The Long-term Evolution of Geosynchronous Transfer Orbits

This document describes an interactive MATLAB script named `gto.m` that predicts the long-term orbital evolution of geosynchronous transfer orbits (GTO) subject to perturbations. These types of orbits are usually associated with the problem of orbital transfer from a low Earth orbit (LEO) to a geosynchronous orbit (GSO). The behavior of these types of orbits is dramatically influenced by the point-mass gravity of both the sun and moon. This script can be used to estimate the orbital lifetime of a GTO by examining the long-term evolution of the perigee altitude.

The `gto` MATLAB script implements a special perturbation solution of orbital motion using a variable step size Runge-Kutta-Fehlberg (RKF78) integration method to numerically solve Cowell's form of the system of differential equation subject to the central body gravity and other external forces. This is also called the orbital initial value problem (IVP).

The user can choose to model one or more of the following perturbations:

- non-spherical Earth gravity
- point mass solar gravity
- point mass lunar gravity

After the orbit propagation is complete, this script can plot the following classical orbital elements:

- semimajor axis
- eccentricity
- orbital inclination
- argument of perigee
- right ascension of the ascending node
- true anomaly
- geodetic perigee altitude
- geodetic apogee altitude

The user can provide the initial orbital elements for this program interactively or by specifying the name of a simple data file. The following is a typical data file for this application. When creating this type of ASCII text file, the user can change the numeric data and annotation, but do not change the number of lines in the file or the line location of the data. Please note the units and valid range for each data item.

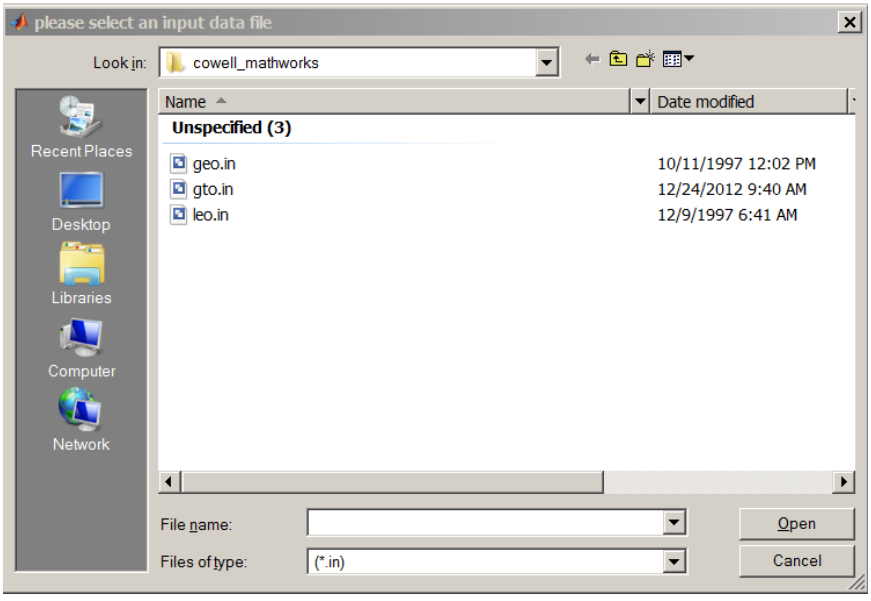
```
semimajor axis (kilometers)
(semimajor axis > 0)
24421.2363

orbital eccentricity (non-dimensional)
(0 <= eccentricity < 1)
0.72654389

orbital inclination (degrees)
(0 <= inclination <= 180)
26.2998
```

```
argument of perigee (degrees)
(0 <= argument of perigee <= 360)
0

right ascension of the ascending node (degrees)
(0 <= RAAN <= 360)
45

true anomaly (degrees)
(0 <= true anomaly <= 360)
0
```

The syntax of the MATLAB function that reads this data file is as follows:

```
function [fid, oev] = readoe1(filename)

% read orbital elements data file

% required by gto.m

% input

%  filename = name of orbital element data file

% output

%  fid = file id

%  oev(1) = semimajor axis
%  oev(2) = orbital eccentricity
%  oev(3) = orbital inclination
%  oev(4) = argument of perigee
%  oev(5) = right ascension of the ascending node
%  oev(6) = true anomaly
```

If the user elects to use a data file for orbital elements input to the software, the script will prompt for the name of the data file with a screen similar to



The file type defaults to names with a `*.in` filename extension. However, you can select any compatible ASCII data file by selecting the Files of type: field or by typing the name of the file directly in the File name: field.

