General Mission Analysis Tool (GMAT)
User Guide

The GMAT Development Team

18 May 2012
# Table of Contents

Preface ............................................................................................................................ vi

I. Introduction .................................................................................................................. 1  
   Introduction to GMAT ............................................................................................. 2  
   Licensing .......................................................................................................... 2  
   Platform Support .............................................................................................. 2  
   User Interfaces ................................................................................................. 3  
   Development Status .......................................................................................... 3  
   Contributors ..................................................................................................... 3  

Getting Started ......................................................................................................... 4  
   Installation ....................................................................................................... 4  
   Starting and Quitting GMAT ............................................................................. 4  
   Running the GMAT Demos .............................................................................. 4  
   User Interfaces Overview .................................................................................. 5  
   Data and Configuration ................................................................................... 10  
   Other Resources ............................................................................................. 16  

II. Creating Your First Mission ......................................................................................... 17  
   Simulating an Orbit ................................................................................................. 18  
   Objective and Overview .................................................................................. 18  
   Configure the Spacecraft .................................................................................. 18  
   Configure the Propagator ................................................................................. 20  
   Configure the Propagate Command .................................................................. 21  
   Run and Analyze the Results ............................................................................ 22  

III. Common Tasks ......................................................................................................... 24  
   Configuring a Spacecraft .......................................................................................... 25  
   Setting the Initial Epoch .................................................................................. 25  
   Configuring the Orbit ...................................................................................... 25  
   Configuring Physical Properties ........................................................................ 26  
   Configuring the Attitude (Fixed) ....................................................................... 27  
   Configuring the Attitude (Spinner) .................................................................... 27  

   Propagating a Spacecraft .......................................................................................... 29  
   Configuring the Force Model ........................................................................... 29  
   Configuring the Force Model: Mars .................................................................. 29  
   Propagating for a Duration .............................................................................. 30  
   Propagating to an Orbit Condition ................................................................... 31  

   Reporting Data ....................................................................................................... 32  
   Reporting Data During a Propagation Span ....................................................... 32  
   Reporting Data at a Specific Mission Event ....................................................... 32  
   Creating a CCSDS Ephemeris File .................................................................... 33  
   Creating an SPK Ephemeris File ...................................................................... 33  

   Visualizing Data ...................................................................................................... 35  
   Manipulating the 3D Orbit View ...................................................................... 35  
   Configuring the Ground Track Plot .................................................................. 35  
   Creating a 2D Plot .......................................................................................... 35  

IV. Tutorials ................................................................................................................... 37  
   Simple Orbit Transfer .............................................................................................. 38  
   Objective and Overview .................................................................................... 38
<table>
<thead>
<tr>
<th>Reference Guide</th>
<th>Commands</th>
<th>Resources</th>
</tr>
</thead>
<tbody>
<tr>
<td>V. Reference Guide</td>
<td>II. Commands</td>
<td>I. Resources</td>
</tr>
<tr>
<td>Run the Mission</td>
<td>Achieve</td>
<td>Configure Maneuvers, Differential Corrector, and Graphics</td>
</tr>
<tr>
<td>Configure the Mission Sequence</td>
<td>BeginFiniteBurn</td>
<td>Configure the Mission Sequence</td>
</tr>
<tr>
<td></td>
<td>BeginMissionSequence</td>
<td>Configure Maneuvers, Differential Corrector, and Graphics</td>
</tr>
<tr>
<td></td>
<td>CallGmatFunction</td>
<td>Configure Maneuvers, Differential Corrector, and Graphics</td>
</tr>
<tr>
<td></td>
<td>CallMatlabFunction</td>
<td>Configure Maneuvers, Differential Corrector, and Graphics</td>
</tr>
<tr>
<td></td>
<td>ClearPlot</td>
<td>Configure Maneuvers, Differential Corrector, and Graphics</td>
</tr>
<tr>
<td></td>
<td>Else</td>
<td>Configure Maneuvers, Differential Corrector, and Graphics</td>
</tr>
<tr>
<td></td>
<td>EndFiniteBurn</td>
<td>Configure Maneuvers, Differential Corrector, and Graphics</td>
</tr>
<tr>
<td></td>
<td>Equation</td>
<td>Configure Maneuvers, Differential Corrector, and Graphics</td>
</tr>
<tr>
<td></td>
<td>For</td>
<td>Configure Maneuvers, Differential Corrector, and Graphics</td>
</tr>
<tr>
<td></td>
<td>Global</td>
<td>Configure Maneuvers, Differential Corrector, and Graphics</td>
</tr>
<tr>
<td></td>
<td>If</td>
<td>Configure Maneuvers, Differential Corrector, and Graphics</td>
</tr>
<tr>
<td></td>
<td>Maneuver</td>
<td>Configure Maneuvers, Differential Corrector, and Graphics</td>
</tr>
<tr>
<td></td>
<td>MarkPoint</td>
<td>Configure Maneuvers, Differential Corrector, and Graphics</td>
</tr>
<tr>
<td></td>
<td>Minimize</td>
<td>Configure Maneuvers, Differential Corrector, and Graphics</td>
</tr>
<tr>
<td></td>
<td>NonlinearConstraint</td>
<td>Configure Maneuvers, Differential Corrector, and Graphics</td>
</tr>
<tr>
<td></td>
<td>NonlinearConstraint</td>
<td>Configure Maneuvers, Differential Corrector, and Graphics</td>
</tr>
<tr>
<td></td>
<td>Orbits</td>
<td>Configure Maneuvers, Differential Corrector, and Graphics</td>
</tr>
<tr>
<td></td>
<td>Orbiter</td>
<td>Configure Maneuvers, Differential Corrector, and Graphics</td>
</tr>
<tr>
<td></td>
<td>OrbitView</td>
<td>Configure Maneuvers, Differential Corrector, and Graphics</td>
</tr>
<tr>
<td></td>
<td>ReportFile</td>
<td>Configure Maneuvers, Differential Corrector, and Graphics</td>
</tr>
<tr>
<td></td>
<td>Run the Mission</td>
<td>Configure Maneuvers, Differential Corrector, and Graphics</td>
</tr>
<tr>
<td></td>
<td>Spacecraft</td>
<td>Configure Maneuvers, Differential Corrector, and Graphics</td>
</tr>
<tr>
<td></td>
<td>SolarSystem</td>
<td>Configure Maneuvers, Differential Corrector, and Graphics</td>
</tr>
<tr>
<td></td>
<td>Spacetrack</td>
<td>Configure Maneuvers, Differential Corrector, and Graphics</td>
</tr>
<tr>
<td></td>
<td>SPQ</td>
<td>Configure Maneuvers, Differential Corrector, and Graphics</td>
</tr>
<tr>
<td></td>
<td>String</td>
<td>Configure Maneuvers, Differential Corrector, and Graphics</td>
</tr>
<tr>
<td></td>
<td>Thruster</td>
<td>Configure Maneuvers, Differential Corrector, and Graphics</td>
</tr>
<tr>
<td></td>
<td>Variable</td>
<td>Configure Maneuvers, Differential Corrector, and Graphics</td>
</tr>
<tr>
<td></td>
<td>VFI3adOptimizer</td>
<td>Configure Maneuvers, Differential Corrector, and Graphics</td>
</tr>
<tr>
<td></td>
<td>XYPlot</td>
<td>Configure Maneuvers, Differential Corrector, and Graphics</td>
</tr>
</tbody>
</table>
The GMAT User’s Guide contains material for new and experienced users and is organized into the following sections:

- Introduction
- Creating Your First Mission
- Common Tasks
- Tutorials
- Reference Guide

Introduction

The Introduction section contains two major parts: Introduction to GMAT and Getting Started.

The Introduction to GMAT section contains a brief project and software overview and discusses project status, licensing, and contributors.

The Getting Started section describes how to install and start GMAT, presents an overview of the user interfaces, and provides information on configuring your system.

Note

We consider the User Interfaces Overview essential reading. If you read nothing else, at least read this section as it will explain the basic philosophy and rules of GMAT’s user interfaces.

Creating Your First Mission

The Creating Your First Mission section walks you step-by-step through a sample mission, including creating a spacecraft, a propagator, and an OrbitView graphical display, and propagating the spacecraft to orbit perigee.

Common Tasks

The Common Tasks section contains many short articles that each describe a single area of functionality. The purpose of the how-to documentation is to show you how to use a specific feature in an analysis context, and these articles often start from the default mission that is loaded when you start GMAT. A common task section is designed to take about five minutes to teach you how to perform a specific task.

Tutorials

The Tutorials section describes how to use GMAT for end-to-end analysis. Tutorials are designed to teach you how to use GMAT in the context of performing real-world analysis and are intended to take between 30 minutes and several hours to complete. Each tutorial has a difficulty level and an approximate duration listed with any prerequisites in its introduction.
Reference Guide

The Reference Guide contains individual topics that describe each of GMAT's resources and commands in detail, including its syntax, options, variable ranges and data types, defaults, and expected behavior.

Typographical Conventions

This document uses two typographical conventions throughout:

- Graphical user interface (GUI) elements are presented in **bold**.
- Filenames, resource and command names, and script examples are presented in `monospace`.

vii
Part I. Introduction

Table of Contents

Introduction to GMAT ..................................................................................................................... 2
  Licensing .................................................................................................................................. 2
  Platform Support ....................................................................................................................... 2
  User Interfaces ........................................................................................................................... 3
  Development Status ................................................................................................................ 3
  Contributors ............................................................................................................................. 3

Getting Started ................................................................................................................................. 4
  Installation ............................................................................................................................... 4
  Starting and Quitting GMAT ..................................................................................................... 4
  Running the GMAT Demos ......................................................................................................... 4
  User Interfaces Overview .......................................................................................................... 5
  Data and Configuration ............................................................................................................. 10
  Other Resources .................................................................................................................... 16
Introduction to GMAT

GMAT is an open source trajectory design and optimization system developed by NASA and private industry. It is developed in an open source process to maximize technology transfer, to permit anyone to develop and validate new algorithms, and to enable those new algorithms to quickly transition into the high fidelity core.

GMAT is designed to model and optimize spacecraft trajectories in flight regimes ranging from low Earth orbit to lunar, interplanetary, and other deep space missions. The system supports constrained and unconstrained trajectory optimization and built-in features make defining cost and constraint functions trivial. GMAT also contains initial value solvers (propagators) and boundary value solvers and efficiently propagates spacecraft either singly or as coupled sets. GMAT’s propagators naturally synchronize the epochs of multiple vehicles and avoid fixed step integration and interpolation when doing so.

Users can interact with GMAT using either a graphical user interface (GUI) or a custom scripting language modeled after the syntax used in The MathWorks’ MATLAB® system. All of the system elements can be expressed through either interface, and users can convert between the two in either direction.

Analysts model space missions in GMAT by first creating and configuring resources such as spacecraft, propagators, optimizers, and data files. These resources are then used in a mission sequence to model the trajectory of the spacecraft and simulate mission events. The mission sequence supports commands such as nonlinear constraints, minimization, propagation, GMAT and MATLAB functions, inline equations, and script events.

GMAT can display trajectories in a realistic three-dimensional view, plot parameters against one another, and save parameters to files for later processing. The graphics capabilities are fully interactive, plotting data as a mission is run and allowing users to zoom into regions of interest. Trajectories and data can be viewed in any coordinate system defined in GMAT, and GMAT allows users to rotate the view and set the focus to any object in the display. The trajectory view can be animated so users can watch the evolution of the trajectory over time.

Licensing

GMAT is licensed under the NASA Open Source Agreement v1.3. The license text is contained in the file `License.txt` in root directory of the GMAT distribution.

Platform Support

GMAT is cross-platform software and runs on Windows, Linux, and Macintosh platforms, on both 32-bit and 64-bit architectures. It uses the wxWidgets cross-platform user interface toolkit and can be built using either Microsoft Visual Studio or the GNU Compiler Collection (GCC). GMAT is written in ANSI standard C++ (approximately 380,000 non-comment source lines of code) using an object-oriented methodology, with a rich class structure designed to make new features simple to incorporate.
User Interfaces

GMAT has several user interfaces. The interactive graphical user interface is introduced in more detail in later sections. The script interface is textual and also allows the user to configure and execute all aspects of GMAT. There is a secondary MATLAB interface that allows for running the system via calls from MATLAB to GMAT and allows GMAT to call MATLAB functions from within the GMAT command sequence. A low-level C API is also currently under development.

Development Status

While GMAT has undergone extensive testing and is mature software, at the present time we consider the software to be in beta form on Windows and alpha on Linux and Mac. GMAT is not yet sufficiently verified to be used as a primary operational analysis system. It has been used to optimize maneuvers for flight projects such as NASA’s LCROSS and ARTEMIS missions, and the Lunar Reconnaissance Orbiter, and for optimization and analysis for the OSIRIS-REx and MMS missions. However, for flight planning, we independently verify solutions generated in GMAT in the primary operational system.

The GMAT team is currently working on several activities including maintenance, bug fixes, and testing, along with selected new functionality.

Contributors

The Navigation and Mission Design Branch at NASA’s Goddard Space Flight Center performs project management activities and is involved in most phases of the development process including requirements, algorithms, design, and testing. The Ground Software Systems Branch performs design, implementation, and integration testing. The Flight Software Branch contributes to design and implementation. GMAT contributors include volunteers and those paid for services they provide.

We welcome new contributors to the project, either as users providing feedback about the features of the system, or as developers interested in contributing to the implementation of the system. Current and past contributors include:

- Thinking Systems, Inc. (system architecture and all aspects of development)
- Air Force Research Lab (all aspects of development)
- a.i. solutions (testing)
- Boeing (algorithms and testing)
- The Schafer Corporation (all aspects of development)
- Honeywell Technology Solutions (testing)
- Computer Sciences Corporation (requirements)

The NASA Jet Propulsion Laboratory (JPL) has provided funding for integration of the SPICE toolkit into GMAT. Additionally, the European Space Agency’s (ESA) Advanced Concepts team has developed optimizer plug-ins for the Non-Linear Programming (NLP) solvers SNOPT (Sparse Nonlinear OPTimizer) and IPOPT (Interior Point OPTimizer).
Getting Started

Installation

Installers and files for Windows are located on the GMAT SourceForge page at https://sourceforge.net/projects/gmat. As of this writing the latest version is R2012a, released May 23, 2012.

The GMAT Windows distribution contains an installer that will install and configure GMAT for you automatically. By default GMAT will be installed into the application data folder in your user profile, and a shortcut will be placed in the Start menu.

GMAT is available as a source code bundle for other platforms. See the GMAT Wiki for compiling instructions.

Starting and Quitting GMAT

Starting a GMAT Session

On Microsoft Windows platforms there are several ways to start a GMAT session. If you used the GMAT installer, you can click the GMAT R2012a item in the Start menu. If you installed GMAT from a zip file or by compiling the system, locate the bin directory in the GMAT root directory and double-click GMAT.exe.

On the Mac, use Finder to open the bin folder located in the GMAT root directory and open the GMAT application. Alternatively, open a Terminal window, change to your installation directory, then type the command open GMAT.app. Once GMAT is open, you can set it to remain in the dock by clicking its dock icon, then Options, then Keep in Dock. This allows you to open GMAT in the future simply by clicking its dock icon.

Quitting a GMAT Session

To end a GMAT session on Windows or Linux, in the menu bar, click File, then click Exit. On the Mac, in the menu bar, click GMAT, then click Quit GMAT, or type Command+Q.

Running the GMAT Demos

The GMAT distribution includes more than 30 sample missions. These samples show how to apply GMAT to problems ranging from the Hohmann transfer to libration point station-keeping to trajectory optimization. To locate and run a sample mission:

1. Open GMAT.
2. On the toolbar click Open.
3. Navigate to the samples folder located in the GMAT root directory.
4. Double-click a script file of your choice.
5. Click Run.

To run optimization missions, you will need MATLAB and the MATLAB Optimization Toolbox and/or the VF13ad plugin based on software in the Harwell Subroutine Library. These are proprietary libraries and are not distributed with GMAT. MATLAB connectivity is not yet fully supported.
in the Mac and Linux GMAT releases, and therefore you cannot run optimization missions that use MATLAB’s *fmincon* optimizer on those platforms.

**User Interfaces Overview**

GMAT offers multiple ways to design and execute your mission. The two primary interfaces are the graphical user interface (GUI) and the script interface. These interfaces are interchangeable and each supports most of the functionality available in GMAT. When you work in the script interface, you are working in GMAT’s custom script language. To avoid issues such as circular dependencies, there are some basic rules you must follow. Below, we discuss these interfaces and then discuss the basic rules and best practices for working in each interface.

**GUI Overview**

When you start a session, the GMAT desktop is displayed with a default mission already loaded. The GMAT desktop has a native look and feel on each platform and most desktop components are supported on all platforms.

**Windows GUI**

When you open GMAT on Windows and click **Run** in the Toolbar, GMAT executes the default mission as shown in the figure below. The tools listed below the figure are available in the GMAT desktop.

![Figure 1. GMAT Desktop (Windows)](image)
Menu Bar

The menu bar contains **File**, **Edit**, **Window** and **Help** functionality.

On Windows, the **File** menu contains standard **Open**, **Save**, **Save As**, and **Exit** functionality as well as **Open Recent** and **New Mission**. The **Edit** menu contains functionality for script editing when the script editor is active. The **Window** menu contains tools for organizing graphics windows and the script editor within the GMAT desktop. Examples include the ability to **Tile** windows, **Cascade** windows and **Close** windows. The **Help** menu contains links to **Online Help**, **Tutorials**, **Forums**, and the **Report An Issue** option links to GMAT's defect reporting system, the **Welcome Page**, and a **Provide Feedback** link.

On the Mac, menus are nearly the same, with a few differences: the **File** menu does not contain an **Exit** option - instead, the **Quit GMAT** menu option is on the GMAT menu, as discussed before; tiling and cascading windows are not supported, so those options do not appear under the **Window** menu; currently, email is not supported, so **Provide Feedback** is nonfunctional under the **Help** menu.

Toolbar

The toolbar provides easy access to frequently used controls such as file controls, **Run**, **Pause**, and **Stop** for mission execution, and controls for graphics animation. On Windows and Linux, the toolbar is located at the top of the GMAT window; on the Mac, it is located on the left of the GMAT frame. Because the toolbar is vertical on the Mac, some toolbar options are abbreviated.

GMAT allows you to simultaneously edit the raw script file representation of your mission and the GUI representation of your mission. It is possible to make inconsistent changes in these mission representations. The **GUI/Script Sync Status** indicator located in the toolbar shows you the state of the two mission representations. See the the section called “GUI/Script Interactions and Synchronization” section for further discussion.

Resources Tab

The **Resources** tab brings the **Resources** tree to the foreground of the desktop.

Resources Tree

The **Resources** tree displays all configured GMAT resources and organizes them into logical groups. All objects created in a GMAT script using a **Create** command are found in the **Resources** tree in the GMAT desktop.

Mission Tab

The **Mission** tab brings the Mission Tree to the foreground of the desktop.

Mission Tree

The **Mission** tree displays GMAT commands that control the time-ordered sequence of events in a mission. The **Mission** tree contains all script lines that occur after the **BeginMissionSequence** command in a GMAT script. You can undock the **Mission** tree as shown in the figure below by right-clicking on the **Mission** tab and dragging it into the graphics window. You can also follow these steps:

1. Click on the **Mission** tab to bring the Mission Tree to the foreground.
2. Right-click on the **Mission Sequence** folder in the **Mission** tree and select **Undock Mission Tree** in the menu.
Figure 2. Undocked Mission Tree

**Output Tab**
The Output tab brings the Output Tree to the foreground of the desktop.

**Output Tree**
The Output tree contains GMAT output such as report files and graphical displays.

**Message Window**
When you run a mission in GMAT, information including warnings, errors, and progress are written to the message window. For example, if there is a syntax error in a script file, a detailed error message is written to the message window.

**Status Bar**
The status bar contains various informational messages about the state of the GUI. When a mission is running, a Busy indicator will appear in the left pane. The center pane displays the latitude and longitude of the mouse cursor as it moves over a ground track window.

**Script Interface Overview**

The GMAT script editor is a textual interface that lets you directly edit your mission in GMAT's built-in scripting language. In Figure 3 below, the script editor is shown maximized in the GMAT desktop and the items relevant to script editing are labeled.
The GMAT desktop allows you to have multiple script files open simultaneously. Open script files are displayed in the Scripts folder in the Resources tree. Double click on a script in the Scripts folder to open it in the script editor. The GMAT desktop displays each script in a separate script editor. GMAT indicates the script currently represented in the GUI with a boldface name. Only one script can be loaded into the GUI at a time.

The Script Status box indicates whether or not the script being edited is loaded in the GUI. The box says Active Script for the script currently represented in the GUI and Inactive Script for all others.

The Save, Sync button saves any script file changes to disk, makes the script active, and synchronizes the GUI with the script.

The Save, Sync, Run button saves any script file changes to disk, makes the script active, synchronizes the GUI with the script, and executes the script.

When you click Save As, GMAT displays the Choose A File dialog box and allows you to save the script using a new file name. After saving, GMAT loads the script into the GUI, making the new file the active script.

The Close button closes the script editor.
GUI/Script Interface Interactions and Rules

The GMAT desktop supports both a script interface and a GUI interface and these interfaces are designed to be consistent with each other. You can think of the script and GUI as different "views" of the same data: the resources and the mission command sequence. GMAT allows you to switch between views (script and GUI) and have the same view open in an editable state simultaneously. Below we describe the behavior, interactions, and rules of the script and GUI interfaces so you can avoid confusion and potential loss of data.

GUI/Script Interactions and Synchronization

GMAT allows you to simultaneously edit both the script file representation and the GUI representation of your mission. It is possible to make inconsistent changes in these representations. The GUI/Script Sync Status window located in the toolbar indicates the state of the two representations. On the Mac, the status is indicated in abbreviated form in the left-hand toolbar. Synchronized (green) indicates that the script and GUI contain the same information. GUI Modified (yellow) indicates that there are changes in the GUI that have not been saved to the script. Script Modified (yellow) indicates that there are changes in the script that have not been loaded into the GUI. Unsyncro-nized (red) indicates that there are changes in both the script and the GUI.

Caution

GMAT will not attempt to merge or resolve simultaneous changes in the Script and GUI and you must choose which representation to save if you have made changes in both interfaces.

The Save button in the toolbar saves the GUI representation over the script. The Save,Sync button on the script editor saves the script representation and loads it into the GUI.

How the GUI Maps to a Script

Clicking the Save button in the toolbar saves the GUI representation to the script file; this is the same file you edit when working in the script editor. GUI items that appear in the Resources tree appear before the BeginMissionSequence command in a script file and are written in a predefined order. GUI items that appear in the Mission Tree appear after the BeginMissionSequence command in a script file in the same order as they appear in the GUI.

Caution

If you have a script file that has custom formatting such as spacing and data organization, you should work exclusively in the script. If you load your script into the GUI, then click Save in the toolbar, you will lose the formatting of your script. (You will not, however, lose the data.)

How the Script Maps to the GUI

Clicking the Save,Sync button on the script editor saves the script representation and loads it into the GUI. When you work in a GMAT script, you work in the raw file that GMAT reads and writes. Each
script file must contain a command called \texttt{BeginMissionSequence}. Script lines that appear before the \texttt{BeginMissionSequence} command create and configure models and this data will appear in the \texttt{Resources} tree in the GUI. Script lines that appear after the \texttt{BeginMissionSequence} command define your mission sequence and appear in the \texttt{Mission} tree in the GUI. Here is a brief script example to illustrate:

\begin{verbatim}
Create Spacecraft Sat
Sat.X = 3000
BeginMissionSequence
Sat.X = 1000
\end{verbatim}

The line \texttt{Sat.X = 3000} sets the x-component of the Cartesian state to 3000; this value will appear on the \texttt{Orbit} tab of the \texttt{Spacecraft} dialog box. However, because the line \texttt{Sat.X = 1000} appears after the \texttt{BeginMissionSequence} command, the line \texttt{Sat.X = 1000} will appear as an assignment command in the \texttt{Mission} tree in the GUI.

