

Methods of Orbit Design

This document describes a suite of MATLAB scripts that can be used to design and analyze special types of Earth orbits. Numerical methods for both preliminary and high fidelity design of repeating ground track, sun-synchronous, frozen and composite orbits are discussed. Several scripts are also provided for mission design and analysis of satellites in geosynchronous orbits.

Repeating Ground Track Orbits

This section describes four MATLAB scripts that can be used to design and analyze repeating ground track orbits. Scripts are provided for both preliminary and high fidelity orbit design.

The repeating ground track design equation is

$$\frac{\left(\frac{N}{K}\right)}{\omega_e - \dot{\Omega}} - \frac{1}{\dot{\omega} + \dot{M}} = 0$$

where

K = integer number of orbits in repeat cycle

N = integer number of days in repeat cycle

ω_e = inertial rotation rate of the Earth

$\dot{\Omega}$ = RAAN perturbation

$\dot{\omega}$ = argument of perigee perturbation

\dot{M} = mean anomaly perturbation

The perturbations of argument of perigee and mean anomaly due to J_2 are given by the next two equations:

$$\dot{M} = \tilde{n} = \frac{dM}{dt} = n \left\{ 1 + \frac{3}{2} J_2 \left(\frac{r_{eq}}{p} \right)^2 \sqrt{1 - e^2} \left(1 - \frac{3}{2} \sin^2 i \right) \right\}$$

$$\dot{\omega} = \frac{d\omega}{dt} = \frac{3}{2} J_2 \tilde{n} \left(\frac{r_{eq}}{p} \right)^2 \left(2 - \frac{5}{2} \sin^2 i \right)$$

where

$n = \sqrt{\mu / a^3}$ = mean motion

$p = a(1 - e^2)$ = semiparameter

a = semimajor axis

e = orbital eccentricity

Orbital Mechanics with MATLAB

i = orbital inclination
 J_2 = second gravity harmonic of the Earth
 r_{eq} = equatorial radius of the Earth

repeat1.m – time to repeat ground track – Kozai orbit propagation

This MATLAB script estimates the time required for an Earth satellite to repeat its ground track. The satellite is propagated using Kozai’s algorithm and the user can select a closure tolerance.

The algorithm begins by initializing the Earth-relative longitude of the ascending node and the total number of days according to

$$\lambda_{an} = 0 \quad ndays = 0$$

The nodal period is computed using the expression

$$\tau_n = \frac{2\pi}{(\tilde{n} + \dot{\omega})}$$

where \tilde{n} is the “perturbed” mean motion and $\dot{\omega}$ is the perturbation of the argument of perigee due to Earth oblateness.

The delta-longitude at the ascending node per nodal period is given by

$$\Delta\lambda = \tau_n (\omega_e - \dot{\Omega})$$

where ω_e is the inertial rotation rate of the Earth and $\dot{\Omega}$ is the perturbation of the right ascension of the ascending node due to Earth oblateness.

The current Earth relative longitude and number of orbits are incremented according to

$$\lambda_{i+1} = \lambda_i + \Delta\lambda$$
$$norbits = norbits + 1$$

After each increment, convergence is checked. If $|\lambda - 2\pi| \leq \varepsilon$ or $|\lambda| \leq \varepsilon$ the method has satisfied the user-defined closure tolerance ε . The total number of days to repeat the ground track is determined from $ndays = norbits \tau_n / 86400$.

Orbital Mechanics with MATLAB

The following is a typical user interaction with this script.

```
program repeat1

< time to repeat ground track - analytic solution >

please input the semimajor axis (kilometers)
(semimajor axis > 0)
? 8000

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

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

please input the closure tolerance (degrees)
(a value between 0.1 and 0.5 degrees is recommended)
? .1
```

The following is the output screen generated with this application.

```
program repeat1

< time to repeat ground track - analytic solution >

mean semimajor axis          8000.000000 kilometers
mean eccentricity (nd)       0.000000
mean inclination             28.500000 degrees
mean argument of perigee    0.000000 degrees
mean raan                   0.000000 degrees

number of orbits to repeat   2075.000000
number of solar days to repeat 170.653126

Keplerian period             118.684684 minutes
nodal period                 118.429158 minutes

length of nodal day         1420.466169 minutes
fundamental interval        30.014440 degrees

closure tolerance            0.100000 degrees
actual closure               0.036832 degrees
```

repeat2.m – time to repeat ground track - numerical integration

This application estimates the time required for a satellite to repeat its ground track. The satellite's orbit is propagated using numerical integration and the user can select a closure tolerance. This script is similar to `repeat1` with the time to each ascending node determined by numerical integration.