To execute the `gto` script, simply type **`gto`** in the MATLAB command window.

The following is a typical user interaction with this MATLAB script. Please note the units and valid range for each input. A smaller error tolerance will predict the orbit more accurately at the expense of longer run time. This example is taken from "A Semi-analytic Technique for Predicting Orbit Lifetime of Geosynchronous Transfer Orbits", AIAA 86-2176, B. R. McCormick, P. L. Fardelos, and V. R. Bond.

```
program gto

 < long-term evolution of geosynch transfer orbits >

initial calendar date and UTC time

please input the calendar date
(1 <= month <= 12, 1 <= day <= 31, year = all digits!)
? 1,1,1984

please input the universal time
(0 <= hours <= 24, 0 <= minutes <= 60, 0 <= seconds <= 60)
? 0,0,0

please input the simulation period (days)
? 400

please input the algorithm error tolerance
(a value between 1.0e-8 and 1.0e-12 is recommended)
? 1d-10


gravity model inputs

please input the degree of the gravity model (zonals)
(0 <= zonals <= 18)
? 4

please input the order of the gravity model (tesserals)
(0 <= tesserals <= 18)
? 4

would you like to include solar point-mass gravity (y = yes, n = no)
? y

would you like to include lunar point-mass gravity (y = yes, n = no)
? y


would you like to create and display graphics (y = yes, n = no)
? y


please input the graphics step size (minutes)
? 120
```

```
 orbital elements menu

   <1> user input

   <2> data file

? 1


initial orbital elements

please input the semimajor axis (kilometers)
(semimajor axis > 0)
? 24421.14

please input the orbital eccentricity (non-dimensional)
(0 <= eccentricity < 1)
? 0.7265427

please input the orbital inclination (degrees)
(0 <= inclination <= 180)
? 28.5

please input the argument of perigee (degrees)
(0 <= argument of perigee <= 360)
? 0

please input the right ascension of the ascending node (degrees)
(0 <= raan <= 360)
? 45

please input the true anomaly (degrees)
(0 <= true anomaly <= 360)
? 0
```

The following is the script output for this example.

```
program gto

 < long-term evolution of geosynch transfer orbits >

initial calendar date       01-Jan-1984
initial UTC time      00:00:00.000

initial orbital elements and state vector

        sma (km)              eccentricity          inclination (deg)         argper (deg)
 +2.44211400000000e+004  +7.26542700000000e-001  +2.85000000000000e+001  +3.60000000000000e+002

       raan (deg)          true anomaly (deg)         arglat (deg)            period (min)
 +4.50000000000000e+001  +0.00000000000000e+000  +0.00000000000000e+000  +6.33007171476290e+002

        rx (km)                 ry (km)                rz (km)                rmag (km)
 +4.72215737778379e+003  +4.72215737778378e+003  +0.00000000000000e+000  +6.67813900732200e+003

        vx (kps)                vy (kps)               vz (kps)               vmag (kps)
 -6.30831439062031e+000  +6.30831439062031e+000  +4.84387272845975e+000  +1.01514907235997e+001


final calendar date         04-Feb-1985
final universal time        00:00:00.000
```

```
final orbital elements and state vector

        sma (km)              eccentricity          inclination (deg)         argper (deg)
 +2.43336716982214e+004  +7.28804003157905e-001  +2.78770382449471e+001  +2.37282861782308e+002

       raan (deg)          true anomaly (deg)         arglat (deg)            period (min)
 +2.59298210578186e+002  +2.23236477860796e+002  +1.00519339643105e+002  +6.29609390584718e+002

        rx (km)                 ry (km)                rz (km)                rmag (km)
 +2.15963406666680e+004  +4.37896292140369e+002  +1.11818912971504e+004  +2.43234121030487e+004

        vx (kps)                vy (kps)               vz (kps)               vmag (kps)
 -2.55353521136534e+000  +2.70839628071372e+000  -1.59326032117169e+000  +4.04900369721820e+000


degree of gravity model     4
order of gravity model      4

simulation includes lunar point-mass gravity

simulation includes solar point-mass gravity
```

