Goddard Mission Analysis Tool (GMAT)
Mathematical Specifications and User's Guide

NASA Goddard Space Flight Center
Greenbelt RD
Greenbelt MD 20771

Preface

What is GMAT, and why have we built it.

Contents

List of Figures

List of Tables

Part I

Mathematical Specifications

Chapter 1

Introduction

Chapter 2

Time

Time is the primary independent variable in GMAT. Time is used in integrating the equations of motion, and calculating planetary ephemerides, the orientations of planets and moons, and atmospheric density among others. GMAT uses three types of time systems depending on the type of calculations being performed: universal time systems based on the Earth's rotation with respect to the Sun; dynamic time systems that are based on the dynamic motion of the solar system and take into account relativistic effects; and atomic time systems based on the oscillation of the cesium atom. Each of these time systems has specific uses and is discussed below. In addition, universal, dynamic, and atomic time systems can be expressed in different time formats. The two time formats used in GMAT are the Modified Julian Date (MJD) format, and the Gregorian Date (GD) format. In the next section we'll take a look at the time systems used in GMAT, and when GMAT uses each time system. Then we'll look at the different time formats.

2.1 Time Systems

GMAT uses several different time systems in physical models and spacecraft dynamics modelling. The choice of time system for a particular calculation is determined by which time system is most natural and convenient, as well as the accuracy required. In general, for determining Earth's orientation at a given epoch, we use one of several forms Universal Time (UT), because universal time is based on the Earth's rotation with respect to the Sun. Planetary ephemerides are usually provided with time in a dynamic time system, because dynamic time is the independent variable in the dynamic theories and ephemerides. The independent variable in spacecraft equations of motion in GMAT is time expressed in an atomic time system. Let's look at each of these three systems, starting with atomic time.

2.1.1 Atomic Time: TAI and A.1

Atomic time (AT) is a highly accurate time system which is independent of the rotation of the Earth.[2] Therefore, AT is a natural system for integrating a spacecraft's equations of motion. AT is defined in terms of the oscillations of the cesium atom at mean sea level. The duration of the SI second is defined to be 9,192,631,770 oscillations of the cesium nuclide 133Ce. Two atomic times systems are are used in GMAT: A.1, and international atomic time (TAI). A.1 is in advance of TAI by 0.0343817 seconds.

$$A.1 = TAI + 0.0343817sec \tag{2.1}$$

where

$$TAI = UTC + \Delta AT \tag{2.2}$$

and ΔAT is the number of leap seconds, added since 1972, needed to keep $\mid UTC - UT1 \mid \leq 0.9$sec. GMAT reads ΔAT from the file named *tai-utc.dat*. Currently, GMAT uses A.1 time as the independent variable in the equations of motion. TAI is used as a time system for defining spacecraft state information.

Now let's look at the universal time system.

2.1.2 Universal Time: UTC and UT1

All of the universal time (UT) time scales are based on the Earth's rotation with respect to a fixed point (sidereal time) or with respect to the Sun (solar time). The observed universal time (UTO) is determined from observations of stellar transits to determine mean local sidereal time. UT1 is UTO corrected for the Earth's polar motion and is used when the instantaneous orientation of the Earth is needed. UTC is the basis for all civil time standards. It is also known as Greenwich mean time (GMT) and Zulu time (Z). The UTC time unit is defined to be an SI second, but UTC is kept within 0.9 seconds of UT1 by occasional leap second adjustments. The equation relating UTC and UT1 is

$$UTC = UT1 - \Delta UT1 \tag{2.3}$$

In GMAT, $\Delta UT1$ is read from the *finals.data* file provided by the National Earth Orientation Service (NEOS) at the U.S. Naval Observatory. The file, containing the latest measurements and predictions, can be found at *http://maia.usno.navy.mil/*. GMAT uses UTC as a time system to define spacecraft state information. UT1 is used to determine the Greenwich hour angle and for the sidereal time portion of FK5 reduction.

2.1.3 Dynamic Time: TT, TDB and TCB

Dynamical time is the independent variable in the dynamical theories and ephemerides. This class of time scales contains terrestrial time (TT), Barycentric Dynamical time (TDB), and Barycentric Coordinate Time (TCB). TDB is the independent variable in the equations of motion referred to the solar system barycenter. It is also the coordinate time in the theory of general relativity. Despite the fact that the Jet Propulsion Laboratory (JPL) J2000.0 ephemerides are referred to in TDB, TT is frequently used. This is because TT and TDB always differ by less than 0.002 sec. As higher accuracy or more sensitive missions are planned, the difference may need to be distinguished. In this section we'll discuss how to calculate TT, TDB and TCB, and discuss where each is used in GMAT.

TT is the independent variable in the equations of motion referred to the Earth's center. It is also the proper time in the theory of general relativity. The unit of TT is a day of 86400 SI seconds at mean sea level. In GMAT, TT is used in FK5 reduction, and as an intermediate time system in the calculation of TDB and TCB. TT can be calculated from the following equation:

$$TT = TAI + 32.184 \text{ sec} \tag{2.4}$$

Calculating TDB exactly is a complicated process that involves iteratively solving a transcendental equation. For this reason, it is convenient to use the following approximation

$$TDB \approx TT + 0.001658 \sin M_E + 0.00001385 \sin 2M_E \tag{2.5}$$

where M_E is the Earth's mean anomaly with respect to the sun and is given approximately as

$$M_E \approx 357.5277233 + 35,999.05034 T_{TT} \tag{2.6}$$

where T_{TT} is the time in TT expressed in the Julian Century format. T_{TT} can be calculated from

$$T_{TT} = \frac{JD_{TT} - 2,451,545.0}{36,525} \tag{2.7}$$

where JD_{TT} is the time in TT expressed in the Julian Date format. For a more complete discussion of the TDB time system, see Vallado[2] (pp. 195-198) and Seidelmann[3] (pp. 41-48). GMAT uses TDB in the JPL ephemerides files.

The last dynamic time system GMAT uses is Barycentic Coordinate Time. In 1992, the IAU adopted this system and clarified the relationships between space-time coordinates.[3] In general, calculating TCB requires a four-dimensional space-time transformation that is well beyond the scope of this discussion. However, TCB can be approximated using the following equation:

$$TCB - TDB = L_B(JD - 2443144.5)86400 \tag{2.8}$$

The present estimate of the value of L_B is $1.550505x10^{-8}$ $(+/- 1x10^{-14})$ (Fukushima et al., Celestial Mechanics, 38, 215, 1986). It is important to note that the main difference between TDB and TCB is a secular drift, and that as of the J2000 Epoch, the difference was approximately 11.25 seconds and growing.[4] GMAT uses time in the TCB system to evaluate the IAU data for the spin axes and prime meridian locations of all planets and moons except for Earth.[1]

2.2 Time Formats

There are two time formats that GMAT uses to represent time in the systems discussed above. These formats are called the Gregorian Date (GD), and the Julian Date (JD). The difference between the GD and JD formats is how they represent the Year, Month, Day, Hours, Minutes, and Seconds of a given epoch. The GD format is well known, and the J2000 epoch is expressed as, 01 Jan 2000 12:00:00.000. The reference epoch for the GD calendar is the beginning of the Christian Epoch. The JD format represents an epoch as a continuous number containing the day and the fraction of day. The reference epoch for the JD format is discussed below.

In the next two sections we'll look at how to convert an epoch in a given time system from the GD format to the JD format, and vice versa.

2.2.1 Julian Date and Modified Julian Date

The Julian date is a time format in which we can express a time known in any of the Atomic, Universal or Dynamic time systems. The Julian Date is composed of the Julian day number and the decimal fraction of the current day. Seidelmann[3] (pp. 55-56) says "The Julian day number represents the number of days that has elapsed, at Greenwich noon on the day designated, since ...the epoch noon Jan 1 4713 B.C. in the Julian proleptic calendar. The Julian date (JD) corresponding to any instant is, by simple extension to this concept, the Julian day number followed by the fraction of the day elapsed since the preceding noon".

The fundamental epoch for most astrodynamic calculations is the J2000.0 epoch.[3] This epoch is GD 01 Jan 2000 12:00:00.000 in the TDB time system and is expressed as JD 2451545.0 TDB. For numerical reasons it is often convenient to work in a Modified Julian Data (MJD) format to ensure we can capture enough significant figures using double precision computers. In GMAT the MJD system is defined as

$$MJD = JD - 2,430,000.5 \tag{2.9}$$

To convert between Julian Date format and Gregorian Date format, GMAT uses Algorithm 14 from Vallado[2]

$$JD = 367Y - Int\left(\frac{7\left(Y + Int\left(\frac{M+9}{12}\right)\right)}{4}\right) + Int\left(\frac{275M}{9}\right) + D + 1,721,013.5 + \frac{\frac{\frac{s}{60} + M}{60} + H}{24} \tag{2.10}$$

where Y is the four digit year, Int signifies real truncation, and M, D, H, M and S are month, day, hour, min, and second respectively. This equation is valid for the time period 01 Mar. 1900 to 28 Feb. 2100.

2.2.2 Gregorian Date

The Gregorian Date format is primarily used as a time system in which to enter state information in GMAT. GD is not a convenient time format for most mathematical calculations. Hence, GMAT often takes input in the GD format and converts it to a MJD format for use internally. The algorithm for converting from GD to MJD is taken from Vallado[2] and reproduced here verbatim.

$$
\begin{aligned}
&T_{1900} = \frac{JD - 2,415,019.5}{365.25} \\
&Year = 1900 + \text{TRUNC}(T_{1900}) \\
&LeapYrs = \text{TRUNC}((Year - 1900 - 1)(0.25)) \\
&Days = (JD - 2,415,019.5) - ((Year - 1900)(365.0) + LeapYrs) \\
&\text{IF} Days < 1.0 \text{ THEN} \\
&\qquad Year = Year - 1 \\
&\qquad LeapYrs = \text{TRUNC}((Year - 1900 - 1)(0.25)) \\
&\qquad Days = (JD - 2,415,019.5) - ((Year - 1900)(365.0) + LeapYrs) \\
&\text{If } (Year \text{ Mod } 4) = 0 \text{ Then} \\
&\qquad LMonth[2] = 29 \\
&DayofYr = \text{TRUNC}(Days) \\
&\text{Sum days in each month until } LMonth + 1 \text{ summation} > DayofYr \\
&Mon = \#\text{ of months in summation} \\
&Day = DayofYr - LMonth \text{ summation} \\
&\tau = (Days - DayofYr)24 \\
&h = \text{TRUNC}(Temp) \\
&min = \text{TRUNC}((Temp - h)60) \\
&s = \left(Temp - h - \frac{min}{60}\right) 3600
\end{aligned}
\tag{2.11}
$$

2.3 Conclusions

In this chapter we looked at three time systems that GMAT uses to perform internal calculations: atomic time, universal time, and dynamic time. Atomic time is used to integrate spacecraft equations of motion,

while universal time is used to determine the sidereal time and greenwhich hour angle for use in FK5 reduction. Dynamic time systems are used in the JPL ephemerides and in the IAU planetary orientation data. Time, in any of these time systems, are represented in two formats: the Gregorian Date, and Julian Date. We looked at how to convert between different time systems, and between different time formats.

Chapter 3

Coordinate Systems

There are numerous coordinate systems used in space mission analysis, that when used appropriately can greatly simplify the work and yield insight that is not obvious otherwise. Some examples are equatorial and ecliptic systems, and rotating coordinate systems based on the relative motion of two bodies such as the Earth and Moon. GMAT has the capability to calculate many parameters in different coordinate systems, and these parameters can then be used in plots, report, targeters, control flow statements and stopping conditions to name a few.

In this chapter we investigate how GMAT performs coordinate system transformations, and how different coordinate systems are defined. We begin by defining some notation. Next we look at how to transform a vector and its first derivative from one coordinate system to another, when the coordinate systems are translating and rotating with respect to one another. Finally, we look at each coordinate system defined in GMAT and discuss how to find its rotation matrix and the first derivative of the rotation matrix to rotate to the J2000 coordinate system.

3.1 General Coordinate System Transformations

GMAT has the capability to take a position and velocity vector in one coordinate system, and convert them to another coordinate that may be both translating and rotating with respect to the original system. In this section we derive the equations governing coordinate system transformations and describe the algorithm GMAT uses to transform position and velocity vectors.

Let's start by defining some notation. In Fig. 3.1, we see an illustration of a point o drawn with respect to two coordinate systems $\mathcal{F}_i$ and $\mathcal{F}_f$. The vector $\mathbf{r}_i^o$ is the position vector of point o with respect to frame $\mathcal{F}_\rangle$. The vector $\mathbf{r}_f^o$ is the position vector of point o with respect to $\mathcal{F}_f$. $\mathbf{r}_{fi}$ is the vector from the origin of $\mathcal{F}_i$ to $\mathcal{F}_f$. Let's define the rotation matrix that rotates from $\mathcal{F}_i$ to $\mathcal{F}_f$ as $\mathbf{R}_{fi}$. Finally, let's define the angular velocity $\boldsymbol{\omega}_{fi}$ as the angular velocity of $\mathcal{F}_i$ with respect to $\mathcal{F}_f$. To simplify the notation, we assume that a vector is expressed in the frame denoted by the right-most subscript. If we need to express a vector in another coordinate system, we use curly brackets. As an example, $\{\mathbf{r}_i^o\}_f$ is the position of point o, with respect to frame $\mathcal{F}_i$, expressed in frame $\mathcal{F}_f$. In summary, we have

$\mathbf{r}_i^o$	Position vector of point o w/r/t frame $\mathcal{F}_i$ expressed in $\mathcal{F}_i$
$\{\mathbf{r}_i^o\}_f$	Position vector of point o w/r/t frame $\mathcal{F}_i$ expressed in $\mathcal{F}_f$
$\mathbf{r}_{fi}$	Position vector from origin of $\mathcal{F}_i$ to origin of $\mathcal{F}_f$ expressed in $\mathcal{F}_f$
$\mathbf{R}_{fi}$	Rotation matrix from frame $\mathcal{F}_i$ to $\mathcal{F}_f$
$\boldsymbol{\omega}_{fi}$	Angular velocity of frame $\mathcal{F}_i$ w/r/t $\mathcal{F}_f$, expressed in frame $\mathcal{F}_i$
$\{\boldsymbol{\omega}_{fi}\}_f$	Angular velocity of frame $\mathcal{F}_i$ w/r/t $\mathcal{F}_f$, expressed in frame $\mathcal{F}_f$

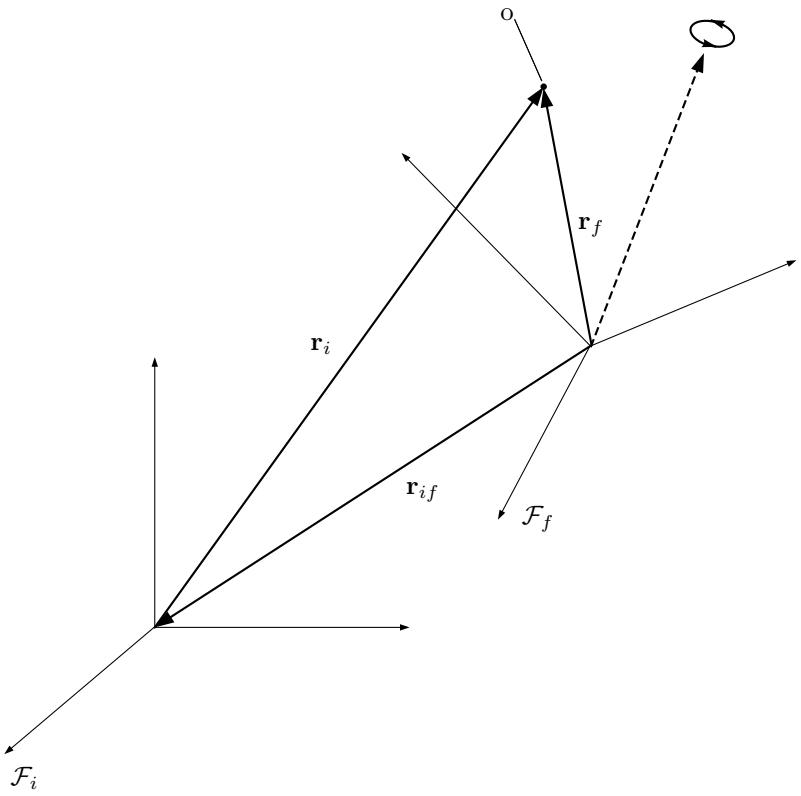

Figure 3.1: Reference Frame Diagram

To further simplify the notation, let's drop the superscript "o" from $\mathbf{r}_i^o$ and $\mathbf{r}_f^o$. Then, from inspection of Figure 3.1 we can write

$$\boxed{\mathbf{r}_f = \underbrace{\mathbf{R}_{fi}\mathbf{r}_i}_{Rot.} + \underbrace{\mathbf{r}_{if}}_{Trans.}} \tag{3.1}$$

Equation (3.1) is the equation used to convert a vector known in frame $\mathcal{F}_i$ to a vector in frame $\mathcal{F}_f$, where both a rotation and a translation are required. The first term in Eq. (3.1) is the term that performs the rotation portion of the transformation. Here, $\mathbf{r}_i$ is the position vector w/r/t to $\mathcal{F}_i$ and is expressed in $\mathcal{F}_i$. $\mathbf{R}_{fi}$ is the rotation matrix that rotates from $\mathcal{F}_i$ to $\mathcal{F}_f$. $\mathbf{r}_{if}$ is the vector that goes from the origin of $\mathcal{F}_f$ to the origin of $\mathcal{F}_i$, and is expressed in $\mathcal{F}_f$.

We also need to be able to determine the time rate of change of $\mathbf{r}$ in frame $\mathcal{F}_f$ if we know the time rate of change of $\mathbf{r}$ in $\mathcal{F}_i$. To determine the equation that describes the transformation, we must take the derivative of Eq. (3.1) with respect to time.

$$\frac{d\mathbf{r}_f}{dt} = \frac{d\mathbf{R}_{fi}\mathbf{r}_i}{dt} + \frac{d\mathbf{r}_{if}}{dt} \tag{3.2}$$

We can expand this to obtain

$$\frac{d\mathbf{r}_f}{dt} = \frac{d\mathbf{R}_{fi}}{dt}\mathbf{r}_i + \mathbf{R}_{fi}\frac{d\mathbf{r}_i}{dt} + \frac{d\mathbf{r}_{if}}{dt} \tag{3.3}$$

In Eq. (3.3) we see a term that contains the time derivative of the rotation matrix from $\mathcal{F}_i$ to $\mathcal{F}_f$. It can be shown that this can be expressed simply as

$$\frac{d\mathbf{R}_{fi}}{dt} = \mathbf{R}_{fi}\boldsymbol{\omega}_{fi}^{\mathrm{x}} = \{\boldsymbol{\omega}_{fi}^{\mathrm{x}}\}_f\mathbf{R}_{fi} \tag{3.4}$$

where $\boldsymbol{\omega}_{fi}$ is the angular velocity of $\mathcal{F}_i$ with respect to $\mathcal{F}_f$ expressed in $\mathcal{F}_i$. The skew symmetric matrix $\boldsymbol{\omega}^x$

is defined as

$$\boldsymbol{\omega}^x = \begin{pmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{pmatrix} \tag{3.5}$$

In summary, using Eq. (3.4) to transform a derivative vector from $\mathcal{F}_i$ to $\mathcal{F}_f$ we can use any of the following three equations:

$$\boxed{\frac{d\mathbf{r}_f}{dt} = \underbrace{\mathbf{R}_{fi}\boldsymbol{\omega}_{fi}^{\mathrm{X}}\mathbf{r}_i + \mathbf{R}_{fi}\frac{d\mathbf{r}_i}{dt}}_{Rot.} + \underbrace{\frac{d\mathbf{r}_{if}}{dt}}_{Trans.}} \tag{3.6}$$

$$\boxed{\frac{d\mathbf{r}_f}{dt} = \underbrace{\{\boldsymbol{\omega}_{fi}^{\mathrm{X}}\}_f\mathbf{R}_{fi}\mathbf{r}_i + \mathbf{R}_{fi}\frac{d\mathbf{r}_i}{dt}}_{Rot.} + \underbrace{\frac{d\mathbf{r}_{if}}{dt}}_{Trans.}} \tag{3.7}$$

$$\boxed{\frac{d\mathbf{r}_f}{dt} = \underbrace{\frac{d\mathbf{R}_{fi}}{dt}\mathbf{r}_i + \mathbf{R}_{fi}\frac{d\mathbf{r}_i}{dt}}_{Rot.} + \underbrace{\frac{d\mathbf{r}_{if}}{dt}}_{Trans.}} \tag{3.8}$$

We choose between Eqs. (3.6), (3.7), or (3.8) depending on which type of information we have and which frame is most convenient to express the angular velocity $\boldsymbol{\omega}_{fi}$ in. In general, we know $\mathbf{r}_i$ and $d\mathbf{r}_i/dt$. To perform the transformation we need to determine $\mathbf{R}$, $\dot{\mathbf{R}}$, and $d\mathbf{r}_{if}/dt$ and these quantities depend on $\mathcal{F}_i$ and $\mathcal{F}_f$.

One of the difficulties in implementing coordinate system transformations in GMAT is that we often can't calculate $\mathbf{R}_{fi}$ and $\dot{\mathbf{R}}_{fi}$ directly. Hence, we need to choose a convenient intermediate frame. We choose the axis system defined by Earth's mean equinox and mean equator at the J2000 epoch, denoted $\mathcal{F}_{J_{2k}}$, as the intermediate reference frame for all transformations that require an intermediate transformation. This choice is motivated by the fact that most of the data needed to calculate $\mathbf{R}$ and $\dot{\mathbf{R}}$ is given so that it is fast and convenient to calculate $\mathbf{R}_{J_{2k},i}$, and $\dot{\mathbf{R}}_{J_{2k},i}$.

The steps taken to perform a general coordinate transformation in GMAT are described below and illustrated in Fig. 3.2. The first step is to perform a rotation from $\mathcal{F}_i$ to $\mathcal{F}_{J_{2k}}$ about the origin of $\mathcal{F}_i$. We define this intermediate system as $\mathcal{F}_1$. No translation is performed in step one. Using only the rotation portions of from Eqs. (3.1) and (3.8) we see that we obtain

$$\{\mathbf{r}_i\}_1 = \mathbf{R}_{J_{2k},i}\mathbf{r}_i \tag{3.9}$$

$$\left\{\frac{d\mathbf{r}_i}{dt}\right\}_1 = \frac{d\mathbf{R}_{J_{2k},i}}{dt}\mathbf{r}_i + \mathbf{R}_{J_{2k},i}\frac{d\mathbf{r}_i}{dt} \tag{3.10}$$

The second step is to perform a translation from the origin of $\mathcal{F}_i$ to the origin of $\mathcal{F}_f$. We define this second intermediate system as $\mathcal{F}_2$. $\mathcal{F}_2$ has the same origin as $\mathcal{F}_f$ but has the same axes as $\mathcal{F}_{J_{2k}}$. From inspection of Fig.3.2 we can see that

$$\{\mathbf{r}_i\}_1 = \{\mathbf{r}_{Ri}\}_{J_{2k}} + \{\mathbf{r}_{fR}\}_{J_{2k}} + \{\mathbf{r}_f\}_2 \tag{3.11}$$

Solving for $\mathbf{r}_f$ we obtain

$$\{\mathbf{r}_f\}_2 = \{\mathbf{r}_i\}_1 - \{\mathbf{r}_{Ri}\}_{J_{2k}} - \{\mathbf{r}_{fR}\}_{J_{2k}} \tag{3.12}$$

where $\{\mathbf{r}_{Ri}\}_{J_{2k}}$ is the vector from the origin of $\mathcal{F}_i$ to the origin of $\mathcal{F}_R$ expressed in $\mathcal{F}_{J2k}$. Similarly $\{\mathbf{r}_{fR}\}_{J_{2k}}$ is the vector from the origin of $\mathcal{F}_R$ to the origin of $\mathcal{F}_f$ expressed in $\mathcal{F}_{J2k}$. Because the vector $\{\mathbf{r}_f\}_2$ is expressed in an inertial system we can we can take the derivative of Eq .(3.12) to obtain

$$\{\mathbf{v}_f\}_2 = \{\mathbf{v}_i\}_1 - \{\mathbf{v}_{Ri}\}_{J_{2k}} - \{\mathbf{v}_{fR}\}_{J_{2k}} \tag{3.13}$$

where $\{\mathbf{v}_{Ri}\}_{J_{2k}}$ is the velocity of the origin of $\mathcal{F}_R$ w/r/t the origin of $\mathcal{F}_i$. Similarly, $\{\mathbf{v}_{fR}\}_{J_{2k}}$ is the velocity of the origin of $\mathcal{F}_f$ w/r/t the origin of $\mathcal{F}_R$. Finally, we perform a rotation from $\mathcal{F}_{J_{2k}}$ to $\mathcal{F}_f$ about the origin of $\mathcal{F}_f$ to obtain the desired quantities.

$$\mathbf{r}_f = \mathbf{R}_{f,J_{2k}}\{\mathbf{v}_f\}_2 \tag{3.14}$$

$$\frac{d\mathbf{r}_f}{dt} = \frac{d\mathbf{R}_{f,J_{2k}}}{dt}\left\{\mathbf{r}_f\right\}_2 + \mathbf{R}_{f,J_{2k}}\left\{\mathbf{v}_f\right\}_2 \tag{3.15}$$

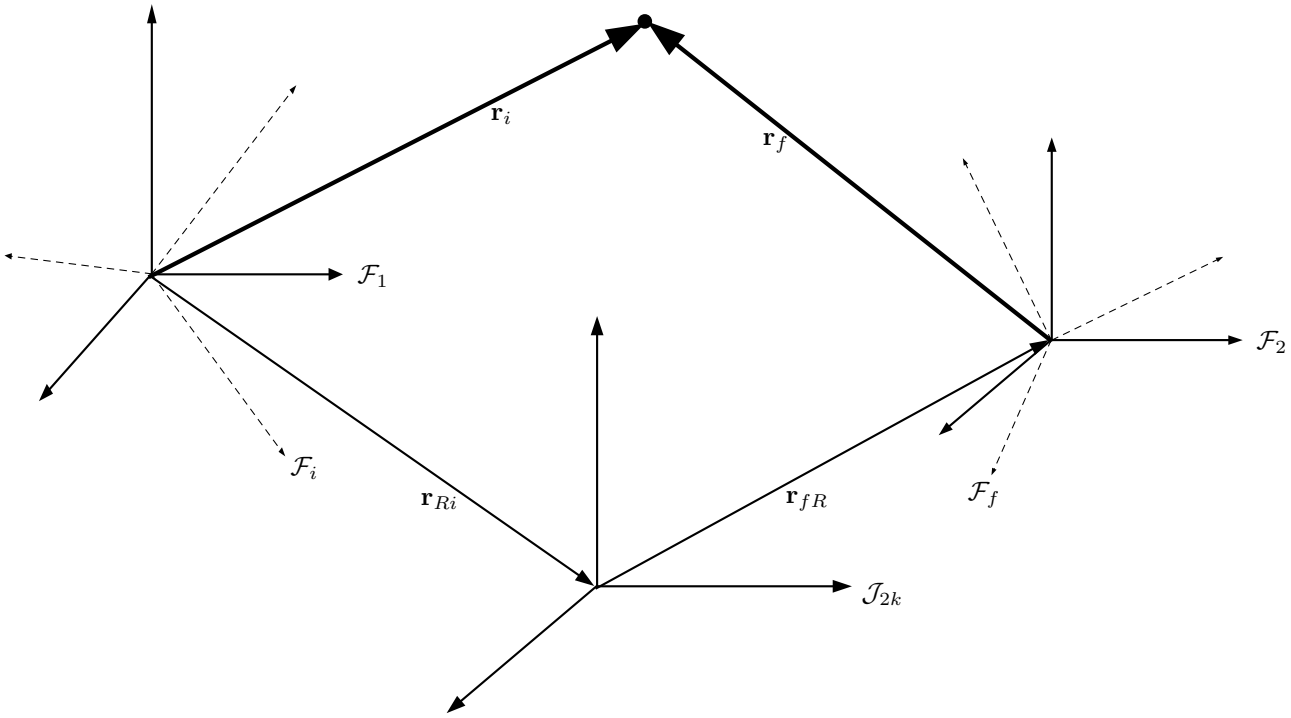

Figure 3.2: General Coordinate System Transformation in GMAT

To perform the above operations, we need to be able to calculate the rotation matrix between any coordinate system and $\mathcal{F}_{J_{2k}}$, and the derivative of the rotation matrix. In the following sections we calculate these matrices for the systems used in GMAT. We assume that we want the transformation from some generic frame $\mathcal{F}_i$ to $\mathcal{F}_{J_{2k}}$ which is the Earth Mean J2000 Equatorial (MJ2000Eq) system. The rotation matrix from $\mathcal{F}_{J_{2k}}$ to $\mathcal{F}_i$ can be found by the simple relationship.

$$\mathbf{R}_{i,J_{2k}} = \mathbf{R}^{-1}_{J_{2k},i} = \mathbf{R}^{T}_{J_{2k},i} \tag{3.16}$$

and

$$\dot{\mathbf{R}}_{if} = \dot{\mathbf{R}}^{-1}_{J_{2k},i} = \dot{\mathbf{R}}^{T}_{J_{2k},i} \tag{3.17}$$

3.2 Deriving $\mathbf{R}_{J_{2k},i}$ and $\dot{\mathbf{R}}_{J_{2k},i}$ for Various Coordinate Systems

In GMAT, there are numerous coordinate systems that can be used to define variables, stopping conditions, spacecraft states, and other quantities. Some examples include the Earth centered mean ecliptic of J2000 system, the Earth fixed system, the Mars equator system, and the Earth-Moon rotating system.

In next few subsections, we determine how $\mathbf{R}_{J_{2k},i}$ and $\dot{\mathbf{R}}_{J_{2k},i}$ are calculated in GMAT for all of the coordinate systems available in GMAT. Let's begin by looking at coordinate systems defined by the equator of a celestial body.

3.2.1 Determining $\mathbf{R}_{J_{2k},i}$ and $\dot{\mathbf{R}}_{J_{2k},i}$ for the Equator coordinate system

The equator coordinate system for any celestial body is referenced to the body's equator and the vernal equinox. The Earth has several equatorial systems which are discussed in other sections. In this section, we'll look at the equator coordinate systems for the other 8 planets and the Earth's Moon.

The equatorial systems of the planets and moons are described with respect to the $\mathcal{F}_{J_{2k}}$ system. The IAU has published data that gives the spin axis direction, with respect to $\mathcal{F}_{J_{2k}}$, of all the planets and major moons as a function of time. The IAU data also gives the rotation of the prime meridian with respect to $\mathcal{F}_{J_{2k}}$. This data for all of the planets and many moons can be found in "Report of the IAU/IAG Working Group on Cartographic Coordinates and Rotational Elements of the Planets and Satellites: 2000" by Seidelmann[?] *et.al.* Figure 1 in this document explains the three variables, α_o, δ_o, and W, that are used to define the body spin axis and prime meridian location w/r/t MJ2000Eq. The values of α_o, δ_o, and W for the nine planets and the Earth's moon are found on pgs. 8 and 9 of the Reference.