**Basic Script Syntax Rules**

- Each script file must contain one and only one \texttt{BeginMissionSequence} command.
- GMAT commands are not allowed before the \texttt{BeginMissionSequence} command.
- You cannot use inline math statements (equations) before the \texttt{BeginMissionSequence} command in a script file. (GMAT considers in-line math statements to be an assignment command. You cannot use equations in the \texttt{Resources} tree, so you also cannot use equations before the \texttt{BeginMissionSequence} command.)
- In the GUI, you can only use in-line math statements in an assignment command. So, you cannot type $3000 + 4000$ or \texttt{Sat.Y - 8} in the text box for setting a spacecraft’s dry mass.
- GMAT's script language is case-sensitive.

**Data and Configuration**

Below we discuss the files and data that are distributed with GMAT and are required for GMAT execution. GMAT uses many types of data files, including planetary ephemeris files, Earth orientation data, leap second files, and gravity coefficient files. This section describes how these files are organized and the controls provided to customize them.

**File Structure**

The default directory structure for GMAT is broken into eight main subdirectories, as shown in Figure 4. These directories organize the files and data used to run GMAT, including binary libraries, data files, texture maps, and 3D models. The only two files in the GMAT root directory are \texttt{license.txt}, which contains the text of the NASA Open Source Agreement, and \texttt{README.txt}, which contains user information for the current GMAT release. A summary of the contents of each subdirectory is provided in the sections below.
Introduction

Getting Started

11

Figure 4. GMAT Root Directory Structure

**bin**

The *bin* directory contains all binary files required for the core functionality of GMAT. These libraries include the executable file (*GMAT.exe* on Windows, *GMAT.app* on the Mac, and *GMAT* on Linux) and platform-specific support libraries. The *bin* directory also contains two text files: *gmat_startup_file.txt* and *gmat.ini*. The startup file is discussed in detail in a separate section below. The *gmat.ini* file is used to configure some GUI panels, set paths to external web links, and define GUI tooltip messages.

**data**

The *data* directory contains all required data files to run GMAT and is organized according to data type, as shown in Figure 5 and described below.

Figure 5. GMAT Data Directory Structure

The *graphics* directory contains data files for GMAT’s visualization utilities, as well as application icons and images. The *splash* directory contains the GMAT splash screen that is displayed briefly...
while GMAT is initializing. The **stars** directory contains a star catalogue used for displaying stars in 3D graphics. The texture folder contains texture maps used for the 2D and 3D graphics resources. The **icons** directory contains graphics files for icons and images loaded at run time, such as the GMAT logo and GUI icons.

The **gravity** directory contains gravity coefficient files for each body with a default non-spherical gravity model. Within each directory, the coefficient files are named according to the model they represent, and use the extension `.cof`.

The **gui_config** directory contains files for configuring some of the GUI dialog boxes for GMAT resources and commands. These files allow you to easily create a GUI panel for a user-provided plugin, and are also used by some of the built-in GUI panels.

The **planetary_coeff** directory contains the Earth orientation parameters (EOP) provided by the International Earth Rotation Service (IERS) and nutation coefficients for different nutation theories.

The **planetary_ephem** directory contains planetary ephemeris data in both DE and SPK formats. The **de** directory contains the binary digital ephemeris DE405 files for the 8 planets, the Moon, and Pluto developed and distributed by JPL. The **spk** directory contains the DE421 SPICE kernel and kernels for selected comets, asteroids and moons. All ephemeris files distributed with GMAT are in the little-endian format.

The **time** directory contains the JPL leap second kernel `naif0009.tls` and the GMAT leap second file `tai-utc.dat`.

The **vehicle** directory contains ephemeris data and 3D models for selected spacecraft. The **ephem** directory contains SPK ephemeris files, including orbit, attitude, frame, and time kernels. The **models** directory contains 3D model files in 3DS or POV format for use by GMAT's **OrbitView** visualization resource.

**docs**


**extras**

The **extras** directory contains various extra convenience files that are helpful for working with GMAT but aren't part of the core codebase. The only file here so far is a syntax coloring file for the GMAT scripting language in the Notepad++ text editor.

**matlab**

The **matlab** directory contains M-files required for GMAT’s MATLAB interfaces, including the interface to the fmincon optimizer and interfaces for driving GMAT from MATLAB. All files in the **matlab** directory and its subdirectories must be included in your MATLAB path for the MATLAB interfaces to function properly.
output

The output directory is the default location for file output such as ephemeris files and report files. If no path information is provided for reports or ephemeris files created during a GMAT session, then those files will be written to the output folder.

plugins

The plugins directory contains optional plugins that are not required for use of GMAT. The proprietary directory is used for third-party libraries that cannot be distributed freely and is an empty folder in the open source distribution.

samples

The samples directory contains over 30 sample missions, ranging from a Hohmann transfer to libration point station-keeping to Mars B-plane targeting. These files are intended to demonstrate GMAT’s capabilities and to provide you with a potential starting point for building common mission types for your application and flight regime. Samples with specific requirements are located in subdirectories such as NeedMatlab and NeedVF13ad.

userfunctions

The userfunctions directory contains GMAT and MATLAB functions that are included in the GMAT distribution. You can also store your own custom GMAT and MATLAB functions in these folders.

Configuring GMAT Data Files

You can configure the data files GMAT loads at run time by editing the gmat_startup_file.txt file located in the bin directory. The startup file contains path information for data files such as ephemeris, Earth orientation parameters and graphics files. By editing the startup file, you can customize which files are loaded and used during a GMAT session. Below we describe the customization features available in the startup file. The order of lines in the startup file does not matter.

Leap Second and EOP files

GMAT reads several files that are used for high fidelity modelling of time and coordinate systems: the leap second files and the Earth orientation parameters (EOP) provided by the IERS. The EOP file is updated daily by the IERS. To update your local file with the latest data, simply replace the file eopc04_08.62-now in the data/planetary_coeff directory. Updated versions of this file are available from the IERS.

There are two leap second files provided with GMAT in the data/time directory. The naif0009.tls file is used by the JPL SPICE libraries when computing ephemerides. When a new leap second is added, you can replace this file with the new file from NAIF. GMAT reads the tai-utc.dat file for all time computations requiring leap seconds that are not performed by the SPICE utilities. When a new leap second is added, you can replace this file with the new file from the US Naval Observatory. In addition, you can modify the file if a new leap second is added by simply
duplicating the last row and updating it with the correct information for the new leap second. For example, if a new leap second were added on 01 Jul 2013, you would add the following line to the bottom of tai-utc.dat:

2013 JUL 1 =JD 2456474.5 TAI-UTC = 35.0 S + (MJD - 41317.) X 0.0

**Loading Custom Plugins**

Custom plugins are loaded by adding a line to the startup file (bin/gmat_startup_file.txt) specifying the name and location of the plugin file. In order for a plugin to work with GMAT, the plugin library must be placed in the folder referenced in the startup file. You specify the path to a plugin file using the "PLUGIN" keyword and specify the file by providing its name without the file extension (.dll on Windows). For example, to load a Windows plugin named libVF13Optimizer.dll located in the plugins/proprietary directory, you would add this line to your startup file:

`PLUGIN = ../plugins/proprietary/libVF13Optimizer`

**User-defined Function Paths**

If you create custom GMAT or MATLAB functions, you can provide the path to those files and GMAT will locate them at run time. The default startup file is configured so you can place GMAT function files (with a .gmf extension) in the userfunctions/gmat directory and place MATLAB functions (with a .m extension) in the userfunctions/matlab directory. GMAT automatically searches those locations at run time. You can change the location of the search path to your GMAT or MATLAB functions by changing these lines in your startup file to reflect the location of your files with respect to the GMAT bin folder:

```plaintext
GMAT_FUNCTION_PATH = ../userfunctions/gmat
MATLAB_FUNCTION_PATH = ../userfunctions/matlab
```

If you wish to organize your custom functions in multiple folders, you can add multiple search paths to the startup file. For example,

```plaintext
GMAT_FUNCTION_PATH = ../MyFunctions/utils
GMAT_FUNCTION_PATH = ../MyFunctions/StateConversion
GMAT_FUNCTION_PATH = ../MyFunctions/TimeConversion
```

GMAT will search the paths in the order specified in the startup file and will use the first function with a matching name.

**Configuring the MATLAB Interface**

GMAT features a MATLAB interface that allows you to run MATLAB functions from within GMAT.

This interface is packaged as an optional GMAT plugin. To use it, make sure the following line is present in your gmat_startup_file.txt and has no comment symbol (#) in front of it.

`PLUGIN = ../plugins/libMatlabInterface`

The MATLAB interface must be able to find your MATLAB installation. The procedure for setting this information varies by platform.
Windows

On Windows, MATLAB must be properly configured in two places: the system Path variable and the Windows registry. Both locations must be configured for the same MATLAB version.

1. The following directories must exist in your system’s Path variable, where <MATLAB> is the path to the MATLAB root directory:

   `<MATLAB>/bin/win32`
   `<MATLAB>/bin`

   If you have multiple versions of MATLAB installed, GMAT will use the one that appears first in the system path.

   **Caution**

   For some versions of MATLAB (e.g. R2010a), MATLAB and Windows are distributed with libraries that have the same name, resulting in a conflict. As a workaround, you may need to place the folders above at the beginning of your system path.

2. When you install MATLAB, it automatically registers itself as a COM server in the Windows registry. If you have multiple versions of MATLAB installed, it may be necessary to re-register a certain version manually. This can be done by running the following command. This may require administrator privileges.

   `matlab.exe -regserver`

3. Add GMAT’s MATLAB files to your MATLAB path. This can be done by placing the following line in a file named `startup.m` in your user MATLAB directory, where `<GMAT>` is the path to your GMAT root directory.

   ```matlab
   addpath(genpath(''<GMAT'>''/matlab'));
   ```

Mac OS X

On Mac OS X, to use MATLAB with GMAT, you must set the MATLABFORGMAT environment variable in your `environment.plist` file, located in the `.MacOSX` directory in your home folder. This environment variable should point to the location of your MATLAB installation (application bundle). GMAT will not interface with MATLAB unless this environment variable is set.

The current Mac application includes the ability to make calls to MATLAB functions from within GMAT, but does not support calls MATLAB to GMAT (including the `fmincon` optimizer).

Note that when GMAT opens MATLAB, it will open X11 first (as is required for MATLAB execution). GMAT currently does not automatically close X11 after quitting MATLAB, so you will need to quit X11 manually.

To add the environment variable:

1. If the `environment.plist` file already exists in your `.MacOSX` directory, edit the file using the Property List Editor to add the MATLABFORGMAT variable and set it to point
to the location of your MATLAB application (e.g. /Applications/MATLAB_R2010a/MATLAB_R2010a.app).

2. If you do not have an environment.plist file in your .MacOSX directory, open a Terminal window and follow these steps:
   1. Create the .MacOSX directory as a directory in your home folder (if it does not exist).
   2. Open the Property List Editor and create the MATLABFORGMAT variable as described above.
   3. Save the property list as environment.plist in the .MacOSX directory.

You must logout and log back in for this to take effect.

Other Resources

If you have further questions, need help with using GMAT, or want to provide feedback, here are some additional resources:

- Official Homepage: http://gmat.gsfc.nasa.gov
- Mailing Lists and Project Resources: http://sourceforge.net/projects/gmat
- Blog: http://gmat.sourceforge.net/blog
- Documentation: http://gmat.sourceforge.net/docs
- Bug Tracker: http://pows003.gsfc.nasa.gov/bugzilla
- Official Contact: <gmat@gsfc.nasa.gov>
# Part II. Creating Your First Mission

## Table of Contents

- Simulating an Orbit ......................................................................................................................... 18
  - Objective and Overview .............................................................................................................. 18
  - Configure the Spacecraft ........................................................................................................ 18
  - Configure the Propagator ........................................................................................................ 20
  - Configure the Propagate Command ......................................................................................... 21
  - Run and Analyze the Results .................................................................................................. 22
Simulating an Orbit

Objective and Overview

Note

The most fundamental capability of GMAT is to propagate, or simulate the orbital motion of, spacecraft. The ability to propagate spacecraft is used in nearly every practical aspect of space mission analysis, from simple orbital predictions (e.g. When will the International Space Station be over my house?) to complex analyses that determine the thruster firing sequence required to send a spacecraft to the Moon or Mars.

This tutorial will teach you how to use GMAT to propagate a spacecraft. You will learn how to configure Spacecraft and Propagator resources, and how to use the Propagate command to propagate the spacecraft to orbit periapsis, which is the point of minimum distance between the spacecraft and Earth. The basic steps in this tutorial are:

1. Configure a Spacecraft and define its epoch and orbital elements.
2. Configure a Propagator.
3. Modify the default OrbitView plot to visualize the spacecraft trajectory.
4. Modify the Propagate command to propagate the spacecraft to periapsis.
5. Run the mission and analyze the results.

Configure the Spacecraft

In this section, you will rename the default Spacecraft and set the Spacecraft’s initial epoch and classical orbital elements. You’ll need GMAT open, with the default mission loaded. To load the default mission, click New Mission or start a new GMAT session.

Rename the Spacecraft

1. In the Resources tree, right-click DefaultSC and click Rename.
2. Type Sat.
3. Click OK.

Set the Spacecraft Epoch

1. In the Resources tree, double-click Sat. Click the Orbit tab if it is not already selected.
2. In the Epoch Format list, select UTCGregorian. You’ll see the value in the Epoch field change to the UTC Gregorian epoch format.
3. In the Epoch box, type 22 Jul 2014 11:29:10.811. This field is case-sensitive.
4. Click Apply or press the ENTER key to save these changes.
Set the Keplerian Orbital Elements

1. In the **StateType** list, select **Keplerian**. In the **Elements** list, you will see the GUI reconfigure to display the Keplerian state representation.
2. In the **SMA** box, type **83474.318**.
3. Set the remaining orbital elements as shown in the table below.

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ECC</td>
<td>0.89652</td>
</tr>
<tr>
<td>INC</td>
<td>12.4606</td>
</tr>
<tr>
<td>RAAN</td>
<td>292.8362</td>
</tr>
<tr>
<td>AOP</td>
<td>218.9805</td>
</tr>
<tr>
<td>TA</td>
<td>180</td>
</tr>
</tbody>
</table>

4. Click **OK**.
5. Click **Save**. If this is the first time you have saved the mission, you’ll be prompted to provide a name and location for the file.

![Figure 6. Spacecraft State Setup](image-url)
Configure the Propagator

In this section you'll rename the default Propagator and configure the force model.

**Rename the Propagator**

1. In the Resources tree, right-click DefaultProp and click Rename.
2. Type LowEarthProp.
3. Click OK.

**Configure the Force Model**

For this tutorial you will use an Earth 10×10 spherical harmonic model, the Jacchia-Roberts atmospheric model, solar radiation pressure, and point mass perturbations from the Sun and Moon.

1. In the Resources tree, double-click LowEarthProp.
2. Under Gravity, in the Degree box, type 10.
3. In the Order box, type 10.
4. In Atmosphere Model list, click JacchiaRoberts.
5. Click the Select button next to the Point Masses box. This opens the CelesBodySelectDialog window.
6. In the Available Bodies list, click Sun, then click -> to add Sun to the Selected Bodies list.
7. Add the moon (named Luna in GMAT) in the same way.
8. Click OK to close the CelesBodySelectDialog.
9. Select Use Solar Radiation Pressure to toggle it on. Your screen should now match Figure 7.
10. Click OK.

![Image](PropSetup - LowEarthProp.png)

*Figure 7. Force Model Configuration*
Configuring the Orbit View Plot

Now you will configure an OrbitView plot so you can visualize Sat and its trajectory. The orbit of Sat is highly eccentric. To view the entire orbit at once, we need to adjust the settings of DefaultOrbitView.

1. In the Resources tree, double-click DefaultOrbitView.
2. In the three boxes to the right of ViewPointVector, type the values -60000, 30000, and 20000 respectively.
3. In the Drawing Options list, clear Draw XY Plane. Your screen should now match Figure 8.
4. Click OK.

Configure the Propagate Command

This is the last step before running the mission. Below you will configure a Propagate command to propagate (or simulate the motion of) Sat to orbit periapsis.

1. Click the Mission tab to display the Mission tree.
2. Double-click Propagate1.
3. Under Stopping Conditions, click the (...) button next to Sat.ElapsedSecs. This will display the ParameterSelectDialog window.
4. In the **Object List** box, click **Sat** if it is not already selected. This directs GMAT to associate the stopping condition with the spacecraft **Sat**.

5. In the **Object Properties** list, double-click **Periapsis** to add it to the **Selected Values** list. This is shown in Figure 9.

![Figure 9. Propagate Command ParameterSelectDialog Configuration](image)

6. Click **OK**. Your screen should now match Figure 10.

7. Click **OK**.

![Figure 10. Propagate Command Configuration](image)

### Run and Analyze the Results

Congratulations, you have now configured your first GMAT mission and are ready to run the mission and analyze the results.

1. Click **Save** to save your mission.
2. Click the **Run**.
Creating Your First Mission

Simulating an Orbit

You will see GMAT propagate the orbit and stop at orbit periapsis. Figure 11 illustrates what you should see after correctly completing this tutorial. Here are a few things you can try to explore the results of this tutorial:

1. Manipulate the DefaultOrbitView plot using your mouse to orient the trajectory so that you can to verify that at the final location the spacecraft is at periapsis. See Manipulating the 3D Orbit View for details.
2. Display the command summary:
   1. Click the Mission tab to display the Mission tree.
   2. Right-click Propagate1 and select Command Summary to see data on the final state of Sat.
   3. Use the Coordinate System list to change the coordinate system in which the data is displayed.
3. Click Start Animation to animate the mission and watch the orbit propagate from the initial state to periapsis.

Figure 11. Orbit View Plot after Mission Run
Part III. Common Tasks

Table of Contents

Configuring a Spacecraft .................................................................................................................. 25
  Setting the Initial Epoch .............................................................................................................. 25
  Configuring the Orbit ................................................................................................................. 25
  Configuring Physical Properties ................................................................................................. 26
  Configuring the Attitude (Fixed) ................................................................................................. 27
  Configuring the Attitude (Spinner) ............................................................................................. 27
Propagating a Spacecraft .................................................................................................................. 29
  Configuring the Force Model ...................................................................................................... 29
  Configuring the Force Model: Mars ............................................................................................ 29
  Propagating for a Duration ........................................................................................................ 30
  Propagating to an Orbit Condition ............................................................................................. 31
Reporting Data .................................................................................................................................. 32
  Reporting Data During a Propagation Span ............................................................................... 32
  Reporting Data at a Specific Mission Event .............................................................................. 32
  Creating a CCSDS Ephemeris File .............................................................................................. 33
  Creating an SPK Ephemeris File ............................................................................................... 33
Visualizing Data ................................................................................................................................ 35
  Manipulating the 3D Orbit View ................................................................................................. 35
  Configuring the Ground Track Plot ......................................................................................... 35
  Creating a 2D Plot ..................................................................................................................... 35
Configuring a Spacecraft

Setting the Initial Epoch

You can configure the initial epoch of a spacecraft in several time systems (TAI, TDB, UTC, etc) and formats (Gregorian, modified Julian). To set the epoch in UTC Gregorian, follow these steps starting from the default mission:

1. In the Resources tree, double-click DefaultSC to open its properties window.
2. Click the Orbit tab if it isn't already selected.
3. In the EpochFormat list, select UTCGregorian.
4. In the Epoch box, type 04 Jul 2014 09:30:15.235. This field is case-sensitive, and must be entered in the exact format shown.
5. Click OK or Apply to save your changes.

The GMAT script for the epoch settings configured above is:

```
Create Spacecraft DefaultSC
DefaultSC.DateFormat = UTCGregorian
DefaultSC.Epoch = '04 Jul 2014 09:30:15.235'
```

Configuring the Orbit

You can set the orbit of a spacecraft in several representations, such as Keplerian and Cartesian, and in any of the default or user-created coordinate systems. Starting from the default mission, first set the initial epoch:

1. In the Resources tree, right-click on DefaultSC and click Rename.
2. In the Rename box type ISS and click OK.
3. In the Resources tree, double-click ISS to open its properties window.
4. Click the Orbit tab if it isn't already selected.
5. In the Epoch Format list, click UTCGregorian.

Now set the orbital state for ISS:

1. In the State Type list, click Keplerian.
2. In the SMA box, type 6771.907.
3. In the ECC box, type 0.00103.
4. In the INC box, type 51.597.
5. In the RAAN box, type 244.300.
6. In the AOP box, type 353.735.
7. In the TA box, type 199.683.
8. Click OK.

The GMAT script for the spacecraft state configured above is:

```
Create Spacecraft ISS
ISS.DateFormat = UTCGregorian
```
Common Tasks

Configuring Physical Properties

GMAT allows you to set the physical properties of a spacecraft, such as the mass and area. Starting from the default mission:

1. In the Resources tree, double-click on DefaultSC to open its properties window.
2. Click the Ballistic/Mass tab.
3. In the Dry Mass box, type 450.
4. In the Coefficient of Drag box, type 2.0.
5. In the Coefficient of Reflectivity box, type 1.7.
6. In the Drag Area box, type 10.5.
7. In the SRP Area box, type 12.5.
8. Click OK or Apply to save your changes.

The script for the physical settings configured above is shown below.
Configuring the Attitude (Fixed)

GMAT can model a spacecraft with an attitude fixed in any defined coordinate system, including user-defined systems. This can be used to model nadir-pointing or inertially-pointed spacecraft.

For example, follow these instructions to set the attitude of the default spacecraft using Euler angle rotations from the built-in EarthMJ2000Eq inertial coordinate system. Starting from the default mission:

1. In the Resources tree, double-click DefaultSC to open its properties window.
2. Click the Attitude tab.
3. In the Attitude Model list, select CoordinateSystemFixed.
4. In the Coordinate System list, select EarthMJ2000Eq.
5. In the Attitude Initial Conditions area, in the Attitude State Type box, select Euler Angles.
6. In the Euler Angle 1 box, type 123.
7. In the Euler Angle 2 box, type 45.
8. In the Euler Angle 3 box, type 157.
9. Click Run. The spacecraft should now be inertially pointed in the graphics window.

Configuring the Attitude (Spinner)

GMAT has a special attitude model that makes it easy to set up a spacecraft that spins about the axes of any defined coordinate system. The steps below define a spacecraft-centered coordinate system with axes rotating with the Sun-Earth line, then define a spacecraft as spinning about the X-axis of that system. Starting from the default mission:

First, define the Solar coordinate system:

1. In the Resources tree, right click Coordinate Systems and click Add Coordinate System.
2. In the Coordinate System Name box, type Solar.
3. In the Origin list, select DefaultSC.
4. In the Type list, select ObjectReferenced.
5. Set the Primary body to Sun, and the Secondary body to Earth.
6. Set the X axis to -R and the Z axis to N. Leave the Y axis at its default blank value.