Orbital Mechanics with MATLAB

The following is a typical user interaction with this script.

```
program repeat2

< time to repeat ground track - integrated solution >

please input the semimajor axis (kilometers)
(semimajor axis > 0)
? 8000

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

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

please input the closure tolerance (degrees)
(a value between 0.1 and 0.5 degrees is recommended)
? .1
```

The output created for this example is as follows:

```
program repeat2

< time to repeat ground track - integrated solution >

semimajor axis          8000.000000  kilometers
eccentricity (nd)      0.000000
inclination             28.500000  degrees
argument of perigee    0.000000  degrees
raan                    0.000000  degrees
Keplerian period       118.684684  minutes
nodal period           118.386712  minutes

time to repeat         0.986556  days
number of orbits to repeat  12.000000
```

repeat3.m – required mean semimajor axis - Wagner's algorithm

This MATLAB script calculates the mean semimajor axis required for a repeating ground track orbit using an algorithm devised by Carl Wagner. This method is described in “A Prograde Geosat Exact Repeat Mission?”, *The Journal of the Astronautical Sciences*, Vol. 39, No. 3, July-September 1991, pp. 313-326.

Orbital Mechanics with MATLAB

This algorithm starts with the following initial guess for the required mean semimajor axis

$$a_0 = \mu^{1/3} \left[\left(\frac{R}{D} \right) \omega_e \right]^{-2/3}$$

and iteratively improves this guess with the following update:

$$a_{i+1} = \mu^{1/3} \left[\left(\frac{R}{D} \right) \omega_e \right]^{-2/3} \left[1 - \frac{3}{2} J_2 \left(\frac{r_e}{a_i} \right)^2 \left(1 - \frac{3}{2} \sin^2 i \right) \right]^{2/3} \left[1 + J_2 \left(\frac{r_e}{a_i} \right)^2 \left\{ \frac{3}{2} \left(\frac{R}{D} \right) \cos i - \frac{3}{4} (5 \cos^2 i - 1) \right\} \right]^{2/3}$$

where

R = integer number of orbits

D = integer number of synodic (or nodal) days

J_2 = second gravity coefficient

ω_e = inertial rotation rate of the Earth

r_{eq} = equatorial radius of the Earth

i = orbital inclination

μ = gravitational constant of the Earth

The equation for the nodal day is given by

$$T_N = \frac{2\pi}{\omega_e - \dot{\Omega}}$$

where $\dot{\Omega}$ is the perturbation of the right ascension of the ascending node. The relationship between R and D is as follows:

$$\frac{R}{D} = \frac{\tilde{n} + \dot{\omega}}{\omega_e - \dot{\Omega}}$$

where \tilde{n} is the perturbed mean motion and $\dot{\omega}$ is the perturbation in argument of perigee.

The following example illustrates a typical user interaction with this script.

```
program repeat3
< repeating ground track - Wagner's method >

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

please input the orbital inclination (degrees)
```

Orbital Mechanics with MATLAB

```
(0 <= inclination <= 180)
? 108
```

```
please input the number of orbits in the repeat cycle
? 271
```

```
please input the number of nodal days in the repeat cycle
? 19
```

The following is the solution calculated by this script.

```
program repeat3

    < repeating ground track - Wagner's method >

mean semimajor axis          7192.231056 kilometers
mean eccentricity            0.000000
mean inclination              108.000000 degrees
mean argument of perigee     0.000000 degrees
mean raan                     0.000000 degrees

number of orbits to repeat   271.000000
number of solar days to repeat 19.054818

Keplerian period             101.170791 minutes
nodal period                  101.250693 minutes

length of nodal day          1444.154622 minutes
fundamental interval         25.239852 degrees
```

repeat4.m – required osculating semimajor axis - numerical integration solution

This application calculates the osculating semimajor axis required for a repeating ground track orbit. The script will ask the user for a semimajor axis initial guess and the J_2 – *perturbed* orbit is propagated using numerical integration during the solution process. Additional perturbations can easily be included in the equations of motion.