Using the notation in the Reference, we can write

$$\mathbf{R}_{J_{2k},i} = \mathbf{R}_3^T(90^\circ + \alpha_o)\mathbf{R}_1^T(90^\circ - \delta_o) \tag{3.18}$$

We can assume the derivative of $\mathbf{R}_{Ii}$ for the Equator system is the zero matrix.

$$\dot{\mathbf{R}}_{J_{2k},i} = \begin{pmatrix} 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 \end{pmatrix} \tag{3.19}$$

3.2.2 Determining $\mathbf{R}_{J_{2k},i}$ and $\dot{\mathbf{R}}_{J_{2k},i}$ for the MJ2000Ec coordinate system

The matrix to rotate from MJ2000 Ecliptic (MJ2000Ec) to MJ2000 Equatorial (MJ2000Eq) is given by

$$\mathbf{R} = \begin{pmatrix} 1.0 & 0.0 & 0.0 \\ 0.0 & 0.917482062076895741 & -0.397777155914121383 \\ 0.0 & 0.397777155914121383 & 0.917482062076895741 \end{pmatrix} \tag{3.20}$$

It's time derivative is the zero matrix.

$$\dot{\mathbf{R}} = \begin{pmatrix} 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 \end{pmatrix} \tag{3.21}$$

3.2.3 Determining $\mathbf{R}_{J_{2k},i}$ and $\dot{\mathbf{R}}_{J_{2k},i}$ for the TOEEq coordinate system

$\mathbf{R}_{Ii}$ and $\dot{\mathbf{R}}_{Ii}$ for the TOEEq system can be calculated using the following equations (Vallado Fig. 3-29)

$$\mathbf{R}_{Ii} = \mathbf{P}(t_o)\mathbf{N}(t_o) \tag{3.22}$$

where t_o is the epoch defined in the coordinate system description provided by the user in the epoch field. Hence t_o is a constant value for the TOEEq system. For a given t_o, the matrices associated with the TOEEq system only need to be evaluated once and can be reused later when necessary. $\mathbf{P}(t_o)$ and $\mathbf{N}(t_o)$ are part of the FK5 reduction algorithm and can be found in Vallado[2] pgs. 214 - 219. Because t_o is fixed for a TOEEq system, the derivative of $\mathbf{R}_{Ii}$ is the zero matrix.

$$\dot{\mathbf{R}} = \begin{pmatrix} 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 \end{pmatrix} \tag{3.23}$$

3.2.4 Determining $\mathbf{R}_{J_{2k},i}$ and $\dot{\mathbf{R}}_{J_{2k},i}$ for the MOEEq coordinate system

$\mathbf{R}_{Ii}$ and $\dot{\mathbf{R}}_{Ii}$ for the MOEEq system can be calculated using the following equations

$$\mathbf{R}_{Ii} = \mathbf{P}(t_o) \tag{3.24}$$

where t_o is the epoch defined in the coordinate system description provided by the user in the epoch field. Hence t_o is a constant value for the MOEEq system. For a given t_o, the matrices associated with the MOEEq system only need to be evaluated once and can be reused later when necessary. $\mathbf{P}(t_o)$ is part of the FK5 reduction algorithm and can be found in Vallado[2] pgs. 214 - 215. Because t_o is fixed for a MOEEq system, the derivative of $\mathbf{R}_{Ii}$ is the zero matrix.

$$\dot{\mathbf{R}} = \begin{pmatrix} 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 \end{pmatrix} \tag{3.25}$$

3.2.5 Determining $\mathbf{R}_{J_{2k},i}$ and $\dot{\mathbf{R}}_{J_{2k},i}$ for the TODEq coordinate system

$\mathbf{R}_{Ii}$ and $\dot{\mathbf{R}}_{Ii}$ for the TODEq system can be calculated using the following equations Vallado[2] Fig. 3-29).

$$\mathbf{R}_{Ii} = \mathbf{P}(t_o)\mathbf{N}(t_o) \tag{3.26}$$

where t_o is the epoch. Unlike the TOEEq sytem, for the TODEq system t_o is a variable and usually comes from the epoch of the object whose state we wish to convert. $\mathbf{P}(t_o)$ and $\mathbf{N}(t_o)$ are part of the FK5 reduction algorithm and can be found in Vallado pgs. 214 - 219. Because t_o is not fixed for a TODEq system the derivative of $\mathbf{R}_{Ii}$ is non-zero. However, we will assume its derivative is negligibly small so that

$$\dot{\mathbf{R}} = \begin{pmatrix} 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 \end{pmatrix} \tag{3.27}$$

3.2.6 Determining $\mathbf{R}_{J_{2k},i}$ and $\dot{\mathbf{R}}_{J_{2k},i}$ for the MODEq coordinate system

$\mathbf{R}_{Ii}$ and $\dot{\mathbf{R}}_{Ii}$ for the MODEq system can be calculated using the following equations

$$\mathbf{R}_{Ii} = \mathbf{P}(t_o) \tag{3.28}$$

where t_o is the epoch. Unlike the MOEEq sytem, for the MODEq system t_o is a variable and usually comes from the epoch of the object whose state we wish to convert. $\mathbf{P}(t_o)$ and $\mathbf{N}(t_o)$ are part of the FK5 reduction algorithm and can be found in Vallado[2] pgs. 214 - 219. Because t_o is not fixed for a MODEq system, the derivative of $\mathbf{R}_{Ii}$ is non-zero. However, we will assume its derivative is negligibly small so that

$$\dot{\mathbf{R}} = \begin{pmatrix} 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 \end{pmatrix} \tag{3.29}$$

3.2.7 Determining $\mathbf{R}_{J_{2k},i}$ and $\dot{\mathbf{R}}_{J_{2k},i}$ for the body fixed coordinate system

The body fixed coordinate system is referenced to the body equator and the prime meridian of the body. The body fixed system for Earth is found by using FK5 reduction to the ITRF system as described by Vallado. The ITRF system is the earth fixed system.

Vallado denotes the four rotation sequences required to transform from the ITRF to the FK5 system as [PM], the polar motion, [ST], the sidereal time, [NUT], the nutation, and [PREC], the precession. GMAT calculates these four rotation matrices as described in Vallado. The rotation matrix from ITRF to FK5 can be written as follows.

$$\mathbf{R}_{Ii} = [PREC]^T[NUT]^T[ST]^T[PM]^T \tag{3.30}$$

GMAT assumes that the the only intermediate rotation that has a significant time derivative is the sidereal time, [ST]. So, we can write

$$\dot{\mathbf{R}}_{Ii} = [PREC]^T[NUT]^T[\dot{ST}]^T[PM]^T \tag{3.31}$$

where $[\dot{ST}]$ is given by

$$[\dot{ST}] = \begin{pmatrix} -\omega_e \sin\theta_{AST} & \omega_e \cos\theta_{AST} & 0.0 \\ -\omega_e \cos\theta_{AST} & -\omega_e \sin\theta_{AST} & 0.0 \\ 0.0 & 0.0 & 0.0 \end{pmatrix} \tag{3.32}$$

and ω_e is given by

$$\omega_e = 7.29211514670698e^{-5}\left(1 - \frac{LOD}{86400}\right) \tag{3.33}$$

Note that the 2nd edition of Vallado[2] has inconsistencies in Eqs. (3.30) and (3.31) and they are discussed in the errata to the 2nd edition. We have modified equations Eqs. (3.30) and (3.31) according to the errata.

For bodies other than the earth, the IAU gives the spin axis direction as a function of time with respect to the MJ2000Eq system and rotation of the prime meridian in the MJ2000Eq system. This data for all of the planets and many moons can be found in "Report of the IAU/IAG Working Group on Cartographic Coordinates and Rotational Elements of the Planets and Satellites: 2000" Seidelmann *et.al.* Figure 1 in this document explains the three variables, α_o, δ_o, and W, that are used to define the body spin axis and prime meridian location w/r/t J2000. The values of α_o, δ_o, and W for the nine planets and the Earth's moon are found on pgs. 8 and 9.

Using the notation found in the reference we can write

$$\mathbf{R}_{Ii} = \mathbf{R}_3^T(90^\circ + \alpha_o)\mathbf{R}_1^T(90^\circ - \delta_o)\mathbf{R}_3^T(W) \tag{3.34}$$

For the derivative we assume that

$$\frac{d}{dt}\left(\mathbf{R}_3^T(90^\circ + \alpha_o)\right) = \begin{pmatrix} 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 \end{pmatrix} \tag{3.35}$$

and

$$\frac{d}{dt}\left(\mathbf{R}_1^T(90^\circ - \delta_o)\right) = \begin{pmatrix} 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 \end{pmatrix} \tag{3.36}$$

$$\frac{d}{dt}\left(\mathbf{R}_3^T(W)\right) = \begin{pmatrix} -\dot{W}\sin(W) & -\dot{W}\cos(W) & 0.0 \\ \dot{W}\cos(W) & -\dot{W}\sin(W) & 0.0 \\ 0.0 & 0.0 & 0.0 \end{pmatrix} \tag{3.37}$$

where $\dot{W}$ is the time derivative of W for the given body in the Reference. In summary

$$\dot{\mathbf{R}} = \mathbf{R}_3^T(90^\circ + \alpha_o)\mathbf{R}_1^T(90^\circ - \delta_o)\begin{pmatrix} -\dot{W}\sin(W) & -\dot{W}\cos(W) & 0.0 \\ \dot{W}\cos(W) & -\dot{W}\sin(W) & 0.0 \\ 0.0 & 0.0 & 0.0 \end{pmatrix} \tag{3.38}$$

Seidelmann[1] does not provide the values for $\dot{W}$ so we include the tables here with the derived values for $\dot{W}$.

3.3 Determining $\mathbf{R}_{J_{2k},i}$ and $\dot{\mathbf{R}}_{J_{2k},i}$ for Object Referenced Systems

An object referenced system is a coordinate system whose axes are defined by the motion of one object with respect to another object. GMAT allows the user to define many different types of Object Referenced system. In Fig. 3.3 we see a diagram that defines the directions a user can choose from in creating an Object

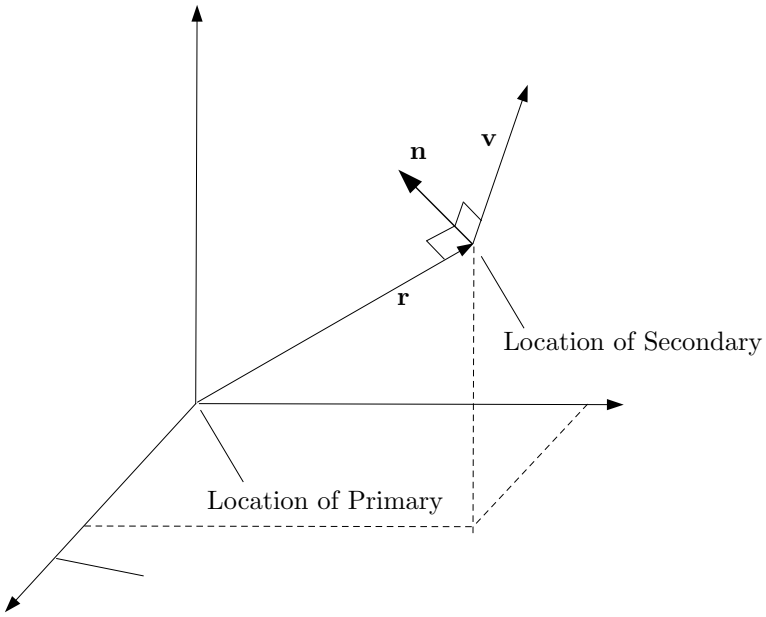

Figure 3.3: Diagram of the Object Referenced System

Referenced coordinate system. There are six directions. One is the relative position, denoted here by $\mathbf{r}$, of the secondary object with respect to the primary object, expressed in an inertial frame. The second is the the relative velocity, denoted here by $\mathbf{v}$, of the secondary object with respect to the primary, expressed in an inertial frame. The third direction is the vector normal to the direction of motion which is denoted by $\mathbf{n}$ and is calculated from $\mathbf{n} = \mathbf{r} \times \mathbf{v}$. The remaining three directions are the negative of of the first three.

In GMAT, a user can use the directions described above to define an Object Referenced coordinate system. In doing so, the user can choose two of the available directions, and define which of the three axes, x, y, and z, they desire the direction to be aligned with. Given this information, GMAT automatically constructs an orthogonal coordinate system. For example, if user chooses the x-axis to be in the direction of $\mathbf{r}$ and the z-axis to be in the direction of $\mathbf{n}$, the GMAT completes the right-handed set by setting the y-axis in the direction of $\mathbf{n} \times \mathbf{r}$. Obviously there are some choices that not yield an orthogonal system, or that yield a left handed system. GMAT does not allow the user to select these pairs of axes and throws an error message.

In general, given the unit vectors that define the axes system of the Object Referenced system, but expressed in the inertial frame, GMAT uses the following equations to determine $\mathbf{R}_{Ii}$ and $\dot{\mathbf{R}}_{Ii}$.

$$\mathbf{R}^{Ii} = \begin{pmatrix} \hat{x}_1 & \hat{y}_1 & \hat{z}_1 \\ \hat{x}_2 & \hat{y}_2 & \hat{z}_2 \\ \hat{x}_3 & \hat{y}_3 & \hat{z}_3 \\ & & \end{pmatrix} \tag{3.39}$$

and

$$\dot{\mathbf{R}}^{Ii} = \begin{pmatrix} \dot{\hat{x}}_1 & \dot{\hat{y}}_1 & \dot{\hat{z}}_1 \\ \dot{\hat{x}}_2 & \dot{\hat{y}}_2 & \dot{\hat{z}}_2 \\ \dot{\hat{x}}_3 & \dot{\hat{y}}_3 & \dot{\hat{z}}_3 \\ & & \end{pmatrix} \tag{3.40}$$

where

$$\hat{\mathbf{x}} = [\hat{x}_1 \quad \hat{x}_2 \quad \hat{x}_3]^T \tag{3.41}$$

$$\dot{\hat{\mathbf{x}}} = \begin{bmatrix} \dot{\hat{x}}_1 & \dot{\hat{x}}_2 & \dot{\hat{x}}_3 \end{bmatrix}^T \tag{3.42}$$

and similarly for $\hat{\mathbf{y}}$, $\dot{\hat{\mathbf{y}}}$, $\hat{\mathbf{z}}$ and $\dot{\hat{\mathbf{z}}}$.

Recall that the user chooses two axes to an Object Referenced system among the following choices: $\hat{\mathbf{r}}$, $\hat{\mathbf{v}}$, $\hat{\mathbf{n}}$, $-\hat{\mathbf{r}}$, $-\hat{\mathbf{v}}$, and $-\hat{\mathbf{n}}$. In general, one of the axes chosen by the user must be either $\hat{\mathbf{n}}$, or $-\hat{\mathbf{n}}$.

If the user defines the x-axis and y-axis then GMAT determines the z axis using

$$\hat{\mathbf{z}} = \hat{\mathbf{x}} \times \hat{\mathbf{y}} \tag{3.43}$$

and

$$\dot{\hat{\mathbf{z}}} = \dot{\hat{\mathbf{x}}} \times \hat{\mathbf{y}} + \hat{\mathbf{x}} \times \dot{\hat{\mathbf{y}}} \tag{3.44}$$

If the user defines the y-axis and z-axis, then GMAT determines the z axis using

$$\hat{\mathbf{x}} = \hat{\mathbf{y}} \times \hat{\mathbf{z}} \tag{3.45}$$

and

$$\dot{\hat{\mathbf{x}}} = \dot{\hat{\mathbf{y}}} \times \hat{\mathbf{z}} + \hat{\mathbf{y}} \times \dot{\hat{\mathbf{z}}} \tag{3.46}$$

And finally, if the user defines the x-axis and z-axis then GMAT determines the z axis using

$$\hat{\mathbf{y}} = \hat{\mathbf{z}} \times \hat{\mathbf{x}} \tag{3.47}$$

and

$$\dot{\hat{\mathbf{y}}} = \dot{\hat{\mathbf{z}}} \times \hat{\mathbf{x}} + \hat{\mathbf{z}} \times \dot{\hat{\mathbf{x}}} \tag{3.48}$$

Depending on the users choice of axes for an Object Referenced coordinate system, GMAT will need to calculate $\hat{\mathbf{r}}$, $\hat{\mathbf{v}}$, $\hat{\mathbf{n}}$, $\dot{\hat{\mathbf{r}}}$, $\dot{\hat{\mathbf{v}}}$, and $\dot{\hat{\mathbf{n}}}$. These are given by:

$$\hat{\mathbf{r}} = \frac{\mathbf{r}}{\|\mathbf{r}\|} = \frac{\mathbf{r}}{r} \tag{3.49}$$

$$\hat{\mathbf{v}} = \frac{\mathbf{v}}{\|\mathbf{v}\|} = \frac{\mathbf{v}}{v} \tag{3.50}$$

$$\hat{\mathbf{n}} = \frac{\mathbf{r} \times \mathbf{v}}{\|\mathbf{r} \times \mathbf{v}\|} \tag{3.51}$$

$$\dot{\hat{\mathbf{r}}} = \frac{\mathbf{v}}{r} - \frac{\hat{\mathbf{r}}}{r}(\hat{\mathbf{r}} \cdot \mathbf{v}) \tag{3.52}$$

$$\dot{\hat{\mathbf{v}}} = \frac{\mathbf{a}}{v} - \frac{\hat{\mathbf{v}}}{v}(\hat{\mathbf{v}} \cdot \mathbf{a}) \tag{3.53}$$

$$\dot{\hat{\mathbf{n}}} = \frac{\mathbf{r} \times \mathbf{a}}{n} - \frac{\hat{\mathbf{n}}}{n}(\mathbf{r} \times \mathbf{a} \cdot \hat{\mathbf{n}}) \tag{3.54}$$

The derivations of the above quantities are shown below. We start by deriving two derivatives with respect to $\mathbf{n}$, where $\mathbf{n}$ is given by:

$$\mathbf{n} = \mathbf{r} \times \mathbf{v} \tag{3.55}$$

We need to determine two derivatives of $\mathbf{n}$. First

$$\frac{d\mathbf{n}}{dt} = \frac{d}{dt}(\mathbf{r} \times \mathbf{v}) = \underbrace{\frac{d\mathbf{r}}{dt} \times \mathbf{v}}_{0} + \mathbf{r} \times \frac{d\mathbf{v}}{\partial t} \tag{3.56}$$

so we know that

$$\boxed{\frac{d\mathbf{n}}{dt} = \mathbf{r} \times \mathbf{a}} \tag{3.57}$$

The next useful derivative is

$$\frac{dn}{dt} = \frac{d\|\mathbf{n}\|}{dt} = \frac{d}{dt}\left(\mathbf{n}^T\mathbf{n}\right)^{1/2} = \frac{\mathbf{n}^T}{n}\frac{d\mathbf{n}}{dt} \tag{3.58}$$

So we can write

$$\boxed{\frac{dn}{dt} = \frac{\mathbf{n}}{n}\cdot(\mathbf{r}\times\mathbf{a})} \tag{3.59}$$

The following two derivatives are also useful

$$\frac{dr}{dt} = \frac{d\|\mathbf{r}\|}{dt} = \frac{d}{dt}(\mathbf{r}^T\mathbf{r})^{1/2} = \frac{\mathbf{v}\cdot\mathbf{r}}{r} \tag{3.60}$$

$$\boxed{\frac{dr}{dt} = \frac{\mathbf{v}\cdot\mathbf{r}}{r}} \tag{3.61}$$

$$\frac{dv}{dt} = \frac{d\|\mathbf{v}\|}{dt} = \frac{d}{dt}(\mathbf{v}^T\mathbf{v})^{1/2} = \frac{\mathbf{v}\cdot\mathbf{a}}{v} \tag{3.62}$$

so we can write

$$\boxed{\frac{dv}{dt} = \frac{\mathbf{v}\cdot\mathbf{a}}{v}} \tag{3.63}$$

$$\boxed{\hat{\mathbf{v}} = \frac{\mathbf{v}}{\|\mathbf{v}\|}} \tag{3.64}$$

$$\boxed{\hat{\mathbf{r}} = \frac{\mathbf{r}}{\|\mathbf{r}\|}} \tag{3.65}$$

$$\boxed{\hat{\mathbf{n}} = \frac{\mathbf{r}\times\mathbf{v}}{\|\mathbf{r}\times\mathbf{v}\|}} \tag{3.66}$$

The time derivatives are derived as follows.

$$\dot{\hat{\mathbf{r}}} = \frac{\partial\hat{\mathbf{r}}}{\partial t} = \frac{\partial}{\partial t}\left(\mathbf{r}r^{-1}\right) = \frac{\mathbf{v}}{r} - \frac{\mathbf{r}}{r^2}\left(\frac{\mathbf{r}\cdot\mathbf{v}}{r}\right) \tag{3.67}$$

which can be rewritten as

$$\boxed{\frac{d\hat{\mathbf{r}}}{dt} = \frac{\mathbf{v}}{r} - \frac{\hat{\mathbf{r}}}{r}\left(\hat{\mathbf{r}}\cdot\mathbf{v}\right)} \tag{3.68}$$

$$\dot{\hat{\mathbf{v}}} = \frac{\partial\hat{\mathbf{v}}}{\partial t} = \frac{\partial}{\partial t}\left(\mathbf{v}v^{-1}\right) = \frac{\mathbf{a}}{v} - \frac{\mathbf{v}}{v^2}\left(\frac{\mathbf{v}\cdot\mathbf{a}}{v}\right) \tag{3.69}$$

which can be rewritten as

$$\boxed{\dot{\hat{\mathbf{v}}} = \frac{\mathbf{a}}{v} - \frac{\hat{\mathbf{v}}}{v}\left(\hat{\mathbf{v}}\cdot\mathbf{a}\right)} \tag{3.70}$$

Finally,

$$\dot{\hat{\mathbf{n}}} = \frac{d}{dt}\left(\mathbf{n}n^{-1}\right) = \frac{\mathbf{r}\times\mathbf{a}}{n} - \frac{\mathbf{n}}{n^3}\left(\mathbf{r}\times\mathbf{a}\cdot\mathbf{n}\right) \tag{3.71}$$

$$\boxed{\dot{\hat{\mathbf{n}}} = \frac{\mathbf{r}\times\mathbf{a}}{n} - \frac{\hat{\mathbf{n}}}{n}\left(\mathbf{r}\times\mathbf{a}\cdot\hat{\mathbf{n}}\right)} \tag{3.72}$$

Chapter 4

Calculation Objects

GMAT has the ability to calculate numerous quantities that are dependent the states of objects, coordinate systems, and the mission sequence. These calculation objects can range from the spacecraft state, to the local atmospheric density, to the positions of celestial bodies with respect to spacecraft, or other celestial bodies. In this section, we present how GMAT performs these calculations by showing the mathematical algorithms.

4.1 Spacecraft State Representations

There are several state representations that can be used in GMAT to define the state of a spacecraft object. These include the Keplerian elements, Cartesian state, Equinoctial elements, Spherical Elements, and the modified Keplerian elements. In the following few subsections we discuss the definitions of these states types, and show how GMAT converts between the different state representations.

4.1.1 Definitions

The Keplerian Elements are one of the most commonly used state representations. They provide a way to define the spacecraft state in way that provides an intuitive understanding of the motion of the spacecraft in orbit. The Keplerian elements are denoted a, e, i, ω, Ω, and ν. They are defined in detail in Table 4.1 and illustrated in Fig. 19.1. Sections 4.1.2 and 4.1.3 show the algorithm that GMAT uses to convert from Keplerian elements to the cartesian state respectively.

The cartesian state is another common state representation and is often used in the numerical integration of the equations of motion. The cartesian state with respect to a given coordinate system is described in detail in Table 4.2.

The equinoctial elements are a set of non-singular elements that can be used to describe the state of a spacecraft. Because they are nonsingular, they are useful for in expressing the equations of motion in Variation of Parameters (VOP) form. The elements can be unintuitive to use however. The equinoctial elements are described in detail in Table 4.3.

The modified Keplerian Elements are similar to the Keplerian elements except a and e are replaced with the radius of periapsis r_p, and the radius of apoapsis r_a. r_p and r_a are often more convenient and intuitive for describing the dimensions of a Keplerian orbit than a and e. The modified Keplerian Elements are defined in detail in Table 4.5.

Table 4.1: The Keplerian Elements

Symbol	Name	Description
a	semimajor axis	The semimajor contains information on the type and size of an orbit. If $a > 0$ the orbit is elliptic. If $a < 0$ the orbit is hyperbolic.
e	eccentricity	The eccentricity contains information on the shape of an orbit. If $e = 0$, then the orbit is circular. If $0 < e < 1$ the orbit is elliptical. If $e = 1$ the orbit is parabolic. If $e > 1$ then the orbit is hyperbolic.
i	inclination	The inclination is the angle between the $\hat{\mathbf{z}}_I$ axis and the orbit normal direction $\mathbf{h}$. If $i \leq 90°$ then the orbit is prograde. If $i > 90°$ then the orbit is retrograde.
ω	argument of periapsis	The argument of periapsis is the angle between a vector pointing at periapsis, $\mathbf{x}_p$, and a vector pointing at the spacecraft. The argument of periapsis is undefined for circular orbits.
Ω	right ascension of the ascending node	Ω is defined as the angle between $\hat{\mathbf{x}}_I$ and $\mathbf{N}$ measured counterclockwise. $\mathbf{N}$ is defined as the vector pointing from the center of the central body to the spacecraft, when the spacecraft crosses the bodies equatorial plane from the southern to the northern hemisphere. Ω is undefined for equatorial orbits.
ν	true anomaly	The true anomaly is defined as the angle between a vector pointing at periapsis, $\mathbf{x}_p$, and a vector pointing at the spacecraft. The true anomaly is undefined for circular orbits.

Table 4.2: The Cartesian State

Symbol	Description
x	x-component of position
y	y-component of position
z	z-component of position
$\dot{x}$	x-component of velocity
$\dot{y}$	y-component of velocity
$\dot{z}$	z-component of velocity

Table 4.3: The Equinoctial Elements

Symbol	Description
a	The semimajor contains information on the type and size of an orbit. If $a > 0$ the orbit is elliptic. If $a < 0$ the orbit is hyperbolic.
h	The projection of the eccentricity vector onto the $\hat{\mathbf{y}}_{ep}$ axis.
k	The projection of the eccentricity vector onto the $\hat{\mathbf{x}}_{ep}$ axis.
p	The projection of $\mathbf{N}$ onto the $\hat{\mathbf{y}}_{ep}$ axis.
q	The projection of $\mathbf{N}$ onto the $\hat{\mathbf{x}}_{ep}$ axis.
λ	The mean longitude.

Table 4.4: The Spherical Elements

Symbol	Name	Description
r	r	Magnitude of the position vector, $\|\mathbf{r}\|$
λ	Right Ascension	The between the projection of $\mathbf{r}$ into the $x-y$ and the x-axis. Measured counterclockwise.
δ	Declination	The angle between $\mathbf{r}$ and the $x-y$ plane measured in the plane formed by $\mathbf{r}$ and $\hat{\mathbf{z}}$.
v	v	Magnitude of the velocity vector, $\|\mathbf{v}\|$
ψ	Flight path angle	The angle measured from a plane normal to $\mathbf{r}$ to the velocity vector $\mathbf{v}$, measured in the plane formed by $\mathbf{r}$ and $\mathbf{v}$
α_f	Flight path azimuth	The angle measured from vector perpendicular $\mathbf{r}$ and pointing north, to the projection of $\mathbf{v}$ into a plane normal to $\mathbf{r}$.

Table 4.5: The Modified Keplerian Elements

Symbol	Name	Description
r_p	radius of periapsis	The radius of periapsis is the radius at the spacecrafts closest approach to the central body. The radius of periapsis must be greater than zero, parabolic orbits are not currently supported.
r_a	radius of apoapsis	For an elliptic orbit r_a is the radius at the spacecrafts farthest distance from the central body and $r_a > r_p$. For hyperbolic orbits, $r_a < r_p$ and $r_a < 0$
i	inclination	The inclination is the angle between the $\hat{\mathbf{z}}_I$ axis and the orbit normal direction $\mathbf{h}$. If $i \leq 90^\circ$ then the orbit is prograde. If $i > 90^\circ$ then the orbit is retrograde.
ω	argument of periapsis	The argument of periapsis is the angle between a vector pointing at periapsis, $\mathbf{x}_p$, and a vector pointing at the spacecraft. The argument of periapsis is undefined for circular orbits.
Ω	right ascension of the ascending node	Ω is defined as the angle between $\hat{\mathbf{x}}_I$ and $\mathbf{N}$ measured counterclockwise. $\mathbf{N}$ is defined as the vector pointing from the center of the central body to the spacecraft, when the spacecraft crosses the bodies equatorial plane from the southern to the northern hemisphere. Ω is undefined for equatorial orbits.
ν	true anomaly	The true anomaly is defined as the angle between a vector pointing at periapsis, $\mathbf{x}_p$, and a vector pointing at the spacecraft. The true anomaly is undefined for circular orbits.

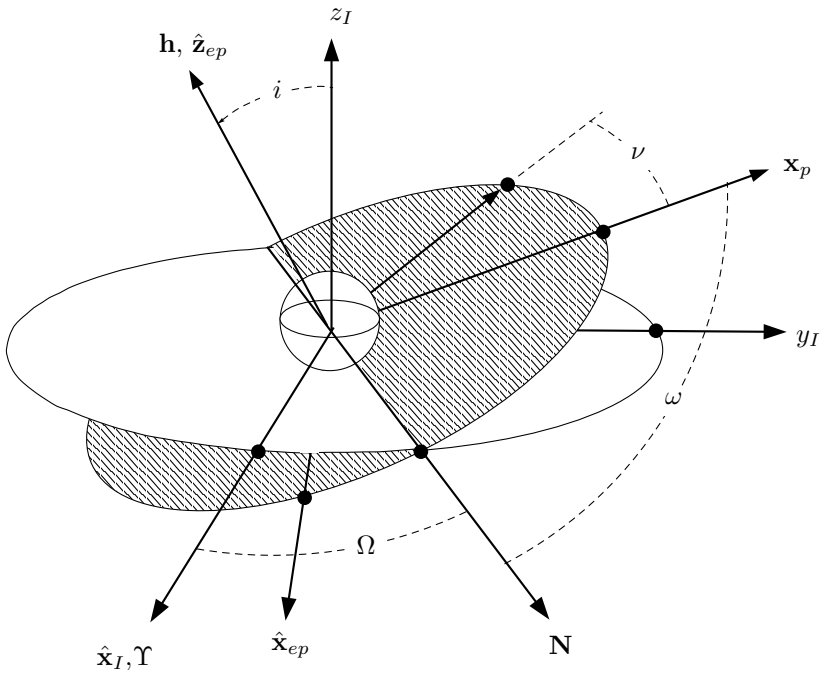

Figure 4.1: The Keplerian Elements

4.1.2 Cartesian State to Keplerian State

Given: $\mathbf{r}$, $\mathbf{v}$, and μ
Find: a, e, i, ω, Ω, and ν

First calculate the specific angular momentum and its magnitude.

$$\mathbf{h} = \mathbf{r} \times \mathbf{v} \tag{4.1}$$

$$h = \|\mathbf{h}\| \tag{4.2}$$

$$\hat{\mathbf{h}} = \frac{\mathbf{h}}{h} \tag{4.3}$$

Next, calculate the orbital parameter, p, the radius magnitude, r, and the speed, v.

$$p = \frac{h^2}{\mu} \tag{4.4}$$

$$r = \|\mathbf{r}\| \tag{4.5}$$

$$v = \|\mathbf{v}\| \tag{4.6}$$

Using the above quantities, the semimajor axis, eccentricity, and inclination are calculated using the following three equations:

$$a = \frac{r}{2 - \dfrac{rv^2}{\mu}}; \tag{4.7}$$

$$e = \sqrt{\left|1 - \frac{p}{a}\right|} \tag{4.8}$$

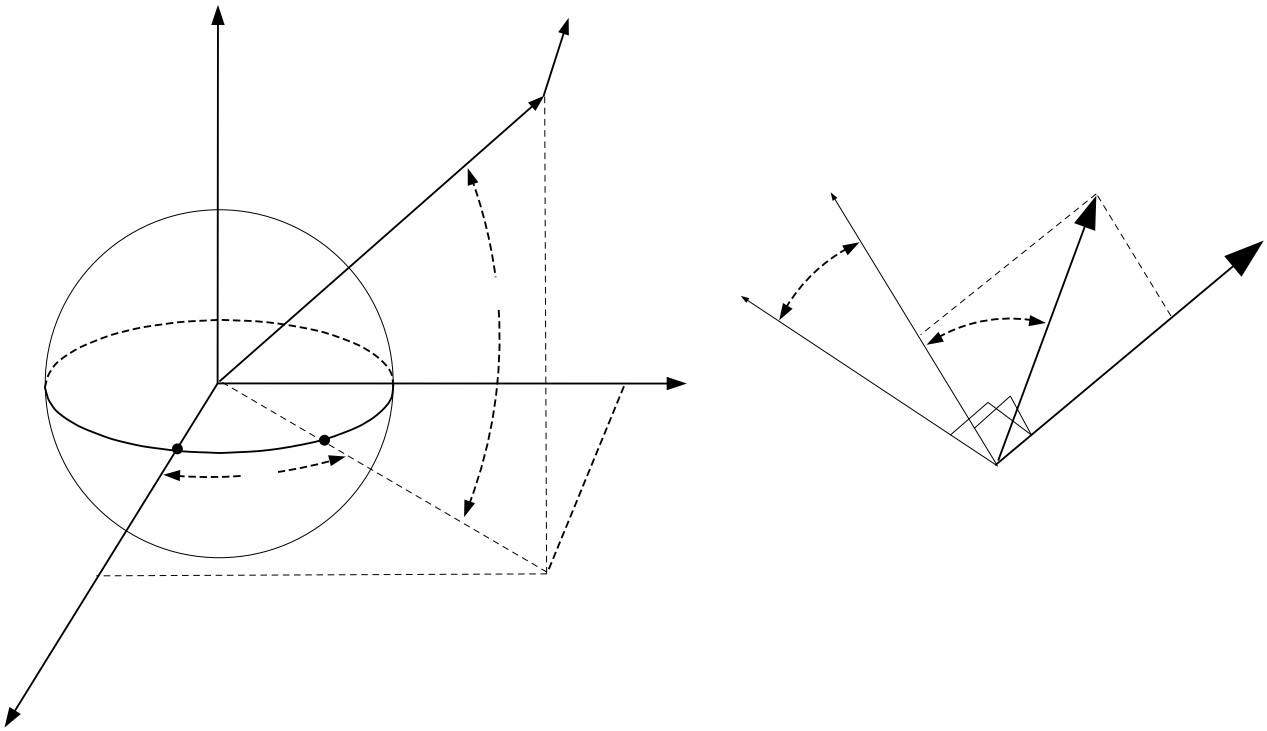

Figure 4.2: The Spherical Elements

$$i = \cos^{-1}(\hat{h}_z) \tag{4.9}$$

There are three special cases that are considered when calculating the true anomaly. The first is for elliptic orbits when $e \geq 1e-6$ GMAT calculates $\cos\nu$ and $\sin\nu$ using the following two equations

$$\cos\nu = \frac{p-r}{er} \tag{4.10}$$

$$\sin\nu = \frac{\mathbf{r}\cdot\mathbf{v}}{\mu e r} \tag{4.11}$$

The next two special cases are for circular orbits. For both cases, when $e < 1e-6$, GMAT sets the eccentricity to be identically zero.

$$e = 0.0 \quad \text{if} \quad e < 1e-6 \tag{4.12}$$

The second special case is the circular inclined case. In this case $e < 1e-6$ and $(i \geq 1e-6)$. For the circular inclined case $\cos\nu$ and $\sin\nu$ are calculated as described below.

$$\mathbf{N} = \begin{bmatrix} -\hat{h}_z\hat{h}_x & -\hat{h}_z\hat{h}_y & \hat{h}_x^2+\hat{h}_y^2 \end{bmatrix}^T \tag{4.13}$$

where $\mathbf{N}$ is a unit vector pointing from the origin to the ascending node.

$$\cos\nu = \frac{-\hat{h}_z\hat{h}_x x - \hat{h}_z\hat{h}_y y}{r\left(\hat{h}_x^2+\hat{h}_y^2\right)} \tag{4.14}$$

$$\sin\nu = \frac{-\hat{h}_z\hat{h}_x x - \hat{h}_z\hat{h}_y y + \left(\hat{h}_x^2+\hat{h}_y^2\right) z}{rN} \tag{4.15}$$

The third special case for calculating ν is the the circular equatorial case. This occurs when $e < 1e-6$ and $(i < 1e-6)$

$$\cos\nu = \frac{x}{r}; \tag{4.16}$$

$$\sin\nu = \frac{y}{r}; \tag{4.17}$$

Finally, knowing $\cos\nu$ and $\sin\nu$ we can solve for ν using

$$\nu = \text{atan2}(\sin\nu, \cos\nu) \tag{4.18}$$

If ν is negative it is adjusted to fall between 0° and 360°.