Now set the default spacecraft to spin about the X-axis of the Solar coordinate system:

1. In the Resources tree, double-click DefaultSC to open its properties window.
2. Click the Attitude tab.
3. In the Attitude Model list, select Spinner. This enables the Attitude Rate Initial Conditions properties.
4. In the Euler Angle Sequence box, select 123. This maps the first Euler angle rotation to the X-axis of the coordinate system.
5. In the **Attitude Rate State Type** list, select **EulerAngleRates**.
6. In the **Euler Angle Rate 1** box, type **180**.
7. Click **Run**. In the graphics window, you will see the spacecraft spinning about the Sun-Earth line.
# Propagating a Spacecraft

## Configuring the Force Model

Propagation in GMAT is governed by a force model and an integrator, which together form a propagator. The force model component offers many options that let you control the fidelity of the simulation, from a simple two-body model with a single point mass to a full-featured model with central body non-spherical gravity, external point masses, atmospheric drag, and more. This example configures an Earth-centered model with some commonly-used parameters. Starting from the default mission:

1. In the **Resources** tree, double-click **DefaultProp** to open its properties window.
2. Under **Gravity**, in the **Degree** box, type **21**.
3. In the **Order** box, type **21**.
4. In the **Atmosphere Model** list, click **MSISE90**.
5. Next to the **PointMasses** box, click the **Select** button.
6. In the **Available Bodies** list on the left, click **Sun**, then click **->** to add **Sun** to your force model.
7. Add **Luna** (Earth's moon) and **Jupiter** using the same steps as above.
8. Click **OK** to accept your changes.
9. Select **Use Solar Radiation Pressure** to activate the solar radiation pressure force.
10. Click **OK** to accept your changes.

The script for the force model configured above is shown below.

```plaintext
Create ForceModel DefaultProp_ForceModel
DefaultProp_ForceModel.CentralBody = Earth
DefaultProp_ForceModel.PrimaryBodies = {Earth}
DefaultProp_ForceModel.PointMasses = {Jupiter, Luna}
DefaultProp_ForceModel.Drag.AtmosphereModel = 'MSISE90'
DefaultProp_ForceModel.SRP = On
DefaultProp_ForceModel.GravityField.Earth.Degree = 21
```

## Configuring the Force Model: Mars

Creating a high-fidelity propagator for Mars is a bit more complex than for Earth or the Moon. GMAT does not by default include Phobos and Deimos as celestial bodies, so these must be added manually to include them in the force model. This example shows how to do this, starting from the default mission.

This example requires an external ephemeris file for Mars' moons. For the following steps to work, download this file from this link: **mar085.bsp**.

First, create Phobos as a **Moon** resource:

1. In the **Resource** tree, expand the **SolarSystem** folder to display all default celestial bodies.
2. Right-click **Mars**, point to **Add**, and click **Moon**.
3. Type **Phobos** and click **OK**.
4. Double-click the new **Phobos** resource to edit its properties.
5. Set the following properties to the values shown. These are actual values for Phobos:

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mu</td>
<td>0.00070934</td>
</tr>
<tr>
<td>Equatorial Radius</td>
<td>13.5</td>
</tr>
<tr>
<td>Flattening</td>
<td>0.3185185185185186</td>
</tr>
</tbody>
</table>

6. Click the **Orbit** tab.
7. In the **Ephemeris Source** list, make sure **SPICE** is selected. This is currently the only available option.
8. In the **NAIF ID** box, type **401**. This is the international identification number for Phobos.
9. Click the **Add** button and choose the file **mar085.bsp** that you downloaded above.
10. Click **OK** to accept these changes.

Now follow the same steps to add Deimos. The only differences are the following values:

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Name</strong></td>
<td>Deimos</td>
</tr>
<tr>
<td><strong>Mu</strong></td>
<td>0.000158817</td>
</tr>
<tr>
<td><strong>Equatorial Radius</strong></td>
<td>7.5</td>
</tr>
<tr>
<td><strong>Flattening</strong></td>
<td>0.3066666666666666</td>
</tr>
<tr>
<td><strong>NAIF ID</strong></td>
<td>402</td>
</tr>
</tbody>
</table>

Now create the propagator itself:

1. In the **Resources** tree, in the **Propagators** folder, double-click **DefaultProp** to edit its properties.
2. In the **Central Body** list, click **Mars**.
3. In the **Primary Body** list, click **Mars**.
4. Under **Gravity**, in the **Model** list, click **Mars-50C**.
5. In the **Degree** and **Order** boxes, type **4** and **4**, respectively.
6. Next to the **Point Masses** box, click **Select**.
7. In the **Available Bodies** list, click **Jupiter**, **Sun**, **Phobos**, and **Deimos**, and click the right arrow to transfer them to the **Selected Bodies** list.
8. Click **OK** to accept this list.
9. Check **Use Solar Radiation Pressure** to enable this force.
10. Click **OK** to accept these changes.

## Propagating for a Duration

**GMAT** can propagate a spacecraft for a duration of time, such as 60 seconds, 10 days, or one orbit period. Starting from the default mission:

1. Click the **Mission** tab to show the **Mission** tree.
2. Double-click **Propagate1**. The default mission is configured to propagate the **DefaultSC** spacecraft for 12000 seconds.
3. In the **Parameter** column, to the left of **DefaultSC.ElapsedSecs**, click …. This will display a window allowing you to choose a new type of duration parameter.
4. In the **Object Properties** list, click **ElapsedDays**, then click -> to add it to the **Selected Value(s)** list.
5. Click **OK**.
6. In the **Condition** column, double-click the value 0.0 and enter 10 instead.
7. Click **OK**, then click **Run**.

GMAT will propagate the spacecraft for 10 days. Observe the 2D and 3D plots showing the propagated orbit.

**Propagating to an Orbit Condition**

GMAT can propagate a spacecraft to a specific orbit condition, such as periapsis, an altitude value, or a latitude value. Starting from the default mission:

1. Click the **Mission** tab to show the **Mission** tree.
2. Double-click **Propagate1**. The default mission is configured to propagate the **DefaultSC** spacecraft for 12000 seconds.
3. In the **Parameter** column, to the left of **DefaultSC.ElapsedSecs**, click …. This will display a window allowing you to choose a new type of duration parameter.
4. In the **Object Properties** list, click **Periapsis**.
5. In the **Central Body** list, make sure **Earth** is selected. Then click -> to add it to the **Selected Value(s)** list.
6. Click **OK** to close the **ParameterSelectDialog** window.
7. Click **OK**, then click **Run**.

GMAT will propagate the spacecraft until it reaches orbit periapsis.
Reporting Data

GMAT provides several ways to report mission data (such as altitude or ΔV values) to plain text files. GMAT can report data at each integration time step in the mission or at specific mission events, such as periapsis passage. The report functionality is controlled via the ReportFile resource and the Report and Toggle commands.

Reporting Data During a Propagation Span

You can report data at each integration step in the mission sequence by creating a ReportFile resource and adding data to it. Starting from the default mission:

1. On the Resources tree, right-click the Output folder, point to Add, and click ReportFile.
2. Double-click ReportFile1 to open its properties window.
3. In the Parameter List area, click Edit.
4. In the Selected Value(s) list, click DefaultSC.EarthMJ2000.X and click <- to remove it from the list.
5. In the Object Properties list, click Altitude and click -> to add it to the Selected Value(s) list.
6. Add DefaultSC.A1ModJulian to the Selected Value(s) list if it doesn’t already exist.
7. Click OK, then in the ReportFile - ReportFile1 window, click OK again.

The script for the report data configured above is shown below.

Create ReportFile ReportFile1;
GMAT ReportFile1.Add = {DefaultSC.A1ModJulian, DefaultSC.Earth.Altitude};

Reporting Data at a Specific Mission Event

You can report data to a ReportFile at a specific time (for example, at orbit apoapsis) using the Report command. Starting from the default mission, first configure the ReportFile resource:

1. In the Resources tree, right-click on the Output folder, point to Add, and click ReportFile.
2. In the Output folder, double-click ReportFile1 to edit its properties.
3. In the Parameter List area, click Edit.
4. In the Selected Values list, click DefaultSC.EarthMJ2000Eq.X and click <- to remove it from the list.
5. Remove DefaultSC.A1ModJulian from the Selected Value(s) list in the same way.
6. Click OK to close the ParameterSelectDialog window and click OK again to close the ReportFile1 window.

Now configure the Propagate1 command to propagate to orbit apoapsis:

1. Click the Mission tab to display the Mission tree.
2. In the Mission tree, double-click Propagate1 to edit its properties.
3. Under Stopping Conditions, click the ... button to the left of DefaultSC.ElapsedSecs.
4. In the Object Properties list, click Apoapsis, then click the -> button to add it to the Selected Value(s) list.
5. Click OK to close the ParameterSelectDialog window, then click OK again to close the Propagate1 window.

Finally, add a Report command:

1. In the Mission tree, right-click Propagate1, point to Insert After, and click Report.
2. Double-click Report1 to edit its properties, then click the View button.
3. Click the <= button to remove all items from the Selected Value(s) list.
4. In the Object Properties list, click TA, then click the -> button to add it to the Selected Value(s) list.
5. Add Altitude to the Selected Value(s) list in the same way.
6. Click OK to close the ParameterSelectDialog window, then click OK to close the Report1 window.
7. Click Run to run the mission.
8. Click the Output tab to show the Output tree.
9. In the Reports folder, double-click ReportFile1 to see the requested data.

The script for the report data configured above is shown below.

```plaintext
Create ReportFile ReportFile1
BeginMissionSequence
Propagate DefaultProp(DefaultSC) {DefaultSC.Earth.Apoapsis}
Report ReportFile1 DefaultSC.Earth.TA DefaultSC.Earth.Altitude
```

**Creating a CCSDS Ephemeris File**

The CCSDS Orbit Ephemeris Message (OEM) is a standardized text-based ephemeris format. In GMAT, you can easily create an OEM file with your desired interpolation order and data frequency. Starting from the default mission:

1. In the Resources tree, right-click the Output folder, point to Add, and click EphemerisFile. A new resource called EphemerisFile1 appears in the tree.
2. Double-click EphemerisFile1 to open it.
3. Make sure that in the File Format list, CCSDS-OEM is selected.
4. Click Ok.
5. Click Run. The OEM file will be written to a file named EphemerisFile1.eph in GMAT's output folder. By default, this folder is <GMAT>/output, where <GMAT> is the path to your GMAT installation.

**Creating an SPK Ephemeris File**

An SPK ephemeris is a binary file format used by the SPICE Toolkit created by NAIF. GMAT can write spacecraft state information to this format using your desired interpolation order and data frequency. Starting from the default mission:

1. In the Resources tree, right-click the Output folder, point to Add, and click EphemerisFile. A new resource called EphemerisFile1 appears in the tree.
2. Double-click EphemerisFile1 to open it.
3. In the **File Format** list, click **SPK**.
4. In the **File Name** box, replace the default value with **EphemerisFile1.bsp**. An SPK ephemeris requires the `.bsp` extension.
5. Click **Ok**.
6. Click **Run**. The SPK file will be written to a file named **EphemerisFile1.bsp** in GMAT's output folder.
Visualizing Data

Manipulating the 3D Orbit View

GMAT’s OrbitView resource offers a three-dimensional realistic view of your mission trajectory in any coordinate system or viewpoint you choose. The view itself can be manipulated using the mouse. Starting from the default mission:

1. Click Run. This will run the mission and will result in a DefaultOrbitView window and a DefaultGroundTrackPlot window on the GMAT desktop. The default view is centered at the Earth, in an Earth-centered inertial reference frame.
2. With the left mouse button, drag in the DefaultOrbitView window. This will rotate the view about the center of the active coordinate system (in this case, the center of the Earth).
3. With the right mouse button, drag left-to-right. This will zoom the view out from the center of the active coordinate system. Dragging right-to-left will zoom the view in.
4. With the wheel button (or middle button), drag up and down. This will rotate the view about an axis perpendicular to the screen.

Configuring the Ground Track Plot

GMAT’s ground track plot can display one or more spacecraft on a two-dimensional map of a celestial body. You can choose which spacecraft are displayed, and which celestial body to use. Keeping the Earth as the central body, let’s add a second spacecraft to the default plot. Starting with the default mission, first add a new spacecraft:

1. Right-click the Spacecraft folder and click Add Spacecraft to add Spacecraft1.
2. In the Mission tree, double-click Propagate1.
3. Under Spacecraft List, click ... to the left of DefaultSC.
4. In the Available SpaceObject list, click Spacecraft1 and click the -> button to move it to the SpaceObject Selected list. Then click OK. This adds Spacecraft1 to the Spacecraft List for Propagate1.
5. Click Apply, then click OK.

Then add the new spacecraft to the ground track plot:

1. In the Resources tree, in the Output folder, double-click DefaultGroundTrackPlot.
2. In the Selected Objects list, select Spacecraft1.
3. Click OK, then click Run.

After the run is complete, the DefaultGroundTrackPlot window will show the trajectory of the default spacecraft and Spacecraft1 on a map of Earth.

Creating a 2D Plot

GMAT offers an XYPlot resource that allows you to visualize the relationship between multiple parameters (for example, orbit altitude and time). Starting from the default mission:

1. In the Resources tree, right-click the Output folder, point to Add, and click XYPlot.
2. Double-click the new **XYPlot1** resource. The default y-axis parameter is the Cartesian "X" position of the spacecraft.
3. Click **Edit Y** to change the y-axis parameter.
4. In the **Selected Value(s)** list, click **DefaultSC.EarthMJ2000Eq.X** and click `<-` to remove it from the list.
5. In the **Object Properties** list, click **Altitude**.
6. In the **Central Body** list, make sure **Earth** is selected, then click `->` to add it to the **Selected Value(s)** list.
7. Click **OK** in the **ParameterSelectDialog** window, then click **OK** again in the **XYPlot - XY-Plot1** window.
8. Click **Run**.

A new plot window will appear.
Part IV. Tutorials

Table of Contents

Simple Orbit Transfer ...................................................................................................................... 38
  Objective and Overview ............................................................................................................. 38
  Configure Maneuvers, Differential Corrector, and Graphics ......................................................... 38
  Configure the Mission Sequence ............................................................................................... 39
  Run the Mission ..................................................................................................................... 46
Simple Orbit Transfer

Audience
Beginner

Length
30 minutes

Prerequisites
Complete Simulating an Orbit

Script File
Tut_SimpleOrbitTransfer.script

Objective and Overview

Note
One of the most common problems in space mission design is to design a transfer from one circular orbit to another circular orbit that lie within the same orbital plane. Circular coplanar transfers are used to raise low-Earth orbits that have degraded due to the effects of atmospheric drag. They are also used to transfer from a low-Earth orbit to a geosynchronous orbit and to send spacecraft to Mars. There is a well known sequence of maneuvers, called the Hohmann transfer, that performs a circular, coplanar transfer using the least possible amount of fuel. A Hohmann transfer employs two maneuvers. The first maneuver raises the orbital apoapsis (or lowers orbital periapsis) to the desired altitude and places the spacecraft in an elliptical transfer orbit. At the apoapsis (or periapsis) of the elliptical transfer orbit, a second maneuver is applied to circularize the orbit at the final altitude.

In this tutorial, we will use GMAT to perform a Hohmann transfer from a low-Earth parking orbit to a geosynchronous mission orbit. This requires a targeting sequence to determine the required maneuver magnitudes to achieve the desired final orbit conditions. In order to focus on the configuration of the targeter, we will make extensive use of the default configurations for spacecraft, propagators, and maneuvers.

The target sequence employs two velocity-direction maneuvers and two propagation sequences. The purpose of the first maneuver is to raise orbit apoapsis to 42,165 km, the geosynchronous radius. The purpose of the second maneuver is to nearly circularize the orbit and yield a final eccentricity of 0.005. The basic steps of this tutorial are:

1. Create and configure a DifferentialCorrector resource.
2. Modify the DefaultOrbitView to visualize the trajectory.
3. Create two ImpulsiveBurn resources with default settings.
4. Create a Target sequence to (1) raise apoapsis to geosynchronous altitude and (2) circularize the orbit.
5. Run the mission and analyze the results.

Configure Maneuvers, Differential Corrector, and Graphics

For this tutorial, you'll need GMAT open, with the default mission loaded. To load the default mission, click New Mission or start a new GMAT session. We will use the default configurations for the spacecraft (DefaultSC), the propagator (DefaultProp), and the two maneuvers. DefaultSC is configured by default to a near-circular orbit, and DefaultProp is configured to use Earth as the
central body with a nonspherical gravity model of degree and order 4. You may want to open the dialog boxes for these objects and inspect them more closely as we will leave them at their default settings.

**Create the Differential Corrector**

The Target sequence we will create later needs a DifferentialCorrector resource to operate, so let’s create one now. We’ll leave the settings at their defaults.

1. In the Resource tree, expand the Solvers folder if it isn’t already.
2. Right-click the Boundary Value Solvers folder, point to Add, and click DifferentialCorrector. A new resource called DC1 will be created.

**Modify the Default Orbit View**

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

1. In the Resource Tree, double-click DefaultOrbitView to edit its properties.
2. Set the values shown in the table below.

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Solver Iterations, under Drawing Option</td>
<td>Current</td>
</tr>
<tr>
<td>Axis, under View Up Defintion</td>
<td>X</td>
</tr>
<tr>
<td>View Point Vector boxes, under View Definition</td>
<td>0, 0, and 120000 respectively</td>
</tr>
</tbody>
</table>

3. Click OK to save these changes.

**Create the Maneuvers.**

We’ll need two ImpulsiveBurn resources for this tutorial, both using default values. Below, we’ll rename the default ImpulsiveBurn and create a new one.

1. In the Resources tree, right-click DefaultIB and click Rename.
2. In the Rename box, type T0I, an acronym for Transfer Orbit Insertion, and click OK.
3. Right-click the Burns folder, point to Add, and click ImpulsiveBurn.
4. Rename the new ImpulsiveBurn1 resource to G0I, an acronym for Geosynchronous Orbit Insertion.

**Configure the Mission Sequence**

Now we will configure a Target sequence to solve for the maneuver values required to raise the orbit to geosynchronous altitude and circularize the orbit. We’ll begin by creating an initial Propagate command, then the Target sequence itself, then the final Propagate command. To allow us to focus on the Target sequence, we’ll assume you have already learned how to propagate an orbit to a desired condition by working through the Simulating an Orbit tutorial.
Configure the Initial Propagate Command

1. Click on the Mission tab to show the Mission tree.
2. Configure Propagate1 to propagate to DefaultSC.Earth.Periapsis.
3. Rename Propagate1 to Prop To Periapsis.

Create the Target Sequence

Now create the commands necessary to perform the Target sequence. Figure 12 illustrates the configuration of the Mission tree after you have completed the steps in this section. We'll discuss the Target sequence after it has been created.

![Mission Sequence Diagram]

**Figure 12. Final Mission Sequence for the Hohmann Transfer**

To create the Target sequence:

1. In the Mission tree, right-click Prop To Periapsis, point to Insert After, and click Target. This will insert two separate commands: Target1 and EndTarget1.
2. Right-click Target1 and click Rename.
3. Type Hohmann Transfer and click OK.
4. Right-click Hohmann Transfer, point to Append, and click Vary.
5. Rename Vary1 to Vary TOI.
6. Complete the Target sequence by appending the commands in Table 3.

Table 3. Additional Target Sequence Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maneuver</td>
<td>Perform TOI</td>
</tr>
<tr>
<td>Propagate</td>
<td>Prop to Apoapsis</td>
</tr>
<tr>
<td>Achieve</td>
<td>Achieve RMAG = 42165</td>
</tr>
<tr>
<td>Vary</td>
<td>Vary GOI</td>
</tr>
<tr>
<td>Maneuver</td>
<td>Perform GOI</td>
</tr>
<tr>
<td>Achieve</td>
<td>Achieve ECC = 0.005</td>
</tr>
</tbody>
</table>

---

40
Note

Let’s discuss what the Target sequence does. We know that two maneuvers are required to perform the Hohmann transfer. We also know that for our current mission, the final orbit radius must be 42,165 km and the final orbital eccentricity must be 0.005. However, we don’t know the size (or ΔV magnitudes) of the maneuvers that precisely achieve the desired orbital conditions. You use the Target sequence to solve for those precise maneuver values. You must tell GMAT what controls are available (in this case, two maneuvers) and what conditions must be satisfied (in this case, a specific orbital radius and eccentricity). You accomplish this using the Vary and Achieve commands. Using the Vary command, you tell GMAT what to solve for—in this case, the ΔV values for TOI and GOI. You use the Achieve command to tell GMAT what conditions the solution must satisfy—in this case, the final orbital conditions.

Create the Final Propagate Command

We need a Propagate command after the Target sequence so that we can see our final orbit.

1. In the Mission tree, right-click EndTarget1, point to Insert After, and click Propagate. A new Propagate3 command will appear.
2. Rename Propagate3 to Prop one Day.
3. Double-click Prop One Day to edit its properties.
4. Under Condition, replace the value 12000.0 with 86400, the number of seconds in one day.
5. Click OK to save these changes.

Figure 13. Prop One Day Command Configuration

Configure the Target Sequence

Now that the structure is created, we need to configure the various parts of the Target sequence to do what we want.
Configure the Hohmann Transfer Command

1. Double-click **Hohmann Transfer** to edit its properties.
2. In the **ExitMode** list, click **SaveAndContinue**. This instructs GMAT to save the final solution of the targeting problem after you run it.
3. Click **OK** to save these changes.

![Figure 14. Hohmann Transfer Command Configuration](image)

Configure the Vary TOI Command

1. Double-click **Vary TOI** to edit its properties. Notice that the variable in the **Variable** box is **TOI.Element1**, which by default is the velocity component of TOI in the local Velocity-Normal-Binormal (VNB) coordinate system. That's what we need, so we'll keep it.
2. In the **Initial Value** box, type **1.0**.
3. In the **Max Step** box, type **0.5**.
4. Click **OK** to save these changes.

![Figure 15. Vary TOI Command Configuration](image)

Configure the Perform TOI Command

1. Double-click **Perform TOI** to edit its properties. Notice that the command is already set to apply the TOI burn to the **DefaultSC** spacecraft, so we don’t need to change anything here.
2. Click **OK**.

![Figure 16. Perform TOI Command Configuration](image)

**Configure the Prop to Apoapsis Command**

1. Double-click **Prop to Apoapsis** to edit its properties.
2. Under **Parameter**, replace **DefaultSC.ElapsedSecs** with **DefaultSC.Earth.Apoapsis**.
3. Click **OK** to save these changes.

![Figure 17. Prop to Apoapsis Command Configuration](image)

**Configure the Achieve RMAG = 42165 Command**

1. Double-click **Achieve RMAG = 42165** to edit its properties.
2. Notice that **Goal** is set to **DefaultSC.Earth.RMAG**. This is what we need, so we make no changes here.
3. In the **Value** box, type **42164.169**, a more precise number for the radius of a geosynchronous orbit (in kilometers).
4. Click **OK** to save these changes.
Figure 18. Achieve RMAG = 42165 Command Configuration

Configure the Vary GOI Command

1. Double-click **Vary GOI** to edit its properties.
2. Next to **Variable**, click the **Edit** button.
3. Under **Object List**, click **GOI**.
4. In the **Object Properties** list, double-click **Element1** to move it to the **Selected Value(s)** list. See the image below for results.

Figure 19. Vary GOI Parameter Selection

5. Click **OK** to close the **ParameterSelectDialog** window.
6. In the **Initial Value** box, type **1.0**.
7. In the **MaxStep** text box, type **0.2**.
8. Click **OK** to save these changes.
Configure the Perform GOI Command

1. Double-click **Perform GOI** to edit its properties.
2. In the **Burn** list, click **GOI**.
3. Click **OK** to save these changes.

