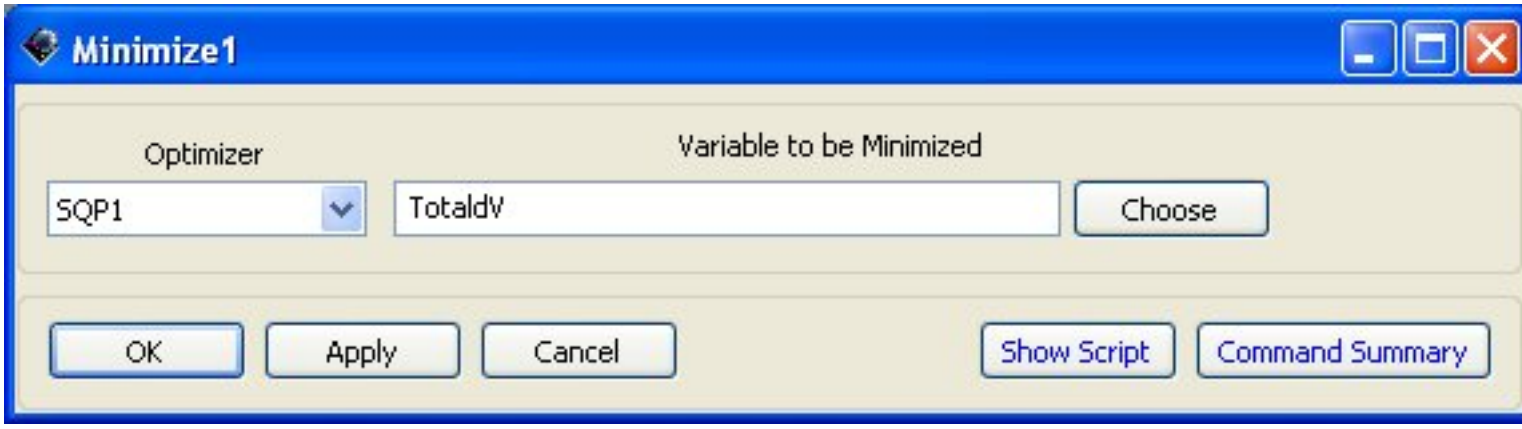


Figure: Minimize Command

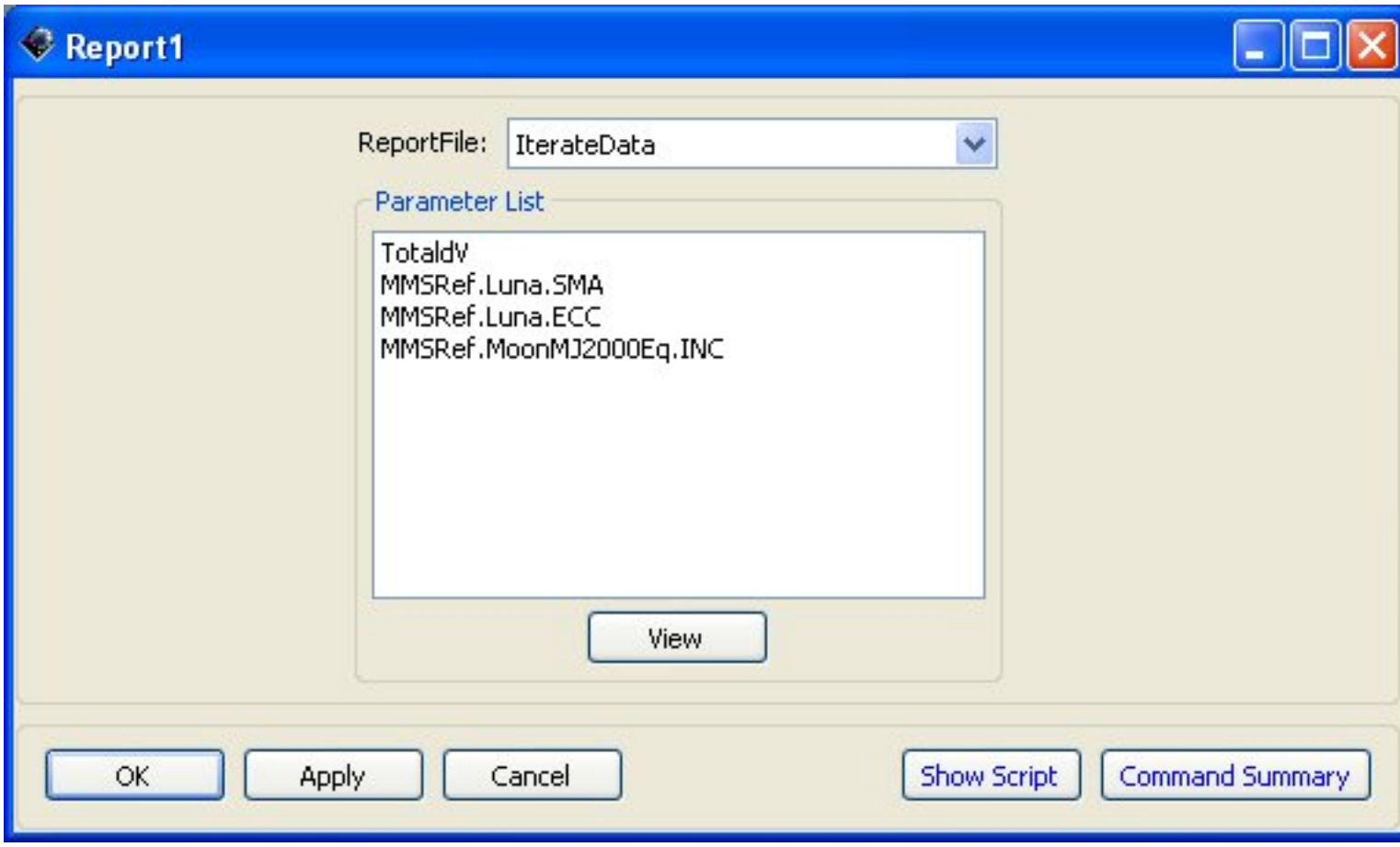


Figure: Report Command

Running the Mission

Resource Mission and Output Trees

The XY plots should look like the ones observed in the screenshot. Notice that the SMA, INC, and ECC plots converge upon the desired values. Additionally, notice how the TotaldVPlot decreases as time goes on. The values that are lower in this plot, on the left side, are iterations that did not complete the required objectives. The Moon plot should show multiple iterations eventually converging upon an iteration that circles the moon. The Earth plot shows a straight line from the Earth to the path of the moon, where you can see the spacecraft follow a sinusoidal shape along the moon's path. This shows the spacecraft orbiting the moon. The spacecraft does not travel in a straight line, this is due to a mistake in the programming of when the plot is on.

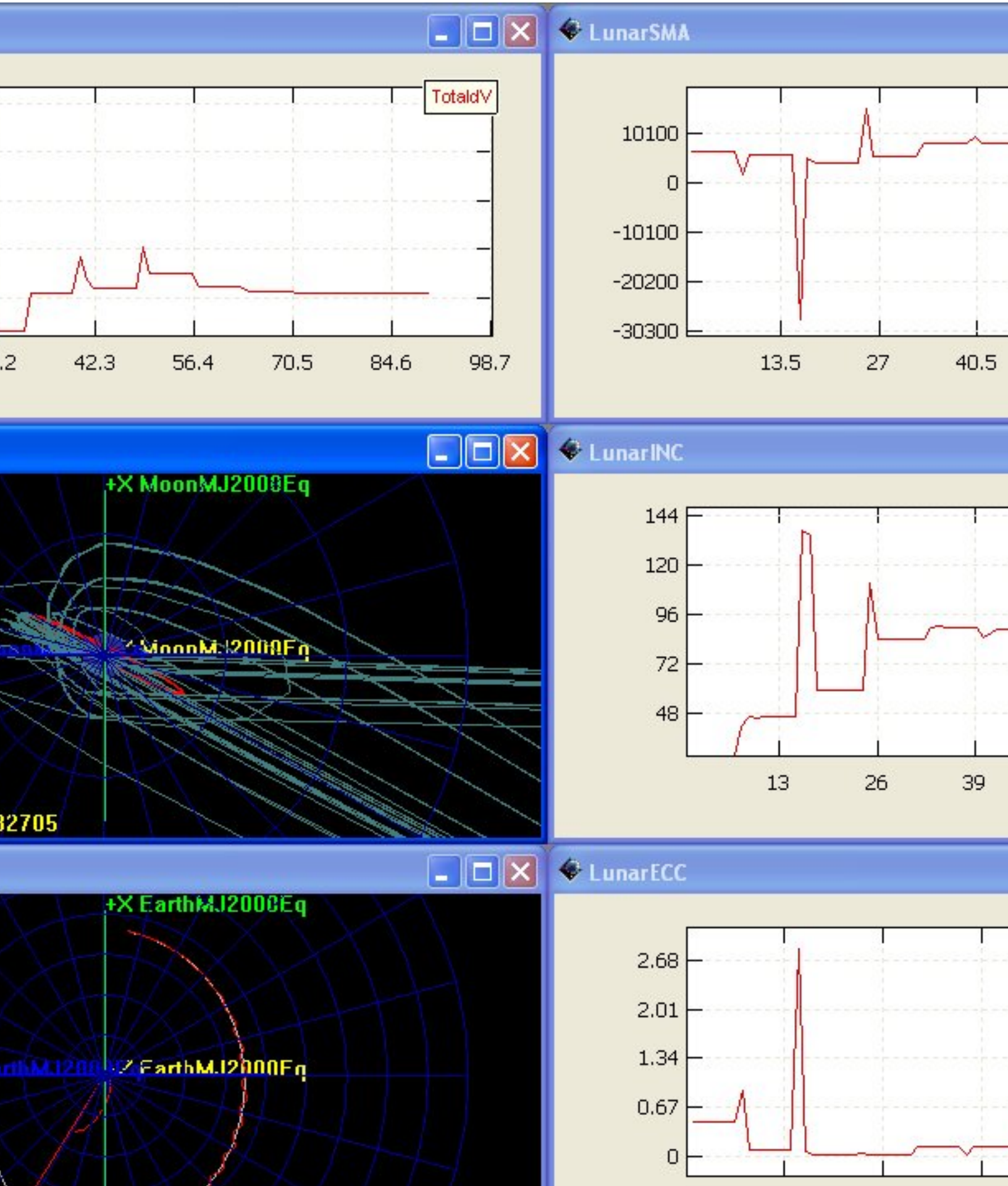

TotaldV
.2
42.3
56.4
70.5
84.6
98.7
LunarSMA
10100
0
-10100
-20200
-30300
13.5
27
40.5
+X MoonMJ2000Eq
32705
LunarINC
144
120
96
72
48
13
26
39
+X EarthMJ2000Eq
LunarECC
2.68
2.01
1.34
0.67
0

Figure: Output

Chapter 23. LEO Station Keeping

This tutorial was started from the default settings in the June 9, 2008 build of GMAT.

Objective and Overview

This basic tutorial explains how to perform basic station keeping of a spacecraft's semi-major axis and right ascension of the ascending node while in low Earth orbit. The strategy used here to keep the spacecraft in its orbit is somewhat crude compared to that used by actual spacecraft. However, this information is still valuable in that it provides a good example for using control logic and targeting sequences in the same mission.

Download the script file: LEOStationKeepingTutorial.script [scripts/LEOStationKeepingTutorial.script]

Prerequisites

- Knowledge
  - Basic understanding of Keplarian orbital parameters
  - Basic understanding of low Earth orbit station keeping
- Tutorials:
  - Creaitng and Propagating a Spacecraft
  - Designing a Hohmann Transfer
- Objects:
  - Spacecraft Object-Orbit Tab
  - Impulsive Burn Object
  - Propagator Object
  - Differential Corrector Object
  - OpenGL Plot Object
  - Coordinate System Objects
- Commands:
  - While Command
  - If
  - Maneuver Command
  - Propagate Command

Mission Description

- Objective

    - Maintain a low Earth orbit by making corrections to the spacecraft's altitude and right ascension of the ascending node.

- Assume:
    - A spacecraft in a preexisting orbit with a given epoch and orbital parameters
    - A realistic force model including non-spherical gravity and atmospheric drag
- Find:
    - The deltaV required in the spacecraft's direction of travel to keep it in orbit within the given altitude
    - The deltaV required perpendicular to the spacecraft's direction of travel to keep its right ascension of the ascending node from drifting with respect to the inertial frame

Resource, Mission, and Output trees

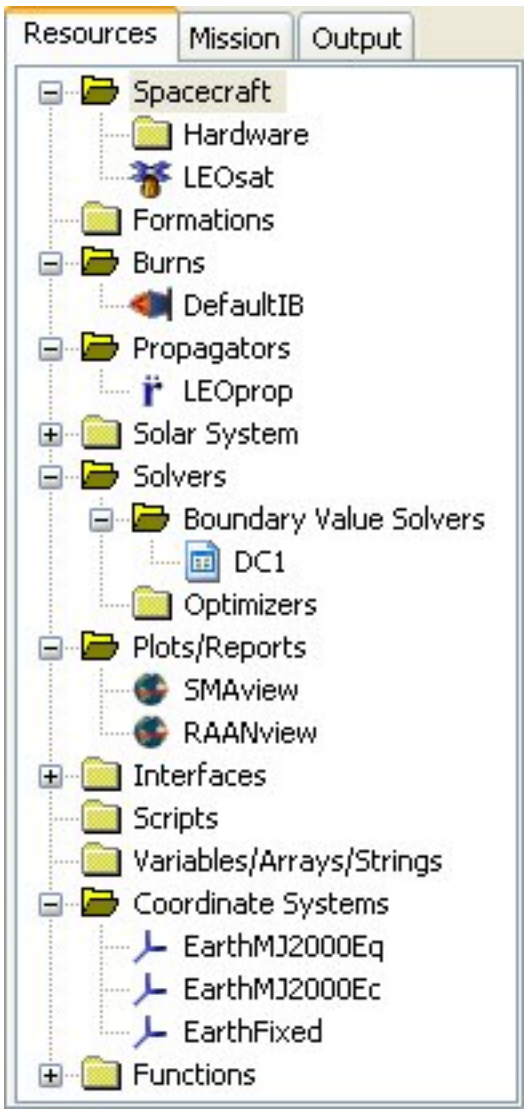


Figure: The Resource, Mission, and Output trees for this particular LEO station keeping mission

Creating and Configuring the Resource Tree

This section will outline exactly what resources are required to complete the mission outlined within this tutorial. The spacecraft Sat will be in a highly inclined orbit of nearly 97 degrees. As stated previously, these resources will give you the tools necessary to keep LEOsat in its orbit.

Objects Required

- Spacecraft: LEOsat
- Burns: DefaultIB (default)
- Propagators: LEOprop
- Solvers: DC1
- Plots/Reports: SMAview , RAANview

Creating and Modifying Objects

- Spacecraft
  - Rename DefaultSC to LEOsat
  - Change the Epoch Format to UTCGregorian
  - Set the Epoch to Jan 01 2000 11:59:28.000
  - Switch the State Type to Keplarian
  - Set SMA = 6738.95113401 , ECC = 0.00168481942805 , INC = 96.8535019521 , RAAN = 279.829547746 , AOP = 246.130924803 , TA = 308.589822969
  - Note that when you click 'Apply' in this window, some of the values for your orbital elements may change but this is normal and does not require you to change them back again

  Script Syntax:

```
Create Spacecraft LEOsat;
GMAT LEOsat.DateFormat = UTCGregorian;
GMAT LEOsat.Epoch = '01 Jan 2000 11:59:28.000';
GMAT LEOsat.CoordinateSystem = EarthMJ2000Eq;
GMAT LEOsat.DisplayStateType = Keplerian;
GMAT LEOsat.SMA = 6738.95113401;
GMAT LEOsat.ECC = 0.001684819428049389;
GMAT LEOsat.INC = 96.8535019521;
GMAT LEOsat.RAAN = 279.829547746;
GMAT LEOsat.AOP = 246.1309248029877;
GMAT LEOsat.TA = 308.5898229690123;
```

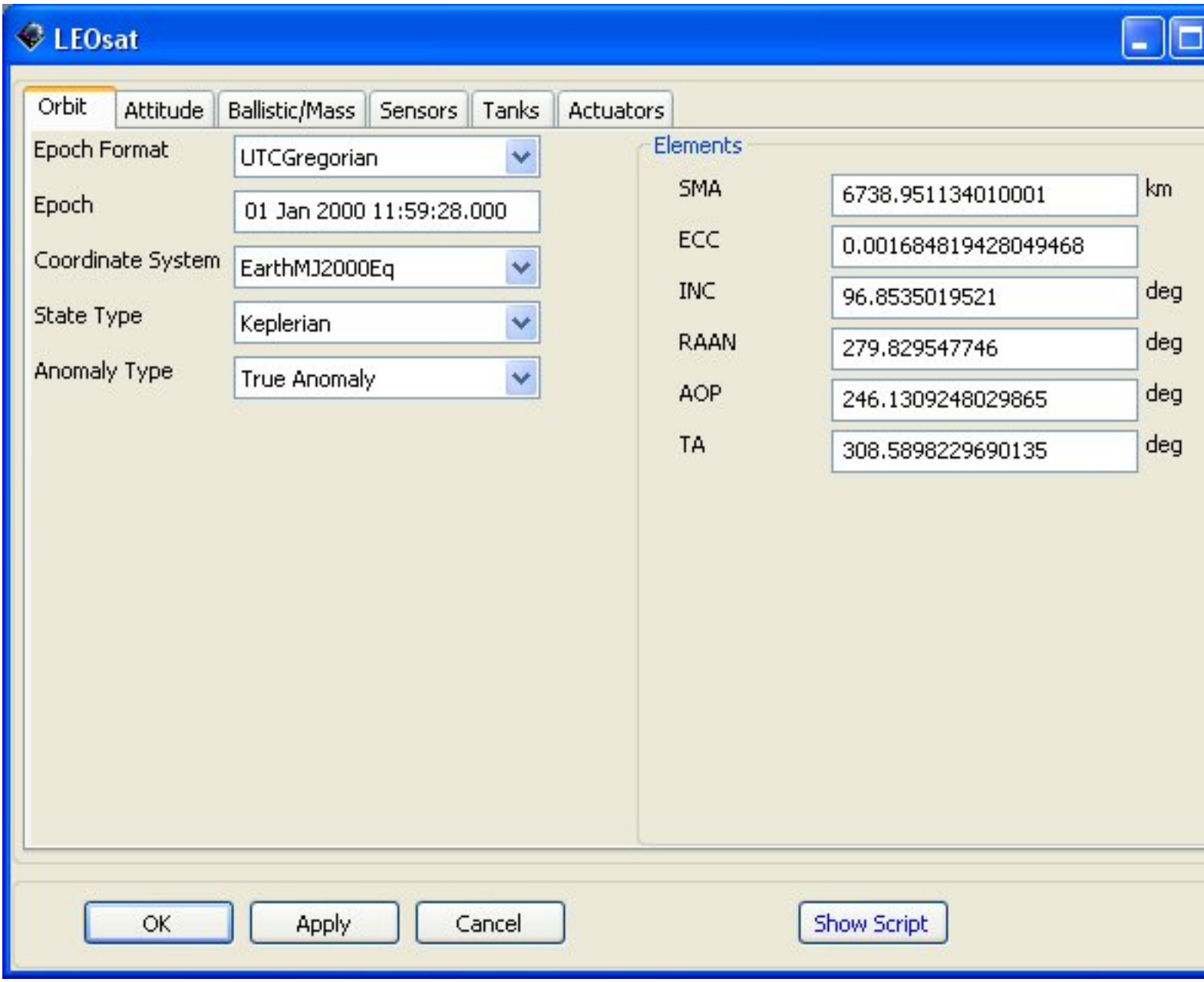


Figure: LEOsat Spacecraft Object

- Propagators
  - Rename DefaultProp to LEOprop
  - Change both the Degree and Order of the gravity model to 10 to make it more realistic
  - Set the Atmosphere Model to JacchiaRoberts
  - Add the Sun, and Luna as Point Masses
  - Check the Use Solar Radiation Pressure box

Script Syntax:

```
Create ForceModel LEOprop_ForceModel;
GMAT LEOprop_ForceModel.CentralBody = Earth;
GMAT LEOprop_ForceModel.PrimaryBodies = {Earth};
GMAT LEOprop_ForceModel.PointMasses = {Sun, Luna};
GMAT LEOprop_ForceModel.Drag = JacchiaRoberts;
GMAT LEOprop_ForceModel.SRP = On;
GMAT LEOprop_ForceModel.ErrorControl = RSSStep;
GMAT LEOprop_ForceModel.GravityField.Earth.Degree = 10;
GMAT LEOprop_ForceModel.GravityField.Earth.Order = 10;
GMAT LEOprop_ForceModel.GravityField.Earth.PotentialFile = 'JGM2.cof';
GMAT LEOprop_ForceModel.Drag.F107 = 150;
GMAT LEOprop_ForceModel.Drag.F107A = 150;
GMAT LEOprop_ForceModel.Drag.MagneticIndex = 3;
GMAT LEOprop_ForceModel.SRP.Flux = 1367;
GMAT LEOprop_ForceModel.SRP.Nominal_Sun = 149597870.691;

Create Propagator LEOprop;
GMAT LEOprop.FM = LEOprop_ForceModel;
GMAT LEOprop.Type = RungeKutta89;
GMAT LEOprop.InitialStepSize = 60;
GMAT LEOprop.Accuracy = 9.999999999999999e-012;
GMAT LEOprop.MinStep = 0.001;
GMAT LEOprop.MaxStep = 2700;
GMAT LEOprop.MaxStepAttempts = 50;
```

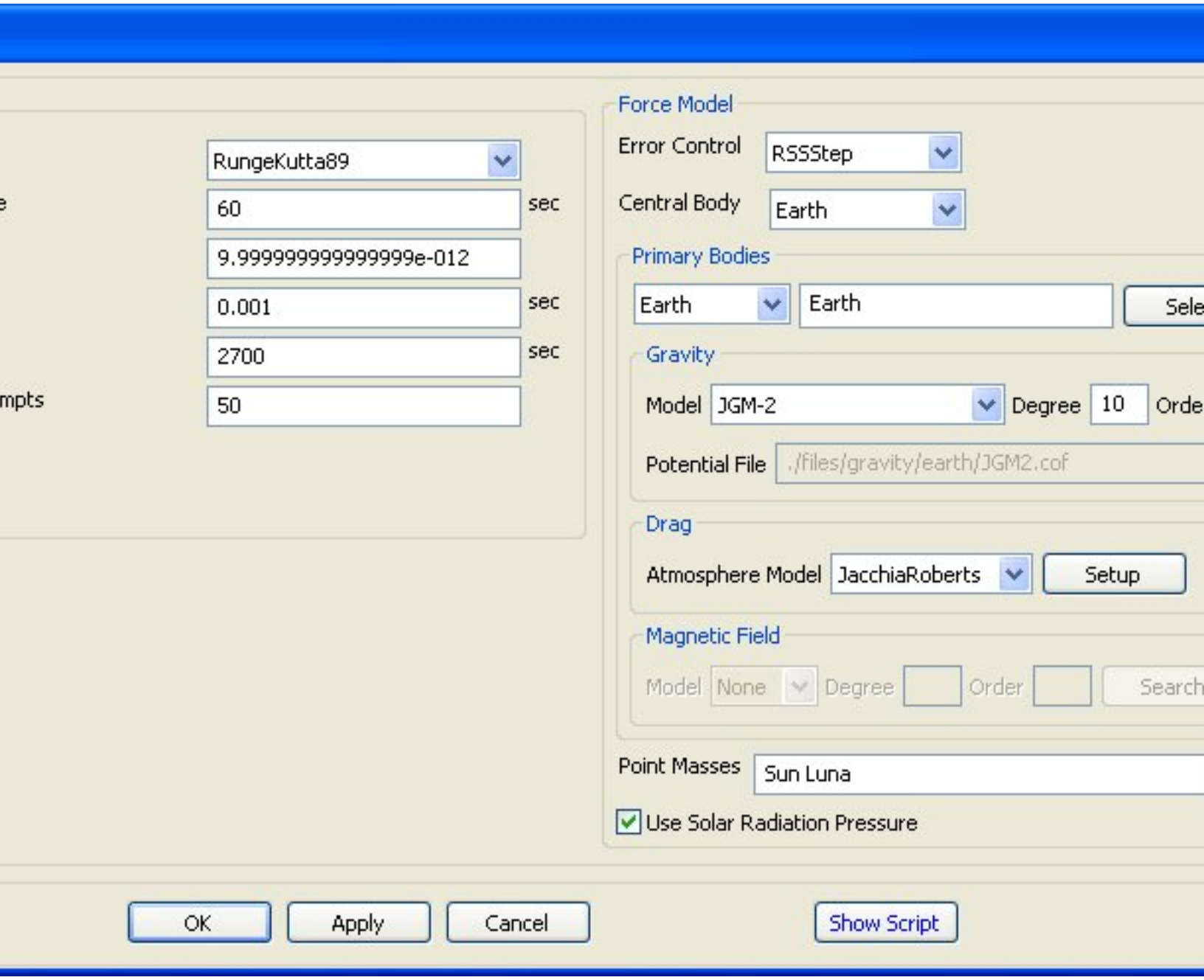


Figure: LEOprop Propagator Object

- Solvers
  - Create a Differential Corrector Object under Solvers > Boundary Value Solvers

  Script Syntax:

```
Create DifferentialCorrector DefaultDC;
GMAT DefaultDC.ShowProgress = true;
GMAT DefaultDC.ReportStyle = Normal;
```

```
GMAT DefaultDC.TargeterTextFile = targeter_DefaultDC.data;
GMAT DefaultDC.Variables = {dV.V, dV.N};
GMAT DefaultDC.MaximumIterations = 25;
GMAT DefaultDC.UseCentralDifferences = false;
```

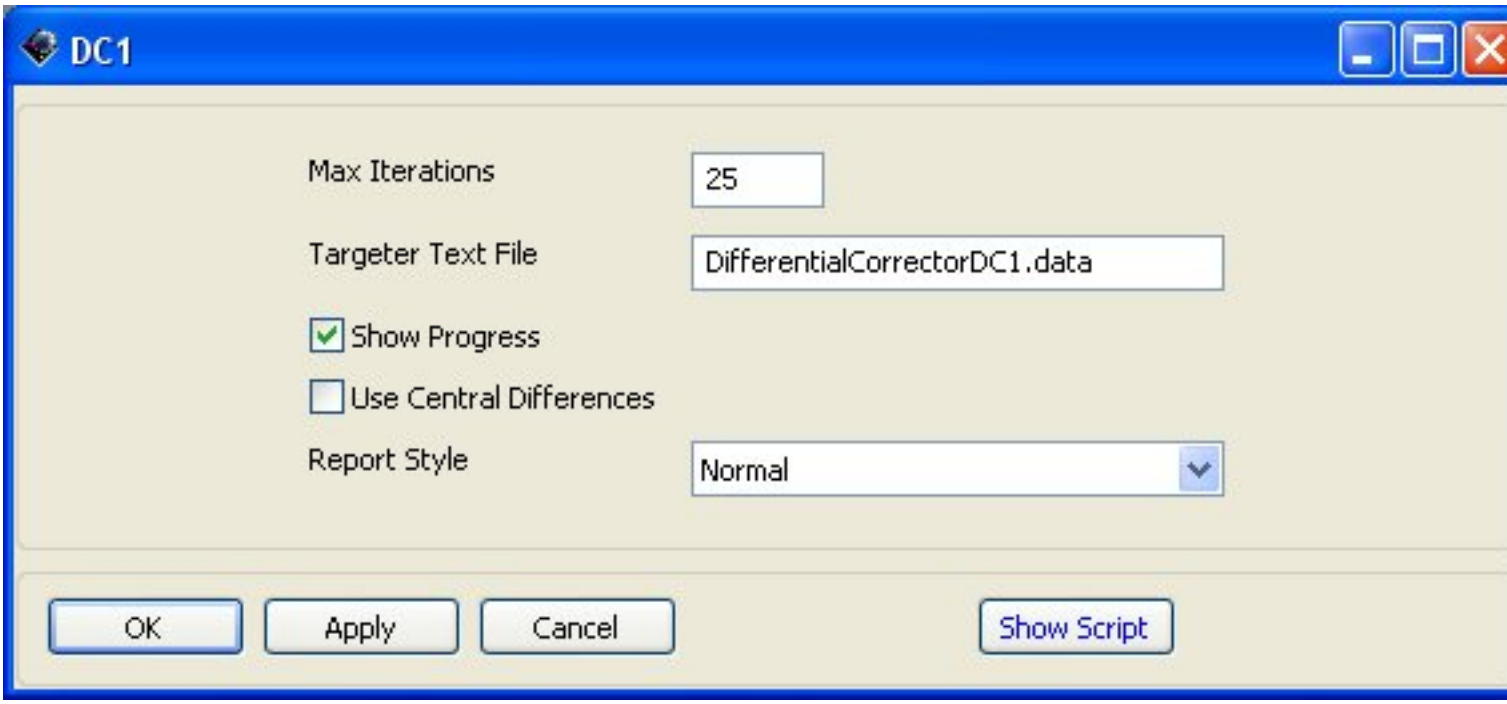


Figure: DefaultDC Boundary Value Solver Object

- Plots/Reports
  - Rename DefaultOpenGL to SMAview
    - Uncheck Draw XY Plane
    - Change the View Point Vector to [15000 , 3000 , 0]

    Script Syntax:

```
Create OpenGLPlot SMAview;
GMAT SMAview.SolverIterations = Current;
GMAT SMAview.Add = {LEOsat, Earth};
GMAT SMAview.OrbitColor = [ 255 32768 ];
GMAT SMAview.TargetColor = [ 8421440 0 ];
GMAT SMAview.CoordinateSystem = EarthMJ2000Eq;
GMAT SMAview.ViewPointReference = Earth;
GMAT SMAview.ViewPointVector = [ 15000 3000 0 ];
GMAT SMAview.ViewDirection = Earth;
GMAT SMAview.ViewScaleFactor = 1;
GMAT SMAview.ViewUpCoordinateSystem = EarthMJ2000Eq;
GMAT SMAview.ViewUpAxis = Z;
GMAT SMAview.XYPlane = Off;
```

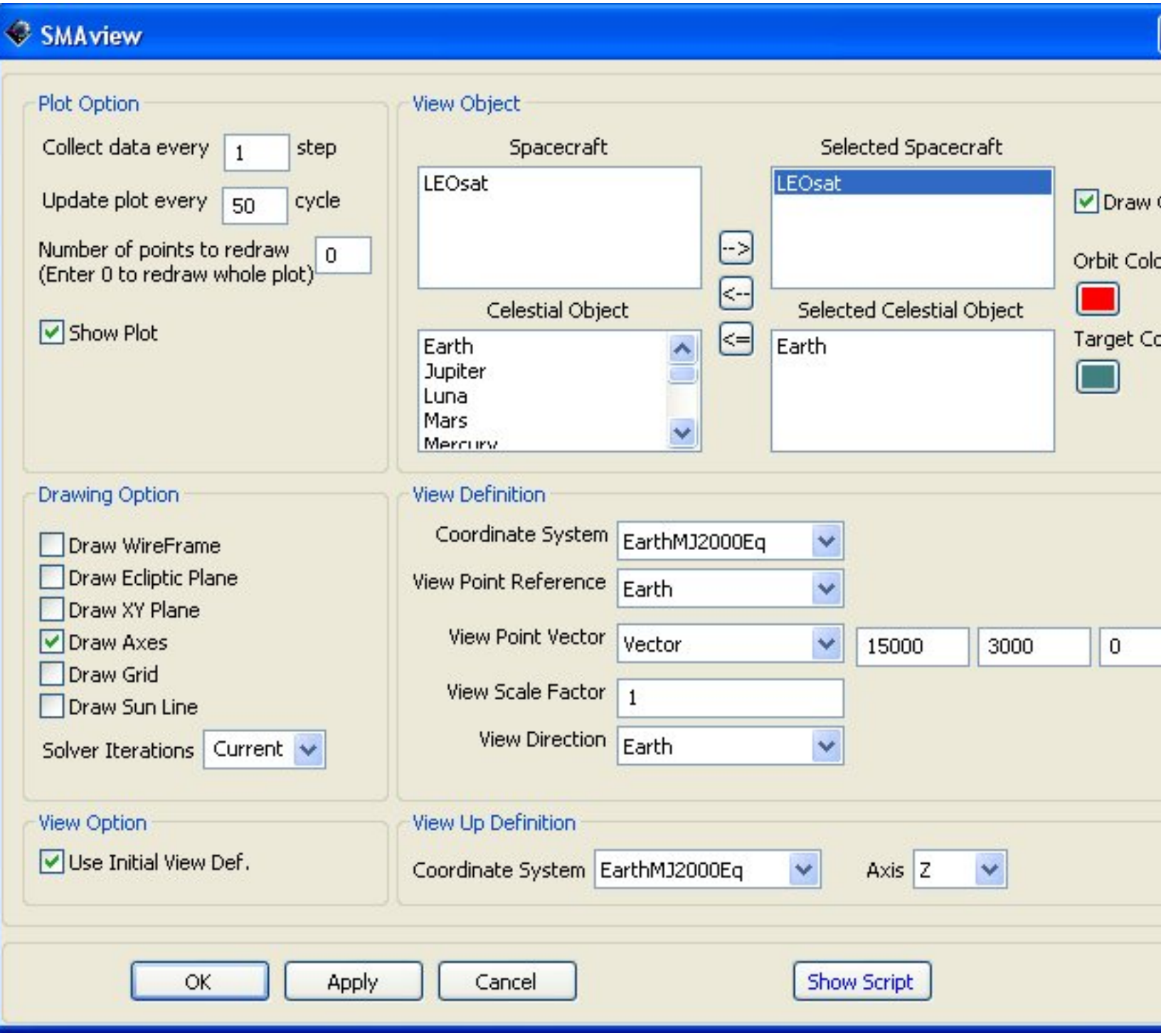


Figure: L1View OpentGL Plot Object

- Create an OpenGL Plot named RAANview
  - Add Earth to the Selected Celestial Objects list
  - Uncheck Draw XY Plane

- Change the View Point Vector to [1800 , 0 , 15000]
- Change the View Up Definition Axis to X.

Script Syntax:

```
Create OpenGLPlot RAANview;
GMAT RAANview.SolverIterations = Current;
GMAT RAANview.Add = {LEOsat, Earth};
GMAT RAANview.OrbitColor = [ 255 1743054 ];
GMAT RAANview.TargetColor = [ 8421440 8421440 ];
GMAT RAANview.CoordinateSystem = EarthMJ2000Eq;
GMAT RAANview.ViewPointReference = Earth;
GMAT RAANview.ViewPointVector = [ 1800 0 15000 ];
GMAT RAANview.ViewDirection = Earth;
GMAT RAANview.ViewScaleFactor = 1;
GMAT RAANview.ViewUpCoordinateSystem = EarthMJ2000Eq;
GMAT RAANview.ViewUpAxis = X;
GMAT RAANview.CelestialPlane = Off;
GMAT RAANview.XYPlane = Off;
```

Creating and Configuring the Mission Sequence

Image of Mission Tree

The mission sequence presented here is relatively simple one. As said before, it is a good example of how to use loops, if statements, and targeting loops to achieve a goal. Again, this is not a perfect representation of how to efficiently implement low Earth orbit station keeping but it does work. How well it works and how 'tight' your orbit variation is will be up to you. Once you have implemented this version of the mission sequence, you may go back through and change the If statements so that they allow LEOsat to drift from its orbit more. This will increase the orbit's deviation but decrease the amount of fuel required.

- Insert a While after Propagate1
  - Change the right hand side value to 10
- After While1, insert another Propagate
  - Remove the propagate's stopping condition parameter and condition so that both fields are blank
- Add the first of two If statements now
  - Change the left hand side to LEOsat.Earth.SMA
  - Change the right hand side to 6738.5
- Insert a Target at this point
- Append this with a Vary, a Maneuver, and an Achieve
  - Change the Goal to LEOsat.Earth.SMA
  - Set the Value as 6738.951134009997

  - Set the Tolerance as 0.05
- Insert another If after EndIf1
  - Set the left hand side as LEOsat.EarthMJ2000Eq.RAAN
  - Change the right hand side to 279.709
  - Add an '|' symbol under the 'If' to continue the statement
  - Set the left hand side of this second expression to the same LEOsat.EarthMJ2000Eq.RAAN
  - Set the Condition as greater than, '>'
  - Add 279.949 as the right hand side value
- Insert a final propagate after If2
  - Set the stopping condition parameter as LEOsat.EarthMJ2000Eq.VZ
- Insert the last target now
- Append this target with another set of vary, maneuver, and achieve so as to match the previous sequence
  - Change the Variable of Vary2 to DefaultIB.N
  - Also set the lower of Vary 2 as -3.14159
  - LEOsat.EarthMJ2000Eq.RAAN should be the Goal for Achieve2
  - 279.829547746 is Achieve2's Value

Script Syntax:

```
Propagate LEOprop(LEOsat) {LEOsat.ElapsedSecs = 8640.0};
While LEOsat.ElapsedDays < 10
   Propagate LEOprop(LEOsat);
   If LEOsat.Earth.SMA < 6738.5
      Target DC1;
         Vary DC1(DefaultIB.V = 0.5, {Perturbation = 0.0001, MaxStep = 0.2, Lower
         Maneuver DefaultIB(LEOsat);
         Achieve DC1(LEOsat.Earth.SMA = 6738.951134009997, {Tolerance = 0.05});
      EndTarget;  % For targeter DefaultDC
   EndIf;
   If LEOsat.EarthMJ2000Eq.RAAN < 279.709 | LEOsat.EarthMJ2000Eq.RAAN > 279.949
      Propagate LEOprop(LEOsat) {LEOsat.EarthMJ2000Eq.VZ = 0.0};
      Target DC1;
         Vary DC1(DefaultIB.N = 0.5, {Perturbation = 0.0001, MaxStep = 0.2, Lower
         Maneuver DefaultIB(LEOsat);
         Achieve DC1(LEOsat.EarthMJ2000Eq.RAAN = 279.829547746, {Tolerance = 0.1})
      EndTarget;  % For targeter DefaultDC
   EndIf;
EndWhile;
```