The program begins by setting the initial right ascension of the ascending node (RAAN) to zero and the initial true anomaly to the ascending node ($\theta = -\omega$).

The bracketing interval for the required osculating semimajor axis a is equal to

$$a_0 - 100 \leq a \leq a_0 + 100$$

where a_0 is the initial semimajor axis guess provided by the user.

The main nonlinear equation solved by this script is given by

$$f(t) = FI - \lambda_{an} = 0$$

Orbital Mechanics with MATLAB

where λ_{an} is the east longitude of the nodal crossing and FI is called the *fundamental interval* which is equal to

$$FI = 2\pi \frac{ndays}{norbits}$$

In this equation $ndays$ is the integer number of days in the user-defined repeat cycle and $norbits$ is the integer number of orbits in the cycle.

By solving the $f(t)$ equation, we are trying to minimize the difference between the required fundamental interval and the east longitude change caused by Earth rotation during one nodal period. While solving for the root of $f(t)$, the algorithm also propagates the orbit to the next ascending node crossing by solving the following equation:

$$g(t) = r_z = 0$$

subject to the mission constraint $v_z > 0$.

The following is an example user interaction with this script.

```
program repeat4

< repeating ground track orbits - integrated solution >

please input the semimajor axis (kilometers)
(semimajor axis > 0)
? 7192

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

please input the orbital inclination (degrees)
(0 <= inclination <= 180)
? 108

please input the number of integer orbits in the repeat cycle
? 271

please input the number of integer days in the repeat cycle
? 19
```

The following is the output created by this script for this orbit design example.

```
program repeat4

< repeating ground track orbits - integrated solution >

semimajor axis          7200.5423 kilometers
```

Orbital Mechanics with MATLAB

eccentricity	0.00000000	
inclination	108.0000	degrees
argument of perigee	0.0000	degrees
raan	0.0000	degrees
nodal period	101.2510	minutes
Keplerian period	101.3462	minutes
number of orbits to repeat	271.0000	
number of days to repeat	19.0000	

Sun-synchronous Orbits

This section describes three MATLAB scripts that can be used to design and analyze sun-synchronous orbits. Scripts are provided for both preliminary and high fidelity orbit design. A sun-synchronous orbit has a nodal regression rate that approximately matches the Earth's orbital rate around the Sun. The orbit plane of a sun-synchronous maintains a fixed geometry with respect to the Earth-Sun line.

The sun synchronous design equation is

$$\cos i + \left\{ -\frac{2}{3} \left(\frac{p}{r_{eq}} \right)^2 \frac{\dot{\lambda}}{nJ_2} \right\} = 0$$

where

$\dot{\lambda}$ = orbital rate of the Earth (≈ 0.985 degrees/day)

a = semimajor axis

e = orbital eccentricity

$p = a(1 - e^2)$

i = orbital inclination

μ = gravitational constant of the Earth

J_2 = second zonal gravity harmonic of the Earth

r_{eq} = equatorial radius of the Earth

$n = \sqrt{\mu/a^3}$ = mean motion

sunsync1.m – required mean orbital inclination – Kozai j2 solution

This MATLAB script calculates the *mean* orbital inclination required for a sun-synchronous Earth orbit.

Orbital Mechanics with MATLAB

This algorithm starts with an initial guess for the orbital inclination given by

$$i_0 = \cos^{-1} \left\{ -\frac{2}{3} \left(\frac{p}{r_{eq}} \right)^2 \frac{\dot{\lambda}}{nJ_2} \right\}$$

It then computes the perturbed mean motion based on this value of inclination using the following expression:

$$\tilde{n} = \frac{dM}{dt} = n \left\{ 1 + \frac{3}{2} J_2 \left(\frac{r_{eq}}{p} \right)^2 \sqrt{1-e^2} \left(1 - \frac{3}{2} \sin^2 i_0 \right) \right\}$$

A new guess for inclination is

$$i_{n+1} = \cos^{-1} \left\{ -\frac{2}{3} \left(\frac{p}{r_{eq}} \right)^2 \frac{\dot{\lambda}}{\tilde{n}J_2} \right\}$$

This iteration continues until convergence, $|i_{n+1} - i_n| \leq \varepsilon$ where ε is a convergence criterion and is equal to 1.0d-8 in this script..