Configure the Achieve ECC = 0.005 Command

1. Double-click **Achieve ECC = 0.005** to edit its properties.
2. Next to **Goal**, click the **Edit** button.
3. In the **Object Properties** list, double-click **ECC**.
4. Click **OK** to close the **ParameterSelectDialog** window.
5. In the **Value** box, type **0.005**.
6. In the **Tolerance** box, type **0.0001**.
7. Click **OK** to save these changes.
Run the Mission

Before running the mission, click **Save** and save the mission to a file of your choice. Now click **Run**. As the mission runs, you will see GMAT solve the targeting problem. Each iteration and perturbation is shown in **DefaultOrbitView** window in light blue, and the final solution is shown in red. After the mission completes, the 3D view should appear as in the image shown below. You may want to run the mission several times to see the targeting in progress.

![Figure 22. Achieve ECC = 0.005 Command Configuration](image)

![Figure 23. 3D View of Hohmann Transfer](image)
If you were to continue developing this mission, you can store the final solution of the Target sequence as the initial conditions of the TOI and GOI resources themselves, so that if you make small changes, the subsequent runs will take less time. To do this, follow these steps:

1. In the Mission tree, double-click Hohmann Transfer to edit its properties.
2. Click Apply Corrections.
3. Now re-run the mission. If you inspect the results in the message window, you will see that the Target sequence converges in one iteration because you stored the solution as the initial condition.
## Part V. Reference Guide

### Table of Contents

I. Resources .......................................................................................................................... 50  
   Array ............................................................................................................................... 51  
   Barycenter ..................................................................................................................... 52  
   CelestialBodies ............................................................................................................. 53  
   CoordinateSystem ....................................................................................................... 57  
   DifferentialCorrector ................................................................................................. 58  
   EphemerisFile ............................................................................................................... 61  
   EphemerisPropagator ................................................................................................. 62  
   FiniteBurn .................................................................................................................... 63  
   Formation ....................................................................................................................... 65  
   FuelTank ........................................................................................................................ 66  
   GMATFunction .............................................................................................................. 69  
   GroundStation ............................................................................................................. 70  
   GroundTrackPlot ......................................................................................................... 71  
   ImpulsiveBurn ............................................................................................................... 74  
   LibrationPoint ............................................................................................................. 76  
   MATLABFunction ......................................................................................................... 78  
   OrbitView ..................................................................................................................... 79  
   Propagator ................................................................................................................... 85  
   ReportFile .................................................................................................................... 92  
   SolarSystem ................................................................................................................. 95  
   Spacecraft .................................................................................................................... 97  
   SQP ............................................................................................................................... 106  
   String ............................................................................................................................ 109  
   Thruster ......................................................................................................................... 110  
   Variable ......................................................................................................................... 115  
   VF13adOptimizer .......................................................................................................... 116  
   XYPlot .......................................................................................................................... 117  
II. Commands .................................................................................................................... 119  
   Achieve .......................................................................................................................... 120  
   BeginFiniteBurn ........................................................................................................... 121  
   BeginMissionSequence .............................................................................................. 122  
   CallGmatFunction ........................................................................................................ 123  
   CallMatlabFunction ..................................................................................................... 125  
   ClearPlot ...................................................................................................................... 126  
   Else ................................................................................................................................. 127  
   EndFiniteBurn .............................................................................................................. 128  
   Equation ....................................................................................................................... 129  
   For ................................................................................................................................ 130  
   Global ........................................................................................................................... 133  
   If ...................................................................................................................................... 134  
   Maneuver ...................................................................................................................... 136  
   MarkPoint ...................................................................................................................... 137
<table>
<thead>
<tr>
<th>Command</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimize</td>
<td>138</td>
</tr>
<tr>
<td>NonlinearConstraint</td>
<td>139</td>
</tr>
<tr>
<td>Optimize</td>
<td>141</td>
</tr>
<tr>
<td>PenUp</td>
<td>142</td>
</tr>
<tr>
<td>PenDown</td>
<td>143</td>
</tr>
<tr>
<td>Propagate</td>
<td>144</td>
</tr>
<tr>
<td>Report</td>
<td>147</td>
</tr>
<tr>
<td>Save</td>
<td>148</td>
</tr>
<tr>
<td>SaveMission</td>
<td>149</td>
</tr>
<tr>
<td>ScriptEvent</td>
<td>150</td>
</tr>
<tr>
<td>Stop</td>
<td>151</td>
</tr>
<tr>
<td>Target</td>
<td>152</td>
</tr>
<tr>
<td>Toggle</td>
<td>156</td>
</tr>
<tr>
<td>Vary</td>
<td>157</td>
</tr>
<tr>
<td>While</td>
<td>160</td>
</tr>
</tbody>
</table>
Resources
Array
A two-dimensional numeric array variable

Synopsis
Create Array name[rows,columns];
name(row,column) = value;
...

Description
An array is a numeric variable that can contain multiple values in either one or two dimensions (i.e. a matrix).

Fields

<table>
<thead>
<tr>
<th>Field Description</th>
<th>Default</th>
<th>Limits</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>$ITEM NAME. WILL MOST LIKELY USE COURIER NEW HTML FORMATTING$</td>
<td>$ITEM DEFAULT VALUE$</td>
<td>$ITEM OPTIONS$</td>
<td>$UNITS$</td>
</tr>
<tr>
<td>$ITEM NAME. WILL MOST LIKELY USE COURIER NEW HTML FORMATTING$</td>
<td>$ITEM DEFAULT VALUE$</td>
<td>$ITEM OPTIONS$</td>
<td>$UNITS$</td>
</tr>
</tbody>
</table>

Interactions

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

Examples

Example 1. Creating an array
This example creates an empty one-dimensional array with 5 elements.

Create Array Array1[1,5];

Example 2. Creating and populating a matrix
This example creates the identity matrix of size 2 and names it I:

Create Array I[2,2];
I(1,1) = 1;
I(1,2) = 0;
I(2,1) = 0;
I(2,2) = 1;
**Barycenter**

A barycenter.

**Synopsis**

Create Barycenter *name*

*name*.BodyNames = \{*bodyName*1,*bodyName*2,...,*bodyName*N\}

**Description**

A barycenter is the center of mass of one or more celestial bodies and can be used as the origin of a CoordinateSystem, a reference point in an OrbitView, or as one of the points in a LibrationPoint.

**Fields**

- **BodyNames**: 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\})
  - **Default**: Earth, Luna
  - **Limits**: Sun, Mercury, Venus, Earth, Luna, Mars, Jupiter, Saturn, Uranus, Neptune, Pluto, or any user-defined celestial body. At least one body must be selected!
  - **Units**: N/A

**Interactions**

- **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.

**Examples**

Create Barycenter EarthMoonBary;
GMAT EarthMoonBary.BodyNames = \{Earth, Luna\};
**CelestialBodies**

A celestial body.

**Synopsis**

Create Planet *name*

name.field = value

**Description**

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.

**Fields**

**Fields Associated with All Celestial Bodies**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Mu</strong></td>
<td>The Mu field allows the user to define the gravitational parameter of a celestial body.</td>
</tr>
<tr>
<td>Default</td>
<td>398600.4414</td>
</tr>
<tr>
<td>Limits</td>
<td>Real Number &lt; 0</td>
</tr>
<tr>
<td>Units</td>
<td>km^3/sec^2.z</td>
</tr>
<tr>
<td><strong>Equatorial Radius</strong></td>
<td>The EquatorialRadius field allows the user to define the equatorial radius of a celestial body.</td>
</tr>
<tr>
<td>Default</td>
<td>6378.1363</td>
</tr>
<tr>
<td>Limits</td>
<td>Real Number &lt; 0</td>
</tr>
<tr>
<td>Units</td>
<td>km</td>
</tr>
<tr>
<td><strong>Flattening</strong></td>
<td>The Flattening field allows the user to define the mass of a celestial body.</td>
</tr>
<tr>
<td>Default</td>
<td>0.00335270</td>
</tr>
<tr>
<td>Limits</td>
<td>Real Number</td>
</tr>
<tr>
<td>Units</td>
<td>None</td>
</tr>
<tr>
<td><strong>InitialEpoch</strong></td>
<td>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 the position and velocity of a celestial body, using the initial epoch and state information described below.</td>
</tr>
<tr>
<td>Default</td>
<td>21544.500371</td>
</tr>
<tr>
<td>Limits</td>
<td>Real Number</td>
</tr>
<tr>
<td>Units</td>
<td>A1ModJulian</td>
</tr>
<tr>
<td><strong>SMA</strong></td>
<td>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.)</td>
</tr>
<tr>
<td>Default</td>
<td>149653978.978377</td>
</tr>
<tr>
<td>Limits</td>
<td>Real Number ≥ 0</td>
</tr>
</tbody>
</table>
**ECC**
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.)
- **Default**: 0.017046
- **Limits**: Real Number ≥ 0
- **Units**: km

**INC**
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.)
- **Default**: 23.439034
- **Limits**: Real Number
- **Units**: deg

**RAAN**
The RAAN field allows the user to define the right ascension of the ascending node 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.)
- **Default**: 0.000186
- **Limits**: Real Number
- **Units**: deg

**AOP**
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.)
- **Default**: 101.741639
- **Limits**: Real Number
- **Units**: deg

**TA**
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.)
- **Default**: 358.127085
- **Limits**: Real Number
- **Units**: deg

**Special Fields Associated with Earth**

**NutationUpdate Interval**
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.
- **Default**: 60
- **Limits**: Real Number ≥ 0
- **Units**: sec
Special Fields Associated with Luna

RotationDataSource

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}.

- **Default**: DE405
- **Limits**: DE405, IAU2002
- **Units**: N/A

Interactions

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.

Examples

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;
<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>GMAT Earth.PotentialFileName</td>
<td>'./files/gravity/earth/JGM2.cof'</td>
</tr>
<tr>
<td>GMAT Earth.AngularVelocity</td>
<td>[ 0 0 7.2921158553e-005 ]</td>
</tr>
<tr>
<td>GMAT Earth.HourAngle</td>
<td>0</td>
</tr>
<tr>
<td>GMAT Earth.AtmosphereModelName</td>
<td>''</td>
</tr>
<tr>
<td>GMAT Earth.Order</td>
<td>4</td>
</tr>
<tr>
<td>GMAT Earth.Degree</td>
<td>4</td>
</tr>
<tr>
<td>GMAT Earth.RotationDataSource</td>
<td>'NotApplicable'</td>
</tr>
<tr>
<td>GMAT Earth.NutationUpdateInterval</td>
<td>60</td>
</tr>
</tbody>
</table>
**CoordinateSystem**

A coordinate system.

**Synopsis**

Create CoordinateSystem *name*

*name.field = value*

**Description**

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.

**Interactions**

**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.

**Examples**

Create CoordinateSystem EarthMJ2000Eq;

GMAT EarthMJ2000Eq.Origin = Earth;

GMAT EarthMJ2000Eq.Axes = MJ2000Eq;

GMAT EarthMJ2000Eq.UpdateInterval = 60;

GMAT EarthMJ2000Eq.OverrideOriginInterval = false;
**DifferentialCorrector**

A differential corrector.

**Synopsis**

Create DifferentialCorrector name
name.field = value

**Description**

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.

**Fields**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Default</th>
<th>Limits</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>MaximumIterations</td>
<td>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.</td>
<td>25</td>
<td>Integer &gt;= 1</td>
<td>N/A</td>
</tr>
<tr>
<td>DerivativeMethod</td>
<td>The DerivativeMethod field allows the user to choose between one-sided and central differencing for numerically determining the Jacobian matrix.</td>
<td>ForwardDifference</td>
<td>ForwardDifference, BackwardDifference, CentralDifference</td>
<td>N/A</td>
</tr>
<tr>
<td>ShowProgress</td>
<td>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</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
current control variable values and the constraint 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.

**Default**
true

**Limits**
true, false

**Units**
N/A

**ReportStyle**
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.

**Default**
Normal

**Limits**
Normal, Concise, Verbose, Debug

**Units**
N/A

**ReportFile**
The ReportFile field allows the user to specify the path and file name for the differential correction report.

**Default**
DifferentialCorrectorDCName

**Limits**
Filename consistent with OS

**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
Examples

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

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

An ephemeris file.

Synopsis

Under Construction.

Description

Under Construction.

Fields

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Fill This In.</td>
</tr>
<tr>
<td>Default</td>
<td>Fill This In.</td>
</tr>
<tr>
<td>Limits</td>
<td>Fill This In.</td>
</tr>
<tr>
<td>Units</td>
<td>Fill This In.</td>
</tr>
</tbody>
</table>

Examples

Example 3. Example Script

%
EphemerisPropagator

Under Construction.

Synopsis

Under Construction.

Description

Under Construction.

Fields

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Fill This In.</td>
</tr>
<tr>
<td>Default</td>
<td>Fill This In.</td>
</tr>
<tr>
<td>Limits</td>
<td>Fill This In.</td>
</tr>
<tr>
<td>Units</td>
<td>Fill This In.</td>
</tr>
</tbody>
</table>

Examples

Example 4. Example Script

```%```
FiniteBurn
A finite burn.

Synopsis

Create FiniteBurn name
name.field = value

Description

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.

Fields

**Origin**
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.

Default: Earth
Limits: Any celestial body, libration point, or barycenter
Units: N/A

**Axes**
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.

Default: VNB
Limits: VNB, MJ2000Eq
Units: N/A

**Thrusters**
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.

Default: No Default
Limits: Any thruster created by user
Units: N/A

**BurnScaleFactor**
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.

Default: 1.0
Limits: Real Number
Units: N/A

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.

Examples

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

An ephemeris file.

**Synopsis**

Under Construction.

**Description**

Under Construction.

**Fields**

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Fill This In.</td>
</tr>
<tr>
<td>Default</td>
<td>Fill This In.</td>
</tr>
<tr>
<td>Limits</td>
<td>Fill This In.</td>
</tr>
<tr>
<td>Units</td>
<td>Fill This In.</td>
</tr>
</tbody>
</table>

**Examples**

Example 5. Example Script

```
**FuelTank**

A fuel tank.

**Synopsis**

```
Create FuelTank name
name.field = value
```

**Description**

A FuelTank is a thermodynamic model of a tank and is required for finite burn modelling or for impulsive burns that use mass depletion. The thermodynamic properties of the tank are modelled using the ideal gas law and assume that there is no energy transfer into or out of the tank as fuel is depleted. To use a FuelTank, you must first create the tank, and then attach it to the desired spacecraft and associate it with a thruster as shown in the examples below.

When working in the script, you must add tanks to spacecraft before the begin mission sequence command.

**Fields**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Pressure</strong></td>
<td>The pressure in the tank.</td>
</tr>
<tr>
<td>Type</td>
<td>Real Number</td>
</tr>
<tr>
<td>Default</td>
<td>1500</td>
</tr>
<tr>
<td>Limits</td>
<td>Pressure &gt; 0</td>
</tr>
<tr>
<td>Units</td>
<td>kPa.</td>
</tr>
<tr>
<td><strong>Temperature</strong></td>
<td>The temperature of the fuel and ullage in the tank. GMAT currently assumes ullage and fuel are always at the same temperature.</td>
</tr>
<tr>
<td>Type</td>
<td>Real Number</td>
</tr>
<tr>
<td>Default</td>
<td>20</td>
</tr>
<tr>
<td>Limits</td>
<td>Temperature &gt; -273.15</td>
</tr>
<tr>
<td>Units</td>
<td>C.</td>
</tr>
<tr>
<td><strong>FuelMass</strong></td>
<td>The FuelMass field is the mass of fuel in the tank.</td>
</tr>
</tbody>
</table>

Caution

By default, GMAT will not allow the fuel mass to be negative. However, occasionally in iterative processes such as targeting, a solver will try values of a maneuver parameter that result in total fuel depletion. Using the default tank settings this will throw an exception stopping the run unless you set the AllowNegativeFuelMass flag to true.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Real Number</td>
</tr>
<tr>
<td>Default</td>
<td>756</td>
</tr>
<tr>
<td>Limits</td>
<td>FuelMass &gt; 0</td>
</tr>
<tr>
<td>Units</td>
<td>kg.</td>
</tr>
</tbody>
</table>
**ReferenceTemperature**
The temperature of the tank when fuel was loaded.

- **Type**: Real Number
- **Default**: 20
- **Options**: ReferenceTemperature $\geq 0$
- **Units**: °C.

**Volume**
The volume of the tank. GMAT checks to ensure that the volume of the tank is larger than the volume of fuel loaded in the tank and throws an exception in the case that the fuel volume is larger than the tank volume.

- **Type**: Real Number
- **Default**: 0.75
- **Options**: Real Number $> 0$ such that fuel volume is $<\text{ tank volume.}$
- **Units**: m$^3$.

**FuelDensity**
The density of the fuel.

- **Type**: Real Number
- **Default**: 1260
- **Limits**: Real Number $> 0$
- **Units**: kg/m$^3$.

**PressureModel**
The pressure model describes how pressure in the tank changes as fuel is depleted.

- **Type**: Enumeration
- **Default**: PressureRegulated
- **Limits**: PressureRegulated, BlowDown
- **Units**: N/A

**AllowNegativeFuelMass**
This field allows the fuel tank to have negative fuel mass which can be useful in optimization and targeting sequences before convergences has occurred.

- **Default**: false
- **Options**: true, false.
- **Units**: N/A
**Examples**

Example 6. Creating a default FuelTank and attaching it to a Spacecraft

```plaintext
% Create the Fuel Tank Object
Create FuelTank aTank;
aTank.AllowNegativeFuelMass = false;
aTank.FuelMass = 756;
aTank.Pressure = 1500;
aTank.Temperature = 20;
aTank.RefTemperature = 20;
aTank.Volume = 0.75;
aTank.FuelDensity = 1260;
aTank.PressureModel = PressureRegulated;

% Create a Thruster and assign it a FuelTank
Create Thruster aThruster;
aThruster.Tank = {aTank};

% Add the FuelTank and Thruster to a Spacecraft
Create Spacecraft aSpacecraft
aSpacecraft.Tanks = {aTank};
aSpacecraft.Thrusters = {aThruster};
```
GMATFunction
Under Construction.

Synopsis
Under Construction.

Description
Under Construction.

Fields

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Fill This In.</td>
</tr>
<tr>
<td>Default</td>
<td>Fill This In.</td>
</tr>
<tr>
<td>Limits</td>
<td>Fill This In.</td>
</tr>
<tr>
<td>Units</td>
<td>Fill This In.</td>
</tr>
</tbody>
</table>

Examples

Example 7. Example Script

%
**GroundStation**

Under Construction.

**Synopsis**

Under Construction.

**Description**

Under Construction.

**Fields**

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Fill This In.</td>
</tr>
<tr>
<td>Default</td>
<td>Fill This In.</td>
</tr>
<tr>
<td>Limits</td>
<td>Fill This In.</td>
</tr>
<tr>
<td>Units</td>
<td>Fill This In.</td>
</tr>
</tbody>
</table>

**Examples**

Example 8. Example Script

```%```
GroundTrackPlot

A GroundTrack Plot.

Synopsis

Create GroundTrackPlot name
name.field = value

Description

A GroundTrackPlot is a graphical display of the locus of subsatellite latitude and longitude points. The GroundTrackPlot in GMAT allows you to view a spacecraft's subsatellite point as illustrated by a spacecraft icon and the label for the spacecraft. Similarly, GroundStation locations are indicated with a ground station icon and label. The GroundTrackPlot object can display the ground track for multiple spacecraft simultaneously and can animate the ground track evolution after a GMAT run is complete. Like other graphical display objects in GMAT, you can control how data is written to a ground track plot in iterative processes.

Fields

Fields Associated with Plot Options

DataCollectFrequency

The DataCollectFrequency field allows you to select a subset of the ephemeris data for drawing to a GroundTrackPlot. It is often inefficient to draw every ephemeris point associated with a trajectory and drawing a subset of the data provides a smooth groundtrack plot with faster execution times. The DataCollectFrequency is an integer that represents how many ephemeris points to skip between plotted data points in a GroundTrackPlot. If DataCollectFrequency is set to 10, then data is collected every 10 integration steps.

Default 1
Limits Integer ≥ 1
Units Integration Steps

UpdatePlotFrequency

The UpdatePlotFrequency field allows you to specify the number of ephemeris data points to collect before updating a GroundTrackPlot with new latitude and longitude data. Data is collected every N propagation steps where N is defined by DataCollectFrequency. After M points are collected, where M is defined by UpdatePlotFrequency, the GroundTrackPlot is updated with new data. For example, 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.

Default 50
Limits Integer ≥ 1
Units Integration Steps

NumPointsToRedraw

When NumPointsToRedraw is set to zero, all collected 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 and UpdatePlotFrequency for an explanation of how data is collected for a GroundTrackPlot.

**Default** 0

**Limits** Integer ≥ 0

**Units** Integration Steps

---

### ShowPlot

The ShowPlot field allows you to turn off the GroundTrackPlot display window without deleting the plot object or removing it from the script. If you select true, then the plot will be displayed. If you select false, then the plot will not be displayed.

**Default** true

**Limits** true, false

**Units** N/A

---

### Fields Associated with Drawing Options

#### Add

The Add field allows you to add Spacecraft and GroundStations to a GroundTrackPlot.

**Default** DefaultSC, Earth

**Limits** SpacecraftName CelestialBodyName

**Units** N/A

---

#### Central Body

The CentralBody field allows you to specify the central body of a GroundTrackPlot. Currently, GMAT

**Default** Earth

**Limits** CelestialBodyName

**Units** N/A

---

### Fields Associated with Other Options

#### SolverIterations

The SolverIterations field determines if and how perturbed trajectories are drawn to a GroundTrackPlot during iterative a solver sequences. When SolverIterations is set to All, all solver iterations perturbations and iterations are shown on the plot. When SolverIterations is set to Current, only the current solver pass is shown on the plot and the iteration history is not retained. When SolverIterations is set to None, no perturbations or iterations are drawn and the GroundTrackPlot is not updated until the solver has converged.

**Default** Current

**Limits** All, Current, None

**Units** N/A

---

#### TextureMap

The TextureMap field allows you to define a custom map file for use in a GroundTrackPlot.

**Default** Current

**Limits** Texture map in jpg or bmp file.

**Units** N/A
**Additional Information**

When working in the GroundTrackPlot GUI, if you change the CentralBody field, the TextureMap field will automatically change to the default texture map for the new central body. If you have specified a custom texture map file and path, that information will be lost when you change the CentralBody field.

**Interactions**

**Spacecraft**
Any spacecraft in your mission is available to a GroundTrackPlot for display.

**GroundStations**
Any GroundStation in your mission is available to a GroundTrackPlot for display.

**PenUp/PenDown Commands**
You can use the PenUp and PenDown commands to control when data is written to a GroundTrackPlot.

**Toggle Command**
You can use the Toggle command to control when data is written to a GroundTrackPlot.

**Examples**

Create `GroundTrackPlot` `GroundTrackPlot1`;

GMAT `GroundTrackPlot1.CentralBody = Earth;`

GMAT `GroundTrackPlot1.Add = {Sat, Earth};`

GMAT `GroundTrackPlot1.SolverIterations = Current;`

GMAT `GroundTrackPlot1.DataCollectFrequency = 1;`

GMAT `GroundTrackPlot1.UpdatePlotFrequency = 50;`

GMAT `GroundTrackPlot1.NumPointsToRedraw = 0;`

GMAT `GroundTrackPlot1.ShowPlot = true;`

GMAT `GroundTrackPlot1.TextureMap = '../MyMaps/MyTexture.jpg';`

GMAT `GroundTrackPlot1.UpperLeft = [ 0 0 ];`

GMAT `GroundTrackPlot1.Size = [ 0 0 ];`

GMAT `GroundTrackPlot1.RelativeZOrder = 0;`
**ImpulsiveBurn**

A impulsive burn.

**Synopsis**

Create ImpulsiveBurn \textit{name}
\textit{name.field} = \textit{value}

**Description**

The impulsive burn object in GMAT allows the spacecraft to undergo an instantaneous $\Delta V$ 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.

**Fields**

**Origin**

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.

- **Default** Earth
- **Limits** Any celestial body
- **Units** N/A

**Axes**

The Axes field, together with the Origin field, describe the 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 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.

- **Default** VNB
- **Limits** VNB, MJ2000Eq
### VectorFormat

<table>
<thead>
<tr>
<th>Units</th>
<th>N/A</th>
</tr>
</thead>
</table>

The VectorFormat field allows the user to define the format of the maneuver vector.

- **Default**: Cartesian
- **Limits**: Cartesian
- **Units**: N/A

### Element1

<table>
<thead>
<tr>
<th></th>
<th>The Element1 field allows the user to define the first element of the impulsive maneuver vector. Element1 is X if VectorFormat is Cartesian.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>0</td>
</tr>
<tr>
<td>Limits</td>
<td>Real Number</td>
</tr>
<tr>
<td>Units</td>
<td>km/sec</td>
</tr>
</tbody>
</table>

### Element2

<table>
<thead>
<tr>
<th></th>
<th>The Element2 field allows the user to define the second element of the impulsive maneuver vector. Element2 is Y if VectorFormat is Cartesian.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>0</td>
</tr>
<tr>
<td>Limits</td>
<td>Real Number</td>
</tr>
<tr>
<td>Units</td>
<td>km/sec</td>
</tr>
</tbody>
</table>

### Element3

<table>
<thead>
<tr>
<th></th>
<th>The Element3 field allows the user to define the third element of the impulsive maneuver vector. Element3 is Z if VectorFormat is Cartesian.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>0</td>
</tr>
<tr>
<td>Limits</td>
<td>Real Number</td>
</tr>
<tr>
<td>Units</td>
<td>km/sec</td>
</tr>
</tbody>
</table>

### Examples

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;
LibrationPoint

A libration point.

Synopsis

Create LibrationPoint name
name.field = value

Description

A Libration point, also called a Lagrange point, is a point of equilibrium in the restricted three-body problem.

Fields

Primary
The Primary field allows you to define the body treated as the primary in the calculation of the libration point location. (See Math. Spec for more details).

Default: Sun
Limits: Sun, Mercury, Venus, Earth, Luna, Mars, Jupiter, Saturn, Uranus, Neptune, Pluto, or any Barycenter. (The Primary and Secondary bodies cannot be the same)
Units: N/A

Secondary
The Secondary field allows you to define the body treated as the secondary in the calculation of the libration point location.

Default: Earth
Limits: Sun, Mercury, Venus, Earth, Luna, Mars, Jupiter, Saturn, Uranus, Neptune, Pluto, or any Barycenter. (The Primary and Secondary bodies cannot be the same)
Units: N/A

Point
The Point field specifies which libration point the object corresponds to.

Default: L1
Limits: L1, L2, L3, L4, L5
Units: N/A

Examples

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';
**MATLABFunction**

Under Construction.

**Synopsis**

Under Construction.

**Description**

Under Construction.

**Fields**

| Field Name | Description...
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Type</strong></td>
<td>Fill This In.</td>
</tr>
<tr>
<td><strong>Default</strong></td>
<td>Fill This In.</td>
</tr>
<tr>
<td><strong>Limits</strong></td>
<td>Fill This In.</td>
</tr>
<tr>
<td><strong>Units</strong></td>
<td>Fill This In.</td>
</tr>
</tbody>
</table>

**Examples**

Example 9. Example Script

```
**OrbitView**

A 3D graphics view

### Synopsis

Create OrbitView name
name.field = value

### Description

Without OrbitView plots, you would have no way of visualizing your spacecraft as it traveled along its trajectory. OrbitView 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.