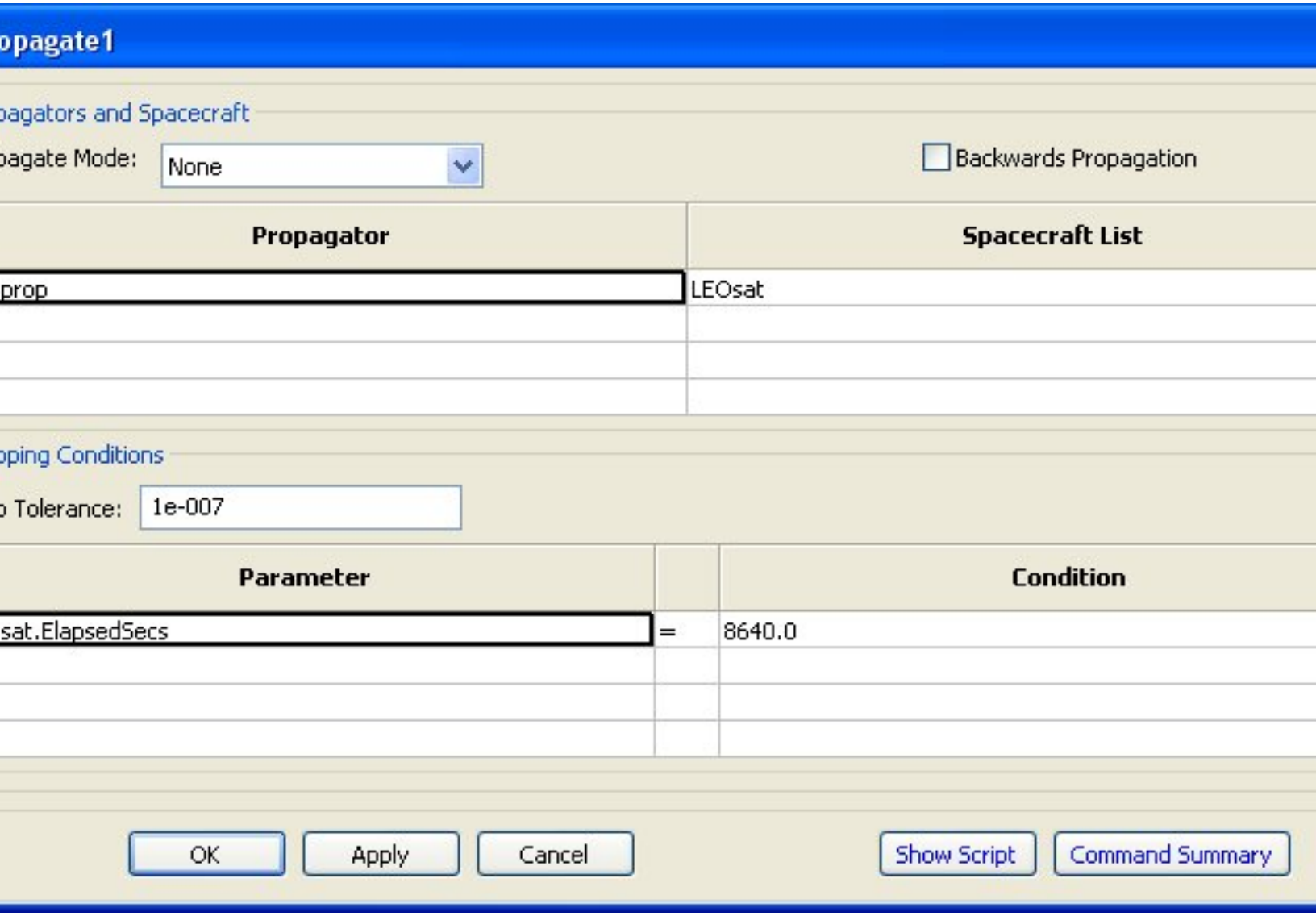


Figure: Propagate1 Propagate Command

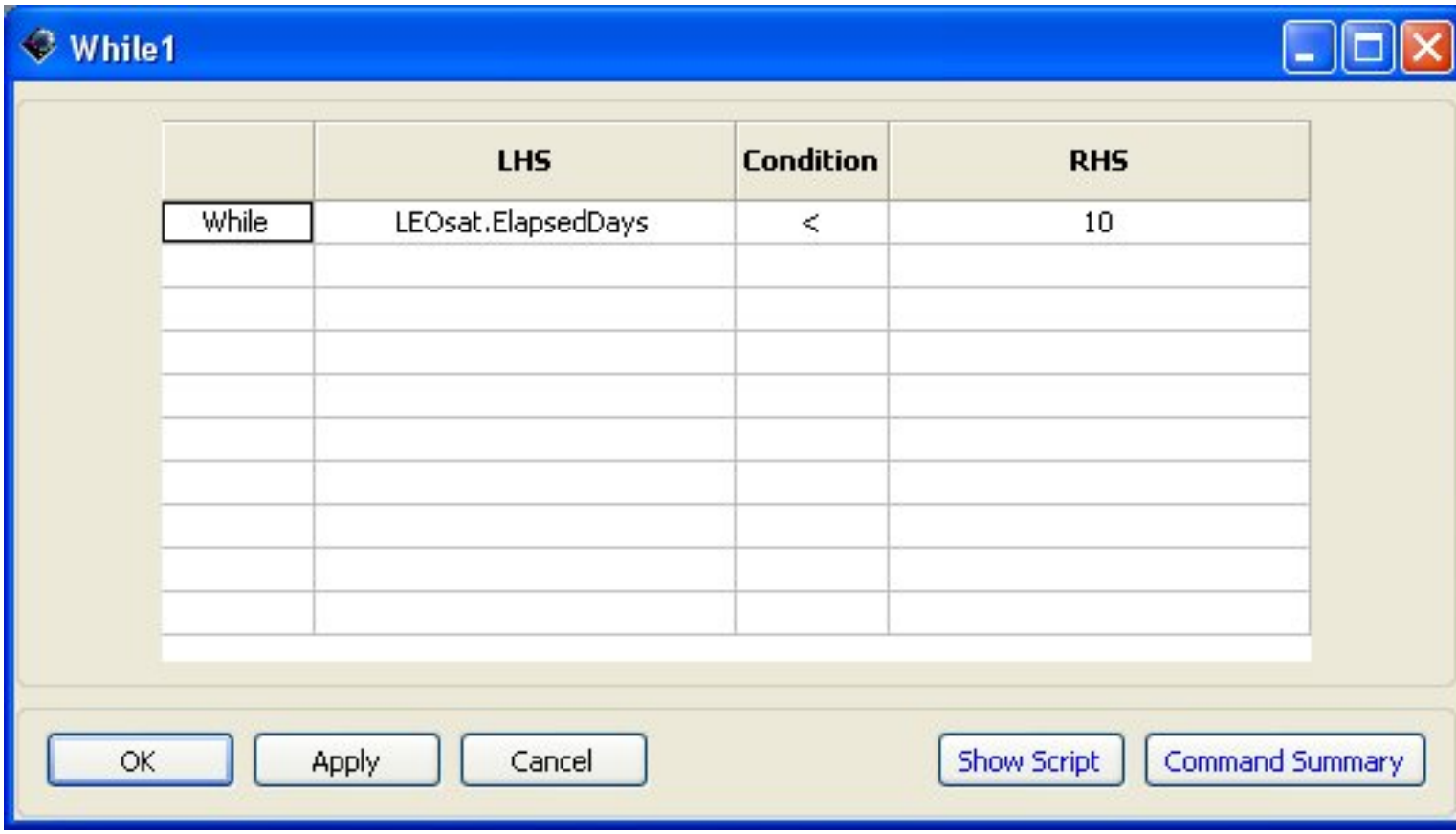


Figure: While1 While Command

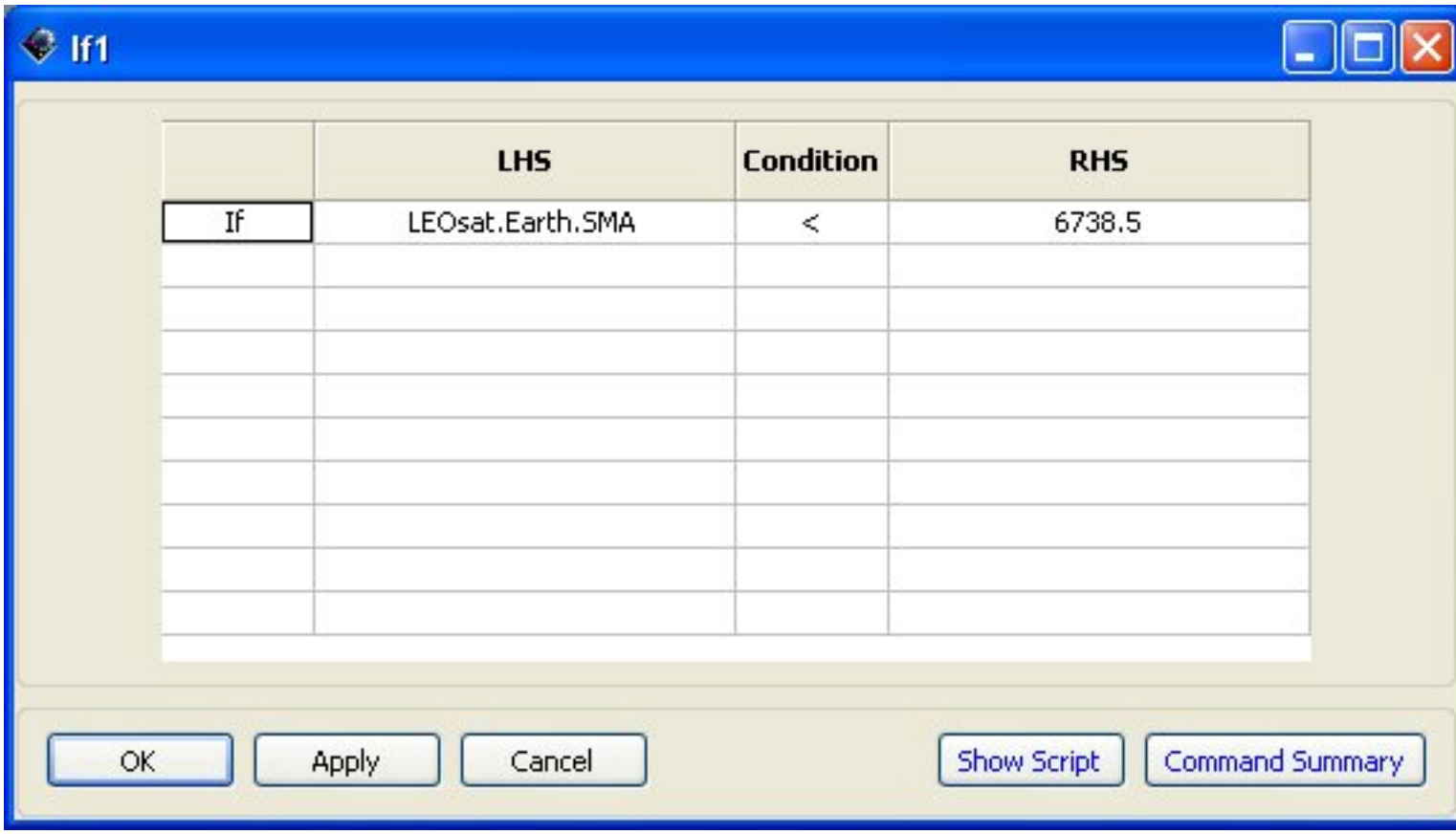


Figure: If1 If Command

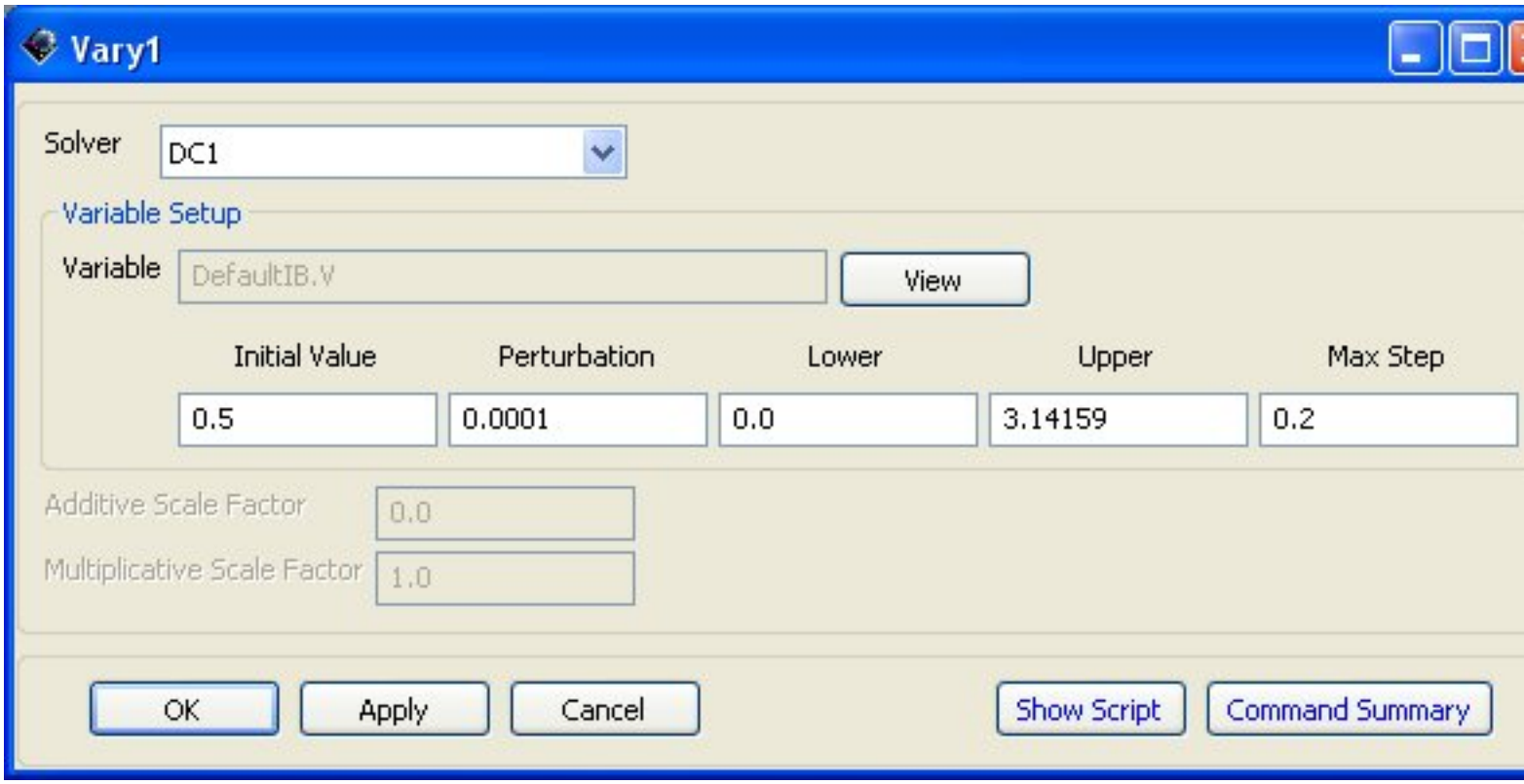


Figure: Vary1 Vary Command

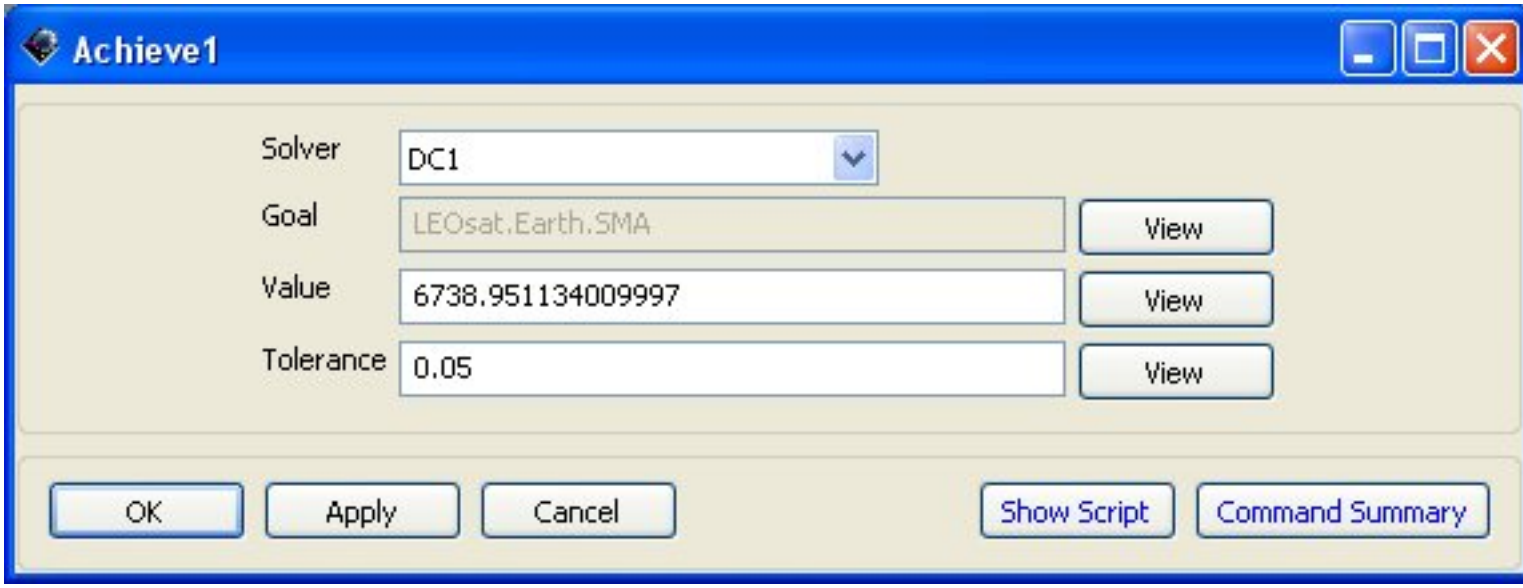
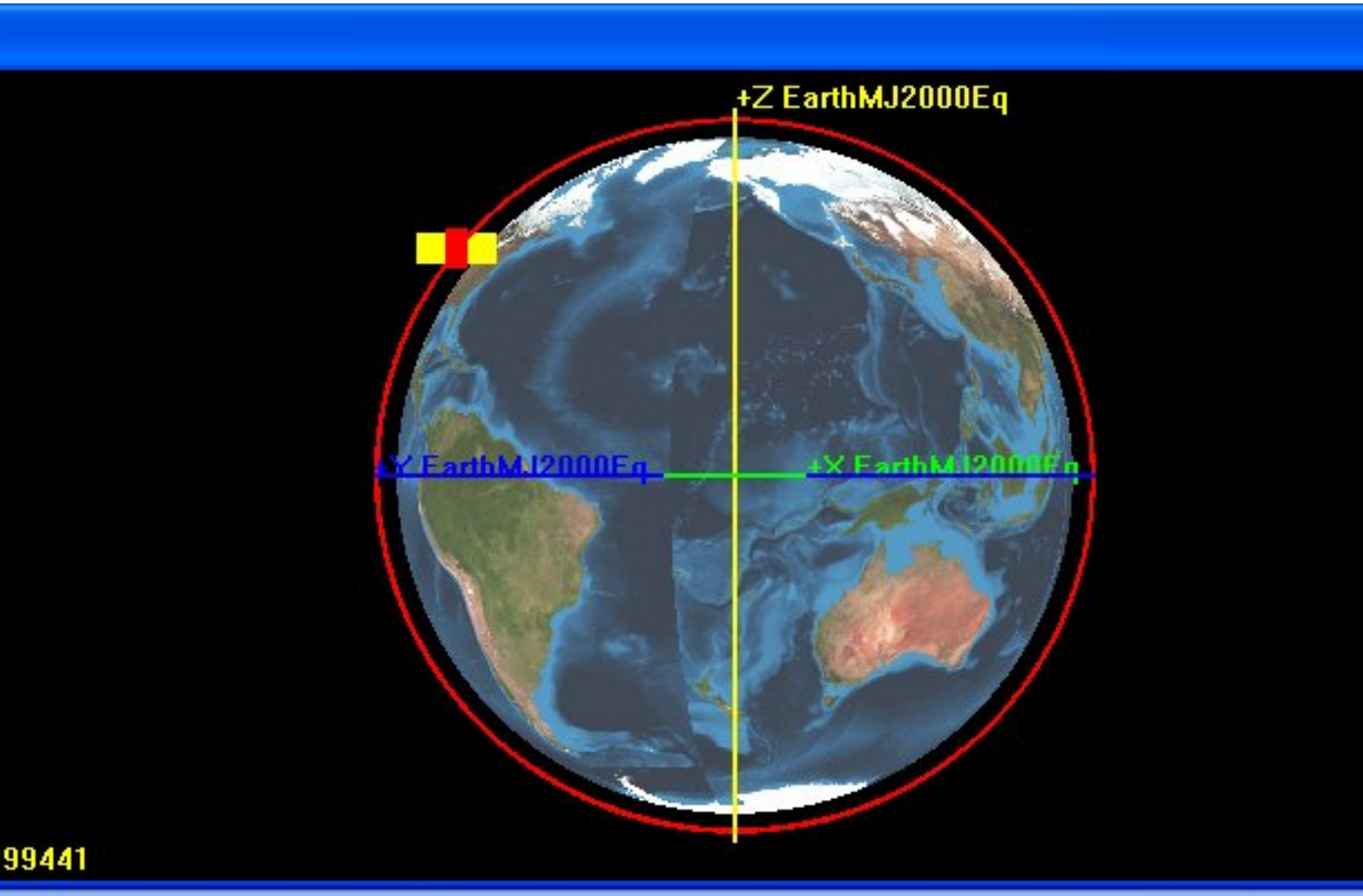


Figure: Achieve1 Achieve Command

Running the Mission

Image of Output Tree

The views that you created are good for seeing how well the mission sequence is keeping LEOsat in its orbit. The SMA view shows some thickness to it while the RAAN view shows barely any at all. You may now try and change the mission parameters as outlined in the mission tree section to see how the spacecraft's drift is affected.


+Z EarthMJ2000Eq
+Y EarthMJ2000Eq
+X EarthMJ2000Eq
99441

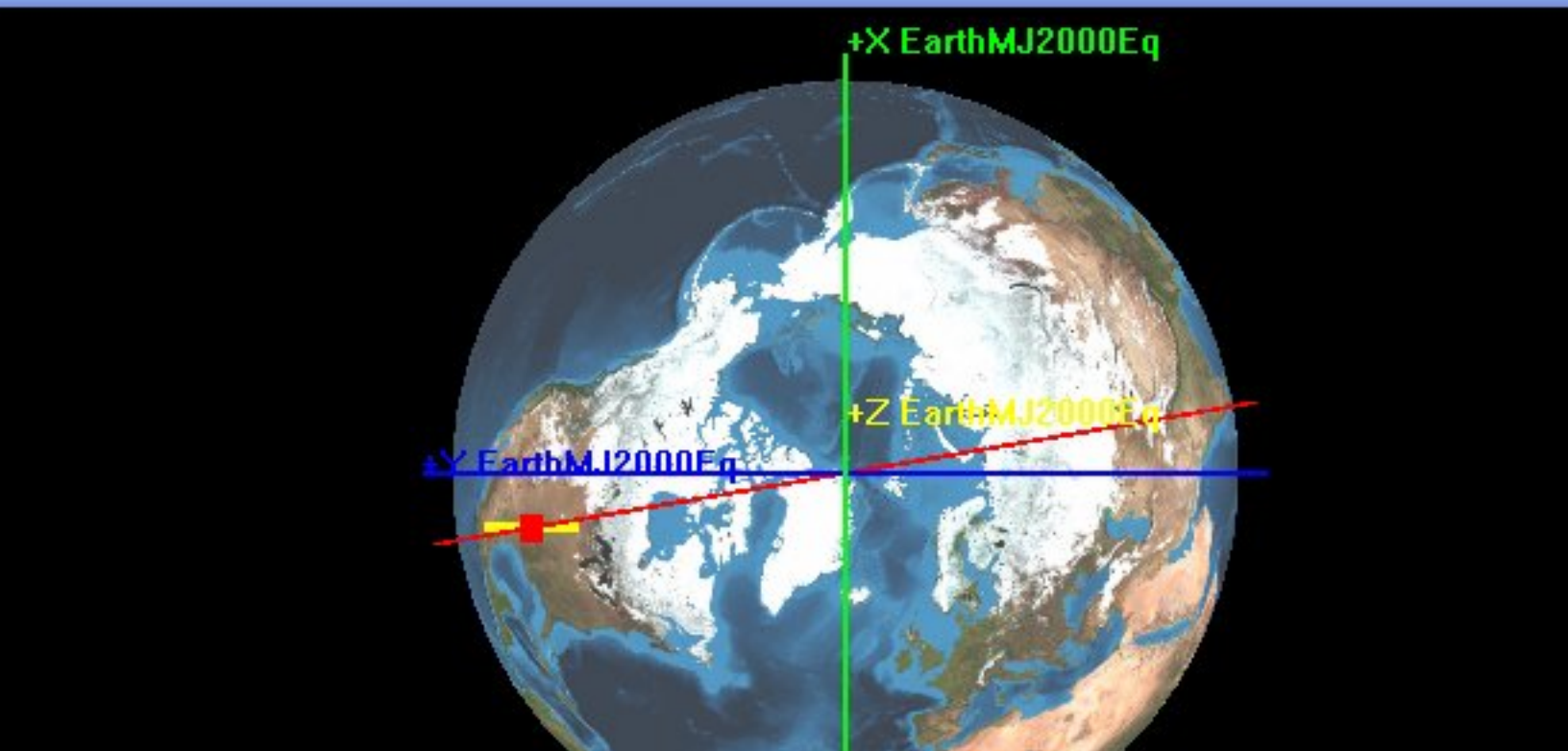

+X EarthMJ2000Eq
+Z EarthMJ2000Eq
+Y EarthMJ2000Eq

Figure: Output

Part IV. Reference Material

Table of Contents

24. Resources 272
- Spacecraft 272
  - Spacecraft Overview 272
  - Orbit Tab 273
  - Attitude Tab 283
  - Ballistic Mass Tab 289
- Propagator 291
  - Overview 291
  - Interfaces 292
  - Reference Table 294
  - Object and Command Interactions 299
- ReportFile 299
  - Overview 299
  - Interfaces 300
  - Reference Table 300
  - Objects and Commands Interactions 302
- Fuel Tank 302
  - Overview 302
  - Using the GUI 303
  - Using the script 304
  - Reference Table 304
- Thruster 305
  - Overview 305
  - Using the GUI 306
  - Using the script 307
  - Reference Table 308
- SQP(fmincon) 313
  - Object Overview 313
  - Interface Overview 314
  - Field Descriptions 314
  - Using an fminconOptimizer 317
- Solar System 317
  - Overview 317
  - Using the GUI 317
  - Reference Table 318
- DifferentialCorrector 319
  - Overview 319
  - Interfaces 320
  - Reference Table 320
  - Object and Command Interactions 322
- Celestial Bodies 322
  - Overview 322
  - Using the GUI 323
  - Using the script 323
  - Reference Table 324
- OpenGL Plot 327
  - Overview 327
  - Using the GUI 327
  - Using the Script 331
  - Reference Tables 332
- XYPlot 338

Overview ... 338
Using the GUI ... 338
Using the script ... 339
Reference Table ... 339
Coordinate Systems ... 341
Overview ... 341
Using the GUI ... 341
Using the script ... 343
Finite Burn ... 343
Overview ... 344
Reference Table ... 344
Examples ... 345
Objects and Commands Interactions ... 345
Variable ... 345
Overview ... 346
Interfaces ... 346
Object and Commands Interactions ... 346
String ... 347
Overview ... 347
Using the GUI ... 347
Using the script ... 348
Array ... 349
Overview ... 349
Using the GUI ... 349
Using the script ... 350
Reference Table ... 350
Barycenter ... 350
Overview ... 351
Using the GUI ... 351
Using the Script ... 352
Reference Table ... 352
Libration Point ... 353
Overview ... 353
Using the GUI ... 353
Using the Script ... 354
Reference Table ... 354
Impulsive Burn ... 355
Overview ... 355
Using the GUI ... 355
Using the Script ... 357
Reference Table ... 357
25. Commands ... 359
Propagate ... 359
Overview of the Propagate Command ... 359
Using the Propagate Command in the GUI ... 359
Using the Propagate Command in the script ... 361
Reference Table ... 362
While ... 364
Overview ... 364
Using GUI ... 364
Using the Script ... 365
Reference Table ... 366
If ... 366
Overview ... 366

Using the GUI 367
Using the script 368
Reference Table 368
Target 369
Overview 369
Using the GUI 369
Using the script 370
Reference Table 371
Report 372
Overview 372
Using the GUI 372
Using the script 373
Reference Table 373
For 374
Overview 374
Interfaces 374
Reference Table 375
Objects and Command Interactions 376
For Loop Behavior 376
Maneuver 376
Overview 376
Using the GUI 376
Using the script 377
Reference Table 377
Equation 378
Overview 378
Using the GUI 378
Using the script 379
Reference Table 379
Toggle 379
Overview 380
Using the GUI 380
Using the script 380
Reference Table 381
Vary 381
Overview 381
Using the GUI 381
Using the script 382
Reference Table 382
Begin Finite Burn 384
Overview 384
Using the GUI 384
Using the script 385
Reference Table 385
Achieve 385
Overview 386
Using the GUI 386
Using the script 387
Reference Table 387
Stop 388
Overview 388
Using the GUI 388
Using the script 389
ScriptEvent 389

Overview ... 389
Using the GUI ... 389
Using the script ... 390
Reference Table ... 391
Else ... 391
Overview ... 391
Using the Else Command in the GUI ... 391
Using the Else Command in the script ... 391
NonLinearConstraint ... 392
Overview ... 392
Using the GUI ... 392
Using the Script ... 393
Reference Table ... 394
Minimize ... 394
Overview ... 394
Using the GUI ... 395
Using the Script ... 395
Reference Table ... 395
Optimize ... 396
Overview ... 396
Using the GUI ... 396
Using the Script ... 397
Reference Table ... 397
Call Function ... 397
Overview ... 397
Using the Call Function Command in the GUI ... 398
Using the Call Function Command in the script ... 399
Reference Table ... 399
26. Miscellaneous ... 401
Math Operators in Scripting ... 401
Objective and Overview ... 401
Addition ... 401
Subtraction ... 401
Division ... 401
Multiplication ... 401
Exponentiation ... 401
Trigonometry ... 401
Square Root ... 402
norm ... 402
Logarithm ... 402
Matrix Manipulation ... 402
Radians and Degrees ... 402
Script Editor Shortcut Keys ... 402
Reference: Script Editor Shortcut Keys ... 402
Process Resources ... 404
Overview ... 404
General Process References ... 404
Configuration Management ... 404
Reviews ... 404
GMAT Commands in MATLAB ... 404
Overview ... 404
Commands ... 405
GMAT Console Instructions ... 406

Chapter 24. Resources

Spacecraft

Spacecraft Overview

This was created using the July 2, 2008 build of GMAT.

Overview

All of the tabs within the spacecraft object are listed below as links. To achieve a better overall understanding of the spacecraft's functions and properties, continue through the links to read through general definitions and explanations of the GUI dialog boxes, script syntax, and HTML quick reference tables.

Links and General Information

Orbit Tab

Parameters included:

- Epoch Information
- Coordinate System
- State Type
- State Type Elements

Orbit Tab Link

Attitude Tab

Parameters included:

- Attitude Model
- Coordinate System
- Euler Angle Sequence
- Attitude Initial Conditions
- Attitude Rate Initial Conditions

Attitude Tab Link

Ballistic/Mass Tab

Parameters included:

- Dry Mass
- Coefficient of Drag
- Coefficient of Reflectivity
- Drag Area
- Solar Radiation Pressure (SRP) Area

Ballistic/Mass Tab Link

Sensors

(To be created)

Tanks

(To be created)

Actuators

(To be created)

Orbit Tab

Overview

The Spacecraft object's Orbit tab controls the state of the spacecraft at the given epoch. The state can be set in reference to any defined coordinate system and in several representations (Cartesian, Keplerian, etc.), with automatic conversions if either is changed.

Interfaces

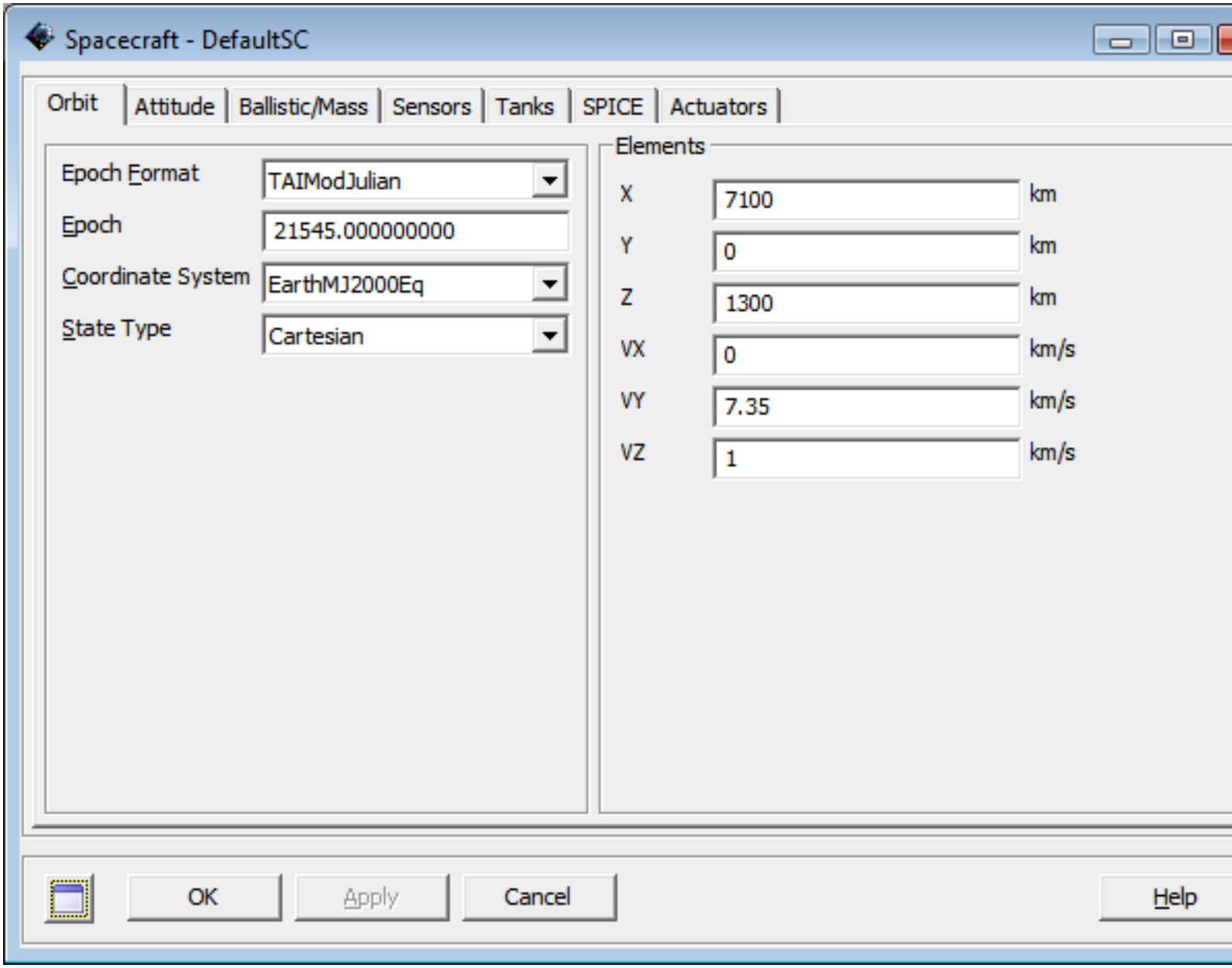


Figure: Default name and settings for the Spacecraft Orbit panel

```
Create Spacecraft DefaultSC;
GMAT DefaultSC.DateFormat = UTCGregorian;
GMAT DefaultSC.Epoch = '01 Jan 2000 11:59:28.000';
GMAT DefaultSC.CoordinateSystem = EarthMJ2000Eq;
GMAT DefaultSC.DisplayStateType = Keplerian;
GMAT DefaultSC.SMA = 7191.938817629013;
GMAT DefaultSC.ECC = 0.02454974900598138;
GMAT DefaultSC.INC = 12.85008005658097;
GMAT DefaultSC.RAAN = 306.6148021947984;
GMAT DefaultSC.AOP = 314.190551535992;
GMAT DefaultSC.TA = 99.88774933204884;
```

Field Descriptions

State Type	Default: Cartesian. Options: { Cartesian, Keplerian, ModifiedKeplerian, SphericalAZFPA, SphericalRADEC, Equinoctial }. The State Type field allows the user to configure the type of state vector that they wish to use. The State Type field has a dependency upon the Coordinate System field. If the coordinate system chosen by the user does not have a gravitational body at the origin, then the state types Keplerian, ModifiedKeplerian, and Equinoctial are not permitted. This is because these state types require a μ value. Units: N/A.
Coordinate System	Default: EarthMJ2000Eq. Options: { EarthMJ2000Eq, EarthMJ2000Ec, EarthFixed, or any user defined system } The Coordinate System field allows the user to choose which coordinate system with which to define the orbit state vector. The Coordinate System field has a dependency upon the State Type field. If the coordinate system chosen by the user does not have a gravitational body at the origin, then the state types Keplerian, ModifiedKeplerian, and Equinoctial are not permitted. This is because these state types require a μ value. Units: N/A.
Epoch Format	Default: TAIModJulian. Options: { A1ModJulian, TAIModJulian, UTCModJulian, TTModJulian, A1Gregorian, TAIGregorian, UTCGregorian, TTGregorian } The Epoch Format field allows the user to specify the format for defining a spacecraft's initial epoch. Epoch Format determines both the time system (TAI, TT, etc) and the time format (Modified Julian or Gregorian). Units: N/A.
Epoch	Default: 21545.000000000. Options: { See Comments } The Epoch field allows the user to specify the initial spacecraft epoch. The format of the epoch must be consistent with the Epoch Format field. If Epoch

	Format is of the "ModJulian" type, then the epoch is in Modified Julian format. If Epoch Format is a "Gregorian" type, the epoch is format like 01 Jan 2000 12:00:00.000. The epoch must be 04 Oct 1957 12:00:00.000 or later in all time systems (6116.0 ModJulian). Units: Modified Julian - days, Gregorian - N/A. The Modified Julian date offset is 2430000.0, which is different from the IAU's standard offset.

Cartesian State Type

X	Default: 7100. Options: { Real Number } X is the x-component of the Spacecraft state in the coordinate system chosen in the Spacecraft Coordinate System field. Units: km.
Y	Default: 0. Options: { Real Number } Y is the y-component of the Spacecraft state in the coordinate system chosen in the Spacecraft Coordinate System field. Units: km.
Z	Default: 1300. Options: { Real Number } Z is the z-component of the Spacecraft state in the coordinate system chosen in the Spacecraft Coordinate System field. Units: km.
VX	Default: 0. Options: { Real Number } VX is the x-component of the Spacecraft velocity in the coordinate system chosen in the Spacecraft Coordinate System field. Units: km/sec.
VY	Default: 7.35. Options: { Real Number }

	VY is the y-component of the Spacecraft velocity in the coordinate system chosen in the Spacecraft Coordinate System field. Units: km/sec.
VZ	Default: 1.0. Options: { Real Number } VZ is the z-component of the Spacecraft velocity in the coordinate system chosen in the Spacecraft Coordinate System field. Units: km/sec.

NOTE: Default values for the remaining state types are obtained through transformations of the default Cartesian spacecraft state values. The Keplerian, ModifiedKeplerian, and Equinoctial are dependant on the origin of the CoordinateSystem, because the state types require a μ value.

Keplerian State Type

SMA	Default: 7191.938817629013. Options: { abs(SMA) > 1 meter } The SMA field is the spacecraft orbit's osculating Keplerian semimajor axis in the coordinate system chosen in the Spacecraft Coordinate System field. SMA must be strictly greater than 1 m or less than -1 m to avoid numerical issues in the conversions to other state types. For circular and elliptical orbits (0 <= ECC < 0.9999999) SMA should only be greater than 1 m and for hyperbolic orbits (ECC > 1.0000001) SMA should be less than -1 m. GMAT does not support the creation of parabolic orbits. Units: km.
ECC	Default: 0.02454974900598137. Options: { 0.0 <= ECC <= 0.9999999, or ECC >= 1.0000001} The ECC field is the spacecraft orbit's osculating eccentricity. For circular or elliptic orbits, ECC must be greater than or equal to 0.0, and less than or equal to 0.9999999 to avoid numerical issues in the conversion to other state types as the Keplerian elements are undefined for parabolic orbits. For hyperbolic orbits ECC must be greater than or equal to 1.0000001. See also the SMA description. Units: Dimensionless.
INC	Default: 12.85008005658097. Options: { Real Number }

	The INC field is the spacecraft orbit's osculating inclination, in degrees, with respect to the selected coordinate system. Units: degrees.
RAAN	Default: 306.6148021947984. Options: { Real Number } The RAAN field is the spacecraft orbit's osculating right ascension of the ascending node, in degrees, with respect to the selected coordinate system. Units: degrees.
AOP	Default: 314.1905515359921. Options: { Real Number } The AOP field is the spacecraft orbit's osculating argument of periapsis, in degrees, with respect to the selected coordinate system. Units: degrees.
TA	Default: 99.8877493320488. Options: { Real Number } The TA field is the spacecraft orbit's osculating true anomaly. Units: degrees.

Modified Keplerian State Type

RadPer	Default: 7015.378524789846. Options: { Real Number > 0 } The RadPer field is the spacecraft orbit's osculating radius of periapsis. RadPer must be greater than zero. Units: km.
RadApo	Default: 7368.49911046818. Options: { Real Number ~= 0 } The RadApo field is the spacecraft orbit's osculating radius of apoapsis. RadApo must be strictly greater than or less than zero. When RadApo is negative, the orbit is hyperbolic. Units: km.
INC	See the Keplerian state section for a description on this field.

AOP	See the Keplerian state section for a description on this field.
RAAN	See the Keplerian state section for a description on this field.
TA	See the Keplerian state section for a description on this field.

Spherical AZFPA State Type

RMAG	Default: 7218.032973047435. Options: { Real Number > 0 } The RMAG field allows the user to set the magnitude of the spacecraft's position vector. Units: km.
RA	Default: 0. Options: { Real Number } The RA field allows the user to set the spacecraft's right ascension. Units: degrees.
DEC	Default: 10.37584492005105. Options: { Real Number } The DEC field allows the user to set the spacecraft's declination. Units: degrees.
VMAG	Default: 7.417715281675348. Options: { Real Number >= 0 } The VMAG field allows the user to set the magnitude of the spacecraft's velocity. Units: km/sec.
AZI	Default: 82.37742168155043. Options: { Real Number } The AZI field allows the user to set the spacecraft's azimuth angle. Units: degrees.
FPA	Default: 88.60870365370448. Options: { Real Number }

	The FPA allows the user to set a spacecraft's flight path angle. Units: degrees.

Spherical RADEC State Type

RMAG	See the SphericalAZFPA state section for a description on this field.
RA	See the SphericalAZFPA state section for a description on this field.
DEC	See the SphericalAZFPA state section for a description on this field.
VMAG	See the SphericalAZFPA state section for a description on this field.
RAV	Default: 90. Options: { Real Number } The RAV field allows the user to set the right ascension of the spacecraft's velocity. Units: degrees.
DECV	Default: 7.747772036108118. Options: { Real Number } The DECV field allows the user to set the declination of the spacecraft's velocity. Units: degrees.

Equinoctial State Type

SMA	See the Keplerian state section for a description on this field.
h	Default: -0.02423431419337062. Options: { Real Number } The h field is the projection of the eccentricity vector onto the y_{ep} axes. The F_{ep} system is a system used in calculating the equinoctial elements and is beyond the scope of this discussion. The GMAT Mathematical Specifications document discusses F_{ep} and the calculation of the equinoctial elements in detail. Units: None.
k	Default: -0.003922778585859663. Options: { Real Number }

	The k field is the projection of the eccentricity vector onto the x_{ep} axes. The F_{ep} system is a system used in calculating the equinoctial elements and is beyond the scope of this discussion. The GMAT Mathematical Specifications document discusses F_{ep} and the calculation of the equinoctial elements in detail. Units: None.
p	Default: -0.09038834725719359. Options: { Real Number } The p field is the projection of the N vector onto the y_{ep} axes. The N vector and the F_{ep} system are used in calculating the equinoctial elements and are beyond the scope of this discussion. The GMAT Mathematical Specifications document discusses N and F_{ep} and the calculation of the equinoctial elements in detail. Units: None.
q	Default: 0.06716454898232072. Options: { Real Number } The q field is the projection of the N vector onto the x_{ep} axes. The N vector and the F_{ep} system are used in calculating the equinoctial elements and are beyond the scope of this discussion. The GMAT Mathematical Specifications document discusses N and F_{ep} and the calculation of the equinoctial elements in detail. Units: None.
MLONG	Default: 357.9131803707104. Options: { Real Number } The MLONG field is the the spacecraft's mean longitude. The GMAT Mathematical Specifications document discusses mean longitude and the calculation of the equinoctial elements in detail. Units: degrees.

Using the Spacecraft Object

Behavior when Changing the Coordinate System

When you select a Coordinate System for a spacecraft, you specify the origin and axes set with respect to which the state is defined. When you change the coordinate system via the GUI, the GUI performs the conversion from the initial coordinate system to the new system. Similarly, when you specify a state component via the script, that state component is set in the coordinate system defined in the

CoordinateSystem field. For example, the following lines would result in the X-component of the Cartesian state of MySat to be 1000, in the EarthFixed system.

MySat.CoordinateSystem = EarthFixed;

MySat.X = 1000;

Internally, GMAT stores the state representation of a spacecraft with respect to a reference celestial body and expressed with respect to the MJ2000 Earth Equator. When the script lines above are executed in a script, GMAT converts the state to the specified coordinate system, in this case EarthFixed, sets the X component to 1000, and then converts the state back to the internal representation.

Behavior when Changing the State Type/Representation

When you select a state type (also called state representation) for a spacecraft you select the state representation to input state data from available types such as Cartesian, Keplerian, and Equinoctial to name a few. Internally, GMAT uses the Cartesian state representation of a spacecraft with respect to a reference celestial body and expressed with respect to the MJ2000 Earth Equator. When you change the state type via the GUI, the GUI performs the conversion from the initial type to the new type. When you specify a state component via the script, that state component is set in the coordinate system defined in the CoordinateSystem field. For example, the following lines would result in the SMA of MySat to be set to 7000, in the EarthFixed system.