Now GMAT calculates Ω. If $\mid i \mid < 1e-6$ then GMAT sets Ω to be identically zero.

$$\Omega = 0.0 \quad \text{if} \quad \mid i \mid < 1e-6 \tag{4.19}$$

Otherwise, Ω is calculated using

$$\Omega = \text{atan2}(\hat{h}_x, -\hat{h}_y) \tag{4.20}$$

If Ω is negative it is adjusted to fall between 0° and 360°.

The last of the orbital elements to be calculated by GMAT is ω. There are several special cases. For near circular, or circular orbits when $e < 1e-6$, ω is set to be identically zero.

$$\omega = 0.0 \quad \text{if} \quad e < 1e-6 \tag{4.21}$$

The next two special cases are for elliptical orbits. Both require $\mathbf{X}$ which is defined as

$$\mathbf{X} = \left(v^2 - \frac{\mu}{r}\right)\mathbf{r} - (\mathbf{r}\cdot\mathbf{v})\,\mathbf{v} \tag{4.22}$$

For elliptical inclined orbits, when $e \geq 1e-6$ and $\mid i \mid \geq 1e-6$, ω is calculated using

$$\omega = \cos^{-1}\frac{\mathbf{N}\cdot\mathbf{X}}{NX} \tag{4.23}$$

If $e \geq 1e-6$ and $\mid i \mid < 1e-6$ the ω is calculated as using

$$\omega = \text{atan2}(X_1, X_0) \tag{4.24}$$

If ω is negative it is adjusted to fall between 0° and 360°.

Comments: The denominators in the following equations are checked before performing division: (4.7), (4.8), (4.10), (4.11) and (4.22). If a denominator is smaller than 10^{-30}, then the division is not performed and the function returns an error message.

4.1.3 Keplerian State to Cartesian State

if $i < 0$ then i = i+180

if $i <$ 1e-6 then $\Omega = 0$

if $e <$ 1e-6 then $\omega = 0$

$$p = a(1 - e^2); \tag{4.25}$$

if $(1 + e\cos\nu < 1\text{e-}30)$ then error and return, else

$$r = \frac{p}{1 + e\cos\nu} \tag{4.26}$$

$$x = r\left(\cos\left(\omega + \nu\right)\cos\Omega - \cos i\sin\left(\omega + \nu\right)\sin\Omega\right) \tag{4.27}$$

$$y = r\left(\cos\left(\omega + \nu\right)\sin\Omega + \cos i\sin\left(\omega + \nu\right)\cos\Omega\right) \tag{4.28}$$

$$z = r\left(\sin\left(\omega + \nu\right)\sin i\right) \tag{4.29}$$

if $(\|p\| < 1e - 30)$ then error and return, else

$$\dot{x} = \sqrt{\frac{\mu}{p}}\left[\left(\cos\nu + e\right)\left(-\sin\omega\cos\Omega - \cos i\sin\Omega\cos\omega\right) - \sin\nu\left(\cos\omega\cos\Omega - \cos i\sin\Omega\sin\omega\right)\right] \tag{4.30}$$

$$\dot{y} = \sqrt{\frac{\mu}{p}}\left[\left(\cos\nu + e\right)\left(-\sin\omega\sin\Omega + \cos i\cos\Omega\cos\omega\right) - \sin\nu\left(\cos\omega\sin\Omega + \cos i\cos\Omega\sin\omega\right)\right] \tag{4.31}$$

$$\dot{z} = \sqrt{\frac{\mu}{p}}\left[\left(\cos\nu + e\right)\sin i\cos w - \sin\nu\sin i\sin\omega\right] \tag{4.32}$$

4.1.4 Equinoctial to Cartesian

The equinoctial elements used in GMAT are defined as follows

a = semimajor axis

h = projection of the eccentricity vector **e**

$$\lambda = F + h\cos F - k\sin F \tag{4.33}$$

$$\beta = \frac{1}{a + \sqrt{1 - h^2 - k^2}} \tag{4.34}$$

$$X_1 = a\left[\left(1 - h^2\beta\right)\cos F + hk\beta\sin F - k\right] \tag{4.35}$$

$$Y_1 = a\left[\left(1 - k^2\beta\right)\sin F + hk\beta\cos F - k\right] \tag{4.36}$$

$$\dot{X}_1 = \frac{na^2}{r}\left[hk\beta\cos F - \left(1 - h^2\beta\right)\sin F\right] \tag{4.37}$$

$$\dot{Y}_1 = \frac{na^2}{r}\left[\left(1 - k^2\beta\right)\cos F - hk\beta\sin F\right] \tag{4.38}$$

The transformation from the equinoctial system to the inertial Cartesian system is given by

$$\mathbf{r} = X_1\hat{\mathbf{f}} + Y_1\hat{\mathbf{g}} \tag{4.39}$$

$$\mathbf{v} = \dot{X}_1\hat{\mathbf{f}} + \dot{Y}_1\hat{\mathbf{g}} \tag{4.40}$$

where

$$\left[\hat{\mathbf{f}} \quad \hat{\mathbf{g}} \quad \hat{\mathbf{w}}\right] = \frac{1}{1+p^2+q^2}\begin{pmatrix} 1-p^2+q^2 & 2pqj & 2p \\ 2pq & \left(1+p^2-q^2\right) & -2q \\ -2pj & 2q & \left(1-p^2-q^2\right)j \end{pmatrix} \tag{4.41}$$

where

j = 1 for direct orbits ($0 \leq i \leq 90^\circ$)
j = -1 for retrograde orbits ($90 < i \leq 180^\circ$)

4.1.5 Cartesian to Equinoctial

$$a = \frac{1}{\frac{2}{r} - \frac{v^2}{\mu}} \tag{4.42}$$

$$\mathbf{e} = -\frac{\mathbf{v}}{r} - \frac{(\mathbf{r}\times\mathbf{v})\times\mathbf{v}}{\mu} \tag{4.43}$$

$$\hat{\mathbf{h}} = \frac{\mathbf{r}\times\mathbf{v}}{\|\mathbf{r}\times\mathbf{v}\|} \tag{4.44}$$

$$f_x = 1 - \frac{w_x^2}{1+w_z^j} \tag{4.45}$$

$$f_y = -\frac{w_x w_y}{1+w_z^j} \tag{4.46}$$

$$f_z = -w_x^j \tag{4.47}$$

where j is described in Section 4.1.4.

$$\hat{\mathbf{g}} = \hat{\mathbf{w}}\times\hat{\mathbf{f}} \tag{4.48}$$

the elements h, k, p, and q are given by

$$h = \mathbf{e}\cdot\hat{\mathbf{g}} \tag{4.49}$$

$$k = \mathbf{e}\cdot\hat{\mathbf{f}} \tag{4.50}$$

$$p = \frac{w_x}{1+w_z^j} \tag{4.51}$$

$$q = -\frac{w_y}{1+w_z^j} \tag{4.52}$$

The mean longitude, λ, is computed using the generalized Kepler equation

$$\lambda = F + h\cos f - k\sin F \tag{4.53}$$

where

$$F = \tan^{-1}\left(\frac{\sin F}{\cos F}\right) \tag{4.54}$$

with

$$\cos F = k + \frac{\left(1-k^2\beta\right)X_1 - hk\beta Y_1}{a\sqrt{1-h^2-k^2}} \tag{4.55}$$

$$\sin F = h + \frac{\left(1 - h^2\beta\right) Y_1 - hk\beta X_1}{a\sqrt{1 - h^2 - k^2}} \tag{4.56}$$

The parameter β is given by Eq. 4.34. Finally, the position coordinates x_{ep} and y_{ep} relative to the equinoctial coordinate system are given by

$$X_1 = \mathbf{r} \cdot \hat{\mathbf{f}} \tag{4.57}$$

$$Y_1 = \mathbf{r} \cdot \hat{\mathbf{g}} \tag{4.58}$$

4.2 Cartesian to Spherical

4.3 Spherical to Cartesian

4.4 Keplerian to Modified Keplerian

Given: a, e, i, ω, Ω, and ν

Find: r_p and r_a

$$r_p = a(1 - e); \tag{4.59}$$

$$r_a = a(1 + e); \tag{4.60}$$

4.5 Modified Keplerian to Keplerian

Given: r_p, r_a, i, ω, Ω, and ν

Find: a and e

$$e = \frac{1 - \frac{r_p}{r_a}}{1 + \frac{r_p}{r_a}} \tag{4.61}$$

$$a = \frac{r_p}{1 - e} \tag{4.62}$$

4.6 RadApo

Given: a, and e

Find: r_a

$$r_a = a(1 + e) \tag{4.63}$$

Comment: a and e are calculated from the satellite cartesian state as shown in Section 4.1.2.

4.7 RadPer

Given: a, and e

Find: r_p

$$r_p = a(1 - e) \tag{4.64}$$

Comment: a and e are calculated from the satellite cartesian state as shown in Section 4.1.2.

4.8 VelApoapsis

Given: a, e, and μ

Find: v_a

$$v_a = \sqrt{2\left(-\frac{\mu}{2a} + \frac{\mu}{a(1+e)}\right)} \tag{4.65}$$

Comment: a and e are calculated from the satellite cartesian state as shown in Section 4.1.2, and μ is associated with the specified central body.

4.9 VelPeriapsis

Given: a, e, and μ

Find: v_p

$$v_p = \sqrt{2\left(-\frac{\mu}{2a} + \frac{\mu}{a(1-e)}\right)} \tag{4.66}$$

Comment: a and e are calculated from the satellite cartesian state as shown in Section 4.1.2, and μ is associated with the specified central body.

4.10 Orbit Period

Given: a, and μ

Find: T

$$T = \sqrt{\frac{a^3}{\mu}} \tag{4.67}$$

Comment: a is calculated from the satellite cartesian state as shown in Section 4.1.2, and μ is associated with the specified central body.

4.11 C3Energy

Given: a, and μ

Find: C_3

$$C_3 = -\frac{\mu}{a} \tag{4.68}$$

Comment: a is calculated from the satellite cartesian state as shown in Section 4.1.2, and μ is associated with the specified central body.

4.12 Energy

Given: a, and μ

Find: $\mathcal{E}$

$$\mathcal{E} = -\frac{\mu}{2a} \tag{4.69}$$

Comment: a is calculated from the satellite cartesian state as shown in Section 4.1.2, and μ is associated with the specified central body.

4.13 Mean Motion

Given: a, e, and μ

Find: n

The orbit is considered elliptic if $e > 1 + 1e^{-10}$. In this case the mean motion, n, is calculated using

$$n = \sqrt{\frac{\mu}{a^3}} \tag{4.70}$$

The orbit is considered hyperbolic if $e > 1 - 1e^{-10}$. In this case the mean motion, n, is calculated using

$$n = \sqrt{-\frac{\mu}{a^3}} \tag{4.71}$$

If neither of the above two conditions are met, the mean motion is calculated using

$$n = 2\sqrt{\mu} \tag{4.72}$$

Comment: a and e are calculated from the satellite cartesian state as shown in Section 4.1.2, and μ is associated with the specified central body.

4.14 Semilatus Rectum

4.15 Time Calculations

4.15.1 Elapsed Days

4.15.2 Hour Angle

4.16 Libration Points

We begin by assuming that the planets move in circular orbits about the sun, and the mass of a spacecraft is negligible compared to the mass of the planets. For illustrative purposes, lets consider the Earth and its orbit about the Sun. In this case, the libration points are locations in space where a spacecraft will stay fixed with respect to the Earth and Sun. Figure 4.3 shows a simple illustration. We see the Sun, the Earth's position with respect to the Sun, and the Libration points L_1 and L_2 at two different epochs. Notice that at t_1, the points L_1 and L_2 are on the Earth-Sun line. At a later time, t_2, although the Earth has moved with respect to the sun, L_1 and L_2 still lie on the Earth-Sun line.

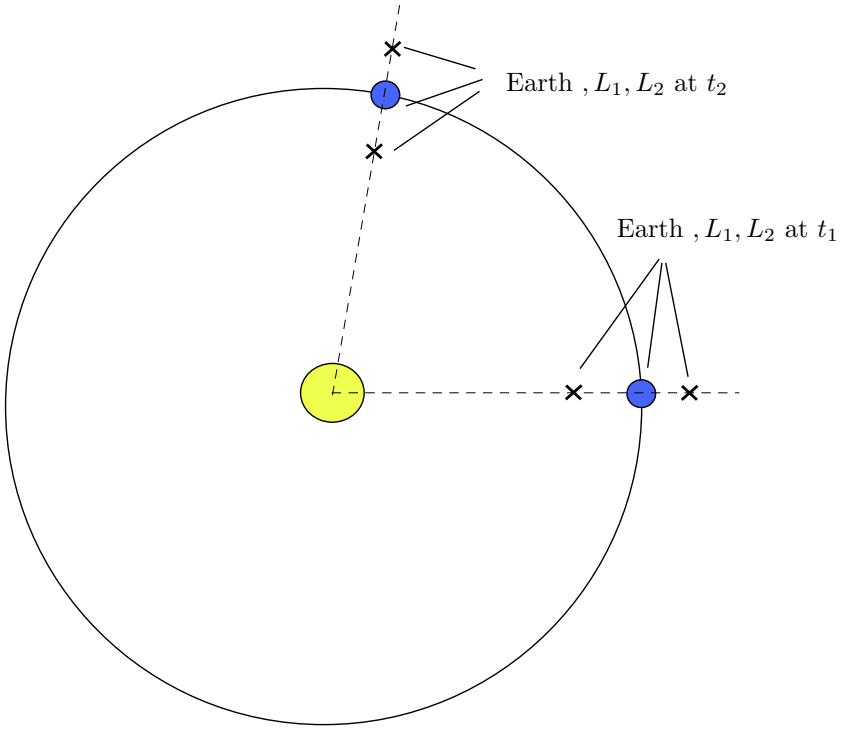

Figure 4.3: Description of Libration Points

The preceding example gives a brief qualitative description of two of the Earth-Sun libration points. In general, there are five libration points for a given three body system. To determine the locations of the libration points, it is convenient to work in a rotating coordinate system rather than the inertial system shown in Fig. 4.3. The system we use is constructed as follows:

- Define the primary as the heavier of the two bodies, the secondary as the lighter.
- Define the coordinate system x-axis as the axis pointing from the primary to the secondary.
- Define the y-axis to be orthogonal to the x-axis in the plane of the secondary's motion about the primary, pointing in the direction the secondary moves about the primary.
- Define the z-axis orthogonal to the x and y axes to form a right-handed system.
- Place the origin at center-of-mass of the system.

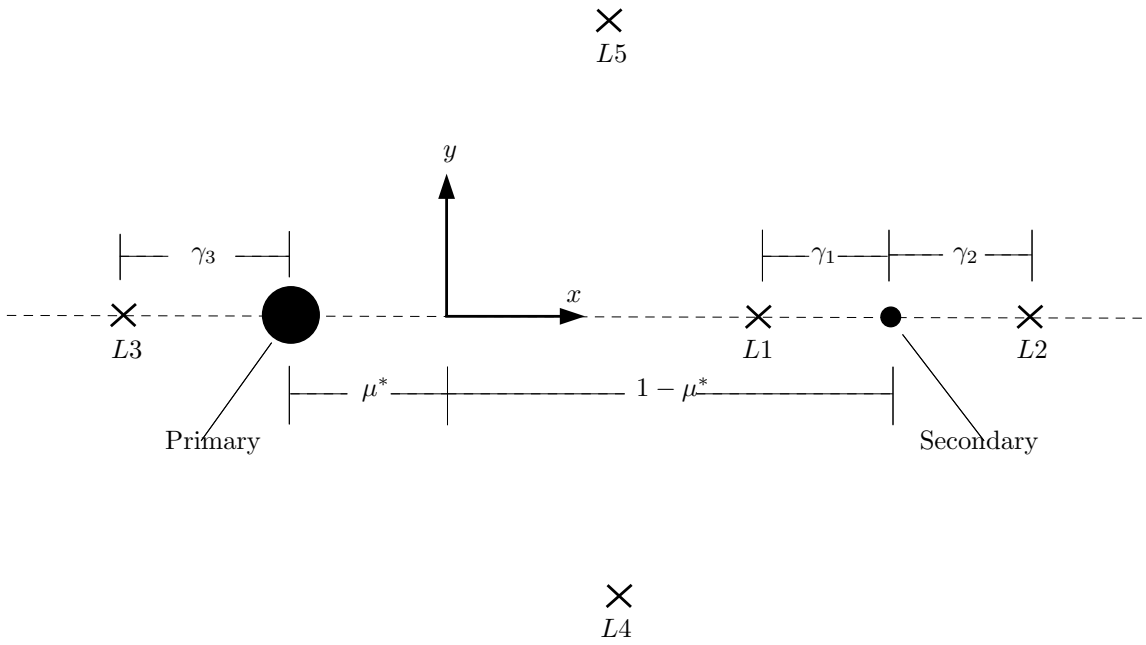

Figure 4.4: Definition of Libration Points

This coordinates system is illustrated in Fig. 4.4. The locations of the libration points in the rotating coordinate system can be found by calculating the values of γ that solve the following equations:

$$\gamma_1^5 - (3-\mu^*)\,\gamma_1^4 + (3-2\mu^*)\,\gamma_1^3 - \mu^*\gamma_1^2 + 2\mu^*\gamma_1 - \mu^* = 0 \quad \text{(For L1)} \tag{4.73}$$

$$\gamma_2^5 - (3-\mu^*)\,\gamma_2^4 + (3-2\mu^*)\,\gamma_2^3 - \mu^*\gamma_2^2 - 2\mu^*\gamma_2 - \mu^* = 0 \quad \text{(For L2)} \tag{4.74}$$

$$\gamma_3^5 + (2+\mu^*)\,\gamma_3^4 + (1+2\mu^*)\,\gamma_3^3 - (1-\mu^*)\,\gamma_3^2 - 2\,(1-\mu^*)\,\gamma_3 - (1-\mu^*) = 0 \quad \text{(For L3)} \tag{4.75}$$

where

$$\mu^* = \frac{m_2}{m_1 + m_2} \tag{4.76}$$

Equations (4.73)-(4.75) do not have exact analytic solutions. Szebehely[5] presents series solutions for γ_1 and γ_2, and γ_3. The expansions for γ_1 and γ_2 are:

$$\gamma_1 = \left(\frac{\mu}{3}\right)^{1/3} - \frac{1}{3}\left(\frac{\mu}{3}\right)^{2/3} - \frac{1}{9}\left(\frac{\mu}{3}\right)^{1} - \frac{23}{81}\left(\frac{\mu}{3}\right)^{4/3} + \frac{151}{243}\left(\frac{\mu}{3}\right)^{5/3} - \frac{1}{9}\left(\frac{\mu}{3}\right)^{2} \tag{4.77}$$

$$\gamma_2 = \left(\frac{\mu}{3}\right)^{1/3} + \frac{1}{3}\left(\frac{\mu}{3}\right)^{2/3} - \frac{1}{9}\left(\frac{\mu}{3}\right)^{1} - \frac{31}{81}\left(\frac{\mu}{3}\right)^{4/3} - \frac{119}{243}\left(\frac{\mu}{3}\right)^{5/3} - \frac{1}{9}\left(\frac{\mu}{3}\right)^{2} \tag{4.78}$$

However, Szebehely[5] notes that Eqs. (4.73)-(4.75) are most easily solved using an iterative method with the following as the initial guesses:

$$\gamma_1 = \gamma_2 = \left(\frac{\mu^*}{3\left(1-\mu^*\right)}\right)^{1/3} \tag{4.79}$$

$$\gamma_3 = 1 \tag{4.80}$$

GMAT uses the Newton-Raphson method to solve for the roots of the equations by iterating on

$$\gamma(i+1) = \gamma(i) - \frac{F(\gamma(i))}{F'(\gamma(i))} \tag{4.81}$$

until the the difference $|\gamma(i+1)-\gamma(i)| < 10^{-8}$. The derivative $F'(\gamma)$ for each libration point is shown below.

$$\text{(For L1)} \quad F'(\gamma) = 5\gamma_1^4 - 4\left(3-\mu^*\right)\gamma_1^3 + 3\left(3-2\mu^*\right)\gamma_1^2 - 2\mu^*\gamma_1 + 2\mu^* \tag{4.82}$$

$$\text{(For L2)} \quad F'(\gamma) = 5\gamma_2^4 - 4\left(3-\mu^*\right)\gamma_2^3 + 3\left(3-2\mu^*\right)\gamma_2^2 - 2\mu^*\gamma_2 - 2\mu^* \tag{4.83}$$

$$\text{(For L3)} \quad F'(\gamma) = 5\gamma_3^4 + 4\left(2+\mu^*\right)\gamma_3^3 + 3\left(1+2\mu^*\right)\gamma_3^2 - 2\left(1-\mu^*\right)\gamma_3 - 2\left(1-\mu^*\right) \tag{4.84}$$

Table 4.6: Location of Libration Points in RLP Frame, with the Origin at the Primary Body

Point	x-Position	y-Position
L1	$1-\gamma_1$	0
L2	$1+\gamma_2$	0
L3	$-\gamma_3$	0
L4	1/2	$\sqrt{3}/2$
L5	1/2	-$\sqrt{3}/2$

We now need to redimensionalize the results found in the rotating system, and perform the necessary transformations to obtain the results in the MJ2000 system. We know the position and velocity of the i^{th} libration point can be expressed in the rotating system with the origin centered on the primary body as

$$\mathbf{r}^i = L^* \begin{bmatrix} x_i & y_i & 0 \end{bmatrix}^T \tag{4.85}$$

$$\mathbf{v}^i = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T \tag{4.86}$$

Where L^* is the distance between the Primary and Secondary bodies, in the desired units (km in GMAT). Now we have the dimensionalized position and velocity vectors of the libration point in the rotating coordinate system defined by the motion of the secondary body with respect to the primary body. To determine the position and velocity vectors in the FK5 system, with the origin located at the primary body, we need to determine the rotation matrix and its derivative as follows:

$$\mathbf{R}^{Ii} = \begin{pmatrix} \hat{x}_1 & \hat{y}_1 & \hat{z}_1 \\ \hat{x}_2 & \hat{y}_2 & \hat{z}_2 \\ \hat{x}_3 & \hat{y}_3 & \hat{z}_3 \end{pmatrix} \tag{4.87}$$

and

$$\dot{\mathbf{R}}^{Ii} = \begin{pmatrix} \dot{\hat{x}}_1 & \dot{\hat{y}}_1 & \dot{\hat{z}}_1 \\ \dot{\hat{x}}_2 & \dot{\hat{y}}_2 & \dot{\hat{z}}_2 \\ \dot{\hat{x}}_3 & \dot{\hat{y}}_3 & \dot{\hat{z}}_3 \end{pmatrix} \tag{4.88}$$

where

$$\hat{\mathbf{x}} = \frac{\mathbf{r}}{r} \tag{4.89}$$

$$\hat{\mathbf{z}} = \frac{\mathbf{r}\times\mathbf{v}}{\|\mathbf{r}\times\mathbf{v}\|} \tag{4.90}$$

$$\hat{\mathbf{y}} = \hat{\mathbf{z}}\times\hat{\mathbf{x}} \tag{4.91}$$

and

$$\dot{\hat{\mathbf{x}}} = \dot{\hat{\mathbf{r}}} = \frac{\mathbf{v}}{r} - \frac{\hat{\mathbf{r}}}{r}(\hat{\mathbf{r}} \cdot \mathbf{v}) \tag{4.92}$$

$$\dot{\hat{\mathbf{z}}} = \dot{\hat{\mathbf{n}}} = \frac{\mathbf{r} \times \mathbf{a}}{n} - \frac{\hat{\mathbf{n}}}{n}(\mathbf{r} \times \mathbf{a} \cdot \hat{\mathbf{n}}) \tag{4.93}$$

$$\dot{\hat{\mathbf{y}}} = \dot{\hat{\mathbf{z}}} \times \dot{\hat{\mathbf{x}}} \tag{4.94}$$

We finally arrive at the position of the Libration Point in the FK5 system with the origin at the primary by performing the calculations:

$$\mathbf{r} = \mathbf{R}^{Ii}\mathbf{r}^i \tag{4.95}$$

$$\mathbf{v} = \dot{\mathbf{R}}^{Ii}\mathbf{v}^i \tag{4.96}$$

Note that in the GMAT script language, we can define this coordinate system using

```
Create CoordinateSystem CS;
CS.Origin    = PrimaryBodyName;
CS.Primary   = PrimaryBodyName;
CS.Secondary = SecondaryBodyName;
CS.XAxis = R;
CS.ZAxis = N;
```

4.17 Barycenter

The barycenter of a system of point masses, $\mathbf{r}_b$, is also called the center of mass. If we have a system of n bodies, and we know the position of the i^{th} body with respect to a common reference system, then we can calculate the barycenter of the system using

$$\mathbf{r}_b = \frac{\sum_{i=1}^{n} m_i \mathbf{r}_i}{\sum_{i=1}^{n} m_i} \tag{4.97}$$

Similarly, we can calculate the velocity of the barycenter using the following equation

$$\mathbf{v}_b = \frac{\sum_{i=1}^{n} m_i \mathbf{v}_i}{\sum_{i=1}^{n} m_i} \tag{4.98}$$

Chapter 5

Dynamics Modelling

5.1 Equations of Motion

$$\frac{d^2\vec{r}_{21}}{dt^2} = -G\frac{(m_2+m_1)}{|\vec{r}_{21}|^3}\vec{r}_{21} - \sum_{j=3}^{N} G\left(\frac{m_j}{|\vec{r}_{j1}|^3}\vec{r}_{j1} - \frac{m_j}{|\vec{r}_{j2}|^3}\vec{r}_{j2}\right) + \frac{\vec{F}_{other}}{m_1} - \frac{\vec{F}_{other}}{m_2} - \frac{\vec{r}_1}{m_1}\frac{dm_1}{dt} + \frac{\vec{v}_2}{m_2}\frac{dm_2}{dt}$$

5.1.1 Multiple Spacecraft Propagation

5.2 Drag Modelling

5.2.1 Jacchia Roberts

5.2.2 MSISE-90

A. E. Hedin, Extension of the MSIS Thermospheric Model into the Middle and Lower Atmosphere, J. Geophys. Res. 96, 1159, 1991.

For testing http://nssdc.gsfc.nasa.gov/space/model/models/msis.html

http://www.agu.org/journals/ja/ja0212/2002JA009430/
go to auxillary material on the left side menu and open the tables-datasets.doc

Other useful models http://nssdc.gsfc.nasa.gov/space/model/

5.2.3 Exponential

5.3 Non-Spherical Gravity

$$\psi\left(r,\phi,\lambda\right)=\frac{-\mu}{r}\sum_{n=1}^{\infty}c_{n}^{o}\left(\frac{r_{e}}{r}\right)^{n}P_{n}^{o}\left(\sin\phi\right)+\frac{\mu}{r}\sum_{n=1}^{\infty}\sum_{m=1}^{n}\left(\frac{r_{e}}{r}\right)^{n}P_{n}^{m}\left(\sin\phi\right)\left[S_{n}^{m}\sin m\lambda+C_{n}^{m}\cos m\lambda\right]$$

5.4 Solar Radiation Pressure

Chapter 6

Numerical Integrators

6.1 RungeKutta Integrators

6.2 Prince-Dormand Integrators

6.3 Adams Bashforth Moulton

Implementation of the Adams-Bashford-Moulton Predictor-Corrector.

This code implements a fourth-order Adams-Bashford predictor / Adams-Moulton corrector pair to integrate a set of first order differential equations. The algorithm is found at `http://chemical.caeds.eng.uml.edu/onlinec/w` or in Bate, Mueller and White, pp. 415-417.

The predictor step extrapolates the next state r_{i+1} of the variables using the the derivative information (f) at the current state and three previous states of the variables, by applying the equation

$$r_{i+1}^{*j} = r_i^j + \frac{h}{24}\left[55f_n^j - 59f_{n-1}^j + 37f_{n-2}^j - 9f_{n-3}^j\right]$$

The corrector uses derivative information evaluated for this state, along with the derivative information at the original state and two preceding states, to tune this state, giving the final, corrected state:

$$r_{i+1}^j = r_i^j + \frac{h}{24}\left[9f_{n+1}^{*j} + 19f_n^j - 5f_{n-1}^j + 1f_{n-2}^j\right]$$

Bate, Mueller and White give the estimated accuracy of this solution to be

$$ee = \frac{19}{270}\left|r_{i+1}^{*j} - r_{i+1}^j\right|$$

Method used to fire the step refinement (the corrector phase).

6.4 Stopping Condition Algorithm

6.5 Bulirsch-Stoer

Chapter 7

Solvers

7.1 Differential Correction

7.2 Broyden's Method

7.3 Newton's Method

7.4 Matlab fmincon

Chapter 8

Spacecraft Model

8.1 Orbit

8.2 Attitude

8.3 Ballistics and Mass

8.4 Actuators

8.4.1 Thrust and Impulse Models

GMAT uses polynomial expressions for the thrust and specific impulse imparted to the spacecraft by thrusters attached to the spacecraft. Both thrust and specific impulse are expressed as functions of pressure and temperature. The pressure and temperature are values obtained from fuel tanks containing the fuel. All measurements in GMAT are expressed in metric units. The thrust, in Newtons, applied by a spacecraft engine is given by

$$\begin{aligned} F_T(P,T) &= \left\{C_1 + C_2P + C_3P^2 + C_4P^{C_5} + C_6P^{C_7} + C_8P^{C_9} \right. \\ &\quad \left. + C_{10}C_{11}^{C_{12}P}\right\}\left(\frac{T}{T_{ref}}\right)^{1+C_{13}+C_{14}P} \end{aligned}$$

Pressures are expressed in kilopascals, and temperatures in degrees centigrade. The coefficients C1 - C14 are set by the user. Each coefficient is expressed in units commiserate with the final expression in Newtons; for example, C1 is expressed in Newtons, C2 in Newtons per kilopascal, and so forth.