The simulation summary screen display contains the following information:

rx (km) = x-component of the object's position vector in kilometers

ry (km) = y-component of the object's position vector in kilometers

rz (km) = z-component of the object's position vector in kilometers

rmag (km) = scalar magnitude of the object's position vector in kilometers

vx (km/sec) = x-component of the object's velocity vector in kilometers per second

vy (km/sec) = y-component of the object's velocity vector in kilometers per second

vz (km/sec) = z-component of the object's velocity vector in kilometers per second

vmag (km/sec) = scalar magnitude of the object's velocity vector in kilometers per second

sma (km) = semimajor axis in kilometers

eccentricity = orbital eccentricity (non-dimensional)

inclination (deg) = orbital inclination in degrees

argper (deg) = argument of perigee in degrees

raan (deg) = right ascension of the ascending node in degrees

true anomaly (deg) = true anomaly in degrees

arglat (deg) = argument of latitude in degrees. The argument of latitude is the sum of true anomaly and argument of perigee.

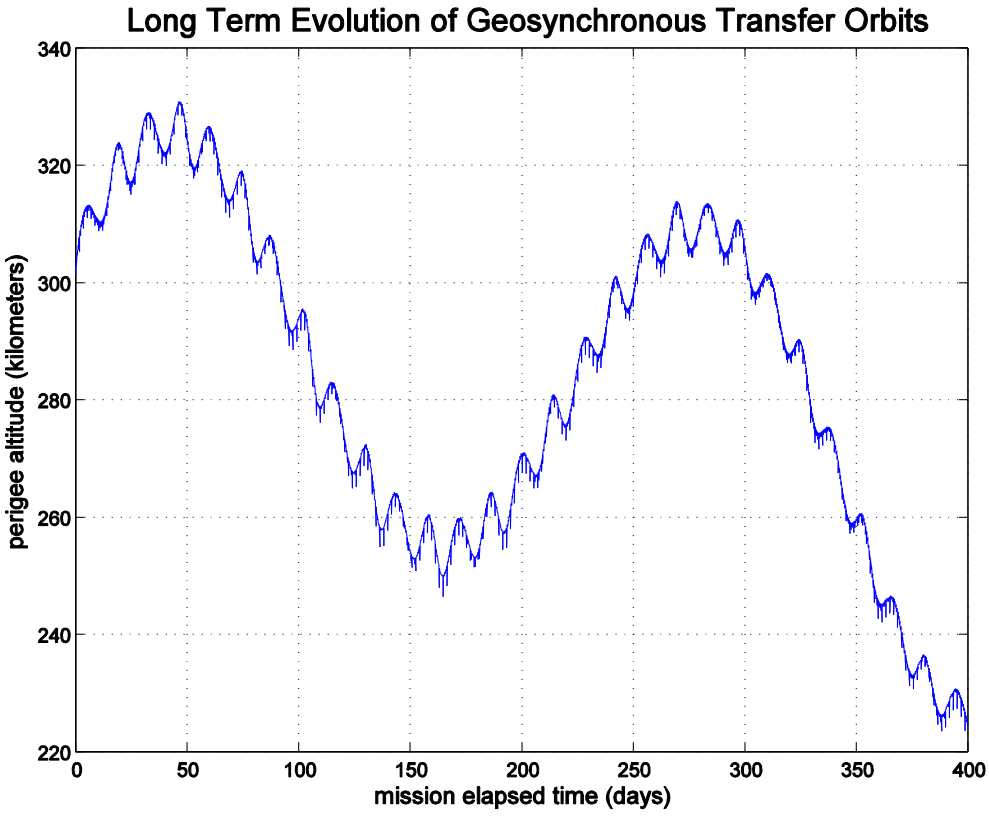
period (min) = orbital period in minutes

After the numerical integration is complete, the user can also elect to graphically display the evolution of several important orbital elements. The prompt and typical user response for this option is as follows:

```
please select the item to plot

  <1> semimajor axis
```

```
  <2> eccentricity

  <3> orbital inclination

  <4> argument of perigee

  <5> right ascension of the ascending node

  <6> true anomaly

  <7> geodetic perigee altitude

  <8> geodetic apogee altitude

? 7
```

The following is the graphics display of the geodetic perigee altitude for this example.