MySat.CoordinateSystem = EarthFixed;

MySat.INC = 25;

When the script lines above are executed in a script, GMAT converts the state to the specified coordinate system and then to the specified state type -- in this case EarthFixed and Keplerian respectively -- sets the INC component to 25, and then converts the state back to the internal representation.

Behavior when Changing the Epoch

When you select an Epoch Format for a spacecraft, you define the time system (TAI, UTC, etc.) and whether the epoch is expressed in the Gregorian or Modified Julian format. When you specify the Epoch, you define the initial epoch of the spacecraft in the specified coordinate system and state representation.

If your choice for a spacecraft's coordinate system is a time varying system such as the EarthFixed system, then when you change the spacecraft's epoch, you also change the spacecraft's inertial state representation. For example, the following lines would result in the cartesian state of MySat to be set to [7000 0 1300 0 7.35 1] in the EarthFixed system at 01 Dec 2000 12:00:00.000 UTC.

Create Spacecraft MySat;

MySat.DateFormat = UTCGregorian;

MySat.Epoch = '01 Dec 2000 12:00:00.000';

MySat.CoordinateSystem = EarthFixed;

MySat.X = 7000;

MySat.Y = 0;

MySat.Z = 1300;

MySat.VX = 0;

MySat.VY = 7.35;

MySat.VZ = 1;

The corresponding EarthMJ2000Eq representation is

X = -2320.302657017663

Y = -6604.250753206916

Z = 1300.025987664349

You can change the epoch of a spacecraft in the mission sequence using a script line like this:

MySat.Epoch = '02 Dec 2000 12:00:00.000';

When the above line is executed in the mission sequence, GMAT converts the state to the specified coordinate system and then to the specified state type — in this case EarthFixed and Cartesian respectively — sets the epoch to the value of 02 Dec 2000 12:00:00.000, and then converts the state back to the internal representation. Again, it is important to realize the because the coordinate system in this case is time varying, changing the spacecraft epoch has resulted in a change to the spacecraft's inertial state representation. After the epoch is changed to 02 Dec 2000 12:00:00.000, the EarthMJ2000Eq state representation is now

X = -2237.878170605275

Y = -6738.089240761469

Z = 1300.021029469492

Object and Command Interactions

The Spacecraft object Orbit tab interacts with the following resources:

- CoordinateSystem

Attitude Tab

This was created using the June 18, 2008 build of GMAT.

Overview

In this section, you will write a brief summary of what the object or command does and how it works in basic terms.

Listing of Objects and Commands that interact

- Spacecraft
  - In order to access the attitude controls a spacecraft must exist.
- Coordinate System
  - The Attitude Model supplies the option of using any system either created or preexisting in the project to use when defining a spacecraft's attitude during the mission.

Using the GUI

This Section will describe the layout of the GUI dialog box, mostly at the group box level, and give a brief overview of what can be changed here.

- Group Box
  - Attitude Model - Drop Down Menu
    - Either a fixed or spinning coordinate system is chosen here to tell GMAT how to calculate spacecraft attitude properties.
  - Coordinate System - Drop Down Menu
    - Any previously existing system or any created in the project are available to choose from. The system selected will depend on the mission information desired and is based on allowed error and other calculations.
  - Euler Angle Sequence - Drop Down Menu
    - All combinations of Euler Angles are available here. The sequence chosen is dependent upon the type of information needed.
- Attitude Initial Conditions - Group Box
- Attitude Rate Initial Conditions - Group Box

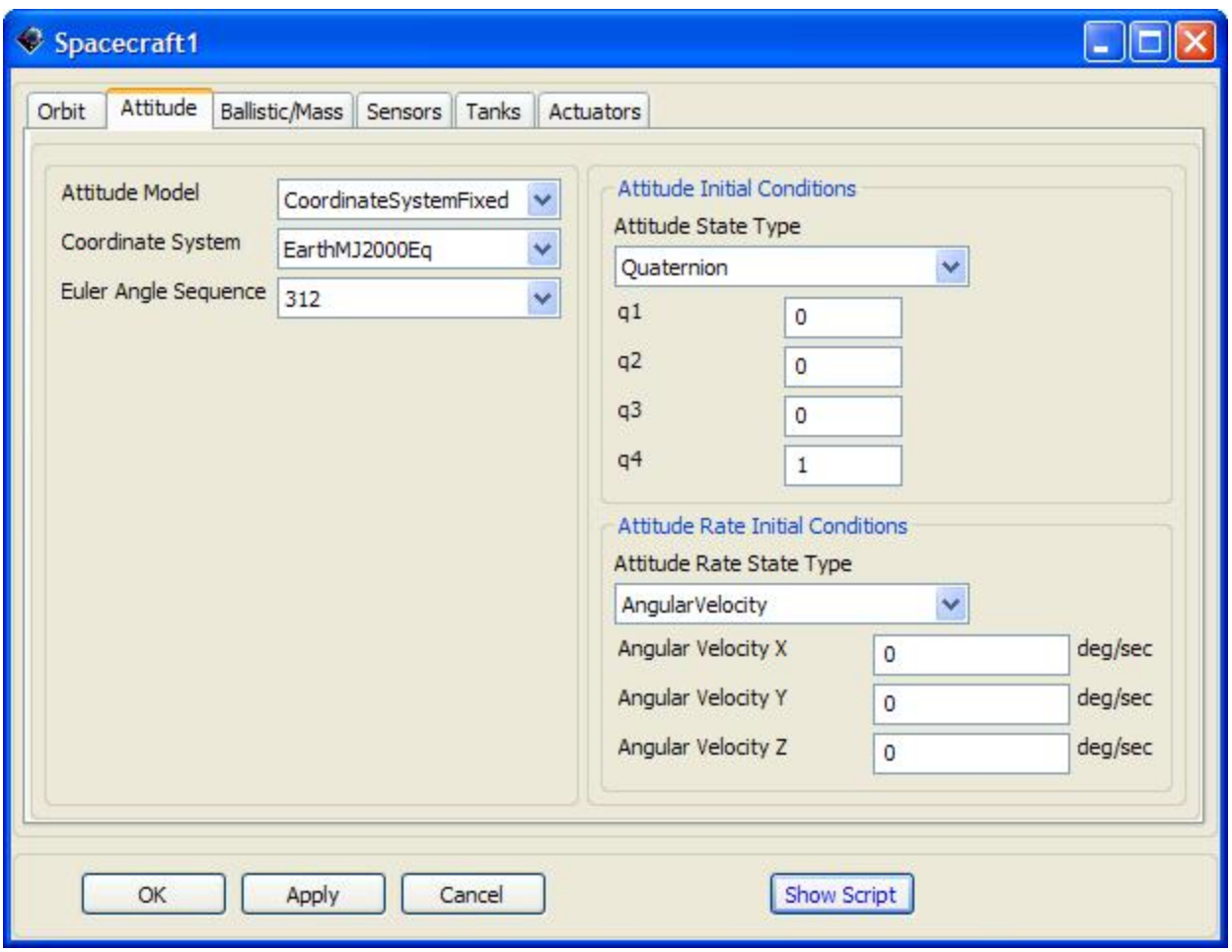


Figure: Default Name and Settings for the Spacecraft Attitude Dialog Box

Using the script

Script Syntax

```
Create Spacecraft Spacecraft1;
GMAT Spacecraft1.DateFormat = TAIModJulian;
GMAT Spacecraft1.Epoch = '21545.000000000';
GMAT Spacecraft1.CoordinateSystem = EarthMJ2000Eq;
GMAT Spacecraft1.DisplayStateType = Cartesian;
GMAT Spacecraft1.X = 7100;
GMAT Spacecraft1.Y = 0;
GMAT Spacecraft1.Z = 1300;
GMAT Spacecraft1.VX = 0;
GMAT Spacecraft1.VY = 7.35;
GMAT Spacecraft1.VZ = 1;
GMAT Spacecraft1.DryMass = 850;
GMAT Spacecraft1.Cd = 2.2;
GMAT Spacecraft1.Cr = 1.8;
```

Reference Table

Attitude Mode	Default: CSFixed. Options: { CSFixed, Spinner } The AttitudeMode mode field allows the user to specify the attitude dynamics model to be used by GMAT to propagate a spacecraft's attitude. The attitude dynamics model uses the initial attitude state and the algorithm associated with AttitudeMode to advance the attitude state in time. Units: N/A.
Attitude Coordinate System	Default: EarthMJ2000Eq. Options: { EarthMJ2000Eq, EarthMJ2000Ec, EarthMJ2000Eq, or any user defined system }A spacecraft's initial body axes orientation as defined by the quaternions or some other parameterizations are expressed with respect to the AttitudeCoordinateSystem. Unlike an orbit state, an attitude state is really information that uniquely defines a rotation matrix. A spacecraft's attitude is the orientation of the spacecraft's body-fixed frame with respect to the inertial frame. However, it is often more convenient to define the initial attitude with respect to an intermediate frame than with respect to an inertial frame. The Attitude CoordinateSystem allows the user to define the initial orientation of a spacecraft's body axes, with respect to any frame GMAT knows how to calculate. Units: N/A.

Attitude StateType	Default: EulerAngles. Options: { EulerAngles, Quaternions, DCM } The AttitudeStateType field allows the user to choose among different attitude parameterizations when defining the attitude initial conditions. Units: N/A.
Attitude Rate StateType	Default: EulerAngleRates. Options: { EulerAngleRates, AngularVelocity } The AttitudeRateStateType field allows the user to define the attitude parameterization to be used in defining the initial attitude rate. Units: N/A.
Euler Angle Sequence	Default: 312. Options: { 123, 132, 121, 131, 213, 231, 212, 232, 312, 321, 313, 323 } The EulerAngleSequence field allows the user to define the Euler sequence used in rotating from the body-fixed to the inertial axes. For example, if EulerAngleSequence is selected as 321, then the first rotation is a 3 rotation through EulerAngle1, the second rotation is a 2 rotation through EulerAngle2, and the third rotation is a 1 rotation through EulerAngle3. Units: N/A.

EulerAngle1	Default: 0. Options: { Real Number } EulerAngle1 is one of three Euler angles that can be used to define the initial conditions of a spacecraft. EulerAngle1 corresponds to the first rotation performed in the sequence that goes from the spacecraft body frame to the inertial frame. For example, if the EulerAngleSequence field is set to 321, the first rotation from the body to the inertial frame would be a 3-rotation throughEulerAngle1. Units: degrees.
EulerAngle2	Default: 0. Options: { Real Number } EulerAngle2 is one of three Euler angles that can be used to define the initial conditions of a spacecraft. EulerAngle2 corresponds to the second

	rotation performed in the sequence that goes from the spacecraft body frame to the inertial frame. For example, if the EulerAngleSequence field is set to 321, the second rotation from the body to the inertial frame would be a 2-rotation through EulerAngle2. Units: degrees.
EulerAngle3	Default: 0. Options: { Real Number } EulerAngle3 is one of three Euler angles that can be used to define the initial conditions of a spacecraft. EulerAngle3 corresponds to the third rotation performed in the sequence that goes from the spacecraft body frame to the inertial frame. For example, if the EulerAngleSequence field is set to 321, the third rotation from the body to the inertial frame would be a 1-rotation through EulerAngle3. Units: degrees.
q1	Default: 0. Options: { Real Number } The q1 parameter is the first element of the quaternion. GMAT normalizes the quaternion to be of length 1. Units: degrees.
q2	Default: 0. Options: { Real Number } The q2 parameter is the second element of the quaternion.GMAT normalizes the quaternion to be of length 1. Units: degrees.
q3	Default: 0. Options: { Real Number } The q3 parameter is the third element of the quaternion. GMAT normalizes the quaternion to be of length 1. Units: degrees.
q4	Default: 1. Options: { Real Number }

	The q4 parameter is the fourth element of the quaternion. GMAT normalizes the quaternion to be of length 1. Units: degrees.
DCM11	Default: 1. Options: { Real Number } The DCM11 parameter is the upper left component of the direction cosine matrix that rotates from the spacecraft body frame to the inertial frame. GMAT normalizes the attitude matrix to have a determinant of 1. The default DCM matrix is the identity matrix. Units: None.
DCM12	Default: 0. Options: { Real Number } The DCM12 parameter is the R_{12} component of the direction cosine matrix that rotates from the spacecraft body frame to the inertial frame. GMAT normalizes the attitude matrix to have a determinant of 1. The default DCM matrix is the identity matrix. Units: None.
DCM33	Default: 1. Options: { Real Number } The DCM33 parameter is the R_{33} component of the direction cosine matrix that rotates from the spacecraft body frame to the inertial frame. GMAT normalizes the attitude matrix to have a determinant of 1. The default DCM matrix is the identity matrix. Units: None.
EulerAngle Rate1	Default: 0. Options: { Real Number } The EulerAngleRate1 defines the time-rate-of-change of EulerAngle1, expressed in the the system defined by AttitudeCoordinateSystem. Units: deg/sec.
EulerAngle Rate2	Default: 0. Options: { Real Number } The EulerAngleRate2 defines the time-rate-of-change of EulerAngle2, expressed in the the system

	defined by AttitudeCoordinateSystem. Units: deg/sec.
EulerAngle Rate3	Default: 0. Options: { Real Number } The EulerAngleRate3 defines the time-rate-of-change of EulerAngle3, expressed in the the system defined by AttitudeCoordinateSystem. Units: deg/sec.
Angular VelocityX	Default: 0. Options: { Real Number } The AngularVelocityX component is the x-component of the spacecraft's body axes with respect to the system defined by AttitudeCoordinateSystem. Units: deg/sec.
Angular VelocityY	Default: 0. Options: { Real Number } The AngularVelocityY component is the y-component of the spacecraft's body axes with respect to the system defined by AttitudeCoordinateSystem. Units: deg/sec.
Angular VelocityZ	Default: 0. Options: { Real Number } The AngularVelocityZ component is the z-component of the spacecraft's body axes with respect to the system defined by AttitudeCoordinateSystem. Units: deg/sec.

Ballistic Mass Tab

This was created using the July 2, 2008 build of GMAT.

Overview

The ballistic and mass properties of a spacecraft include the drymass, coefficient of reflectivity and and coefficient of drag to name a few. These properties are grouped under the Ballistic/Mass tab on the spacecraft dialog box.

Interfaces

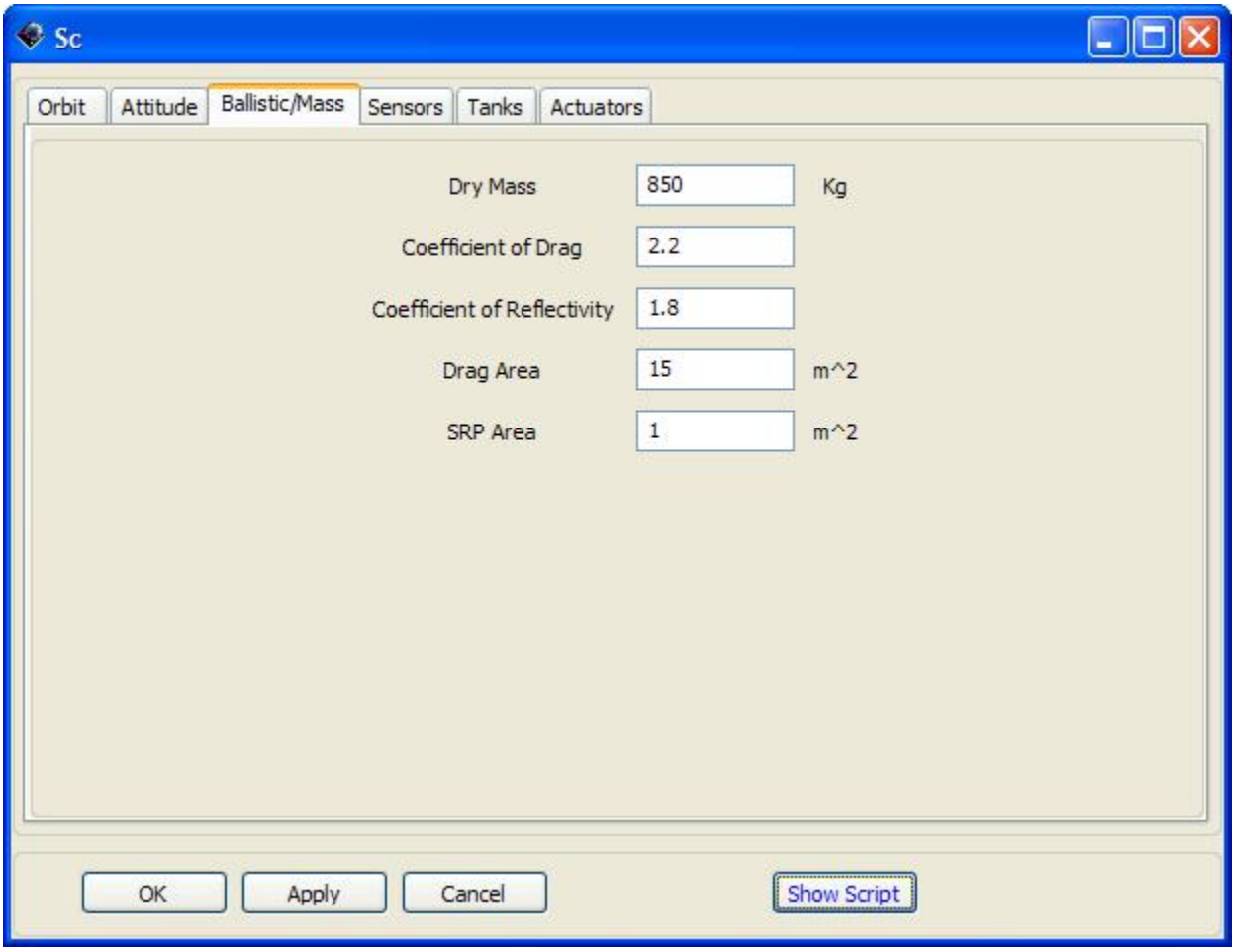


Figure: Default Name and Settings for the Spacecraft Ballistic Mass Dialog Box

```
GMAT Sc.DryMass = 850;
GMAT Sc.Cd = 2.2;
GMAT Sc.Cr = 1.8;
GMAT Sc.DragArea = 15;
GMAT Sc.SRPArea = 1;
```

Reference Table

Cd	Default: 2.2. Options: { Real Number ~062~ 0 } Cd is the spacecraft's drag coefficient assuming a spherical spacecraft. Units: Dimensionless.
Cr	Default: 2.2. Options: { 0 ~060~= Real Number ~060~= 2.0 }

	Cr is the spacecraft's coefficient of reflectivity. Units: Dimensionless.
DragArea	Default: 15.0. Options: { Real Number ~062~ 0 } The DragArea is the area of the spacecraft that is used in calculating atmospheric drag. Units: m~094~2
SRPArea	Default: 1.0. Options: { Real Number ~062~ 0 } The SRPArea is the area of the spacecraft that is used in calculating the force due to solar radiation pressure. SRPArea is the average surface area as seen from the Sun. Units: m~094~2
DryMass	Default: 850.0. Options: { Real Number ~062~ 0 } The DryMass is the mass of the spacecraft without the mass of tanks and fuel. Units: kg

Objects and Command Interactions

- Spacecraft
  - The properties in this section of the spacecraft object involve what the spacecraft will experience while on the mission, thus a spacecraft object is essential.

Propagator

This was created using the June 9, 2008 build of GMAT.

Overview

In GMAT, a Propagator is a combination of an integrator and a force model. Hence, a Propagator contains a physical model of the space environment that is used to model the motion of a spacecraft as it moves forwards or backwards in time (VOP formulation is not currently supported). You configure a Propagator by selecting among different numerical integrators and environment models to create a Propagator appropriate to the flight regime of your spacecraft during its mission. GMAT supports numerous numerical integrators as well as Force Models like point mass and non-spherical gravity, atmospheric drag (Earth), and solar radiation pressure.

To propagate spacecraft in GMAT, you first create and configure a Propagator object in the script or in the Resource Tree. Then, in the mission sequence, you create a Propagate command, the topic of another

section, and select among previously existing Propagators and Spacecraft. Hence, a Propagator is different from a Propagate command: A Propagator is a resource and is found in the GUI under the resource tree, and a Propagate Event is configured under the Mission Tree and is how you instruct GMAT to propagate spacecraft.

Interfaces

The Propagator dialog box is illustrated below and contains two group boxes: the Integrator group and the Force Model group. This section discusses the items in each group on the Propagate Panel. It will present how to configure a propagator and discuss all possible user settable fields in detail.

Integrator Group

The Integrator group allows you to select and configure a numerical integrator appropriate to your application. You select the type of numerical integrator in the -+Type+- pull-down menu. After selecting the integrator type, the fields below the -+Type+- pull-down menu dynamically configure to allow you to set relevant parameters for the selected integrator type. All integrators except for Adams-Bashforth-Moulton (ABM) are configured using the same fields. The ABM integrator has the following additional fields: -+MinIntegrationerror+- and -+NomIntegrationerror+-.

Force Model Group

The Force Model group allows you to configure a force model appropriate to the flight regime of your application. The central body of propagation and error control method are also defined here. On a Propagator, GMAT classifies all celestial bodies into two mutually exclusive categories: Primary Bodies, and Point Masses. Primary bodies can have a complex force model that includes non-spherical gravity, drag, and magnetic field. Point mass bodies only have a point-mass gravitational force.

You can add a Primary Body by clicking the Select button in the Primary Bodies group box. Once you have added a Primary Body (or mulitiple bodies in future versions), the pull down menu allows you to configure the force model for each Primary Body. The text box, next to the Select button contains a list of all Primary Bodies so you can see which bodies are being treated with complex force models. In future versions, GMAT will support multiple primary bodies on a propagator allowing you to use a non-spherical gravity model for the Earth and Moon simultaneously.

Configuring certain fields in the Force Model group affects the availability of other fields. For example, if you remove all bodies from the Primary Bodies list, the Gravity Field, Atmosphere Model, and Magnetic Field groups are disabled. Similarly, in the Gravity Field group, the search button and the Model File field are only active if "Other" is selected in the -+Type+- pull-down. In the Atmosphere Model group, the Setup button is only active when -+MSISE90+- or -+JacchiaRoberts+- are selected in the -+Type+- pull-down.

GMAT allows you to define Solar flux properties if you select either the -+MSISE90+- or -+JacchiaRoberts +- atmosphere models. By selecting one of these models in the -+Type+- pull-down menu in the Atmosphere Model group, the Setup button is enabled. Clicking on the Setup button brings up the panel illustrated below. Here you can input Solar flux values. GMAT does not currently support flux files though future versions will.

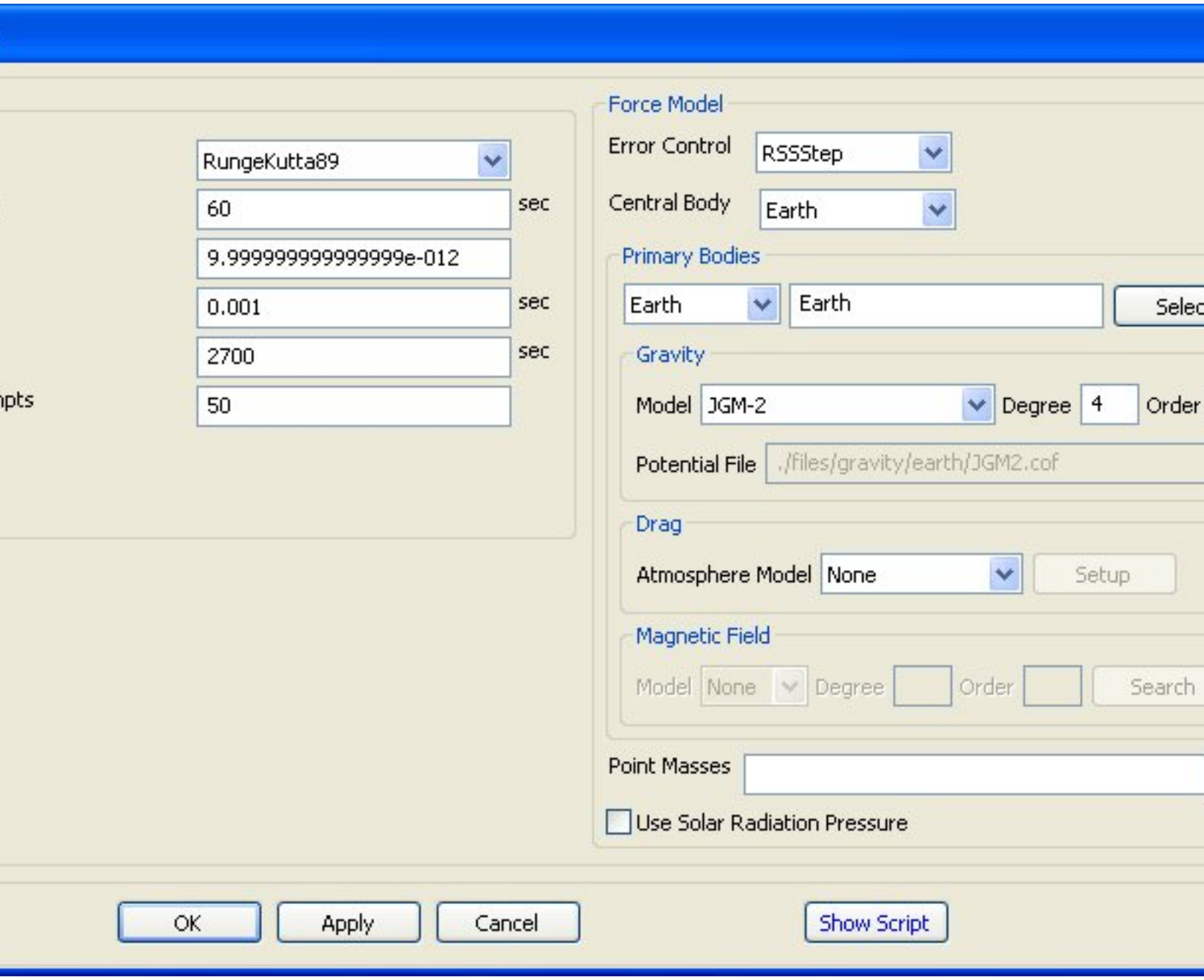


Figure: Default Name and Settings for the Propagator Object Dialog Box

```
Create ForceModel DefaultProp_ForceModel;
GMAT DefaultProp_ForceModel.CentralBody = Earth;
GMAT DefaultProp_ForceModel.PrimaryBodies = {Earth};
GMAT DefaultProp_ForceModel.Drag = None;
GMAT DefaultProp_ForceModel.SRP = Off;
GMAT DefaultProp_ForceModel.ErrorControl = RSSStep;
GMAT DefaultProp_ForceModel.GravityField.Earth.Degree = 4;
GMAT DefaultProp_ForceModel.GravityField.Earth.Order = 4;
GMAT DefaultProp_ForceModel.GravityField.Earth.PotentialFile = 'JGM2.cof';
```

```
Create Propagator DefaultProp;
GMAT DefaultProp.FM = DefaultProp_ForceModel;
GMAT DefaultProp.Type = RungeKutta89;
GMAT DefaultProp.InitialStepSize = 60;
GMAT DefaultProp.Accuracy = 9.999999999999999e-012;
GMAT DefaultProp.MinStep = 0.001;
GMAT DefaultProp.MaxStep = 2700;
GMAT DefaultProp.MaxStepAttempts = 50;
```

Reference Table

Force Model Group Box Fields

ErrorControl	Default: RSSStep Options: { RSSStep, RSSState, LargestState, LargestStep } The ErrorControl field allows you to choose how a Propagator measures the error in an integration step. The algorithm selected in the ErrorControl field is used to determine the error in the current step, and this error is compared to the value set in the Accuracy field to determine if the step has an acceptable error or needs to be improved. All error measurements are relative error, however, the reference for the relative error changes depending upon the selection of ErrorControl. RSSStep is the Root Sum Square (RSS) relative error measured with respect to the current step. RSSState is the (RSS) relative error measured with respect to the current state. LargestStep is the state vector component with the largest relative error measured with respect to the current step. LargestState is the state vector component with the largest relative error measured with respect to the current state. For a more detailed discussion see the GMAT Mathematical Specification. Units: N/A.
CentralBody	Default: Earth Options: { Sun, Mercury, Venus, Earth, Luna, Mars, Jupiter, Saturn, Uranus, Neptune, Pluto } The CentralBody field allows the user to select the origin for the propagation. All propagation occurs in the FK5 axes system, about the CentralBody chosen by the user. The CentralBody must be a gravitational body and so cannot be a LibrationPoint or other special point. Units: N/A.
PrimaryBodies	Default: Earth Options: { Sun, Mercury, Venus, Earth, Luna, Mars, Jupiter, Saturn, Uranus, Neptune, Pluto }

	The PrimaryBodies field is a list of all celestial bodies that are to be modelled with a force model more complex than point mass gravity. Lists are surrounded by curly braces. For each PrimaryBody, the user can choose a drag, magnetic field, and aspherical gravity model. There is a coupling between the PrimaryBodies filed and the PointMasses field. A primary body can be any planet or moon not included in the PointMasses field. Units: N/A.
Gravity.PrimaryBody.PotentialFile	Default: JGM2 Options: {CentralBody-based models, Other. See Comments } This field allows the user to define the source for the non-spherical gravity coefficients for a primary body. If a gravity file is located in the Primary Body's potential path as defined in the startup file, you only need to specify the model name and not the entire path. For example, if the JGM2 coefficients file is contained in the directory defined in the startup file by the line EARTH_POT_PATH, then you only need to specify the model name JGM2. If the model is not contained in the body's potential path, you must supply the entire path as well as the file name. If GMAT does not successfully find the file requested, it uses the default gravity model as defined in the startup file. From the GUI, only models for Earth appear if Earth is the active primary body. This is to avoid allowing the user to select a lunar potential model for the Earth. If the Other option is selected the user has the ability of selecting a gravity model file on their local computer. Units: None.
Gravity.PrimaryBody.Degree	Default: 4 Options: { Integer >= 0 and < the maximum specified by the model, Order <= Degree} This field allows the user to select the the degree, or number of zonal terms, in the non-spherical gravity model. Ex. Gravity.Earth.Degree = 2 tells GMAT to use only the J2 zonal term for the Earth. The value for Degree must be less than the maximum degree specified by the Model. Units: None.
Gravity.PrimaryBody.Order	Default: 4 Options: { Integer >= 0 and < the maximum specified by the model, Order <= Degree} This field allows the user to select the the order, or number of tesseral terms, in the non-spherical

	gravity model. Ex. Gravity.Earth.Order = 2 tells GMAT to use 2 tesseral terms. Note: Order must be greater than or equal to Degree. Units: None.
Drag	Default: None. Options: { None, JacchiaRoberts, MSISE90, Exponential } The Drag field allows a user to specify a drag model. Currently, only one drag model can be chosen for a particular propagator and only Earth models are available. Units: N/A. Note: This field will be deprecated in future versions of GMAT. Currently, the Drag field and the Drag.AtmosphereModel field must be set to the same value.
Drag.AtmosphereModel	Default: None. Options: { None, JacchiaRoberts, MSISE90, Exponential } The Drag.AtmosphereModel field allows a user to specify a drag model. Currently, only one drag model can be chosen for a particular propagator and only Earth models are available. Units: N/A.
Drag.F107	Default: 150 Options: { Real Number >= 0 } The Drag.F107 field allows you to set the $F_{10.7}$ solar flux value used in computing atmospheric density. $F_{10.7}$ is the solar radiation at a wavelength of 10.7 cm. Units: $W/m^2/Hz$
Drag.F107A	Default: 150 Options: { Real Number >= 0 } The Drag.F107A field allows you to set the average $F_{10.7}$ value. $F_{10.7}$ is the average of $F_{10.7}$ over one month. Units: $W/m^2/Hz$
Drag.MagneticIndex	Default: 3 Options: { 0 <= Real Number <= 9 } The Drag.MagneticIndex index field allows you to set the k_p value for use in atmospheric density calculations. k_p is a planetary 3-hour-average, geomagnetic index that measures magnetic effects of solar radiation. Units: None.
PointMasses	Default: None Options: { Sun, Mercury, Venus, Earth, Luna, Mars, Jupiter, Saturn, Uranus, Neptune, Pluto }

	A PointMass is a planet or moon that is modelled by a point source located at its center of gravity. A PointMass body can be any planet or moon not included in the PrimaryBodies field. Units: N/A.
SRP	Default: Off Options: { On, Off } The SRP field allows the user to include the force due to solar radiation pressure in the total sum of forces. Units: N/A.

Integrator Group Box Fields

Type	Default: RungeKutta89 Options: { RungeKutta89, RungeKutta68, RungeKutta56, PrinceDormand45, PrinceDormand78, BulirschStoer, AdamsBashforthMoulton } The Type field is used to set the type of numerical integrator.
InitialStepSize	Default: 60 (sec) Options: { Real Number } The InitialStepSize is the size of the first attempted step by the integrator. If the step defined by InitialStepSize does not satisfy Accuracy, the integrator adapts the step according an algorithm defined in the mathematical specifications document to find an acceptable first step that meets the user's requested. Units: seconds
Accuracy	Default: 1e-11 Options: { Real Number >= 0 } The Accuracy field is used to set the desired accuracy for an integration step. When you set a value for Accuracy, GMAT uses the method selected in ErrorControl field on the Force Model, to determine a metric of the accuracy. For each step, the integrator ensures that the accuracy, as calculated using the method defined by ErrorControl, is less than the limit defined by Accuracy. If an integrator exceeds MaxStepAttempts trying to meet the requested accuracy, and error message is thrown and propagation stops.

	Units: N/A
MinStep	Default: .001 (sec) Options: { Real Number > 0, MinStep <= MaxStep } The MinStep field is used to set the minimum allowable step size. Units: seconds
MaxStep	Default: 2700.0 (sec.) Options: { Real Number > 0, MinStep <= MaxStep } The MaxStep field is used to set the maximum allowable step size. Units: seconds
MaxStepAttempts	Default: 50 Options: { Integer > 0 } The MaxStepAttempts field allows the user to set the number of attempts the integrator takes to meet the tolerance defined by Accuracy. Units: None

Fields associated only with Adams-Bashforth-Moulton Integrator

MinIntegrationerror	Default: 1.0e-13 Options: { Real Number > 0, MinIntegrationerror < NomIntegrationerror < Accuracy } The MinIntegrationerror field is used by the ABM integrator (and other predictor-corrector integrators when implemented) as the desired integration error to be obtained when the step size is changed. Predictor-Corrector integrators adapt step size when the obtained integration error falls outside of the range of acceptable steps, as determined by the bounds set by the MinIntegrationerror and Accuracy fields. The integrator then applies an internal calculation to recompute the step size, attempting to hit the NomIntegrationerror, and restarts the integrator. Units: N/A
NomIntegrationerror	Default: 1.0e-11 Options: { Real Number > 0, MinIntegrationerror , NomIntegrationerror , Accuracy } The NomIntegrationerror field is used by the ABM integrator (and other predictor-corrector integrators

	when implemented) as the desired integration error to be obtained when the step size is changed. Predictor-Corrector integrators adapt step size when the obtained integration error falls outside of the range of acceptable steps, as determined by the bounds set by the MinIntegrationerror and Accuracy fields. The integrator then applies an internal calculation to recompute the step size, attempting to hit the NomIntegrationerror, and restarts the integrator. Units: N/A

Object and Command Interactions

A Propagator Requires Other Objects/Commands of Type: Force Model (Script Only). (Note: There are slight differences in how you configure a Propagator in the script and GUI and we refer you to the script example shown in the script section for details. Effort has been made to reduce any difference between the script and GUI.)

ReportFile

This was created using the $MONTH$ $DATE$, $YEAR$ build of GMAT.

Overview

The Report Object is a file where values and qualities of objects can be stored so that they can be viewed at a later time.

Interfaces

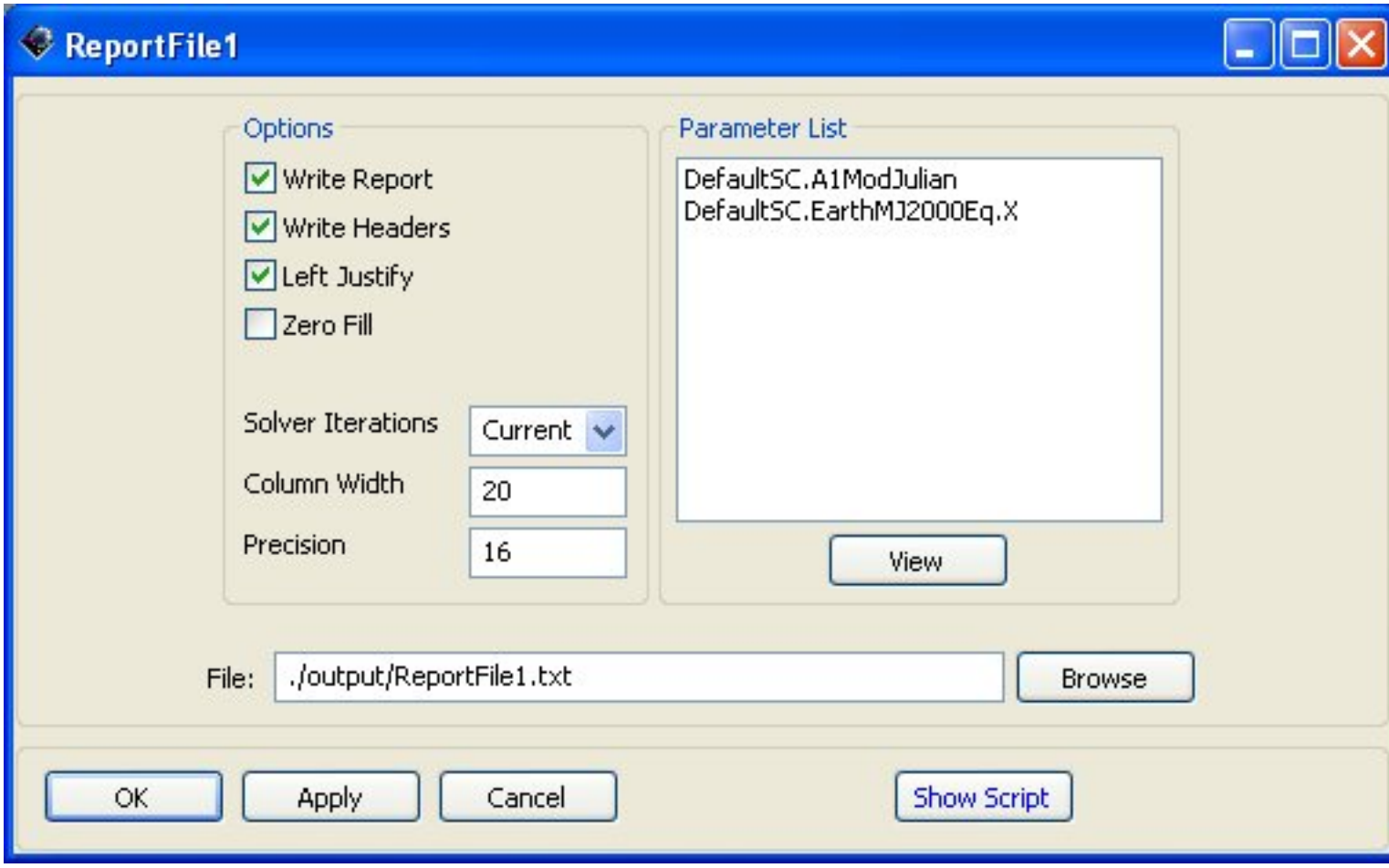


Figure: Default Name and Settings for the Report Object Dialog Box

```
Create ReportFile ReportFile1;
GMAT ReportFile1.SolverIterations = Current;
GMAT ReportFile1.Filename = 'ReportFile1.txt';
GMAT ReportFile1.Precision = 16;
GMAT ReportFile1.Add = {DefaultSC.A1ModJulian, DefaultSC.EarthMJ2000Eq.X};
GMAT ReportFile1.WriteHeaders = On;
GMAT ReportFile1.LeftJustify = On;
GMAT ReportFile1.ZeroFill = Off;
GMAT ReportFile1.ColumnWidth = 20;
```

Reference Table

FileName	Default: /RunReports/ReportFile1.txt. Options: [Valid File Path and Name] The FileName field allows the user to define the file path and file name for a report. Units: None.

Precision	Default: 16. Options: [Integer > 0] The Precision field allows the user to set the precision of thevariable written to a report. Units: Same as variable being reported.
Add	Default: N/A. Options: [Any user-defined parameter. Ex.Variables, Arrays, S/C parameters] The \st{Add} field allows a userto add user-defined variables to a report file. To add multipleuser-defined variables, enclose the variables with curly brackets.Ex. MyReportName.Add ={Sat.X, Sat.Y, Var1, Array(1,1)}; TheGUI's Selected field is the equivalent of the script's Add field.In the event of no Add command or no objects in the Selected field, GMATshould run without the Report output and a warning message displayed in the message window. Thefollowing warning message is sufficient: Report plot will be turned off. No object has beenselected for reporting. Units: N/A.
WriteReport	Default: On . Options: [On,Off] The WriteReport field specifies whether to write data tothe report FileName. Units: N/A.
WriteHeaders	Default: On . Options: [On,Off] The WriteHeaders field specifies whether to includeheaders that describe the variables in a report. Units: N/A.
LeftJustify	Default: On. Options: [On, Off] When the LeftJustify field is set to On, then the data isleft justified and appears at the left most side of the column. Ifthe LeftJustify field is set to Off, then the data iscentered in the column. Units: N/A.

ZeroFill	Default: On. Options: [On, Off] Units: N/A .
ColumnWidth	Default: 20. Options: [Integer > 0] The ColumnWidth field is used to define the width of the datacolumns in a report file. The value for ColumnWidth is appliedto all columns of data. For example, if ColumnWidth is set to20, then each data column will be 20 white-spaces wide. Units: Characters.
SolverIterations	Default: Off. Options: [On, Off] The SolverIterations field determines whether or not data associated with perturbed trajectories during a solver (Targeter, Optimize) sequence is written to a report file. When SolverIterations is set to On, solver iterations are written to the report file. When SolverIterations is Off, the solver iterations are not written to the report file. Units: N/A.

Objects and Commands Interactions

- Report Command
  - Located in the mission tree and will retrieve values at that particular time and insert them at the bottom of the report file.

Fuel Tank

This was created using the June 9,2008 build of GMAT.

Overview

The tank of the spacecraft is added under the resource tree, under spacecraft, and added to the folder Hardware. This object is a tank of fuel that would be used on a finite burn.

Listing of Objects and Commands that interact

- BeginFiniteBurn/EndFiniteBurn
  - These commands use the tank and the thruster to start a finite burn, where the delta V is not instantaneous.