Specific Impulse, measured in m/s (or, equivalently, Newton Seconds/kilogram) is expressed using a similar equation:

$$I_{sp}(P,T) = \left\{K_1 + K_2P + K_3P^2 + K_4P^{K_5} + K_6P^{K_7} + K_8P^{K_9} + K_{10}K_{11}^{K_{12}P}\right\}\left(\frac{T}{T_{ref}}\right)^{1+K_{13}+K_{14}P} \quad (8.1)$$

Table 8.1: Default Thrust and Specific Impulse Coefficients

Thrust Coefficient	Default	Units	Impulse Coefficient	Default	Units
C_1	500	N	K_1	2150	m/s
C_2	0	N/kPa	K_2	0	$\mathrm{m/(s \cdot kPa)}$
C_3	0	$\mathrm{N/kPa^2}$	K_3	0	$\mathrm{m/(s \cdot kPa^2)}$
C_4	0	$\mathrm{N/kPa^{C_5}}$	K_4	0	$\mathrm{m/(s \cdot kPa^{K_5})}$
C_5	0	none	K_5	0	none
C_6	0	$\mathrm{N/kPa^{C_7}}$	K_6	0	$\mathrm{m/(s \cdot kPa^{K_7})}$
C_7	0	none	K_7	0	none
C_8	0	$\mathrm{N/kPa^{C_9}}$	K_8	0	$\mathrm{m/(s \cdot kPa^{K_9})}$
C_9	0	none	K_9	0	none
C_{10}	0	N	K_{10}	0	m/s
C_{11}	1	none	K_{11}	1	none
C_{12}	0	1/kPa	K_{12}	0	1/kPa
C_{13}	0	none	K_{13}	0	none
C_{14}	0	1/kPa	K_{14}	0	1/kPa

Table 8.1 shows the default values for these coefficients.

8.5 Sensors

8.6 Tanks

Mass is depleted from the fuel tanks using equation

$$\frac{dm}{dt} = \frac{F_T}{I_{sp}}, \tag{8.2}$$

where the thrust and specific impulse are given by **??** and 8.1.

This mass depletion is integrated along with the other parameters during propagation.

The tank model in GMAT manages the fuel mass and the input variables for the thrust and specific impulse polynomials. The tank can be run in either a blow-down or pressure regulated mode. In pressure regulated mode, the pressure in the polynomial is held at a fixed value. In blow-down mode, the pressure decreases as fuel is used, following the ideal gas law:

$$PV = nRT \tag{8.3}$$

In GMAT's blow-down model, the temperature T and the number of pressurant molecules n in the tank are held constant, so the right side of this equation is constant. The gas volume available in the tank grows as fuel is consumed, and the pressure decreases accordingly. The gas volume V_G in the tank is computed from the total tank volume, V_T, the mass of the fuel, M_F, and the density of the fuel, ρ:

$$V_G = V_T - \frac{M_F}{\rho} \tag{8.4}$$

Table 8.2: Default Fuel Tank Parameters

Parameter	Default Value	Units
FuelMass	756	kg
Pressure	1500	kPa
Temperature	20	C
RefTemperature	20	C
Volume	0.75	m^3
FuelDensity	1260	kg/m^3
PressureRegulated	true	

Table 2 shown the default values for the tank parameters.

Chapter 9

MathSpecAppendices

9.1 Coordinate System Transformation Algorithm

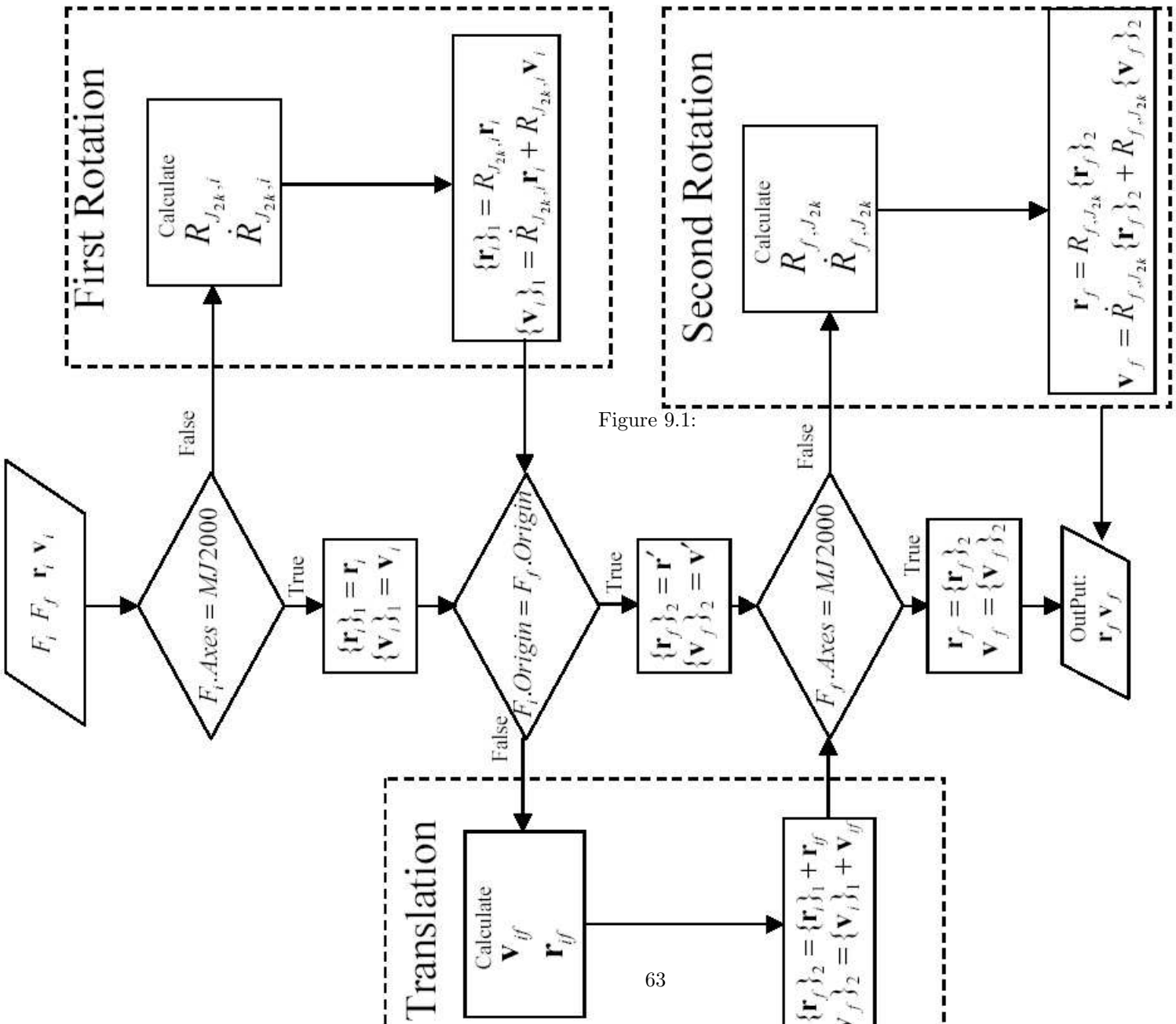

Figure 9.1:

9.2 Appendix: Pole and Meridian Locations

Table 9.1: Recommended Values for Pole and Prime Meridian Locations of the Sun and Planets[1]

Name	Values
Sun	$\alpha_o = 286.13°$ (deg) $\delta_o = 63.87°$ (deg) $W = 84.10° + 14.1844000°d$ (deg) $\dot{W} = 14.1844000°$ (deg/s)
Mercury	$\alpha_o = 281.01 - 0.033T$ $\delta_o = 61.45 - 0.005T$ $W = 329.548 + 6.1385025d$ $\dot{W} = 6.1385025$
Venus	$\alpha_o = 272.76$ $\delta_o = 67.16$ $W = 160.20 - 1.4813688d$ $\dot{W} = -1.4813688$
Earth	$\alpha_o = 0.00 - 0.641T$ $\delta_o = 90.00 - 0.557T$ $W = 190.147 + 360.9856235d$ $\dot{W} = 360.9856235$ *Earth Data is included for completeness only, GMAT uses FK5 reduction for the Earth*
Mars	$\alpha_o = 317.68143 - 0.1061T$ $\delta_o = 52.88650 - 0.0609T$ $W = 176.630 + 350.89198226d$ $\dot{W} = 350.89198226$
Jupiter	$\alpha_o = 285.05 - 0.009T$ $\delta_o = 64.49 + 0.003T$ $W = 284.95 + 870.5366420d$ $\dot{W} = 870.5366420$
Saturn	$\alpha_o = 40.589 - 0.036T$ $\delta_o = 83.537 - 0.004T$ $W = 38.90 + 810.7939024d$ $\dot{W} = 810.7939024$
Uranus	$\alpha_o = 257.311$ $\delta_o = -15.175$ $W = 203.81 - 501.1600928d$ $\dot{W} = -501.1600928$
Neptune	$\alpha_o = 299.36 + 0.70 \sin N$ $\delta_o = 43.46 - 0.51 \cos N$ $W = 253.18 + 536.3128492d - 0.48 \sin N$ $\dot{W} = 536.3128492 - 0.48\dot{N} \cos N$ $N = 357.85 + 52.316T$ $\dot{N} = 6.0551 \times 10^{-4}$ (deg/day)
Pluto	$\alpha_o = 313.02$ $\delta_o = 9.09$ $W = 236.77 - 56.3623195d$ $\dot{W} = -56.3623195$

Table 9.2: Recommended Values for Pole and Prime Meridian Locations of Luna[1]

Name	Values			
Luna				
	$\alpha_o =$ 269.9949	$+0.0031T$	$-3.8787 \sin E1$	$-0.1204 \sin E2$
		$+0.0700 \sin E3$	$-0.0172 \sin E4$	$+0.0072 \sin E6$
		$-0.0052 \sin E10$	$-0.0043 \sin E13$	
	$\delta_o =$ 66.5392	$+0.0130T$	$+1.5419 \cos E1$	$+0.0239 \cos E2$
		$-0.0278 \cos E3$	$+0.0068 \cos E4$	$-0.00292 \cos E6$
		$+0.0009 \cos E7$	$+.0008 \cos E10$	$-0.0009 \cos E13$
	$W =$ 38.3213	$+13.17635815d$	$-1.4 \times 10^{-12} d^2$	$+3.5610 \sin E1$
		$+0.1208 \sin E2$	$-0.0642 \sin E3$	$+0.0158 \sin E4$
		$+0.0252 \sin E5$	$-0.0066 \sin E6$	$-0.0047 \sin E7$
		$-0.0046 \sin E8$	$+0.0028 \sin E9$	$+0.0052 \sin E10$
		$+0.0040 \sin E11$	$+0.0019 \sin E12$	$-0.0044 \sin E13$
	$\dot{W} =$	$+13.17635815$	$-2.8 \times 10^{-12} d$	$-.18870 \cos E1$
		$-.01280 \cos E2$	$-.835 \cos E3$	$+.211 \cos E4$
		$+.0248 \cos E5$	$-.17 \cos E6$	$-.061 \cos E7$
		$-.0015 \cos E8$	$+.0049 \cos E9$	$-.00083 \cos E10$
		$+.00001 \cos E11$	$+.00031 \cos E12$	$-.057 \cos E13$

where

$E1 = 125.045 - 0.0529921d$ $E2 = 250.089 - 0.1059842d$
$E3 = 260.008 + 13.0120009d$ $E4 = 176.625 + 13.3407154d$
$E5 = 357.529 + 0.9856003d$ $E6 = 311.589 + 26.4057084d$
$E7 = 134.963 + 13.0649930d$ $E8 = 276.617 + 0.3287146d$
$E9 = 34.226 + 1.7484877d$ $E10 = 15.134 - 0.1589763d$
$E11 = 119.743 + 0.0036096d$ $E12 = 239.961 + 0.1643573d$
$E13 = 25.053 + 12.9590088d$

Part II

Users Guide

Chapter 10

Introduction

10.1 Script Overview

The GMAT system must be configurable from either a Graphical User Interface (GUI), or a script file. This document defines the script syntax for GMAT. The script syntax is designed with three main goals in mind. The script is to be powerful and allow full configuration of the system from a script file. However, the script syntax must be intuitive and easy to use. Finally, the script syntax should allow a for a seamless integration into scripts existing in Matlab. This is advantageous for two reasons. First, many FDAB engineers are familiar with Matlab's script syntax, and a similar syntax will be intuitive for many analysts. Secondly, providing a script syntax that can be used in Matlab as well as GMAT, with no change to the script, will be extraordinarily powerful.

General Syntax Definition

To provide a script that can be used in either Matlab or GMAT, there are some constraints on the syntax. These constraints are imposed mainly due to the capabilities of Matlab's interpreter. Hence, we must use a script approach that allows Matlab's interpreter to easily distinguish between Matlab commands and GMAT commands. Fortunately, this can be accomplished by placing only limited restrictions on the GMAT script syntax.

In general, the syntax has the following structure.

GMAT CommandName Arg1 Arg2 ArgN

where GMAT CommandName is the name of a specific GMAT command, and Arg1 Arg2 are alpha-numeric characters. Limitations on the argument sequence are listed below.

Arg1 cannot be a right parenthesis GMAT CommandName cannot be the same as an existing matlab function name Single quotes cannot appear in the argument list Commas can only occur inside ( ), [ ], or Semi colons can only appear at the end of a line Arguments cannot be a percent sign All expressions in Matlab that are to be sent to GMAT for evaluation, must be preceded by the command GMAT Line extensions cannot occur within (), [], or

Some examples of script commands are shown below.

Propagate Mode PropName(SC1, SC2,..,SCN,StopConditions); Create Array UserArray = [0 90 180 270];

To provide compatibility with Matlab, GMAT's script syntax should use Matlab's syntax for common programming functionality. Hence line extensions are defined by three consecutive periods (). Parenthesis

are used to specify function input arguments. Square brackets are used to define arrays of real numbers and integers. Hence, strings, structures, and objects cannot appear within square brackets. Curly brackets are used to define cell arrays and can contain real numbers, integers, strings, structures, or other cell arrays. Matlab function calls have the following calling sequence.

[Out1, Out2,, OutN] = FunctionName(In1, In2,, In3)

The remainder of this document is devoted to defining specific commands and capabilities in GMAT. The commands are broken down into two sections. The first section is devoted to creating objects in Matlab. In this section we define how to create spacecraft, plots, and propagators among others. The second section is devoted to mission sequence commands. In this section we address how use predefined objects in the mission sequence.

10.2 GUI Overview

Chapter 11

Spacecraft

In GMAT, the user can define the state of a spacecraft in many different ways. The user can also attach thrusters, tanks, and sensors to a spacecraft for use in the mission sequence. In this chapter, we discuss how to create and configure a spacecraft object in GMAT. This includes defining the orbit state, the ballistic and mass properties, and any tanks and thrusters that the spacecraft possesses.

11.1 Example

Let's begin by looking at a simple script example. Below we see how to create and configure a basic spacecraft using the script. In this particular example, we are defining the orbit state using the Keplerian elements, and we are setting the orbit epoch in the TAI time system using the Gregorian Date format. Working from the GUI, as shown in Fig. 11.1, the user can create a new spacecraft by starting at the Resource Tree and right clicking on the Spacecraft folder, and selecting Add Spacecraft. Using a double left click, you can open the Spacecraft dialogue box. There are a number of tabs that allow you to set different types of physical properties of a Spacecraft.

```
Create Spacecraft Sat1
GMAT Sat1.Epoch.TAIGregorian  = 01 Jan 2000 12:00:00.000;
GMAT Sat1.SMA        = 8000;
GMAT Sat1.ECC        = .01;
GMAT Sat1.INC        = 28.5;
GMAT Sat1.AOP        = 0;
GMAT Sat1.RAAN       = 90;
GMAT Sat1.TA         = 180;
GMAT Sat1.Cd         = 2.0;
GMAT Sat1.Cr         = 1.4;
GMAT Sat1.DragArea   = 1;
GMAT Sat1.SRPArea    = 1;
GMAT Sat1.DryMass    = 100;
GMAT Sat1.Tanks      = {};
GMAT Sat1.Thrusters  = {};
```

In the next few sections, we will discuss how to set the orbit state and epoch, spacecraft ballistic and mass properties, and configure and attach tanks and thrusters to a spacecraft. Let's being by looking at the orbit state and epoch.

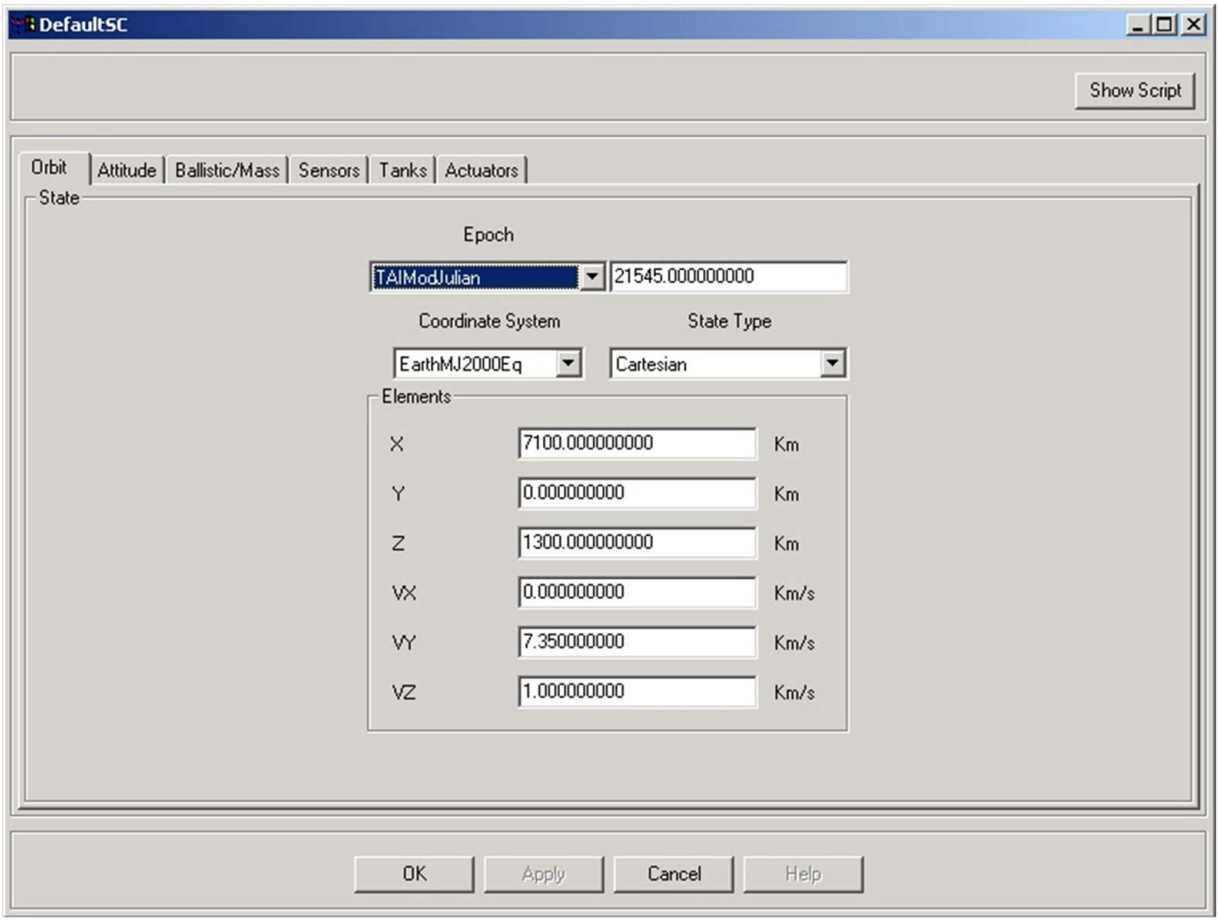

Figure 11.1: Spacecraft Dialogue Box

11.2 Spacecraft State and Epoch

The user can choose between the following state representations to define the orbit state: `Cartesian`, `Keplerian`, `ModifiedKeplerian`, `SphericalAZFPA`, `SphericalRADEC` . A detailed discussion of these state representations is discussed below and summarized in Tables.

The default state representation in GMAT is the cartesian state vector in the Earth Centered FK5 coordinate system. From the script, you can set the cartesian state one of two ways. The first approach is seen below.

```
Create Spacecraft Sat;
GMAT Sat.StateType = Cartesian;
GMAT Sat.X = 7082.960306079;
GMAT Sat.Y = 7.052885901083;
GMAT Sat.Z = -48.94363747128;
GMAT Sat.VX = 0.05237087208446;
GMAT Sat.VY = -1.069925309259;
GMAT Sat.VZ = 7.424767278548;
```

An alternative approach to define the Cartesian state of a spacecraft is given by.

```
Create Spacecraft Sat;
GMAT Sat.StateType = Cartesian;
GMAT Sat.Element1 = 7000;    % X
GMAT Sat.Element2 = 0.0;     % Y
GMAT Sat.Element3 = 0.0;     % Z
GMAT Sat.Element4 = 0.0;     % VX
GMAT Sat.Element5 = 8.0;     % VY
GMAT Sat.Element6 = 0.0;     % VZ
```

Similarly, the user can define the spacecraft state using Keplerian elements using the following script.

```
Create Spacecraft Sat;
GMAT Sat.StateType = Keplerian;
GMAT Sat.SMA  = 8000;
GMAT Sat.ECC  = 0.01;
GMAT Sat.INC  = 28.5;
GMAT Sat.AOP  = 0;
GMAT Sat.RAAN = 90;
GMAT Sat.TA   = 180;
```

There are three remaining state types that the user can use to specify the spacecraft state. They are `ModKeplerian, SphericalAZFPA`, and `SphericalRADEC`. Below are script examples for setting a spacecraft using any of these three state representations.

```
Create Spacecraft Sat
GMAT Sat.StateType = ModifiedKeplerian;
GMAT Sat.RadPer = 8000;
GMAT Sat.RadApo = 20000;
GMAT Sat.INC = 28.5;
GMAT Sat.RAAN = 90;
GMAT Sat.AOP = 45;
GMAT Sat.TA = 180;
```

```
Create Spacecraft Sat;
GMAT Sat.StateType  = SphericalAZFPA;
GMAT Sat.RMAG = 8000;
GMAT Sat.RA   = 90;
GMAT Sat.DEC  = 0;
GMAT Sat.VMAG = 7.3;
GMAT Sat.AZI  = 45;
GMAT Sat.FPA  = 0;
```

```
Create Spacecraft Sat;
GMAT Sat.StateType  = SphericalRADEC;
GMAT Sat.RMAG = 8000;
GMAT Sat.RA = 10;
GMAT Sat.DEC = 45;
GMAT Sat.VMAG = 7.3;
GMAT Sat.RAV = 45;
GMAT Sat.DECV = 45;
```

If working in the GUI, you first need to choose the desired state representation which is found on the orbit tab of the Spacecraft dialogue box. After choosing the desired state representation, the GUI configures to allow you to input the state vector. Tables .... contain a description of each element for all of the available state representations.

Table 11.1: Fields Associated with a Spacecraft Orbit State

Field	Options and Description
`StateType`	Default: `Cartesian`. Options: [`Cartesian`, `Keplerian`, `ModifiedKeplerian`, `SphericalAZFPA`, `SphericalRADEC`].
	Fields associated with `Cartesian` state.
`X`	Default: 7100. Options: [Real Number]: `X` is the x-component of the Spacecraft state in the coordinate system chosen in the Spacecraft `CoordinateSystem` field. Units: km.
`Y`	Default: 0. Options: [Real Number]: `Y` is the y-component of the Spacecraft state in the coordinate system chosen in the Spacecraft `CoordinateSystem` field. Units: km.
`Z`	Default: 1300. Options: [Real Number]: `Z` is the z-component of the Spacecraft state in the coordinate system chosen in the Spacecraft `CoordinateSystem` field. Units: km.
`VX`	Default: 0. Options: [Real Number]: `VX` is the x-component of the Spacecraft velocity in the coordinate system chosen in the Spacecraft `CoordinateSystem` field. Units: km.
`VY`	Default: 7.35. Options: [Real Number]: `VY` is the y-component of the Spacecraft velocity in the coordinate system chosen in the Spacecraft `CoordinateSystem` field. Units: km.
`VZ`	Default: 1.0. Options: [Real Number]: `VZ` is the z-component of the Spacecraft velocity in the coordinate system chosen in the Spacecraft `CoordinateSystem` field. Units: km.
	Fields associated with `Keplerian` state.
`SMA`	Default: [Real Number {7191.938817629}]: The SMA field is the spacecraft orbit's osculating Keplerian semimajor axis in km. SMA must be be strictly greater than or less than zero.
`ECC`	[Real Number {0.024549749}]: The ECC field is the spacecraft orbit's osculating eccentricity. ECC must be greater than zero.
`INC`	[Real Number {12.850080057}]: The INC field is the spacecraft orbit's osculating inclination, in degrees, w/r/t to the selected coordinate system.
`AOP`	[Real Number {314.190551536}]: The AOP field is the spacecraft orbit's osculating argument of periapsis, in degrees, w/r/t to the selected coordinate system.
`RAAN`	[Real Number {306.614802195}]: The RAAN field is the spacecraft orbit's osculating right ascension of the ascending node, in degrees, w/r/t to the selected coordinate system.
`TA`	[Real Number {306.614802195}]: The TA field is the spacecraft orbit's osculating true anomaly, in degrees.
	Fields associated with `ModifiedKeplerian` state.
`RadApo`	Default: . Options: . The `RadApo` field is the spacecraft orbit's osculating radius of apoapsis. `RadApo` must be be strictly greater than or less than zero. Units: km.

Table 11.1: (Fields Associated with a Spacecraft Orbit State. continued)

Field	Options and Description
RadPer	Default: Options: The RadPer field is the spacecraft orbit's osculating radius of periapsis. RadPer must be be greater than zero. Units: km.
	Fields associated with SphericalAZFPA state.
RMAG	Default: [Real Number {7191.938817629}]: The SMA field is the spacecraft orbit's osculating Keplerian semimajor axis in km. SMA must be be strictly greater than or less than zero.
RA	[Real Number {0.024549749}]: The ECC field is the spacecraft orbit's osculating eccentricity. ECC must be greater than zero.
DEC	[Real Number {12.850080057}]: The INC field is the spacecraft orbit's osculating inclination, in degrees, w/r/t to the selected coordinate system.
VMAG	[Real Number {314.190551536}]: The AOP field is the spacecraft orbit's osculating argument of periapsis, in degrees, w/r/t to the selected coordinate system.
AZI	[Real Number {306.614802195}]: The RAAN field is the spacecraft orbit's osculating right ascension of the ascending node, in degrees, w/r/t to the selected coordinate system.
FPA	[Real Number {306.614802195}]: The TA field is the spacecraft orbit's osculating true anomaly, in degrees.
	Fields associated with SphericalRADEC state.
SMA	Default: [Real Number {7191.938817629}]: The SMA field is the spacecraft orbit's osculating Keplerian semimajor axis in km. SMA must be be strictly greater than or less than zero.
ECC	[Real Number {0.024549749}]: The ECC field is the spacecraft orbit's osculating eccentricity. ECC must be greater than zero.
INC	[Real Number {12.850080057}]: The INC field is the spacecraft orbit's osculating inclination, in degrees, w/r/t to the selected coordinate system.
AOP	[Real Number {314.190551536}]: The AOP field is the spacecraft orbit's osculating argument of periapsis, in degrees, w/r/t to the selected coordinate system.
RAAN	[Real Number {306.614802195}]: The RAAN field is the spacecraft orbit's osculating right ascension of the ascending node, in degrees, w/r/t to the selected coordinate system.
TA	[Real Number {306.614802195}]: The TA field is the spacecraft orbit's osculating true anomaly, in degrees.

The user can specify the epoch of the spacecraft in four different formats: TAIModJulian, TAIGregorian, UTCModJulian, UTCGregorian . Below is a script example that shows how to set the Epoch using each of the available formats. In the GUI the epoch is set by chosing the desired format in the Epoch combo box located on the orbit tab of the spacecraft dialogue box.

```
Sat1.Epoch.TAIModJulian  = 21545.0003703704  % TAI in Modified Julian Date Format
Sat1.Epoch.TAIGregorian  = 01 Jan 2000 12:00:32.0000;   % TAI in Gregorian Date Format
```

```
Sat1.Epoch.UTCModJulian  = 21545.00000; % UTC in Modified Julian Date Format
Sat1.Epoch.UTCGregorian  = 01 Jan 2000 12:00:00.000;  % UTC in Gregorian Date Format
```

Table 11.2: Fields Associated with a Spacecraft Epoch

Field	Options and Description
`TAIGregorian`	[Gregorian Date {01 Jan 2000 12:00:00.000}]: The TAIGregorian field allows the user to enter the initial spacecraft epoch in the TAI time system using a Gregorian Date format.
`TAIModJulian`	[Real Number {21545.000000000}]: The TAIModJulian field allows the user to enter the initial spacecraft epoch in the TAI time system using a Modified Julian format.
`UTCGregorian`	[Gregorian Date {01 Jan 2000 11:59:27.966}]: The UTCGregorian field allows the user to enter the initial spacecraft epoch in the UTC time system using a Gregorian Date format.
`UTCModJulian`	[Real Number {21544.999629236}]: The UTCModJulian field allows the user to enter the initial spacecraft epoch in the UTC time system using a Modified Julian format.

11.3 Spacecraft Ballistic and Mass Properties

Spacecraft ballistic and mass properties include the drag coefficient, the coefficient of refectivity, the drag area, the solar radiation pressure, and the dry mass. The script example below shows how to define these input parameters. Working from the GUI, these parameters are set from the Ballistic/Mass tab found on the spacecraft dialogue box. A description of each field is found in Table 20.3

```
GMAT Sat1.Cd       = 2.0;
GMAT Sat1.Cr       = 1.4;
GMAT Sat1.DragArea = 1;
GMAT Sat1.SRPArea  = 1;
GMAT Sat1.DryMass  =   100;
```

Table 11.3: Fields Associated with a Spacecraft Ballistic and Mass Properties

Field	Options and Description
`Cd`	Default: 2.2. Options: [Real Number]: `Cd` is the spacecraft's drag coefficient. `Cd` must be greater than 0. Units: Dimensionless.
`Cr`	Default: 2.2. Options: [Real Number]: `Cr` is the spacecraft's coefficient of reflectivity. `Cr` must be greater than 0. Units: Dimensionless.

Table 11.3: (continued)

Field	Options and Description
`DragArea`	Default: 15.0. Options: [Real Number]: The `DragArea` is the area of the spacecraft that is used in calculating atmospheric drag. `DragArea` must be greater than 0. Units: m^2
`SRPArea`	Default: 1.0. Options: [Real Number]: The `SRPArea` is the area of the spacecraft that is used in calculating the force due to solar radiation pressure. `SRPArea` must be greater than 0. Units: m^2
`DryMass`	Default: 850.0. Options: [Real Number]: The `DryMass` is the mass of the spacecraft without the mass of tanks and fuel. `DryMass` must be greater than 0. Units: kg

11.4 Spacecraft Hardware

In GMAT, the user can configure tanks and thrusters, and then attach them to spacecraft for use in manuever events. In this section, we'll look at how to create a tank and a thruster and define their physical properties. We'll also look at how to attach tanks and thrusters to spacecraft. GMAT makes copies of each of these elements before assigning them to the spacecraft so that users can assign the same tank or thruster to multiple spacecraft. This allows a user to configure multiple spacecraft with identical thruster and tank models.