### Fields

#### Fields Associated with Plot Options

**DataCollectFrequency**
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.

Default: 1
Limits: Integer ≥ 1
Units: Integration Steps

**UpdatePlotFrequency**
The UpdatePlotFrequency field allows the user to specify how often to update an OrbitView 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 OrbitView 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.

Default: 50
Limits: Integer ≥ 1
Units: Integration Steps

**NumPointsToRedraw**
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 OrbitView plot.

Default: 0
Limits: Integer ≥ 0
**Units**

Integration Steps

**ShowPlot**

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.

*Default: true*

*Limits: true, false*

*Units: N/A*

**Fields Associated with Viewed Objects**

**Add**

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 OrbitView plot and a warning message displayed in the message window. The following warning message is sufficient: OrbitView plot will be turned off. No object has been selected for plotting.

*Default: DefaultSC, Earth*

*Limits: SpacecraftName, CelestialBodyName, LibrationPointName, BarycenterName*

*Units: N/A*

**Remove**

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.

*Default: No Default*

*Limits: Any object included in the Add list*

*Units: N/A*

**Fields Associated with Drawing Options**

**WireFrame**

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.

*Default: Off*

*Limits: On, Off*

*Units: N/A*

**EclipticPlane**

The EclipticPlane field allows the user to tell GMAT to draw a grid representing the ecliptic plane in an OrbitView plot. Note, the ecliptic plane can currently only be drawn for plots whose coordinate system uses the MJ2000Eq axis system.

*Default: Off*

*Limits: On, Off Note: Only allowed for OrbitView plots with Coordinate Systems that use the MJ2000Eq axis system*

*Units: N/A*

**XYPlane**

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 OrbitView plot.

*Default: On*
Limits On, Off
Units N/A

Axes
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 OrbitView plot.
Default On
Limits On, Off
Units N/A

Grid
The Grid flag allows the user to tell GMAT to draw a grid representing the longitude and latitude lines celestial bodies added to an OrbitView plot.
Default On
Limits On, Off
Units N/A

EarthSunLines
The EarthSunLines allows the user to tell GMAT to draw a line that starts at the center of Earth and points towards the Sun.
Default On
Limits On, Off
Units N/A

SolverIterations
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.
Default Off
Limits On, Off
Units N/A

Fields Associated with View Definition

CoordinateSystem
The CoordinateSystem field on an OrbitView 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 OrbitView plot. See the CoordinateSystem object fields for information of defining different types of coordinate systems.
Default EarthMJ2000Eq
Limits Any default or user defined coordinate system
Units N/A

ViewPointReference
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.
Default Earth
Limits SpacecraftName, CelestialBodyName, LibrationPointName, BarycenterName, or a 3-vector of numerical values
Units N/A
<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Default</th>
<th>Limits</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>ViewPointVector</td>
<td>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.</td>
<td>[0 0 30000]</td>
<td>SpacecraftName, CelestialBodyName, LibrationPointName, BarycenterName, or a 3-vector of numerical values</td>
<td>km or N/A</td>
</tr>
<tr>
<td>ViewScaleFactor</td>
<td>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.</td>
<td>1</td>
<td>Real Number ≥ 0</td>
<td>N/A</td>
</tr>
<tr>
<td>ViewDirection</td>
<td>The ViewDirection field allows the user to select the direction of view in an OrbitView 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.</td>
<td>Earth</td>
<td>SpacecraftName, CelestialBodyName, LibrationPointName, BarycenterName, or a 3-vector of numerical values</td>
<td>km or N/A</td>
</tr>
</tbody>
</table>

**Fields Associated with View Options**

**UseInitialView**

The UseInitialView field allows the user to control the view of an OrbitView plot between multiple runs of a mission sequence. The first time a specific OrbitView 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 OrbitView plot will be returned to the view defined by the initial settings.

Default: On

Limits: On, Off

Units: N/A

**Fields Associated with View Up Definition**

**ViewUpCoordinateSystem**

The ViewUpCoordinateSystem and ViewUpAxis fields are used to determine which direction appears as up in an OrbitView plot. The user can select the coordinate system in which the up vector appears. The default coordinate system is Earth-centered Earth-fixed (ECEF).

Default: Earth

Limits: Earth-centered Earth-fixed (ECEF), spacecraft-centered (CS), celestial body-centered (CB), Sun-centered (SU), or Earth-centered solar system barycenter (ESB)

Units: km or N/A
bitView plot and together with the fields associated 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.

**ViewUpAxis**

- **Default**: EarthMJ2000Eq
- **Limits**: Any default or user defined coordinate system
- **Units**: N/A

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

- **Default**: Z
- **Limits**: X, -X, Y, -Y, Z, -Z
- **Units**: N/A

**Interactions**

- **Spacecraft**: Any spacecraft in your mission is available to the OrbitView plot for display
- **Solar System**: The Sun and all of the Planets may be plotted or referenced in the OrbitView plot
  - If you add any Barriycenters 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

**Examples**

```plaintext
Create OrbitView 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;
Propagator

A propagator.

Synopsis

Create Propagator name
name.field = value

Description

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 multiple 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](image)

**Fields**

**Force Model Group Box Fields**

**ErrorControl**

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**: RSSStep
- **Limits**: RSSStep, RSSState, LargestState, LargestStep

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.

- **Default**: Earth
- **Limits**: Sun, Mercury, Venus, Earth, Luna, Mars, Jupiter, Saturn, Uranus, Neptune, Pluto
- **Units**: N/A

**PrimaryBodies**

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 field and the PointMasses field. A primary body can be any planet or moon not included in the PointMasses field.

- **Default**: Earth
- **Limits**: Sun, Mercury, Venus, Earth, Luna, Mars, Jupiter, Saturn, Uranus, Neptune, Pluto
- **Units**: N/A

**Gravity.PrimaryBody.PotentialFile**

This field allows the user to define the source for the nonspherical 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.

**Default**  JGM2  
**Limits**  CentralBody-based models, Other. See Comments  
**Units**  N/A

**Gravity.PrimaryBody.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.

**Default**  4  
**Limits**  Integer >= 0 and < the maximum specified by the model, Order <= Degree  
**Units**  N/A

**Gravity.PrimaryBody.Order**  
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.

**Default**  4  
**Limits**  Integer >= 0 and < the maximum specified by the model, Order <= Degree  
**Units**  N/A

**Drag**  
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.

**Default**  N/A  
**Limits**  None, JacchiaRoberts, MSISE90, Exponential  
**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**  
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.

**Default**  None  
**Limits**  None, JacchiaRoberts, MSISE90, Exponential  
**Units**  N/A

**Drag.F107**  
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.

**Default**  150  
**Limits**  Real Number >= 0  
**Units**  W/m²/Hz

**Drag.F107A**  
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.

**Default**  150
Drag.MagneticIndex

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.

Default: 3
Limits: $0 \leq$ Real Number $\leq 9$
Units: N/A

PointMasses

A PointMass is a planet or moon that is modeled 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.

Default: None
Limits: Sun, Mercury, Venus, Earth, Luna, Mars, Jupiter, Saturn, Uranus, Neptune, Pluto
Units: N/A

SRP

The SRP field allows the user to include the force due to solar radiation pressure in the total sum of forces.

Default: Off
Limits: On, Off
Units: N/A

Integrator Group Box Fields

Type

The Type field is used to set the type of numerical integrator.

Default: RungeKutta89
Limits: RungeKutta89, RungeKutta68, RungeKutta56, PrinceDormand45, PrinceDormand78, BulirschStoer, AdamsBashforthMoulton

InitialStepSize

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 to an algorithm defined in the mathematical specifications document to find an acceptable first step that meets the user's requested.

Default: 60 (sec)
Limits: Real Number
Units: seconds

Accuracy

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.

Default: 1e-11
Limits: Real Number $\geq 0$
Units: N/A

MinStep

The MinStep field is used to set the minimum allowable step size.
Reference Guide

Propagator

Default: 0.001 (sec)
Limits: Real Number > 0, MinStep <= MaxStep
Units: seconds

MaxStep
The MaxStep field is used to set the maximum allowable step size.
Default: 2700.0 (sec)
Limits: Real Number > 0, MinStep <= MaxStep
Units: seconds

MaxStepAttempts
The MaxStepAttempts field allows the user to set the number of attempts the integrator takes to meet the tolerance defined by Accuracy.
Default: 50
Limits: Integer > 0
Units: None

Fields Associated Only with Adams-Bashforth-Moulton Integrator

MinIntegrationerror
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.
Default: 1.0e-13
Limits: Real Number > 0, MinIntegrationerror < NomIntegrationerror < Accuracy
Units: None

NomIntegrationerror
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.
Default: 1.0e-11
Limits: Real Number > 0, MinIntegrationerror, NomIntegrationerror, Accuracy
Units: None

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.)
Examples

Create ForceModel DefaultProp_ForceModel;
DefaultProp_ForceModel.CentralBody = Earth;
DefaultProp_ForceModel.PrimaryBodies = {Earth};
DefaultProp_ForceModel.Drag = None;
DefaultProp_ForceModel.SRP = Off;
DefaultProp_ForceModel.ErrorControl = RSSStep;
DefaultProp_ForceModel.GravityField.Earth.Degree = 4;

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

A ReportFile.

Synopsis

Create ReportFile name
name.field = value

Description

The ReportFile 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]

Fields

FileName

The FileName field allows the user to define the file path and file name for a report.

- **Default**: /RunReports/ReportFile1.txt
- **Limits**: Valid File Path and Name
- **Units**: None

Precision

The Precision field allows the user to set the precision of the variable written to a report.

- **Default**: 16
Add

The `{Add}` field allows a user to add user-defined variables to a report file. To add multiple user-defined variables, enclose the variables with curly brackets. Ex. `MyReportName.Add = {Sat.X, Sat.Y, Var1, Array(1,1)}`; 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 Report output and a warning message displayed in the message window. The following warning message is sufficient: Report plot will be turned off. No object has been selected for reporting.

**Default** None

**Limits** Any user-defined parameter. Ex. Variables, Arrays, S/C parameters

**Units** None

WriteReport

The `WriteReport` field specifies whether to write data to the report FileName.

**Default** On

**Limits** Off

**Units** None

WriteHeaders

The `WriteHeaders` field specifies whether to include headers that describe the variables in a report.

**Default** On

**Limits** Off

**Units** None

LeftJustify

When the `LeftJustify` field is set to On, then the data is left justified and appears at the left most side of the column. If the `LeftJustify` field is set to Off, then the data is centered in the column.

**Default** On

**Limits** Off

**Units** None

ZeroFill

**Default** On

**Limits** Off

**Units** None

ColumnWidth

The `ColumnWidth` field is used to define the width of the data columns in a report file. The value for `ColumnWidth` is applied to all columns of data. For example, if `ColumnWidth` is set to 20, then each data column will be 20 white spaces wide.

**Default** 20

**Limits** Integer > 0

**Units** Characters

SolverIterations

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.

**Default** Off

**Limits** Off

**Units** None
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.

Examples

Create ReportFile ReportFile1;
ReportFile1.SolverIterations = Current;
ReportFile1.Filename = 'ReportFile1.txt';
ReportFile1.Precision = 16;
ReportFile1.WriteHeaders = On;
ReportFile1.LeftJustify = On;
ReportFile1.ZeroFill = Off;
ReportFile1.ColumnWidth = 20;
**SolarSystem**

A solar system.

**Synopsis**

Create SolarSystem name
name.field = value

**Description**

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.

**Fields**

- **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.
  - **Default**: DE405
  - **Limits**: DE405, DE200, SLP, Analytic
  - **Units**: None

- **Ephemeris UpdateInterval**
  - 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.
  - **Default**: 0
  - **Limits**: Real Number ≥ 0
  - **Units**: sec

- **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).
  - **Default**: false
  - **Limits**: true, false
  - **Units**: None
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.

- **Default**: Same as startup file.
- **Limits**: Filepath and file name consistent with operating system.
- **Units**: None

AnalyticModel

- **Default**: LowFidelity
- **Limits**: LowFidelity
- **Units**: None

**Interactions**

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.
Spacecraft

A spacecraft

Description

A Spacecraft resource contains information about the spacecraft's orbit, its attitude, its physical parameters (such as mass and drag coefficient), and any attached hardware, including thrusters and fuel tanks. It also contains information about the visual model used to represent it in an OrbitView.

Epoch

The epoch of a Spacecraft is the time and date corresponding to the specified orbit state. It defines the start time for propagation using the Propagate command. See the Spacecraft Orbit State section for interactions between the epoch, coordinate system, and spacecraft state fields.

Caution

GMAT's Modified Julian Date (MJD) format differs from that of other software. The Modified Julian format is a constant offset from the full Julian date (JD):

\[
\text{MJD} = \text{JD} - \text{offset}
\]

GMAT uses a non-standard offset, as shown in the following table.

<table>
<thead>
<tr>
<th>Field</th>
<th>GMAT</th>
<th>common</th>
</tr>
</thead>
<tbody>
<tr>
<td>reference epoch</td>
<td>05 Jan 1941 12:00:00.000</td>
<td>17 Nov 1858 00:00:00.000</td>
</tr>
<tr>
<td>Modified Julian offset</td>
<td>2430000.0</td>
<td>2400000.5</td>
</tr>
</tbody>
</table>

Fields

Epoch

Caution

The range limits on the Epoch field are validated only on input. It is possible to propagate beyond these limits, then use the resulting Epoch in a calculation or output it to a report. This situation has not been tested, and is not expected to be accurate.
<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DateFormat</td>
<td>The time system and format of the <strong>Epoch</strong> field. In the GUI, this field is called <strong>Epoch Format</strong>.</td>
</tr>
<tr>
<td>Data Type</td>
<td>Enumeration</td>
</tr>
<tr>
<td>Allowed Values</td>
<td>A1ModJulian, TAIModJulian, UTCModJulian, TTModJulian, TDBModJulian, A1Gregorian, TAIGregorian, TTAGregorian, UTCGregorian, TDBGregorian</td>
</tr>
<tr>
<td>Access</td>
<td>set only</td>
</tr>
<tr>
<td>Default Value</td>
<td>TAIModJulian</td>
</tr>
<tr>
<td>Interfaces</td>
<td>GUI, script</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Epoch</th>
<th>The time and date corresponding to the specified orbit state. It defines the start time for propagation using the <strong>Propagate</strong> command.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Type</td>
<td>Time</td>
</tr>
<tr>
<td>Allowed Values</td>
<td>Gregorian: 04 Oct 1957 12:00:00.000 &lt;= Epoch &lt;= 28 Feb 2100 00:00:00.000</td>
</tr>
<tr>
<td></td>
<td>Modified Julian: 6116.0 &lt;= Epoch &lt;= 58127.5</td>
</tr>
<tr>
<td>Access</td>
<td>set only</td>
</tr>
<tr>
<td>Default Value</td>
<td>21545</td>
</tr>
<tr>
<td>Interfaces</td>
<td>GUI, script</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CurrA1MJD</th>
<th>The current epoch in the A1ModJulian format. This field can only be used within the mission sequence.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Type</td>
<td>Time</td>
</tr>
<tr>
<td>Allowed Values</td>
<td>6116.0 &lt;= CurrA1MJD &lt;= 58127.5</td>
</tr>
<tr>
<td>Access</td>
<td>get, set (mission sequence only)</td>
</tr>
<tr>
<td>Default Value</td>
<td>converted equivalent of 21545 Modified Julian (TAI)</td>
</tr>
<tr>
<td>Interfaces</td>
<td>script only</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Epoch.Format</th>
<th>The spacecraft orbit epoch in the specified system and format. For example, <strong>Epoch.TAIModJulian</strong> is the epoch in the TAI system and the Modified Julian format.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Type</td>
<td>Time</td>
</tr>
<tr>
<td>Allowed Values</td>
<td>Gregorian: 04 Oct 1957 12:00:00.000 &lt;= Epoch &lt;= 28 Feb 2100 00:00:00.000</td>
</tr>
<tr>
<td></td>
<td>Modified Julian: 6116.0 &lt;= Epoch &lt;= 58127.5</td>
</tr>
<tr>
<td>Access</td>
<td>get, set</td>
</tr>
<tr>
<td>Default Value</td>
<td>converted equivalent of 21545 Modified Julian (TAI)</td>
</tr>
<tr>
<td>Interfaces</td>
<td>script only</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>format</td>
<td>The spacecraft orbit epoch in the specified system and format. For example, <strong>TAI-ModJulian</strong> is the epoch in the TAI system and the Modified Julian format. These fields can only be used within the mission sequence.</td>
</tr>
</tbody>
</table>

**Data Type**  
**Allowed Values**  
Gregorian: **04 Oct 1957 12:00:00.000 <= Epoch <= 28 Feb 2100 00:00:00.000**  
Modified Julian: **6116.0 <= Epoch <= 58127.5**

**Access**  
get, set (mission sequence only)

**Default Value**  
converted equivalent of 21545 Modified Julian (TAI)

**Interfaces**  
GUI, script

---

**Orbit**

**CoordinateSystem**  
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 \( \mu \) value.

**Type**  
enumeration

**Values**  
- EarthMJ2000Eq (default)
- EarthMJ2000Ec
- EarthFixed
- any user-defined coordinate system

**DisplayStateType**  
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 \( \mu \) value.

**Type**  
enumeration

**Values**  
- Cartesian (default)
- Keplerian
- ModifiedKeplerian
- SphericalAZFPA
- SphericalRADEC
- Equinoctial

---

**Cartesian State**

**X**  
X is the x-component of the Spacecraft state in the coordinate system chosen in the Spacecraft Coordinate System field.

**Type**  
real number

**Default**  
7100

**Limits**  

Units km

Y Y is the y-component of the Spacecraft state in the coordinate system chosen in the Spacecraft Coordinate System field.
Type real number
Default 0
Limits
Units km

Z Z is the z-component of the Spacecraft state in the coordinate system chosen in the Spacecraft Coordinate System field.
Type real number
Default 1300
Limits
Units km

VX VX is the x-component of the Spacecraft velocity in the coordinate system chosen in the Spacecraft Coordinate System field.
Type real number
Default 0
Limits
Units km

VY VY is the y-component of the Spacecraft velocity in the coordinate system chosen in the Spacecraft Coordinate System field.
Type real number
Default 7.35
Limits
Units km

VZ VZ is the z-component of the Spacecraft velocity in the coordinate system chosen in the Spacecraft Coordinate System field.
Type real number
Default 1
Limits
Units km

Keplerian State

SMA 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.
Type real number
Default
Limits |SMA| > 1e-3 km
Units km

ECC 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.
The INC field is the spacecraft orbit's osculating inclination, in degrees, with respect to the selected coordinate system.

INC

Type: real number

Default

Limits: $0.0 \leq \text{ECC} \leq 0.9999999$ or $\text{ECC} \geq 1.0000001$

Units: none

RAAN

The RAAN field is the spacecraft orbit's osculating right ascension of the ascending node, in degrees, with respect to the selected coordinate system.

RAAN

Type: real number

Default

Limits

Units: degrees

AOP

The AOP field is the spacecraft orbit's osculating argument of periapsis, in degrees, with respect to the selected coordinate system.

AOP

Type: real number

Default

Limits

Units: degrees

TA

The TA field is the spacecraft orbit's osculating true anomaly.

TA

Type: real number

Default

Limits

Units: degrees

### Modified Keplerian State

RadPer

The RadPer field is the spacecraft orbit's osculating radius of periapsis. RadPer must be greater than zero.

RadPer

Type: real number

Default

Limits

Units: km

RadApo

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.

RadApo

Type: real number

Default

Limits

Units: km

INC

See the section called “Keplerian State” section for a description of this field.

RAAN

See the section called “Keplerian State” section for a description of this field.

AOP

See the section called “Keplerian State” section for a description of this field.

TA

See the section called “Keplerian State” section for a description of this field.
Spherical AZFPA State

**RMAG**  The RMAG field allows the user to set the magnitude of the spacecraft's position vector.
- **Type**: real number
- **Default**: 
- **Limits**: RMAG > 0
- **Units**: km

**RA**  The RA field allows the user to set the spacecraft's right ascension.
- **Type**: real number
- **Default**: 
- **Limits**: 
- **Units**: degrees

**DEC**  The DEC field allows the user to set the spacecraft's declination.
- **Type**: real number
- **Default**: 
- **Limits**: 
- **Units**: degrees

**VMAG**  The VMAG field allows the user to set the magnitude of the spacecraft's velocity.
- **Type**: real number
- **Default**: 
- **Limits**: VMAG >= 0
- **Units**: km/s

**AZI**  The AZI field allows the user to set the spacecraft's azimuth angle.
- **Type**: real number
- **Default**: 
- **Limits**: 
- **Units**: degrees

**FPA**  The FPA allows the user to set a spacecraft's flight path angle.
- **Type**: real number
- **Default**: 
- **Limits**: 
- **Units**: degrees
GUI

Orbit

A change in Epoch Format causes an immediate update to Epoch to reflect the chosen time system and format.

Remarks

Epoch

GMAT supports five time systems or scales and two formats:

Table 4. Time Systems

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A.1</td>
<td>USNO atomic time; GMAT’s internal time system</td>
</tr>
<tr>
<td>TAI</td>
<td>International Atomic Time</td>
</tr>
<tr>
<td>TDB</td>
<td>Barycentric Dynamical Time</td>
</tr>
<tr>
<td>TT</td>
<td>Terrestrial Time</td>
</tr>
<tr>
<td>UTC</td>
<td>Coordinated Universal Time</td>
</tr>
</tbody>
</table>
Table 5. Epoch Formats

| Gregorian       | Text with the following format: \[dd\ \text{mmm}\ \text{yyyy} \]
|-----------------|\[\text{HH}:\text{MM}:\text{SS}.\text{FFF}\]
|                 | \[\text{dd}\] two-digit day of month
|                 | \[\text{mmm}\] first three letters of month
|                 | \[\text{yyyy}\] four-digit year
|                 | \[\text{HH}\] two-digit hour
|                 | \[\text{MM}\] two-digit minute
|                 | \[\text{SS}\] two-digit second
|                 | \[\text{FFF}\] three-digit fraction of second

| Modified Julian | Floating-point number of days from a reference epoch. In GMAT, the reference epoch is 05 Jan 1941 12:00:00.000 (JD 2430000.0).

The epoch can be set in multiple ways. The default method is to set the DateFormat field to the desired time system and format, then set the Epoch field to the desired epoch. This method cannot be used to get the epoch value, such as on the right-hand side of an assignment statement.