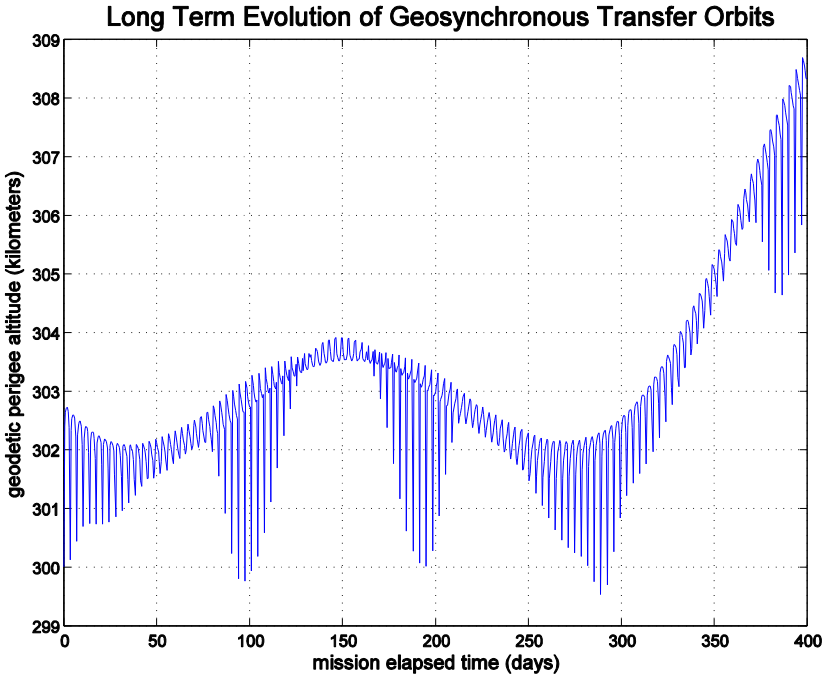
The `gto` MATLAB script will also create a color Postscript disk file with the following code.

```
print -depsc -tiff -r300 gto1.eps
```

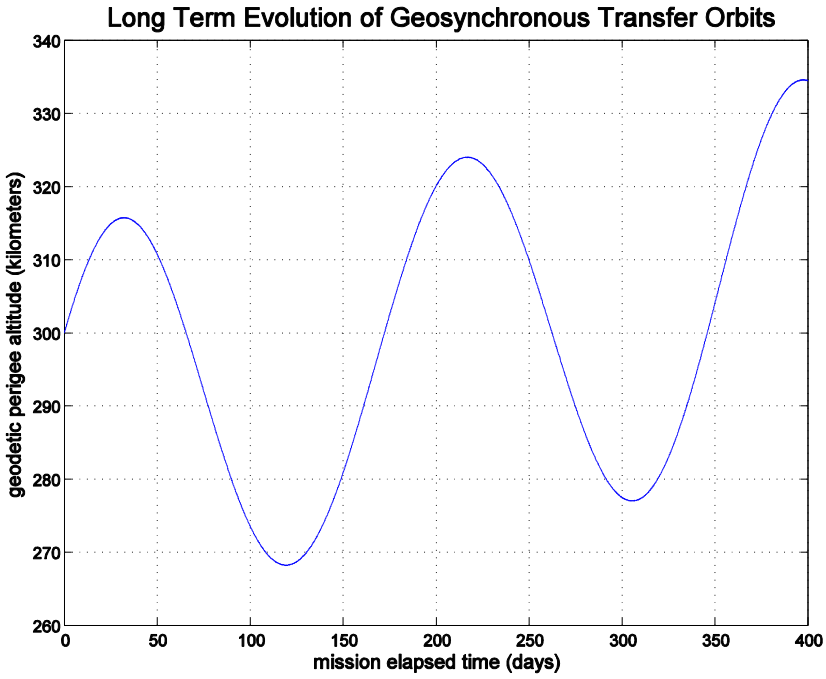
After the first graph is created and displayed, the user can create additional graphic plots by responding with `y` to the following program prompt:

```
would you like to create another plot (y = yes, n = no)
```