11.4.1 Tanks

Scripting for the tank model involves creating the tank and setting several parameters: the temperature and reference temperature, fuel mass, fuel density, initial tank pressure, total tank volume, and a flag indicating if the tank is running in blow-down mode or pressure regulated mode. The following example illustrates how to set these parameters working from the script.

```
Create FuelTank DefaultFuelTank;
GMAT DefaultFuelTank.X_Direction = 1;
GMAT DefaultFuelTank.Y_Direction = 0;
GMAT DefaultFuelTank.Z_Direction = 0;
GMAT DefaultFuelTank.FuelMass = 756;
GMAT DefaultFuelTank.Pressure = 1500;
GMAT DefaultFuelTank.Temperature = 20;
GMAT DefaultFuelTank.RefTemperature = 20;
GMAT DefaultFuelTank.Volume = 0.75;
GMAT DefaultFuelTank.FuelDensity = 1260;
GMAT DefaultFuelTank.PressureRegulated = true;
```

Table 11.4: Fields Associated with a Spacecraft Tank

Field	Options and Description
`FuelMass`	Default: 756. Options: [Real Number]: The `FuelMass` field is the mass of fuel in the tank. Units: kg.

Table 11.4: (continued)

Field	Options and Description
`Pressure`	Default: 1500. Options: [Real Number]: The `Pressure` field is the pressure of the fuel in the tank. Units: kPa.
`Temperature`	Default: 20. Options: [Real Number]: The `Temperature` field is the temperature of the fuel in the tank. Units: C.
`RefTemperature`	Default: 20. Options: [Real Number]: `RefTemperature` Units: C.
`Volume`	Default: 0.75. Options: The `Volume` field is the volume of the tank. [Real Number]:
`FuelDensity`	Default: 1260. Options: [Real Number]:
`PressureRegulated`	Default: `true`. Options: [`true` `false`]:

11.4.2 Thrusters

GMAT's thruster model contains the polynomial coefficients used to model thrust and specific impulse and parameters that identify the fuel tank used by the thruster, the coordinate system used to orient the thruster in spaceand the direction of the thrust in that coordinate system. (Note that the orientation specified in the model is the direction of the applied thrust, not the direction that gas is expelled from the thruster.

```
Create Thruster DefaultThruster;
GMAT DefaultThruster.CoordinateSystem = MJ2000EarthEquator;
GMAT DefaultThruster.X_Direction = 1;
GMAT DefaultThruster.Y_Direction = 0;
GMAT DefaultThruster.Z_Direction = 0;
GMAT DefaultThruster.ThrustScaleFactor = 1;
GMAT DefaultThruster.Tank = {};
GMAT DefaultThruster.C1 = 500;
GMAT DefaultThruster.C2 = 0;
   .
   .
   .
GMAT DefaultThruster.C14 = 0;
GMAT DefaultThruster.K1 = 2150;
GMAT DefaultThruster.K2 = 0;
   .
   .
   .
GMAT DefaultThruster.K14 = 0;
```

Table 11.5: Fields Associated with a Spacecraft Thruster

Field	Options and Description
`CoordinateSystem`	Default: EarthMJ2000Eq. Options: [Coordinate System Name]: 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.
`X_Direction`	Default: 1. Options: [Real Number]: `X_Direction`, divided by the RSS of the three direction vectors, forms the x direction of the spacecraft thrust vector direction.
`Y_Direction`	Default: 0. Options: [Real Number]: `Y_Direction`, divided by the RSS of the three direction vectors, forms the y direction of the spacecraft thrust vector direction
`Z_Direction`	Default: 0. Options: [Real Number]: `Z_Direction`, divided by the RSS of the three direction vectors, forms the z direction of the spacecraft thrust vector direction
`ThrustScaleFactor`	Default: 1. Options: [Real Number]: `ThrustScaleFactor` is a scale factor that is multiplied by the thrust vector for a given thruster, before the thrust vector is added into the total accleration. `ThrustScaleFactor` must be greater than 0. Units: None.
`Tank`	Default: None. Options: [Tank Name]: The `Tank` field specifies which tank the thruster draws propellant from.
`C1`	Default: 500. Options: [Real]: Thruster coefficient. Units: N
`C2`	Default: 0. Options: [Real]: Thruster coefficient. Units: N/kPa.
`C3`	Default: 0. Options: [Real]: Thruster coefficient. Units: N/kPa^2
`C4`	Default: 0. Options: [Real]: Thruster coefficient. Units: N/kPa^{C5}.
`C5`	Default: 0. Options: [Real]: Thruster coefficient. Units: None
`C6`	Default: 0. Options: [Real]: Thruster coefficient. Units: N/kPa^{C7}.
`C7`	Default: 0. Options: [Real]: Thruster coefficient. Units: None
`C8`	Default: 0. Options: [Real]: Thruster coefficient. Units: N/kPa^{C9}.
`C9`	Default: 0. Options: [Real]: Thruster coefficient. Units: None
`C10`	Default: 0. Options: [Real]: Thruster coefficient. Units: N.
`C11`	Default: 1. Options: [Real]: Thruster coefficient. Units: None
`C12`	Default: 0. Options: [Real]: Thruster coefficient. Units: 1/kPa.
`C13`	Default: 0. Options: [Real]: Thruster coefficient. Units: None.

Table 11.5: (continued)

Field	Options and Description
C14	Default: 0. Options: [Real]: Thruster coefficient. Units 1/kPa.
K1	Default: 2150. Options: [Real]: Thruster coefficient. Units: m/s
K2	Default: 0. Options: [Real]: Thruster coefficient. Units: m/(s· kPa).
K3	Default: 0. Options: [Real]: Thruster coefficient. Units: m/(s· kPa^2)
K4	Default: 0. Options: [Real]: Thruster coefficient. Units: m/(s· kPa^{K5}).
K5	Default: 0. Options: [Real]: Thruster coefficient. Units: None
K6	Default: 0. Options: [Real]: Thruster coefficient. Units: m/(s· kPa^{K7}).
K7	Default: 0. Options: [Real]: Thruster coefficient. Units: None
K8	Default: 0. Options: [Real]: Thruster coefficient. Units: m/(s· kPa^{K9}.
K9	Default: 0. Options: [Real]: Thruster coefficient. Units: None
K10	Default: 0. Options: [Real]: Thruster coefficient. Units: m/s.
K11	Default: 1. Options: [Real]: Thruster coefficient. Units: None
K12	Default: 0. Options: [Real]: Thruster coefficient. Units: 1/kPa.
K13	Default: 0. Options: [Real]: Thruster coefficient. Units: None.
K14	Default: 0. Options: [Real]: Thruster coefficient. Units 1/kPa.

The default coefficients for the thrusters are shown in Table 8.1. The initial thrust direction is along the coordinate system X-axis. The default coordinate system is an inertial MJ2000 coordinate system.

11.4.3 Attaching Hardware to Spacecraft

The tank and thruster configurations described above are associated with spacecraft using several fields under the Spacecraft. GMAT makes copies of each of these elements before assigning them to the spacecraft so that users can assign the same tank or thruster to multiple spacecraft. This allows a user to configure multiple spacecraft with identical thruster and tank models. The following script example demonstrates how to attach predefined tanks and thrusters to a spacecraft.

```
   Create Spacecraft Sc;
GMAT Sc.Tanks = {tank1, tank2}
GMAT Sc.Thrusters = {engine1, engine2, engine3, engine4}
```

11.5 Formations and Constellations

Formations and Constellations are groups of spacecraft that are treated as a single object internally in GMAT. Once spacecraft are grouped into a formation, only the formation name needs to be used in a Propagate event, and all spacecraft will be propagated as a coupled system. The same is true for a constellation. Assuming we have previously created `Sat1, Sat2`, and `Sat3`, we can add them to a formation, and remove them from formation using the commands seen below.

```
   % To add spacecraft to formation
Create Formation MyFormation;
MyFormatoin.Add = Sat1;
MyFormatoin.Add = Sat2;
MyFormation.Add = Sat3;

% To remove spacecraft from formation
Create Formation MyFormation;
MyFormatoin.Add = Sat1;
MyFormatoin.Add = Sat2;
MyFormation.Add = Sat3;
```

Chapter 12

Propagators

In this chapter we discuss how to create a propagator in GMAT. A propagator is a combination of a force model and a numerical integrator. So, when a user sets up a propagator they are defining what terms to include in the equations of motion, and how the equations of motion should be numerically integrated. A propagator is different from a propagate event.A propagator is a resource and is found in the GUI under the resource tree. A propagate event is configured under the mission sequence tree. In a propagate event, the user tells GMAT what spacecraft to move forward or backward in time by selecting propagators, spacecraft, and stopping conditions.

We'll begin by looking at a simple example that illustrates how to create a propagator in the script. Then we'll look at the equivalent propagator setup in the GUI. In GMAT, configuring a propagator is slightly different depending on whether the user is working in the script or the GUI. Differences in the script and the GUI have been kept to a minimum, and usually occur to make each approach intuitive to the user. Tables 20.7 and 20.6 contain detailed information on the properties of a propagator that the user can configure.

12.1 Script Example

To create a propagator from the script, the user must first create a force model. Then, the user attaches this force model to a propagator. Let's start by looking at how the user creates a force model from the script by looking at an example:

```
%  Create force model named MyForceModel
Create ForceModel MyForceModel;
GMAT MyForceModel.CentralBody   = Earth;
GMAT MyForceModel.PrimaryBodies = {Earth};
GMAT MyForceModel.Gravity.Earth.Model  = JGM2;
GMAT MyForceModel.Gravity.Earth.Order  = 20;
GMAT MyForceModel.Gravity.Earth.Degree = 20;
GMAT MyForceModel.Drag          = JacchiaRoberts;
GMAT MyForceModel.Drag.F107     = 100;
GMAT MyForceModel.Drag.F107A    = 120;
GMAT MyForceModel.Drag.MagneticIndex   = 20;
GMAT MyForceModel.PointMasses   = {Sun, Luna, Jupiter};
```

The user first issues a Create command telling GMAT to create a propagator and give it a name, in this case `MyForceModel`. Issuing a `Create ForceModel` command creates a the force model that is configured

according to the default parameters detailed in Table 20.6. After creating and naming a propagator, the user can configure the propagator to include desired affects from gravitational bodies, atmospheric drag, and solar radiation pressure. The user configures the force model by setting the appropriate fields associated with a force model. The default values for the fields for a force model, along with all the possible options, units, and disallowed values are discussed in Table 20.6.

The next step in creating a propagator from the script is to create a propagator object. Below is a script example for creating a Propagator:

```
Create Propagator  RKV89JR;
GMAT RKV89JR.Type      = RungeKutta89;
GMAT RKV89JR.StepSize  = 100;
GMAT RKV89JR.Accuracy  = 1e-12;
GMAT RKV89JR.MinStep   = .001;
GMAT RKV89JR.MaxStep   = 2700;
GMAT RKV89JR.MaxStepAttempts = 50;
GMAT RKV89JR.FM        =   MyForceModel;
```

When creating a propagator the user chooses the numerical integrator and its settings, and attaches a force model that has been created previously. The default values for the fields associated with an integrator, along with all the possible options, units, and disallowed values are discussed in Table 20.6.

12.2 Creating a Propagator from the GUI

Now let's look at how to create a propagator from the GUI. To find the setup panel for a propagator in the GUI, start from the Resource Tree, and find the folder named Propagators. You can open a dialogue box to configure a new propagator by right clicking on the Propagators folder and selecting Add Propagator from the menu. The propagator dialogue box is shown in Fig. 12.1. On the left side of the propagator dialogue box we see where you can select the desired numerical integrator and configure it. On the right hand side of the panel are combo boxes and lists that allow the user to set up the force model.

Table 12.1: Fields Associated with a Force Model

Field	Options and Description
CentralBody	Default: `Earth`. Options: [`Sun`, `Mercury`, `Venus`, `Earth`, `Luna`, `Mars`, `Jupiter`, `Saturn`, `Uranus`, `Neptune`, `Pluto`]. The CentralBody field allows the user to select the origin for the propagation. All propagation occurs in the FK5 axes system, about the CentralBody chosen by the user. The CentralBody must be a gravitational body and so cannot be a LibrationPoint or other special point.
PrimaryBodies	Default: {`Earth`}. Options: [`Sun`, `Mercury`, `Venus`, `Earth`, `Luna`, `Mars`, `Jupiter`, `Saturn`, `Uranus`, `Neptune`, `Pluto`]. The PrimaryBodies field is a list of all celestial bodies that are to be modelled with a force model more complex than point mass gravity. Lists are surrounded by curly braces. A primary body can be any planet or moon not included in the PointMasses field, and for each PrimaryBody, the user can choose a drag, magnetic field, and aspherical gravity model.

Table 12.1: (continued)

Field	Options and Description
Gravity.*PrimaryBody*.Model	Default: `JGM2`. Options: [`JGM2`, `JGM3`,`EGM96`]. This field allows the user to define the source for the non-spherical gravity coefficients for a body. For example, Gravity.Earth.Model = JGM2, sets a propagator to use the JGM2 gravity coefficients for Earth. Currently there are only Earth gravity files including JGM2, JGM3, and EGM96.
Gravity.*PrimaryBody*.Degree	Default: `4`. Options: [Integer]. 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.
Gravity.*PrimaryBody*.Order	Default: `4`. Options: [Integer]. 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.
Drag	Default: None. Options: [`JachhiaRoberts`, `MSISE90`, `Exponential`]. The Drag field allows a user to specify a drag model. Currently, only one drag model can be chosen for a particular propagator and only Earth models are available.
Drag.F107	Default: `150`. Options: [Real Number]. Solar Flux
Drag.F107A	Default: `150`. Options: [Real Number]. Average Solar Flux
Drag.MagneticIndex	Default:`20`. Options: [Real Number]. Average Solar Flux
PointMasses	Default: None. Options [`Sun`, `Mercury`, `Venus`, `Earth`, `Luna`, `Mars`, `Jupiter`, `Saturn`, `Uranus`, `Neptune`, `Pluto`]. A PointMass is a planet or moon that is modelled by a point source located at its center of gravity. A PointMass body can be any planet or moon not included in the PrimaryBodies field.

Table 12.2: Fields Associated with an Integrator

Field	Options and Description
Type	Default: `RungeKutta89`. Options: [`RungeKutta89`, `RungeKutta68`, `RungeKutta56`, `PrinceDormand45`, `PrinceDormand78`, `BulirschStoer`, `AdamsBashforthMoulton`]. The `Type` field is used to set the type of numerical integrator.
StepSize	Default: 60 (sec). Options: [Real Number]. The `StepSize` is the step size of the first integration step.
Accuracy	Default: 1e-11. Options: [Real Number]. The `Accuracy` field is used to set the desired accuracy for an integration step.
MinStep	Default: .001 (sec). Options: [Real Number]. The `MinStep` field is used to set the minimum allowable step size, in seconds. `MinStep` $\leq$ `MaxStep`

Table 12.2: (continued)

Field	Options and Description
MaxStep	Default: 2700.0 (sec.). Options: [Real Number]. The `MaxStep` field is used to set the maximum allowable step size, in seconds. `MinStep` $\leq$ `MaxStep`
MaxStepAttempts	Default: 50. Options: [Integer]. The `MaxStepAttempts` field allows the user to set the number of attempts the integrator takes to meet the tolerance defined by `Accuracy`.
FM	Default: None. Options: [`RungeKutta89`, `RungeKutta68`, `RungeKutta56`, `PrinceDormand45`, `PrinceDormand78`, `BulirschStoer`, `AdamsBashforthMoulton`]. The `Type` field is used to set the type of numerical integrator.

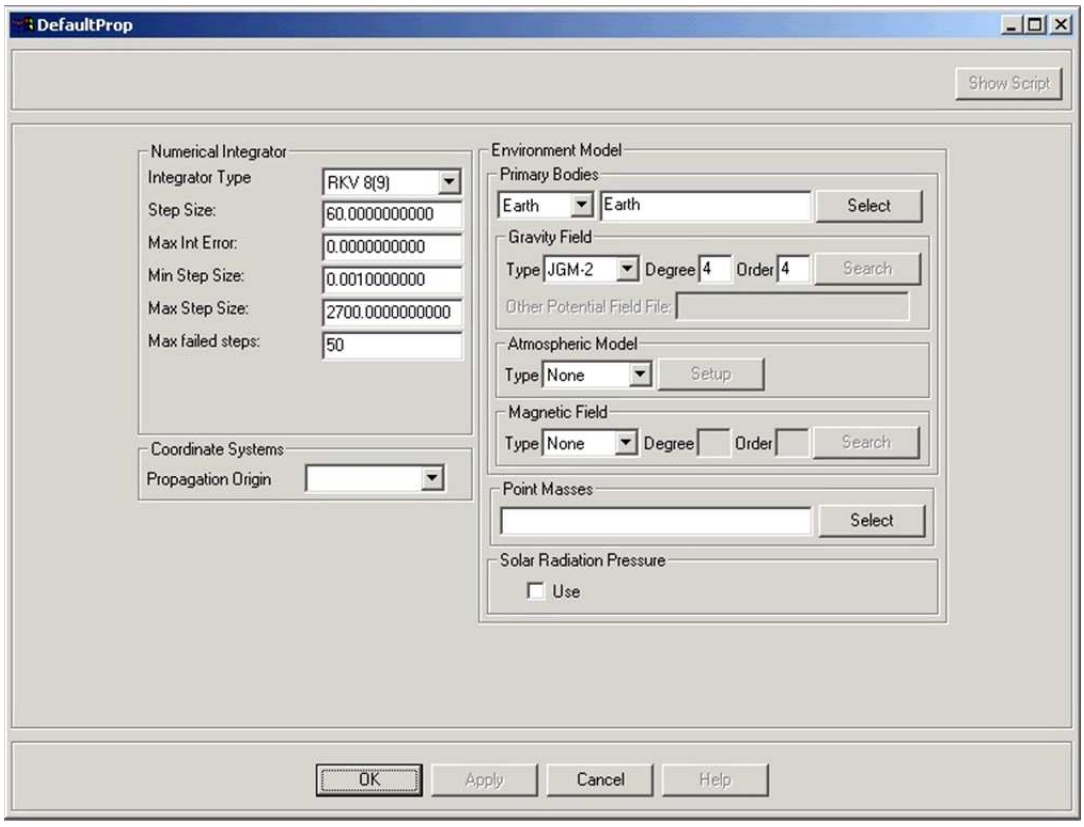

Figure 12.1: Propagator Dialogue Box

Chapter 13

Spacecraft Maneuvers

In this chapter we'll take a look at creating spacecraft maneuvers, and using them in the mission sequence. This includes both impulsive and finite maneuvers. For maneuvers, most of the information the user must supply is found in the Resource Tree. However, there are events found in the Mission Sequence that allow the user to tell GMAT when to apply an impulsive maneuver, and when to turn a finite maneuver on and off. We'll begin by learning how to create and configure maneuvers in the Resource Tree. Then we'll look at applying maneuvers in the mission sequence.

13.1 Impulsive Maneuvers

Let's begin by looking at how to create impulsive maneuvers under the resource tree. An impulsive maneuver is a maneuver that takes place over an infinitesimal length of time. The user configures an impulsive maneuver by defining the $\Delta\mathbf{v}$ vector in the desired coordinate system and vector format. A script example of how to create an impulsive maneuver is shown below.

```
%  Create an impulsive maneuver
Create ImpulsiveBurn MyBurn;
GMAT MyBurn.CoordinateSystem = EarthMJ2000Eq;
GMAT MyBurn.VectorFormat = Cartesian;
GMAT MyBurn.Element1 = 0.01;
GMAT MyBurn.Element2 = 0;
GMAT MyBurn.Element3 = 0;
```

The `CoordinateSystem` field allows the user to select which coordinate system the $\Delta\mathbf{v}$ vector is defined in. The `VectorFormat` fields allow the user to choose the between the `Cartesian`, `Spherical` and other other vector formats. The `Element` fields are used to define the maneuver direction in the chosen Coordinate System using the vector format selected. Table 13.1.

The approach to create and define an impulsive maneuver using the GUI is similar to that in the script. First, from the Resource Tree, left click on the Burn folder and select Add Impulsive Maneuver. By double right-clicking on the new maneuver, you can open its associated dialogue box and configure the same fields as are discussed above.

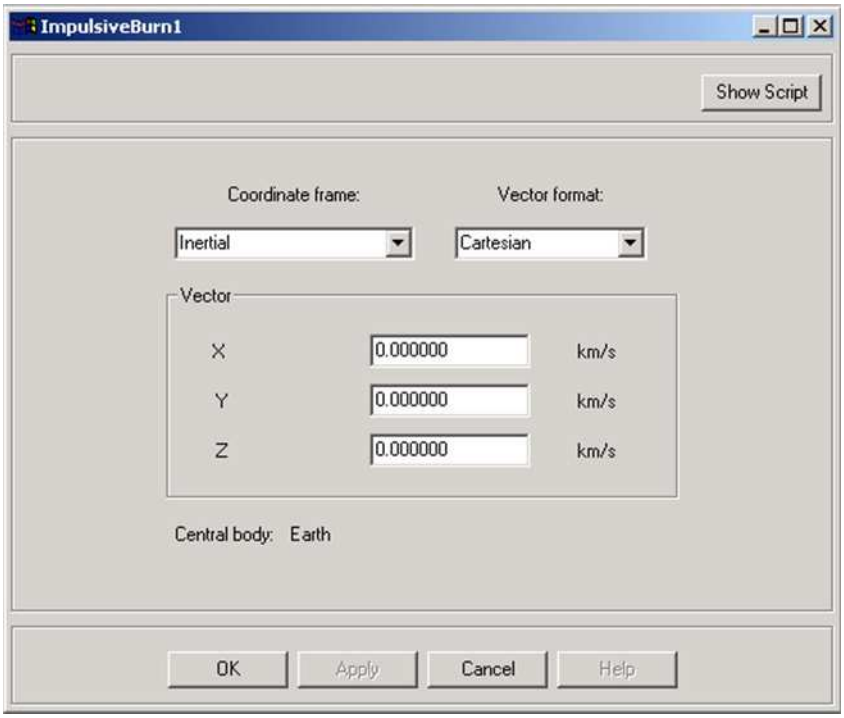

Figure 13.1: Impulsive Burn Dialogue Box

13.2 Finite Maneuvers

Now let's look at how to create an impulsive maneuver. The bulk of the work in creating a finite maneuver is performed in defining tanks and thrusters and is discussed in Ch. 11. To finish creating a finite maneuver, the user only needs to select which tanks and thrusters will be used for the maneuver. The command to turn thrusters on and off, and subsequently perform a finite maneuver are found in under the mission sequence and are discussed in a later section.

Below we see a script example that illustrates how to create and configure a finite maneuver, assuming the tanks and thrusters have already been created.

```
% Create a finite maneuver
Create FiniteBurn FiniteBurn1;
GMAT FiniteBurn1.Thrusters = {};
GMAT FiniteBurn1.Tanks = {};
GMAT FiniteBurn1.BurnScaleFactor = 1;
```

The direction of a finite maneuver is determined by the direction of the thrusters. Refer to Ch. 11 to see how to set up the direction of the thruster to achieve the desired direction for a finite maneuver.

13.3 Performing Maneuvers in the Mission Sequence

13.3.1 Impulsive Maneuvers

13.3.2 Finite Maneuvers

13.3.3 Fields associated with Spacecraf Maneuvers

Table 13.1: Fields Associated with an Impulsive Burn

Field	Options and Description
`CoordinateSystem`	Default: `EarthMJ2000Eq` . Options: [CoordinateSystemName]: The `CoordinateSystem` field allows the user to define the coordinate system for the impulsive burn.
`VectorFormat`	Default: `Cartesian` . Options: [`Cartesian`, `Spherical`]: The `VectorFormat` field allows the user to define the format of the maneuver vector.
`Element1`	Default: 0. Options: [Real Number]: The `Element1` field allows the user to define the first element of the impulsive maneuver vector. `Element1` is x if `VectorFormat` is `Cartesian`. `Element1` is the magnitude of the burn if `VectorFormat` is `spherical`.
`Element2`	Default: 0. Options: [Real Number]: The `Element2` field allows the user to define the second element of the impulsive maneuver vector. `Element2` is y if `VectorFormat` is `Cartesian`.
`Element3`	Default: 0. Options:]Real Number]: The `Element3` field allows the user to define the second element of the impulsive maneuver vector. `Element3` is z if `VectorFormat` is `Cartesian`.

Table 13.2: Fields Associated with a Finite Burn

Field	Options and Description
`CoordinateFrame`	Default: `EarthMJ2000Eq` . Options: [Coordinate System Name]:
`VectorFormat`	Default: `Cartesian` . Options: [`Cartesian`]:
`Thrusters`	Default: None . Options: [Thruster Name]:
`Tanks`	Default: None . Options: [Tank Name]:
`BurnScaleFactor`	Default: 1.0 . Options: [Real Number]:

Chapter 14

Solvers

Chapter 15

Coordinate Systems

15.1 Creating Coordinate Systems via Script and GUI

Below is how the user would create a coordinate system object using the script or the GUI. The fields are described in detail in the next section.

```
Create CoordinateSystem CoordSys;
CoordSys.Origin = ObjectName;
CoordSys.Axes   = AxesType;
CoordSys.Primary =  ObjectName ;
CoordSys.Secondary = ObjectName ;
CoordSys.Epoch.Format = Epoch;
CoordSys.XAxis        = AxisName;
CoordSys.YAxis        = AxisName;
CoordSys.ZAxis        = AxisName;
```

15.2 Definition of Fields in Coordinate System Object

15.2.1 Origin

The origin is a required field for a coordinate system object. An origin for a coordinate system can be any of the following:

1) Any Celestial Body created under the universe

2) Any Spacecraft that exists stand-alone or in formation

3) Any Barycenter that has been created under the solar system

4) Any Libration Point that has been created under the universe

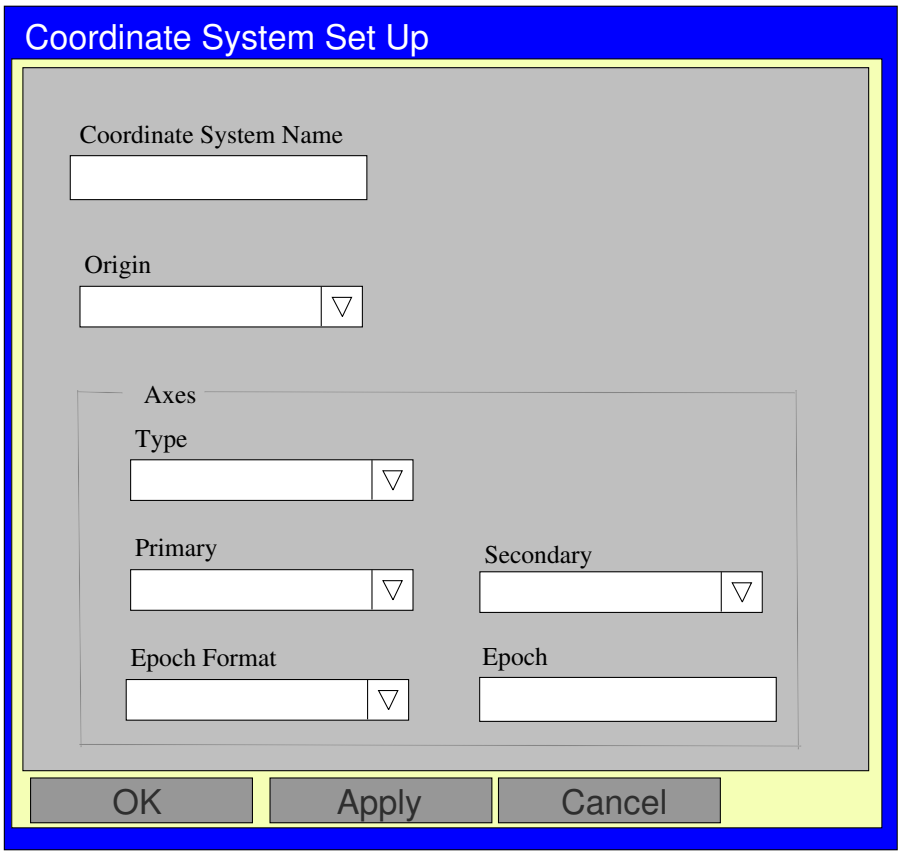

Figure 15.1: Coordinate System Dialogue Box

15.2.2 Axes Systems

The axes system is a required field for a coordinate system object. The following are allowable axes types: Equator (Section 3.2.1), MJ2000Ec (Section 3.2.2), MJ2000Eq, TOEEq (Section 3.2.3), MOEEq (Section 3.2.4), TODEq (Section 3.2.5), MODEq (Section 3.2.6), TOEEc , MOEEc, TODEc, MODEc, Fixed (Section 3.2.7), ObjectReferenced.

There are several fields in the coordinate system object that may or may not be required depending on the axes type. These are explained in the next three subsections.

15.2.3 Primary

The Primary field is optional for the following axes systems: Equator, Fixed. For these two axes types, the Primary field allows the user to specify another object, other than the origin, to be used to define the equator or fixed system. For example if AxesType = Equator, Origin = Venus, Primary = Earth, then this system would be centered on Venus but using the Earth's MJ2000 equatorial axes system.

The Primary field is required for the ObjectReferenced axes system. The relative motion that defines an ObjectReferenced axes system is defined with respect to the Primary. For example if AxesType = ObjectReferenced, Origin = Sat1, Primary = Earth, then R points from the Earth to Sat1, V is the velocity of Sat1 w/r/t the Earth, and N in the normal direction of the motion of Sat1 w/r/t the Earth. The Secondary field can be used to change which object is used to define the motion w/t/t the primary and is described in the next section.

In the GUI, if any of the axes types Equator, Fixed, or Object Referenced are chosen, the combo box should allow the user to select between any celestial body, spacecraft, Libration Point or Barycenter as a primary. The default for the Primary field should be the same as the origin.

15.2.4 Secondary

The Secondary field is optional for the axes type ObjectReferenced. The user can choose between any celestial body, spacecraft Libration Point or Barycenter as a secondary. The Secondary Object overides the origin as the object that defines the relative motion w/r/t the primary for an object referenced system. For example, if we want the origin to be Sat1 in an Earth-Moon rotating system we would choose the following. AxesType = ObjectReferenced, Origin = Sat1, Primary = Earth, Secondary = Moon.

15.2.5 Epoch

The Epoch Format and Epoch fields are active for axes types: TOEEq MOEEq TOEEc MOEEc. When these are active, the epoch format combo box needs to allow the user to choose between epoch formats similarly to how this is done on the spacecraft orbit panel. The epoch formats currently are UTCGregorian, UTCModJulian, TAIGregorian, TAIModJulian. The Epoch box is a field for the user to enter the epoch value.

15.3 Defining Parameters that are Coordinate System Dependent

General Form: Object.CoordinateSystemName.Property
Examples

```
MyVar = Sat1.EarthFixed.X;
MyVar = Sat2.Sat1LVLH.Z;
```

The following line defaults to the EarthMJ2000Eq coordinate system

```
MyVar = Sat1.X;
```

Note: See Appendix for parameters that are Coordinate System dependent.

15.4 Defining Parameters that are Central Body dependent

General Form: Object.CentralBody.Property
Examples

```
Sat1.Earth.Periapsis
 Sat1.Venus.Apoapsis
```

The following line defaults to Earth as central body

```
Sat1.Apoapsis
```

Note: See Appendix for parameters that are Central Body dependent.