The following is a typical user interaction with this program.

```
program sunsync1
    < sun-synchronous orbits - j2 solution >
    <1> input mean semimajor axis and eccentricity
    <2> input mean perigee and apogee altitudes
    selection (1 or 2) ? 2
    please input the mean perigee altitude (kilometers)? 350
    please input the mean apogee altitude (kilometers)? 1000
```

The following is the screen display created by this program.

```
program sunsync1
    < sun-synchronous orbits - j2 solution >
    mean perigee altitude (kilometers)      350.0000
    mean apogee altitude (kilometers)      1000.0000
    mean semimajor axis (kilometers)       7053.1400
```

```

mean orbital eccentricity          0.0460787678
mean orbital inclination (degrees)  98.0571
number of iterations                2.0000
    
```

sunsync2.m – required mean orbital inclination – Kozai j2+j4 solution

This application calculates the mean orbital inclination required for a sun-synchronous orbit. It uses a $J_2 + J_4$ form of Kozai’s method during the solution process.

This script uses Brent’s method to find a real root of the following nonlinear sun-synchronous *constraint* equation:

$$f(i) = \dot{\lambda} - \dot{\Omega} = 0$$

where $\dot{\lambda} = 2\pi / 365.2422$ is the orbital rate of the Earth around the Sun and $\dot{\Omega}$ is the first-order secular perturbation in the right ascension of the ascending node given by

$$\dot{\Omega} = \frac{d\Omega}{dt} = -\frac{3}{2} J_2 \tilde{n} \left(\frac{r_{eq}}{p} \right)^2 \cos i \left[1 + \frac{3}{2} J_2 \left(\frac{r_{eq}}{p} \right)^2 \left\{ \frac{3}{2} + \frac{e^2}{6} - 2\sqrt{1-e^2} - \left(\frac{5}{3} - \frac{5}{24} e^2 - 3\sqrt{1-e^2} \right) \sin^2 i \right\} \right] - \frac{35}{8} J_4 \left(\frac{r_{eq}}{p} \right)^4 \tilde{n} \left(1 + \frac{3}{2} e^2 \right) \left(\frac{12 - 21 \sin^2 i}{14} \right) \cos i$$

The perturbed mean motion due to J_2 and J_4 is given by

$$\tilde{n} = \frac{dM}{dt} = n \left\{ \begin{array}{l} 1 + \frac{3}{2} J_2 \left(\frac{r_{eq}}{p} \right)^2 \sqrt{1-e^2} \left(1 - \frac{3}{2} \sin^2 i \right) \\ + \frac{3}{128} J_2^2 \left(\frac{r_{eq}}{p} \right)^4 \sqrt{1-e^2} \left[\begin{array}{l} 16\sqrt{1-e^2} + 25(1-e^2) - 15 \\ + [30 - 96\sqrt{1-e^2} - 90(1-e^2)] \cos^2 i \\ + [105 + 144\sqrt{1-e^2} + 25(1-e^2)] \end{array} \right] \\ - \frac{45}{128} J_4 \left(\frac{r_{eq}}{p} \right)^4 \sqrt{1-e^2} e^2 (3 - 30 \cos^2 i + 35 \cos^4 i) \end{array} \right\}$$

The bracketing interval for this algorithm is

$$i_0 + 1^\circ \leq i \leq i_0 + 1^\circ$$

where i_0 is an orbital inclination initial guess given by

Orbital Mechanics with MATLAB

$$i_0 = \cos^{-1} \left\{ -\frac{2}{3} \left(\frac{p}{r_{eq}} \right)^2 \frac{\dot{\lambda}}{nJ_2} \right\}$$