```plaintext
aSat.DateFormat = UTCGregorian
aSat.Epoch = '18 May 2012 12:00:00.000'
```

An alternate method is to specify the DateFormat in the parameter name. This method works in both “get” and “set” modes.

```plaintext
aSat.Epoch.UTCGregorian = '18 May 2012 12:00:00.000'
Report aReport aSat.Epoch.UTCGregorian
```

A third method can be used in “get” mode everywhere, but in “set” mode only in the mission sequence (i.e. after the BeginMissionSequence command).

```plaintext
aSat.UTCGregorian = '18 May 2012 12:00:00.000'
Report aReport aSat.UTCGregorian
```

GMAT uses the A.1 time system in the Modified Julian format for its internal calculations. The system converts all other systems and formats on input and again at output.

### Leap Seconds

When converting to and from the UTC time system, GMAT includes leap seconds as appropriate, according to the `tai-utc.dat` data file from the IERS. This file contains the conversion between TAI and UTC, including all leap seconds that have been added or announced.

GMAT applies the leap second as the last second before the date listed in the `tai-utc.dat` file, which historically has been either January 1 or July 1. In the Gregorian date format, the leap second appears as a “60th second”: for example, “31 Dec 2008 23:59:60.000”. GMAT will correctly output this epoch, and will accept it as input. GMAT's Modified Julian format is based on an 86,400-second day, however, and will repeat the first second of the following day. Input of the leap second in Modified Julian format is not supported.
For epochs prior to the first entry in the leap-second file, the UTC and TAI time systems are considered identical (i.e. zero leap seconds are added). For epochs after the last entry, the leap second count from the last entry is used.

The `tai-utc.dat` file is periodically updated by the IERS when new leap seconds are announced. The latest version of this file can always be found at http://maia.usno.navy.mil/ser7/tai-utc.dat. To replace it, download the latest version and replace GMAT's file in the location `<GMAT>/data/time/tai-utc.dat`, where `<GMAT>` is the install directory of GMAT on your system.

**Examples**

Setting the epoch for propagation

```
Create Spacecraft aSat
 aSat.DateFormat = TAIModJulian
 aSat.Epoch = 25562.5

Create ForceModel aFM
 Create Propagator aProp
 aProp.FM = aFM

BeginMissionSequence
 Propagate aProp(aSat) {aSat.ElapsedDays = 1}
```

Plotting and reporting the epoch (syntax #1)

```
Create Spacecraft aSat
 aSat.DateFormat = A1Gregorian
 aSat.Epoch = '12 Jul 2015 08:21:45.921'

Create XYPlot aPlot
 aPlot.XVariable = aSat.UTCModJulian
 aPlot.YVariables = aSat.Earth.Altitude

Create Report aReport
 aReport.Add = {aSat.UTCGregorian, aSat.EarthMJ2000Eq.ECC}
```

Plotting and reporting the epoch (syntax #2)

```
Create Spacecraft aSat
 aSat.DateFormat = TTGregorian
 aSat.Epoch = '01 Dec 1978 00:00:00.000'

Create XYPlot aPlot
 aPlot.XVariable = aSat.Epoch.TTModJulian
 aPlot.YVariables = aSat.Earth.RMAG

Create Report aReport
 aReport.Add = {aSat.Epoch.A1Gregorian, aSat.Earth.RMAG}
```
SQP

A SQP(fmincon).

Synopsis

Create FminconOptimizer name
name.field = value

Description

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.

Interfaces

Figure: FminconOptimizer Dialog Box
## Fields

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
<th>Default</th>
<th>Limits</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DiffMax Change</strong></td>
<td>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.</td>
<td>0.1</td>
<td>Real Number &gt; 0</td>
<td>None</td>
</tr>
<tr>
<td><strong>DiffMin Change</strong></td>
<td>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 perturbation.</td>
<td>1e-8</td>
<td>Real Number &gt; 0</td>
<td>None</td>
</tr>
<tr>
<td><strong>MaxFunEvals</strong></td>
<td>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.</td>
<td>1000</td>
<td>Integer &gt; 0</td>
<td>None</td>
</tr>
<tr>
<td><strong>MaxIter</strong></td>
<td>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.</td>
<td>400</td>
<td>Integer &gt; 0</td>
<td>None</td>
</tr>
<tr>
<td><strong>TolX</strong></td>
<td>The TolX parameter is the termination tolerance on the vector of independent variables, and is used only if the user sets a value.</td>
<td>1e-4</td>
<td>Real Number &gt; 0</td>
<td>None</td>
</tr>
<tr>
<td><strong>TolFun</strong></td>
<td>The TolFun parameter is the convergence tolerance on the cost function value.</td>
<td>1e-4</td>
<td>Real Number &gt; 0</td>
<td>None</td>
</tr>
<tr>
<td><strong>TolCon</strong></td>
<td>The TolCon parameter is the convergence tolerance on the constraint functions.</td>
<td>1e-4</td>
<td>Real Number &gt; 0</td>
<td>None</td>
</tr>
</tbody>
</table>
**ShowProgress**
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 ReportStyle field.

- **Default**: true
- **Limits**: true, false
- **Units**: None

**ReportStyle**
The ReportStyle 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.

- **Default**: Normal
- **Limits**: Normal, Concise, Verbose, Debug
- **Units**: None

**ReportFile**
The ReportFile field contains the path and file name of the report file.

- **Default**: Normal
- **Limits**: '.\output\OptimizerData.txt'
- **Units**: None

### Using an fminconOptimizer

- Optimize Command
- Minimize
- Nonlinear Constraint

### Examples

```matlab
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';
```
String

A string.

Synopsis

Create String name
name.field = value

Description

This page will show you how to create and use String objects. Strings 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.

Interactions

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 Script

Script Syntax

GMAT String Name = String;

Script Examples

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

A thruster.

Synopsis

Create Thruster name
name.field = value

Description

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

Fields

**CoordinateSystem**

The CoordinateSystem field for a thruster determines what coordinate system the orientation parameters X_Direction, Y_Direction, and Z_Direction are referenced to. This is a temporary fix in GMAT. Eventually, the user will specify the attitude of a spacecraft, and then X_Direction, Y_Direction, and Z_Direction will be referenced to the spacecraft body frame.

- **Default**: EarthMJ2000Eq
- **Limits**: EarthMJ2000Eq, EarthMJ2000Ec, EarthMJ2000Eq, or any user-defined system

**Axis**

The Axis field allows the user to define a local coordinate system for a thruster. Note that there is a coupling between the Axis parameter and the CoordinateSystem parameter for a thruster. Only one of the two can be specified.

- **Default**: VNB
- **Limits**: InertialVNB
- **Units**: None

**Origin**

The Origin field allows the user to define a local origin for a thruster. Note that there is a coupling between the Origin parameter and the CoordinateSystem parameter for a thruster. Only one of the two can be specified.

- **Default**: Earth
- **Limits**: Sun, Mercury, Venus, Earth, Luna, Mars, Jupiter, Saturn, Uranus, Neptune, Pluto
- **Units**: None

**X_Direction**

X_Direction, divided by the RSS of the three direction components, forms the x direction of the spacecraft thrust vector direction.

- **Default**: 1
- **Limits**: Real Number

**Y_Direction**

Y_Direction, divided by the RSS of the three direction components, forms the y direction of the spacecraft thrust vector direction.

- **Default**: 0
- **Limits**: Real Number
### Z_Direction

Z_Direction, divided by the RSS of the three direction components, forms the z direction of the spacecraft thrust vector direction.

- **Default**: 0
- **Limits**: Real Number

### ThrustScaleFactor

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 acceleration.

- **Default**: 1
- **Limits**: Real Number > 0
- **Units**: None

### Tank

The Tank field specifies which tank the thruster draws propellant from.

### Thrust Equation

The constants $C_i$ below are used in the following equation to calculate thrust $F_T$ as a function of pressure $P$ and temperature $T$:

$$F_T(P,T) = \left\{ C_1 + C_2 P + C_3 P^2 + C_4 P^{C_5} + C_6 P^{C_7} + C_8 P^{C_9} + C_{10} P^{C_{11}} \right\} \left( \frac{T}{T_{ref}} \right)^{1+C_{12}P}$$

- **C1**: Thrust coefficient. Default: 500
- **Units**: None
- **Limits**: Real Number
- **C2**: Thrust coefficient. Default: 0
- **Units**: N
- **Limits**: Real Number
- **C3**: Thrust coefficient. Default: 0
- **Units**: N/kPa
- **Limits**: Real Number
- **C4**: Thrust coefficient. Default: 0
- **Units**: N/kPa
- **Limits**: Real Number
- **C5**: Thrust coefficient. Default: 0
- **Units**: None
- **Limits**: Real Number
- **C6**: Thrust coefficient. Default: 0
- **Units**: N/kPa
- **Limits**: Real Number
- **C7**: Thrust coefficient. Default: 0
- **Units**: None
- **Limits**: Real Number
- **C8**: Thrust coefficient. Default: 0
- **Limits**: Real Number
### Thruster

<table>
<thead>
<tr>
<th>Constant</th>
<th>Description</th>
<th>Default</th>
<th>Limits</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>C9</td>
<td>Thrust coefficient.</td>
<td>0</td>
<td>Real Number</td>
<td>N/kPa&lt;sup&gt;C9&lt;/sup&gt;</td>
</tr>
<tr>
<td>C10</td>
<td>Thrust coefficient.</td>
<td>0</td>
<td>Real Number</td>
<td>None</td>
</tr>
<tr>
<td>C11</td>
<td>Thrust coefficient.</td>
<td>1</td>
<td>Real Number</td>
<td>None</td>
</tr>
<tr>
<td>C12</td>
<td>Thrust coefficient.</td>
<td>0</td>
<td>Real Number</td>
<td>1/kPa</td>
</tr>
<tr>
<td>C13</td>
<td>Thrust coefficient.</td>
<td>0</td>
<td>Real Number</td>
<td>None</td>
</tr>
<tr>
<td>C14</td>
<td>Thrust coefficient.</td>
<td>0</td>
<td>Real Number</td>
<td>None</td>
</tr>
</tbody>
</table>

The constants $K_i$ below are used in the following equation to calculate $I_{sp}$ as a function of pressure $P$ and temperature $T$:

$$
I_{sp}(P, T) = \frac{K_1 + K_2 P + K_3 P^2 + K_4 P^{K_5} + K_6 P^{K_7} + K_8 P^{K_9} + K_9 P^{K_{10}}}{(T/T_{ref})^{1+K_{11}P^{K_{12}}}}
$$

<table>
<thead>
<tr>
<th>Constant</th>
<th>Description</th>
<th>Default</th>
<th>Limits</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>K1</td>
<td>Isp coefficient.</td>
<td>2150</td>
<td>Real Number</td>
<td>m/sec</td>
</tr>
<tr>
<td>K2</td>
<td>Isp coefficient.</td>
<td>0</td>
<td>Real Number</td>
<td>m/(sec*kPa)</td>
</tr>
<tr>
<td>K3</td>
<td>Isp coefficient.</td>
<td>0</td>
<td>Real Number</td>
<td>m/(sec*kPa&lt;sup&gt;2&lt;/sup&gt;)</td>
</tr>
<tr>
<td>K4</td>
<td>Isp coefficient.</td>
<td>0</td>
<td>Real Number</td>
<td>m/(sec*kPa&lt;sup&gt;K5&lt;/sup&gt;)</td>
</tr>
<tr>
<td>K5</td>
<td>Isp coefficient.</td>
<td>0</td>
<td>Real Number</td>
<td>None</td>
</tr>
</tbody>
</table>
**Interactions**

**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.
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;
Variable

A variable.

Synopsis

Create Variable name
name = value

Description

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

![Parameter Create Dialog Box](image)

Figure: Parameter Create Dialog Box

Interactions

- Spacecraft Object
- Report Command
- Equation Command
- Vary Command
- Achieve Command
- Minimize Command

Examples

Create Variable pi Energy mu SMA
mu = 398600.4415;
pi = 3.14159265358979;
Energy = MySpacecraft.VMAG^2/2 - mu/r;
SMA = -mu/2/Energy;
VF13adOptimizer

Under Construction.

Synopsis

Under Construction.

Description

Under Construction.

Fields

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Fill This In.</td>
</tr>
<tr>
<td>Default</td>
<td>Fill This In.</td>
</tr>
<tr>
<td>Limits</td>
<td>Fill This In.</td>
</tr>
<tr>
<td>Units</td>
<td>Fill This In.</td>
</tr>
</tbody>
</table>

Examples

Example 10. Example Script

```
XYPlot

A XY Plot.

Synopsis

Create XYPlot name
name.field = value

Description

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.

Interactions

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.

Fields

IndVar

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.

Default
DefaultSC.A1ModJulian
Limits
Any user variable, array element, or spacecraft parameter
Units
None

Add

The Add field allows the user to add dependent variables to an xy-plot. All dependent variables are plotted on the y-axis vs 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.

Default
DefaultSC.EarthMJ2000Eq.X
Limits
Any user variable, array element, or spacecraft parameter
Grid

Units: None
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.
Default: On
Limits: On, Off
Units: None

SolverIterations

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.
Default: Off
Limits: On, Off
Units: None

ShowPlot

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.
Default: true
Limits: true, false
Units: None

Examples

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;
Commands
**Achieve**

Perform an achieve command

**Synopsis**

**Description**

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 preceded by a Vary, Maneuver, and Propagate command.

**Options**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Goal</strong></td>
<td>The option allows the user to select any single element user defined parameter, except a number, to Achieve.</td>
</tr>
<tr>
<td>Default</td>
<td>DefaultSC.Earth.RMAG</td>
</tr>
<tr>
<td>Limits</td>
<td>Spacecraft parameter, Array element, Variable, or any other single element user defined parameter, excluding numbers</td>
</tr>
<tr>
<td><strong>Arg1</strong></td>
<td>The Arg1 option is the desired value for after the solver has converged.</td>
</tr>
<tr>
<td>Default</td>
<td>42165</td>
</tr>
<tr>
<td>Limits</td>
<td>Real Number, Array element, Variable, or any user defined parameter that obeys the conditions of Chapter~\ref{Ch:ObjectsNResources} for the selected</td>
</tr>
<tr>
<td>Units</td>
<td>None</td>
</tr>
<tr>
<td><strong>Tolerance</strong></td>
<td>The Tolerance option sets Arg2. Arg2 is the convergence tolerance for Arg1.</td>
</tr>
<tr>
<td>Default</td>
<td>0.1</td>
</tr>
<tr>
<td>Limits</td>
<td>Real Number, Array element, Variable, or any user defined parameter &gt; 0</td>
</tr>
<tr>
<td>Units</td>
<td>None</td>
</tr>
<tr>
<td><strong>SolverName</strong></td>
<td>The SolverName option allows the user to choose which solver to assign to the Achieve command.</td>
</tr>
<tr>
<td>Default</td>
<td>DefaultDC</td>
</tr>
<tr>
<td>Limits</td>
<td>Any user defined differential corrector</td>
</tr>
<tr>
<td>Units</td>
<td>None</td>
</tr>
</tbody>
</table>

**Examples**

```
Achieve SolverName(Goal) = Arg1, Tolerance = Arg2
```
**BeginFiniteBurn**

Perform a begin finite burn

**Synopsis**

**Description**

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.

**Options**

**ManeuverName**

The ManeuverName option allows the user to choose between any previously created finite burn. As an example, to maneuver DefaultSC using DefaultFB, the script line would appear as Maneuver DefaultFB(DefaultSC).

- **Default**: DefaultFB
- **Limits**: Any finite burn existing in the resource tree or created in the script
- **Units**: None

**SpacecraftName**

The SpacecraftName option allows the user to select which spacecraft to maneuver using the maneuver selected in the ManeuverName option.

- **Default**: DefaultSC
- **Limits**: Any spacecraft existing in the resource tree or created in the script
- **Units**: None

**Examples**

```
BeginFiniteBurn ManeuverName (SpacecraftName);
EndFiniteBurn DefaultFB(DefaultSC);
```
BeginMissionSequence
Under construction.

Synopsis
Under construction.

Under construction.

Examples

Script Syntax
Under construction.

Script Examples
Under construction.
**CallGmatFunction**

Call a GMAT function

**Synopsis**

&lt;incomplete&gt;

**Definition**

GMAT functions are very useful and work nearly the same as they do in most programming languages. They may be invoked using the CallGmatFunction command covered here.

**Options**

**OutputList**

The OutputList option allows the user to set the output of Function to a user defined parameter.

- **Default**: None
- **Limits**: Variables, Arrays, S/C, Parameters, any other user-defined parameters, or blank. Multiple outputs must be expressed in a comma delimited list format
- **Units**: None

**InputList**

The InputList option allows the user to set the input of Function to a user defined parameter.

- **Default**: None
- **Limits**: Variables, Arrays, S/C, Parameters, any other user-defined parameters, or blank. Multiple inputs must be expressed in a comma delimited list format
- **Units**: None

**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.

- **Default**: None
- **Limits**: GMAT of Matlab Function
- **Units**: None

**Examples**

**Script Syntax**

```matlab
%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
```
Script Examples

% Matlab function call without inputs or outputs

% Syntax 1
GMAT clearAll

% Syntax 2
GMAT [ ] = clearAll( )
**CallMatlabFunction**

Call a MATLAB function

**Script Syntax**

<incomplete>

**Description**

This reference page is incomplete at this time.
ClearPlot

Clear a graphical plot window

Script Syntax

<incomplete>

Description

This reference page is incomplete at this time.
Else

Perform an else statement

Synopsis

Description

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.

Examples

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;
EndFiniteBurn
Under construction.

Synopsis
Under construction.

Under construction.

Examples
Script Syntax
Under construction.

Script Examples
Under construction.
**Equation**

Perform an equation command

**Synopsis**

**Description**

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.

**Options**

**Arg1**

The Arg1 option allows the user to set Arg1 to Arg2.

- **Default**: None
- **Limits**: Spacecraft Parameter, Array element, Variable, or any other single element user defined parameter
- **Units**: None

**Arg2**

The Arg2 option allows the user to define Arg1.

- **Default**: None
- **Limits**: Spacecraft Parameter, Array element, Variable, any other single element user defined parameter, or a combination of the aforementioned parameters using math operators
- **Units**: None

**Examples**

**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;
```
**For**
Iterate a specified number of times

**Script Syntax**

```
For Index = Start:[Increment:]End
    [script statement]
...  
EndFor
```

**Description**

The `For` command is a control logic statement that executes a nested set of script statements a specified number of times.

The command argument must have one of the following forms:

- `Index = Start:End`
  
  Increments `Index` from `Start` to `End` in steps of 1, repeating the script statements until `Index` is greater than `End`. If `Start` is greater than `End`, then the script statements do not execute.

- `Index = Start:Increment:End`
  
  Increments `Index` from `Start` to `End` in steps of `Increment`, repeating the script statements until `Index` is greater than `End` if `Increment` is positive and less than `End` if `Increment` is negative. If `Start` is less than `End` and `Increment` is negative, then the script statements do not execute.

*See Also:* If, While

**Options**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Index</strong></td>
<td>The independent variable in a <code>For</code> loop. <code>Index</code> is computed according to the arithmetic progression defined by the values for <code>Start</code>, <code>Increment</code>, and <code>End</code>.</td>
</tr>
<tr>
<td>Accepted Data Types</td>
<td>Variable</td>
</tr>
<tr>
<td>Allowed Values</td>
<td>$-\infty &lt; \text{Index} &lt; \infty$</td>
</tr>
<tr>
<td>Default Value</td>
<td>Variable I</td>
</tr>
<tr>
<td>Required</td>
<td>yes</td>
</tr>
<tr>
<td>Interfaces</td>
<td>GUI, script</td>
</tr>
</tbody>
</table>

<p>| <strong>Start</strong> | The <code>Start</code> parameter is the initial value for the <code>Index</code> parameter. |
| Accepted Data Types | Real Number, Variable, Array Element, or Scalar Real Object Parameter |
| Allowed Values | $-\infty &lt; \text{Start} &lt; \infty$ |
| Default Value | 1 |
| Required | yes |
| Interfaces | GUI, script |</p>
<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Increment</strong></td>
<td>The <strong>Increment</strong> parameter is used to compute the arithmetic progression of the loop Index such that the Ith pass through the loop is Start + i * Increment if the resulting value satisfies the constraint defined by <strong>End</strong>.</td>
</tr>
<tr>
<td>Accepted Data Types</td>
<td>Real Number, Variable, Array Element, or Scalar Real Object Parameter</td>
</tr>
<tr>
<td>Allowed Values</td>
<td>$-\infty &lt; \text{Increment} &lt; \infty$</td>
</tr>
<tr>
<td>Default Value</td>
<td>1</td>
</tr>
<tr>
<td>Required</td>
<td>no</td>
</tr>
<tr>
<td>Interfaces</td>
<td>GUI, script</td>
</tr>
<tr>
<td><strong>End</strong></td>
<td>The <strong>End</strong> parameter is the upper (or lower if <strong>Increment</strong> is negative) bound for the <strong>Index</strong> parameter.</td>
</tr>
<tr>
<td>Accepted Data Types</td>
<td>Real Number, Variable, Array Element, or Scalar Real Object Parameter</td>
</tr>
<tr>
<td>Allowed Values</td>
<td>$-\infty &lt; \text{End} &lt; \infty$</td>
</tr>
<tr>
<td>Default Value</td>
<td>10</td>
</tr>
<tr>
<td>Required</td>
<td>no</td>
</tr>
<tr>
<td>Interfaces</td>
<td>GUI, script</td>
</tr>
</tbody>
</table>

**GUI**

![For command dialog box](image)

The **For** command dialog box allows you to set the following **For** command parameters: **Index**, **Start**, **Increment**, and **End**. You can click in any text box and type a value, for example the number 1, or you can type any existing variable name, or an array element such as `myArray(1,1)`. If you right-click in any textbox, or click the button to the left of any text box, then the **ParameterSelectDialog** box will open and you can select from objects to set the **For** command parameters.

**Remarks**

If the loop executes, after execution of the loop, the value of **Index** remains the same as the last pass through the loop. If the loop does not execute, the value of **Index** is equal to the value before the loop.

Changes made to the index variable inside of a for loop are overwritten by the for loop statement. For example, the output from the following snippet

```plaintext
For I = 1:1:3
    I = 100
```
Changes made to the the **Start**, **Increment**, and **End** parameters made inside of a loop do not affect the behavior of the loop. For example, the output from the following snippet

```
J = 2
K = 2
L = 8
For I = J:K:L
   J = 1
   K = 5
   L = 100
   Report aReport I
EndFor
```

is

```
100
100
100
```

**Examples**

Propagate a spacecraft to apogee 3 times:

```plaintext
Create Spacecraft aSat
Create Propagator aPropagator
Create Variable I

BeginMissionSequence
   For I = 1:1:3
      Propagate aPropagator(aSat, {aSat.Apoapsis})
   EndFor

Index into an array:

```plaintext
Create Variable I J
Create Array anArray[10,5]
BeginMissionSequence
   For I = 1:10
      For J = 1:5
         anArray(I,J) = I*J
      EndFor
   EndFor
```
Global

Declare a global resource

Script Syntax

<incomplete>

Description

This reference page is incomplete at this time.
If
Perform an if command

Synopsis

Description

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.