15.5 Examples: Creating Coordinate Systems

15.5.1 AnyBody Inertial Equator

```
%Venus Centered, Venus Equator,  Inertial System
Create CoordinateSystem VenusEquator;
VenusEquator.Origin = Venus;
VenusEquator.Axes   = Equator;


%Venus Centered, Earth Equator, Inertial System
Create CoordinateSystem VenusEarthEquator;
VenusEarthEquator.Origin = Venus;
VenusEarthEquator.Axes   = Equator;
VenusEarthEquator.Primary   = Earth;
```

15.5.2 AnyBody Mean Ecliptic

```
% The standard MJ2000 ecliptic system
Create CoordinateSystem EarthMJ2000Ec;
EarthMJ2000Ec.Origin = Earth;
EarthMJ2000Ec.Axes = MJ2000Ec;

% The venus mean of date ecliptic system
Create CoordinateSystem VenusMODEc;
VenusMODEc.Origin = Venus;
VenusMODEc.Axes = MODEc;

% The earth mean of epoch ecliptic system
Create CoordinateSystem EarthMOEEc;
EarthMOEEc.Origin = Earth;
EarthMOEEc.Axes = MOEEc;
EarthMOEEc.Epoch.TAIModJulian = 21545.00;
```

15.5.3 AnyBody True Ecliptic

```
% The earth true of date ecliptic system
Create CoordinateSystem EarthTODEc;
EarthTODEc.Origin = Earth;
EarthTODEc.Axes = TODEc;

% The mars true of epoch ecliptic system
Create CoordinateSystem MarsTOEEc;
MarsTOEEc.Origin = Mars;
MarsTOEEc.Axes = TOEEc;
MarsTOEEc.Epoch.TAIModJulian = 21545.00;
```

15.5.4 AnyBody Mean J2000 equator

```
% The standard MJ2000 Earth Equator system
 Create CoordinateSystem EarthMJ2000Eq;
 EarthMJ2000Eq.Origin = Earth;
 EarthMJ2000Eq.Axes = MJ2000Eq;

% Venus centered Earth MJ2000 Equator system
 Create CoordinateSystem VenusMJ2000Eq ;
 VenusMJ2000Eq.Origin = Venus;
 VenusMJ2000Eq.Axes   = MJ2000Eq;
```

15.5.5 Rotating Libration Point

```
% The Earth-Moon rotating libragion point system:
%
% Here we assume we have the ability to define a libration point
% using the script or under the resource tree. The system below
% has its origin at the Earth-Moon L2 point.  The directions of R,
% -R, V, -V, N, and -N are defined by the motion of the secondary,
% the Moon, about the Primary, the Sun.

%  Create CoordinateSystem EarthMoonRLP
EarthMoonRLP.Origin = EarthMoonL2;
EarthMoonRLP.Axes = ObjectReferenced;
EarthMoonRLP.Primary = Earth;
EarthMoonRLP.Secondary = Moon;
EarthMoonRLP.XAxis = R;
EarthMoonRLP.ZAxis = N;
```

15.5.6 Any Body Fixed Equator

```
% Earth Fixed System;
Create CoordinateSystem EarthFixed;
EarthFixed.Origin = Earth;
EarthFixed.Axes = Fixed;

% Venus Fixed System
Create CoordinateSystem VenusFixed
VenusFixed.Origin = Venus\
VenusFixed.Axes   = Fixed;
```

15.5.7 Object Referenced Frames

```
%  Spacecraft (LVLH) system
Create CoordinateSystem Sat1LVLH
Sat1LVLH.Origin    = Sat1;
Sat1LVLH.Axes      = ObjectReferenced;
Sat1LVLH.Primary   = Earth;
Sat1LVLH.YAxis     = -N;
```

```
Sat1LVLH.ZAxis     = -R;

%  VNB system defined by Sat1 and Sun
Create CoordinateSystem Sat1VNB
Sat1VNB.Origin    = Sat1;
Sat1VNB.Axes      = ObjectReferenced;
Sat1VNB.Primary   = Sun;
Sat1VNB.XAxis     = V;
Sat1VNB.YAxis     = N;

%  Sun pointing coordinate system defined by Sat1 and Sun
Create CoordinateSystem Sat1SunPointing
Sat1SunPointing.Origin    = Sat1;
Sat1SunPointing.Axes      = DNB;
Sat1SunPointing.Primary   = Sun;
Sat1SunPointing.XAxis     = -R;
Sat1SunPointing.ZAxis     =  N;

%  Earth Moon Rotating system
Create CoordinateSystem EarthMoonRotating;
EarthMoonRotating.Origin    = Moon;
EarthMoonRotating.Axes      = ObjectReferenced;
EarthMoonRotating.Primary   = Earth;
EarthMoonRotating.XAxis     = R;
EarthMoonRotating.ZAxis     = N;

%  Venus Origin,  Earth Moon Rotating Axes
Create CoordinateSystem VEMRotating
VEMRotating.Origin     = Venus;
VEMRotating.Axes       = ObjectReferenced;
VEMRotating.Primary    = Earth;
VEMRotating.Seconday   = Moon;
VEMRotating.XAxis      = R;
VEMRotating.YAxis      = N;

%  Sat1 Origin , Sat2 pointing
Create CoordinateSystem Sat1Sat2Pointing;
Sat1Sat2Pointing.Origin = Sat1;
Sat1Sat2Pointing.Axes = DNB;
Sat1Sat2Pointingg.Primary   = Sat2;
Sat1Sat2Pointingg.XAxis     = -R;
Sat1Sat2Pointingg.YAxis     =  N;
```

Chapter 16

Control Flow

Chapter 17

Plots and Reports

Note: This is a first cut and will likely need to be modified or perhaps thrown out all together. I have not had time to address the View Azimuth, which is the angle of rotation around the line of site. Nor have I had time to include a way to set the field of view angle. –SPH

17.1 OpenGL Plotting

GMAT is capable of creating and drawing OpenGL plots to allow the user to visualize the motion of spacecraft and celestial bodies throughout the evolution of a mission. In this section we discuss the options a user has in setting up and viewing OpenGL plots. The user can choose many properties including the coordinate system of the OpenGL plot and the view location and direction to name a few. The script to create an OpenGL plot is shown below:

```
Create OpenGLPlot PlotName;
GMAT PlotName.TargetStatus     = [On{Off}];
GMAT PlotName.CoordinateSystem = CoordinateSystemName;
GMAT PlotName.Add              = [SpacecraftName, BodyName, ...
                                  LibrationPoint, Barycenter];
GMAT PlotName.ViewPointRef     = [ObjectName, VectorName];
GMAT PlotName.ViewPointVector  = [ObjectName, VectorName];
GMAT PlotName.ViewDirection    = [ObjectName, VectorName];
GMAT PlotName.ViewScaleFactor  = [Real Number];
GMAT PlotName.CelestialPlane   = [On {Off}];
GMAT PlotName.EquatorialPlane  = [On {Off}];
GMAT PlotName.ObjectName.Properties = PropertyValue;
```

Table 17.1 discusses the definitions of the fields for the OpenGL plot object in detail. The notation is as follows. The field name is contained in the first column. If it is bold, it is a field required by GMAT to draw an OpenGL plot and GMAT will use the default value if the user does not supply a value. In the second column, the options for the particular field are shown in square brackets, and the default value is contained in curly brackets. The options are followed by a description of the field.

17.1.1 Setting the View Location and Direction

The user specifies the view location and direction of an OpenGL plot using the ViewPointRef, ViewPointVector and ViewPointDirection fields of an OpenGL plot object. Figure 17.1 shows a graphical definition of ViewPointRef, ViewPointVector, and ViewPointDirection and how they determine the actual view location and view direction. The user can supply ViewPointRef, ViewPointVector and ViewPointDirection by either giving a vector in the format [x y z], or by specifying an object name. If a vector is given for one of the quantities, then we simply use it in its appropriate place in the computations below. If, an object is given, we must determine the vector associated with it. The remainder of this section is devoted to determining ViewPointRef, ViewPointVector and ViewPointDirection if the user specifies an object.

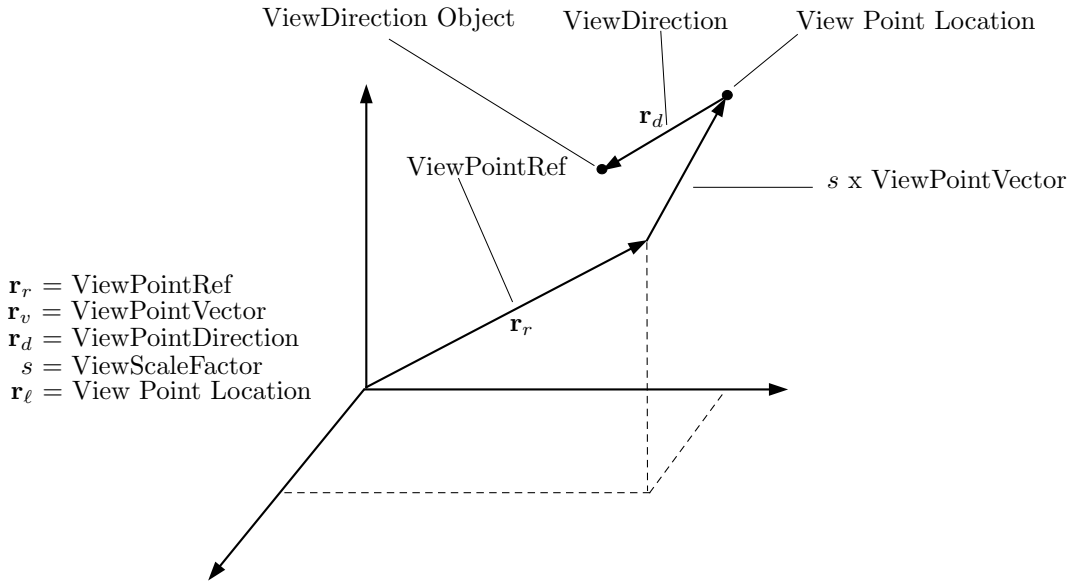

Figure 17.1: OpenGL View Definition Diagram

Let's begin by looking at ViewPointRef. ViewPointRef is the point from which ViewPointVector is measured. If an object is given for ViewPointRef , i.e. the user has the following in the script,

```
GMAT MyOpenGLPlot.CoordinateSystem = MyCoordSys;
GMAT MyOpenGLPlot.ViewPointRef = ViewRefObject;
```

then we need to determine $\mathbf{r}_r$ as illustrated in Fig. 17.1. If ViewRefObject is the same as the origin of MyCoordSys, $\mathbf{r}_r = [0\ 0\ 0]$. Otherwise. $\mathbf{r}_r$ is the cartesian position of ViewPointRef in MyCoordSys.

$$\mathbf{r}_r = \begin{bmatrix} \text{ViewRefObject.MyCoordSys.X} \\ \text{ViewRefObject.MyCoordSys.Y} \\ \text{ViewRefObject.MyCoordSys.X} \end{bmatrix} \tag{17.1}$$

ViewPointVector points from ViewPointRef ($\mathbf{r}_r$) in the direction of the of the view location. If an object is given for ViewPointVector, i.e. the user has the following in the script,

```
GMAT MyOpenGLPlot.CoordinateSystem = MyCoordSys;\\
GMAT MyOpenGLPlot.ViewPointVector  = ViewPointObject;\\
```

then we need to determine $\mathbf{r}_v$. by using the coordinate system conversion routine to calculate the following:

$$\mathbf{r}_v = \begin{bmatrix} \text{ViewPointObject.MyCoordSys.X} \\ \text{ViewPointObject.MyCoordSys.Y} \\ \text{ViewPointObject.MyCoordSys.X} \end{bmatrix} \tag{17.2}$$

We now know everything to calculate the location of the view point in the desired coordinate system. From inspection of Fig 17.1 we see that the relation is

$$\mathbf{r}_\ell = \mathbf{r}_r + s\mathbf{r}_v \tag{17.3}$$

Now that we know the view location, we need to determine the view direction, $\mathbf{r}_d$. If a vector was specified for ViewDirection, then no computations are required. However, if an object was given such as

```
GMAT MyOpenGLPlot.CoordinateSystem    = MyCoordSys;
GMAT MyOpenGLPlot.ViewPointDirection  = ViewDirectionObject;
```

then we calculate $\mathbf{r}_d$ from the following:

$$\mathbf{r}_d = \begin{bmatrix} \text{ViewDirectionObject.MyCoordSys.X} \\ \text{ViewDirectionObject.MyCoordSys.Y} \\ \text{ViewDirectionObject.MyCoordSys.X} \end{bmatrix} - \mathbf{r}_\ell \tag{17.4}$$

Note that $\mathbf{r}_d$ must not be the zero vector, [0 0 0]. If Eq. (17.4) results in the zero vector, or if the user inputs the zero vector, the we set $\mathbf{r}_d$ = [0 0 10000]

17.2 Examples: Creating OpenGL Plots

17.2.1 Earth-Based OpenGL Plots

Earth Inertial view with Moon and Spacecraft
Below is an example of an OpenGL plot that shows the Earth, Sat1, and the Moon. The view is from 400000 km above the Earth's equatorial plane on the z-axis of the EarthMJ2000Eq system. The view direction is towards the earth.

```
Create OpenGLPlot EarthInertial;
GMAT EarthInertial.TargetStatus     = Off;
GMAT EarthInertial.CoordinateSystem = EarthMJ2000Eq;
GMAT EarthInertial.Add              = Sat1;
GMAT EarthInertial.Add              = Luna;
GMAT EarthInertial.ViewPointVector  = [0 0 400000];
GMAT EarthInertial.ViewDirection    = Earth;
GMAT EarthInertial.ViewScaleFactor  = 1;
GMAT EarthInertial.EquatorialPlane  = On;
```

View of Earth from Spacecraft
Below is an example of an OpenGL plot that shows the Earth from a vantage point that is 10 km above Sat1, as Sat1 orbits Earth.

```
Create OpenGLPlot Sat1ToEarth;
```

```
GMAT Sat1ToEarth.TargetStatus     = Off;
GMAT Sat1ToEarth.CoordinateSystem = Sat1LVLH;
GMAT Sat1ToEarth.Add              = Sat1;
GMAT Sat1ToEarth.ViewPointVector  = [0 0 -10];;
GMAT Sat1ToEarth.ViewDirection    = Earth;
GMAT Sat1ToEarth.ViewScaleFactor  = 1;
GMAT Sat1ToEarth.EquatorialPlane  = Off;
```

View towards Sat1 from Sat2 in Inertial Frame
Below is an example of an OpenGL plot that shows Sat1 as viewed from Sat2 as they move in an inertial reference frame.

```
Create OpenGLPlot Sat2ToSat1;
GMAT Sat2ToSat1.TargetStatus     = Off;
GMAT Sat2ToSat1.CoordinateSystem = EarthMJ2000Eq;
GMAT Sat2ToSat1.Add              = Sat1;
GMAT Sat2ToSat1.Add              = Sat2;
GMAT Sat2ToSat1.ViewPointVector  = Sat2;
GMAT Sat2ToSat1.ViewDirection    = Sat1;
GMAT Sat2ToSat1.ViewScaleFactor  = 1;
GMAT Sat2ToSat1.EquatorialPlane  = Off;
```

View of Formation from above reference spacecraft RefSat
Below is an example of an OpenGL plot that shows a formation centered on RefSat from a perspective 10 km above RefSat and looking towards Earth.

```
Create OpenGLPlot FormationView;
GMAT FormationView.TargetStatus     = Off;
GMAT FormationView.CoordinateSystem = RefSatLVLH;
GMAT FormationView.Add              = RefSat;
GMAT FormationView.Add              = Sat1;
GMAT FormationView.Add              = Sat2;
GMAT FormationView.ViewPointRef     = RefSat;
GMAT FormationView.ViewPointVector  = [0 0 -10];
GMAT FormationView.ViewDirection    = Earth;
GMAT FormationView.ViewScaleFactor  = 1;
GMAT FormationView.EquatorialPlane  = On;
```

A similar view, but in the inertial coordinate system can be defined by

```
Create OpenGLPlot FormationView2;
GMAT FormationView2.TargetStatus     = Off;
GMAT FormationView2.CoordinateSystem = EarthMJ2000Eq;
GMAT FormationView2.Add              = RefSat;
GMAT FormationView2.Add              = Sat1;
GMAT FormationView2.Add              = Sat2;
GMAT FormationView2.ViewPointVector  = RefSat;
GMAT FormationView2.ViewDirection    = Earth;
GMAT FormationView2.ViewScaleFactor  = 1.01;
GMAT FormationView2.EquatorialPlane  = Off;
```

17.2.2 Libration Point and Deep Space OpenGL Plots

OpenGL plot of Earth-Moon Rotating System
Below is an example of an OpenGL plot that shows the Earth, Sat1, and the Moon in the Earth-Moon rotating system. The view is from 500000 km above the X-Y plane of the Earth-Moon Rotating system.

```
Create OpenGLPlot EMRotating;
GMAT EMRotating.TargetStatus     = Off;
GMAT EMRotating.CoordinateSystem = EMRotating;
GMAT EMRotating.Add              = Sat1;
GMAT EMRotating.Add              = Sun;
GMAT EMRotating.Add              = Luna;
GMAT EMRotating.ViewPointVector  = [0 0 500000];
GMAT EMRotating.ViewDirection    = Earth;
GMAT EMRotating.ViewScaleFactor  = 1;
GMAT EMRotating.EclipticPlane    = On;
```

Heliocentric ecliptic OpenGL plot
Below is an example of an OpenGL plot that shows the Sun, Earth and Venus in the Sun Centered Mean Ecliptic of J2000 system. The view point is above the Sun, looking down on the ecliptic plane.

```
Create OpenGLPlot SunMJ2000Ec;
GMAT SunMJ2000Ec.TargetStatus     = Off;
GMAT SunMJ2000Ec.CoordinateSystem = SunMJ2000Ec;
GMAT SunMJ2000Ec.Add              = Venus;
GMAT SunMJ2000Ec.Add              = Sun;
GMAT SunMJ2000Ec.ViewPointVector  = [0 0 149598000];
GMAT SunMJ2000Ec.ViewDirection    = Sun;
GMAT SunMJ2000Ec.ViewScaleFactor  = 1;
GMAT SunMJ2000Ec.EclipticPlane    = On;
```

Table 17.1: Definition of Fields Associated with OpenGL Plots

Name	Options and Description
TargetStatus	[On {Off}]: The TargetStatus field determines whether or not perturbed trajectories are plotted during a targeting sequence. When TargetStatus is set to on, the targeter iterations are shown on the plot. When TargetStatus is off, the targeter iterations are not shown on the plot.
CoordinateSystem	[CoordinateSystemName {EarthMJ2000Eq}]:
Add	[SpacecraftName CelestialBodyName LibrationPointName BarycenterName {Earth}]: 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.
Remove	[SpacecraftName CelestialBodyName LibrationPointName BarycenterName]: 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.
ViewPointRef	[SpacecraftName CelestialBodyName LibrationPointName BarycenterName {CoordinateSystem Origin}]: The ViewPointRef field is an optional field that allows the user to change the reference point from which ViewPointVector is measured. ViewPointRef defaults to the origin of the coordinate system for the plot. A ViewPointRef can be any spacecraft, celestial body, libration point, or barycenter.
ViewPointVector	[Vector SpacecraftName CelestialBodyName LibrationPointName BarycenterName {[0 0 10000]}]: The product of ViewScaleFactor and ViewPointVector field determines the view point location with respect to ViewPointRef. 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 ViewPointRef, and the vector defined by product of ViewScaleFactor and ViewPointVector in the coordinate system chosen by the user.
ViewDirection	[Vector SpacecraftName CelestialBodyName LibrationPointName BarycenterName {Earth}]: The ViewDirection field allows the user to select the direction of view in an OpenGL plot. The user can specify the view direction by choosing an object to point at such as a spacecraft,celestial body, libration point,or barycenter. Alternatively, the user can specify a vector of the form [x y z]. If the user specification of ViewDirection, ViewPointRef, and ViewPointVec, results in a zero vector in Eq .(17.4), GMAT uses [0 0 10000] for ViewDirection.
ViewScaleFactor	[Real Number {1}]: The ViewScaleFactor field scales ViewPointVector before adding it to ViewPointRef. The ViewScaleFactor allows the user to back away from an object to fit in the field of view.
CelestialPlane	[On {Off}]: The CelestialPlane field allows the user to turn the ecliptic plane grid on or off.
EquatorialPlane	[On {Off}]: The EquatorialPlane field allows the user to turn the ecliptic plane grid on or off.
ObjectName.Properties	[ObjectName.Property]: For future GMAT versions. The user can change options for different objects added to a plot using the ObjectName.Property field. For example, if the user has added the moon to a plot, and also wants to see the moons orbit, we would say "PlotName.Luna.Orbit = On"

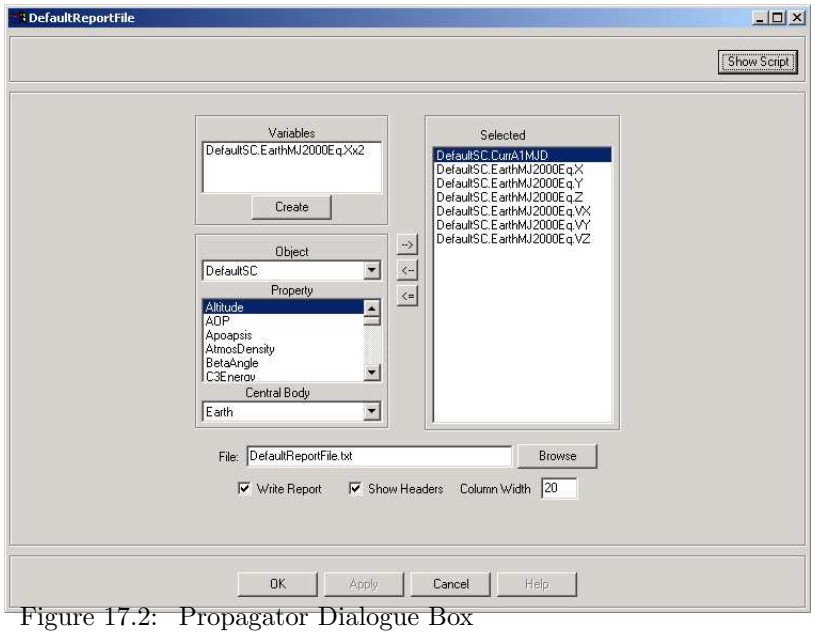

Figure 17.2: Propagator Dialogue Box

17.3 Implementation in OpenGL

Let's begin by assuming we know, $\mathbf{r}_\ell$, which is the location of the view point in the desired reference frame. Let's also assume we know $\mathbf{r}_v$, the view direction in the desired reference frame. We need to convert these vectors to the form that OpenGL uses to define the viewer location and direction. First, let's define

$$\hat{\mathbf{z}} = [0 \quad 0 \quad 1]^T \tag{17.5}$$

In OpenGL, we first perform a translation to achieve the correct viewpoint location. This is performed in the negative $\mathbf{r}_\ell$ direction because in OpenGL, you essentially define everything with respect to the origin, and then move the coordinate system away from the origin to achieve the desired view. So, using the high-level function *glTranslatef*, we perform a translation of $-\mathbf{r}_\ell$, or

$$\text{glTranslatef}(-r_\ell(1), -r_\ell(2), -r_\ell(3))$$

Next we need to rotate the system so that we achieve the desired view direction. OpenGL assumes the default view direction is in the negative z direction. So, we need to rotate the system so that the negative z-axis is aligned with $\mathbf{r}_v$. We can do this using the high-level function *glRotatef*. The inputs to glRotatef are the angle of the rotation, and the axis which we would like to rotate about. We can calculate the axis of rotation, $\mathbf{a}$, using

$$\mathbf{a} = -\mathbf{z} \times \mathbf{r}_v = [r_v(2) \quad -r_v(1) \quad 0]^T$$

We can calculate the rotation angle, θ, using the following:

$$\theta = \cos^{-1}\left(\frac{-\hat{\mathbf{z}} \cdot \mathbf{r_v}}{r_v}\right) = \cos^{-1}\left(-\frac{r_v(3)}{r_v}\right)$$

where

$$r_v = \|\mathbf{r}_v\|$$

Now, we can call *glRotatef* using

$$\text{glRotatef}(\theta, r_v(2), -r_v(1), 0)$$

Chapter 18

Functions

18.1 User Defined GMAT Functions

GMAT has the capability to allow the user to define custom functions. User defined functions are useful for isolating repetitive calculations that must be performed in the mission sequence. GMAT is designed to allow the user to create functions in a general way by defining a list of input arguments, a series of operations and events to be performed on the input arguments, and a list of output arguments to be passed back to the main mission sequence.

One way to think about user defined GMAT functions is to consider them as mini mission sequences with their own set of resources and events. The idea is to isolate small mission analysis problems, such as Lambert's problem, into mini-missions that GMAT can solve as a separate process, and then bring the desired output back into the main mission sequence.

The input and output arguments to a user function can be variables, arrays, strings, and objects including spacecraft, propagators, solvers etc. Inside a user-defined function the user can create and define local variables, arrays, strings, and objects. All input and locally created data can be used in mathematical expressions containing +, -, x, / , sin, cos, tan, asin, acos, atan, atan2, sqrt, and ∧ (raise to power) as well as events such as Propagate, Solve, Control Flow etc. Changes to the input arguments are local only to the user-function unless the input argument is also contained in the output argument list. Finally, other user defined functions can be called from within a user defined function.

User Defined Function Characteristics

- The input argument list can contain variables, arrays, strings, and objects (spacecraft, propagators, solvers etc).
- The output argument list can contain variables, arrays, strings, and objects (spacecraft, propagators, solvers etc).
- The user can create local variables, arrays, strings and objects inside of user functions.
- Changes to inputs are local to the function, unless the variable is included in the output argument list.
- The user can perform the following mathematical operations on variables, arrays, and spacecraft properties: +, -, x, / , sin, cos, tan, asin, acos, atan, atan2 sqrt, ^(raise to a power).
- The can make calls to other GMAT functions or matlab from within a user-function.
- The user can perform any GMAT command using input and locally created data: Propagate, Target, If, While, For, Solve etc.

18.1.1 Creating User Defined GMAT Functions

User-functions can be used in mission sequences run from either the script or the GMAT GUI. Currently however, GMAT functions can only be defined using the GMAT script language. From the GUI, user-functions can be created by going to the Resource Tree, right-clicking on the GMAT Functions folder, and selecting Create New. This brings up a script window where a user-defined function can be entered. After the function is written, it is named, and saved to disk.

The syntax for defining a GMAT function is easily described with an example below.

```
GMATFunction Velocity = Lambert(Sat, RKVHighEarth, TOF, R_Desired)

%---------------------------------------------------------------------------
%--------------------------Declarations-------------------------------------
%---------------------------------------------------------------------------
%  Validate Input.  Here GMAT checks to make sure the inputs are of the
%  appropriate type.  This catches problems when the user accidentally passes
%  in the wrong data type.
Validate Sat(Spacecraft);
Validate RKVHighEarth(Propagator) ;
Validate TOF(Variable);
Validate  R_Desired(Array);

%  Define Output
Create Array Velocity[3,1];

%  Define Internal variables
Create Maneuver dV
Create DifferentialCorrector DC

%  External Function References:  None

%---------------------------------------------------------------------------
%---------------------------------Function----------------------------------
%---------------------------------------------------------------------------

Target DC

     Vary DC(dV.MJ2000Eq.X = .01);
     Vary DC(dV.MJ2000Eq.Y = .01);
     Vary DC(dV.MJ2000Eq.Z = .01);

     Manuever  dV(Sat);

     Propagate RKVHighEarth(Sat, {Sat.ElapsedDays = TOF});

     Achieve DC(Sat.MJ2000Eq.X = R_Desired(1,1));
     Achieve DC(Sat.MJ2000Eq.Y = R_Desired(2,1));
     Achieve DC(Sat.MJ2000Eq.Z = R_Desired(3,1));

EndTarget

GMAT Velocity(1,1) = dV.MJ2000Eq.X;
GMAT Velocity(2,1) = dV.MJ2000Eq.Y;
GMAT Velocity(3,1) = dV.MJ2000Eq.Z;
```

18.1.2 Using User-Defined GMAT Functions in the Mission Sequence

18.2 Internal GMAT Functions

Chapter 19

Events

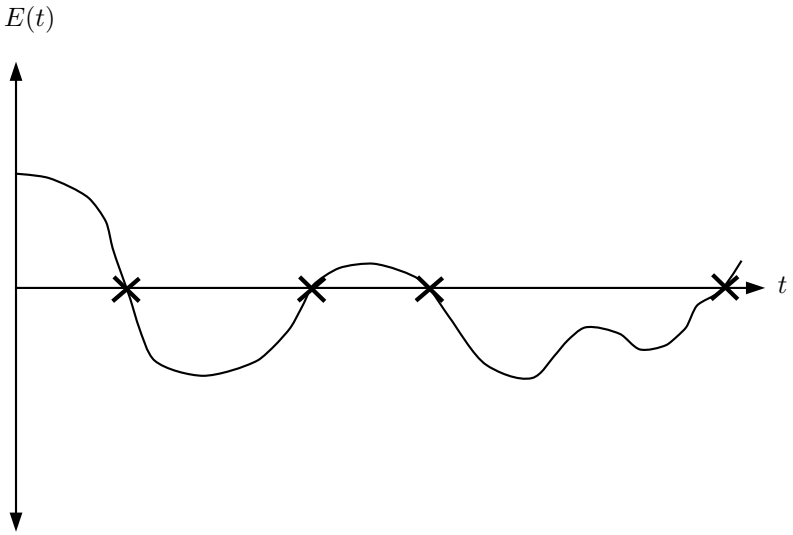

Figure 19.1: Generic Event Function

Chapter 20

Object Fields: Quick Look-up Tables

20.1 Spacecraft Fields

Table 20.1: Fields Associated with a Spacecraft Orbit State

Field	Options and Description
StateType	Default: Cartesian. Options: [Cartesian, Keplerian, ModifiedKeplerian, SphericalAZFPA, SphericalRADEC].
	Fields associated with Cartesian state.
X	Default: 7100. Options: [Real Number]: X is the x-component of the Spacecraft state in the coordinate system chosen in the Spacecraft CoordinateSystem field. Units: km.
Y	Default: 0. Options: [Real Number]: Y is the y-component of the Spacecraft state in the coordinate system chosen in the Spacecraft CoordinateSystem field. Units: km.
Z	Default: 1300. Options: [Real Number]: Z is the z-component of the Spacecraft state in the coordinate system chosen in the Spacecraft CoordinateSystem field. Units: km.
VX	Default: 0. Options: [Real Number]: VX is the x-component of the Spacecraft velocity in the coordinate system chosen in the Spacecraft CoordinateSystem field. Units: km.
VY	Default: 7.35. Options: [Real Number]: VY is the y-component of the Spacecraft velocity in the coordinate system chosen in the Spacecraft CoordinateSystem field. Units: km.
VZ	Default: 1.0. Options: [Real Number]: VZ is the z-component of the Spacecraft velocity in the coordinate system chosen in the Spacecraft CoordinateSystem field. Units: km.
	Fields associated with Keplerian state.
SMA	Default: [Real Number {7191.938817629}]: The SMA field is the spacecraft orbit's osculating Keplerian semimajor axis in km. SMA must be be strictly greater than or less than zero.

Table 20.1: (Fields Associated with a Spacecraft Orbit State. continued)

Field	Options and Description
ECC	[Real Number {0.024549749}]: The ECC field is the spacecraft orbit's osculating eccentricity. ECC must be greater than zero.
INC	[Real Number {12.850080057}]: The INC field is the spacecraft orbit's osculating inclination, in degrees, w/r/t to the selected coordinate system.
AOP	[Real Number {314.190551536}]: The AOP field is the spacecraft orbit's osculating argument of periapsis, in degrees, w/r/t to the selected coordinate system.
RAAN	[Real Number {306.614802195}]: The RAAN field is the spacecraft orbit's osculating right ascension of the ascending node, in degrees, w/r/t to the selected coordinate system.
TA	[Real Number {306.614802195}]: The TA field is the spacecraft orbit's osculating true anomaly, in degrees.
	Fields associated with ModifiedKeplerian state.
RadApo	Default: . Options: . The RadApo field is the spacecraft orbit's osculating radius of apoapsis. RadApo must be be strictly greater than or less than zero. Units: km.
RadPer	Default: Options: The RadPer field is the spacecraft orbit's osculating radius of periapsis. RadPer must be be greater than zero. Units: km.
	Fields associated with SphericalAZFPA state.
RMAG	Default: [Real Number {7191.938817629}]: The SMA field is the spacecraft orbit's osculating Keplerian semimajor axis in km. SMA must be be strictly greater than or less than zero.
RA	[Real Number {0.024549749}]: The ECC field is the spacecraft orbit's osculating eccentricity. ECC must be greater than zero.
DEC	[Real Number {12.850080057}]: The INC field is the spacecraft orbit's osculating inclination, in degrees, w/r/t to the selected coordinate system.
VMAG	[Real Number {314.190551536}]: The AOP field is the spacecraft orbit's osculating argument of periapsis, in degrees, w/r/t to the selected coordinate system.
AZI	[Real Number {306.614802195}]: The RAAN field is the spacecraft orbit's osculating right ascension of the ascending node, in degrees, w/r/t to the selected coordinate system.
FPA	[Real Number {306.614802195}]: The TA field is the spacecraft orbit's osculating true anomaly, in degrees.
	Fields associated with SphericalRADEC state.
SMA	Default: [Real Number {7191.938817629}]: The SMA field is the spacecraft orbit's osculating Keplerian semimajor axis in km. SMA must be be strictly greater than or less than zero.
ECC	[Real Number {0.024549749}]: The ECC field is the spacecraft orbit's osculating eccentricity. ECC must be greater than zero.