- Thruster
  - This object uses the tank for power for a finite burn.
- FiniteBurn
  - This takes the parameters of the tank and the thruster and apply it to a coordinate system, with a scaling method available if wanted.
- Spacecraft
  - This is the object that the burn is applied to.

Using the GUI

- Fuel Mass
  - The total mass of fuel available in the fuel tank.
- Pressure
  - The pressure of the fuel in the tank.
- Temperature
  - The temperature of the fuel in the tank.
- Reference Temperature
- Volume
  - The original volume of the fuel in the tank.
- Fuel Density
  - The density of the fuel
- Pressure Regulated
  - If selected, it means that the pressure will be maintained a constant during the duration of a thrust. If not, it means that the pressure in the tank will decrease with time as more fuel is released by a thruster.

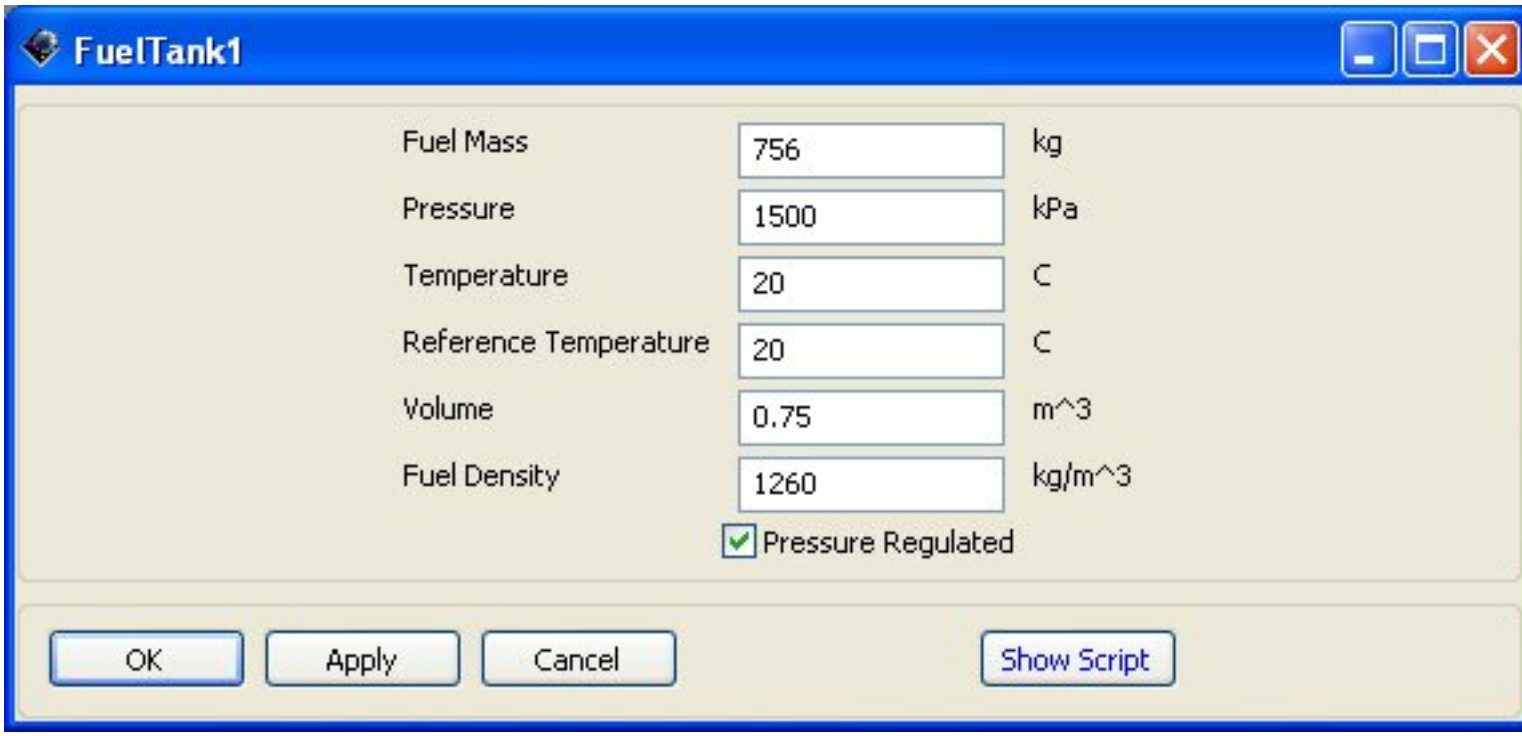


Figure: Default Name and Settings for the Fuel Tank Dialog Box

Using the script

Script Examples

```
Create FuelTank FuelTank1;
GMAT FuelTank1.FuelMass = 756;
GMAT FuelTank1.Pressure = 1500;
GMAT FuelTank1.Temperature = 20;
GMAT FuelTank1.RefTemperature = 20;
GMAT FuelTank1.Volume = 0.75;
GMAT FuelTank1.FuelDensity = 1260;
GMAT FuelTank1.PressureRegulated = true;
```

Reference Table

FuelMass	Default: 756. Options: [Real Number $\geq$ 0] The FuelMass field is the mass of fuel in the tank. Units: kg.
Pressure	Default: 1500. Options: [Real Number $\geq$ 0] The Pressure field isthe pressure of the fuel in the tank.

	Units: kPa.
Temperature	Default: 20. Options: [Real Number] The Temperature field is the temperature of the fuel in the tank. Units: C.
RefTemperature	Default: 20. Options: [Real Number] RefTemperature Units: C.
Volume	Default: 0.75. Options: [Real Number $\geq$ 0] The Volume field is the volume of the tank. Units: m^3.
FuelDensity	Default: 1260. Options: [Real Number $\geq$ 0] The FuelDensity parameter is the fuel density. Units: kg/m^3
PressureRegulated	Default: true. Options: [true, false] The PressureRegulated flag allows the userto choose between a pressure regulated tank or a blow down tank. If PressureRegulated is true, then thepressure is held constant as fuel mass is depleted. If PressureRegulated is false, then the pressuredecreases as fuel is depleted.

Thruster

This was created using the June 9, 2008 build of GMAT.

Overview

The Thruster uses the fuel tank and directs the thrust of the rocket engine while in space. It is used for finite burns.

Listing of Objects and Commands that interact

- BeginFiniteBurn/EndFiniteBurn

  - These commands use the tank and the thruster to start a finite burn, where the delta V is not instantaneous.

- Fuel Tank
  - This object contains the fuel used to power the thruster and subsequently the finite burn.

- FiniteBurn
  - This takes the parameters of the tank and the thruster and apply it to a coordinate system, with a scaling method available if wanted.

- Spacecraft
  - This is the object that the burn is applied to.

Using the GUI

- Coordinate System
  - This is the coordinate system used for determining the directions that the thruster will be fired in.

- X Direction, Y Direction, Z Direction
  - These enable the user to determine precisely what the vector is of the thrust in relation to the chosen coordinate system.

- Thrust Scale Factor
  - This is the factor by which the thrust will be multiplied by, thus strengthening or weakening the thrust.

- Tank
  - This chooses which tank is wanted to be accessed for fuel.

- Edit Thruster Coefficients
  - Here you can change the performance of the thruster. For more information see http://gmat.gsfc.nasa.gov/docs/GMATMathSpec.pdf on pg 87.

- Edit Impulse Coefficients
  - Here you can change the impulse of the thruster. For more information see http://gmat.gsfc.nasa.gov/docs/GMATMathSpec.pdf on pg 87.

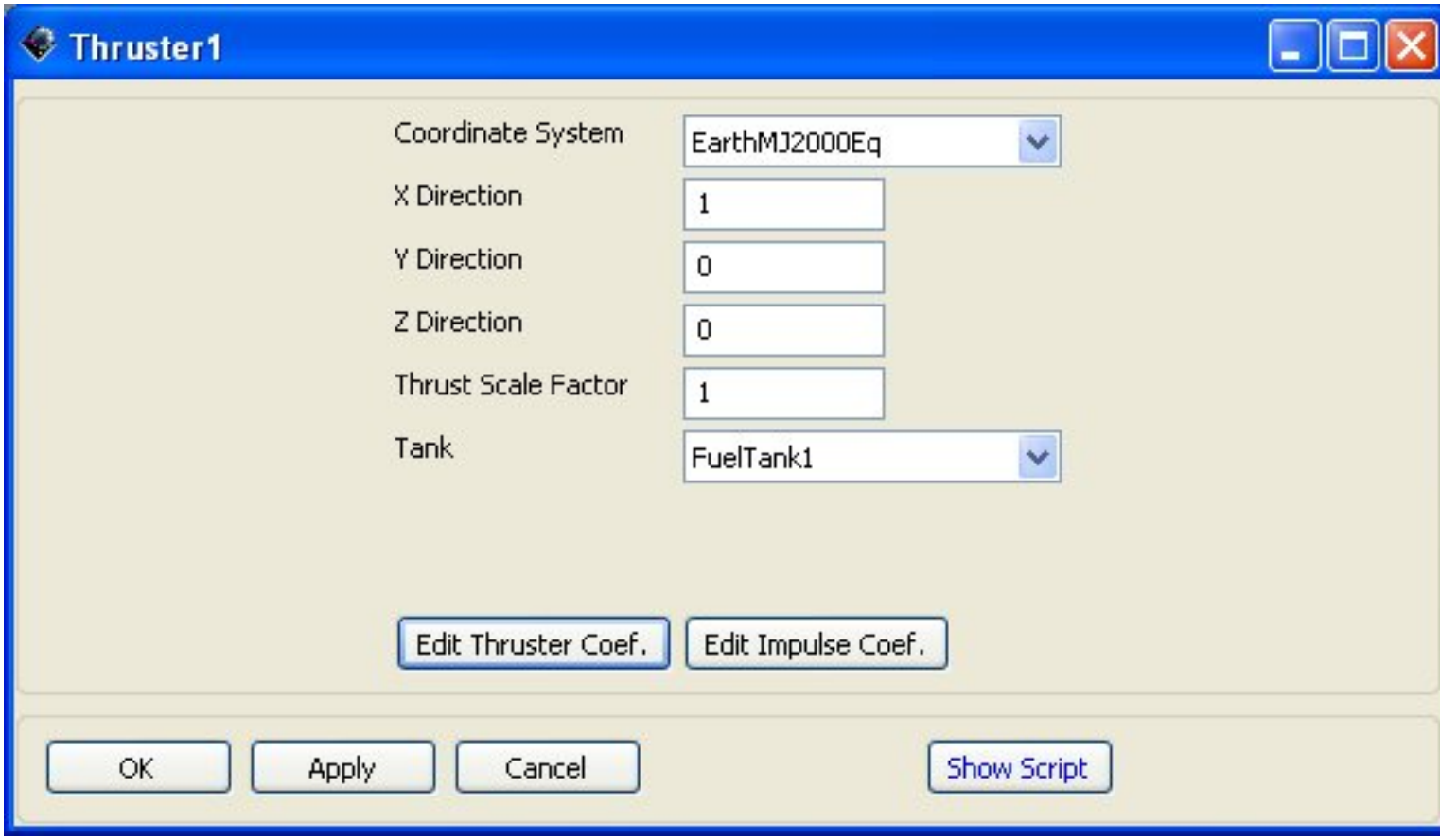


Figure: Default Name and Settings for the Thruster object Dialog Box

Using the script

Script Examples

```
Create Thruster Thruster1;
GMAT Thruster1.Element1 = 1;
GMAT Thruster1.Element2 = 0;
GMAT Thruster1.Element3 = 0;
GMAT Thruster1.C1 = 500;
GMAT Thruster1.C2 = 0;
GMAT Thruster1.C3 = 0;
GMAT Thruster1.C4 = 0;
GMAT Thruster1.C5 = 0;
GMAT Thruster1.C6 = 0;
GMAT Thruster1.C7 = 0;
GMAT Thruster1.C8 = 0;
GMAT Thruster1.C9 = 0;
GMAT Thruster1.C10 = 0;
GMAT Thruster1.C11 = 1;
GMAT Thruster1.C12 = 0;
GMAT Thruster1.C13 = 0;
GMAT Thruster1.C14 = 0;
GMAT Thruster1.K1 = 2150;
```

```
GMAT Thruster1.K2 = 0;
GMAT Thruster1.K3 = 0;
GMAT Thruster1.K4 = 0;
GMAT Thruster1.K5 = 0;
GMAT Thruster1.K6 = 0;
GMAT Thruster1.K7 = 0;
GMAT Thruster1.K8 = 0;
GMAT Thruster1.K9 = 0;
GMAT Thruster1.K10 = 0;
GMAT Thruster1.K11 = 1;
GMAT Thruster1.K12 = 0;
GMAT Thruster1.K13 = 0;
GMAT Thruster1.K14 = 0;
GMAT Thruster1.CoordinateSystem = 'MJ2000EarthEquator';
GMAT Thruster1.ThrustScaleFactor = 1;
```

Reference Table

CoordinateSystem	Default: EarthMJ2000Eq. Options: [EarthMJ2000Eq, EarthMJ2000Ec, EarthMJ2000Eq, or anyuser defined system] The CoordinateSystem field for a thrusterdetermines what coordinate system the orientation parametersX_Direction, Y_Direction, and Z_Direction arereferenced to. This is a temporary fix in GMAT. Eventually, theuser will specify the attitude of aspacecraft, and then X_Direction, Y_Direction, and Z_Direction will be referenced to the spacecraft body frame.
Axis	Default: VNB. Options: [InertialVNB] The Axis field allows the user to define a localcoordinate system for a thruster. Note that there is a couplingbetween the Axis parameter and the CoordinateSystemparameter for a thruster. Only one of the two canbe specified. Units: N/A.
Origin	Default: Earth. Options:[Sun, Mercury, Venus, Earth, Luna, Mars,Jupiter, Saturn, Uranus, Neptune, Pluto] The Origin fieldallows the user to define a local origin for a thruster. Note thatthere is a coupling between the Origin parameter and theCoordinateSystem parameter for a thruster. Only one of the twocan be specified. Units: N/A.

X_Direction	Default: 1. Options: [Real Number] X_Direction, divided by the RSS of the three directioncomponents, forms the x direction of the spacecraft thrust vector direction.
Y_Direction	Default: 0. Options: [Real Number] Y_Direction, divided by the RSS of the three directioncomponents, forms the y direction of the spacecraft thrust vector direction.
Z_Direction	Default: 0. Options: [Real Number] Z_Direction, divided by the RSS of the three directioncomponents, forms the z direction of the spacecraft thrust vector direction.
ThrustScaleFactor	Default: 1. Options: [Real Number>0] ThrustScaleFactor is a scale factor that is multiplied by the thrust vector for a given thruster, before the thrust vector is added into the total accleration. Units: None.
Tank	Default: None. Options: [Tank Name] The Tank field specifies which tank the thruster draws propellant from. The constants C_i below are used in the following equation to calculate thrust F_T as a function of pressure P and temperature T $F_T(P,T)$ = $\{C_1+C_2P+C_3P^2+C_4P^{C_5}+C_6P^{C_7}+C_8P^{C_9}+C_{10}C_{11}^{C_{12}P}\}(T/T_{ref})^{1+C_{13}+C_{14}P}$
C1	Default: 500. Options: [Real Number] Thrust coefficient. Units: N
C2	Default: 0. Options: [Real Number]

	Thrust coefficient. Units: N/kPa.
C3	Default: 0. Options: [Real Number] Thrust coefficient. Units: N/kPa^2
C4	Default: 0. Options: [Real Number] Thrust coefficient. Units: N/kPa^{C5}
C5	Default: 0. Options: [Real Number] Thrust coefficient. Units: None
C6	Default: 0. Options: [Real Number] Thrust coefficient. Units: N/kPa^{C7}
C7	Default: 0. Options: [Real Number] Thrust coefficient. Units: None
C8	Default: 0. Options: [Real Number] Thrust coefficient. Units: N/kPa^{C9}
C9	Default: 0. Options: [Real Number] Thrust coefficient. Units: None
C10	Default: 0.

	Options: [Real Number] Thrust coefficient. Units: N.
C11	Default: 1. Options: [Real Number] Thrust coefficient. Units: None
C12	Default: 0. Options: [Real Number] Thrust coefficient. Units: 1/kPa.
C13	Default: 0. Options: [Real Number] Thrust coefficient. Units: None.
C14	Default: 0. Options: [Real Number] Thrust coefficient. Units 1/kPa. The constants K_i below are used in the following equation tocalculate Isp as a function of pressure P and temperatureT $I_{sp}(P,T)$ = $\{K_1+K_2P+K_3P^2+K_4P^{K_5}+K_6P^{K_7}+K_8P^{K_9}+ K_{10}K_{11}{}^{K_{12}P}\} (T/T_{ref})^{1+K_{13}+K_{14}P}$
K1	Default: 2150. Options: [Real Number] Isp coefficient. Units: m/sec
K2	Default: 0. Options: [Real Number] Isp coefficient. Units: m/(sec?kPa).

K3	Default: 0. Options: [Real Number] Isp coefficient. Units: m/(sec?kPa^{2})
K4	Default: 0. Options:[Real Number] Isp coefficient. Units: m/(sec?kPa^{K5}).
K5	Default: 0. Options: [Real Number] Isp coefficient. Units: None
K6	Default: 0. Options: [Real Number] Isp coefficient. Units: m/(sec?kPa^{K7}).
K7	Default: 0. Options: [Real Number] Isp coefficient. Units: None
K8	Default: 0. Options: [Real Number] Isp coefficient. Units: m/(sec?kPa^{K9}.
K9	Default: 0. Options: [Real Number] Isp coefficient. Units: None
K10	Default: 0. Options: [Real Number] Isp coefficient.

	Units: m/sec.
K11	Default: 1. Options: [Real Number] Isp coefficient. Units: None
K12	Default: 0. Options: [Real Number] Isp coefficient. Units: 1/kPa.
K13	Default: 0. Options: [Real Number] Isp coefficient. Units: None.
K14	Default: 0. Options: [Real Number] Isp coefficient. Units 1/kPa.

SQP(fmincon)

This was created using the July 2, 2008 build of GMAT.

Object Overview

fmincon is an Nonlinear Programming solver provided in MATLAB's Optimization Toolbox. fmincon performs nonlinear constrained optimization and supports linear and nonlinear constraints. This optimizer is only available to users who have both MATLAB and MATLAB's Optimization toolbox.

GMAT contains an interface to the fmincon optimizer and it appear as if fmincon is a built in optimizer in GMAT. Field names for this object have been copied from those used in MATLABS optimset function for consistency with MATLAB as opposed to other solvers in GMAT.

Interface Overview

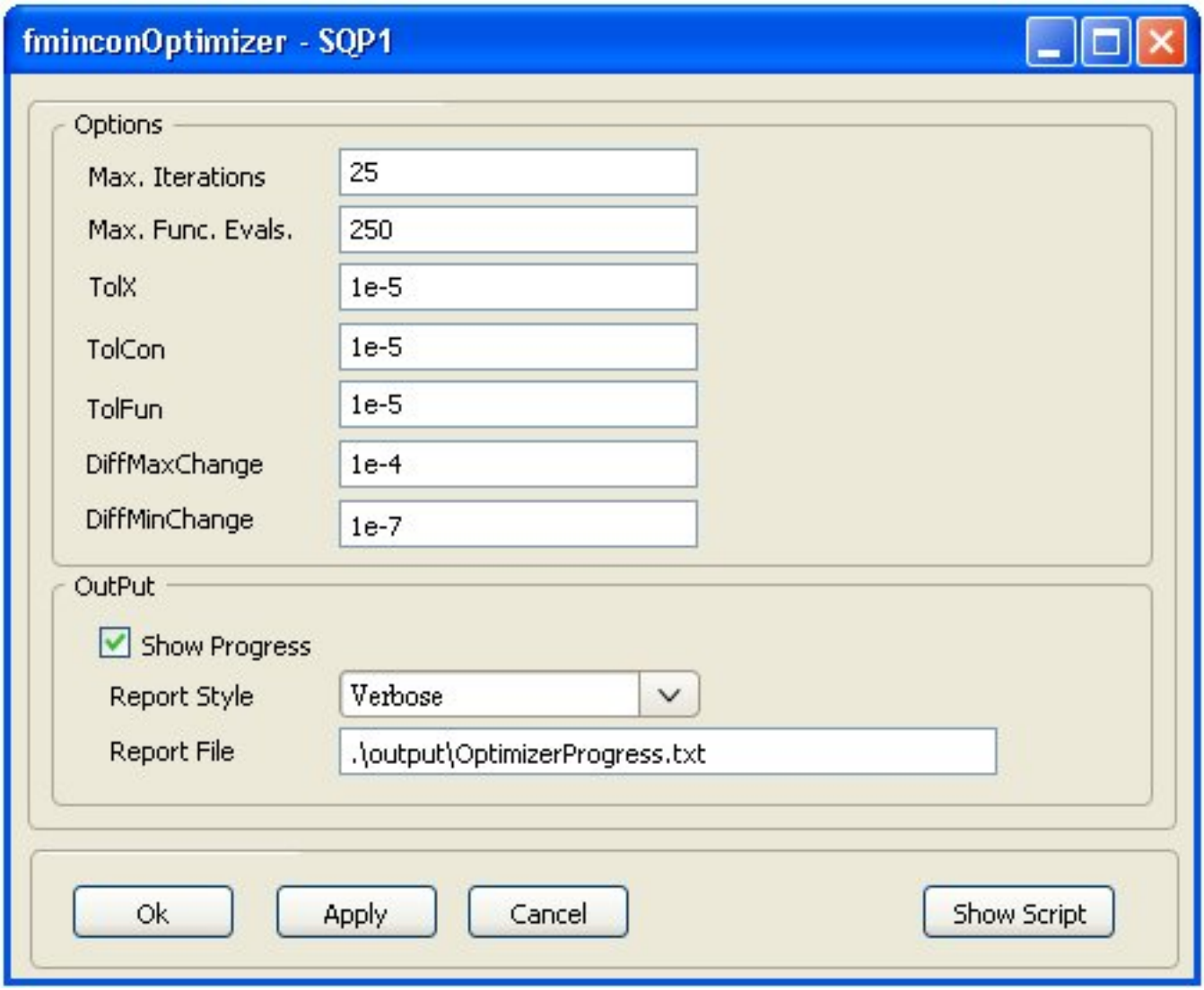


Figure: FminconOptimizer Dialog Box

```
Create FminconOptimizer SQP1;
GMAT SQP1.MaxIter       = 25;
GMAT SQP1.MaxFunEvals   = 250;
GMAT SQP1.TolX          = 1e-5;
GMAT SQP1.TolFun        = 1e-5;
GMAT SQP1.TolCon        = 1e-5;
GMAT SQP1.DiffMaxChange = 1e-4;
GMAT SQP1.DiffMinChange = 1e-7;
GMAT SQP1.ShowProgress  = true;
GMAT SQP1.ReportStyle   = 'Verbose';
GMAT SQP1.ReportFile    = '\output\OptimizerProgress.txt';
```

Field Descriptions

DiffMax	Default: 0.1 .

Change	Options: [Real Number > 0] The DiffMaxChange parameter is the upper limit on the perturbation used in MATLAB's finite differencing algorithm. For fmincon, you don't specify a single perturbation value, but rather give MATLAB a range, and it uses an adaptive algorithm that attempts to find the optimal perturbation. Units: N/A.
DiffMin Change	Default: 1e-8 . Options: [Real Number > 0] The DiffMinChange parameter is the lower limit on the perturbation used in MATLAB's finite differencing algorithm. For fmincon, you don't specify a single perturbation value, but rather give MATLAB a range, and it uses an adaptive algorithm that attempts to find the optimal perturbatin. Units: N/A.
MaxFunEvals	Default: 1000. Options: [Integer > 0] The MaxFunEvals parameter allows the user to set the maximum number of cost function evaluations in an attempt to find an optimal solution. This is equivalent to setting the maximum number of passes through an optimization loop in a GMAT script. If a solution is not found before the maximum function evaluations, fmincon outputs an ExitFlag of zero, and GMAT continues. Units: N/A.
MaxIter	Default: 400. Options: [Integer > 0] The MaxIter parameter allows the user to set the maximum allowable number of optimizer iterations. Depending upon the nature of the problem, and whether gradients are provided, it may take many function evaluations for each optimizer iteration. The MaxIter parameter allows the user to control the maximum function evaluations, and maximum iterations independently. Units: N/A.
TolX	Default: 1e-4. Options: [Real Number > 0]

	The TolX parameter is the termination toler-ance on the vector of independent variables, and is used only if the user sets a value. Units: N/A.
TolFun	Default: 1e-4. Options: [Real Number > 0] The TolFun parameter is the convergence tolerance on the cost function value. Units: N/A.
TolCon	Default: 1e-4 . Options: [Real Number > 0] The TolCon parameter is the convergence tolerance on the constraint functions. Units: N/A.
ShowProgress	Default: true . Options: [true , false] The ShowProgress field determines whether data pertaining to iterations of the solver is displayed in the message window. When ShowProgress is true, the amount of information contained in the message window is controlled by the ReportSytle field. Units: N/A.
ReportStyle	Default: Normal . Options: [Normal , Concise, Verbose, Debug] The ReportStyel field determines the amount and type of data written to the message window for each iteration of the solver (When ShowProgress is true). ADD DESCRIPTIONS OF CONCISE,VERBOSE, ADN NORMAL. I CAN'T RUN THE OPTIMIZER RIGHT NOW SO I CAN'T TELL WHAT EACH SETTING DOES. Units: N/A.
ReportFile	Default: Normal . Options: [.\output\OptimizerData.txt] The ReportFile field contains the path and file name of the report file. Units: N/A.

Using an fminconOptimizer

- Optimize Command
- Minimize
- Nonlinear Constraint

Solar System

This was created using the June 9,2008 build of GMAT.

Overview

This folder, found if the Solar System folder itself is double-clicked, enables the user to determine where he gets his data on the movements of planets, how often it updates, and how accurate the data is.

Listing of Objects and Commands that interact

- CelestialBodies, BaryCenter, and Libration Point
  - The position and data on all these depend on the source of the Solar System data and how often it is updated.
- Propagator
  - How often the position of the planetary bodies are updated will have an impact on how a spacecraft will propagate.
- Spacecraft
  - A number of parameters of a spacecraft are based off the position of the planets

Using the GUI

- EphemerisSource
  - The EphemerisSource field allows the user to select the source used for planetary ephemerides. The source is used globally whenever planetary ephemeris information is required.
- EphemerisUpdateInterval
  - The EphemerisUpdateInterval is used to set how often planetary positions are updated when calculating accelerations during propagation. For low-Earth orbits, EphemerisUpdateInterval can be set to around 60 for faster numerical integration with little effect on the accuracy of the propagation. For deep space propagation, EphemerisUpdateInterval should be set to zero.
- UseTTForEphemeris
  - GMAT uses time in the TDB system as the default time system in the JPL ephemeris files. However, often it is possible to use time in the TT time system, without significant difference in propagation accuracy. (TT and TDB are within 1 millisecond of each other). The advantage to using TT is that it avoids the transformation from TT to TDB and therefore orbit propagation will execute faster. The UseTTForEphemeris field allows the user to choose between the default of TDB in the ephemeris files (UseTTForEphemeris=false), or TT in the ephemeris files (UseTTForEphemeris = true).

- EphemerisFile
  - The EphemerisFile field allows the user to specify the location and name of the file for each type of ephemeris GMAT supports. For example, if Ephemeris is set to DE405, you can set the path for a DE405 file using SolarSystem.EphemerisFile = c:/MyPath/MyDE405.file.
- AnalyticModel

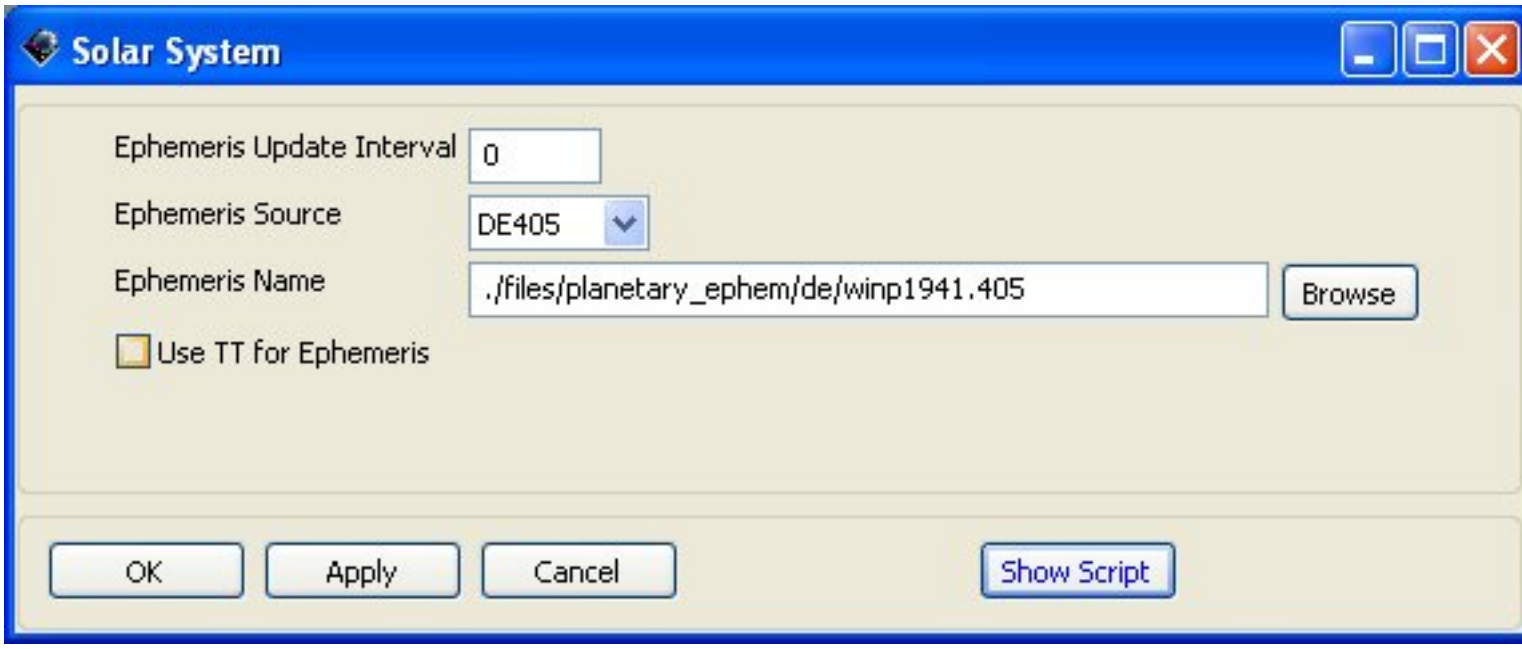


Figure: Default Name and Settings for the Solar System Object Dialog Box

Reference Table

EphemerisSource	Default: DE405. Options: [DE405,DE200, SLP, Analytic] The EphemerisSourcefield allows the user to select the source used for planetaryephemerides. The source is used globally whenever planetaryephemeris information is required. Units: None.
Ephemeris UpdateInterval	Default: 0. Options: [Real Number $\geq$ 0] The EphemerisUpdateInterval is used to set howoften planetary positions are updated when calculating accelerationsduring propagation. For low-Earth orbits,EphemerisUpdateInterval can be set to around 60 for fasternumerical integration with little effect on the accuracy of thepropagation. For

	deep space propagation,EphemerisUpdateInterval should be set to zero. Units: sec.
UseTTForEphemeris	Default: false. Options: [true, false] GMAT uses time in the TDB system as thedefault time system in the JPL ephemeris files. However, often itis possible to use time in the TT time system, without significantdifference in propagation accuracy. (TT and TDB are within 1millisecond of each other). The advantage to using TT is that itavoids the transformation from TT to TDB and therefore orbitpropagation will execute faster. The UseTTForEphemeris fieldallows the user to choose between the default of TDB in theephemeris files (UseTTForEphemeris=false), or TT in theephemeris files (UseTTForEphemeris = true). Units: N/A.
EphemerisFile	Default: Same as startup file. Options:[Filepath and file name consistent with operating system] The EphemerisFile field allows the user to specify the locationand name of the file for each type of ephemeris GMAT supports. Forexample, if Ephemeris is set to DE405, you can set thepath for a DE405 file using SolarSystem.EphemerisFile =c:/MyPath/MyDE405.file. Units: N/A.
AnalyticModel	Default: LowFidelity. Options: [LowFidelity]: Units: N/A.

DifferentialCorrector

This was created using the July 2, 2008 build of GMAT.

Overview

A differential corrector is a numerical solver for solving two-point boundary value problems. The DC in GMAT uses a simple shooting method where the derivatives are determined using finite differencing. In the mission sequence, you use the differential corrector object in a Target sequence to solve two-point value problems. For example, differential correctors are often used to determine the maneuver components required to achieve desired orbital conditions, say, B-plane conditions at a planetary flyby.

You must create and configure a differential corrector object according to your application by setting numerical properties of the solver such as tolerance and maximum iterations. You can also select among different output options that show increasing levels of information for each differential corrector iteration.

The allowable settings for a differential corrector are shown in the GUI screen shots and reference table below. You can learn more about how to use a DC in a targeting sequence by reading the help files for Target, Vary, and Achieve.

Interfaces

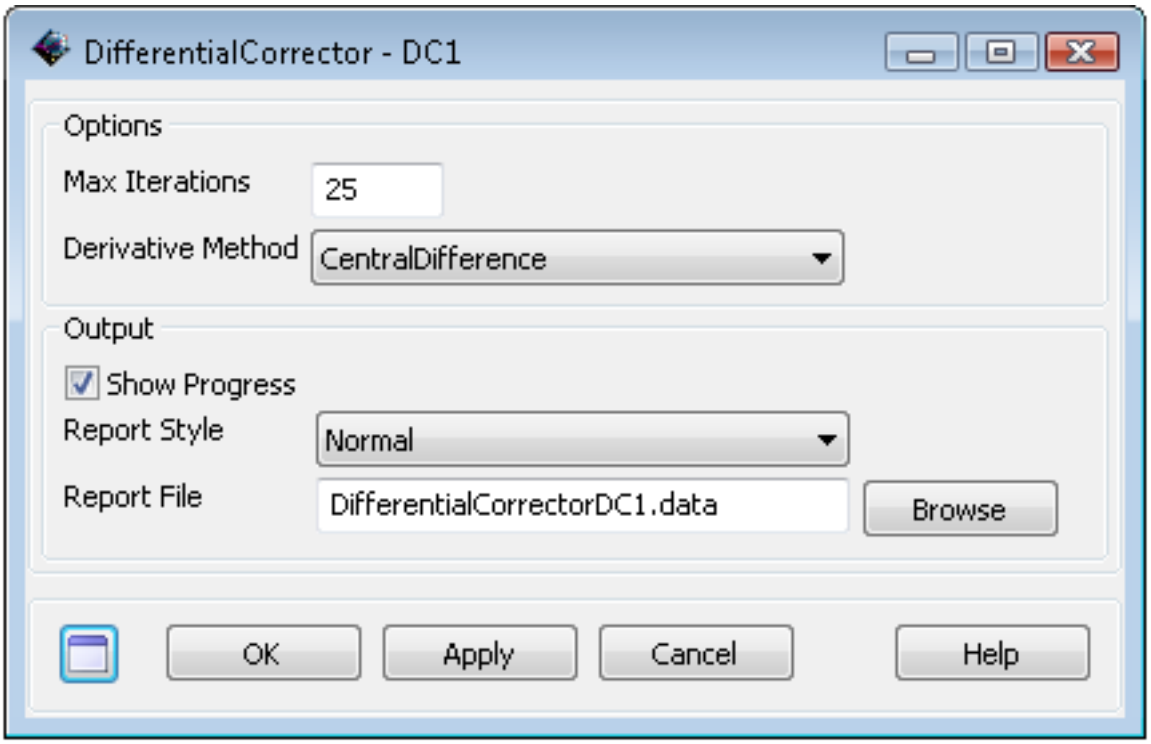


Figure: Default Name and Settings for the Differential Corrector Dialog Box

```
Create DifferentialCorrector DefaultDC;
GMAT DefaultDC.ShowProgress = true;
GMAT DefaultDC.ReportStyle = 'Normal';
GMAT DefaultDC.TargeterTextFile = 'DifferentialCorrectorDefaultDC.data';
GMAT DefaultDC.MaximumIterations = 25;
GMAT DefaultDC.UseCentralDifferences = false;
```

Reference Table

MaximumIterations	Default: 25 Options: Integer >= 1 The MaximumIterations field allows the user to set the maximum number of iterations the differential corrector is allowed during the attempt to find a solution. If the maximum iterations is reached, GMAT exits the target loop and continues to the next command in the mission sequence. In this case,

	the objects retain their states as of the last nominal pass through the targeting loop. Units: N/A.
DerivativeMethod	Default: ForwardDifference Options: [ForwardDifference, BackwardDifference, CentralDifference] The DerivativeMethod field allows the user to choose between one-sided and central differencing for numerically determining the Jacobian matrix. Units: N/A.
ShowProgress	Default: true. Options: [true, false] When the ShowProgress field is set to true, then data illustrating the progress of the differential correction process are written to the message window. The message window is updated with information on the current control variable values and the contraint variances for both on perturbation and iteration passes. When the ShowProgress field is set to false, no information on the progress of the differential correction process is displayed. Units: N/A.
ReportStyle	Default: Normal . Options: [Normal, Concise, Verbose, Debug]: The ReportStyle field allows the user to control the amount and type of information written to the file defined in the ReportFile field. Currently, the Normal and Concise options contain the same information: the Jacobian, the inverse of the Jacobian, the current values of the control variables, and achieved and desired values of the constraints. Verbose contains values of the perturbation variables in addition to the data for Normal and Concise. Debug contains detailed script snippets at each iteration for objects who have control variables. Units: N/A.
ReportFile	Default: DifferentialCorrectorDCName. Options: [Filename consistent with OS] The ReportFile field allows the user to specify the path and file name for the differential correction report.

	Units: N/A.

Object and Command Interactions

The Differential Corrector does not interact directly with any resource objects.

The Differential Corrector is used in the following mission sequence commands:

- Target
- Vary
- Achieve

Celestial Bodies

This was created using the July 9, 2008 build of GMAT.

Overview

The Celestial Bodies are the main bodies of the Solar System and the Moon. They are part of the resource tree and are under Solar System. They can be selected for a large variety of options for points of reference.

Listing of Objects and Commands that interact

- Coordinate System
  - Interacts with the celestial body in that the body can be selected as the origin of the coordinate system or as a primary or secondary point for defining the axes.
- OpenGL plot
  - Interacts with a celestial body in a number of ways. It can be selected to be drawn onto the plot, with the color of the orbit an option. A celestial body can also be selected as the View Point Reference or the View Direction.
- Propagator
  - Can interact with a celestial body in a number of ways. Under the Force Model heading, any celestial body can be chosen for use as the central body, as the primary bodies, and the secondary bodies.
- Bary Center
  - Can interact with celestial bodies in that a bary center is calculated using some number of celestial bodies which can be selected.
- Libration point
  - Can interact with celestial bodies in that a libration point is calculated using two celestial bodies which can be selected.
- Spacecraft
  - Interacts with celestial bodies in that for a number of parameters describe how the spacecraft is positioned in relation to a celestial body. This can include the periapsis, apoapsis, and altitude, as well as a great many of other parameters.

Using the GUI

The first three parameters describe how the body is shaped. These parameters include the mu constant, the equatorial radius, and the flattening factor. The last 7 parameters describe the orbit of the body at that particular moment in time about some central body.