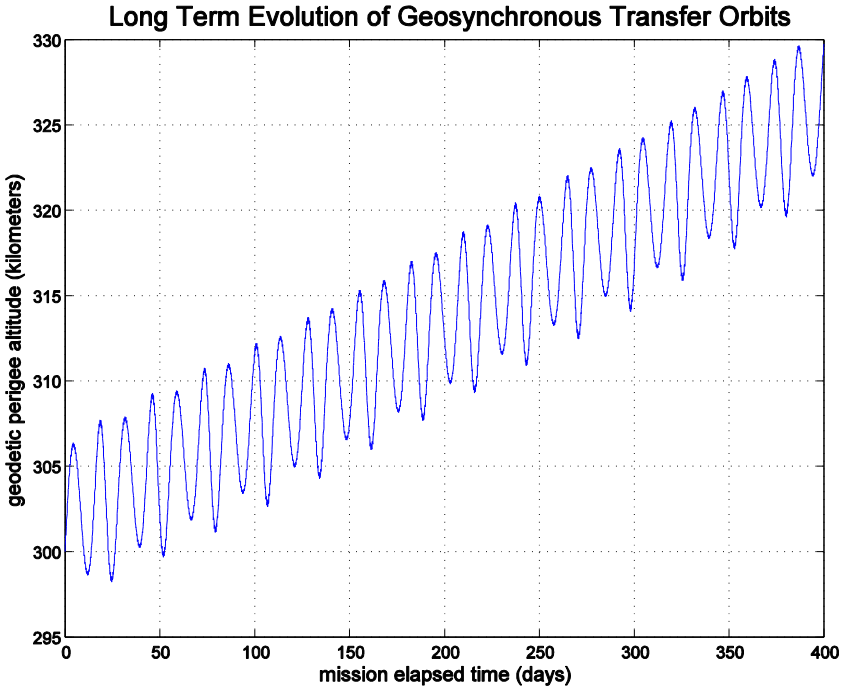
It is also possible to examine the effect of individual perturbations on the evolution of a GTO. For example, the following is the graphics display of the long-term evolution of the geodetic perigee altitude due to non-spherical Earth gravity only.



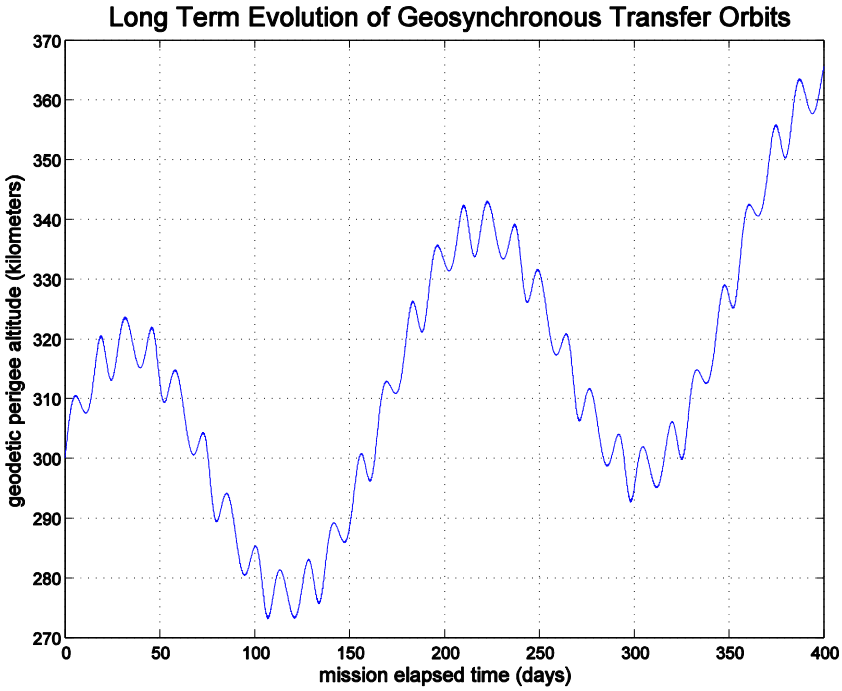
The following is the graphics display of the long-term evolution of the geodetic perigee altitude due to spherical Earth gravity and the point-mass gravity of the sun only.



This next graphic display is the long-term evolution of the geodetic perigee altitude due to spherical Earth gravity and the point-mass gravity of the moon only.