The following is a typical user interaction with this MATLAB script.

```
program sunsync2
< sun-synchronous orbits - j2 + j4 solution >
<1> input mean semimajor axis and eccentricity
<2> input mean perigee and apogee altitudes
selection (1 or 2) ? 2
please input the mean perigee altitude (kilometers)? 350
please input the mean apogee altitude (kilometers)? 1000
```

The following is the screen display created by this program for this example.

```
program sunsync2
< sun-synchronous orbits - j2 + j4 solution >
mean perigee altitude      350.0000 kilometers
mean apogee altitude      1000.0000 kilometers
mean semimajor axis       7053.1400 kilometers
mean orbital eccentricity  0.0460787678
mean orbital inclination   98.0306 degrees
```

sunsync3.m – required osculating orbital inclination – numerical integration solution

This MATLAB script calculates the *osculating* orbital inclination required for a sun-synchronous orbit. The orbit is propagated by numerical integration during the solution process, and the software allows the user to specify the degree and order of the gravity model to use during the simulation. This script can also include the point mass gravity perturbation of the Sun during the solution process. This permits high fidelity orbit design of a sun-synchronous orbit and minimizes the number and magnitude of orbit maintenance maneuvers.

The program begins by setting the initial right ascension of the ascending node (RAAN) to zero and the initial true anomaly to the ascending node ($\theta = -\omega$).

The bracketing interval for the required osculating orbital inclination i is equal to

Orbital Mechanics with MATLAB

$$i_0 - 5^\circ \leq i \leq i_0 + 5^\circ$$

where i_0 is the initial inclination guess provided by the user.

The main nonlinear equation solved by this script is given by

$$f(t) = \dot{\lambda} - \left\{ \frac{\Omega(t_0) - \Omega(t)}{t - t_0} \right\} = 0$$

where the quantity in the braces is the *finite-difference numerical* derivative of RAAN. By solving this equation, we are trying to match the required heliocentric orbital rate of the Earth, $\dot{\lambda}$, with the perturbed motion of the orbit plane, $\dot{\Omega}$.

While solving for the roots of $f(t)$, the algorithm is also propagating the orbit to the times of ascending node crossings by solving the following equation

$$g(t) = r_z = 0$$

subject to the mission constraint $v_z > 0$. This constraint ensures that the solution is found at an ascending node.

The simulation duration of this application is defined by the user in terms of the number of nodal periods. More nodal periods will provide “better” orbit design at the expense of longer computer run times.

The following is a typical user interaction with this program.

```
program sunsync3
< sun-synchronous orbits - integrated solution >

please input the calendar date
(1 <= month <= 12, 1 <= day <= 31, year = all digits!)
? 1,1,1998

please input the simulation duration (nodal periods)
? 10

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

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

would you like to include solar perturbations (y = yes, n = no)
? y
```

Orbital Mechanics with MATLAB

```
please input the semimajor axis (kilometers)
(semimajor axis > 0)
? 8000

please input the orbital eccentricity (non-dimensional)
(0 <= eccentricity < 1)
? .015

please input the orbital inclination (degrees)
(0 <= inclination <= 180)
? 98.75

please input the argument of perigee (degrees)
(0 <= argument of perigee <= 360)
? 270
```

The following is the screen display created by this program.

```
program sunsync3

< sun-synchronous orbits - integrated solution >

semimajor axis          8000.0000 kilometers
eccentricity            0.01500000
inclination             102.6105 degrees
argument of perigee    270.0000 degrees
raan                    0.0000 degrees
nodal period           118.6088 minutes
degree of gravity model 4.0000
order of gravity model  4.0000
simulation includes solar perturbations
```