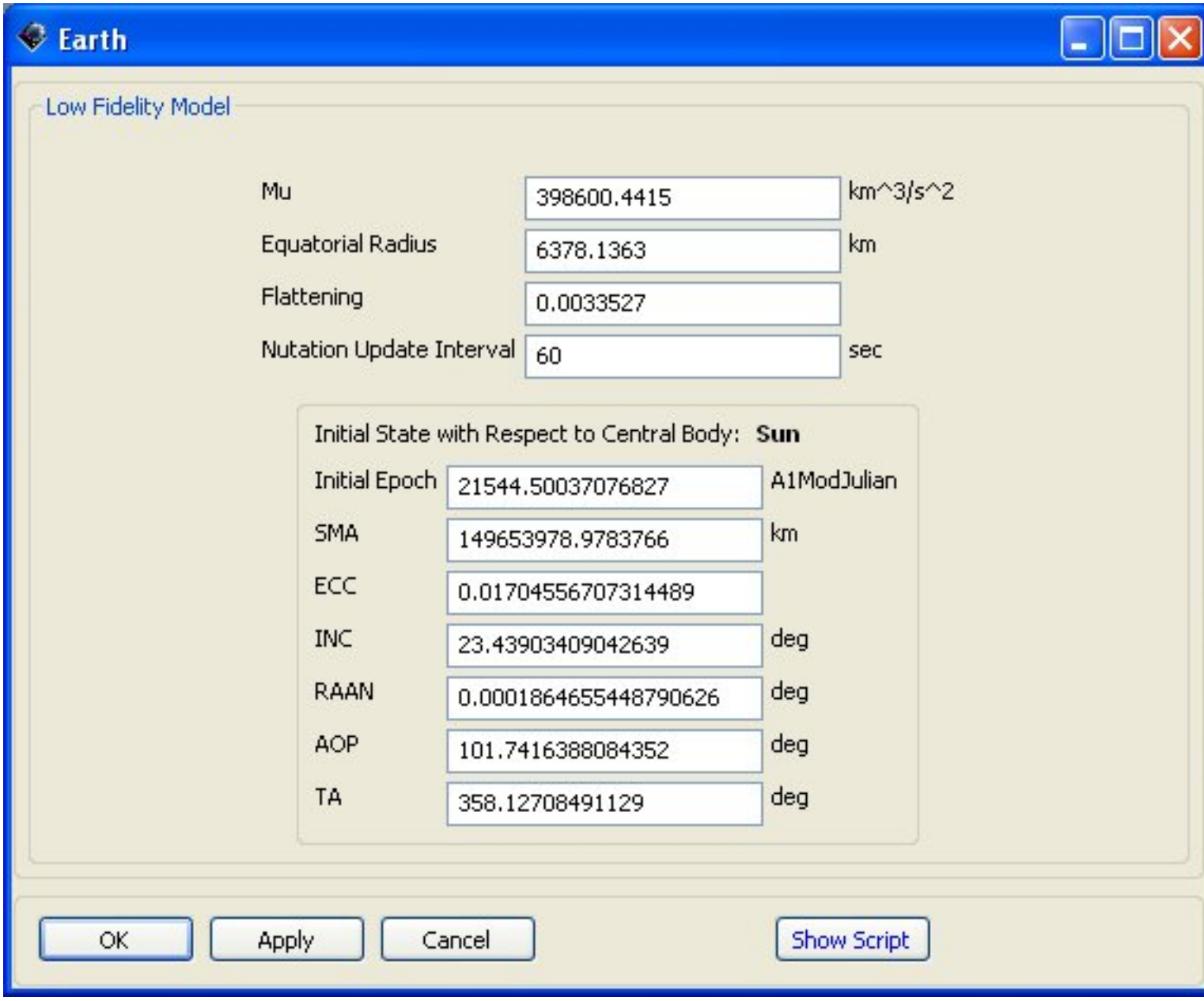


Figure: Celestial Body Object Dialog Box

Using the script

Script Syntax

```
Create Planet Earth;
GMAT Earth.BodyType = 'Planet';
GMAT Earth.Mass = 5.973331957140716e+024;
GMAT Earth.EquatorialRadius = 6378.1363;
GMAT Earth.Flattening = 0.0033527;
GMAT Earth.PolarRadius = 6356.75232242699;
GMAT Earth.Mu = 398600.4415;
GMAT Earth.PosVelSource = 'DE_405';
GMAT Earth.State = [[ 0 0 0 0 0 0 ];
GMAT Earth.StateTime = 0;
GMAT Earth.CentralBody = 'Sun';
GMAT Earth.BodyNumber = 2;
GMAT Earth.RefBodyNumber = 3;
GMAT Earth.SourceFilename = './files/planetary_ephem/de/winp1941.405';
GMAT Earth.SourceFile = ./files/planetary_ephem/de/winp1941.405;
GMAT Earth.UsePotentialFileFlag = false;
GMAT Earth.PotentialFileName = './files/gravity/earth/JGM2.cof';
GMAT Earth.AngularVelocity = [[ 0 0 7.2921158553e-005 ];
GMAT Earth.HourAngle = 0;
GMAT Earth.AtmosphereModelName = '';
GMAT Earth.Order = 4;
GMAT Earth.Degree = 4;
GMAT Earth.RotationDataSource = 'NotApplicable';
GMAT Earth.NutationUpdateInterval = 60;
```

Reference Table

Fields Associated with All Celestial Bodies (Using Default Values for Earth as an Example)

Mu	Default: 398600.4414. Options: { Real Number < 0 } The Mu field allows the user to define the gravitational parameter of a celestial body. Units: km^3/sec^2. z
Equatorial Radius	Default: 6378.1363. Options: { Real Number < 0 } The EquatorialRadius field allows the user to define the equatorial radius of a celestial body. Units: km.
Flattening	Default: 0.00335270. Options: { Real Number } The Flattening field allows the user to define the mass of a celestial body.Units: None.
InitialEpoch	Default: 21544.500371. Options: { Real Number }

	The InitialEpoch field allows the user to define the initial epoch, in A1 Modified Julian Date, for a celestial body. The initial epoch is only used when the user selects Analytic for the Ephemeris field on the solar system. In this case, GMAT solves Kepler's problem to determine theposition and velocity of a celestial body, using the initial epoch and state information described below. Units: A1ModJulian.
SMA	Default: 149653978.978377. Options: { Real Number >= 0 } The SMA field allows the user to define the semimajor axis of a celestial body's orbit about its central body. (Only used when the user selects Analytic for the Ephemeris field on the Solar System.) Units: km.
ECC	Default: 0.017046. Options: [Real Number >=0 } The ECC field allows the user to define the eccentricity of a celestial body's orbit about its central body. (Only used when the user selects Analytic for the Ephemeris field on the Solar System.) Units: None.
INC	Default: 23.439034. Options: { Real Number } The INC field allows the user to define the inclination of a celestial body's orbit about its central body, in the FK5 coordinate system. (Only used when the user selects Analytic for the Ephemeris field on the Solar System.) Units: deg.
RAAN	Default: 0.000186. Options: { Real Number } The RAAN field allows the user to define the right ascension of the ascending node of a celestialbody's orbit about its central body, in the FK5 coordinate system. (Only used when the user selects Analytic for the Ephemeris field on the Solar System.) Units: deg.

AOP	Default: 101.741639. Options: { Real Number } The AOP field allows the user to define the argument of periapsis of a celestial body's orbit about its central body, in the FK5 coordinate system. (Only used when the user selects Analytic for the Ephemeris field on the Solar System.) Units: deg.
TA	Default: 358.127085. Options: { Real Number } The TA field allows the user to define the true anomaly of a celestial body's orbit about its central body. (Only used when the user selects Analytic for the Ephemeris field on the Solar System.) Units: deg.

Special Fields Associated with Earth

NutationUpdate Interval	Default: 60. Options: { Real Number >=0 } The NutationUpdateInterval field, on the Earth Celestial Body, determines how often GMAT updates the Nutation matrix used in FK5 reduction. If NutationUpdateInterval is set to zero, the Nutation is updated every time a request is made to calculate the orientation of the Earth. If NutationUpdateInterval is set to a real number greater than zero, then GMAT only updates the Nutation matrix if the number of seconds defined by NutationUpdateInterval have elapsed since the last request for the Earth's orientation data. Units: sec.

Special Fields Associated with Luna

RotationData Source	Default: {DE405}. Options: { DE405, IAU2002 } The RotationDataSource, on the Luna Celestial Body, determines what source GMAT uses to obtain data describing the orientation of the moon with respect to the FK5 system. The RotationDataSource field is only used for lunar orientation data when calculating moon-based coordinate systems with the axes types of {Fixed} and {Equator}.

	Units: N/A

OpenGL Plot

This tutorial was started from the default settings in the June 9, 2008 build of GMAT.

Overview

Without OpenGL Plots, you would have no way of visualizing your spacecraft as it traveled along its trajectory. OpenGL Plots also have a multitude of options that allow you to customize your view of the spacecraft. This makes these types of plots very useful and in most cases necessary to using GMAT.

Listing of Objects and Commands that interact

- Spacecraft
  - Any spacecraft in your mission is available to the OpenGL Plot for display
- Solar System
  - The Sun and all of the Planets may be plotted or referenced in the OpenGL Plot
  - If you add any Barrycenters or Libration Points, they will also be available for plotting and reference
- Coordinate Systems
  - Both View Definition and View Up Definition may use the three default or user added coordinate systems

Using the GUI

The OpenGL Plots dialog box is quite extensive and one of the largest in GMAT. The below bullets outline each option available to you and how they all work. Refer to the screen shot below this list if you need a reference as you read. The box itself is split into several Group boxes which in turn contain up to several fields each. All of this allows you to not only change what is shown in the plot, but also how far away you view the object(s) and from what direction. There is a great deal of functionality beyond this as well.

When you have a plot open, you may change your view using the mouse. Left click on the plot and move the mouse while still holding down the button. This will change the view point of the plot. Right click on the plot and again while holding the button down, move the mouse to the left to zoom in and to the left to zoom out.

- Plot Option - Group Box
  - Collect data every BLANK step - Field
    - This value determines how many integration steps are taken between the drawing of a point on the plot. Basically, the greater this number, the less precise the plot is and the faster it will run.
  - Update plot every BLANK cycle - Field
    - This value represents the number of integration steps taken between the plot's refreshing. This means that the plot will be redrawn with the new points from the data collected above.
  - Number of points to redraw - Field

    - This value represents how many previously collected plotting points will be shown. If set to 0, all points will be drawn. If set to some positive integer then that number of previous points will be drawn. This is useful when running a long mission when you only want the most recent information displayed.

  - Show Plot - Check Box

    - If for any reason you do not want to display the plot, uncheck this box so that the object is still present in the mission tree but will not show up as an output when the mission is run.

- View Object - Group Box

  - Spacecraft - Field

    - This is a list of all Spacecraft in the resource tree. Any number of them may be added to the Selected Spacecraft Field.

  - Selected Spacecraft - Field

    - This is the list of Spacecraft that will be shown in the OpenGL Plot, as long as the Draw Object Check box is selected. These are added from those in the Spacecraft Field.

  - Draw Object - Check Box

    - Like the Show Plot Check box does for the whole Plot, this box allows an object, whether it be a spacecraft or celestial, to be not shown in the plot if unchecked. When checked, the Plot will draw this object and its path.

  - Celestial Object - Field

    - Similar to the Spacecraft Field, this list contains all of the Celestial Bodies and points that are in the Solar System section of the resource tree. These include all of the Planets, the Sun, the Moon, and any points that you add under Special Points in the Resource tree such as Barycenters or Libration Points.

  - Selected Celestial Object - Field

    - This is a list of all Celestial Objects to be displayed in the plot. Any object in the Celestial Object field may be moved here so that you may see it.

  - Orbit Color - Selection

    - This changes the color of the line that a celestial object or spacecraft will leave behind as it travels through space. In the case of a spacecraft in a targeting or other sequence that requires multiple iterations, this selection represents the color of its final trajectory.

  - Target Color - Selection

    - A spacecraft that uses multiple iterations to achieve a targeting, optimization, or other sort of problem, can be drawn through each of its runs using the color selected here. It is a good idea to keep this color different from the spacecraft's orbit color.

- Drawing Option - Group Box

  - Draw Wire Frame - Check Box

    - When selected, the Plot will display all spacecraft and celestial objects as wire frame models.
  - Draw Ecliptic Plane - Check Box
    - When selected, the ecliptic plane, or rather, the plane that the Earth travels in around the Sun, will be displayed.
  - Draw XY Plane - Check Box
    - This will draw a wire mesh plane in the plane of the X and Y axes that the Plot is referencing as the View Definition Coordinate System.
  - Draw Axes - Check Box
    - This will draw the axes of the selected coordinate system.
  - Draw Grid - Check Box
    - When selected, this option will have GMAT draw a longitude and latitude grid in the Plot.
  - Draw Sun Line - Check Box
    - When checked, this option will draw a line at all times between the centers of the Earth and Sun.
- View Definition - Group Box
  - Coordinate System - Drop Down Menu
    - This is what coordinate system will be used in the
  - View Point Reference - Drop Down Menu
    - This is set to the center of the Plot's coordinate system by default but may be changed to any spacecraft or celestial object. It is where the View Point Vector is referenced from.
  - View Point Vector - Drop Down Menu
    - The View Point Vector is where you will view the Plot from. It is the camera location, so to speak, with respect to the View Point Reference whether it be the center of the plot's coordinate system or defined by you above in the View Point Reference drop down menu.
  - View Point Vector - Fields
    - These are the fields beside the View Point Vector drop down menu where you define the actual view point vector in the X, Y, Z format.
  - View Scale Factor - Field
    - The View Scale Factor is basically a multiplier of the View Point Vector. For example, if you put 5 as a value here then you will see the objects 5 times further away.
  - View Direction - Drop Down Menu
    - This allows you to point the direction of view for the plot, the camera, toward the direction of your choosing. This may be towards a spacecraft or celestial object.
- View Option - Group Box

- Use Initial View Def. - Check Box
  - Use Initial View Definition is very useful when doing multiple runs of a mission. If left checked, your view of the plot will always start out the same. If you uncheck this box and move your view of the plot with the mouse once the mission has finished running, you can re-run the mission and the view will stay where you left it.

- View Up Definition - Group Box
  - Coordinate System - Drop Down Menu
    - The View Definition group box only gives you where your view of the plot is located from but does not tell GMAT which way is up within that view. This drop down menu allows you to set what coordinate system is referenced when setting the up direction. This will usually be the same coordinate system as used to define the View Direction.
  - Axis - Drop Down Menu
    - This tells GMAT what axis of the View Up Definition Coordinate System to point up in your plot.

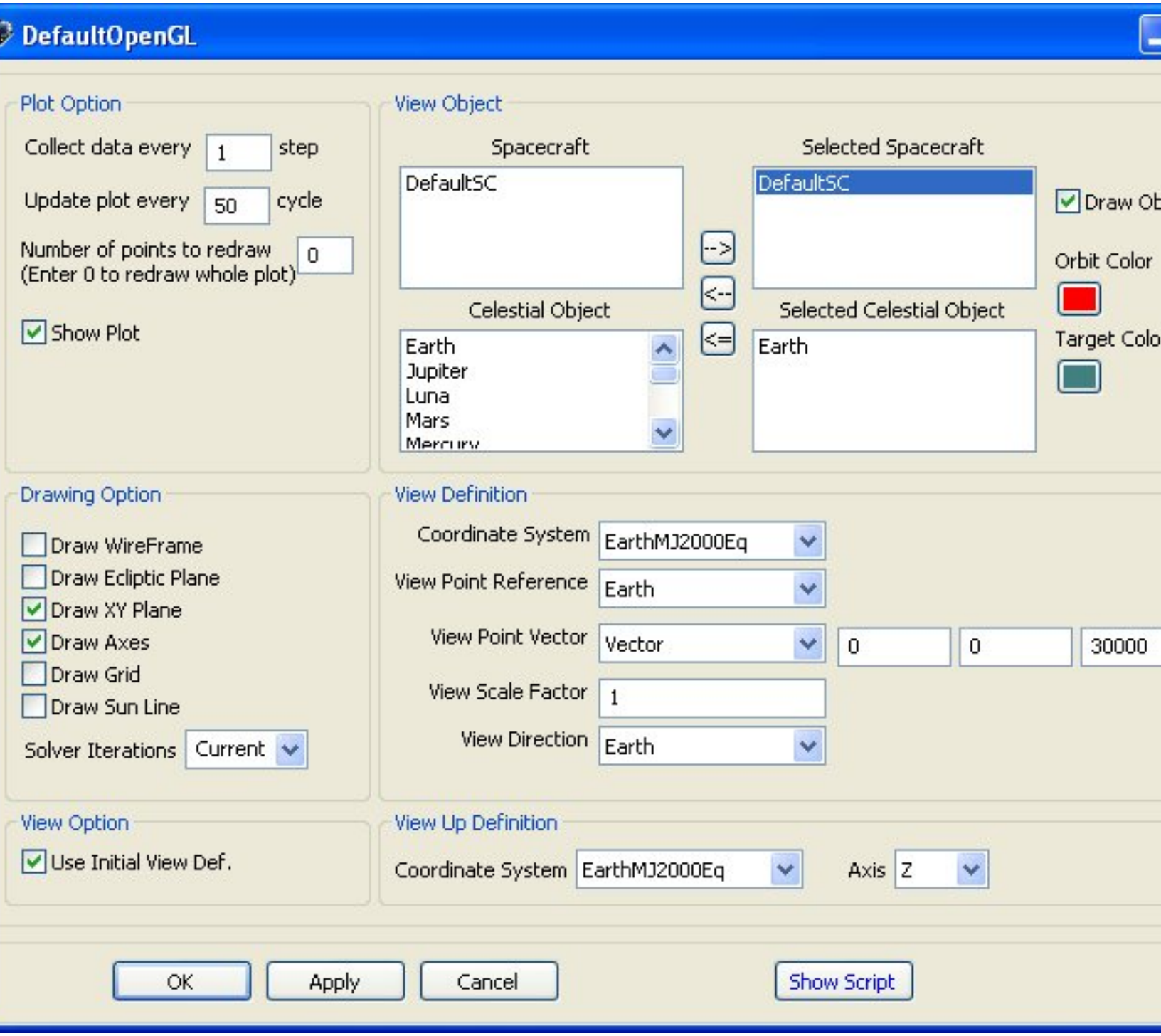


Figure: OpenGL Plot Default Dialog Box

Using the Script

Script Syntax

```
Create OpenGLPlot DefaultOpenGL;
```

```
GMAT DefaultOpenGL.SolverIterations = Current;
GMAT DefaultOpenGL.Add = {DefaultSC, Earth};
GMAT DefaultOpenGL.OrbitColor = [ 255 32768 ];
GMAT DefaultOpenGL.TargetColor = [ 8421440 0 ];
GMAT DefaultOpenGL.CoordinateSystem = EarthMJ2000Eq;
GMAT DefaultOpenGL.ViewPointReference = Earth;
GMAT DefaultOpenGL.ViewPointVector = [ 0 0 30000 ];
GMAT DefaultOpenGL.ViewDirection = Earth;
GMAT DefaultOpenGL.ViewScaleFactor = 1;
GMAT DefaultOpenGL.ViewUpCoordinateSystem = EarthMJ2000Eq;
GMAT DefaultOpenGL.ViewUpAxis = Z;
GMAT DefaultOpenGL.CelestialPlane = Off;
GMAT DefaultOpenGL.XYPlane = On;
GMAT DefaultOpenGL.WireFrame = Off;
GMAT DefaultOpenGL.Axes = On;
GMAT DefaultOpenGL.Grid = Off;
GMAT DefaultOpenGL.SunLine = Off;
GMAT DefaultOpenGL.UseInitialView = On;
GMAT DefaultOpenGL.DataCollectFrequency = 1;
GMAT DefaultOpenGL.UpdatePlotFrequency = 50;
GMAT DefaultOpenGL.NumPointsToRedraw = 0;
GMAT DefaultOpenGL.ShowPlot = true;
```

Reference Tables

Fields associated with Plot Options

DataCollectFrequency	Default: 1 Options: { Integer >= 1 } The DataCollectFrequency field allows the user to define how data is collected for plotting. It is often inefficient to draw every ephemeris point associated with a trajectory. Often, drawing a smaller subset of the data still results in smooth trajectory plots, while executing more quickly. The DataCollectFrequency is an integer that represents how often to collect data and store for plotting. If DataCollectFrequency is set to 10, then Data is collected every 10 integration steps. Units: Integration Steps
UpdatePlotFrequency	Default: 50 Options: { Integer >= 1 } The UpdatePlotFrequency field allows the user to specify how often to update an OpenGL plot is updated with new data collected during the process of propagating spacecraft and running a mission. Data is collected for a plot according the value defined by DataCollectFrequency. An

	OpenGL plot is updated with the new data, according to the value set in UpdatePlotFrequency. If UpdatePlotFrequency is set to 10 and DataCollectFrequency is set to 2, then the plot is updated with new data every 20 (10*2) integration steps. Units: Integration Steps
NumPointsToRedraw	Default: 0 Options: { Integer >= 0 } When NumPointsToRedraw is set to zero, all ephemeris points are drawn. When NumPointsToRedraw is set to a positive integer, say 10 for example, only the last 10 collected data points are drawn. See DataCollectFrequency for explanation of how data is collected for an OpenGL plot. Units: Integration Steps
ShowPlot	Default: true Options: { true , false } The ShowPlot field allows the user to turn off a plot for a particular run, without deleting the plot object, or removing it from the script. If you select true, then the plot will be shown. If you select false, then the plot will not be shown. Units: N/A

Fields associated with Viewed Objects

Add	Default: DefaultSC, Earth Options: { SpacecraftName CelestialBodyName LibrationPointName BarycenterName } The Add subfield adds a spacecraft,celestial body, libration point,or barycenter to a plot. When creating a plot the Earth is added as a default body and may be removed by using the Remove command. The user can add a spacecraft, celestial body, libration point, or barycenter to a plot by using the name used to create the object. The GUI's Selected field is the equivalent of the script's Add field. In the event of no Add command or no objects in the Selected field, GMAT should run without the OpenGL plot and a warning message displayed in the message window. The following warning message is sufficient: OpenGL plot will be turned off. No object has been selected for plotting.

	Units: N/A
Remove	Default: No Default Options: { Any object included in the Add list } The Remove subfield removes a spacecraft,celestial body, libration point, or barycenter from a plot. The user can remove any object that has been added to a plot by using the name used to add the object. Units: N/A

Fields associated with Drawing Options

WireFrame	Default: Off Options: { On , Off } When the WireFrame field is set to On, celestial bodies are drawn using a wireframe model. When the WireFrame field is set to Off, then celestial bodies are drawn using a full map. Units: N/A
EclipticPlane	Default: Off Options: { On , Off } Note: Only allowed for OpenGL plots with Coordinate Systems that use the MJ2000Eq axis system The EclipticPlane field allows the user to tell GMAT to draw a grid representing the ecliptic plane in an OpenGL plot. Note, the ecliptic plane can currently only be drawn for plots whose coordinate system uses the MJ2000Eq axis system. Units: N/A
XYPlane	Default: On Options: { On , Off } The XYPlane flag allows the user to tell GMAT to draw a grid representing the XY-plane of the coordinate system selected under the CoordinateSystem field of the OpenGL plot. Units: N/A
Axes	Default: On Options: { On , Off } The Axis flag allows the user to tell GMAT to draw the Cartesian axis system associated with the coordinate system selected under the CoordinateSystem field of an OpenGL plot.

	Units: N/A
Grid	Default: On Options: { On , Off } The Grid flag allows the user to tell GMAT to draw a grid representing the longitude and latitude lines celestial bodies added to an OpenGL plot. Units: N/A
EarthSunLines	Default: On Options: { On , Off } The EarthSunLines allows the user to tell GMAT to draw a line that starts at the center of Earth and points towards the Sun. Units: N/A
SolverIterations	Default: Off Options: { On , Off } The SolverIterations field determines whether or not perturbed trajectories are plotted during a solver (Targeter, Optimize) sequence. When SolverIterations is set to On, solver iterations are shown on the plot. When SolverIterations is Off, the solver iterations are not shown on the plot. Units: N/A

Fields Associated with View Definition

CoordinateSystem	Default: EarthMJ2000Eq Options: { Any default or user defined coordinate system } The CoordinateSystem field on an OpenGL plot allows the user to select which coordinate system to use to draw the plot data. A coordinate system is defined as an origin and an axis system, and the CoordinateSystem field allows the user to determine the origin and axis system of an OpenGL plot. See the CoordinateSystem object fields for information of defining different types of coordinate systems. Units: N/A
ViewPointReference	Default: Earth Options: { SpacecraftName CelestialBodyName LibrationPointName BarycenterName, or a 3-vector of numerical values }

	The ViewPointReference field is an optional field that allows the user to change the reference point from which ViewPointVector is measured. ViewPointReference} defaults to the origin of the coordinate system for the plot. A ViewPointReference can be any spacecraft, celestial body, libration point, or barycenter. Units: N/A
ViewPointVector	Default: [0 0 30000] Options: { SpacecraftName CelestialBodyName LibrationPointName BarycenterName, or a 3-vector of numerical values } The product of ViewScaleFactor and ViewPointVector field determines the view point location with respect to ViewPointReference. ViewPointVector can be a vector, or any of the following objects: spacecraft,celestial body, libration point,or barycenter. The location of the Viewpoint in three-space is defined as the vector addition of ViewPointReference, and the vector defined by product of ViewScaleFactor and ViewPointVector in the coordinate system chosen by the user. Units: km or N/A
ViewScaleFactor	Default: 1 Options: { Real Number >= 0 } The ViewScaleFactor field scales ViewPointVector before adding it to ViewPointReference. The ViewScaleFactor allows the user to back away from an object to fit in the field of view. Units: None
ViewDirection	Default: Earth Options: { SpacecraftName CelestialBodyName LibrationPointName BarycenterName, or a 3-vector of numerical values } The ViewDirection field allows the user to select the direction of view in an OpenGL plot. The user can specify the view direction by choosing an object to point at such as a spacecraft,celestial body, libration point,or barycenter. Alternatively, the user can specify a vector of the form [x y z]. If the user specification of ViewDirection, ViewPointReference, and ViewPointVector, results in a zero vector, GMAT uses [0 0 10000] for ViewDirection.

	Units: km or N/A

Fields Associated with View Options

UseInitialView	Default: On Options: { On , Off } The UseInitialView field allows the user to control the view of an OpenGL plot between multiple runs of a mission sequence. The first time a specific OpenGL plot is created, GMAT will automatically use the view as defined by the fields associated with View Definition, View Up Direction, and Field of View. However, if the user changes the view using the mouse, GMAT will retain this view upon rerunning the mission if UseInitialView is set to false. If UseInitialView is set to true, the view for an OpenGL plot will be returned to the view defined by the initial settings. Units: N/A

Fields Associated with View Up Definition

ViewUpCoordinate System	Default: EarthMJ2000Eq Options: { Any default or user defined coordinate system } The ViewUpCoordinateSystem and ViewUpAxis fields are used to determine which direction appears as up in an OpenGL plot and together with the fields associated the the View Direction, uniquely define the view. The fields associated with the View Definition allow the user to define the point of view in 3-space, and the direction of the line of sight. However, this information alone is not enough to uniquely define the view. We also must provide how the view is oriented about the line of sight. This is accomplished by defining what direction should appear as the up direction in the plot and is configured using the ViewUpCoordinateSystem field and the ViewUpAxis field. The ViewUpCoordinateSystem allows the user to select a coordinate system to define the up direction. Most of the time this system will be the same as the coordinate system chosen under the CoordinateSystem field. Units: N/A
ViewUpAxis	Default: Z Options: { X , -X , Y , -Y , Z , -Z }

	The ViewUpAxis allows the user to define which axis of the ViewUpCoordinateSystem that will appear as the up direction in an OpenGL plot. See the comments under ViewUpCoordinateSystem for more details of fields used to determine the up direction in an OpenGL plot. Units: N/A

XYPlot

This was created using the June 18, 2008 build of GMAT.

Overview

The XY Plot is a graph under the Plots/Reports folder in the resource tree that plots data onto the X and Y axes of the graph. Any two parameters can be chosen to plot from the Parameter Select dialog box when the View radio button is clicked. The plot has the capability to be turned on and/or off throughout the mission if desirable.

Listing of Objects and Commands that interact

- Spacecraft
  - Spacecraft interact with an XY Plot throughout the entire mission. The data retrieved from the spacecraft, as it carries out the command, is what gets plotted onto the graph.
- Toggle Command
  - A Toggle can be inserted into the mission sequence to control when the XY Plot is to plot data by subscribing it to the Toggle list. If it is necessary to only plot data at a certain point during the mission, besides at the beginning or end points, then two Toggle commands can be added to switch it on and off.

Using the GUI

The XY Plots dialog box has a group box, two fields, and a couple View buttons that open the Parameter Select dialog box. When the mission is run, the XY Plot will set aside a portion of the screen for itself alongside any other plots created in the resource tree.

- Options - Group Box
  - This group box located on the far left column gives the options of whether the plot shows the faint background grid, none, some, or all of the solver iterations, the color of the plotted data, or whether the plot even appears while running the mission.
- Selected X - Field
  - This field is where an X parameter is chosen to be plotted on the X axes.
- Selected Y - Field
  - This field is where an Y parameter is chosen to be plotted on the Y axes.

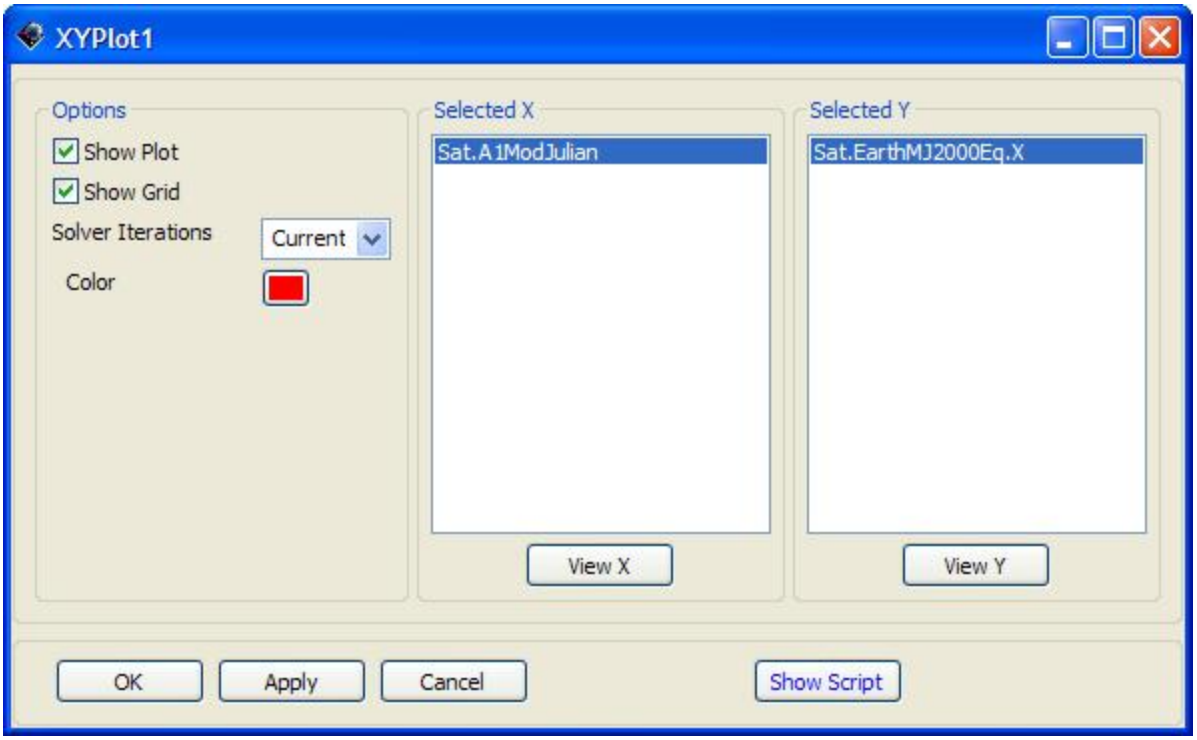


Figure: Default Name and Settings for the XY Plot Dialog Box

Using the script

Script Syntax

```
Create XYPlot XYPlot1;
GMAT XYPlot1.SolverIterations = Current;
GMAT XYPlot1.IndVar = Sat.A1ModJulian;
GMAT XYPlot1.Add = {Sat.EarthMJ2000Eq.X};
GMAT XYPlot1.Grid = On;
GMAT XYPlot1.ShowPlot = true;
```

Reference Table

IndVar	Default: DefaultSC.A1ModJulian Options: { Any user variable, array element, or spacecraft parameter } The IndVar field allows the user to define the independent variable for an xy-plot. Only one variable can be defined as an independent variable. For example, the line MyXYPlot.IndVar = DefaultSC.A1ModJulian sets the independent variable to be the epoch of DefaultSC in the A1 time system and modified Julian format. Units: N/A.

Add	Default: DefaultSC.EarthMJ2000Eq.X. Options: { Any user variable, array element, or spacecraft parameter } The Add field allows the user to add dependent variables to an xy-plot. All dependent variables are plotted on the y-axisvs the independent variable defined by IndVar. To define multiple dependent variables, they should be included in curly braces. For example, MyXYPlot.Add = DefaultSC.EarthMJ2000Eq.Y , DefaultSC.EarthMJ2000Eq.Z. The GUI's Selected field is the equivalent of the script's Add field. In the event of no Add command or no objects in the Selected field, GMAT should run without the XYPlot and a warning message displayed in the message window. The following warning message is sufficient: XYPlot will be turned off. No object has been selected for plotting. Units: N/A.
Grid	Default: On. Options: { On, Off } When the Grid field is set to On, then a grid is drawn on an xy-plot. When the Grid field is set to Off, then a grid is not drawn. Units: N/A.
SolverIterations	Default: Off. Options: { On, Off } The SolverIterations field determines whether or not perturbed trajectories are plotted during a solver (Targeter, Optimize) sequence. When SolverIterations is set to On, solver iterations are shown on the plot. When SolverIterations is set to Off, solver iterations are not shown on the plot. Units: N/A.
ShowPlot	Default: true. Options: { true, false } The ShowPlot field allows the user to turn off a plot for a particular run, without deleting the plot object, or removing it from the script. If you select true, then the plot will be shown. If you select false, then the plot will not be shown. Units: N/A.

Coordinate Systems

This was created using the June 9, 2008 build of GMAT.

Overview

Coordinate Systems are critical to GMAT for several reasons. They are what every object represented within the software is referenced to. They are used by GMAT as the basis for which all calculations are made. They also provide the reference for any OpenGL Plot that is created.

Listing of Objects and Commands that interact

- Thruster
  - The Thruster object allows you to set a coordinate system as its reference.
- Spacecraft
  - In the spacecraft dialog box you may change what coordinate system the spacecraft's position is defined in reference, whatever the state type may be set as.
- OpenGL Plot
  - Coordinate Systems are very key to the display of OpenGL Plots. They rely on coordinate systems to set how exactly the view of the plot will look using both the View Definition and View Up Definition sections of the OpenGL Plot dialog box.
- Parameter Select Dialog Box
  - Whenever you may select a parameter using the parameter select dialog box, you have the option of selecting certain options such X, Y, Z, and several others that will require to set a coordinate system for them to reference.

Using the GUI

Firstly, you cannot edit the three default coordinate systems. You may add as many as you need to complete your mission. When you add a coordinate system, you will see a dialog box asking you to name the new coordinate system. You may also set its origin and axes type. Depending on the type of axes you set, you will be able to change certain parameters of it. The only difference between the dialog boxes for creating and modifying an existing coordinate system that you have created is that the former will have a field for you to enter the new system's name where as the latter will already be named and not have this option.

- Coordinate System Name - Field
  - This is an option only if you are creating a new coordinate system. This will be the name of the new coordinate system you create.
- Origin - Drop Down Menu
  - This is where the origin, otherwise known as the point [[0 , 0 , 0] on the Axes, will be.
- Axes - Group Box
  - Type - Drop Down Menu

    - This is where you select what type of axes you want for the coordinate system. For a number of these options, there are no other parameters to change and this will be the only changeable option in the Axes group box.

- Primary - Drop Down Menu
    - This and the 'Secondary' drop down menu are only changeable when the Axes type is set to Object Referenced.
- Secondary - Drop Down Menu
    - The Object Referenced coordinate system will always be fixed with respect to this and the Primary object.
- Epoch Format - Drop Down Menu
    - Unchangeable
- Epoch - Field
    - If the coordinate system you select requires a set epoch, this is where it will be changed.
- X - Drop Down Menu
    - Along with the 'Primary' and 'Secondary' drop down menus, X, Y, and Z may only be set when the Axes type is object referenced.
- Y - Drop Down Menu
    - Setting one of these axes to R will mean it is the axis that always points between the two objects. V will be the axis of their relative velocity. These are both expressed in an inertial frame of reference.
- Z - Drop Down Menu
    - Setting one of these axes to N will mean it is the orthogonal axis to R and V.
- Update Interval - Field
    - If the selected axes type changes with respect to another object such as a planet, it may need to be updated every so often in order to maintain a correct position. This value is, in seconds, how much time passes in between these updates.

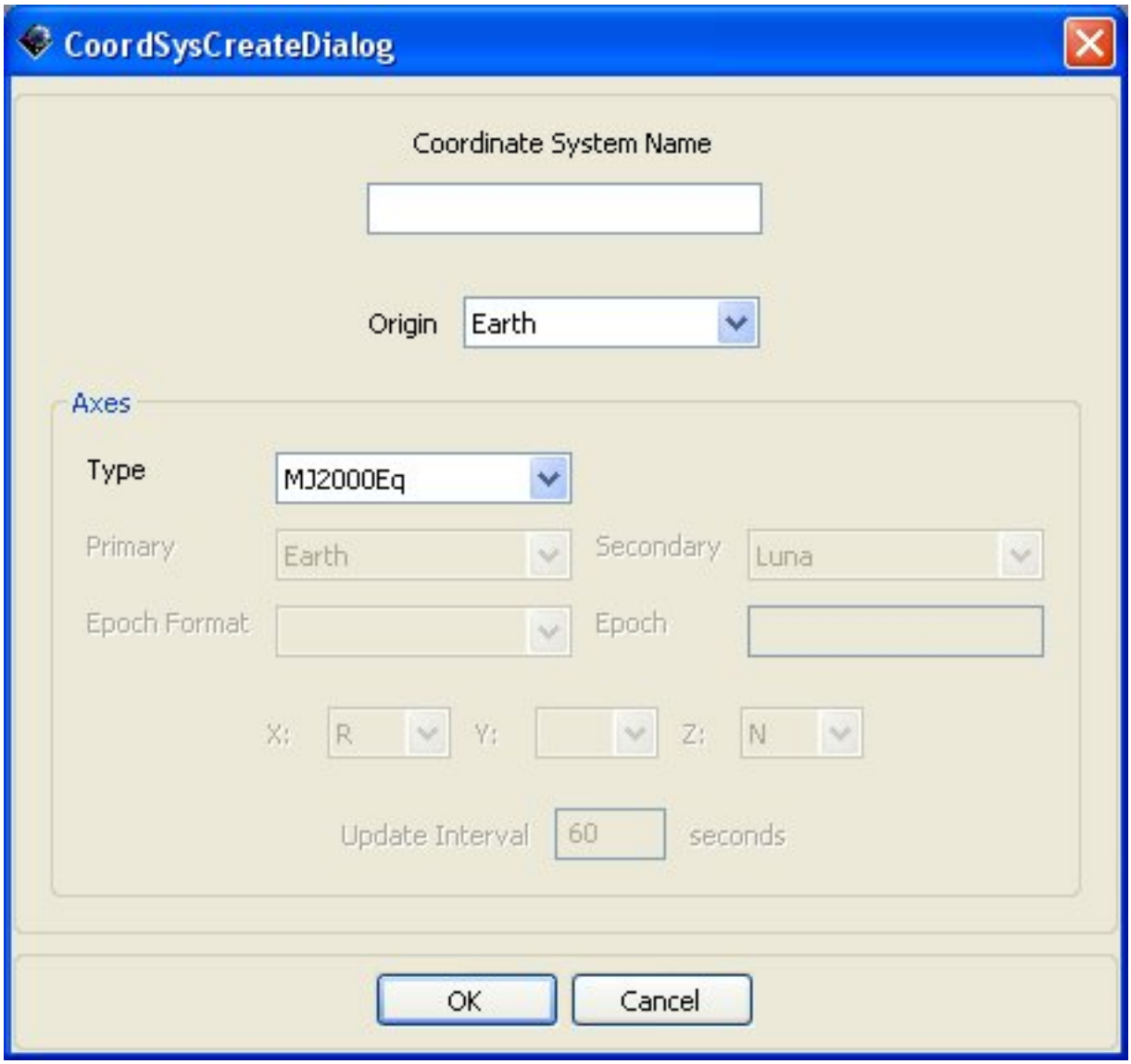


Figure: Default Name and Settings for the Create Coordinate System Dialog Box

Using the script

Script Syntax

```
Create CoordinateSystem EarthMJ2000Eq;
GMAT EarthMJ2000Eq.Origin = Earth;
GMAT EarthMJ2000Eq.Axes = MJ2000Eq;
GMAT EarthMJ2000Eq.UpdateInterval = 60;
GMAT EarthMJ2000Eq.OverrideOriginInterval = false;
```

Finite Burn

This was created using the July 2, 2008 build of GMAT.

Overview

The Finite Burn object is used when a continuous propulsion is desired. Impulsive burns happen instantaneously and through a Maneuver command, while finite burns occur until the End Finite Burn command is reached in the mission sequence and are typically coupled with Propagate commands.

Reference Table

Origin	Default: Earth. Options: { Any celestial body, libration point, or barycenter } Together the Origin and Axes fields describe the coordinate system in which a maneuver is applied. The Origin field determines the origin of the maneuver coordinate system. The ability to define the coordinate system locally avoids having to create many coordinate systems, associated with specific spacecraft, in order to perform finite maneuvers for multiple spacecraft. Units: N/A.
Axes	Default: VNB. Options: { VNB, MJ2000Eq } The Axes field, together with the Origin field, describe the coordinate system in which a finite maneuver is applied. If VNB is chosen for Axes, a local coordinate system is created such that the x-axis points in the velocity direction of the spacecraft, with respect to the point defined by Origin, the y-axis points in the normal direction of the spacecraft with respect to Origin, and the z-axis completes the right-handed set. Units: N/A.
Thrusters	Default: No Default. Options: { Any thruster created by user } The Thrusters field allows the selection of which thrusters to use when applying a finite maneuver. The user can select more than one thruster, from the list of thrusters previously created, by including all thrusters in curly braces. An example is MyFiniteBurn.Thrusters = Thruster1,Thruster2,Thruster3. Units: N/A.
BurnScaleFactor	Default: 1.0. Options: { Real Number }

	The BurnScaleFactor is used to scale the total acceleration before adding the acceleration due to a finite burn into the sum of the accelerations of a spacecraft. The scaling is performed by taking the sum of the accelerations applied by all thrusters specified under the Thrusters field, and multiplying the total thrust by BurnScaleFactor. Units: None.

Examples

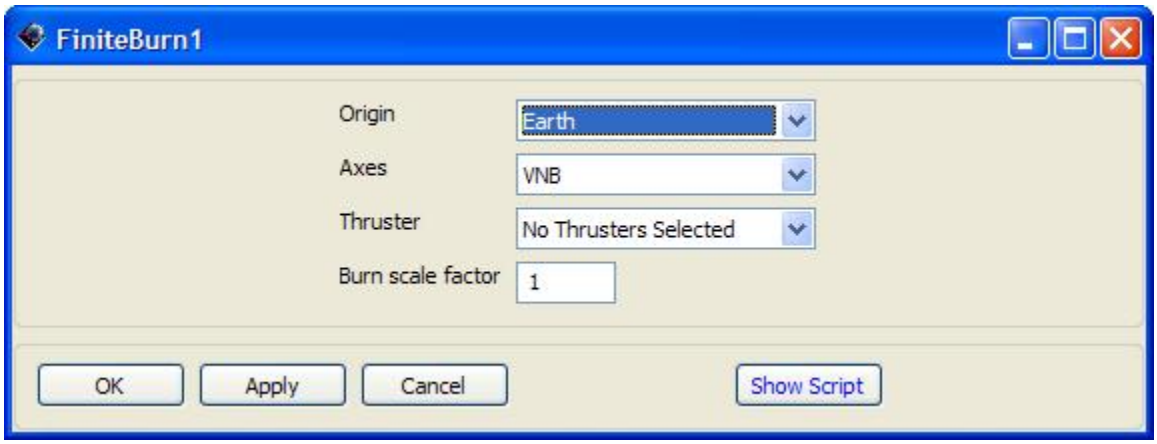


Figure: Default Name and Settings for the Finite Burn Object Dialog Box

```
Create FiniteBurn FiniteBurn1;
GMAT FiniteBurn1.Origin = Earth;
GMAT FiniteBurn1.Axes = VNB;
GMAT FiniteBurn1.BurnScaleFactor = 1;
```

Objects and Commands Interactions

- Spacecraft
  - A spacecraft must be created in order to apply any burn.
- Thruster
  - Any thruster created in the resource tree can be incorporated into a finite burn to be used on the spacecraft.
- Begin and End Finite Burn command
  - After a finite burn is created, to apply it to the mission sequence, a Begin and End Finite Burn command must be appended to the mission tree.

Variable

This was created using the June 9, 2008 build of GMAT.

Overview

The Variable object allows you to create and name a variable and assign to it a real number value. A variable can be used in numerous commands which allows you to customize the mission sequence to your application. In the simplest case, a variable can be defined by a simple assignment to a numeric literal. In more complex cases, a variable can be defined using an assignment that contains a complicated mathematical expression.