Table 20.1: (Fields Associated with a Spacecraft Orbit State. continued)

Field	Options and Description
INC	[Real Number {12.850080057}]: The INC field is the spacecraft orbit's osculating inclination, in degrees, w/r/t to the selected coordinate system.
AOP	[Real Number {314.190551536}]: The AOP field is the spacecraft orbit's osculating argument of periapsis, in degrees, w/r/t to the selected coordinate system.
RAAN	[Real Number {306.614802195}]: The RAAN field is the spacecraft orbit's osculating right ascension of the ascending node, in degrees, w/r/t to the selected coordinate system.
TA	[Real Number {306.614802195}]: The TA field is the spacecraft orbit's osculating true anomaly, in degrees.

Table 20.2: Fields Associated with a Spacecraft Epoch

Field	Options and Description
TAIGregorian	[Gregorian Date {01 Jan 2000 12:00:00.000}]: The TAIGregorian field allows the user to enter the initial spacecraft epoch in the TAI time system using a Gregorian Date format.
TAIModJulian	[Real Number {21545.000000000}]: The TAIModJulian field allows the user to enter the initial spacecraft epoch in the TAI time system using a Modified Julian format.
UTCGregorian	[Gregorian Date {01 Jan 2000 11:59:27.966}]: The UTCGregorian field allows the user to enter the initial spacecraft epoch in the UTC time system using a Gregorian Date format.
UTCModJulian	[Real Number {21544.999629236}]: The UTCModJulian field allows the user to enter the initial spacecraft epoch in the UTC time system using a Modified Julian format.

Table 20.3: Fields Associated with a Spacecraft Ballistic and Mass Properties

Field	Options and Description
Cd	Default: 2.2. Options: [Real Number]: Cd is the spacecraft's drag coefficient. Cd must be greater than 0. Units: Dimensionless.
Cr	Default: 2.2. Options: [Real Number]: Cr is the spacecraft's coefficient of reflectivity. Cr must be greater than 0. Units: Dimensionless.
DragArea	Default: 15.0. Options: [Real Number]: The DragArea is the area of the spacecraft that is used in calculating atmospheric drag. DragArea must be greater than 0. Units: m^2

Table 20.3: (continued)

Field	Options and Description
SRPArea	Default: 1.0. Options: [Real Number]: The SRPArea is the area of the spacecraft that is used in calculating the force due to solar radiation pressure. SRPArea must be greater than 0. Units: m^2
DryMass	Default: 850.0. Options: [Real Number]: The DryMass is the mass of the spacecraft without the mass of tanks and fuel. DryMass must be greater than 0. Units: kg

Table 20.4: Fields Associated with a Spacecraft Tank

Field	Options and Description
FuelMass	Default: 756. Options: [Real Number]: The FuelMass field is the mass of fuel in the tank. Units: kg.
Pressure	Default: 1500. Options: [Real Number]: The Pressure field is the pressure of the fuel in the tank. Units: kPa.
Temperature	Default: 20. Options: [Real Number]: The Temperature field is the temperature of the fuel in the tank. Units: C.
RefTemperature	Default: 20. Options: [Real Number]: RefTemperature Units: C.
Volume	Default: 0.75. Options: The Volume field is the volume of the tank. [Real Number]:
FuelDensity	Default: 1260. Options: [Real Number]:
PressureRegulated	Default: true. Options: [true false]:

Table 20.5: Fields Associated with a Spacecraft Thruster

Field	Options and Description
CoordinateSystem	Default: EarthMJ2000Eq. Options: [Coordinate System Name]: 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.
X_Direction	Default: 1. Options: [Real Number]: X_Direction, divided by the RSS of the three direction vectors, forms the x direction of the spacecraft thrust vector direction.
Y_Direction	Default: 0. Options: [Real Number]: Y_Direction, divided by the RSS of the three direction vectors, forms the y direction of the spacecraft thrust vector direction

Table 20.5: (continued)

Field	Options and Description
`Z_Direction`	Default: 0. Options: [Real Number]: `Z_Direction`, divided by the RSS of the three direction vectors, forms the z direction of the spacecraft thrust vector direction
`ThrustScaleFactor`	Default: 1. Options: [Real Number]: `ThrustScaleFactor` is a scale factor that is multiplied by the thrust vector for a given thruster, before the thrust vector is added into the total accleration. `ThrustScaleFactor` must be greater than 0. Units: None.
`Tank`	Default: None. Options: [Tank Name]: The `Tank` field specifies which tank the thruster draws propellant from.
`C1`	Default: 500. Options: [Real]: Thruster coefficient. Units: N
`C2`	Default: 0. Options: [Real]: Thruster coefficient. Units: N/kPa.
`C3`	Default: 0. Options: [Real]: Thruster coefficient. Units: N/kPa^2
`C4`	Default: 0. Options: [Real]: Thruster coefficient. Units: N/kPa^{C5}.
`C5`	Default: 0. Options: [Real]: Thruster coefficient. Units: None
`C6`	Default: 0. Options: [Real]: Thruster coefficient. Units: N/kPa^{C7}.
`C7`	Default: 0. Options: [Real]: Thruster coefficient. Units: None
`C8`	Default: 0. Options: [Real]: Thruster coefficient. Units: N/kPa^{C9}.
`C9`	Default: 0. Options: [Real]: Thruster coefficient. Units: None
`C10`	Default: 0. Options: [Real]: Thruster coefficient. Units: N.
`C11`	Default: 1. Options: [Real]: Thruster coefficient. Units: None
`C12`	Default: 0. Options: [Real]: Thruster coefficient. Units: 1/kPa.
`C13`	Default: 0. Options: [Real]: Thruster coefficient. Units: None.
`C14`	Default: 0. Options: [Real]: Thruster coefficient. Units 1/kPa.
`K1`	Default: 2150. Options: [Real]: Thruster coefficient. Units: m/s
`K2`	Default: 0. Options: [Real]: Thruster coefficient. Units: m/(s· kPa).
`K3`	Default: 0. Options: [Real]: Thruster coefficient. Units: m/(s· kPa^2)
`K4`	Default: 0. Options: [Real]: Thruster coefficient. Units: m/(s· kPa^{K5}).
`K5`	Default: 0. Options: [Real]: Thruster coefficient. Units: None
`K6`	Default: 0. Options: [Real]: Thruster coefficient. Units: m/(s· kPa^{K7}).
`K7`	Default: 0. Options: [Real]: Thruster coefficient. Units: None

Table 20.5: (continued)

Field	Options and Description
K8	Default: 0. Options: [Real]: Thruster coefficient. Units: m/(s· kPa^{K9}.
K9	Default: 0. Options: [Real]: Thruster coefficient. Units: None
K10	Default: 0. Options: [Real]: Thruster coefficient. Units: m/s.
K11	Default: 1. Options: [Real]: Thruster coefficient. Units: None
K12	Default: 0. Options: [Real]: Thruster coefficient. Units: 1/kPa.
K13	Default: 0. Options: [Real]: Thruster coefficient. Units: None.
K14	Default: 0. Options: [Real]: Thruster coefficient. Units 1/kPa.

20.2 Propagator Fields

Table 20.6: Fields Associated with a Force Model

Field	Options and Description
CentralBody	Default: Earth. Options: [Sun, Mercury, Venus, Earth, Luna, Mars, Jupiter, Saturn, Uranus, Neptune, Pluto]. The CentralBody field allows the user to select the origin for the propagation. All propagation occurs in the FK5 axes system, about the CentralBody chosen by the user. The CentralBody must be a gravitational body and so cannot be a LibrationPoint or other special point.
PrimaryBodies	Default: {Earth}. Options: [Sun, Mercury, Venus, Earth, Luna, Mars, Jupiter, Saturn, Uranus, Neptune, Pluto]. The PrimaryBodies field is a list of all celestial bodies that are to be modelled with a force model more complex than point mass gravity. Lists are surrounded by curly braces. A primary body can be any planet or moon not included in the PointMasses field, and for each PrimaryBody, the user can choose a drag, magnetic field, and aspherical gravity model.
Gravity.*PrimaryBody*.Model	Default: JGM2. Options: [JGM2, JGM3,EGM96]. This field allows the user to define the source for the non-spherical gravity coefficients for a body. For example, Gravity.Earth.Model = JGM2, sets a propagator to use the JGM2 gravity coefficients for Earth. Currently there are only Earth gravity files including JGM2, JGM3, and EGM96.
Gravity.*PrimaryBody*.Degree	Default: 4. Options: [Integer]. 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.

Table 20.6: (continued)

Field	Options and Description
Gravity.*PrimaryBody*.Order	Default: 4. Options: [Integer]. 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.
Drag	Default: None. Options: [JachhiaRoberts, MSISE90, Exponential]. The Drag field allows a user to specify a drag model. Currently, only one drag model can be chosen for a particular propagator and only Earth models are available.
Drag.F107	Default: 150. Options: [Real Number]. Solar Flux
Drag.F107A	Default: 150. Options: [Real Number]. Average Solar Flux
Drag.MagneticIndex	Default:20. Options: [Real Number]. Average Solar Flux
PointMasses	Default: None. Options [Sun, Mercury, Venus, Earth, Luna, Mars, Jupiter, Saturn, Uranus, Neptune, Pluto]. A PointMass is a planet or moon that is modelled by a point source located at its center of gravity. A PointMass body can be any planet or moon not included in the PrimaryBodies field.

Table 20.7: Fields Associated with an Integrator

Field	Options and Description
Type	Default: RungeKutta89. Options: [RungeKutta89, RungeKutta68, RungeKutta56, PrinceDormand45, PrinceDormand78, BulirschStoer, AdamsBashforthMoulton]. The Type field is used to set the type of numerical integrator.
StepSize	Default: 60 (sec). Options: [Real Number]. The StepSize is the step size of the first integration step.
Accuracy	Default: 1e-11. Options: [Real Number]. The Accuracy field is used to set the desired accuracy for an integration step.
MinStep	Default: .001 (sec). Options: [Real Number]. The MinStep field is used to set the minimum allowable step size, in seconds. MinStep $\leq$ MaxStep
MaxStep	Default: 2700.0 (sec.). Options: [Real Number]. The MaxStep field is used to set the maximum allowable step size, in seconds. MinStep $\leq$ MaxStep
MaxStepAttempts	Default: 50. Options: [Integer]. The MaxStepAttempts field allows the user to set the number of attempts the integrator takes to meet the tolerance defined by Accuracy.
FM	Default: None. Options: [RungeKutta89, RungeKutta68, RungeKutta56, PrinceDormand45, PrinceDormand78, BulirschStoer, AdamsBashforthMoulton]. The Type field is used to set the type of numerical integrator.

20.3 Maneuvers

Table 20.8: Fields Associated with an Impulsive Burn

Field	Options and Description
`CoordinateSystem`	Default: `EarthMJ2000Eq` . Options: [CoordinateSystemName]: The `CoordinateSystem` field allows the user to define the coordinate system for the impulsive burn.
`VectorFormat`	Default: `Cartesian` . Options: [`Cartesian`, `Spherical`]: The `VectorFormat` field allows the user to define the format of the maneuver vector.
`Element1`	Default: 0. Options: [Real Number]: The `Element1` field allows the user to define the first element of the impulsive maneuver vector. `Element1` is x if `VectorFormat` is `Cartesian`. `Element1` is the magnitude of the burn if `VectorFormat` is `spherical`.
`Element2`	Default: 0. Options: [Real Number]: The `Element2` field allows the user to define the second element of the impulsive maneuver vector. `Element2` is y if `VectorFormat` is `Cartesian`.
`Element3`	Default: 0. Options:]Real Number]: The `Element3` field allows the user to define the second element of the impulsive maneuver vector. `Element3` is z if `VectorFormat` is `Cartesian`.

Table 20.9: Fields Associated with a Finite Burn

Field	Options and Description
`CoordinateFrame`	Default: `EarthMJ2000Eq` . Options: [Coordinate System Name]:
`VectorFormat`	Default: `Cartesian` . Options: [`Cartesian`]:
`Thrusters`	Default: None . Options: [Thruster Name]:
`Tanks`	Default: None . Options: [Tank Name]:
`BurnScaleFactor`	Default: 1.0 . Options: [Real Number]:

Chapter 21

Tutorial Scripts

21.1 Integrators

```
%  Integrator Script Example

%  This script  demonstrates how to choose different numerical integrators
%  in the propagator set up

% ----------------------------------------------------------------------
% -------------------------- Create Objects ----------------------------
% ----------------------------------------------------------------------

%----------------------------Create the Spacecraft-------------------------
% Create Sat1 and define its orbit
Create Spacecraft Sat1;
GMAT Sat1.Epoch.UTCGregorian  = 04 Jan 2003 00:00:00.000;
GMAT Sat1.StateType = Cartesian;
GMAT Sat1.X     = -20320.831700;
GMAT Sat1.Y     = 7859.549800 ;
GMAT Sat1.Z     = 4411.786100;
GMAT Sat1.VX    = 1.491571300;
GMAT Sat1.VY    = -2.771848000;
GMAT Sat1.VZ    = 1.649569600;
GMAT Sat1.Cd    = 2.2;
GMAT Sat1.Cr    = 1.2;
GMAT Sat1.DragArea = 4;
GMAT Sat1.SRPArea  = 4;
GMAT Sat1.DryMass  = 100;

%----------------------------Create ForceModels---------------------------
%  Define Force Model with point mass only
Create ForceModel PointMass;
GMAT PointMass.PrimaryBodies        =  {Earth};
GMAT PointMass.Gravity.Earth.Model  = JGM2;
GMAT PointMass.Gravity.Earth.Degree = 0;
GMAT PointMass.Gravity.Earth.Order  = 0;

%----------------------------Create Propagators---------------------------
```

```
%Create a Runge Kutta Fehlburg (56) Propagator
Create Propagator RungeKutta56;
GMAT  RungeKutta56.Type     = RungeKuttaFehlberg56;
GMAT  RungeKutta56.StepSize = 30;;
GMAT  RungeKutta56.Accuracy = 1e-12;
GMAT  RungeKutta56.MinStep  = 1e-5;
GMAT  RungeKutta56.MaxStep  = 1000;
GMAT  RungeKutta56.FM       =  PointMass;

%Create a Runge Kutta Nystrom (68) Propagator
Create Propagator RungeKutta68;
GMAT  RungeKutta68.Type     = DormandElMikkawyPrince68;
GMAT  RungeKutta68.StepSize = 30;;
GMAT  RungeKutta68.Accuracy = 1e-12;
GMAT  RungeKutta68.MinStep  = 1e-5;
GMAT  RungeKutta68.MaxStep  = 1000;
GMAT  RungeKutta68.FM       =  PointMass;

% Create a Runge Kutta Verner (89) Propagator
Create Propagator RungeKutta89;
GMAT  RungeKutta89.Type     = RungeKutta89;
GMAT  RungeKutta89.StepSize = 30;;
GMAT  RungeKutta89.Accuracy = 1e-12;
GMAT  RungeKutta89.MinStep  = 1e-5;
GMAT  RungeKutta89.MaxStep  = 1000;
GMAT  RungeKutta89.FM       =  PointMass;

% Create a Prince Dormand (45) Propagator
Create Propagator PrinceDormand45;
GMAT  PrinceDormand45.Type     = PrinceDormand45;
GMAT  PrinceDormand45.StepSize = 30;;
GMAT  PrinceDormand45.Accuracy = 1e-12;
GMAT  PrinceDormand45.MinStep  = 1e-5;
GMAT  PrinceDormand45.MaxStep  = 1000;
GMAT  PrinceDormand45.FM       =  PointMass;

% Create a Prince Dormand (78) Propagator
Create Propagator PrinceDormand78;
GMAT  PrinceDormand78.Type     = PrinceDormand78;
GMAT  PrinceDormand78.StepSize = 30;;
GMAT  PrinceDormand78.Accuracy = 1e-12;
GMAT  PrinceDormand78.MinStep  = 1e-5;
GMAT  PrinceDormand78.MaxStep  = 1000;
GMAT  PrinceDormand78.FM       =  PointMass;

% Create a Bulirsch Stoer Propagator
Create Propagator BulirschStoer;
GMAT  BulirschStoer.Type     = BulirschStoer;
GMAT  BulirschStoer.StepSize = 30;;
GMAT  BulirschStoer.Accuracy = 1e-12;
GMAT  BulirschStoer.MinStep  = 1e-5;
GMAT  BulirschStoer.MaxStep  = 1000;
GMAT  BulirschStoer.FM       =  PointMass;

% Create a Adams Bashforth Moulton Propagator
Create Propagator AdamsBashfourthMoulton;
GMAT  AdamsBashfourthMoulton.Type     = AdamsBashforthMoulton;
```

```
GMAT  AdamsBashfourthMoulton.StepSize = 30;;
GMAT  AdamsBashfourthMoulton.Accuracy = 1e-12;
GMAT  AdamsBashfourthMoulton.MinStep  = 1e-5;
GMAT  AdamsBashfourthMoulton.MaxStep  = 1000;
GMAT  AdamsBashfourthMoulton.FM       =  PointMass;

%----------------------------Create OpenGL Plot_---------------------------
Create OpenGLPlot GLPlot;
GMAT GLPlot.Add = Sat1;

% ---------------------------------------------------------------------------
% --------------------------  Begin Mission Sequence
% ---------------------------------------------------------------------------
%  Propagate using each of the propagators created above

Propagate  RungeKutta56(Sat1, {Sat1.ElapsedDays = 4.0});

Propagate  RungeKutta68(Sat1, {Sat1.ElapsedDays = 4.0});

Propagate  RungeKutta89(Sat1, {Sat1.ElapsedDays = 4.0});

Propagate  PrinceDormand45(Sat1, {Sat1.ElapsedDays = 4.0});

Propagate  PrinceDormand78(Sat1, {Sat1.ElapsedDays = 4.0});

Propagate  BulirschStoer(Sat1, {Sat1.ElapsedDays = 4.0});

Propagate  AdamsBashfourthMoulton(Sat1, {Sat1.ElapsedDays = 4.0});
```

21.2 Force Models

```
%  Force Models Script Example

%  This script demonstrates how to set up different force models in the
%  propagator setup

% ---------------------------------------------------------------------------
% --------------------------- Create Objects ------------------------------
% ---------------------------------------------------------------------------

%----------------------------Create the Spacecraft----------------------
% Create Sat1 and define its orbit
Create Spacecraft Sat1;
GMAT Sat1.Epoch.UTCGregorian  = 04 Jan 2003 00:00:00.000;
GMAT Sat1.StateType = Cartesian;
GMAT Sat1.X     = -20320.831700;
GMAT Sat1.Y     = 7859.549800 ;
GMAT Sat1.Z     = 4411.786100;
GMAT Sat1.VX    = 1.491571300;
GMAT Sat1.VY    = -2.771848000;
GMAT Sat1.VZ    = 1.649569600;
GMAT Sat1.Cd    = 2.2;
```

```
GMAT Sat1.Cr    = 1.2;
GMAT Sat1.DragArea = 4;
GMAT Sat1.SRPArea  = 4;
GMAT Sat1.DryMass  = 100;

%---------------------------Create ForceModels----------------------

%  Define Force Model with point mass only
Create ForceModel PointMass;
GMAT PointMass.PrimaryBodies        = {Earth};
GMAT PointMass.Gravity.Earth.Model  = JGM2;
GMAT PointMass.Gravity.Earth.Degree = 0;
GMAT PointMass.Gravity.Earth.Order  = 0;

%  Define Force Model with third bodies
Create ForceModel ThirdBodies;
GMAT ThirdBodies.PrimaryBodies        = {Earth};
GMAT ThirdBodies.Gravity.Earth.Model  = JGM2;
GMAT ThirdBodies.Gravity.Earth.Degree = 0;
GMAT ThirdBodies.Gravity.Earth.Order  = 0;
GMAT ThirdBodies.PointMasses   = { Sun, Luna,  Venus,...
                           Mars, Jupiter, Saturn, Uranus, Neptune};

%  Define Force Model with 12x12 gravity
Create ForceModel NonSpherical12;
GMAT NonSpherical12.PrimaryBodies        = {Earth};
GMAT NonSpherical12.Gravity.Earth.Model  = JGM2;
GMAT NonSpherical12.Gravity.Earth.Degree = 12;
GMAT NonSpherical12.Gravity.Earth.Order  = 12;

%  Define Force Model with MSISE90 Drag
Create ForceModel  MSISE90Drag;
GMAT  MSISE90Drag.PrimaryBodies        = {Earth};
GMAT  MSISE90Drag.Gravity.Earth.Model  = JGM2;
GMAT  MSISE90Drag.Gravity.Earth.Degree = 0;
GMAT  MSISE90Drag.Gravity.Earth.Order  = 0;
GMAT  MSISE90Drag.Drag       = MSISE90;
GMAT  MSISE90Drag.Drag.F107  = 150;
GMAT  MSISE90Drag.Drag.F107A = 150;
GMAT  MSISE90Drag.Drag.MagneticIndex = 3;

%  Define Force Model with Jacchia Roberts Drag
Create ForceModel  JRDrag;
GMAT  JRDrag.PrimaryBodies        = {Earth};
GMAT  JRDrag.Gravity.Earth.Model  = JGM2;
GMAT  JRDrag.Gravity.Earth.Degree = 0;
GMAT  JRDrag.Gravity.Earth.Order  = 0;
GMAT  JRDrag.Drag       = JacchiaRoberts;
GMAT  JRDrag.Drag.F107  = 150;
GMAT  JRDrag.Drag.F107A = 150;
GMAT  JRDrag.Drag.MagneticIndex = 3;

%  Define Force Model with SRP
Create ForceModel  SRP;
GMAT  SRP.PrimaryBodies        = {Earth};
GMAT  SRP.Gravity.Earth.Model  = JGM2;
GMAT  SRP.Gravity.Earth.Degree = 0;
```

```
GMAT  SRP.Gravity.Earth.Order  = 0;
GMAT  SRP.SRP = On;

%----------------------------Create Propagators----------------------
% Create propgator with point mass only
Create Propagator  RKV89PointMass;
GMAT  RKV89PointMass.Type     = RungeKutta89;
GMAT  RKV89PointMass.StepSize = 30;;
GMAT  RKV89PointMass.Accuracy = 1e-12;
GMAT  RKV89PointMass.MinStep  = 30;
GMAT RKV89PointMass.MaxStep   = 30;
GMAT  RKV89PointMass.FM       = PointMass;

% Create propgator with third bodies only
Create Propagator  RKV89ThirdBodies;
GMAT  RKV89ThirdBodies.Type     = RungeKutta89;
GMAT  RKV89ThirdBodies.StepSize = 30;;
GMAT RKV89ThirdBodies.Accuracy  = 1e-12;
GMAT RKV89ThirdBodies.MinStep   = 30;
GMAT RKV89ThirdBodies.MaxStep   = 30;
GMAT  RKV89ThirdBodies.FM       =  ThirdBodies;

% Create propgator with 12x12 gravity only
Create Propagator  RKV8912x12;
GMAT RKV8912x12.Type      = RungeKutta89;
GMAT  RKV8912x12.StepSize = 30;;
GMAT  RKV8912x12.Accuracy = 1e-12;
GMAT RKV8912x12.MinStep   = 30;
GMAT RKV8912x12.MaxStep   = 30;
GMAT  RKV8912x12.FM       =  NonSpherical12;

% Create propgator with MSISE-90 drag only
Create Propagator  RKV89MSISE90;
GMAT RKV89MSISE90.Type     = RungeKutta89;
GMAT RKV89MSISE90.Accuracy = 1e-12;
GMAT RKV89MSISE90.MinStep  = 30;
GMAT RKV89MSISE90.MaxStep  = 30;
GMAT  RKV89MSISE90.FM      =   MSISE90Drag;

% Create propgator with Jacchia-Roberts drag only
Create Propagator  RKV89JR;
GMAT RKV89JR.Type     = RungeKutta89;
GMAT RKV89JR.StepSize = 30;
GMAT RKV89JR.Accuracy = 1e-12;
GMAT RKV89JR.MinStep  = 30;
GMAT RKV89JR.MaxStep  = 30;
GMAT  RKV89JR.FM      =   JRDrag;

% Create propgator with SRP only
Create Propagator  RKV89SRP;
GMAT  RKV89SRP.Type     = RungeKutta89;
GMAT  RKV89SRP.StepSize = 30;
GMAT  RKV89SRP.Accuracy = 1e-12;
GMAT  RKV89SRP.MinStep  = 30;
GMAT  RKV89SRP.MaxStep  = 30;
GMAT  RKV89SRP.FM       =   SRP;
```

```
% ----------------------------------------------------------------------
% --------------------------  Begin Mission Sequence --------------------
% ----------------------------------------------------------------------
%Propagate using point mass propagator
Propagate  RKV89PointMass(Sat1, {Sat1.ElapsedDays = 0.1});

%Propagate using third bodies propagator
Propagate   RKV89ThirdBodies(Sat1, {Sat1.ElapsedDays = 0.1});

%Propagate using 12x12 gravity model propgator
Propagate  RKV8912x12(Sat1, {Sat1.ElapsedDays = 0.1});

%Propagate using MSISE-90 propagator
Propagate  RKV89MSISE90(Sat1, {Sat1.ElapsedDays = 0.1});

%Propagate using Jacchia-Roberts propagator
Propagate  RKV89JR(Sat1, {Sat1.ElapsedDays = 0.1});

%Propagate using SRP propagator
Propagate  RKV89SRP(Sat1, {Sat1.ElapsedDays = 0.1});
```

21.3 Control Flow

```
%  Script Example:

% ----------------------------------------------------------------------
% --------------------------- Create Objects ---------------------------
% ----------------------------------------------------------------------

% Create Sat1 and define its orbit
Create Spacecraft Sat1;
GMAT Sat1.Epoch.TAIGregorian  = 09 Oct 2004 16:30:00.124;
GMAT Sat1.StateType = Cartesian;
GMAT Sat1.X  = 10000;
GMAT Sat1.Y  = 0;
GMAT Sat1.Z  = 0;
GMAT Sat1.VX = 0;
GMAT Sat1.VY = 8.0;
GMAT Sat1.VZ = 1.0;

%  Define Force Model with JR drag
Create ForceModel FM;
GMAT FM.PrimaryBodies        = {Earth};
GMAT FM.PointMasses          = {Sun, Luna};
GMAT FM.Gravity.Earth.Model  = JGM2;
GMAT FM.Gravity.Earth.Degree = 4;
GMAT FM.Gravity.Earth.Order  = 4;

% Create Propagator
Create Propagator RKV_LowEarth_JR;
GMAT RKV_LowEarth_JR.Type    = RungeKutta89;
```

```
GMAT RKV_LowEarth_JR.MinStep = .01;
GMAT RKV_LowEarth_JR.MaxStep = 900;
GMAT RKV_LowEarth_JR.FM      = FM;

%  Create Variables and Arrays
Create Array MMS1M_ApoEphem[20,7];
Create Variable I Count;

%  Create OpenGL Plot
Create OpenGLPlot GLPlot;
GMAT GLPlot.Add = Sat1;

% ---------------------------------------------------------------------
% --------------------------  Begin Mission Sequence --------------------
% ---------------------------------------------------------------------

%  Prop for 20 orbits and save ephem at every apogee
For I = 1:20

     %Propagate to next apoapsis
     Propagate RKV_LowEarth_JR(Sat1,  {Sat1.Apoapsis} );

     GMAT MMS1M_ApoEphem(I,1) = MMS1M.Epoch;
     GMAT MMS1M_ApoEphem(I,2) = MMS1M.X;
     GMAT MMS1M_ApoEphem(I,3) = MMS1M.Y;
     GMAT MMS1M_ApoEphem(I,4) = MMS1M.Z;
     GMAT MMS1M_ApoEphem(I,5) = MMS1M.VX;
     GMAT MMS1M_ApoEphem(I,6) = MMS1M.VY;
     GMAT MMS1M_ApoEphem(I,7) = MMS1M.VZ;

EndFor

While Sat1.ElapsedDays < 10

   Propagate RKV_LowEarth_JR(Sat1,  {Sat1.Apoapsis} );

EndWhile

If  Sat1.TA > 0

    GMAT Sat1.VZ =  2.0;
    Propagate RKV_LowEarth_JR(Sat1,  {Sat1.Apoapsis} );

EndIf
```

21.4 Multiple Spacecraft Propagation

```
%  Multiple Spacecraft Propagation Script Example
```

```
%
%  This script shows how to propagate multiple spacecraft in many different ways
%  including coupled and synchronized methods

% ----------------------------------------------------------------------
% -------------------------- Create Objects ----------------------------
% ----------------------------------------------------------------------

% Create Sat1 and define its orbit
Create Spacecraft Sat1;
GMAT Sat1.Epoch.TAIGregorian  = 09 Oct 2004 16:30:00.120;
GMAT Sat1.StateType = Keplerian;
GMAT Sat1.SMA      = 8000;
GMAT Sat1.ECC      = .01;
GMAT Sat1.INC      = 28.5;
GMAT Sat1.AOP      = 0;
GMAT Sat1.RAAN     = 90;
GMAT Sat1.TA       = 180;

% Create Sat2 and define its orbit
Create Spacecraft Sat2;
GMAT Sat2.Epoch.TAIGregorian  = 09 Oct 2004 16:30:00.120;
GMAT Sat2.StateType = Keplerian;
GMAT Sat2.SMA     = 25000;
GMAT Sat2.ECC     = .1;
GMAT Sat2.INC     = 28.5;
GMAT Sat2.AOP     = 0;
GMAT Sat2.RAAN    = 90;
GMAT Sat2.TA      = 181;

% Create Sat3 and define its orbit
Create Spacecraft Sat3;
GMAT Sat3.Epoch.TAIGregorian  = 09 Oct 2004 16:30:00.120;
GMAT Sat3.StateType = Keplerian;
GMAT Sat3.SMA   = 25000;
GMAT Sat3.ECC     = 0.1;
GMAT Sat3.INC     = 28.5;
GMAT Sat3.AOP   = 0;
GMAT Sat3.RAAN  = 90;
GMAT Sat3.TA    = 182;

%  Define Force Model for HEO orbit
Create ForceModel HighEarth;
GMAT HighEarth.PrimaryBodies        = {Earth};
GMAT HighEarth.PointMasses          = {Sun, Luna, Jupiter};
GMAT HighEarth.Drag                 = None;
GMAT HighEarth.Gravity.Earth.Model  = JGM2;
GMAT HighEarth.Gravity.Earth.Degree = 12;
GMAT HighEarth.Gravity.Earth.Order  = 12;
GMAT HighEarth.SRP                  = On;

% Create Propagator for HEO orbit
Create Propagator RKF_HighEarth;
GMAT RKF_HighEarth.Type    = RungeKutta89;
GMAT RKF_HighEarth.MinStep = .01;
GMAT RKF_HighEarth.MaxStep = 900;
GMAT RKF_HighEarth.FM      = HighEarth;
```

```
%  Define Force Model for  LEO orbit
Create ForceModel LowEarth_JR;
GMAT LowEarth_JR.PrimaryBodies      = {Earth};
GMAT LowEarth_JR.PointMasses        = {Sun, Luna, Jupiter};
GMAT LowEarth_JR.Drag               = JacchiaRoberts;
GMAT LowEarth_JR.Drag.F107          = 100;
GMAT LowEarth_JR.Drag.F107A         = 120;
GMAT LowEarth_JR.Drag.MagneticIndex = 20;
GMAT LowEarth_JR.Gravity.Earth.Model  = JGM2;
GMAT LowEarth_JR.Gravity.Earth.Degree = 12;
GMAT LowEarth_JR.Gravity.Earth.Order  = 12;

% Create Propagator for LEO orbit
Create Propagator RKV_LowEarth_JR;
GMAT RKV_LowEarth_JR.Type     = RungeKutta89;
GMAT RKV_LowEarth_JR.MinStep  = .01;
GMAT RKV_LowEarth_JR.MaxStep  = 900;
GMAT RKV_LowEarth_JR.FM       = LowEarth_JR;

Create OpenGLPlot GLPlot
GMAT GLPlot.Add = Sat1;
GMAT GLPlot.Add = Sat2;
GMAT GLPlot.Add = Sat3;

% ---------------------------------------------------------------------
% -------------------- Begin Mission Sequence -------------------------
% ---------------------------------------------------------------------

%  The Propagate command below tells GMAT to propagate all three spacecraft
%  as a coupled system.  The propagator stops when the condition Sat2.MA = 0
%  is satisfied.  At the end of this propagate event all spacecraft will be
%  at the same epoch, the epoch when Sat to reaches MA = 0.
Propagate RKF_HighEarth(Sat1,Sat2,Sat3,{Sat2.MA = 0});

%  The propagation sequence below propagates Sat1 using the
%  RKV_LowEarth_JR propgagator.  The line also tells GMAT to propagate Sat2
%  and Sat3 as a coupled system.  The "Syncronized" command tells GMAT to
%  propagate the spacecraft such that the spacecraft are synchronized in time
%  at each time step.  The first spacecraft in the first propagate command
%  is used as the synchronization epoch.  The Propagator will propagate all
%  spacecraft until either Sat1.Periapsis is achieved, or until Sat2.TA =
%  155.  Once the first of these conditions is met, the propagator exits.
Propagate Synchronized RKV_LowEarth_JR(Sat1,{Sat1.Periapsis})   ...
                                  RKF_HighEarth(Sat2,Sat3,{Sat2.TA = 155});


%  The Propagate command below tells GMAT to propagate Sat1 and Sat 2, however,
%  they will not be propagated synchonously.  Hence at each time step,
%  including the final time step, the spacecraft will be at different
%  epochs.
Propagate RKV_LowEarth_JR(Sat1,{Sat1.Periapsis})   ...
                                  RKF_HighEarth(Sat2,{Sat2.Periapsis});
```