Frozen Orbits

This section describes two MATLAB scripts that can be used to design and analyze frozen orbits. A frozen orbit is characterized by the absence of long-term changes in orbital eccentricity and argument of perigee. This type of orbit maintains almost constant altitude over any particular point on a planet's surface. The design of frozen orbits involves selecting the correct value of orbital eccentricity and argument of perigee, for a given semimajor axis and orbital inclination, which satisfies the following system of nonlinear perturbation equations:

$$\frac{de}{dt} = \frac{3 J_3 r_{eq}^3}{2 p^3} (1 - e^2) n \sin i \cos \omega \left(\frac{5}{4} \sin^2 i - 1 \right) = 0$$

$$\frac{d\omega}{dt} = \frac{3 J_2 r_{eq}^2}{2 p^2} n \left(2 - \frac{5}{2} \sin^2 i \right) - \frac{3 J_3 r_{eq}^3 \sin \omega}{2 p^3 e \sin i} n \left\{ \left(\frac{5}{4} \sin^2 i - 1 \right) \sin^2 i + e^2 \left(1 - \frac{35}{4} \sin^2 i \cos^2 i \right) \right\} = 0$$

where

a = semimajor axis

e = orbital eccentricity

i = orbital inclination

ω = argument of perigee

$p = a(1 - e^2)$ = semiparameter

r_{eq} = Earth equatorial radius

μ = Earth gravitational constant

$n = \sqrt{\mu/a^3}$ = mean motion

J_2 = second gravity coefficient

J_3 = third gravity coefficient

By solving this system, the perturbing effect of the J_2 even zonal gravity harmonic on eccentricity and argument of perigee are balanced by the J_3 odd zonal gravity harmonic. For argument of perigee values equal to 90 and 270 degrees, the eccentricity perturbation vanishes.

frozen1.m – required mean orbital eccentricity

This MATLAB application determines the mean orbital eccentricity required for a frozen orbit. The user inputs the mean semimajor axis and inclination and the program calculates the eccentricity using Brent's method to solve the single constraining nonlinear equation. The program also calculates and displays the real roots of the frozen eccentricity cubic equation:

$$a_1 e^3 + a_2 e^2 + a_3 e + a_4 = 0$$

The derivation of this equation can be found in the technical report, "Frozen Orbits in the J2 + J3 Problem", by Krystyna Kiedron and Richard Cook, AAS 91-426, AAS/AIAA Astrodynamics Specialist Conference, Durango, Colorado, August 19-22, 1991.

The coefficients of this equation are given by

$$a_1 = -\frac{3}{4} n \left(\frac{R}{a} \right)^2 J_2 \sin i (1 - 5 \cos^2 i)$$

$$a_2 = \frac{3}{2} n \left(\frac{R}{a} \right)^3 J_3 \left(1 - \frac{35}{4} \sin^2 i \cos^2 i \right)$$

Orbital Mechanics with MATLAB

$$a_3 = -a_1$$

$$a_4 = \frac{3}{2}n\left(\frac{R}{a}\right)^3 J_3 \sin^2 i \left(\frac{5}{4}\sin^2 i - 1\right)$$

In these equations n is the mean motion, R is the equatorial radius of the Earth, a is the semimajor axis, and i is the orbital inclination.

The following is a typical user interaction application with this software.

```
program frozen1
< orbital eccentricity of frozen orbits >

mean orbital elements

please input the semimajor axis (kilometers)
(semimajor axis > 0)
? 8000

please input the orbital inclination (degrees)
(0 <= inclination <= 180)
? 45

mean orbital elements

      sma (km)      eccentricity      inclination (deg)      argper (deg)
8.0000000000e+003  6.5941377284e-004  4.5000000000e+001  9.0000000000e+001

      raan (deg)      true anomaly (deg)      arglat (deg)      period (min)
0.0000000000e+000  0.0000000000e+000  9.0000000000e+001  1.1868468430e+002

roots of the frozen eccentricity cubic equation

ans =

-1.00241917246590
 0.99758348478212
 0.00065941377284
```

frozen2.m – long-term evolution of frozen orbits

This application allows the user to analyze the long-term behavior of frozen orbits by numerically integrating the equations of orbital motion subject to zonal gravity perturbations. The software can model up to 18 zonal coefficients and the user can plot the long-term behavior of eccentricity versus time, argument of perigee versus time, or argument of perigee versus orbital eccentricity.