This final graphic display is the long-term evolution of the geodetic perigee altitude due to spherical Earth gravity and the point-mass gravity of the sun and moon.



Technical Discussion

Cowell's method is a *special perturbation* technique that numerically integrates the vector system of second-order, nonlinear differential equations of motion of a satellite given by

$$\mathbf{a}(\mathbf{r},\mathbf{v},t) = \ddot{\mathbf{r}}(\mathbf{r},\dot{\mathbf{r}},t) = \mathbf{a}_g(\mathbf{r}) + \mathbf{a}_s(\mathbf{r},t) + \mathbf{a}_m(\mathbf{r},t)$$

where

t = time
$\mathbf{r}$ = inertial position vector
$\mathbf{v}$ = inertial velocity vector
$\mathbf{a}_g$ = acceleration due to Earth gravity
$\mathbf{a}_s$ = acceleration due to the point-mass gravity of the Sun
$\mathbf{a}_m$ = acceleration due to the point-mass gravity of the Moon

The orbital motion is modeled with respect to a *true-of-date* Earth-centered-inertial (ECI) coordinate system. The origin of this system is the center of the Earth and the fundamental plane is the Earth's equator. The *x*-axis is aligned with the true-of-date Vernal Equinox, the *z*-axis is aligned with the Earth's spin axis, and the *y*-axis completes this orthogonal, right-handed coordinate system.

Acceleration due to non-spherical Earth gravity

This MATLAB script uses a *spherical harmonic* representation of the Earth's geopotential function given by

$$\Phi(r,\phi,\lambda) = \frac{\mu}{r} + \frac{\mu}{r}\sum_{n=1}^{\infty} C_n^0 \left(\frac{R}{r}\right)^n P_n^0(u) + \frac{\mu}{r}\sum_{n=1}^{\infty}\sum_{m=1}^{n}\left(\frac{R}{r}\right)^n P_n^m(u)\left[S_n^m \sin m\lambda + C_n^m \cos m\lambda\right]$$

where ϕ is the geocentric latitude, λ is the geocentric east longitude and $r = |\mathbf{r}| = \sqrt{x^2 + y^2 + z^2}$ is the geocentric distance. In this expression the *S's* and *C's* are *unnormalized* harmonic coefficients of the geopotential, and the *P's* are associated Legendre polynomials of degree *n* and order *m* with argument $u = \sin\phi$.

The software calculates the acceleration due to the Earth's gravity field with a vector equation derived from the gradient of the potential function expressed as

$$\mathbf{a}_g(\mathbf{r},t) = \nabla\Phi(\mathbf{r},t)$$

This acceleration vector is a combination of pure two-body or *point mass* gravity acceleration and the gravitational acceleration due to higher order non-spherical terms in the Earth's geopotential. In terms of the Earth's geopotential Φ, the inertial rectangular Cartesian components of the acceleration vector are as follows:

$$\ddot{x} = \left(\frac{1}{r}\frac{\partial \Phi}{\partial r} - \frac{z}{r^2\sqrt{x^2+y^2}}\frac{\partial \Phi}{\partial \phi} \right) x - \left(\frac{1}{x^2+y^2}\frac{\partial \Phi}{\partial \lambda} \right) y$$

$$\ddot{y} = \left(\frac{1}{r}\frac{\partial \Phi}{\partial r} - \frac{z}{r^2\sqrt{x^2+y^2}}\frac{\partial \Phi}{\partial \phi} \right) y + \left(\frac{1}{x^2+y^2}\frac{\partial \Phi}{\partial \lambda} \right) x$$

$$\ddot{z} = \left(\frac{1}{r}\frac{\partial \Phi}{\partial r} \right) z + \left(\frac{\sqrt{x^2+y^2}}{r^2}\frac{\partial \Phi}{\partial \phi} \right)$$