Interfaces

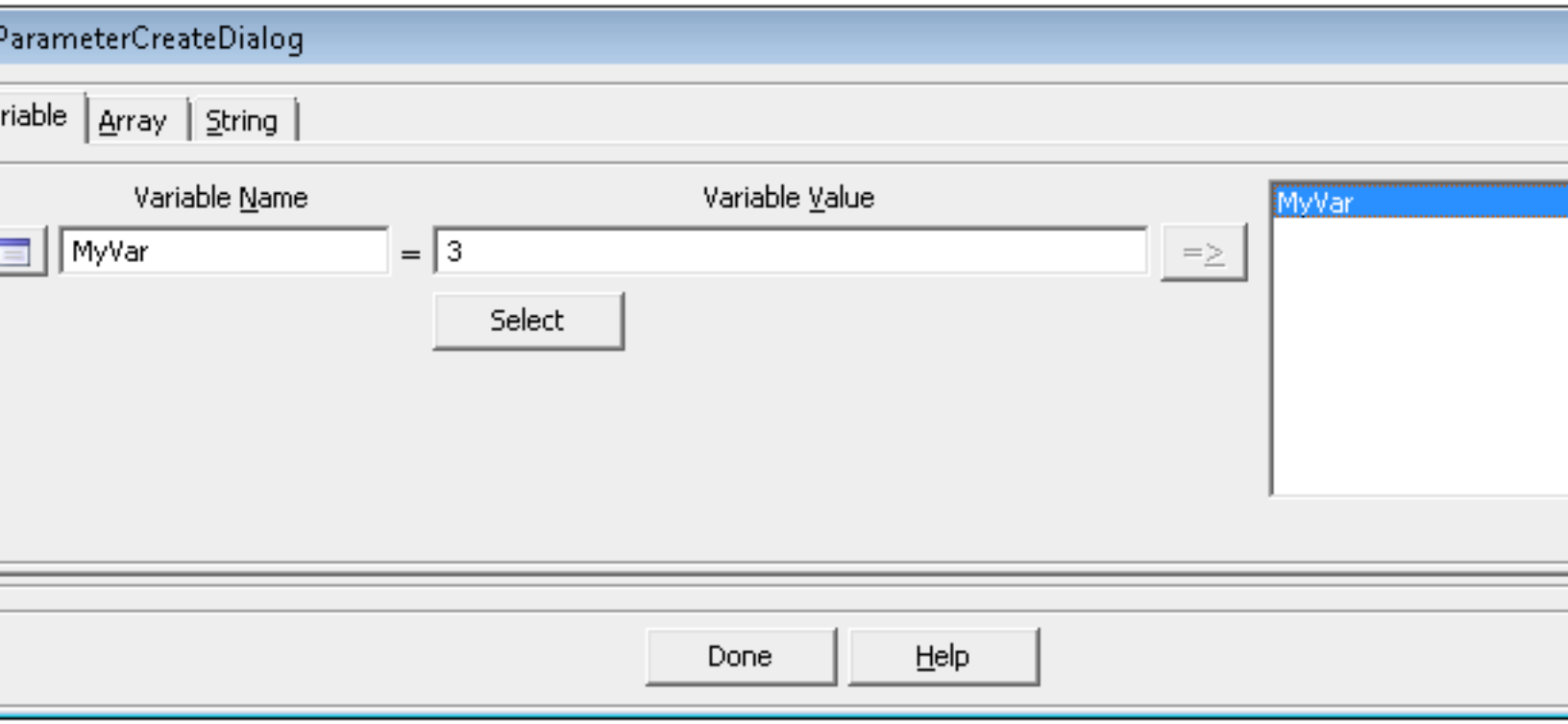


Figure: Parameter Create Dialog Box

Script Syntax

```
GMAT [Variable Name] = [expression]
```

Script Examples

```
Create Variable pi Energy mu SMA
mu     = 398600.4415;
pi     = 3.14159265358979;
Energy = MySpacecraft.VMAG^2/2 - mu/r;
SMA    = -mu/2/Energy;
```

Object and Commands Interactions

- Spacecraft Object

- Report Command
- Equation Command
- Vary Command
- Achieve Command
- Minimize Command

String

This was created from the default settings in the July 23, 2008 build of GMAT.

Overview

This page will show you how to create and use String objects. Stings are useful for storing characters as a set. One possible use of them is to report back a specific message at a set point in the mission sequence.

Listing of Objects and Commands that interact

- ReportFile
  - You may add a string to the Parameter List in the Report Object dialog box via the ParameterSelectDialog box.
- Report Command
  - Alternatively, you may add a string to the Parameter List in the Report Command dialog box

Using the GUI

A string's value may be set at its creation or later by editing it by double clicking on it in the Resource tree. The dialog box for creating a String may be used to create many at once while defining their values if you choose.

- Create - Button
  - Press this button after giving a name to a string. You may also define its value in the Value field before creating it if you wish.
- Name - Field
  - You can define the name of the String here.
- Value - Field
  - You can define the value of the String here.
- String - List
  - This is where the strings you have previously created will appear. They appear here once you click the Create button.

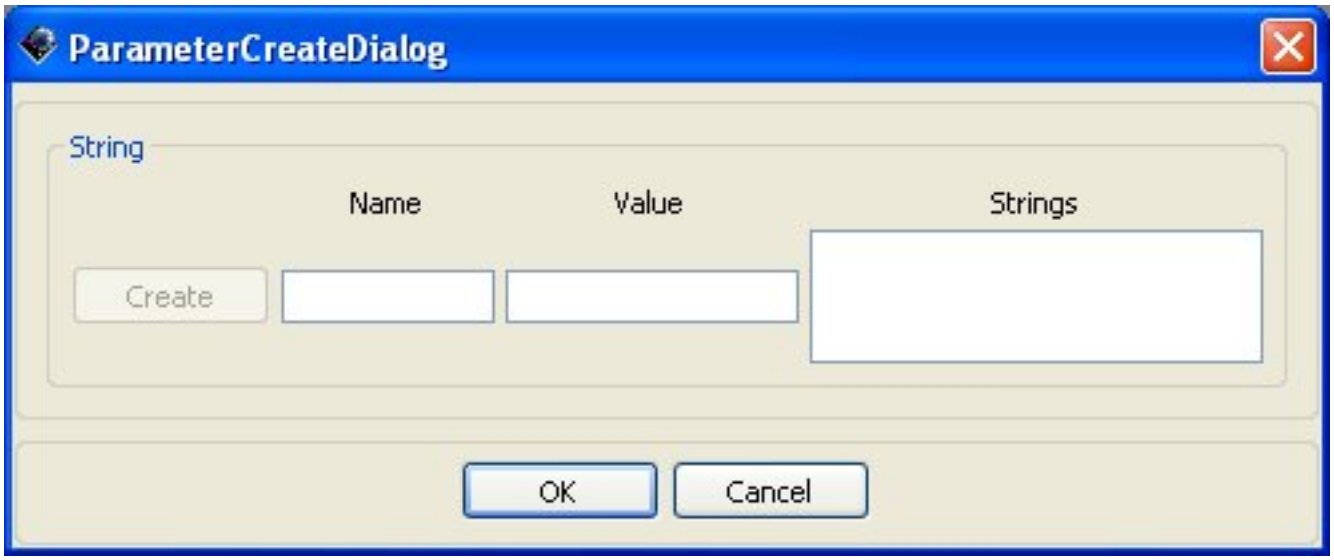


Figure: Default Name and Settings for the create String Object Dialog Box

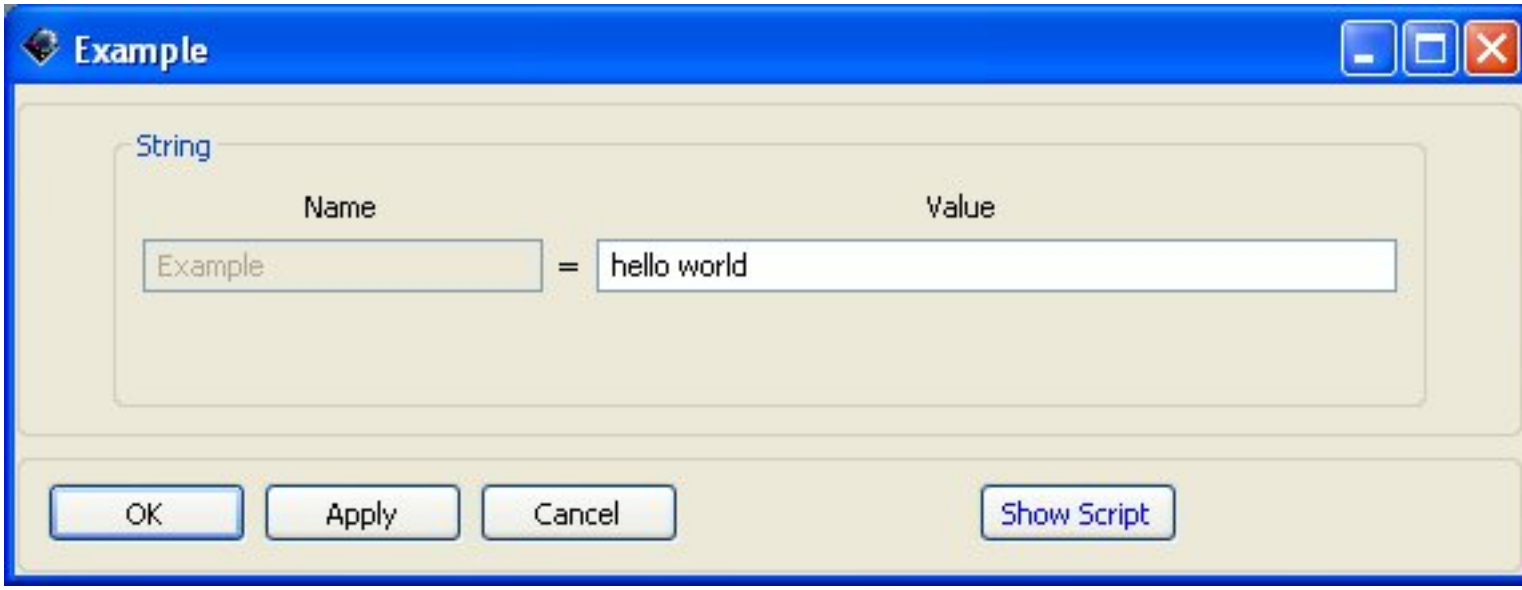


Figure: Default Name and Settings for the edit String Object Dialog Box

Using the script

Script Syntax

```
GMAT String Name = String;
```

Script Examples

```
% The following is an example String declaration
GMAT Example = hello world;
```

Array

Note: This page is a work in progress.

This tutorial was started from the default settings in the July 2, 2008 build of GMAT.

Overview

The purpose of an array is to store multiple parameters under one object, such as components of a vector.

Listing of Objects and Commands that interact

- Report Command
  - Report commands can be used to retrieve information within arrays or from the entire array.

Using the GUI

- Name - Field
  - User can define the name of the Array.
- Row - Field
  - Input how many rows are needed based on given or desired information
- Column - Field
  - Input how many columns are needed based on given or desired information
- Arrays - Field
  - To be added

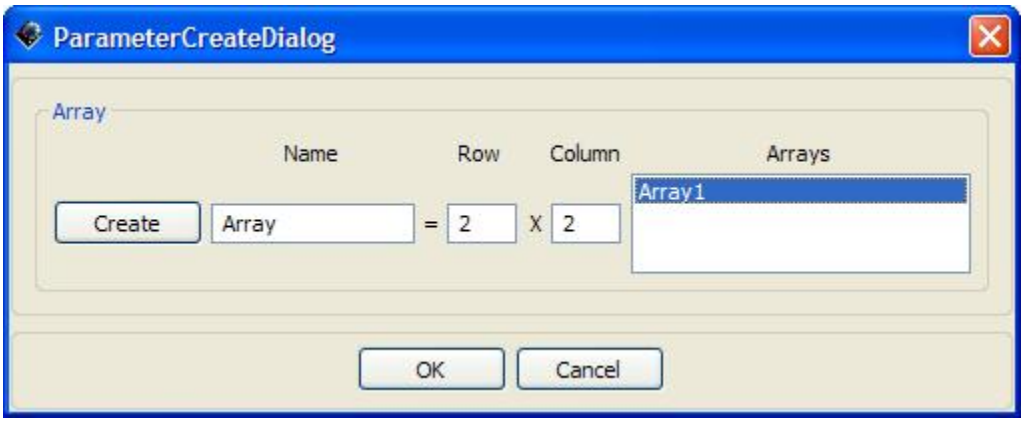


Figure: Default Name and Settings for the Array Object Parameter Select Dialog Box

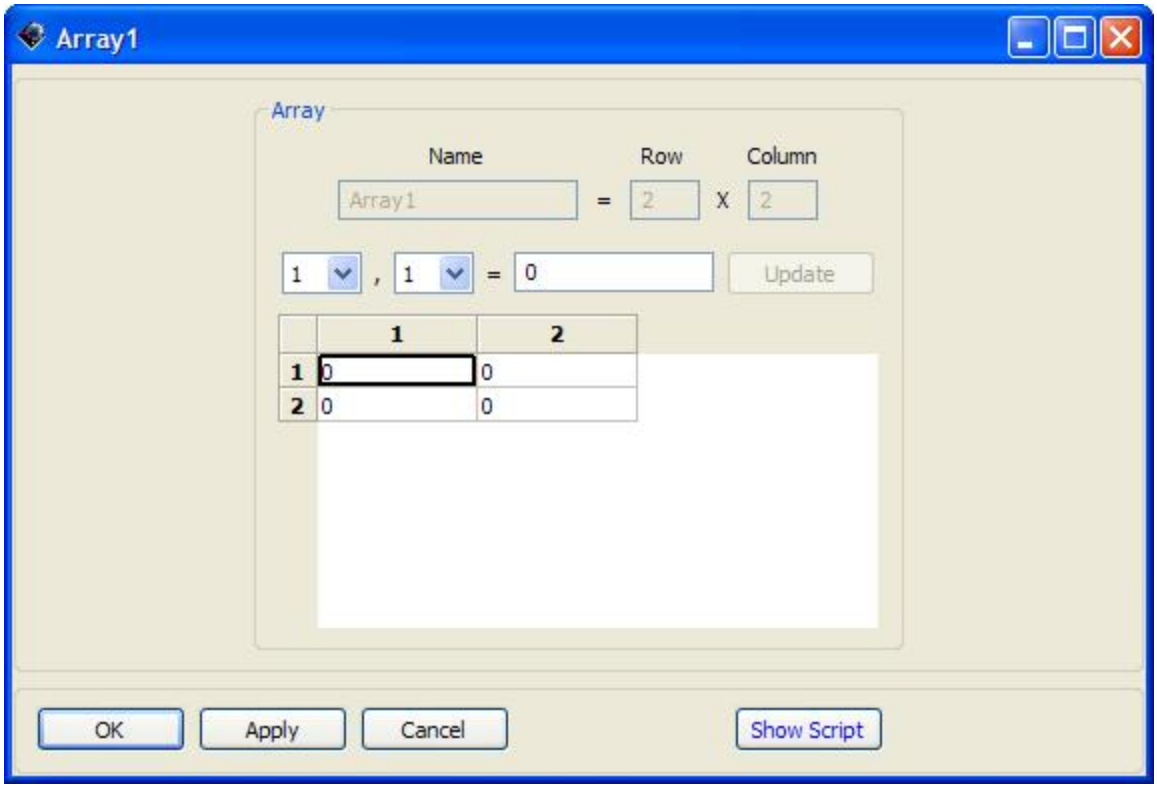


Figure: Default Name and Settings for the Array Object Dialog Box

Using the script

Script Syntax

```
Create Array Array1[2,2];
```

Reference Table

$ITEM NAME. WILL MOST LIKELY USE COURIER NEW HTML FORMATTING$	Default: $ITEM DEFAULT VALUE$ Options: { $ITEM OPTIONS$ } $ITEM DESCRIPTION$ Units: $UNITS$
$ITEM NAME. WILL MOST LIKELY USE COURIER NEW HTML FORMATTING$	Default: $ITEM DEFAULT VALUE$ Options: { $ITEM OPTIONS$ } $ITEM DESCRIPTION$ Units: $UNITS$

Barycenter

This was created using the July 9, 2008 build of GMAT.

Overview

A barycenter is the center of mass of two or more objects and is located under the Solar System folder in the resource tree. The Barycenter object allows you to create a special point that can be used as a reference point or a point to be used for the plots.

Listing of Objects and Commands that interact

- Coordinate Systems
  - Interacts with the barycenter object selecting it as the origin of the coordinate system or as a primary or secondary point for defining the axes.
- OpenGL Plot
  - It can be selected as a celestial object to be drawn onto the plot, a View Point Reference, or the View Direction within the dialog box.

Using the GUI

- Available Bodies
  - Under this column are the list of celestial bodies that can be combined to make a barycenter or the center of mass. This list includes: Sun, Mercury, Venus, Jupiter, Saturn, Uranus, Neptune, Pluto, Earth, and Luna. As some of these bodies are added to Bodies Selected, they are automatically removed from the list of available bodies. When a body is selected in the Available Bodies column, the right arrow can be clicked to move it to the Bodies Selected column.
- Bodies Selected
  - The barycenter is calculated using the bodies that are under this column. This column should contain two or more bodies for this object to be effective, unless a body's center of mass is significantly off center. By default, Earth and Luna start in this column. If an object in this column is selected and the left arrow is clicked, that object is cleared from the Body Selected column and moved back to the list of potential bodies used in the Available Bodies column. If the double left arrow is clicked, all of the bodies in the Bodies Selected column are moved to the Bodies Selected column.

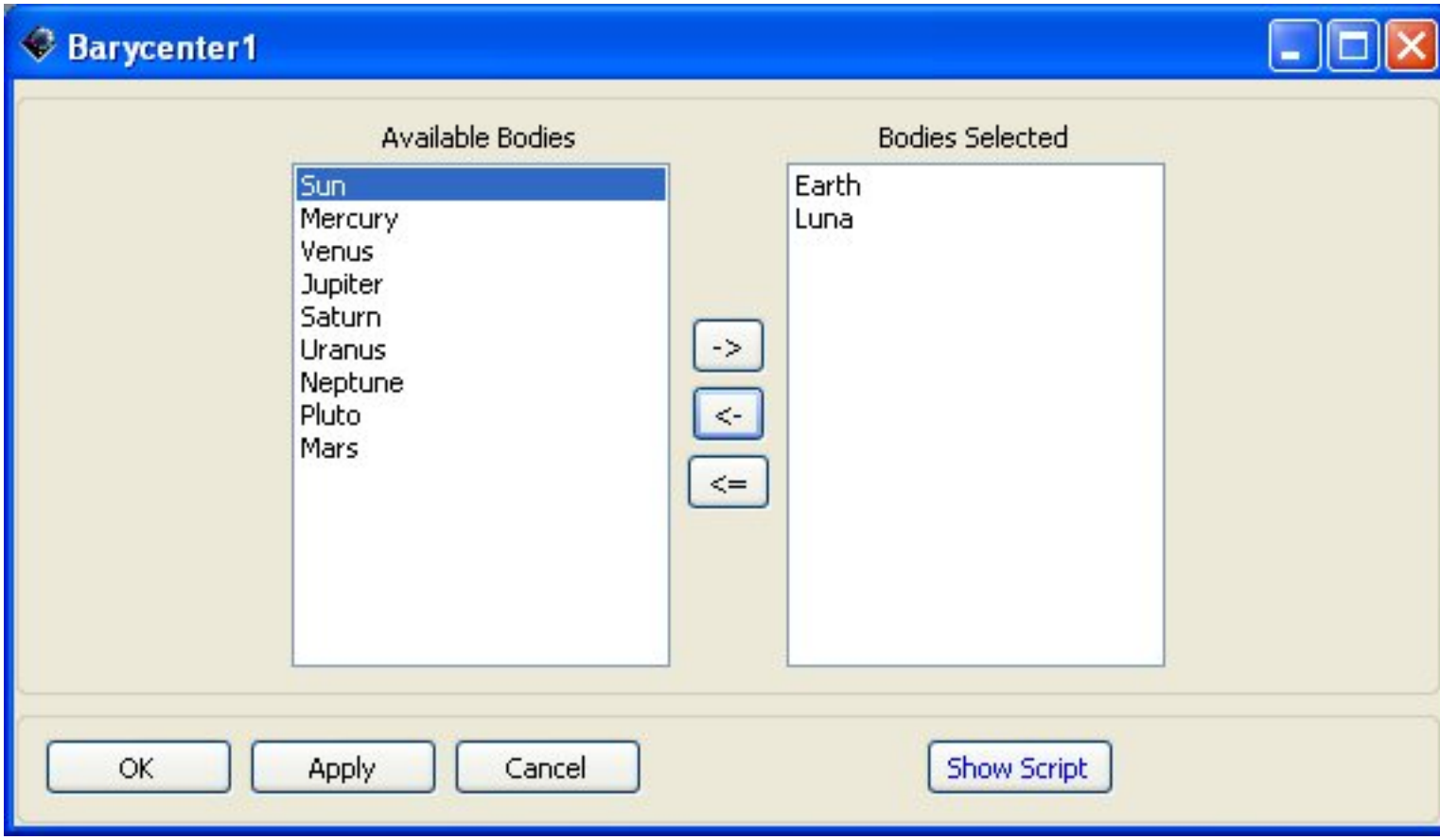


Figure: Barycenter Object Dialog Box

Using the Script

Script Syntax

```
Create Barycenter EarthMoonBary;
GMAT EarthMoonBary.BodyNames = {Earth, Luna};
```

Reference Table

<BodyNames>	Default: Earth, Luna. Options: { Sun, Mercury, Venus, Earth, Luna, Mars, Jupiter, Saturn, Uranus, Neptune, Pluto. (At least one body must be selected!) } The BodyNames field is a list that contains the bodies used to define a barycenter. In a script, the list must be surrounded by curly braces. (i.e. BaryCenterName.BodyNames = { Earth, Luna }) Units: N/A.

Libration Point

This was created using the July 9, 2008 build of GMAT.

Overview

The object Libration Point allows you to add a special point that can be used as a reference point or a point to be used for the plots. The libration point, also called a Lagrange point, is a point in space that a satellite can orbit about even though there is no central body. it is a phenomenon of a three-body system. It is part of the resource tree and can be added under Solar System.

Listing of Objects and Commands that interact

- Coordinate System
  - Interacts with libration point in that libration point can be selected as the origin of the coordinate system or as a primary or secondary point for defining the axes.
- OpenGL plot
  - Interacts with a libration point in a number of ways. It can be selected as a celestial object to be drawn onto the plot. A libration point can also be selected as the View Point Reference or the View Direction.
- Barycenter
  - Interacts with a libration point in that the bary center can be selected as the primary or secondary body for calculating the libration point.

Using the GUI

- Primary Body
  - This is the main body of the system, picked from the list of celestial bodies.
- Secondary Body
  - This is the second body that will be used to perform calculations in the system for the finding of the libration point.
- Libration Point
  - The other option is to select which libration point is wanted. There are a total of 5 libration points. The first libration point is between the two bodies. The second libration point is in a line from the primary body, through the secondary body, to the libration point. The third libration point is in a line from the secondary body through the primary body, out to the libration point. The fourth and fifth libration points are nearly on the orbit path of the secondary body, placed symmetrically. For more information, see http://en.wikipedia.org/wiki/Libration_point .

Figure: Default Name and Settings for the Libration Point object Dialog Box

Using the Script

Script Syntax

```
Create Libration Point;
.Primary = ;
.Secondary = ;
.Point = <'L1', 'L2', 'L3', 'L4', 'L5'>
```

Sample Script

```
Create LibrationPoint Libration1;
GMAT Libration1.Primary = Sun;
GMAT Libration1.Secondary = Earth;
GMAT Libration1.Point = 'L1';
```

Reference Table

Primary	Default: Sun. Options: [Sun, Mercury, Venus, Earth, Luna, Mars, Jupiter, Saturn, Uranus, Neptune, Pluto], or any Barycenter. (The Primary and Secondary bodies cannot be the same)] The Primary field tells GMAT which body to consider the primary body in the calculation of the location of a libration point. Units: N/A.
Secondary	Default: Earth.

	Options: [Sun, Mercury, Venus, Earth, Luna, Mars, Jupiter, Saturn, Uranus, Neptune, Pluto, or any Barycenter. (The Primary and Secondary bodies cannot be the same)] The Secondary field tells GMAT which body to consider the secondary body in the calculation of the location of a libration point. Units: N/A.
Point	Default: L1. Options: [L1, L2, L3, L4, L5] The Point field specifies which libration point the object corresponds to. Units: N/A.

Impulsive Burn

This tutorial was started from the default settings in the June 9, 2008 build of GMAT.

Overview

The impulsive burn object in GMAT allows the spacecraft to undergo an instantaneous ?V in up to three dimensions as opposed to a finite burn which is not instantaneous. The user can configure the burn by defining its origin, type of axes, vector format, and magnitude of the vectors. Depending on the mission, it will be simpler to use one axes or vector format over the other.

Possible Coupling with Other Objects

- Spacecraft
  - Must be created in order to apply any burn. The purpose of the impulsive burn is to instantaneously propel the spacecraft to either target or optimize a goal during its mission.
- Maneuver command
  - Must be created to call the burn into the mission sequence because without a maneuver, the spacecraft simply propagates around a specified trajectory. If there are several burns that exist, in the Maneuver dialog box the user can choose which burn to utilize for that part of the mission sequence. In addition, a Propagate command must follow the maneuver to allow the trajectory to unfold after a burn has been applied.
- Vary command
  - Required a burn to be specified in the Variable Setup group box. The purpose of the Vary command is to apply a burn in order to change a parameter of the spacecraft's trajectory.

Using the GUI

The Impulsive Burn dialog box contains three drop down menus and one group box to set the magnitude of the burn in three dimensions. Setting the first three menus to any property installs a tentative coordinate

system about that object without having to add one under the Coordinate Systems folder in the resource tree.

- Origin - Field
  - The parameter for the origin can be any celestial body or other spacecraft. Selecting the spacecraft that is experiencing the burn will return a null set since the position would always be zero in all directions.
- Axes - Field
  - Depending on the mission, two types of axes are available. The MJ2000Eq option applies a Cartesian approach where the burn is specified in an XYZ format, while the VNB option utilizes the burn as specified in the velocity, normal, and binormal directions, the velocity direction being positive X. Notice when the axes type is changed, the vector magnitude fields will change to correspond with it.
- Vector Format - Field
  - This field only supplies a Cartesian option for the vector to maintain it in an XYZ or VNB format.
- Vector - Group Box
  - This group box allows magnitudes to be entered for each of the three dimensions of the burn. As mentioned above, by changing the Axes field from MJ2000Eq to VNB, the group box will change accordingly to match up with the axes chosen. Also, the VNB axes requires additional information at the start of the mission. The main issue with VNB Axes is that it is impossible to locate a spacecraft with just that information. Some of the angles (RAAN, azimuth, TA, etc.) must be known to adequately specify a location, but the system can be advantageous if dealing with a spinning axis on the spacecraft, for example.

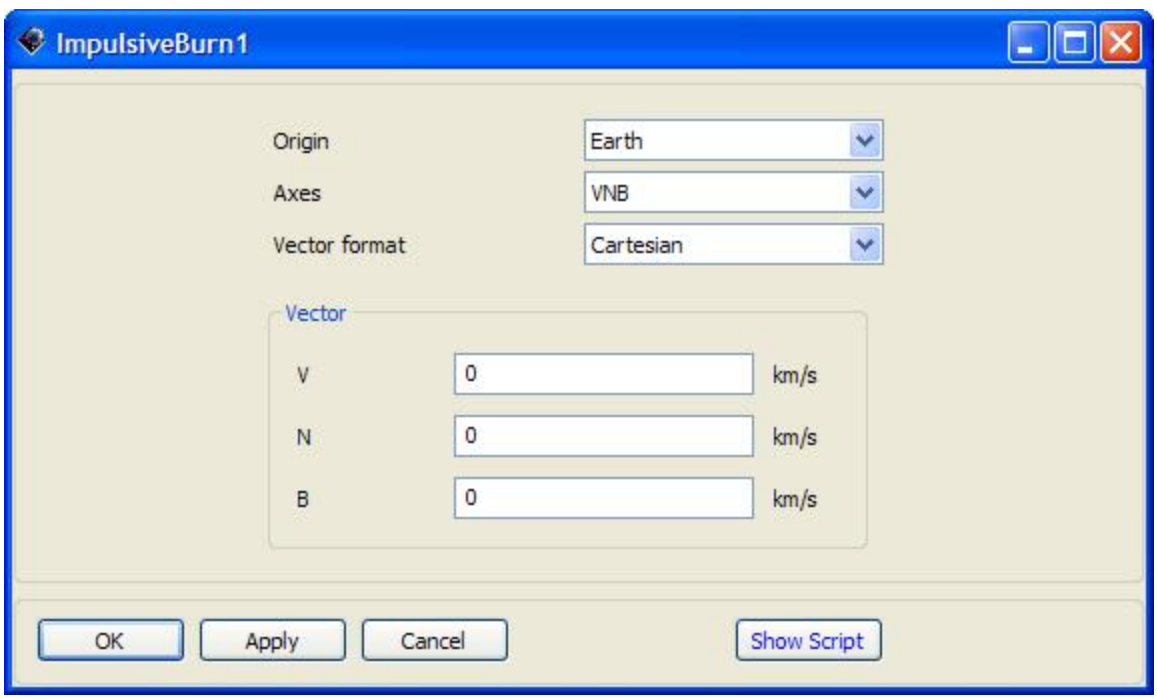


Figure: Impulsive Burn Dialog Box

Using the Script

Script Syntax

```
Create ImpulsiveBurn ImpulsiveBurn1;
GMAT ImpulsiveBurn1.Origin = Earth;
GMAT ImpulsiveBurn1.Axes = VNB;
GMAT ImpulsiveBurn1.VectorFormat = Cartesian;
GMAT ImpulsiveBurn1.Element1 = 0;
GMAT ImpulsiveBurn1.Element2 = 0;
GMAT ImpulsiveBurn1.Element3 = 0;
```

Reference Table

Origin	Default: Earth. Options: { Any celestial body } Together the Origin and Axes fields describe the coordinate system in which a maneuver is applied. The Origin field determines the origin of the maneuver coordinatesystem. The ability to define the coordinate system locally avoids having to create many coordinate systems, associated with specific spacecraft, in order to perform finite maneuvers for multiple spacecraft. Units: N/A.
Axes	Default: VNB. Options: { VNB,MJ2000Eq } The Axes field, together with the Origin field, describethe coordinate system in which an impulsive maneuver is applied. If VNB is chosen for Axes, a local coordinate system is created such that the x-axis points in the velocity direction of thespacecraft, with respect to the point defined by Origin, the y-axis points in the normal direction of the spacecraft with respect to Origin, and the z-axis completes the right-handed set. Units: N/A.
VectorFormat	Default: Cartesian. Options: { Cartesian } The VectorFormat field allows the user to define the format of the maneuver vector. Units: N/A.
Element1	Default: 0.

	Options: { Real Number } The Element1 field allows the user to define the first element of the impulsive maneuver vector. Element1 is X if VectorFormat is Cartesian. Units: km/sec.
Element2	Default: 0. Options: { Real Number } The Element2 field allows the user to define the second element of the impulsive maneuver vector. Element2 is Y if VectorFormat is Cartesian. Units: km/sec.
Element3	Default: 0. Options: { Real Number } The Element3 field allows the user to define the third element of the impulsive maneuver vector. Element3 is Z if VectorFormat is Cartesian. Units: km/sec.

Chapter 25. Commands

Propagate

This was created using the June 9, 2008 build of GMAT.

Overview of the Propagate Command

The Propagate Command is a very important one and will be covered in this section. Basically, Propagate will take the given spacecraft and, using the Propagat"or" specified as its guide, make it travel until the given condition is met whether it be a location or something else such as elapsed time.

Listing of Objects and Commands that interact

- Spacecraft
  - It is pretty intuitive that you would need to choose a spacecraft to propagate in order to use the command. Basically, once you create a spacecraft in the resource tree, it will appear as a spacecraft that you can propagate using this command. Note that you can propagate several spacecraft using the same Propagate command.
- Formation
  - The same way that you can propagate a Spacecraft, you can propagate a group of spacecraft that are assigned to a Formation.
- Propagator
  - Just as the propagate command needs a spacecraft to propagate, it also needs a propagator that has the parameters of propagation. This basically means that you must setup a propagator that governs the spacecraft as it travels through space. This includes what gravity is affecting it, as well as drag and solar radiation pressure.
- Parameters
  - Whatever parameters of object type Spacecraft, Variable, or Array, that are present within the Parameter Select Dialog box, are available to this field.

Using the Propagate Command in the GUI

The Propagate Command dialog box is split into two sections. The first of these is the Propagators and Spacecraft Group Box. It is where you specify which Propagator the Propagate Command will use. It is also where you specify which spacecraft to propagate. The next Groups Box is the Stopping Conditions. This Box is where you set the condition that will end the propagation of the spacecraft.

- Propagators and Spacecraft - Field
  - Propagate Mode - Drop Down Menu
    - This is where you set what mode you want the Propagator to use, None or Synchronized
  - Backwards Propagation - Check Box

    - Checking this box will make the spacecraft travel backwards until the given stopping condition is met. Examples of this would make a spacecraft will travel backwards in its orbit or travel backwards in time for a given period.
    - Backwards propagation can be useful if you know where a spacecraft must be at the end of its journey and you must see what possible path it can take to reach that objective.
  - Propagator - Field
    - This specifies the Propagator that the Propagate Command uses
  - Spacecraft List - Field
    - This is where the spacecraft that the Propagate Command is operating upon is specified.
    - Multiple Spacecraft may be entered here with their trajectories all being governed by the given propagator. If they are, they may be synchronized as given in the script example below.
    - Besides Spacecraft, entire formations may be entered here for propagation.
- Stopping Conditions - Field
  - Stop Tolerance - Field
    - This is how close you allow GMAT to be off from the given stopping condition. The smaller the number, the greater the precision.
  - Parameter - Field
    - This is what the Propagate Command will stop on. It can be set as a date, a location in orbit, or any other option given in the Parameter Select Dialog Box.
    - Multiple stopping conditions may be chosen. GMAT will stop propagation at the earliest met parameter.
  - Condition - Field
    - If the Parameter you set requires a value, such as a number of elapsed days, then you enter it here.

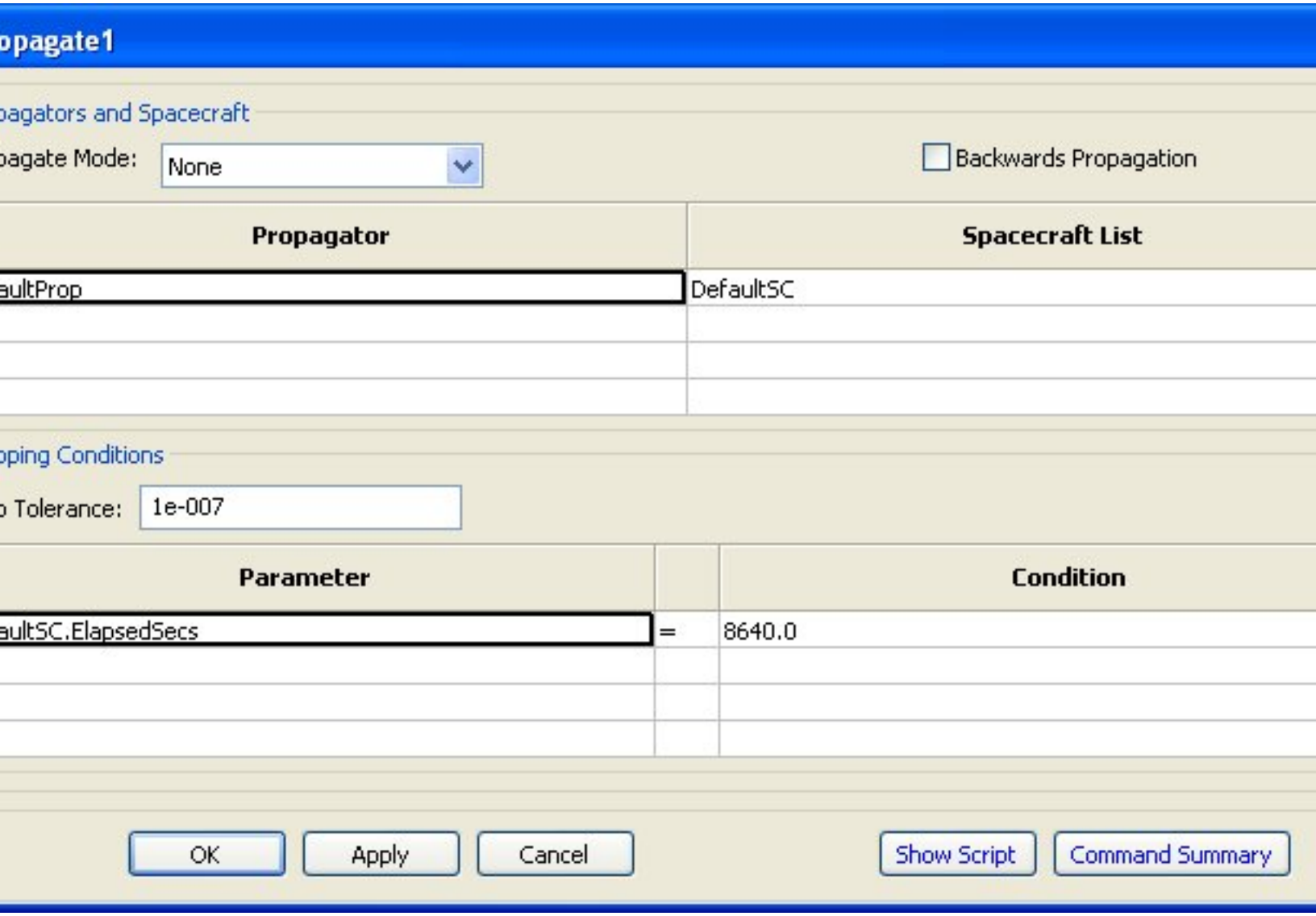


Figure: Default Name and Settings for the Propagate Command Dialog Box

Using the Propagate Command in the script

Script Syntax

```
Propagate Mode BackProp PropagatorName(SatList1,fStopCondList1g) ...
BackPropPropagatorName (SatListN, {StopCondList})
```

Script Examples

```
% Single spacecraft propagation with one stopping condition
% Syntax #1
Propagate DefaultProp(DefaultSC, {DefaultSC.ElapsedSecs = 8640.0});

% Single spacecraft propagation with one stopping condition
% Syntax #2
```

```
Propagate DefaultProp(DefaultSC) {DefaultSC.ElapsedSecs = 8640.0};

% Single spacecraft propagation by one integration step
Propagate DefaultProp(DefaultSC);

% Multiple spacecraft propagation by one integration step
Propagate DefaultProp(Sat1, Sat2, Sat3);

% Single formation propagation by one integration step
Propagate DefaultProp(DefaultFormation);

% Single spacecraft backwards propagation by one integration step
Propagate Backwards DefaultProp(DefaultSC);

% Two spacecraft synchronized propagation with one stopping condition
Propagate Synchronized DefaultProp(Sat1, Sat2, {DefaultSC.ElapsedSecs = 8640.0});

% Multiple spacecraft propagation with multiple stopping conditions and propagatio
% Syntax #1
Propagate Prop1(Sat1,Sat2, {Sat1.ElapsedSecs = 8640.0, Sat2.MA = 90}) ...
Prop2(Sat3, {Sat3.TA = 0.0});

% Multiple spacecraft propagation with multiple stopping conditions and propagatio
% Syntax #2
Propagate Prop1(Sat1,Sat2) {Sat1.ElapsedSecs = 8640.0, Sat2.MA = 90} ...
Prop2(Sat3), {Sat3.TA = 0.0};
```

Reference Table

BackProp	Default: None Options: { Backwards or None } The BackProp option allows the user to set the flag to enable or disable backwards propagation for all spacecraft in the SatListN option. The Backward Propagation GUI check box field stores all the data in BackProp. A check indicates backward propagation is enabled and no check indicates forward propagation. In the script, BackProp can be the word Backwards for backward propagation or blank for forward propagation. Units: N/A
Mode	Default: None Options: { Synchronized or None } The Mode option allows the user to set the propagation mode for the propagator that will affect all of the spacecraft added to the SatListN option. For example, if synchronized is selected, all spacecraft are propagated at the same step size. The Propagate Mode GUI field stores all the data in

	Mode. In the script, Mode is left blank for the None option and the text of the other options available is used for their respective modes. Units: N/A
PropagatorName	Default: DefaultProp Options: { Default propagator or any user-defined propagator } The PropagatorName option allows the user to select a user defined propagator to use in spacecraft and/or formation propagation. The Propagator GUI field stores all the data in PropagatorName. Units: N/A
SatListN	Default: DefaultSC Options: { Any existing spacecraft or formations, not being propagated by another propagator in the same Propagate event. Multiple spacecraft must be expressed in a comma delimited list format. } The SatListN option allows the user to enter all the satellites and/or formations they want to propagate using the PropagatorName propagator settings. The Spacecraft List GUI field stores all the data in SatListN. Units: N/A
StopCondListN /Parameter	Default: DefaultSC.ElapsedSecs = Options: { Any single element user accessible spacecraft parameter followed by an equal sign } The StopCondListN option allows the user to enter all the parameters used for the propagator stopping condition. See the StopCondListN/ Condition Option/Field for additional details to the StopCondListN option. Units: N/A
StopCondListN /Condition	Default: 8640.0 Options: { Real Number, Array element, Variable, spacecraft parameter, or any user defined parameter }. The StopCondListN option allows the user to enter the propagator stopping condition's value for the StopCondListN Parameter field. Units: Dependant on the condition selected.

While

This tutorial was started from the default settings in the June 18, 2008 build of GMAT.

Overview

The while loop is a control logic function that allows GMAT to check the spacecraft's status on a given parameter while performing a command or another control logic function within the mission sequence. When a spacecraft has reached the given property, the while loop will check its condition and react according to the equation defined in the loop's dialog box.