The steps involved in this algorithm are as follows:

Orbital Mechanics with MATLAB

- (1) user inputs mean orbital elements
- (2) convert to osculating orbital elements and then to state vector
- (3) propagate the second-order “zonal” orbital equations of motion
- (4) convert the state vector at each step to mean orbital elements
- (5) plot the user-requested orbital elements

The following is a typical user interaction with this MATLAB script.

```
program frozen2

< orbital motion of frozen orbits >

please input the simulation period (days)
? 1000

please input the algorithm step size (minutes)
(a value between 1 and 2 is recommended)
? 2

please input the number of zonals to include
(0 <= zonals <= 18)
? 4

please input the graphics step size (days)
? 10

initial mean orbital elements

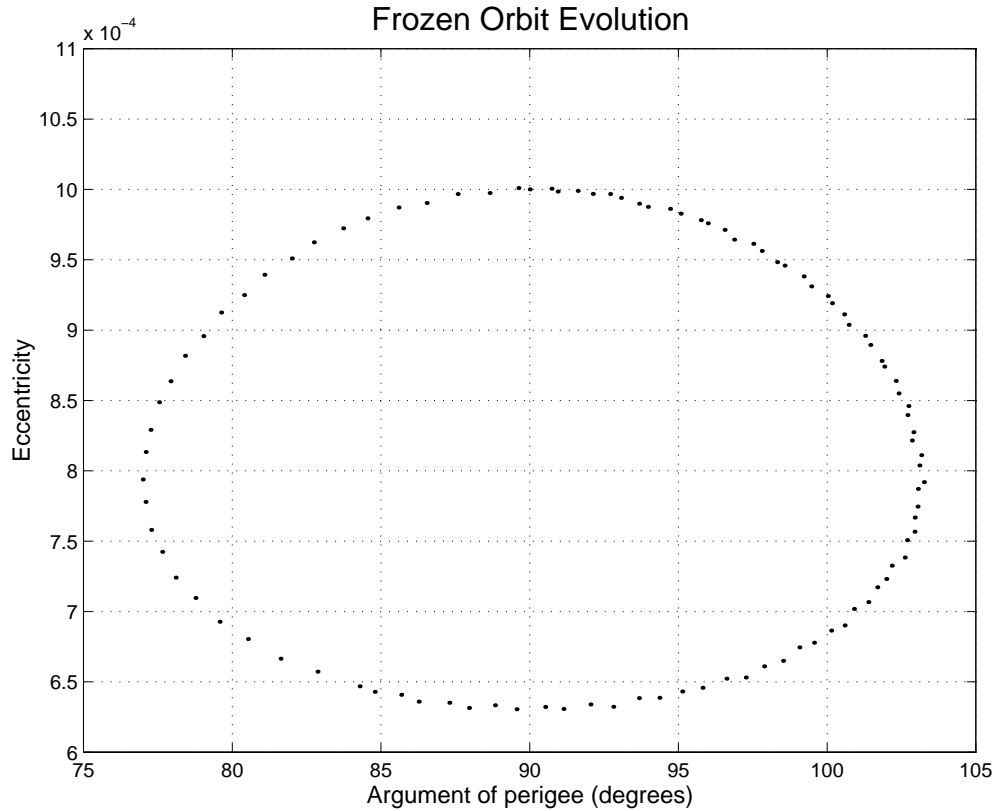
please input the semimajor axis (kilometers)
(semimajor axis > 0)
? 8000

please input the orbital eccentricity (non-dimensional)
(0 <= eccentricity < 1)
? .001

please input the orbital inclination (degrees)
(0 <= inclination <= 180)
? 60

please input the argument of perigee (degrees)
(0 <= argument of perigee <= 360)
? 90
```

The following is a graphics display of argument of perigee versus orbital eccentricity for this example. The orbit evolution for this example is clockwise in this display starting from the top ($e = 0.001, \omega = 90^\circ$). We can see that the frozen orbit eccentricity for this example is about $8e-4$ for an argument of perigee of 90 degrees. The step size for this example was 50 days and the simulation had a very long run time.



Composite Orbits

ssrepeat.m – sun-synchronous, repeating ground track orbit

This MATLAB application can be used to design sun-synchronous, repeating ground track, orbits. The nonlinear system of orbit design equations is solved to find the mean classical orbital elements of the composite orbit. The orbital element perturbation equations used in this application are based on Kozai's method.