Options

- **<If Command>**
  - Arg1 and Arg2 can be any of the following: Real Number, Array element, Variable, Spacecraft Parameter or any other user defined parameter.
  - **Default**: DefaultSC.ElapsedDays < 1.0
  - **Limits**: Arg1 < Arg2 and < can be >, <, >=, <=, ==, ~=
  - **Units**: None

- **<Statements>**
  - **Default**: None
  - **Limits**: Any script line that can be in the mission sequence
  - **Units**: None

The | option allows the user to set an OR operator in between <logical expression>s.

- **Default**: None
- **Limits**: None
- **Units**: None

The & option allows the user to set an AND operator in between <logical expression>s.

- **Default**: None
- **Limits**: None
- **Units**: None

Examples

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**

```plaintext
If <logical expression>;  
  <Statements>;  
EndIf;
```
• Compound If statement

```plaintext
If <logical expression> | <logical expression> & <logical expression>;  
   <Statements>;  
EndIf;
```

**Script Examples**

```plaintext
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;
```
**Maneuver**

Perform a maneuver command

**Synopsis**

**Description**

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.

**Options**

**BurnName**

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 Maneuver DefaultIB(DefaultSC).

- **Default**: DefaultIB
- **Limits**: Any impulsive burn existing in the resource tree or created in the script
- **Units**: None

**SpacecraftName**

The SpacecraftName field allows the user to select which spacecraft to maneuver using the maneuver selected in the BurnName field.

- **Default**: DefaultSC
- **Limits**: Any spacecraft existing in the resource tree or created in the script
- **Units**: None

**Examples**

**Script Syntax**

Maneuver BurnName (SpacecraftName);

**Script Examples**

% Impulsive Burn
Maneuver DefaultIB(DefaultSC);
**MarkPoint**

Mark a specific point on a graphical plot

**Script Syntax**

<incomplete>

**Description**

This reference page is incomplete at this time.
Minimize

Perform a minimize command

Synopsis

Description

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.

Options

<table>
<thead>
<tr>
<th>OptimizerName</th>
<th>The OptimizerName option allows the user to specify which solver to use to minimize the cost function.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>SQP1</td>
</tr>
<tr>
<td>Limits</td>
<td>Any existing fmincon solver</td>
</tr>
<tr>
<td>Units</td>
<td>None</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Arg</th>
<th>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.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>DefaultSC.ECC</td>
</tr>
<tr>
<td>Limits</td>
<td>Variable, Spacecraft parameter, or Array element</td>
</tr>
<tr>
<td>Units</td>
<td>None</td>
</tr>
</tbody>
</table>
NonlinearConstraint

Apply nonlinear constraint

Synopsis

Description

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.

Options

OptimizeName

The OptimizerName option allows the user to specify which solver to use in satisfying nonlinear constraints.

- **Default**: SQP1
- **Limits**: Any existing fmincon solver
- **Units**: None

{logical expression}

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.

- **Default**: DefaultSC.SMA = 7000
- **Limits**: Arg1 \( \leq \) Arg2 where \( \leq \) can be \( \geq \); \( \leq \); =
- **Units**: None

Examples

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 fminconSQP
NonLinearConstraint fminconSQP( Sat.SMA >= 7000a );
Optimize

Perform an optimize command

Synopsis

Description

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

Options

SolverName

The SolverName field allows the user to choose between any previously created optimizer for use in an optimization sequence. For example, to begin an optimization sequence using DefaultSQP, the script is Optimize DefaultSQP.

Default
Limits
Units

<Statements>

Default
Limits
Units

Examples

Script Syntax

Optimize SolverName;
  <Statements>;
EndOptimize;

Script Examples

% Beginning and ending syntax for the Optimize command
Optimize DefaultDC;
EndOptimize;
PenUp
Under construction.

Synopsis
Under construction.

Examples

Script Syntax
Under construction.

Script Examples
Under construction.
PenDown
Under construction.

Synopsis
Under construction.

Under construction.

Examples

Script Syntax
Under construction.

Script Examples
Under construction.
Propagate
Perform a propagate command

Synopsis

Description
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.

Reference Table

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BackProp</td>
<td>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.</td>
</tr>
<tr>
<td>Default</td>
<td>None</td>
</tr>
<tr>
<td>Limits</td>
<td>Backwards or None</td>
</tr>
<tr>
<td>Units</td>
<td>None</td>
</tr>
<tr>
<td>Mode</td>
<td>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.</td>
</tr>
<tr>
<td>Default</td>
<td>None</td>
</tr>
<tr>
<td>Limits</td>
<td>Synchronized or None</td>
</tr>
<tr>
<td>Units</td>
<td>None</td>
</tr>
<tr>
<td>PropagatorName</td>
<td>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.</td>
</tr>
<tr>
<td>Default</td>
<td>DefaultProp</td>
</tr>
<tr>
<td>Limits</td>
<td>Default propagator or any user-defined propagator</td>
</tr>
<tr>
<td>Units</td>
<td>None</td>
</tr>
<tr>
<td>SatListN</td>
<td>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.</td>
</tr>
</tbody>
</table>
**StopCondListN / Parameter**

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.

- **Default**: DefaultSC.ElapsedSecs
- **Limits**: Any single element user accessible spacecraft parameter followed by an equal sign
- **Units**: None

**StopCondListN / Condition**

The StopCondListN option allows the user to enter the propagator stopping condition's value for the StopCondListN Parameter field.

- **Default**: 8640.0
- **Limits**: Real Number, Array element, Variable, spacecraft parameter, or any user defined parameter
- **Units**: Dependant on the condition selected

### Examples

**Script Syntax**

Propagate Mode BackProp PropagatorName(SatList1,fStopCondList1g) ...
BackPropPropagatorName (SatListN, {StopCondList})

**Script Examples**

```plaintext
% 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 propagation settings
% 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 propagation settings
% Syntax #2
Propagate Prop1(Sat1,Sat2) {Sat1.ElapsedSecs = 8640.0, Sat2.MA = 90} ...
    Prop2(Sat3), {Sat3.TA = 0.0};
Report

Output a report

Synopsis

Description

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.

Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReportName</td>
<td>The ReportName option allows the user to specify the ReportFile for data output.</td>
</tr>
<tr>
<td>Default</td>
<td>None</td>
</tr>
<tr>
<td>Limits</td>
<td>Any ReportFile created</td>
</tr>
<tr>
<td>Units</td>
<td>None</td>
</tr>
<tr>
<td>DataList</td>
<td>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.</td>
</tr>
<tr>
<td>Default</td>
<td>None</td>
</tr>
<tr>
<td>Limits</td>
<td>Spacecraft parameter, Array, Variable, String, or any other single user defined parameter</td>
</tr>
<tr>
<td>Units</td>
<td>None</td>
</tr>
</tbody>
</table>

Examples

Script Syntax

Report  ReportName  DataList

Script Examples

%Report the time and position of DefaultSC
Report DefaultReport DefaultSC.A1ModJulian DefaultSC.X DefaultSC.Y DefaultSC.Z;
Save
Under construction.

Synopsis
Under construction.

Under construction.

Examples

Script Syntax
Under construction.

Script Examples
Under construction.
**SaveMission**

Save the mission to an external file

**Script Syntax**

<incomplete>

**Description**

This reference page is incomplete at this time.
**ScriptEvent**

Perform a ScriptEvent command

**Synopsis**

**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.

**Options**

<table>
<thead>
<tr>
<th>Options</th>
<th>Default</th>
<th>Limits</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;Statements&gt;</td>
<td>None</td>
<td>Any valid line of GMAT script</td>
<td>None</td>
</tr>
</tbody>
</table>

**Examples**

**Script Syntax**

```
BeginScript
  <Statements>
EndScript;
```

**Script Examples**

```
% Assignment Command inside Script Event

BeginScript
  GMAT testVar = 24;
EndScript;
```
Stop

Perform a stop command

Synopsis

Description

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.

Examples

Script Syntax

Stop

Script Examples

% Stop between propagation sequences
Propagate DefaultProp(DefaultSC) DefaultSC.ElapsedSecs = 8640.0;
Stop;
Propagate DefaultProp(DefaultSC) DefaultSC.ElapsedDays = 10.0;
**Target**

Solve for condition(s) by varying one or more parameters

**Script Syntax**

```
Target SolverName {[[SolveMode = value], [ExitMode = value]]]  
  Vary command ...  
  script statement ...  
  Achieve command ...  
EndTarget
```

**Note**

See sections the section called “Remarks” and the section called “Description” for this complex command. Multiple *Vary* and *Achieve* commands are permitted. Script statements can appear anywhere in the *Target* sequence.

**Description**

The *Target* and *EndTarget* commands are used to define a “Target Sequence” - to determine, for example, the maneuver components required to raise the orbit apogee to 42164 km. Another common targeting example is to determine the parking orbit orientation required to align a lunar transfer orbit with the moon. *Target* sequences in GMAT are general and these are just examples. Let’s define the quantities that you don’t know precisely but need to determine as the *control variables*. Define the conditions that must be satisfied as the *constraints*. A *Target* sequence numerically solves a boundary value problem to determine the value of the control variables required to satisfy the constraints. You define your control variables by using *Vary* commands and you define the problems constraints using *Achieve* commands. The *Target/EndTarget* sequence is an advanced command. The examples later in this section are useful for more detailed explanation.

See also: DifferentialCorrector, Optimize

**Options**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ApplyCorrections</strong></td>
<td>The <em>ApplyCorrections</em> button replaces the initial guess values specified in the <em>Vary</em> commands. If the <em>Target</em> sequence converged, the converged values are applied. If the <em>Target</em> sequence did not converge, the last calculated values are applied.</td>
</tr>
<tr>
<td>Accepted Data Types</td>
<td>N/A</td>
</tr>
<tr>
<td>Allowed Values</td>
<td>N/A</td>
</tr>
<tr>
<td>Default Value</td>
<td>N/A</td>
</tr>
<tr>
<td>Required</td>
<td>no</td>
</tr>
<tr>
<td>Interfaces</td>
<td>GUI</td>
</tr>
</tbody>
</table>
### Option Description

**ExitMode**  
The *ExitMode* parameter allows you to control the initial guess values for *Target* sequences nested in control flow. If *ExitMode* is set to *SaveAndContinue*, the solution of a *Target* sequence is saved and used as the initial guess for the next *Target* sequence execution. If *ExitMode* is set to *DiscardAndContinue*, then the solution is discarded and the initial guess values specified in the *Vary* commands are used for each target sequence execution.

- **Accepted Data Types**: PredefinedString
- **Allowed Values**: DiscardAndContinue, SaveAndContinue, Stop
- **Default Value**: DiscardAndContinue
- **Required**: no
- **Interfaces**: GUI, script

**SolveMode**  
The *SolveMode* parameter allows you to specify how the *Target* loop behaves during mission execution. When *SolveMode* is set to *Solve*, the target loop executes and attempts to solve the boundary value problem satisfying the targeter goals. When *SolveMode* is set to *RunInitialGuess*, the targeter does not attempt to solve the boundary value problem, and the commands in the *Target* sequence execute with the initial guess values defined in the *Vary* command.

- **Accepted Data Types**: PredefinedString
- **Allowed Values**: Solve, RunInitialGuess
- **Default Value**: Solve
- **Required**: no
- **Interfaces**: GUI, script

**SolverName**  
The *SolverName* parameter allows you to select the solver for a target sequence.

- **Accepted Data Types**: DifferentialCorrector
- **Allowed Values**: N/A
- **Default Value**: N/A
- **Required**: yes
- **Interfaces**: GUI, script

**GUI**

![GUI Image](image-url)
When you add a Target command to the mission sequence, an EndTarget command is automatically added as shown below.

Remarks

A Target/EndTarget sequence must contain at least one Vary command and at least one Achieve Command. See the Vary and Achieve command sections for details on the syntax for those commands. The first Vary command must occur before the first Achieve command. A Target command must be coupled with one and only one EndTarget command. Each Target command field in the curly braces is optional. You can omit the entire list and the curly braces and the default values will be used for Target configuration fields such as SolveMode and ExitMode. GMAT Target sequences can solve square problems (i.e. when the number of control variables equals the number of constraints), over-determined problems (the number of control variables is less than the number of constraints) and under-determined problems (the number of control variables is greater than the number of constraints). In any of these cases, there may not be a solution, and the type of solution found depends on the selection of the solver (currently, only differential correctors are supported). Assuming a solution to the problem exists, there is one solution for a square problem and many solutions for an under-determined problem. Problems with more goals than variables may not have a solution. If your problem is under-determined, consider using an Optimize sequence to find an optimal solution in the space of feasible solutions.

Caution

If you configure a Target sequence and get the error “Rmatrix error: matrix is singular”, then your control variables defined in the Vary commands do not affect the constraints defined in the Achieve commands. A common mistake in this case is that you forgot to apply a maneuver.

Examples

Use a Target sequence to solve for a root of an algebraic equation. Here we provide an initial guess of 5 for the control variable (or independent variable) x, and solve for the value of x that satisfies the constraint y = 3*x^3 + 2*x^2 - 4*x + 6. After executing this example you can look in the message window to see the solution for the variable x.

Create Variable x y
Create DifferentialCorrector aDC
BeginMissionSequence
Target aDC
    Vary aDC(x = 5)
    \[ y = 3x^3 + 2x^2 - 4x + 6 \]
    Achieve aDC(y = 0)
EndTarget

Use a Target sequence to raise orbit apogee. Here the control variable is the velocity component of an ImpulsiveBurn object. The constraint is that the position vector magnitude at orbit apogee is 42164 km.

Create Spacecraft aSat
Create Propagator aPropagator
Create Variable I
Create ImpulsiveBurn aBurn
Create DifferentialCorrector aDC

Create OrbitView EarthView
EarthView.Add = {Earth,aSat}
EarthView.ViewScaleFactor = 5

BeginMissionSequence
Target aDC
    Vary aDC(aBurn.Element1 = 1.0, \{Upper = 3\})
    Maneuver aBurn(aSat)
    Propagate aPropagator(aSat,\{aSat.Apoapsis\})
    Achieve aDC(aSat.RMAG = 42164)
EndTarget
**Toggle**

Perform a toggle command

**Synopsis**

**Description**

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.

**Options**

**OutputNames**

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.

- **Default** = DefaultOpenGL
- **Limits** = Any OpenGL, Report, XYplot, or any other Plot/Report type
- **Units** = None

**Arg**

The Arg option allows the user to turn off or on the data output to a Plot/Report.

- **Default** = On
- **Limits** = On or Off
- **Units** = None

**Examples**

**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})
**Vary**

Perform a vary command

**Synopsis**

**Description**

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.

**Options**

Parameters Associated with All Solvers

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SolverName</strong></td>
<td>The SolverName option allows the user to choose which solver to assign to</td>
</tr>
<tr>
<td>Default</td>
<td>DefaultDC</td>
</tr>
<tr>
<td>Limits</td>
<td>Any user defined solver</td>
</tr>
<tr>
<td>Units</td>
<td>None</td>
</tr>
<tr>
<td><strong>Variable</strong></td>
<td>The Variable option allows the user to select any single element user defined</td>
</tr>
<tr>
<td>Default</td>
<td>DefaultIB.V</td>
</tr>
<tr>
<td>Limits</td>
<td>Spacecrafty parameter, Array element, Variable, or any other single</td>
</tr>
<tr>
<td></td>
<td>element user defined parameter, excluding numbers</td>
</tr>
<tr>
<td>Units</td>
<td>None</td>
</tr>
<tr>
<td><strong>InitialGuess</strong></td>
<td>The InitialGuess option allows the user to set the initial guess for the</td>
</tr>
<tr>
<td>Default</td>
<td>0.5</td>
</tr>
<tr>
<td>Limits</td>
<td>Real Number, Array element, Variable, or any user defined parameter that</td>
</tr>
<tr>
<td></td>
<td>obeys the conditions for the selected Variable object</td>
</tr>
<tr>
<td>Units</td>
<td>km/s</td>
</tr>
<tr>
<td><strong>Lower</strong></td>
<td>The Lower option allows the user to set Arg3 to the lower bound of the</td>
</tr>
<tr>
<td>Default</td>
<td>0.0</td>
</tr>
<tr>
<td>Limits</td>
<td>Real Number, Array element, Variable, or any user defined parameter</td>
</tr>
<tr>
<td></td>
<td>(Upper &gt; Lower)</td>
</tr>
<tr>
<td>Units</td>
<td>None</td>
</tr>
<tr>
<td><strong>Upper</strong></td>
<td>The Upper option allows the user to set Arg4 to the upper bound of the</td>
</tr>
<tr>
<td>Default</td>
<td>3.14159</td>
</tr>
<tr>
<td>Limits</td>
<td>Real Number, Array element, Variable, or any user defined parameter</td>
</tr>
<tr>
<td></td>
<td>(Upper &gt; Lower)</td>
</tr>
</tbody>
</table>
Parameters Associated with Differential Corrector

**Perturbation**
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.

- **Default**: 1e-4
- **Limits**: Real Number, Array element, Variable, or any user defined parameter > 0
- **Units**: None

**MaxStep**
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.

- **Default**: 0.2
- **Limits**: Real Number, Array element, Variable, or any user defined parameter > 0
- **Units**: None

Parameters Associated with fmincon Optimizer

**Additive Scale Factor**
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: \( x_n = \frac{(x_d - a)}{m} \).

- **Default**: 0
- **Limits**: Real Number, Array element, Variable, or any user defined parameter
- **Units**: None

**Multiplicative Scale Factor**
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: \( x_n = \frac{(x_d - a)}{m} \).

- **Default**: 1.0
- **Limits**: Real Number, Array element, Variable, or any user defined parameter
- **Units**: None

**Examples**

**Script Syntax**

```plaintext
Vary SolverName(Variable=InitialGuess,{Perturbation=Arg1, MaxStep=Arg2, Lower=Arg3,...Upper=Arg4, AdditiveScalefactor=Arg6,MultiplicativeScalefactor=Arg6})
```
Script Examples

% Impulsive Burn Vary Command

Vary DefaultDC(DefaultIB.V = 0.5, {Perturbation = 0.0001, MaxStep = 0.2, Lower = 0, Upper = 3.14159});
**While**

Run a while loop

**Synopsis**

**Description**

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.

**Options**

<logical expression>  
Arg1 and Arg2 can be any of the following: Real Number, Array, Variable, Spacecraft Parameter or any other user defined parameter.

- **Default**: DefaultSC.ElapsedDays < 1.0
- **Limits**: Arg1 < Arg2 and < can be >, <, >=, <=, ==, ~=
- **Units**: None

<Statements>

- **Default**: None
- **Limits**: Any script line that can be in the mission sequence
- **Units**: None

|  
- **Default**: None
- **Limits**: None
- **Units**: None

&  

The & option allows the user to set an AND operator in between <logical expression>s.

- **Default**: None
- **Limits**: None
- **Units**: None

Examples

**Script Syntax**

- Simple While Loop

```plaintext
While <logical expression>;
    <Statements>;
EndWhile;
```

- Compound While Loop

```plaintext
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;
Part VI. Release Notes

Table of Contents

GMAT R2011a Release Notes ........................................................................................................ 163
  New Features ........................................................................................................................ 163
  Improvements ....................................................................................................................... 166
  Compatibility Changes ........................................................................................................... 168
  Fixed Issues .......................................................................................................................... 170
  Known Issues ....................................................................................................................... 170
GMAT R2012a Release Notes ........................................................................................................ 174
  New Features ........................................................................................................................ 174
  Improvements ....................................................................................................................... 176
  Compatibility Changes ........................................................................................................... 179
  Known & Fixed Issues .......................................................................................................... 180
GMAT R2011a Release Notes

The General Mission Analysis Tool (GMAT) version R2011a was released April 29, 2011 on the following platforms:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows (XP, Vista, 7)</td>
<td>Beta</td>
</tr>
<tr>
<td>Mac OS X (10.6)</td>
<td>Alpha</td>
</tr>
<tr>
<td>Linux</td>
<td>Alpha</td>
</tr>
</tbody>
</table>

This is the first release since September 2008, and is the 4th public release for the project. In this release:

- 100,000 lines of code were added
- 798 bugs were opened and 733 were closed
- Code was contributed by 9 developers from 4 organizations
- 6216 system tests were written and run nightly

New Features

OrbitView

GMAT’s old OpenGLPlot 3D graphics view was completely revamped and renamed OrbitView. The new OrbitView plot supports all of the features of OpenGLPlot, but adds several new ones:

- Perspective view instead of orthogonal
- Stars and constellations (with names)
- A new default Earth texture
- Accurate lighting
- Support for user-supplied spacecraft models in 3ds and POV formats.

All existing scripts will use the new OrbitView object automatically, with no script changes needed. Here’s a sample of what can be done with the new graphics:
User-Defined Celestial Bodies

Users can now define their own celestial bodies (Planets, Moons, Asteroids, and Comets) through the GMAT interface, by right-clicking on the Sun resource (for Planets, Asteroids, and Comets) or any other Solar System resource (for Moons). User-defined celestial bodies can be customized in many ways:

- Mu (for propagation), radius and flattening (for calculating altitude)
- User-supplied texture file, for use with OrbitView
- Ephemeris from two-body propagation of an initial Keplerian state or from a SPICE kernel
- Orientation and spin state

Ephemeris Output

GMAT can now output spacecraft ephemeris files in CCSDS-OEM and SPK formats by using the EphemerisFile resource. For each ephemeris, you can customize:

- Coordinate system
- Interpolation order
- Step size
- Epoch range

SPICE Integration for Spacecraft

Spacecraft in GMAT can now be propagated using data from a SPICE kernel rather than by numerical integration. This can be activated on the SPICE tab of the Spacecraft resource, or through the script. The following SPICE kernels are supported:

- SPK/BSP (orbit)
- CK (attitude)
- FK (frame)
Plugins

New features can now be added to GMAT through plugins, rather than being compiled into the GMAT executable itself. The following plugins are included in this release, with their release status indicated:

<table>
<thead>
<tr>
<th>Plugin</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>libMatlabPlugin</td>
<td>Beta</td>
</tr>
<tr>
<td>libFminconOptimizer (Windows only)</td>
<td>Beta</td>
</tr>
<tr>
<td>libGmatEstimation</td>
<td>Alpha (preview)</td>
</tr>
</tbody>
</table>

Plugins can be enabled or disabled through the startup file (`gmat_startup_file.txt`), located in the GMAT bin directory. All plugins are disabled by default.