Listing of Objects and Commands that Interact

- Spacecraft
  - On either side of the equation and inside the loop's Parameter Select Dialog box, an Object Type must be called from the drop down menu to reference for the loop to function properly. Available options include a spacecraft, variables, arrays, or strings.
- Variables/Arrays/Strings
  - See Spacecraft bullet.
- Maneuver Command
  - As part of the while loop definition, the sequence must be performing a command or include an internal control logic function while something else is happening. Typical commands include Maneuver and Propagate and any control function, If, If-Else, For, or While, is possible.
- Propagate Command
  - See Maneuver Command bullet
- Control Logic Functions
  - See Maneuver Command bullet

Using GUI

The While Loop dialog box contains four sections, a left hand (LHS) and right hand (RHS) side of the equation, a logical operator column and a condition column. The LHS and RHS supply the parameters to put on either side of the condition for the equation, and the logical operators exist so more complex loops can be run through GMAT.

- LHS/RHS - Field
  - Both the LHS and RHS columns have the capability of either entering a value or selecting a property from a second dialog box, but typically the LHS is a property and the RHS is a value. You can manually add a value by applying a double left click, or you can open the Parameter Select dialog box by applying a right click. In addition to selecting a property, an Object Type and an object from the Object List must be highlighted. If a property is selected that requires a coordinate system, then a system must be chosen from the drop down menu that appears below the Object Properties menu.
- Condition - Field
  - The condition column supplies the iequality symbols (<, >, <=, >=, ==, and ~=) for the expression.

- Logical Operator - Field
  - The logical operator column only supplies one option, the ability to right click the box to add an And or Or operator (& or |). Except for the uppermost box in the far left column that states the loop or statement being used, the rest of the boxes are open to add operators.

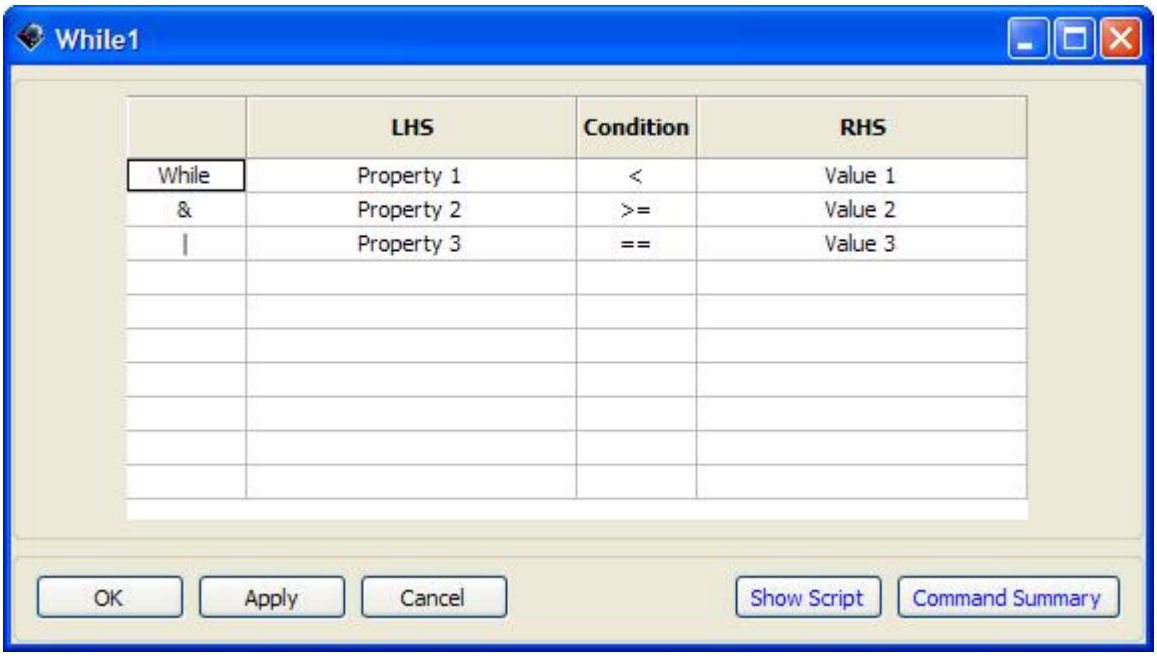


Figure: Default Name and Settings for the While Loop Dialog Box

Using the Script

Script Syntax

- Simple While Loop

```
While <logical expression>;
    <Statements>;
EndWhile;
```

- Compound While Loop

```
While <logical expression> | <logical expression> & <logical expression>;
    <Statements>
EndWhile;
```

Script Examples

```
While DefaultSC.ElapsedDays < 1;
Propagate DefaultProp (DefaultSC , DefaultSC.Elapsed Days = 0.01);
EndWhile;

While MyVariable < MyArray(1,1);
MyArray(1,1) = 5;
EndWhile;
```

Reference Table

<logical expression>	Default: DefaultSC.ElapsedDays < 1.0. Options: { Arg1 < Arg2 and < can be > , < , >= , <= , == , ~= }. Arg1 and Arg2 can be any of the following: Real Number, Array, Variable, Spacecraft Parameter or any other user defined parameter. Units: N/A.
<Statements>	Default: N/A. Options: { Any script line that can be in the mission sequence }. Units: N/A.
|	Default: N/A. Options: { N/A }. The | option allows the user to set an OR operator in between s. Units: N/A.
&	Default: N/A. Options: { N/A }. The & option allows the user to set an AND operator in between <logical expression>s. Units: N/A.

If

This was created using the Default settings in the June 9, 2008 build of GMAT.

Overview

The If command gives you the ability to use a logical statement within GMAT. At some point during a mission sequence, when a particular command should only be executed when a certain condition is met, use of the If command is recommended. The If command also gives you the ability to make a command's execution reliant upon multiple conditions.

Listing of Objects and Commands that interact

- Parameters
  - Whatever parameters of object type Spacecraft, Variable, or Array, that are present within the Parameter Select Dialog box, are available to this field. Numerical values are also available.

Using the GUI

The If Command Dialog box looks similar to the other control logic Dialog boxes. Basically, you may change the parameters on the left hand side (LHS) and right hand side (RHS) of the logical expression, as well as the condition operator between them. This means that the expression can compare nearly any values from the mission you wish. In addition to this there is the ability to have the If statement dependent upon more than one condition. Or and And operators, along with their accompanying logical statements, may be added under the initial If to further refine the conditions that must be met.

- LHS/RHS - Field
  - These fields can either be set to a numerical value or whatever object value is available in the parameter select dialog box. The objects available in that dialog box depend upon what resources have been created in the mission tree.
- Condition - Field
  - This may be set as equal, not equal, greater/less than, and greater/less than or equal to. This comparison applies to the LHS and RHS on the same line as it.
- Logic Selection - Field
  - Selecting And/Or here will allow you to use another logical expression to better create your If statement's condition. Multiple lines may be added to the If statement in this way.

Figure: Default Name and Settings for the If Command Dialog Box

Using the script

Using the If command in the script is quite simple. If you have ever programmed before in the higher level languages such as C, Matlab, or Java, GMAT will be very familiar. The statement reads like you see it basically: If the given statement after the 'If' is true, then execute the statement(s) following until the 'EndIf' is reached.

Script Syntax

- Simple If Statement

```
If <logical expression>;
   <Statements>;
EndIf;
```

- Compound If statement

```
If <logical expression> | <logical expression> & <logical expression>;
   <Statements>;
EndIf;
```

Script Examples

```
If DefaultSC.ElapsedDays < 1;
   Propagate DefaultProp( DefaultSC , { DefaultSC.ElapsedDays = 0.01 });
EndIf;

If MyVariable < MyArray(1,1);
   MyArray(1,1) = 5;
EndIf;

If DefaultSC.Earth.TA < MyArray(1,2);
   Propagate DefaultProp( DefaultSC );
EndIf;
```

Reference Table

<If Command>	Default: DefaultSC.ElapsedDays < 1.0. Options: { Arg1 < Arg2 and < can be >, <, >=, <=, ==, ~126~= } Arg1 and Arg2 can be any of the following: Real Number, Array element, Variable, Spacecraft Parameter or any other user defined parameter. Units: N/A.
<Statements>	Default: N/A. Options: { Any script line that can be in the mission sequence } Units: N/A.
|	Default: N/A.

	Options: { N/A } The | option allows the user to set an OR operator in between <logical expression>s. Units: N/A.
&	Default: N/A. Options: { N/A } The & option allows the user to set an AND operator in between <logical expression>s Units: N/A.

Target

This was created using the Default settings in the June 16, 2008 build of GMAT.

Overview

When building a mission, the targeting sequence is one of the most useful tools you have at your disposal. It allows you to use the power of a differential corrector to find solutions to complex two and three-body problems with a limited number of known values and several variables. Target is the base command that is appended with more commands such as Vary, Maneuver, Propagate, and Achieve in order to meet your goals. This makes it the first part of the sequence to be put into the Mission tree.

Listing of Objects and Commands that interact

- Boundary Value Solvers
  - In the Solver Name drop down menu you select which Differential Corrector you want to use for the targeting sequence. Any Corrector that you add to the Boundary Value Solvers sub menu in the Resource tree will be added here. If you don't provide one before adding the Target command to Mission tree then GMAT will add one for you and name it DefaultDC.
- Others
  - Though there is no direct connection within the GUI window to any other commands, Target is one of the few commands that you can 'Append'. When you Append a Target, you are basically adding a sequence of commands to be executed, in turn, within the targeting sequence. In fact, these commands make up the targeting sequence itself. A simple sequence may append Target with Vary, Maneuver, and Achieve. These will allow Target to try and achieve a certain value given in that command by changing and solving for a value given in the Vary command. The Maneuver is when the Impulsive Burn would be executed in the targeting sequence.

Using the GUI

The GUI window is relatively straightforward and has a few options that you may change.

- Solver Name - Drop Down Menus
  - This is where you choose which Solver you would like the Target command to use.

- Solver Mode - Drop Down Menus
  - Solve mode will be used most of the time from this menu because it tells the targeter to search for a solution to satisfy the goals that you have laid out for it.
  - Use of the RunInitialGuess mode will simply mean that no targeting is performed and the provided initial values are run once but no more.
- Exit Mode - Drop Down Menus
  - DiscardAndContinue will scrub the values that the targeter achieves after it has calculated them. This way, any subsequent calls to that same targeting sequence will start with fresh initial values and not those from the calculations of the previous run.
  - SaveAndContinue is similar to DiscardAndContinue except that the values that the targeter finds from its calculations will be saved and used as the initial values for any subsequent runs of the this particular targeting sequence.
  - Setting this menu to Stop will simply stop the mission after the targeting sequence is finished its calculations.
- Apply Corrections - Button
  - Once you have run the mission at least once, you may press this button. Apply Corrections will replace the initial values provided by either you or the default settings in the mission sequence with the ones calculated by the targeting sequence in its latest run. You can use this feature instead of having multiple targeting sequences where one sequence uses the calculated values from the previous one.

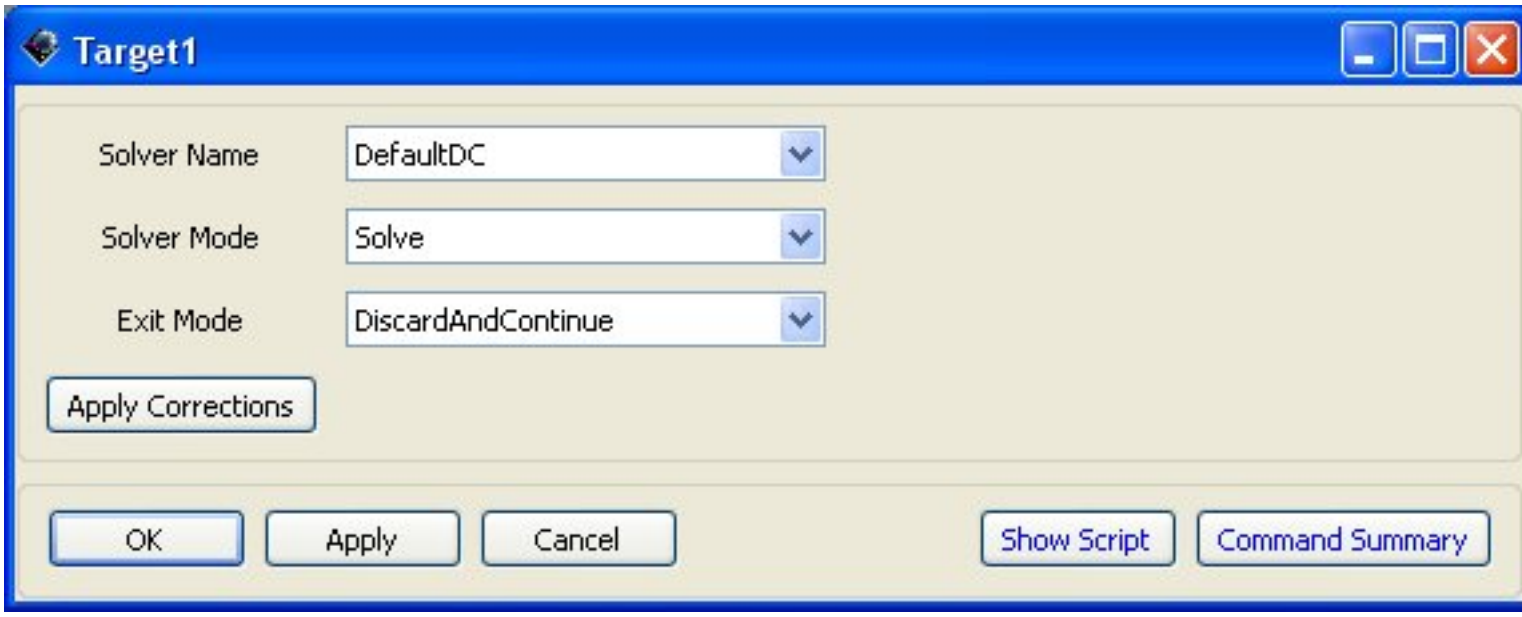


Figure: Default Name and Settings for the Target Command Dialog Box

Using the script

Script Syntax

```
Target SolverName;
```

```
     <Statements>
EndTarget;
```

Script Examples

```
% Beginning and ending syntax for the Target command
Target DefaultDC;

EndTarget;

Target DefaultDC {SolveMode = RunInitialGuess, ExitMode = SaveAndContinue};
     <Statements>
EndTarget;  % For targeter DefaultDC
```

Reference Table

SolverName	Default: DefaultDC Options: { Any differential corrector existing in the resource tree or created in the script } The SolverName option allows the user to choose between any previously created differential correctors for use in a targeting sequence. For example, to begin a targeting sequence using DefaultDC, the script is Target DefaultDC. Units: N/A
<Statements>	Default: None Options: { Any non-targeter and non-optimizer command lines used in the mission sequence, as well as the targeter dependent command lines Achieve and Vary } Units: N/A
SolveMode	Default: Solve Options: { Solve , RunInitialGuess } Tells the targeter state machine how to manage the targeter loop. When running in the RunInitialGuess mode, the target loop is executed one time using the initial values of the targeter variables. No targeting is performed. When running in Solve mode, the targeter searches for a solution satisfying the targeter goals. Units: N/A
ExitMode	Default: DiscardAndContinue Options: { DiscardAndContinue , SaveAndContinue , Stop }

	Tells the Mission Control Sequence how to proceed after the targeter loop has finished running. When running in DiscardAndContinue mode, subsequent calls to the same targeter loop will reset the variables to their initial values. If running in SaveAndContinue mode, the targeter variables used on a subsequent run start with the values obtained the last time the targeter loop was run. In Stop mode, the mission run halts when the targeter loop completes its work. Units: N/A

Report

This was created using the June 9, 2008 build of GMAT.

Overview

The report command allows the user find parameters of the orbit and the spacecraft at particular moments in time. This command is inserted into the mission tree at various locations in the mission tree. The parameters found by this command are placed into a report file that can be accessed at a later time.

Listing of Objects and Commands that interact

- ReportFile
  - Object where the parameters retrieved by the report command.

Using the GUI

The report command has two main boxes. The first box allows one to select which report the parameters will be stored in. The second box interacts with the ParameterSelectDialog and allows the user to pick which parameters are desired to be seen at that particular moment.

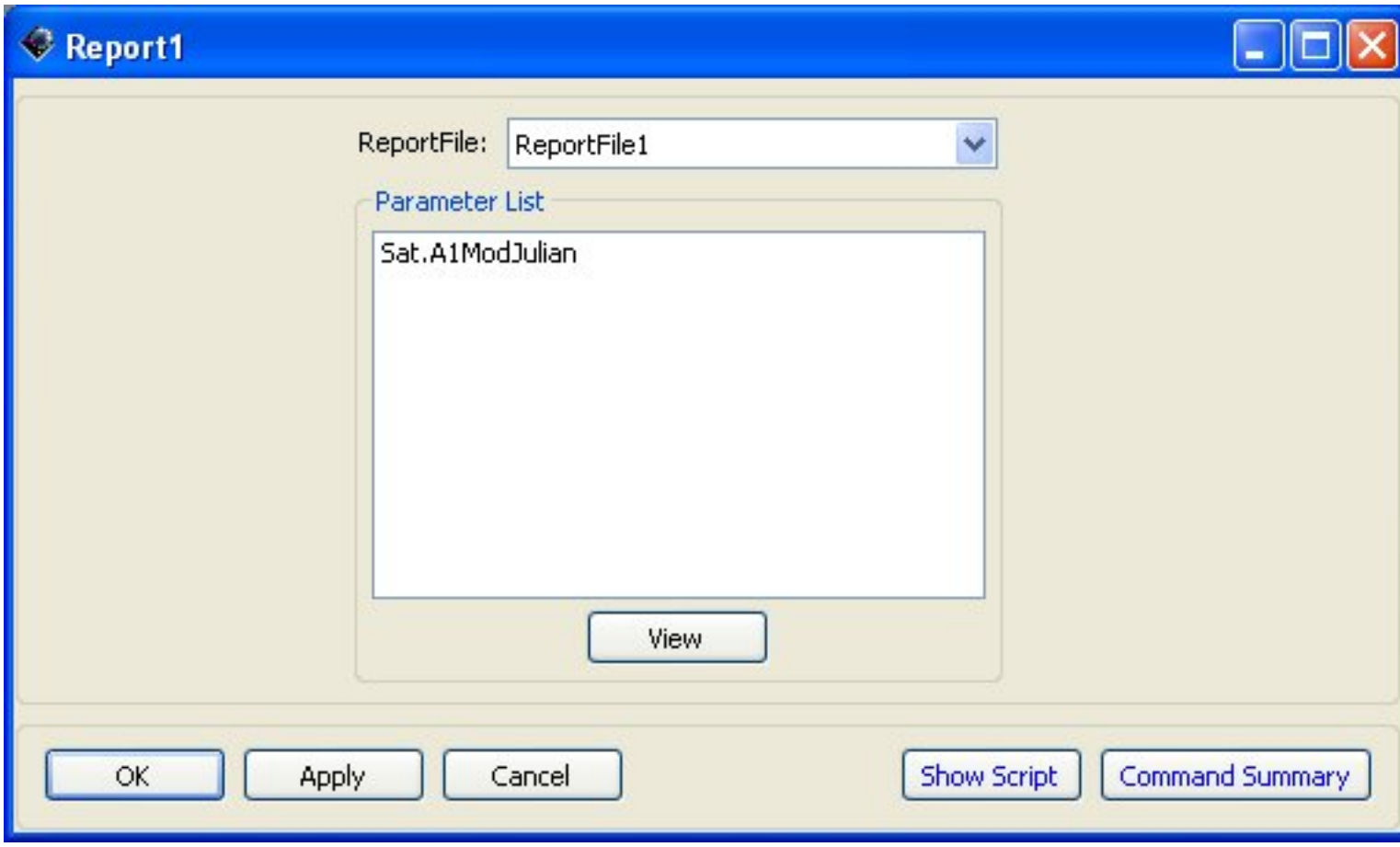


Figure: Default Name and Settings for the Report Command Dialog Box

Using the script

Script Syntax

```
Report  ReportName  DataList
```

Script Examples

```
%Report the time and position of DefaultSC

Report DefaultReport DefaultSC.A1ModJulian DefaultSC.X DefaultSC.Y DefaultSC.Z;
```

Reference Table

ReportName	Default: N/A Options: { Any ReportFile created } The ReportName option allows the user to specift the ReportFile for data output. Units: N/A

DataList	Default: N/A Options: { Spacecraft parameter, Array, Variable, String, or any other single user defined parameter } The DataList option allows the user to output data to the Filename specified by the ReportName. Multiple objects can be in the DataList when they are separated by spaces. Units: N/A

For

This was created using the June 9, 2008 build of GMAT.

Overview

The for loop is a control flow statement that allows portions of code to be executed iteratively using an explicit loop variable (Wikipedia). GMAT for loops are three-expression loops that allow the user to set the initial value of the loop variable, its increment, and the test to exit the loop. A parameter must be defined explicitly using a Create Variable statement or GUI equivalent before it can be used in a for loop command statement. The parameters used to define Start, Increment, and End can be any of the following GMAT parameters: numeric literal (real number), variable, array element, object property.

Interfaces

The GUI for the For Loop command is divided into four sections.

- The first section, the index, is where the counter variable name is displayed.
- The second section, the start, is the number with which the counter variable is first stored with.
- The third section, the increment, is the value that the counter variable will change by each time the program goes through the loop.
- The fourth section, the end, is the value of the counter variable when the loop is exited.

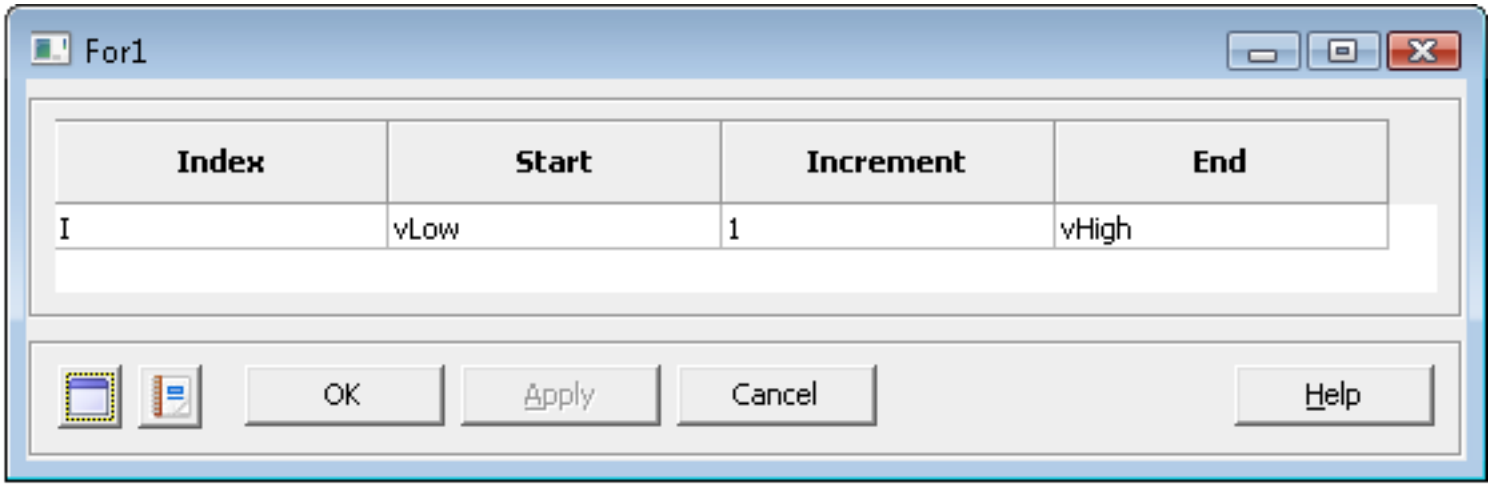


Figure: Default Name and Settings for the For Loop command Dialog Box

Script Syntax

(Simple For Loop)

```
For Variable = Start:End;

EndFor; }
```

(Expanded For Loop)

```
For Variable = Start:Increment:End;

EndFor;
```

Script Examples

```
% Output the value of the For loop Variable to a file
For I = 1:1:10;
GMAT testVar = I;
Report DefaultReportFile I;
EndFor;
```

Reference Table

Variable	Default: No Default. Options:[Variable] The Variable field allows the user to define the variable that to be incremented during the loop process. Units: N/A.
Start	Default: 1. Options: [Real Number, Array element, Variable, or any user defined parameter]. The Start option allows the user to set the starting value of the For Loop. Units: N/A.
Increment	Default: 1. Options: [Real Number, Array element, Variable, or any user defined parameter]. The Increment option allows the user to set the increment value of the For Loop. When the increment value is not included in a for loop statement, the default value is used.

	Units: N/A.
End	Default: 10. Options: [Real Number, Array, Variable, or any user defined parameter]. The End option allows the user to set the ending value of the For Loop. Units: N/A.

Objects and Command Interactions

For Loop Behavior

When the Increment option is left out of the script syntax the default value is used. If an Increment value of 0 is used, the For Loop should not execute but GMAT should continue to run. If End>Start and Increment<0, then the For Loop should not execute. If Start>End and Increment>0, then the For Loop should not execute. End can be equal to Start, but the For Loop will not execute.

Maneuver

This was created using the 6-8-2008 build of GMAT.

Overview

The Maneuver command is placed in the mission tree and applies a selected impulsive burn to a selected spacecraft. A finite burn requires something else to be applied.

Listing of Objects and Commands that interact

- Spacecraft
  - Object which performs the maneuver.
- Impulsive Burn
  - Item that the Maneuver command applies to the Spacecraft.

Using the GUI

There are two boxes. One box is there to select which impulsive burn the user would like to apply to a spacecraft. The other box enables the user to select which spacecraft the impulsive burn will be applied to.

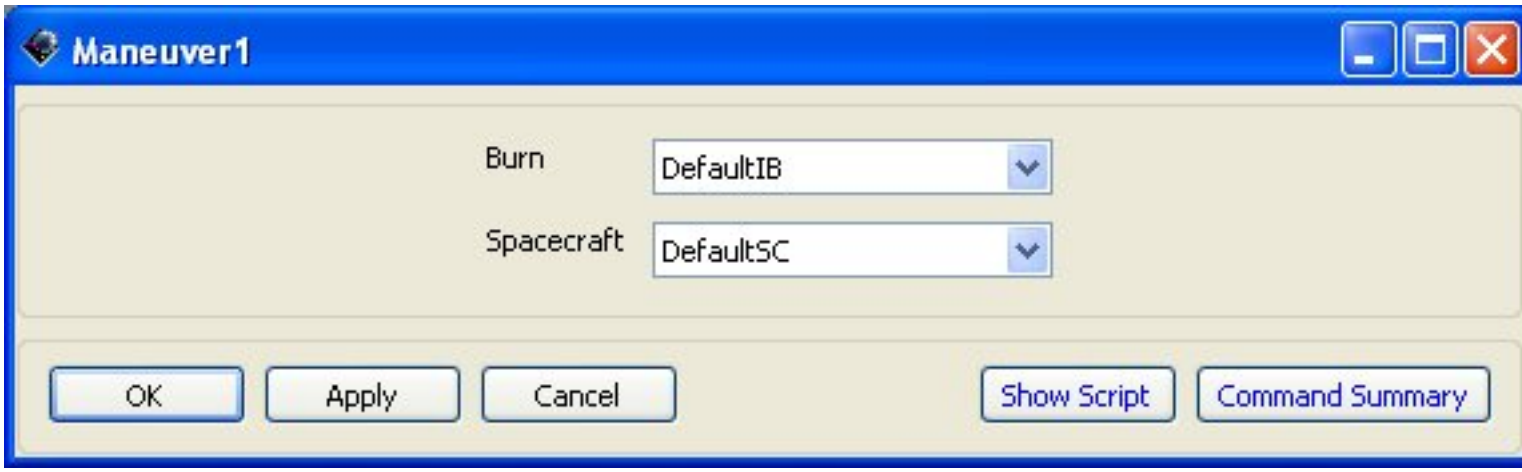


Figure: Default Name and Settings for the Maneuver Command Dialog Box

Using the script

Script Syntax

```
Maneuver BurnName (SpacecraftName);
```

Script Examples

```
% Impulsive Burn
Maneuver DefaultIB(DefaultSC);
```

Reference Table

BurnName	Default: DefaultIB. Options: [Any impulsive burn existing in the resource tree or created in the script] The BurnName field allows the user to choose between any previously created impulsive burn. As an example, to maneuver DefaultSC using DefaultIB, the script line would appear as Manevuer DefaultIB(DefaultSC). Units: N/A.
SpacecraftName	Default: DefaultSC. Options: [Any spacecraft existing in the resource tree or created in the script] The SpacecraftName field allows the user to select which spacecraft to maneuver using the maneuver selected in the BurnName field. Units: N/A.

Equation

This was created using the July 2, 2008 build of GMAT.

Overview

The Equation command uses the Equation to make one variable equal to some combination of previously defined variables and values. It is highly useful for storing values so that they aren't lost. Additionally, it is very useful for advanced commands.

Listing of Objects and Commands that interact

- ParameterSelectDialog
  - The standard box for Propagate and other commands, the box can select variables that can contain values which were established using the Assignment command. Additionally, the parameters listed can be changed using the Assignment command.
- Variable
  - Most often used in conjunction with the Assignment command, the variable object is set up to store values.
- Spacecraft
  - Can be used in conjunction with the Assignment command so that one spacecraft can take on the characteristics of another spacecraft. This technique is useful for storing conditions of a particular spacecraft before the values of the spacecraft are changed through Target sequences and Propagate commands.

Using the GUI

- Arg1 (left side of equal sign)
  - This is where the variable that will be taking on a new value is typed. It can never be a value, but generally will be a variable or spacecraft. If it is a spacecraft, this is the spacecraft that will be changing its old conditions.
- Arg2 (Right side of equal sign)
  - This is what the variable/spacecraft is set equal to whatever is placed in this box. The right side can contain variables and values. It can also contain any of the math operations listed in the Math Operators in Scripting under the Reference-Misc menu in GMAT Help.

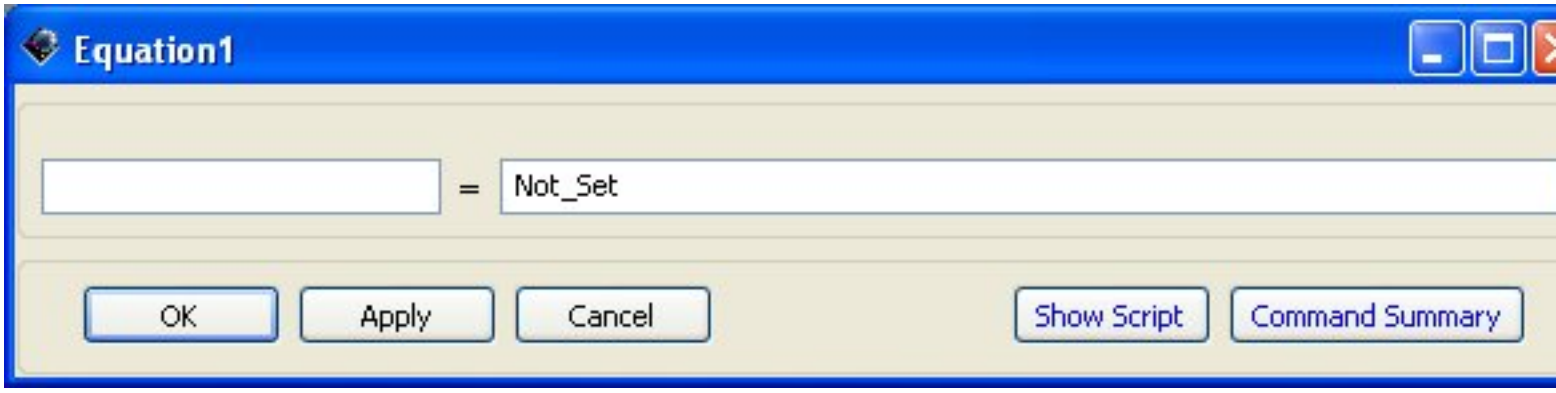


Figure: Default Name and Settings for the Equation Command Dialog Box

Using the script

Script Syntax

```
GMAT Arg1 = Arg2;
```

Script Examples

```
% Setting a variable to a number
 GMAT testVar = 24;
 % Setting a variable to the value of a math statement
 GMAT testVar = (testVar2 + 50)/2;
```

Reference Table

Arg1	Default: N/A. Options: [Spacecraft Parameter, Array element, Variable, or any other single element user defined parameter] The Arg1 option allows the user to set Arg1 to Arg2. Units: N/A.
Arg2	Default: N/A. Options: [Spacecraft Parameter, Array element, Variable, any other single element user defined parameter, or a combination of the aforementioned parameters using math operators] The Arg2 option allows the user to define Arg1. Units: N/A.

Toggle

This was created using the July 2,2008 build of GMAT.

Overview

The Toggle command is useful in turning on/off certain plots so that they do not become unintelligible. They are particularly useful if a mission requires multiple sequences to fine-tune the answer. If this is the case, it might be better to turn off a plot until the final or second-to-last sequence so that the rougher sequences don't obstruct the view of what is going on in the finesse sequences.

Listing of Objects and Commands that interact

- OpenGL plots, XY Plots, and Report File
  - These can all be turned on or off for durations of time so that data is not collected always, but rather only the data desired.

Using the GUI

- Subscribers to Toggle
  - Select the plots/reports that you want to turn on or off.
- On/Off radio button

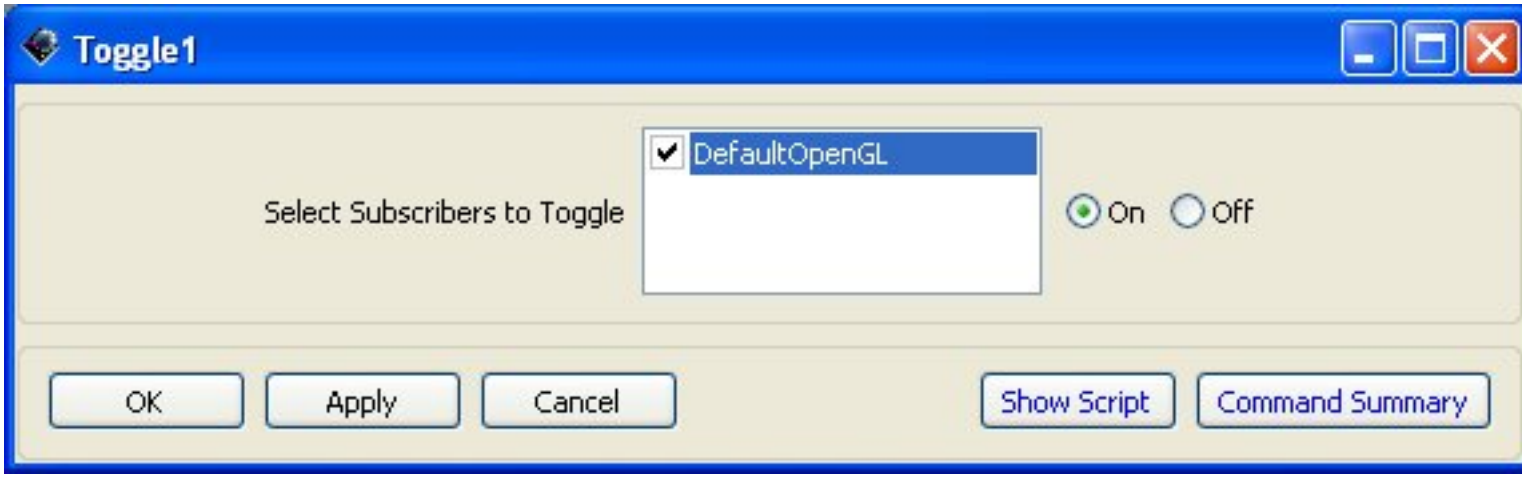


Figure: Default Name and Settings for the $NAME OF OBJECT OR COMMAND$ Dialog Box

Using the script

Script Syntax

```
Toggle OutputNames Arg
```

Script Examples

```
% Turn off Report file for the first day of propagation
Toggle ReportFile1 Off
Propagate DefaultProp(DefaultSC, {DefaultSC.ElapsedDays = 1});
Toggle ReportFile1 On
Propagate DefaultProp(DefaultSC, {DefaultSC.ElapsedDays = 1});
```

```
% Turn off XYPlot and Report file for the first day of propagation
Toggle XYPlot1 ReportFile1 Off
Propagate DefaultProp(DefaultSC, {DefaultSC.ElapsedDays = 1});

Toggle XYPlot1 ReportFile1 On
Propagate DefaultProp(DefaultSC, {DefaultSC.ElapsedDays = 1})
```

Reference Table

OutputNames	Default: DefaultOpenGL . Options:[Any OpenGL, Report, XYplot, or any other Plot/Report type] The Toggle option allows the user to assign the Plot/Report(s) to be toggled. When more than one Plot/Report is being toggled they need to be separated by a space. Units: N/A.
Arg	Default: On. Options:[On or Off] The Arg option allows the user to turn off or on the data output to a Plot/Report. Units: N/A.

Vary

This was created using the July 2, 2008 build of GMAT.

Overview

The Vary command is used in conjunction with the Target and Optimize Commands. The Vary command varies a particular parameter within the Target and Optimize Commands until certain conditions are met. It is a highly useful command for creating missions. Multiple Vary commands can be used within the same targeting or optimizing sequence.

Listing of Objects and Commands that interact

- Target and Achieve and Differential Corrector
  - These are all used in conjunction to achieve/target a particular goal or characteristics.
- Optimize and Minimize or Nonlinear Constraint and Optimizer
  - These are all used in conjunction to achieve certain conditions or minimize a certain thing while under several constraints.

Using the GUI

- Solver
  - This allows the user to choose which Solver to use to achieve the desired conditions.
- Variable Setup
  - This Menu box contains the information needed for the Vary command to vary the variables correctly, in addition to choosing which variable is desired to be changed. The initial guess for the variable is provided by the user here. Also, here the upper and lower bounds for the variable are given. Additionally, the maximum step size can be provided by the user. Also, the perturbation that will be used to assist in the calculations is also given by the user.

- Additive Scale Factor
- Multiplicative Scale Factor
  - This value will scale the maneuver burn strength. The burn values are all multiplied by this number.

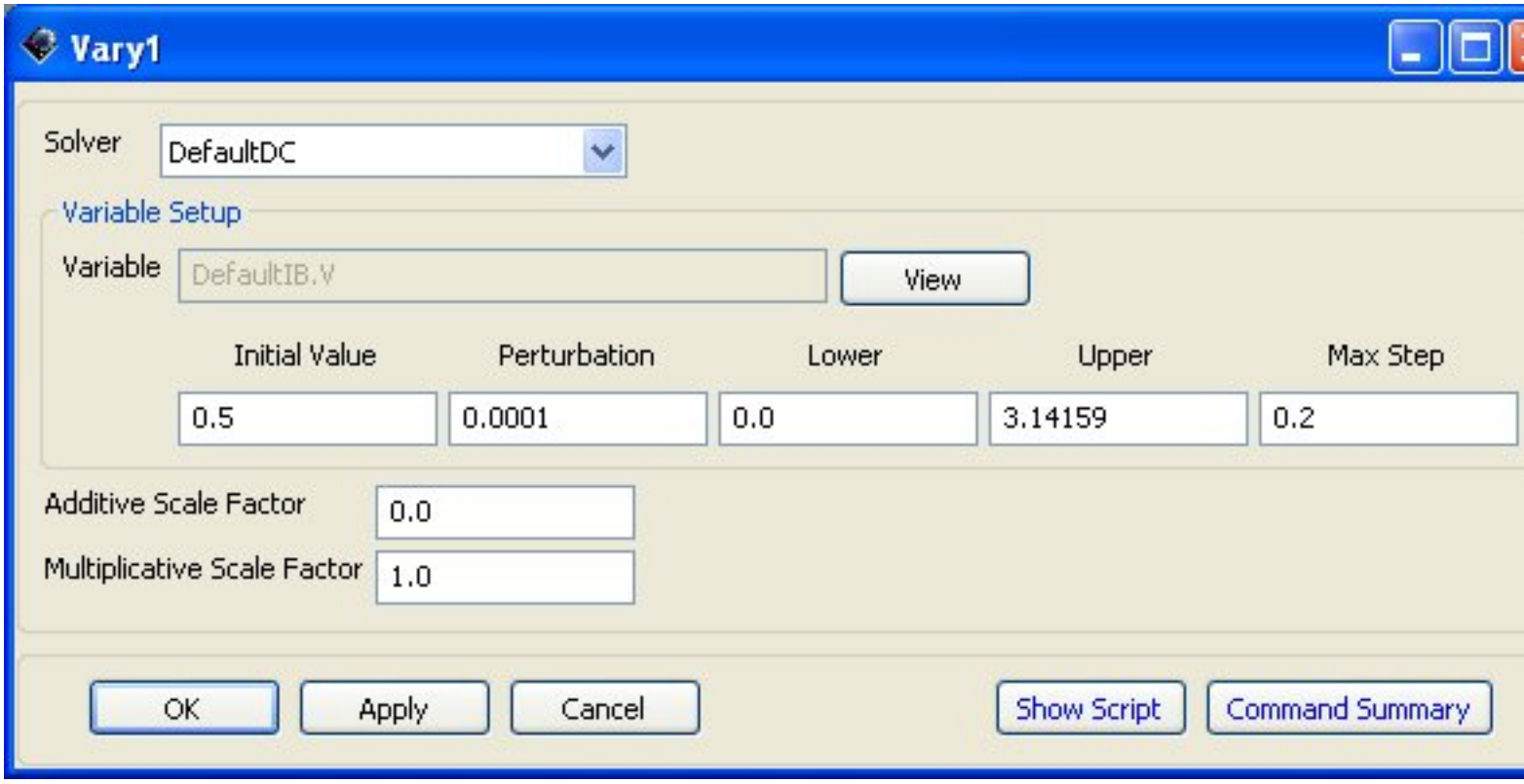


Figure: Default Name and Settings for the Vary Dialog Box

Using the script

Script Syntax

```
Vary SolverName(Variable=InitialGuess,{Perturbation=Arg1, MaxStep=Arg2,
  Lower=Arg3,...Upper=Arg4, AdditiveScalefactor=Arg6,MultiplicativeScalefactor=Arg
```

Script Examples

```
% Impulsive Burn Vary Command

Vary DefaultDC(DefaultIB.V = 0.5, {Perturbation = 0.0001, MaxStep = 0.2,
Lower = 0, Upper = 3.14159});
```

Reference Table

Parameters Associated with All Solvers

SolverName	Default: DefaultDC. Options: [Any user defined solver] the SolverName option allows the user to

	choose which solver to assign to the vary command. units: N/A.
Variable	Default: DefaultIB.V.Options: [Spacecrafy paremeter, Array element, Variable, or any other single element user defined parameter, excluding numbers] The Variable option allows the user to select any single element user defined parameter, except a number, to vary. For example, DefaultIB.V, DefaultIB.N, DefaultIB.Element1, DefaultSC.TA, Array(1,1), and Variable are all valid values. The three element burn vector or multidimensional Arrays are not valid values. Units: N/A.
InitialGuess	Default: 0.5. Options: [Real Number, Array element, Variable, or any user defined parameter that obeys the conditions for the selected Variable object] The InitialGuess option allows the user to set the initial guess for the selected Variable. Units: km/s.
Lower	Default: 0.0 . Options: [Real Number, Array element, Variable, or any user defined parameter (Upper > Lower)] The Lower option allows the user to set Arg3 to the lower bound of the quantity being varied. Units: N/A.
Upper	Default: 3.14159 . Options: [Real Number, Array element, Variable, or any user defined parameter (Upper > Lower)] The Upper option allows the user to set Arg4 to the upper bound of the quantity being varied. Units: N/A.