21.5 Calling Functions in Matlab

```
%  Calling Matlab Functions Script Example

%  This script demonstrates how to send different types of data to matlab
%  and receive different types of data in return.  The types of data that
%  can be sent to matlab include variables, arrays, strings, spacecraft parameters
%  and entire objects.  Currently GMAT can receive all of these in return
%  except for entire objects.

% ---------------------------------------------------------------------
% -------------------------- Create Objects ---------------------------
% ---------------------------------------------------------------------

%  Create variables:
%        Datatype:  This is a flag that is sent to the matlab function
%                   GetMMSStates.  The flag determines whether to include
%                   navigation errors in the return arguments from the
%                   function
Create Variable DataType;

%  Create arrays:
%        MMS1AState[6,1]:  This is a 6x1 vector that is returned from the
%        matlab function GetMMSSates.  The vector is the cartesian state
%        for spacecraft MMS1A.
%
%        MMS1MState[6,1]:  This is a 6x1 vector that is returned from the
%        matlab function GetMMSSates.  The vector is the cartesian state
%        for spacecraft MMS1M.
Create Array MMS1AState[6,1]  MMS1MState[6,1]

%  Create strings
%        SaveData:  This is a string that is used in a switch/case command
%        in the matlab file SaveMMSData.  When 'Yes', the data is save.
%        When any thing else, the data is not saved.
Create String SaveData;

%  Create matlab functions.  Here we only tell GMAT that this is a matlab
%  function as opposed to a GMAT predefined function or a user defined GMAT
%  function.  The inputs and outputs are defined in the function call later
%  in the script.
Create MatlabFunction GetMMSStates;
Create MatlabFunction SaveMMSData;

%  Define MMS1A orbit,  the orbit of the actual spacecraft
Create Spacecraft MMS1A;
GMAT MMS1A.Epoch.TAIModJulian = 22800.589;
GMAT MMS1A.ReferenceFrame = EarthMJ2000Eq;
GMAT MMS1A.StateType = Keplerian;
GMAT MMS1A.SMA     = 42500;
GMAT MMS1A.ECC     = .8383;
GMAT MMS1A.INC     = 18;
GMAT MMS1A.AOP   = 90;
GMAT MMS1A.RAAN  = 0;
GMAT MMS1A.TA    = 180;
```

```
%  Define MMS1M orbit,  the measured orbit of that includes OD errors
Create Spacecraft MMS1M;
GMAT MMS1M.Epoch.TAIModJulian = 22800.589;
GMAT MMS1M.ReferenceFrame = EarthMJ2000Eq;
GMAT MMS1M.StateType = Keplerian;
GMAT MMS1M.SMA     = 42500;
GMAT MMS1M.ECC     = .8383;
GMAT MMS1M.INC     = 18;
GMAT MMS1M.AOP   = 90;
GMAT MMS1M.RAAN  = 0;
GMAT MMS1M.TA    = 180.05;

%  Define Force Model with no drag
Create ForceModel LowEarth;
GMAT LowEarth.PrimaryBodies      = {Earth};
%GMAT LowEarth.Drag               = None;
GMAT LowEarth.Gravity.Earth.Model  = JGM3;
GMAT LowEarth.Gravity.Earth.Degree = 2;
GMAT LowEarth.Gravity.Earth.Order  = 0;

% Create Propagator
Create Propagator PD_LowEarth;
GMAT PD_LowEarth.Type    = PrinceDormand78;
GMAT PD_LowEarth.MinStep = .0001;
GMAT PD_LowEarth.MaxStep = 2700;
GMAT PD_LowEarth.FM      = LowEarth;

%  Create an OpenGLPlot
Create OpenGLPlot MMSPlot;
GMAT MMSPlot.Add = MMS1M;
GMAT MMSPlot.Add = MMS1A;

% ---------------------------------------------------------------------
% --------------------------- Mission Sequence ------------------------
% ---------------------------------------------------------------------

%  Set flags and switches
GMAT SaveData = Yes;
GMAT DataType = 1;

%  Call matlab.  Here we send Datatype, which currently equals 1, and
%  receive two vectors in return.
GMAT [MMS1MState, MMS1AState] = GetMMSStates(DataType);

%  This is a temporary fix.  Currently you must switch the StateType to
%  Cartesian before you can set the spacecraft cartesian state, because
%  the original StateType was Keplerian.
GMAT MMS1A.StateType = Cartesian;
GMAT MMS1M.StateType = Cartesian;

%  Set state of MMS1A;
GMAT MMS1A.X  = MMS1AState(1,1);
GMAT MMS1A.Y  = MMS1AState(2,1);
GMAT MMS1A.Z  = MMS1AState(3,1);
GMAT MMS1A.VX = MMS1AState(4,1);
GMAT MMS1A.VY = MMS1AState(5,1);
GMAT MMS1A.VZ = MMS1AState(6,1);
```

```
%  Set state of MMS1M;
GMAT MMS1M.X  = MMS1MState(1,1);
GMAT MMS1M.Y  = MMS1MState(2,1);
GMAT MMS1M.Z  = MMS1MState(3,1);
GMAT MMS1M.VX = MMS1MState(4,1);
GMAT MMS1M.VY = MMS1MState(5,1);
GMAT MMS1M.VZ = MMS1MState(6,1);

%  Prop for 1 day
Propagate PD_LowEarth(MMS1A, MMS1M, {MMS1A.ElapsedDays = 1} );

%  Prop to apoapsis of MMS1M
Propagate PD_LowEarth(MMS1A, MMS1M, {MMS1A.Earth.Apoapsis} );

%  Call matlab.  Here we send DataType, which now equals 2, and the entire structure
%  of the MMS1A Spacecraft.  Matlab takes the state of MMS1A, and adds
%  random error to it as a simple navigation error model.
GMAT DataType     = 2;
GMAT [MMS1MState] = GetMMSStates(DataType, MMS1A);

%  Set state of MMS1M;
GMAT MMS1M.X  = MMS1MState(1,1);
GMAT MMS1M.Y  = MMS1MState(2,1);
GMAT MMS1M.Z  = MMS1MState(3,1);
GMAT MMS1M.VX = MMS1MState(4,1);
GMAT MMS1M.VY = MMS1MState(5,1);
GMAT MMS1M.VZ = MMS1MState(6,1);

%  For loop
For I = 1:10

    %  Propagate to apoapsis
    Propagate PD_LowEarth(MMS1A, MMS1M, {MMS1M.Apoapsis} );

    %  Call matlab.  This call demonstrates how to send  individual
    %  spacecraft parameters to matlab.
    GMAT SaveMMSData(MMS1M.SMA, MMS1M.INC, MMS1M.ECC, SaveData);

EndFor;
```

21.6 Maneuvers

```
% Finite Burn Script Example

%  This script demonstrates how to set up tanks and thrusters, and use them
%  in a finite maneuver sequence.


% ----------------------------------------------------------------------
% -------------------------- Create Objects ----------------------------
% ----------------------------------------------------------------------
```

```
%  Create a default spacecraft
Create Spacecraft Sc;

%  Create a fuel tank and name it tank1
%          Here we create a fuel tank and set up its physical properties
%          including Temperature, Fuel Mass, Fuel Density etc.
Create FuelTank tank1
GMAT tank1.Temperature = 20.0;
GMAT tank1.RefTemperature = 12.0;
GMAT tank1.FuelMass = 725;
GMAT tank1.FuelDensity = 1029;
GMAT tank1.Pressure = 1200.0;
GMAT tank1.Volume = 0.8;
GMAT tank1.PressureRegulated = true;

%  Create a fuel tank and name it tank2
Create FuelTank tank2
GMAT tank2.RefTemperature = 32.0;
GMAT tank2.FuelMass = 325;

%  Create a thruster
%         Here we create a thruster and tell the thruster which tank to
%         draw fuel from.  We also set up the direction of the thruster.
%         Currently, you specify the thruster orientation with respect to
%         the spacecraft VNB or EarthMJ2000Eq systems.  This will change
%         when attitude capabilities are added to GMAT.
Create Thruster engine1
GMAT engine1.Tank = {tank1};
GMAT engine1.CoordinateSystem = Sat1VNB;
GMAT engine1.C1 = 400;
GMAT engine1.C2 = 0.15;
GMAT engine1.K1 = 1500;
GMAT engine1.K3 = 0.000451;
GMAT engine1.X_Direction = 1.0;
GMAT engine1.Y_Direction = 2.0;
GMAT engine1.Z_Direction = 2.0;

%  Create another thruster, tell it to draw from tank2,  and leave its
%  other settings as default.
Create Thruster engine2
GMAT engine2.Tank = {tank2};

%  Here we attach the tanks and thrusters created above to the spacecraft
GMAT Sc.Tanks = {tank1, tank2}
GMAT Sc.Thrusters = {engine1, engine2};

Create FiniteBurn fb;
GMAT fb.Thrusters = {engine1, engine2};
GMAT fb.Tanks = {tank1, tank2};
GMAT fb.CoordinateFrame = VNB;

% Test thrust scale factor code
GMAT fb.BurnScaleFactor = .01;
GMAT engine1.ThrustScaleFactor = .2;
GMAT engine2.ThrustScaleFactor = .2;

Create ForceModel fm;
```

```
GMAT fm.PointMasses = {Earth, Sun, Luna};

Create Propagator prop;
GMAT prop.FM = fm;

Create ReportFile rf;
GMAT rf.Filename = finiteBurn.txt;
GMAT rf.Precision = 15;
GMAT rf.Add = Sc.ElapsedSecs;
GMAT rf.Add = Sc.X;
GMAT rf.Add = Sc.Y;
GMAT rf.Add = Sc.Z;
GMAT rf.Add = Sc.SMA;
GMAT rf.Add = Sc.Energy;

%  Create XYPlot
Create XYPlot energy
GMAT energy.IndVar = Sc.ElapsedDays;
GMAT energy.Add    = Sc.Energy;

%  Create OpenGL Plot
Create OpenGLPlot SatOpenGL
GMAT SatOpenGL.Add = Sc;

% Mission sequence
Propagate prop(Sc, {Sc.ElapsedSecs = 8640});

BeginFiniteBurn fb(Sc);
   Propagate prop(Sc, {Sc.ElapsedDays = 5});
EndFiniteBurn fb(Sc);

Propagate prop(Sc, {Sc.ElapsedDays =5});
```

21.7 OpenGL

```
%  Build 4, Test Case 3.  This is a formation flying mission.  The mission
%  requires spacecraft dependent coordinate systems and has plots in different
%  spacecraft dependent coordinate systems and targets variables that are
%  expressed in s/c dependent coordinate systems.

% ----------------------------------------------------------------------
% -------------------------- Create Objects ----------------------------
% ----------------------------------------------------------------------


% ----------------------Spacecraft/Propagators----------------------------

%  Create Sat1
Create Spacecraft Sat1;
GMAT Sat1.Epoch.UTCGregorian = 01 Jan 2000 11:59:27.966;
%GMAT Sat1.CoordinateSystem = EarthMJ2000Eq;
GMAT Sat1.StateType = Cartesian;
GMAT Sat1.X   = -6648.080001;
```

```
GMAT Sat1.Y   = 160.991;
GMAT Sat1.Z   = -982.996;
GMAT Sat1.VX  = 0.481415;
GMAT Sat1.VY  = -6.39538;
GMAT Sat1.VZ  = -4.27317;

%  Create Sat2
Create Spacecraft Sat2;
GMAT Sat2.Epoch.UTCGregorian = 01 Jan 2000 11:59:27.966;
%GMAT Sat2.CoordinateSystem = EarthMJ2000Eq;
GMAT Sat2.StateType = Cartesian;
GMAT Sat2.X   = -6646.708142345;
GMAT Sat2.Y   = 160.1539527656;
GMAT Sat2.Z   = -983.7055310239;
GMAT Sat2.VX  = 0.4835092171146;
GMAT Sat2.VY  = -6.395907187307;
GMAT Sat2.VZ  = -4.274772233631;

%  Define Force Model with Exponential drag
Create ForceModel LowEarth;
%GMAT LowEarth.Origin               = Earth;
GMAT LowEarth.PrimaryBodies        = {Earth};
GMAT LowEarth.PointMasses          = {Sun, Moon, Jupiter};
GMAT LowEarth.Drag                 = Exponential;
GMAT LowEarth.Gravity.Earth.Model  = JGM2;
GMAT LowEarth.Gravity.Earth.Degree = 10;
GMAT LowEarth.Gravity.Earth.Order  = 10;

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

%  Create maneuver
Create ImpulsiveBurn Burn1;
GMAT Burn1.CoordinateFrame = VNB;
GMAT Burn1.Element1        = .0001;

%  Create Differential Corrector
Create DifferentialCorrector DC;
%GMAT DC.MaxIterations = 25;

% ------------------------------Coordinate Systems-------------------------

%  Create Sat1LVLH coordinate system
Create CoordinateSystem Sat1LVLH;
GMAT Sat1LVLH.Axes    = ObjectReferenced;
GMAT Sat1LVLH.Origin  = Sat1;
GMAT Sat1LVLH.Primary = Earth;
GMAT Sat1LVLH.YAxis   = -N;
GMAT Sat1LVLH.ZAxis   = -R;

%  Create Sat1 Sun Pointing system
Create CoordinateSystem Sat1SunPointing;
GMAT Sat1SunPointing.Axes    = ObjectReferenced;
```

```
GMAT Sat1SunPointing.Origin  = Sat1;
GMAT Sat1SunPointing.Primary = Sun;
GMAT Sat1SunPointing.XAxis   = -R;
GMAT Sat1SunPointing.YAxis   = N;

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

% ----------------------------XYPlots-------------------------------------
%  Create XYPlot of Sat2 in Sat1LVLH system
Create XYPlot Sat2wrtSat1LVLH;
GMAT Sat2wrtSat1LVLH.TargetStatus = Off;
GMAT Sat2wrtSat1LVLH.IndVar   = Sat2.Sat1LVLH.Y;
GMAT Sat2wrtSat1LVLH.Add      = Sat2.Sat1LVLH.Z;

%  Create XYPlot of Sat2 in Sat1 Sun Pointing System
Create XYPlot Sat2Sat1SunPointing;
GMAT Sat2Sat1SunPointing.TargetStatus = Off;
GMAT Sat2Sat1SunPointing.IndVar       = Sat2.Sat1SunPointing.Y;
GMAT Sat2Sat1SunPointing.Add          = Sat2.Sat1SunPointing.Z;

%  Create XYPlot Sat1 in Sat2 VNB system
Create XYPlot Sat1wrtSat2VNB;
GMAT Sat1wrtSat2VNB.TargetStatus = Off;
GMAT Sat1wrtSat2VNB.IndVar       = Sat1.Sat2VNB.X;
GMAT Sat1wrtSat2VNB.Add          = Sat1.Sat2VNB.Y;

% ----------------------------OpenGL Plots-------------------------------------

%  Create OpenGL Plot of Sat2 in Sat1RBN system
Create OpenGLPlot Sat2Sat1OpenGL;
GMAT Sat2Sat1OpenGL.TargetStatus     = Off;
GMAT Sat2Sat1OpenGL.CoordinateSystem = Sat1RBN;
GMAT Sat2Sat1OpenGL.Add              = Sat1;
GMAT Sat2Sat1OpenGL.Add              = Sat2;
GMAT Sat2Sat1OpenGL.ViewPoint        = Sat2;
GMAT Sat2Sat1OpenGL.ViewDirection    = Sat1;
GMAT Sat2Sat1OpenGL.ViewScaleFactor  = 1.05;

%  Create OpenGL Plot of Sat2 in Sat1 Sun pointing System
Create OpenGLPlot Sat2Sat1SunOpenGL;
GMAT Sat2Sat1SunOpenGL.TargetStatus   = Off;
GMAT Sat2Sat1SunOpenGL.CoordinateSystem = Sat1SunPointing;
GMAT Sat2Sat1SunOpenGL.Add             = Sat1;
GMAT Sat2Sat1SunOpenGL.Add             = Sat2;
GMAT Sat2Sat1SunOpenGL.ViewPoint       = Sat1;
%GMAT Sat2Sat1SunOpenGL.ViewDirection  = Sat2;
GMAT Sat2Sat1SunOpenGL.ViewScaleFactor = 1.05;

%  Toggle plots
Toggle Sat2wrtSat1RBN On;
Toggle Sat2Sat1SunPointing On;
```

```
Toggle Sat1wrtSat2VNB On;
Toggle Sat2Sat1OpenGL On;
Toggle Sat2Sat1SunOpenGL On;

% ---------------------------- Variables-----------------------------------

%  Create Variables
Create Variable TOF;

% ------------------------------------------------------------------------
% ---------------------------- Mission Sequence----------------------------
% ------------------------------------------------------------------------

%  Propagate and watch cool plots!!
Propagate RKV_LowEarth(Sat1, Sat2, {Sat1.ElapsedDays = .2});

%  Set TOF
GMAT TOF = Sat1.Period / 2;

%  Target for rendezvous
Target DC;

     Vary DC(Burn1.V = 0.0001, {Pert = 0.000001};
     Vary DC(Burn1.N = 0.0001, {Pert = 0.000001};
     Vary DC(Burn1.B = 0.0001, {Pert = 0.000001};

     Maneuver Burn1(Sat2);

     Propagate  RKV_LowEarth(Sat1, Sat2, {Sat1.ElapsedSecs = TOF});

     Achieve DC(Sat2.Sat1RBN.X = 0, {Tolerance = 0.0001});
     Achieve DC(Sat2.Sat1RBN.Y = 0, {Tolerance = 0.0001});
     Achieve DC(Sat2.Sat1RBN.Z = 0, {Tolerance = 0.0001});

EndTarget;

%  Propagate and watch cool plots!!
Propagate RKV_LowEarth(Sat1, Sat2, {Sat1.ElapsedDays = .2});
```

21.8 Libration Point Targeting

```
%  Build 4, Test Case 4.   This is a libration point mission.  Starts out
%  in low earth and performs a sequence of targeting loops to achieve a "stable"
%  libration orbit about L2.  It requires rotating coordinate system plots and variables.

% ------------------------------------------------------------------------
% ---------------------------- Create Objects ----------------------------
% ------------------------------------------------------------------------
```

```
% ------------------------Coordinate Systems-----------------------------
%  Create Earth Moon Barycenter
Create BaryCenter EarthMoonBary
GMAT EarthMoonBary.Add = Earth;
GMAT EarthMoonBary.Add = Luna;

%  Create Libration Point Sun_EarthMoonBaryL2
Create LibrationPoint Sun_EarthMoonBaryL2;
GMAT Sun_EarthMoonBaryL2.Point     = L2;
GMAT Sun_EarthMoonBaryL2.Primary   = Sun;
GMAT Sun_EarthMoonBaryL2.Secondary = EarthMoonBary;

%  Create Libration Point SunEarthL2
Create LibrationPoint SunEarthL2;
GMAT SunEarthL2.Point     = L2;
GMAT SunEarthL2.Primary   = Sun;
GMAT SunEarthL2.Secondary = Earth;

%  Create Coordinate System SEML2
Create CoordinateSystem SEML2
GMAT SEML2.Origin = Sun_EarthMoonBaryL2;
GMAT SEML2.Axes = RBN;
GMAT SEML2.Primary = Sun;
GMAT SEML2.Secondary = EarthMoonBary;

%  Create Coordinate System SEL2
Create CoordinateSystem SEL2
GMAT SEL2.Origin = SunEarthL2;
GMAT SEL2.Axes = RBN;
GMAT SEL2.Primary = Sun;
GMAT SEL2.Secondary = Earth;

% ---------------------------------------Plots---------------------------------
%  Create OpenGL Plot SEML2OpenGL
Create OpenGLPlot SEML2OpenGL
GMAT SEML2OpenGL.TargetStatus = Off;
GMAT SEML2OpenGL.CoordinateSystem = SEML2;
GMAT SEML2OpenGL.Add = Sat;

%  Create OpenGL Plot SEL2OpenGL
Create OpenGLPlot SEL2OpenGL
GMAT SEL2OpenGL.TargetStatus = Off;
GMAT SEL2OpenGL.CoordinateSystem = SEL2;
GMAT SEL2OpenGL.Add = Sat;

%  Create XYPlot L2XZ
Create OpenGL Plot L2XZ;
GMAT L2XZ.TargetStatus = On;
GMAT L2XZ.IndVar = Sat.SEML2.X;
GMAT L2XZ.Add = Sat.SEML2.Z;

% ------------------------S/C, Force models, Propagators----------------------
%  Define Spacecraft
Create Spacecraft Sat;
GMAT Sat.Epoch.UTCGregorian = 10 Jun 2003 02:08:10.13;
GMAT Sat.ReferenceFrame = EarthMJ2000Eq;
GMAT Sat.StateType = Cartesian;
```

```
GMAT Sat.X     = 196.29222100000 ;
GMAT Sat.Y     = 6551.77147700000;
GMAT Sat.Z     = 332.45624900000;
GMAT Sat.VX   = -6.84977882700 ;
GMAT Sat.VY  = 0.39287054000;
GMAT Sat.VZ    = -3.69804959700;

Create Spacecraft TempSat;
TempSat = Sat;

%  Define Force Model with no drag
Create ForceModel DeepSpace;
GMAT DeepSpace.Origin             = Earth;
GMAT DeepSpace.PointMasses        = {Sun, Earth, Luna};

% Create Propagator
Create Propagator DeepSpace;
GMAT DeepSpace.Type    = PrinceDormand78;
GMAT DeepSpace.MinStep = 1;
GMAT DeepSpace.MaxStep = 900;
GMAT DeepSpace.FM      = DeepSpace;

% ---------------------------------DC and Maneuvers-------------------------
%  Create maneuver
Create ImpulsiveBurn TTI;
GMAT TTI.CoordinateFrame = VNB;
GMAT TTI.Element1        = 3.20130050000;

%  Create maneuver
Create ImpulsiveBurn dVCorrect;
GMAT dVCorrect.CoordinateFrame = VNB;
GMAT dVCorrect.Element1        = .000001;

%  Create Differential Corrector
Create DifferentialCorrector DC;
GMAT DC.MaxIter = 25;

% ---------------------------------Variables, Arrays, Functions-----------
Create Variable SKNum I NumHalfRev J;      %  SKNum = Number of station keeping maneuvers
                                       %  NumHalfRev is the number of 1/2 revs
                                       %  between station keeping maneuvers
GMAT SKNum = 2;
GMAT NumRevs = 2;

% ----------------------------------------------------------------------
% --------------------------- Mission Sequence --------------------------
% ----------------------------------------------------------------------

%  Apply Insertion delta V, TTI
Maneuver TTI(Sat);

%  Propagate for one day
Propagate DeepSpace(Sat, {Sat.ElapsedDays = 1});


%-------------------------- First targeting loop---------------------------
%  Target to get desired C3 Energy,  This is to provide initial guess to
```

```
%  the next targeting loop.
TempSat = Sat;
Target DC;

     %  Vary the maneuver
     Vary DC( dVCorrect.V = .01 , {Pert = .0000001});
     Maneuver dVCorrect(Sat);

     %  Achieve desired C3
     Achieve DC( Sat.Earth.C3  = -0.5635 , {Tolerance = 0.0001});

EndTarget;
Maneuver dVCorrect(TempSat);
GMAT Sat = TempSat;

%-------------------------- Second targeting loop---------------------------
%  Target to achieve a small vx-component at first xz plane crossing,
%  This is to provide initial guess to the next targeting loop.
Target DC;

     %  Vary the maneuver
     Vary DC( dVCorrect.V = .01 , {Pert = .0000001});
     Maneuver dVCorrect(Sat);

     %  Propagate to XZ-plane crossing in L2 system
     Propagate DeepSpace(Sat, {Sat.SEML2.Y = 0});

     % Achieve a small vx component in SEML2 coordinate system
     Achieve DC( Sat.SEML2.VX  = .00348 , {Tolerance = 0.0001});

EndTarget;
Maneuver dVCorrect(TempSat);
GMAT Sat = TempSat;

%-------------------------- Third targeting loop---------------------------
%  Target to achieve a zero vx-component at second xz plane crossing,
%  This is the last insertion targeting loop to find delta v to insert into
%  a relatively stable libration orbit.
Target DC;

     %  Vary the maneuver
     Vary DC( dVCorrect.V = .01 , {Pert = .0000001});
     Maneuver dVCorrect(Sat);

     %  Propagate to second XZ-plane crossing in L2 system
     For J = 1:2;
         Propagate DeepSpace(Sat, {Sat.SEML2.Y = 0});
     EndFor; % EndFor J = 1:NumRevs;

     % Achieve a zero vx component in SEML2 coordinate system
     Achieve DC( Sat.SEML2.VX  = 0 , {Tolerance = 0.0001});

EndTarget;
Maneuver dVCorrect(TempSat);
GMAT Sat = TempSat;

%  Propagate to firts xz-plane crossing in L2 system
```

```
Propagate DeepSpace(Sat, {Sat.SEML2.Y = 0});


%-------------------------- Station Keeping---------------------------------
%  Perform a station keeping maneuver every half rev, by targeting vx = 0
%  at one full rev propagation.
For I = 1:SKNum;

     TempSat = Sat;
     Target DC

        %  Vary delta V and apply maneuver
        Vary DC( dVCorrect.V = -0.00026070300 , {Pert = .0000001});
        Maneuver dVCorrect(Sat);

        %  Propagate through NumHalfRevs to XZ-plane crossing in SEML2 system
        For J = 1:NumHalfRevs;
           Propagate DeepSpace(Sat, {Sat.SEML2.Y = 0});
        EndFor; % EndFor J = 1:NumRevs;


        Achieve DC( Sat.SEML2.VX  = 0, {Tolerance = 0.000001});

     EndTarget

     %  Now add in maneuver and prop for 1/2 rev
     Maneuver dVCorrect(TempSat);
     Sat = TempSat;
     Propagate DeepSpace(Sat, {Sat.SEML2.Y = 0});

EndFor  % EndFor For I = 1:SKNum,  StationKeeping Loop
```

Part III

Appendices

Chapter 22

Variable Definitions

Table 22.1: Variable Definitions

Script	Symbol	Name	Description	Dependency
X	x		x-component of position	CS
Y	y		y-component of position	CS
Z	z		z-component of position	CS
VX	$\dot{x}$		x-component of velocity	CS
VY	$\dot{y}$		y-component of velocity	CS
VZ	$\dot{z}$		z-component of velocity	CS
RMAG	r	r	Magnitude of the position vector, $\|\mathbf{r}\|$	CB
RA	λ	Right Ascension	The between the projection of $\mathbf{r}$ into the $x-y$ and the x-axis. Measured counter-clockwise.	CS
DEC	δ	Declination	The angle between $\mathbf{r}$ and the $x-y$ plane measured in the plane formed by $\mathbf{r}$ and $\hat{\mathbf{z}}$.	CS
VMAG	v	v	Magnitude of the velocity vector, $\|\mathbf{v}\|$	
FPA	ψ	Flight path angle	The angle measured from a plane normal to $\mathbf{r}$ to the velocity vector $\mathbf{v}$, measured in the plane formed by $\mathbf{r}$ and $\mathbf{v}$	
AZI	α_f	Flight path azimuth	The angle measured from vector perpendicular $\mathbf{r}$ and pointing north, to the projection of $\mathbf{v}$ into a plane normal to $\mathbf{r}$.	
SMA	a	semimajor axis	The semimajor contains information on the type and size of an orbit. If $a > 0$ the orbit is elliptic. If $a < 0$ the orbit is hyperbolic.	
ECC	e	eccentricity	The eccentricity contains information on the shape of an orbit. If $e = 0$, then the orbit is circular. If $0 < e < 1$ the orbit is elliptical. If $e = 1$ the orbit is parabolic. If $e > 1$ then the orbit is hyperbolic.	
INC	i	inclination	The inclination is the angle between the $\hat{\mathbf{z}}_I$ axis and the orbit normal direction $\mathbf{h}$. If $i \leq 90°$ then the orbit is prograde. If $i > 90°$ then the orbit is retrograde.	
AOP	ω	argument of periapsis	The argument of periapsis is the angle between a vector pointing at periapsis, $\mathbf{x}_p$, and a vector pointing at the spacecraft. The argument of periapsis is undefined for circular orbits.	

Table 22.2: Variable Definitions

Script	Symbol	Name	Description	Dependency
RAAN	Ω	right ascension of the ascending node	Ω is defined as the angle between $\hat{\mathbf{x}}_I$ and $\mathbf{N}$ measured counterclockwise. $\mathbf{N}$ is defined as the vector pointing from the center of the central body to the spacecraft, when the spacecraft crosses the bodies equatorial plane from the southern to the northern hemisphere. Ω is undefined for equatorial orbits.	
TA	ν	true anomaly	The true anomaly is defined as the angle between a vector pointing at periapsis, $\mathbf{x}_p$, and a vector pointing at the spacecraft. The true anomaly is undefined for circular orbits.	
RadPer	r_p	radius of periapsis	The radius of periapsis is the radius at the spacecrafts closest approach to the central body. The radius of periapsis must be greater than zero, parabolic orbits are not currently supported.	CB
VelApoapsis	v_a	velocity at apoapsis	The velocity when the spacecraft is at apoapsis	CB
VelPeriapsis	v_a	velocity at periapsis	The velocity when the spacecraft is at periapsis	CB
OrbitPeriod	T	Period of osculating Keplerian orbit.	CB	
RadApo	r_a	radius of apoapsis	Distance from central body at closest approach	CB
RadPer	r_p	radius of periapsis	Distance from central body at farthest approach	CB
C3Energy	C_3	C3 Energy	The C_3 energy is two times the kinetic energy when the spacecraft is at an infinite radius from the central body	CB
Energy	$\mathcal{E}$	Orbit energy	Orbit energy (specific energy, m^2/s^2)	CB
MM	n	Mean motion	Orbit mean angular rate	CB

Chapter 23

Brainstorm and Notes

Add Modified Julian Date to time discussion

Change finite burn scripting. Need to remove the tanks definition line from thruster and put it under Finite Burn.

TCB time system

Add in new parameters

Add spherical state conversion stuff

Add time system conversion details

Add discussion of ephemeris sources

Add that TDB is used in DE files

Add that LOD is not interpolated

Add that Polar Motion x and y are linearly interpolated.

Add file source for EOP data. polar motion, lod, dut1, dat.

Add Body1-Body2-Body3 angle to math spec

Describe source of Nutation Data and what model is being used. IERS 1996 from Vallado's fortan distribution.

Describe how GMAT handles when user sets order ¡ degree in gravity model. This needs to write message to screen probably.

Kp in drag models. Referenec Vallado Eq. 8-31 and explain where it is used.

In Barycenter section.... only celestial bodies

In Libration Points, only Barycenters and celestial bodies.

Does steve queen's code use TT instead of TDB for FK5 reduction.

Bibliography

[1] Seidelmann, P. K., "Report of the IAU/IAG Working Group on Cartographic Coordinates and Rotational Elements of the Planets and Satellites: 2000," *Celestial Mechanics and Dynamical Astronomy*, Vol. 82, No. 1, 2002, pp. 83 – 111.

[2] Vallado, D. A., *Fundamentals of Astrodynamics and Applications, 2nd Ed.*, Microcosm Press, El Segundo, CA, 2001.

[3] P. Kenneth Seidelmann, E., *Explanatory Supplement to the Astronomical Almanac*, Univesity Science Books, Mill Valley, CA, 1992.

[4] P. K. Seidelmann, T. F., "Why New Time Scales?" *Astronomy and Astrophysics*, Vol. 265, 1992, pp. 833 – 838.

[5] Szebehely, V., *Theory of Orbits*, Academic Press Inc., New York, NewYork, 1967.

Index