The three partial derivatives of the geopotential with respect to r, ϕ, λ are given by

$$\frac{\partial \Phi}{\partial r} = -\frac{1}{r}\left(\frac{\mu}{r} \right) \sum_{n=2}^{N} \left(\frac{R}{r} \right)^n (n+1) \sum_{m=0}^{n} \left(C_n^m \cos m\lambda + S_n^m \sin m\lambda \right) P_n^m (\sin \phi)$$

$$\frac{\partial \Phi}{\partial \phi} = \left(\frac{\mu}{r} \right) \sum_{n=2}^{N} \left(\frac{R}{r} \right)^n \sum_{m=0}^{n} \left(C_n^m \cos m\lambda + S_n^m \sin m\lambda \right) \left[P_n^{m+1} (\sin \phi) - m \tan \phi \, P_n^m (\sin \phi) \right]$$

$$\frac{\partial \Phi}{\partial \lambda} = \left(\frac{\mu}{r} \right) \sum_{n=2}^{N} \left(\frac{R}{r} \right)^n \sum_{m=0}^{n} m \left(S_n^m \cos m\lambda - C_n^m \sin m\lambda \right) P_n^m (\sin \phi)$$

where

$R =$ radius of the Earth
$r =$ geocentric distance
$S_n^m, C_n^m =$ harmonic coefficients
$\phi =$ geocentric latitude $= \sin^{-1}(z/r)$
$\lambda =$ geographic longitude $= \alpha - \alpha_g$
$\alpha =$ right ascension $= \tan^{-1}(r_y / r_x)$
$\alpha_g =$ right ascension of Greenwich

Right ascension is measured positive east of the vernal equinox, longitude is measured positive east of Greenwich, and latitude is positive above the Earth's equator and negative below.

For $m = 0$, the coefficients are called *zonal* terms, when $m = n$ the coefficients are *sectorial* terms, and for $n > m \neq 0$ the coefficients are called *tesseral* terms.

The Legendre polynomials with argument $\sin \phi$ are computed using recursion relationships given by:

$$P_n^0(\sin\phi) = \frac{1}{n}\left[(2n-1)\sin\phi\, P_{n-1}^0(\sin\phi) - (n-1)P_{n-2}^0(\sin\phi)\right]$$

$$P_n^n(\sin\phi) = (2n-1)\cos\phi\, P_{n-1}^{n-1}(\sin\phi), \qquad m \neq 0,\ m < n$$

$$P_n^m(\sin\phi) = P_{n-2}^m(\sin\phi) + (2n-1)\cos\phi\, P_{n-1}^{m-1}(\sin\phi), \qquad m \neq 0,\ m = n$$

where the first few associated Legendre functions are given by

$$P_0^0(\sin\phi) = 1, \quad P_1^0(\sin\phi) = \sin\phi, \quad P_1^1(\sin\phi) = \cos\phi$$

and $P_i^j = 0$ for $j > i$.

The trigonometric arguments are determined from expansions given by

$$\sin m\lambda = 2\cos\lambda\sin(m-1)\lambda - \sin(m-2)\lambda$$

$$\cos m\lambda = 2\cos\lambda\cos(m-1)\lambda - \cos(m-2)\lambda$$

$$m\tan\phi = (m-1)\tan\phi + \tan\phi$$

This script is "hard-wired" to use an EGM96 gravity model data file. The name of the data file can be changed by editing the following section of the main `gto.m` script.

```
% read gravity model coefficients

gmfile = 'egm96.dat';


[ccoef, scoef] = readgm(gmfile);
```

The following are the first 14 lines of the 18 by 18 `egm96.dat` gravity model file. Column 1 is the degree *l*, column 2 is the order *m*, column 3 is the *C* coefficients and the last column contains the *S* gravity model coefficients.

```
 2     0    -1.08262668355E-003    0.00000000000E+000
 3     0     2.53265648533E-006    0.00000000000E+000
 4     0     1.61962159137E-006    0.00000000000E+000
 5     0     2.27296082869E-007    0.00000000000E+000
 6     0    -5.40681239107E-007    0.00000000000E+000
 7     0     3.52359908418E-007    0.00000000000E+000
 8     0     2.04799466985E-007    0.00000000000E+000
 9     0     1.20616967365E-007    0.00000000000E+000
10     0     2.41145438626E-007    0.00000000000E+000
11     0    -2.44402148325E-007    0.00000000000E+000
12     0     1.88626318279E-007    0.00000000000E+000
13     0     2.19788001661E-007    0.00000000000E+000
14     0    -1.30744533118E-007    0.00000000000E+000
```

Gravity model coefficients are often published in *normalized* form. The relationship between normalized $\bar{C}_{l,m}, \bar{S}_{l,m}$ and un-normalized gravity coefficients $C_{l,m}, S_{l,m}$ is given by the following expression:

$$\begin{Bmatrix} \bar{C}_{l,m} \\ \bar{S}_{l,m} \end{Bmatrix} = \left[\frac{1}{(2-\delta_{m0})(2l+1)} \frac{(l+m)!}{(l-m)!} \right]^{1/2} \begin{Bmatrix} C_{l,m} \\ S_{l,m} \end{Bmatrix}$$

where δ_{m0} is equal to 1 if *m* is zero and equal to zero if *m* is greater than zero.