Parameters Associated with Differential Corrector

Perturbation	Default: 1e-4 . Options: [Real Number, Array element, Variable, or any user defined parameter > 0] The Perturbation option is set by specifying a value for Arg1. The value of Arg1 is the perturbation size in calculating the finite difference derivative. Units: N/A.
MaxStep	Default: 0.2 . Options: [Real Number, Array element, Variable, or any user defined parameter > 0] The MaxStep option is set by specifying a value for Arg2. The value of Arg2 limits the size of the step taken during an interaction of the differential corrector. Units: N/A.

Parameters Associated with fmincon Optimizer

Additive Scale Factor	Default: 0 . Options: [Real Number, Array element, Variable, or any user defined parameter] The AdditiveScaleFactor Field is used to nondimensionalize the independent variable. fmincon sees only the nondimensional form of the variable. The nondimensionalization is performed

	using the following equation: $x_n = (x_d - a)/m$. (x_n is the non-dimensional parameter. x_d is the dimensional parameter. a= additive scale factor. m= multiplicative scale factor.) Units: N/A.
Multiplicative Scale Factor}	Default: 1.0 . Options: [Real Number, Array element, Variable, or any user defined parameter] The MultiplicativeScaleFactor Field is used to nondimensionalize the independent variable. fmincon sees only the nondimensional form of the variable. The nondimensionalization is performed using the following equation: $x_n = (x_d - a)/m$. (x_n is the non-dimensional parameter. x_d is the dimensional parameter. a= additive scale factor. m= multiplicative scale factor.) Units: N/A.

Begin Finite Burn

This was created using the June 18, 2008 build of GMAT.

Overview

The Begin Finite Burn and End Finite Burn commands are very simple. When the Begin Finite Burn command is entered into the mission sequence it will initiate the thrusters of the spacecraft until the End Finite Burn command is reached. After the finite burn is turned off, the spacecraft's thrusters will shut down.

Listing of Objects and Commands that interact

- Spacecraft
  - A spacecraft must exist so the finite burn has an object to apply itself to, and a Begin Finite Burn command is how to do just that.
- Finite Burn
  - To apply a Begin Finite Burn command, a Finite Burn must exist in the resource tree under the Burns folder.

Using the GUI

In the dialog boxes of both the Begin and End commands there are two fields that incorporate a finite burn from the resource tree and any spacecraft created for the mission. All of the spacecraft are in a list alongside radio buttons and are checked or unchecked, depending on whether or not the burn should be applies to them.

- Apply - Drop Down Menu
  - This menu lists all of the finite burns created, and from this, one burn can be selected to be used on the spacecraft checked in the adjacent menu.
- To - Radio Button List
  - Here all of the spacecraft are listed and are accompanied by a radio button. If the box is checked, that spacecraft will experience the finite burn wherever it is set to occur in the mission sequence.

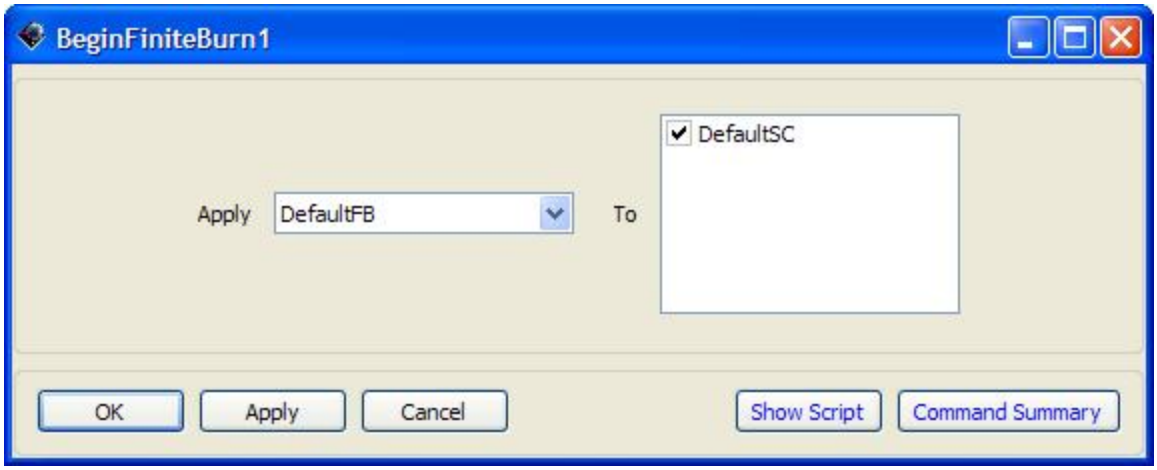


Figure: Default Name and Settings for the Begin Finite Burn Command Dialog Box

Using the script

Script Syntax

```
BeginFiniteBurn ManeuverName (SpacecraftName);
EndFiniteBurn DefaultFB(DefaultSC);
```

Reference Table

ManeuverName	Default: DefaultFB. Options: { Any finite burn existing in the resource tree or created in the script } The ManeuverName option allows the user to choose between any previously created finite burn. As an example, to maneuver DefaultSC using DefaultFB, the script linewould appear as Manevuer DefaultFB(DefaultSC). Units: N/A.
SpacecraftName	Default: DefaultSC. Options: { Any spacecraft existing in the resource tree or created in the script } The SpacecraftName option allows the user to select which spacecraft to maneuver using the maneuver selected in the ManeuverName option. Units: N/A.

Achieve

This was created using the June 18, 2008 build of GMAT.

Overview

The purpose of the Achieve command is to define a goal for the spacecraft to reach at some point in its trajectory. The goal must have a corresponding value and tolerance so the differential corrector can solve for the best solution during the spacecraft's flight. To define a goal, a property must be chosen out of the Parameter Select dialog box along with the correct components in the other fields. The command can only be appended within a targeting sequence and must be accompanied and preceeded by a Vary, Maneuver, and Propagate command.

Listing of Objects and Commands that interact

- Spacecraft
  - In order to achieve a goal at all, a spacecraft must exist so GMAT has something to propagate and maneuver.
- Differential Corrector
  - A differential corrector is needed to calculate and solve for the goal to be achieved.
- Vary Command
  - This command pulls variables into a targeting sequence for GMAT to know what parameters are being changed. As soon as a change occurs in a parameter, the differential corrector solves for the best trajectory using that data in order to achieve a goal.
- Maneuver Command
  - Changing any parameter of the spacecraft requires a Maneuver command and some sort of burn, so this, too, is an essential part of the Achieve command
- Burn
  - Changing parameters by means of maneuvering requires a burn to do so, and in this process the spacecraft is achieving a goal.
- Propagate Command
  - To achieve a goal, the spacecraft must be moving and, thus, be executing a Propagate command.

Using the GUI

The Achieve command dialog box has four fields to consider including a drop down menu for the Solver. By clicking on the View buttons, parameters from the Parameter Select dialog box can be chosen and entered in as a goal. Once a legitimate value and tolerance are entered for the parameter, the command is complete and is ready to run in the mission.

- Solver - Drop Down Menu
  - All of the differential correctors created are located in this menu to choose from for the Achieve command to utilize during operation.
- Goal - Field
  - Choosing a goal requires an object, parameter, and potentially a coordinate system to be highlighted in the Parameter Select dialog box. Once all are selected, the field will read (spacecraft).(coordinate

system).(parameter), depending on the need for a coordinate system. If a system is not required, the field will read (spacecraft).(parameter).

- Value - Field
  - A number is to be entered here that corresponds to the goal chosen. If the value does not make sense, the mission will run, but it will not converge. Also, depending on the situation, excess data points may be created and will produce the same outcome.
- Tolerance - Field
  - The tolerance tells GMAT how lenient the differential corrector should be when accepting solutions around the value entered in previous field.

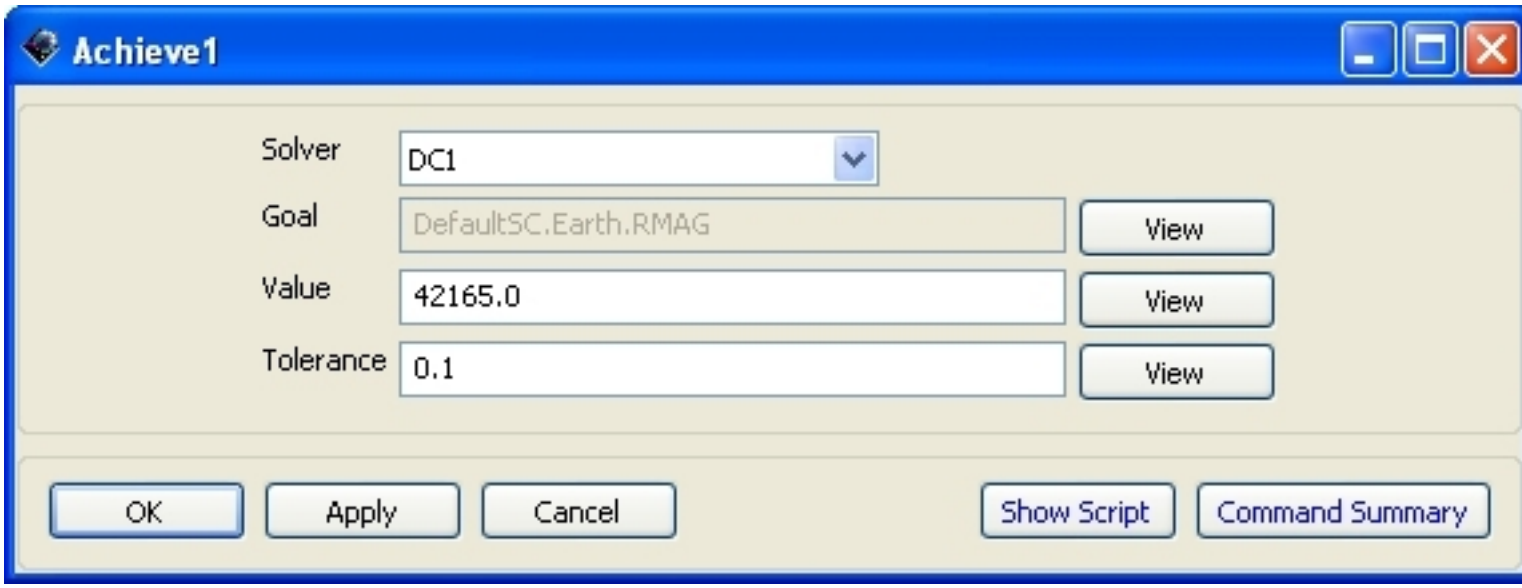


Figure: Default Name and Settings for the Achieve Command Dialog Box

Using the script

Script Syntax

```
Achieve SolverName(Goal) = Arg1, Tolerance = Arg2
```

Reference Table

Goal	Default: DefaultSC.Earth.RMAG. Options: { Spacecraft parameter, Array element, Variable, or any other single element user defined parameter, excluding numbers } The option allows the user to select any single element user defined parameter, except a number, to Achieve.

Arg1	Default: 42165. Options: { Real Number, Array element, Variable, or any user defined parameter that obeys the conditions of Chapter~\\ref \{Ch:ObjectsNResources\} for the selected } The Arg1 option is the desired value for after the solver has converged. Units: N/A.
Tolerance	Default: 0.1. Options: { Real Number, Array element, Variable, or any user defined parameter > 0 } The Tolerance option sets Arg2. Arg2 is the convergence tolerance for Arg1. Units: N/A.
SolverName	Default: DefaultDC. Options: { Any user defined differential corrector } The SolverName option allows the user to choose which solver to assign to the Achieve command. Units: N/A.

Stop

This was created using the June 9, 2008 build of GMAT.

Overview

The Stop command simply ends a mission in GMAT. Whether placed in the script or GUI, Stop will halt the execution of the mission and have GMAT report back that the Command Sequence was intentionally interrupted.

Listing of Objects and Commands that interact

- No Objects or Commands interact with Stop

Using the GUI

Since there is no GUI window for the Stop command, the screen shot below shows how it looks in the Mission tree.

Figure: Stop Command as it appears in the Mission Tree

Using the script

Script Syntax

```
Stop
```

Script Examples

```
% Stop between propagation sequences
Propagate DefaultProp(DefaultSC) DefaultSC.ElapsedSecs = 8640.0;
Stop;
Propagate DefaultProp(DefaultSC) DefaultSC.ElapsedDays = 10.0;
```

ScriptEvent

This was created using the June 9, 2008 build of GMAT.

Overview

The ScriptEvent command allows a user to enter in script within the GUI in the mission sequence. This can be useful in a number of ways if its easier to enter the script than to use the GUI interface. Also, if multiple things needed to be changed rapidly, this could be useful. Additionally, the ScriptEvent allows a user to have more freedom if the GUI is problematic and won't allow the user to perform the desired operation.

Using the GUI

- Field
  - You can type the script commands needed between BeginScript and EndScript;.

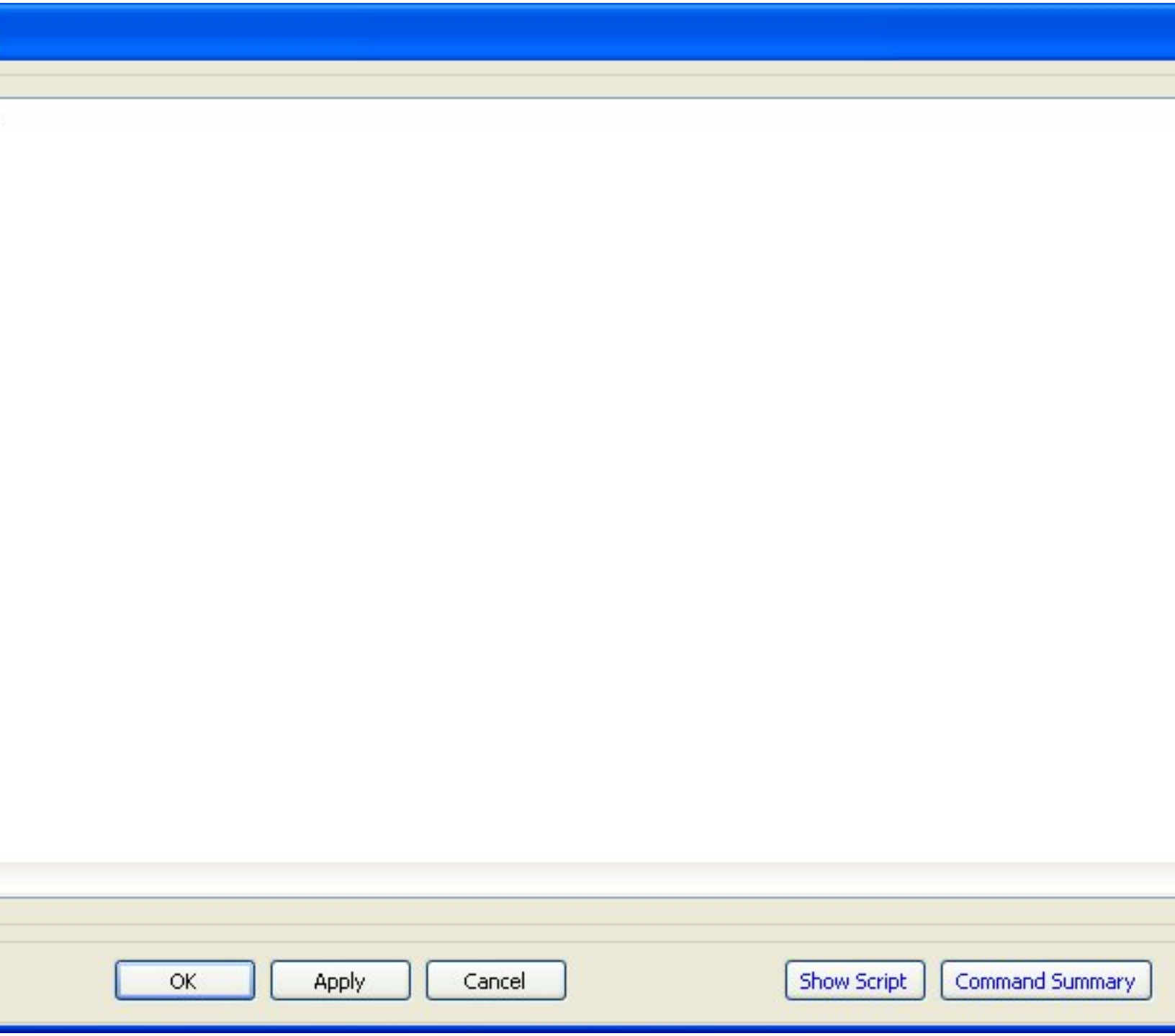


Figure: Default Name and Settings for the ScriptEvent Dialog Box

Using the script

Script Syntax

```
BeginScript
```

```
     <Statements>
EndScript;
```

Script Examples

```
% Assignment Command inside Script Event

BeginScript
     GMAT testVar = 24;
EndScript;
```

Reference Table

<Statements>	Default: N/A Options: { Any valid line of GMAT script } Units: N/A

Else

This was created using the June 9, 2008 build of GMAT.

Overview

If-Else statements in GMAT work as they do in other programming languages, especially Matlab. The Else command adds another dimension to an If statement. You use an Else statement when you want something to happen when the conditions of an If statement are not met. For example, an If statement who's condition is "x < 5" will only execute the script within its scope when x is indeed less than 5. GMAT would otherwise pass over the If statement's associated script and continue. However, having an Else statement after the If will ensure that the lines of script within the scope of that Else are executed when x is equal to 5 or greater.

Listing of Objects and Commands that interact

- If
  - The If command is the only part of GMAT that interacts with the Else command. Whenever Else is used, If must precede it as described below.

Using the Else Command in the GUI

Else appears as the option 'If-Else' in the GUI when adding it to the Mission tree. Once added to the tree, there will appear an If with an Else appended to it. You may add any command(s) you wish after the Else statement.

Figure: Default Appearance of the Else Command in the Mission tree

Using the Else Command in the script

Script Syntax

```
If <logical expression>;
```

```
   <Statements>;
Else;
   <Statements>;
EndIf;
```

Script Examples

```
If DefaultSC.ElapsedDays < 1;
   Propagate DefaultProp( DefaultSC , { DefaultSC.ElapsedDays = 0.01 });
Else;
   Propagate DefaultProp( DefaultSC , { DefaultSC.ElapsedDays = 0.2 });
EndIf;
```

NonLinearConstraint

This tutorial was started from the default settings in the June 9, 2008 build of GMAT.

Overview

The nonlinear constraints in GMAT allows spacecraft properties, variable or array to apply constraint values, and also NonLinearConstraint can be created in optimize sequences. By using the fmincon optimizer, users can give various nonlinear constraints.

Possible Coupling with Other Objects

- Optimize Command
  - NonLinearConstraints are used in Optimize Command.
- Optimizers (Solvers)
  - Must set optimizer in order to apply NonLinearConstraints.

Using the GUI

The NonLinearConstraint dialog contains four menus, which are Optimizer, Constraint, sign of equality or inequality and Constraint Value.

- Optimizer - Drop Down Menus
  - Must select Optimizer in order to apply NonLinearConstraint
- Constraint - Drop Down Menus or Field
  - Must set Constraint by using Choose dialog or giving real number
- sign of equality or inequality - Drop Down Menus
  - Must select the sign of equality or inequality from the box
- Constraint Value - Drop Down Menus or Field
  - Must set Constraint Value by using Choose dialog or giving real number

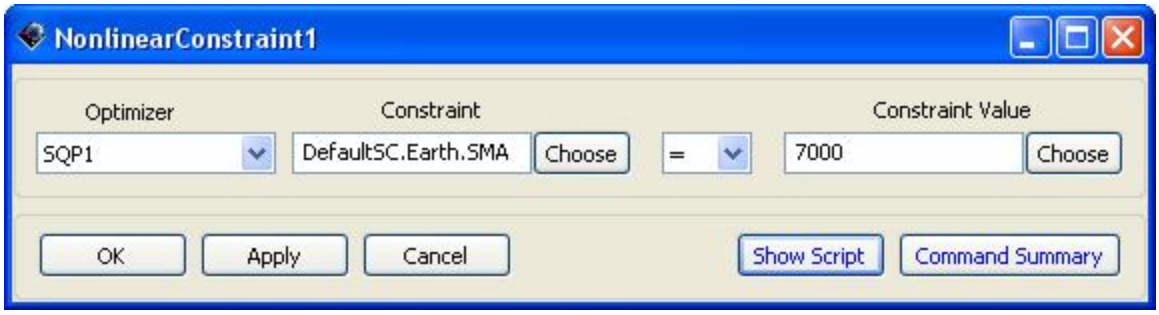


Figure: Default NonLinearConstraint Command Dialog Box

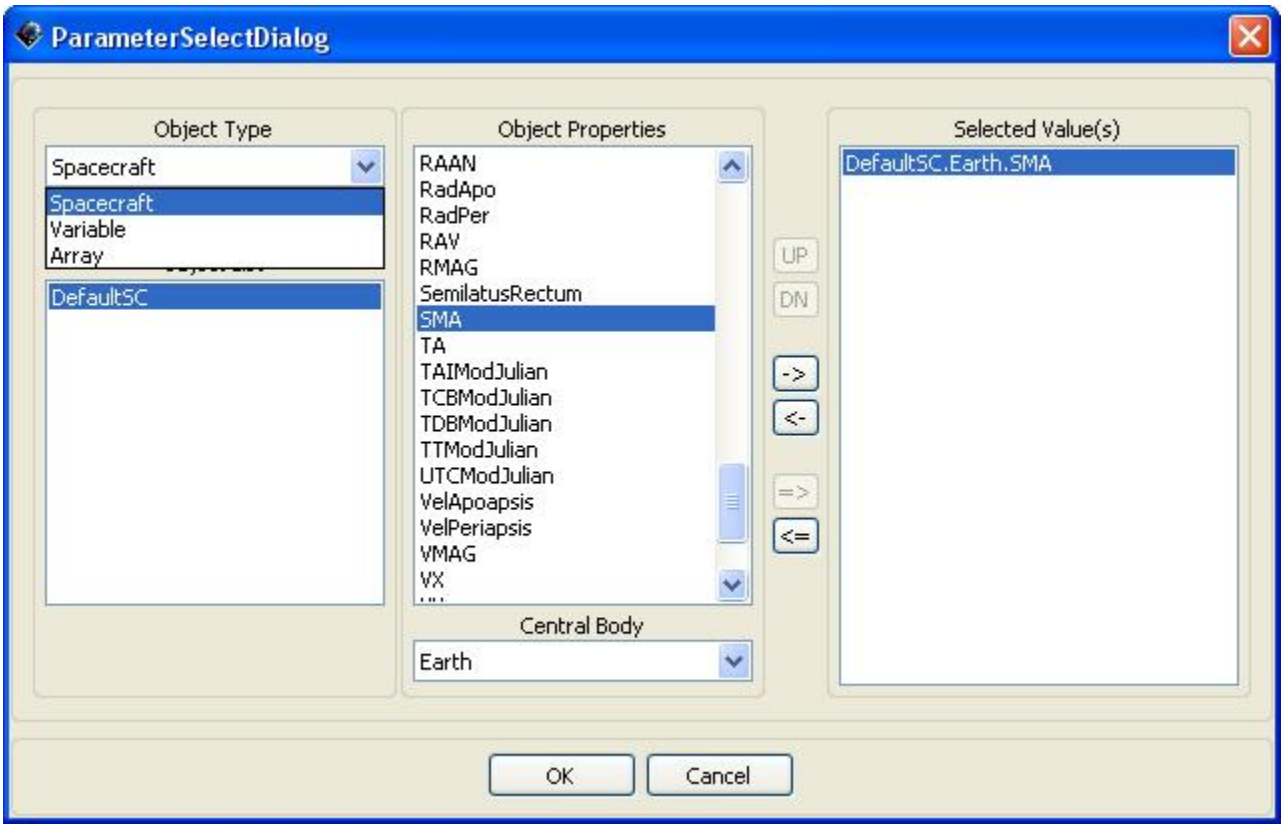


Figure: Parameter Selecting Dialog Box of NonLinearConstraint Command

Using the Script

Script Syntax

```
NonLinearConstraint OptimizerName ({logical expression})
```

Script Examples

```
% Constrain the SMA of Sat to be 7000 km, using fminconSQP
NonLinearConstraint fminconSQP( Sat.SMA = 7000 );

% Constrain the SMA of Sat to be less than or equal to 7000 km, using fminconSQP
```

```
NonLinearConstraint fminconSQP( Sat.SMA <= 7000 );

% Constrain the SMA of Sat to be greater than or equal to 7000 km, using fminconSQ
NonLinearConstraint fminconSQP( Sat.SMA >= 7000a );
```

Reference Table

OptimizeName	Default: SQP1. Options: { Any existing fmincon solver } The OptimizerName option allows the user to specify which solver to use in satisfying nonlinear constraints. Units: N/A.
{logical expression}	Default: DefaultSC.SMA = 7000. Options: { Arg1 <= Arg2 where <= can be >= ; <= ; = } The logical expression field allows the user to specify the constraint to be satisfied upon convergence of the solver given by OptimizerName. Arg1 and Arg2 can be any of the following: Real Number, a 1-D Array (column vector),Array element, Variable, Spacecraft Parameter or any other numeric user defined parameter. If Arg1 is a 1-D Array, then Arg2 must be a 1-D Array with the same dimensions and vice-versa. Units: N/A.

Minimize

This tutorial was started from the default settings in the June 9, 2008 build of GMAT.

Overview

The minimize command in GMAT allows variables to minimize by using a defined optimizer in optimize sequence.

Possible Coupling with Other Objects

- fmincon Object
  - Must set in order to use Minimize Command.
- Optimize Command
  - Must be defined in order to use minimize command in optimize sequence.

Using the GUI

The Minimize dialog contains two menus, which are optimizer and variable to be minimized. Each properties are needed to be selected to accomplish a minimization. Also, users have to define an optimizer before, and users need to define an undefined variable or function if users would like to apply.

- Optimizer - Drop Down Menus
  - Must select the optimizer, generally you can choose the solver in optimize command
- Variable to be Minimized - Drop Down Menus
  - Must select the variable or the function to be minimized

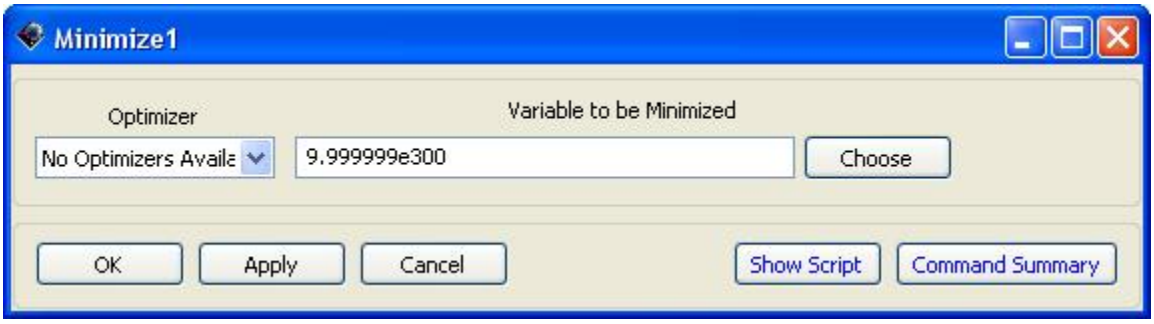


Figure: Default Minimize Command Dialog Box without setting fmincon and variable to be minimized

Using the Script

Script Syntax

```
Minimize OptimizerName (Arg)
```

Script Examples

```
% Minimize the eccentricity of Sat, using fminconSQP
Minimize fminconSQP(Sat.ECC);

% Minimize the Variable DeltaV, using fminconSQP
Minimize fminconSQP(DeltaV);

% Minimize the first component of MyArray, using fminconSQP
Minimize fminconSQP(MyArray(1,1));
```

Reference Table

OptimizerName	Default: SQP1. Options: { Any existing fmincon solver } The OptimizerName option allows the user to specify which solver to use to minimize the cost function.

	Units: N/A.
Arg	Default: DefaultSC.ECC. Options: { Variable, Spacecraft parameter, or Array element } The Arg field allows the user to specify the function to be minimized upon convergence of the solver given by OptimizerName. Arg can be any of the following: Variable, Array element, or Spacecraft Parameter or any other 1x1 numeric user defined parameter. Units: N/A.

Optimize

This tutorial was started from the default settings in the June 9, 2008 build of GMAT.

Overview

The optimize command in GMAT allows variables to optimize by using a solver fmincon object.

Possible Coupling with Other Objects

- fmincon Object
  - Must set in order to use Optimize Command.
- Minimize
  - Must be defined if you would like to minimize variable in an optimize sequence.
- Vary
- NonLinearConstraint

Using the GUI

The Optimize dialog contains only solver user can selected. Also, in optimize sequence user can give some commands.

- Solver Name - Drop Down Menus
  - Must select solver applying for optimize sequence

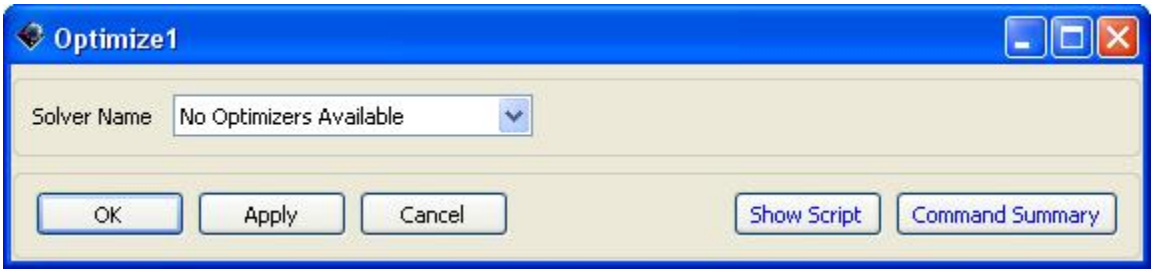


Figure: Default Optimize Command Dialog Box without setting fmincon

Using the Script

Script Syntax

```
Optimize SolverName;
   <Statements>;
EndOptimize;
```

Script Examples

```
% Beginning and ending syntax for the Optimize command
Optimize DefaultDC;

EndOptimize;
```

Reference Table

SolverName	Default: DefaultSQP. Options: { Any existung optimizer } The SolverName field allows the user to choose between any previously created optimizer for use in an optimization sequence. For example, to begin a optimization sequence using DefaultSQP, the script is Optimize DefaultSQP. Units: N/A.
<Statements>	Default: None. Options: { Any non-targeter and non-optimizer command lines used in the mission sequence, as well as the optimizer dependent command lines Vary, NonLinearConstraint, and Minimize. } Units: N/A.

Call Function

This was created using the June 9, 2008 build of GMAT.

Overview

GMAT functions are very useful and work nearly the same as they do in most programming languages. They may be invoked using the Call Function command covered here.

Listing of Objects and Commands that interact

- Function

  - Any function that you create will be available to the Call Function command for use.
- Spacecraft
  - All of the standard spacecraft object properties are available for function inputs and outputs.
- Impulsive Burn
  - Both the standard direction sets of Elements 1, 2, and 3, as well as V, N, and B are available for function inputs and outputs.
- Variable
  - Variables are available as values for both input and output.
- Array
  - Arrays are available as data for both input and output.
- String
  - Strings are available as data for both input and output.

Using the Call Function Command in the GUI

The Call Function Command is is useful in that

- Output - Display
  - This is where the Objects or values selected for output are displayed.
- View (Output) - Button
  - This button brings up the standard Parameter Select dialog box. It allows you to add objects to the list of outputs.
- Function - Drop Down Menu
  - This is where you select the function that will be called by the Call Function command.
- Input - Display
  - This is where the Objects or values selected for input are displayed.
- View (Input) - Button
  - This button also brings up the standard Parameter Select dialog box. It allows you to add objects to the list of inputs.

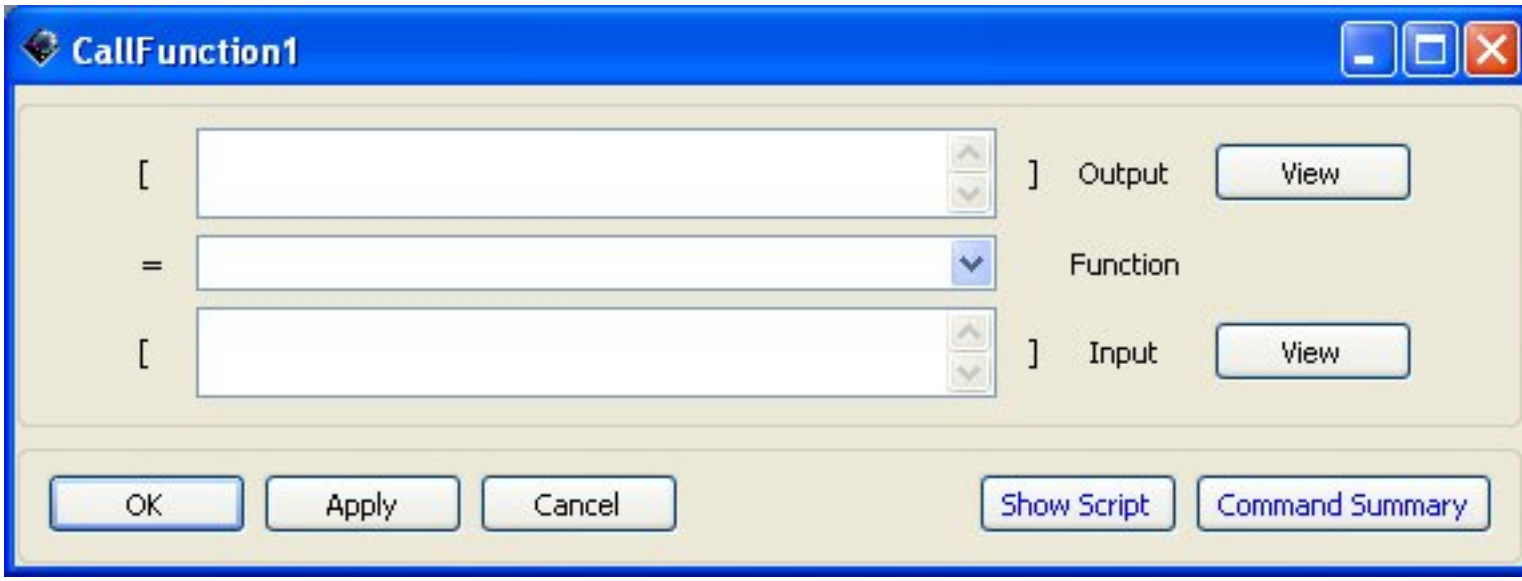


Figure: Default Name and Settings for the Call Function Command Dialog Box

Using the Call Function Command in the script

Script Syntax

```
%Function call with Inputs and Outputs
GMAT [[OutputList] = Function(InputList)

%Function call with Outputs only
GMAT [OutputList] = Function

%Function call with Inputs only
GMAT Function(InputList)

Function call with no Inputs or Outputs
GMAT Function
```

Script Examples

```
% Matlab function call without inputs or outputs

% Syntax 1
GMAT clearAll

% Syntax 2
GMAT [ ] = clearAll( )
```

Reference Table

OutputList	Default: None Options: [Variables, Arrays, S/C, Paramters, any other user-defined parameters, or blank. Multiple

	outputs must be expressed in a comma delimited list format] The OutputList option allows the user to set the output of Function to a user defined parameter. Units: N/A
InputList	Default: None Options: [Variables, Arrays, S/C, Paramters, any other user-defined parameters, or blank. Multiple inputs must be expressed in a comma delimited list format.] The InputList option allows the user to set the input of Function to a user defined parameter. Units: N/A
Function	Default: None Options: [GMAT of Matlab Function] The Function option allows the user to set the function that will be called in a specific location of the mission sequence. The function has to be defined before it can be used in the CallFunction Command. Units: N/A

Chapter 26. Miscellaneous

Math Operators in Scripting

Objective and Overview

The objective of this reference article is to inform GMAT users of the Math Operators available to them and their syntax in scripting.

(i.e. GMAT MyVar = 5 + 6 * sqrt(4))

All examples assume variables and arrays, MyVar* and MyArr*, have already been created.

Addition

GMAT MyVar = 4 + 5;

Subtraction

GMAT MyVar = 3 - 1;

Division

GMAT MyVar = 8 / 2;

Multiplication

GMAT MyVar = 12 * 2;

Exponentiation

GMAT MyVar = 2 ^ 2;

Trigonometry

Standard Units of Radians

Sine

sin()

Cosine

cos()

Tangent

tan()

Arcsine

asin()

Arccosine

acos()

Arctangent

atan()

atan2

atan2()

Square Root

GMAT MyVar = sqrt(4);

norm

norm()

Logarithm

log()

log10()

exp()

Matrix Manipulation

inv()

()^()

transpose()

()'

det()

Radians and Degrees

DegToRad()

RadToDeg()

Script Editor Shortcut Keys

Reference: Script Editor Shortcut Keys

The GMAT Script Editor provides a user with text editor capabilities inside of the GMAT GUI. This article provides a listing of the current shortcut keys used in the Script Editor by release date. Shortcut key are subject to change in future releases. Check back to this article for Script Editor shortcut key updates.

Would you like to take part in discussions for possible changes to the current shortcut keys? If so, visit the following forum topic:

[http://gmat.ed-pages.com/forum/viewtopic.php?f=2&t=107|Proposed Shortcut Keys inside GMAT (Text Fields)]

2008-09-30 and 2007-12-10

Ctrl + C	Copy
Ctrl + X	Cut
Ctrl + V	Paste
Ctrl + Z	Undo
Ctrl + Y	Redo
Ctrl + A	Select All
Tab	Move cursor to the next input field
Shift + Tab	Move cursor to the previous input field
Ctrl + L	Aligns highlighted text to the left (default alignment)
Ctrl + E	Aligns highlighted text to the center (This alignment is not saved when the script is reloaded)
Ctrl + R	Aligns highlighted text to the right (This alignment is not saved when the script is reloaded)
Ctrl + J	Aligns highlighted text to justified left?? (This alignment is not saved when the script is reloaded)
Ctrl + track mouse scroll	Enlarge/Decrease text size in script editor

2007-08-23

Ctrl + C	Copy
Ctrl + X	Cut
Ctrl + V	Paste
Ctrl + Z	Undo
Ctrl + Shift + Z	Redo
Ctrl + A	Select All
Tab	Move cursor to the next input field
Shift + Tab	Move cursor to the previous input field
Ctrl + L	Aligns highlighted text to the left (default alignment)
Ctrl + E	Aligns highlighted text to the center (This alignment is not saved when the script is reloaded)
Ctrl + R	Aligns highlighted text to the right (This alignment is not saved when the script is reloaded)
Ctrl + T	Comments highlighted text. (The entire line has to be highlighted)

Ctrl + M	Uncomments highlighted text. (The entire line has to be highlighted)
Ctrl + track mouse scroll	Enlarge/Decrease text size in script editor

Process Resources

Overview

This page contains links and references for process improvement. The items below have been or may be useful to this project...they are not endorsements!!

Document Status: Very Early Draft (This means unreviewed by the team!!)

Primary Author: S. Hughes

General Process References

http://en.wikipedia.org/wiki/Capability_Maturity_Model

http://en.wikipedia.org/wiki/Process_area_%28CMMI%29

http://software.gsfc.nasa.gov/

http://en.wikipedia.org/wiki/Rational_Unified_Process

http://en.wikipedia.org/wiki/Feature_Driven_Development

Configuration Management

http://www.amazon.com/Software-Configuration-Management-Handbook-Second/dp/1580538827/ref!si3rdrbbproduct

http://www.amazon.com/gp/product/0201741172/refcm_cr_pr_product_top

http://www.cmcrossroads.com/bradapp/acme/scm-readings.html#SEI_SCM_Docs

http://www.perforce.com/perforce/bestpractices.html

Reviews

GMAT Commands in MATLAB

Overview

The following list of commands are originally from GMAT, but are designed to be called from MATLAB® to be run in GMAT. Under the Interfaces folder in the resource tree, the MATLAB® Server must be started so MATLAB® can call GMAT commands and run them in GMAT. After a command is entered into MATLAB®, run the and then notice that GMAT automatically executes the command.

Commands

GMAT Commands with Similar Syntax

The commands below all have similar MATLAB® syntax and can be seen by opening up any one of the files. The script files send the functions from MATLAB® to GMAT and into their respective places in the mission sequence. The commands are then built and run in the sequence as they would normally have been if originally created and used in GMAT.

- Achieve
- Assignment
- BeginFiniteBurn
- BeginScript
- BuildGMAT
- Case
- Create
- Else
- ElseIf
- EndFiniteBurn
- EndFor
- EndIf
- EndOptimize
- EndScript
- EndTarget
- EndWhile
- For
- GMAT
- If
- Maneuver
- Minimize
- NonLinearConstraint
- Optimize
- PenDown
- PenUp

- Propagate
- Report
- Save
- SendGMATObject
- Stop
- Target
- Toggle
- Vary
- While

MATLAB® Commands with Similar Syntax

The purpose of the following files is to identify and convert any line of structured GMAT code into a readable string for MATLAB®. Once a string or array of strings is converted, GMAT is called and the strings are sent back into a GMAT script to be built and then run. An example of a multilevel string, depending on the object and parameter being defined, is when an spacecraft is called with X, Y, and Z coordinates.

- SBInput2Str
- Struct2Str

GMAT Console Instructions

Thanks to DJC for the instructions.

This page needs to be changed to match the style guide.

If you just run gmatConsole from a terminal window, it will prompt you for a script file name. Entering the name runs the script, and returns you to a prompt to either quit or run another script.

The console application can also be run in "batch" mode. To do that, create a file listing the scripts you want to run (including path information if the scripts are not in the bin directory), one per line, and save that file where you can find it. Then run the batch file by typing "gmatConsole --batch" to run all of the scripts listed in the batch file.