GUI/Script Synchronization

For those that work with both the script and the graphical interface, GMAT now makes it explicitly clear if the two are synchronized, and which script is active (if you have several loaded). The possible states are:

- Synchronized (the interface and the script have the same data)
- GUI or Script Modified (one of them has been modified with respect to the other)
- Unsynchronized (different changes exist in each place)

The only state in which manual intervention is necessary is Unsynchronized, which must be merged manually (or one set of changes must be discarded). The following status indicators are available on Windows and Linux (on Mac, they appear as single characters on the GMAT toolbar).

Estimation [Alpha]

GMAT R2011a includes significant new state estimation capabilities in the libGmatEstimation plugin. The included features are:

- Measurement models
- Geometric
- TDRSS range
- USN two-way range
- Estimators
- Batch
- Extended Kalman
- Resources
- GroundStation
- Antenna
• Transmitter
• Receiver
• Transponder

Note
This functionality is alpha status, and is included with this release as a preview only. It has not been rigorously tested.

User Documentation

GMAT’s user documentation has been completely revamped. In place of the old wiki, our formal documentation is now implemented in DocBook, with HTML, PDF, and Windows Help formats shipped with GMAT. Our documentation resources for this release are:

• Help (shipped with GMAT, accessed through the Help > Contents menu item)
• Online Help (updated frequently, http://gmat.sourceforge.net/docs/)
• Video Tutorials (http://gmat.sourceforge.net/docs/videos.html)
• Help Forum (http://gmat.ed-pages.com/forum/)

Screenshot (📷)

GMAT can now export a screenshot of the OrbitView panel to the output folder in PNG format.

Improvements

Automatic MATLAB Detection

MATLAB connectivity is now automatically established through the libMatlabInterface plugin, if enabled in your gmat_startup_file.txt. We are no longer shipping separate executables with and without MATLAB integration. Most recent MATLAB versions are supported, though configuration is necessary.

Dynamics Model Numerics

All included dynamics models have been thoroughly tested against truth software (AGI STK, and A.I. Solutions FreeFlyer, primarily), and all known numeric issues have been corrected.
Script Editor [Windows]

GMAT’s integrated script editor on Windows is much improved in this release, and now features:

- Syntax highlighting for GMAT keywords
- Line numbering
- Find & Replace
- Active script indicator and GUI synchronization buttons

```matlab
9
10 Create Spacecraft DefaultSC;
11 GMAT DefaultSC.DateFormat = TAIModJulian;
12 GMAT DefaultSC.Epoch = 21545;
13 GMAT DefaultSC.CoordinateSystem = EarthMJ2000Eq;
14 GMAT DefaultSC.DisplayStateType = Cartesian;
15 GMAT DefaultSC.X = 7100;
16 GMAT DefaultSC.Y = 0;
17 GMAT DefaultSC.Z = 1300;
18 GMAT DefaultSC.VX = 0;
19 GMAT DefaultSC.VY = 7.349999999999996;
20 GMAT DefaultSC.VZ = 1;
21 GMAT DefaultSC.DryMass = 850;
22 GMAT DefaultSC.Cd = 2.2;
23 GMAT DefaultSC.Cr = 1.8;
24 GMAT DefaultSC.DragArea = 15;
25 GMAT DefaultSC.SRFArea = 1;
```

Regression Testing

The GMAT project developed a completely new testing system that allows us to do nightly, automated tests across the entire system, and on multiple platforms. The new system has the following features:

- Focused on GMAT script testing
- Written in MATLAB language
- Includes 6216 tests with coverage of most of GMAT’s functional requirements
- Allows automatic regression testing on nightly builds
- Compatible with all supported platforms

The project is also regularly testing the GMAT graphical interface on Windows using the SmartBear TestComplete tool. This testing occurs approximately twice a week, and is focused on entering and running complete missions through the interface and checking that the results match those generated in script mode.

Visual Improvements

This release features numerous visual improvements, including:

- A new application icon and splash screen (shown below)
- Many new, professionally-created icons
- A welcome page for new users
Compatibility Changes

Platform Support

GMAT supports the following platforms:

- Windows XP
- Windows Vista
- Windows 7
- Mac OS X Snow Leopard (10.6)
- Linux (Intel 64-bit)

With the exception of the Linux version, GMAT is a 32-bit application, but will run on 64-bit platforms in 32-bit mode. The MATLAB interface was tested with 32-bit MATLAB 2010b on Windows, and is expected to support 32-bit MATLAB versions from R2006b through R2011a.

Mac: MATLAB 2010a was tested, but version coverage is expected to be identical to Windows.

Linux: MATLAB 2009b 64-bit was tested, and 64-bit MATLAB is required. Otherwise, version coverage is expected to be identical to Windows.

Script Syntax Changes

The `BeginMissionSequence` command will soon be required for all scripts. In this release a warning is generated if this statement is missing.

The following syntax elements are deprecated, and will be removed in a future release:

<table>
<thead>
<tr>
<th>Resource</th>
<th>Field</th>
<th>Replacement</th>
</tr>
</thead>
<tbody>
<tr>
<td>DifferentialCorrector</td>
<td>TargeterTextFile</td>
<td>ReportFile</td>
</tr>
<tr>
<td>DifferentialCorrector</td>
<td>UseCentralDifferences</td>
<td>DerivativeMethod = &quot;CentralDifference&quot;</td>
</tr>
<tr>
<td>EphemerisFile</td>
<td>FileName</td>
<td>Filename</td>
</tr>
<tr>
<td>FiniteBurn</td>
<td>Axes</td>
<td></td>
</tr>
<tr>
<td>FiniteBurn</td>
<td>BurnScaleFactor</td>
<td></td>
</tr>
<tr>
<td>FiniteBurn</td>
<td>CoordinateSystem</td>
<td></td>
</tr>
<tr>
<td>FiniteBurn</td>
<td>Origin</td>
<td></td>
</tr>
<tr>
<td>FiniteBurn</td>
<td>Tanks</td>
<td></td>
</tr>
<tr>
<td>Resource</td>
<td>Field</td>
<td>Replacement</td>
</tr>
<tr>
<td>-----------------</td>
<td>------------------------</td>
<td>------------------------------------</td>
</tr>
<tr>
<td>FiniteBurn</td>
<td>CoordinateSystem</td>
<td>= CoordinateSystem</td>
</tr>
<tr>
<td>ImpulsiveBurn</td>
<td>&quot;Inertial&quot;</td>
<td>= &quot;MJ2000Eq&quot;</td>
</tr>
<tr>
<td>FiniteBurn</td>
<td>VectorFormat</td>
<td></td>
</tr>
<tr>
<td>ImpulsiveBurn</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FiniteBurn</td>
<td>V</td>
<td>Element1</td>
</tr>
<tr>
<td>ImpulsiveBurn</td>
<td>N</td>
<td>Element2</td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>Element3</td>
</tr>
<tr>
<td>FuelTank</td>
<td>PressureRegulated</td>
<td>PressureModel = PressureRegulated</td>
</tr>
<tr>
<td>OpenGLPlot</td>
<td></td>
<td>OrbitView</td>
</tr>
<tr>
<td>OrbitView</td>
<td>EarthSunLines</td>
<td>SunLine</td>
</tr>
<tr>
<td>OrbitView</td>
<td>ViewDirection = Vector</td>
<td>ViewDirection = [0 0 1]</td>
</tr>
<tr>
<td></td>
<td>ViewDirection = [0 0 1]</td>
<td></td>
</tr>
<tr>
<td>OrbitView</td>
<td>ViewPointRef</td>
<td>ViewPointReference</td>
</tr>
<tr>
<td>OrbitView</td>
<td>ViewPointRef = Vector</td>
<td>ViewPointReference = [0 0 1]</td>
</tr>
<tr>
<td></td>
<td>ViewPointRefVector =</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[0 0 1]</td>
<td></td>
</tr>
<tr>
<td>OrbitView</td>
<td>ViewPointVector =</td>
<td>ViewPointVector = [0 0 1]</td>
</tr>
<tr>
<td></td>
<td>Vector</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ViewPointVectorVector =</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[0 0 1]</td>
<td></td>
</tr>
<tr>
<td>SolarSystem</td>
<td>Ephemeris</td>
<td>EphemerisSource</td>
</tr>
<tr>
<td>Spacecraft</td>
<td>StateType</td>
<td>DisplayStateType</td>
</tr>
<tr>
<td>Thruster</td>
<td>X_Direction</td>
<td>ThrustDirection1</td>
</tr>
<tr>
<td></td>
<td>Y_Direction</td>
<td>ThrustDirection2</td>
</tr>
<tr>
<td></td>
<td>Z_Direction</td>
<td>ThrustDirection3</td>
</tr>
<tr>
<td></td>
<td>Element1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Element2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Element3</td>
<td></td>
</tr>
<tr>
<td>XYPlot</td>
<td>Add</td>
<td>YVariable</td>
</tr>
</tbody>
</table>
### Fixed Issues

733 bugs were closed in this release, including 368 marked “major” or “critical”. See the full report for details.

### Known Issues

There remain 268 open bugs in the project's Bugzilla database, 42 of which are marked “major” or “critical”. These are tabulated below.

<table>
<thead>
<tr>
<th>Resource</th>
<th>Field</th>
<th>Replacement</th>
</tr>
</thead>
<tbody>
<tr>
<td>XYPlot</td>
<td>Grid</td>
<td>ShowGrid</td>
</tr>
<tr>
<td>XYPlot</td>
<td>IndVar</td>
<td>XVariable</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Command</th>
<th>Old Syntax</th>
<th>New Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>Propagate</td>
<td>Propagate DefaultProp(sc)</td>
<td>Propagate BackProp DefaultProp(sc)</td>
</tr>
</tbody>
</table>
### Table 6. Multiple platforms

<table>
<thead>
<tr>
<th>Issue</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>407</td>
<td>Multi-Matlab run bug</td>
</tr>
<tr>
<td>636</td>
<td>MATLAB Callbacks on Linux and Mac</td>
</tr>
<tr>
<td>648</td>
<td>DOCUMENT BEHAVIOR - Final orbital state does not match for the two report methods</td>
</tr>
<tr>
<td>776</td>
<td>Batch vs Individual Runs different</td>
</tr>
<tr>
<td>1604</td>
<td>Keplerian Conversion Errors for Hyperbolic Orbits</td>
</tr>
<tr>
<td>1668</td>
<td>Decimal marker not flexible enough for international builds</td>
</tr>
<tr>
<td>1684</td>
<td>MMS script in GMAT takes 300 times longer than similar run in FreeFlyer</td>
</tr>
<tr>
<td>1731</td>
<td>Major Performance issue in GMAT Functions</td>
</tr>
<tr>
<td>1734</td>
<td>Spacecraft allows conversion for singular conic section.</td>
</tr>
<tr>
<td>1992</td>
<td>Determinant of &quot;large&quot; disallowed due to poor algorithm performance</td>
</tr>
<tr>
<td>2058</td>
<td>Can't set SRP Flux and Nominal Sun via GUI</td>
</tr>
<tr>
<td>2088</td>
<td>EOP file reader uses Julian Day</td>
</tr>
<tr>
<td>2147</td>
<td>Empty parentheses &quot;()&quot; are not caught in math validation</td>
</tr>
<tr>
<td>2313</td>
<td>Finite Burn/Thruster Tests Have errors &gt; 1000 km but may be due to script differences</td>
</tr>
<tr>
<td>2322</td>
<td>DOCUMENT: MATLAB interface requires manual configuration by user</td>
</tr>
<tr>
<td>2344</td>
<td>when a propagator object is deleted, its associated force model is not deleted</td>
</tr>
<tr>
<td>2349</td>
<td>Performance Issue in Force Modelling</td>
</tr>
<tr>
<td>2410</td>
<td>Ephemeris propagator has large numeric error</td>
</tr>
<tr>
<td>2416</td>
<td>STM Parameters are wrong when using Coordinate System other than EarthMJ2000Eq</td>
</tr>
</tbody>
</table>
### Table 7. Windows

<table>
<thead>
<tr>
<th>Issue Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>970</td>
<td>Matlab connection issue</td>
</tr>
<tr>
<td>1012</td>
<td>Quirky Numerical Issues 2 in Batch mode</td>
</tr>
<tr>
<td>1128</td>
<td>GMAT incompatible with MATLAB R14 and earlier</td>
</tr>
<tr>
<td>1417</td>
<td>Some lines prefixed by &quot;function&quot; are ignored</td>
</tr>
<tr>
<td>1436</td>
<td>Potential performance issue using many propagate commands</td>
</tr>
<tr>
<td>1528</td>
<td>GMAT Function scripts unusable depending on file ownership/permissions</td>
</tr>
<tr>
<td>1580</td>
<td>Spacecraft Attitude Coordinate System Conversion not implemented</td>
</tr>
<tr>
<td>1592</td>
<td>Atmosphere Model Setup File Features Not Implemented</td>
</tr>
<tr>
<td>2056</td>
<td>Reproducibility of script run not guaranteed</td>
</tr>
<tr>
<td>2065</td>
<td>Difficult to read low number in Spacecraft Attitude GUI</td>
</tr>
<tr>
<td>2066</td>
<td>SC Attitude GUI won't accept 0.0:90.0:0.0 as a 3-2-1 Euler Angle input</td>
</tr>
<tr>
<td>2067</td>
<td>Apply Button Sometimes Not Functional in SC Attitude GUI</td>
</tr>
<tr>
<td>2374</td>
<td>Crash when GMAT tries to write to a folder without write permissions</td>
</tr>
<tr>
<td>2381</td>
<td>TestComplete does not match user inputs to DefaultSC</td>
</tr>
<tr>
<td>2382</td>
<td>Point Mass Issue when using Script vs. User Input</td>
</tr>
</tbody>
</table>

### Table 8. Mac OS X

<table>
<thead>
<tr>
<th>Issue Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1216</td>
<td>MATLAB-&gt;GMAT not working</td>
</tr>
<tr>
<td>2081</td>
<td>Texture Maps not showing on Mac for OrbitView</td>
</tr>
<tr>
<td>2092</td>
<td>GMAT crashes when MATLAB engine does not open</td>
</tr>
<tr>
<td>2291</td>
<td>LSK file text ctrl remains visible when source set to DE405 or 2Body</td>
</tr>
<tr>
<td>2311</td>
<td>Resource Tree - text messed up for objects in folders</td>
</tr>
<tr>
<td>2383</td>
<td>Crash running RoutineTests with plots ON</td>
</tr>
</tbody>
</table>
Table 9. Linux

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1851</td>
<td>On Linux, STC Editor crashes GMAT on Close</td>
</tr>
<tr>
<td>1877</td>
<td>On Linux, Ctrl-C crashes GMAT if no MDIChildren are open</td>
</tr>
</tbody>
</table>
GMAT R2012a Release Notes

The General Mission Analysis Tool (GMAT) version R2012a was released May 23, 2012. This is the first public release in over a year, and is the 5th public release for the project. In this release:

- 52,000 lines of code were added
- Code and documentation was contributed by 9 developers from 2 organizations
- 6847 system tests were run every weeknight

This is a beta release. It has undergone extensive testing in many areas, but is not considered ready for production use.

New Features

Ground Track Plot

GMAT can now show the ground track of a spacecraft using the new GroundTrackPlot resource. This view shows the orbital path of one or more spacecraft projected onto a two-dimensional map of a celestial body, and can use any celestial body that you have configured. Here’s an example of the plot created as part of the default mission:

Orbit Designer

Sometimes you need to create a spacecraft in a particular orbit but don’t exactly know the proper orbital element values. Before, you had to make a rough estimate, or go back to the math to figure it out. Now, GMAT R2012a comes with a new Orbit Designer that does this math for you.

The Orbit Designer helps you create one of six different Earth-centered orbit types, each with a flexible array of input options:

- sun-synchronous
- repeat sun-synchronous
- repeat ground track
- geostationary
- molniya
- frozen
Once you've created your desired orbit, it is automatically imported into the Spacecraft resource for later use. Here's an example of a sun-synchronous orbit using the Designer. To open the **Orbit Designer**, click the button on the **Spacecraft** properties window.

![Orbit Designer Dialog](image)

**Eclipse Locator [alpha]**

We've done significant work toward having a robust eclipse location tool in GMAT, but this work is not complete. This release comes with an alpha-stage plugin (disabled by default) called **libEvent-Locator**. When enabled, this plugin adds a new **EclipseLocator** resource that can be configured to calculate eclipse entry and exit times and durations with respect to any configured Spacecraft and celestial bodies. The eclipse data can be reported to a text file or plotted graphically. Some known limitations include an assumption of spherical celestial bodies and a lack of light-time correction. This feature has not been rigorously tested, and may be brittle. We’ve included it here as a preview of what's coming in future releases.
C Interface [alpha]

Likewise, we've included an experimental library and plugin that exposes a plain-C interface to GMAT's internal dynamics model functionality. This interface is intended to fill a very specific need: to expose force model derivates from GMAT to external software, especially MATLAB, for use with an external integrator (though GMAT can do the propagation also, if desired). The interface is documented by an API reference for now.

Improvements

Dynamics Models

We've made lots of improvements to GMAT's already capable force model suite. Here's some highlights:

- GMAT now models Earth ocean and pole tides. This is a script-only option that can be turned on alongside an Earth harmonic gravity model; turn it on with a line like this:

  \[
  \text{ForceModel.GravityField.Earth.EarthTideModel = 'SolidAndPole'}
  \]

- You can now apply relativistic corrections using the checkbox on the properties for Propagator.

Solar System

GMAT can now use the DE421 and DE424 ephemerides for the solar system. These files are included in the installer, but are not activated by default. To use either of these ephemerides, double-click the SolarSystem folder and select it from the Ephemeris Source list. Or include the following script line:

\[
\text{SolarSystem.EphemerisSource = 'DE421'}
\]

There's also a new SolarSystem resource called SolarSystemBarycenter that represents the barycenter as given by the chosen ephemeris source (DE405, DE421, SPICE, etc.). This resource can be used directly in reports or as the origin of a user-defined coordinate system.
**TDB Input**

You can now input the epoch of a *Spacecraft* orbit in the TDB time system (in both Modified Julian and Gregorian formats).

**Mission Tree**

We've made significant improvements to the mission tree to make it more user-friendly to heavy users. The biggest improvement is that you can now filter the mission sequence in different ways to make complex missions easier to understand, for example by hiding non-physical events or collapsing the tree to only its top-level elements.

GMAT also now lets you name your mission sequence commands. Thus, instead of a sequence made up of commands like "Optimize1" and "Propagate3", you can label them "Optimize LOI" and "Prop to Periapsis". This example shows the *Ex_HohmannTransfer.script* sample with labeled commands.
Finally, we added the ability to undock the mission tree so you can place it and the resources tree side by side and see both at the same time. To undock the tree, right-click the Mission tab and drag it from its docked position. To dock it again, just close the new Mission window.

Mission Summary

You can now change the coordinate system shown in the Mission Summary on the fly: just change the Coordinate System list at the top of the window and the numbers will update. This feature can use any coordinate system currently defined in GMAT, including user-defined ones.

There's also a new Mission Summary - Physics-Based Commands that shows only physical events (Propagate commands, burns, etc.), and further data was added to both Mission Summary types.

Window Persistency

The locations of output windows are now saved with the mission in the script file. This means that when running a mission, all the output windows that were open when the mission was last saved will reappear in their old positions.
In addition, the locations of certain GMAT windows, like the mission tree, the script editor, and the application window itself are saved to the user preferences file (MyGMAT.ini).

**Switch to Visual Studio on Windows**

With this release, the official GMAT binaries for Windows are now compiled with Microsoft Visual Studio 2010 instead of GCC. The biggest benefit of this is in performance; we've seen up to a 50% performance improvement in certain cases in unofficial testing. It also leads to more an industry-standard development process on Windows, as the MinGW suite is no longer needed.

**New Icons**

The last release saw a major overhaul of GMAT's GUI icons. This time we've revised some and added more, especially in the mission tree.

**Training Manual**

The non-reference material in the GMAT User Guide has been overhauled, partially rewritten, and reformatted to form a new GMAT Training Manual. This includes the "Getting Started" material, some short how-to articles, and some longer tutorials. All of this information is included in the GMAT User Guide as well, in addition to reference material that is undergoing a similar rewrite later this year.

**Infrastructure**

The GMAT project has implemented several infrastructure improvements in the last year. The biggest of these was switching from our old Bugzilla system to JIRA for issue tracking.

This year also saw the creation of the GMAT Blog and the GMAT Plugins and Extensions Blog with a fair number of posts each, plus reorganizations for the wiki and the forums. We reactivated our two mailing lists, gmat-developers and gmat-users, but haven't seen much usage of each yet. And finally, we created a new mailing list, gmat-buildtest, for automated daily build and test updates.

**Compatibility Changes**

**Application Control Changes**

The command-line arguments for the GMAT executable have changed. See the following table for replacements.
### Old & New

<table>
<thead>
<tr>
<th>Old</th>
<th>New</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-help</td>
<td>--help, -h</td>
<td>Shows available options</td>
</tr>
<tr>
<td>-date</td>
<td>--version, -v</td>
<td>Shows GMAT build date</td>
</tr>
<tr>
<td>-ms</td>
<td>--start-server</td>
<td>Starts GMAT server on startup</td>
</tr>
<tr>
<td>-br filename</td>
<td>--run, -r scriptname</td>
<td>Builds and runs the script</td>
</tr>
<tr>
<td>-minimize</td>
<td>--minimize, -m</td>
<td>Minimizes GMAT window</td>
</tr>
<tr>
<td>-exit</td>
<td>--exit, -x</td>
<td>Exits GMAT after a script is run</td>
</tr>
</tbody>
</table>

### Script Syntax Changes

<table>
<thead>
<tr>
<th>Resource</th>
<th>Field</th>
<th>Replacement</th>
</tr>
</thead>
<tbody>
<tr>
<td>ForceModel</td>
<td>Drag</td>
<td>Drag.AtmosphereModel</td>
</tr>
<tr>
<td>Propagator</td>
<td>MinimumTolerance (BulirschStoer)</td>
<td>(none)</td>
</tr>
</tbody>
</table>

### Known & Fixed Issues

Many bugs were closed in this release, but a comprehensive list is difficult to create because of the move from Bugzilla to JIRA. See the "Bugs closed in R2012a" report in for a partial list.

All known issues that affect this version of GMAT can be seen in the "Known issues in R2012a" report in JIRA.
Index

A
Achieve, 120
Array, 51

B
BaryCenter, 52
BeginFiniteBurn, 121
BeginMissionSequence, 122
BodyNames, 52

C
CallGmatFunction, 123
CallMatlabFunction, 125
ClearPlot, 126

E
Else, 127
EndFiniteBurn, 128
EndFor, 130
EndTarget, 152
Equation, 129

F
For, 130

G
Global, 133

I
If, 134

M
Maneuver, 136
MarkPoint, 137
Minimize, 138

N
NonlinearConstraint, 139

O
Optimize, 141

P
PenDown, 143
PenUp, 142
Propagate, 144

R
Report, 147

S
Save, 148
SaveMission, 149
ScriptEvent, 150
Spacecraft, 97
Stop, 151

T
Target, 152
Toggle, 156

V
Vary, 157

W
While, 160