Acceleration due to the sun and moon

The acceleration contribution of the moon represented by a *point mass* is given by

$$\vec{a}_m(\vec{r},t) = -\mu_m \left(\frac{\vec{r}_{m-b}}{|\vec{r}_{m-b}|^3} + \frac{\vec{r}_{e-m}}{|\vec{r}_{e-m}|^3} \right)$$

where

$\mu_m =$ gravitational constant of the moon
$\vec{r}_{m-b} =$ selenocentric position vector
$\vec{r}_{e-m} =$ position vector from the Earth to the moon

Likewise, the acceleration contribution of the sun represented by a *point mass* is given by

$$\vec{a}_s(\vec{r},t) = -\mu_s \left(\frac{\vec{r}_{s-b}}{|\vec{r}_{s-b}|^3} + \frac{\vec{r}_{e-s}}{|\vec{r}_{e-s}|^3} \right)$$

where

$\mu_s =$ gravitational constant of the sun
$\vec{r}_{s-b} =$ heliocentric position vector
$\vec{r}_{e-s} =$ position vector from the Earth to the sun

To avoid numerical problems, use is made of Richard Battin's $f(q)$ function given by

$$f(q_k) = q_k \left[\frac{3+3q_k+q_k^2}{1+\left(\sqrt{1+q_k}\right)^3} \right]$$

where

$$q_k = \frac{\mathbf{r}^T(\mathbf{r}-2\mathbf{s}_k)}{\mathbf{s}_k^T\mathbf{s}_k}$$

The point-mass acceleration due to *n* gravitational bodies can now be expressed as

$$\ddot{\mathbf{r}} = -\sum_{k=1}^{n} \frac{\mu_k}{d_k^3}\left[\mathbf{r} + f\left(q_k\right)\mathbf{s}_k\right]$$

In this equation, $\mathbf{s}_k$ is the vector from the primary body to the secondary body *k*, μ_k is the gravitational constant of the secondary body and $\mathbf{d}_k = \mathbf{r} - \mathbf{s}_k$, where $\mathbf{r}$ is the position vector of the spacecraft relative to the primary body.

The solar and lunar ephemerides used in this program are computer implementations of the numerical methods described in *Low-Precision Formulae for Planetary Positions*, T. C. Van Flandern and K. F. Pulkkinen, *The Astrophysical Journal Supplement Series*, **41**:391-411, November 1979.

Fundamental constants

The fundamental constants used in the `gto` script are defined by the user in a MATLAB function named `om_constants.m`. The following is the source code for a typical function. Please note the proper units and the use of a MATLAB `global` statement to make this information available to other functions in a MATLAB application.

```
function om_constants

% astrodynamic and utility constants

% Orbital Mechanics with MATLAB

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

global dtr rtd mu mmu smu omega req flat j2 aunit

dtr = pi / 180.0;

rtd = 180.0 / pi;

% earth gravitational constant (km**3/sec**2)

mu = 398600.436233;

% moon gravitational constant (km**3/sec**2)

mmu = 4902.800076;

% sun gravitational constant (km**3/sec**2)

smu = 132712440040.944;

% earth inertial rotation rate (radians/second)

omega = 7.292115486e-5;

% earth equatorial radius (kilometers)

req = 6378.1363;

% earth flattening factor (non-dimensional)

flat = 1.0 / 298.257;

% earth oblateness gravity coefficient (non-dimensional)
```

```
j2 = 0.00108263;

% astronomical unit (kilometers)

aunit = 149597870.691;
```

Another resource for this type of orbital motion is “A Study of the Lifetimes of Geosynchronous Transfer Orbits”, AAS 79-105, Otis F. Graf, Jr. and Alan C. Mueller.