The repeating ground track governing equation is

$$\frac{\left(\frac{N}{K}\right)}{\omega_e - \dot{\Omega}} - \frac{1}{\dot{\omega} + \dot{M}} = 0$$

The sun synchronous design equation is

$$\cos i + \frac{2\lambda a^{3/2} (1 - e^2)^2}{3J_2 r_{eq}^2 \sqrt{\mu}} = 0$$

where

Orbital Mechanics with MATLAB

K = number of orbits in repeat cycle
 N = number of days in repeat cycle
 ω_e = inertial rotation rate of the Earth
 $\dot{\Omega}$ = RAAN perturbation
 $\dot{\omega}$ = argument of perigee perturbation
 \dot{M} = mean anomaly perturbation
 a = semimajor axis
 e = orbital eccentricity
 i = orbital inclination
 μ = gravitational constant of the Earth
 r_{eq} = equatorial radius of the Earth

The following is a typical example interaction with this software.

```
program ssrepeat

  < sun-synchronous, repeating ground track orbits >

  please input an initial guess for the semimajor axis (kilometers)
  ? 8000

  please input an initial guess for the inclination (degrees)
  (90 < inclination <= 180)
  ? 100

  please input the orbital eccentricity (non-dimensional)
  (0 <= eccentricity < 1)
  ? .001

  please input the argument of perigee (degrees)
  (0 <= argument of perigee <= 360)
  ? 120

  please input the number of integer orbits in the repeat cycle
  ? 271

  please input the number of integer days in the repeat cycle
  ? 19
```

The following is the screen output provided by this script.

```
program ssrepeat

  < sun-synchronous, repeating ground track orbits >

  mean semimajor axis          7176.6158 kilometers
  mean eccentricity            0.0010000000
```

Orbital Mechanics with MATLAB

mean orbital inclination	98.5964	degrees
mean argument of perigee	120.0000	degrees
Keplerian period	100.8415	minutes
nodal period	100.9594	minutes
number of orbits in repeat cycle	271.0000	
number of days in repeat cycle	19.0000	
ground trace repetition factor	14.2632	

composit.m – sun-synchronous, repeating ground track, frozen orbit design

This MATLAB application can be used to design sun-synchronous, repeating ground track, frozen orbits. The nonlinear system of orbit design equations is solved to find the mean classical orbital elements of the *composite* orbit. The orbital element perturbation equations used in this application are based on Kozai’s method.

The frozen orbit design equations are

$$\frac{de}{dt} = 0 \text{ and } \frac{d\omega}{dt} = 0$$

The repeating ground track governing equation is

$$\frac{N}{\omega_e - \dot{\Omega}} - \frac{1}{\dot{\omega} + \dot{M}} = 0$$

The sun synchronous design equation is

$$\cos i + \frac{2\dot{\lambda}a^{3/2}(1-e^2)^2}{3J_2r_{eq}^2\sqrt{\mu}} = 0$$

where

- K = number of orbits in repeat cycle
- N = number of days in repeat cycle
- ω_e = inertial rotation rate of the Earth
- $\dot{\Omega}$ = RAAN perturbation
- $\dot{\omega}$ = argument of perigee perturbation
- \dot{M} = mean anomaly perturbation

Orbital Mechanics with MATLAB

a = semimajor axis

e = orbital eccentricity

i = orbital inclination

μ = gravitational constant of the Earth

r_{eq} = equatorial radius of the Earth

The perturbations of eccentricity and argument of perigee are given by

$$\dot{e} = \frac{de}{dt} = \frac{3 J_3 r_{eq}^3}{2 p^3} (1 - e^2) n \sin i \cos \omega \left(\frac{5}{4} \sin^2 i - 1 \right)$$

$$\dot{\omega} = \frac{d\omega}{dt} = \frac{3 J_2 r_{eq}^2}{2 p^2} n \left(2 - \frac{5}{2} \sin^2 i \right) - \frac{3 J_3 r_{eq}^3 \sin \omega}{2 p^3 e \sin i} n \left\{ \left(\frac{5}{4} \sin^2 i - 1 \right) \sin^2 i + e^2 \left(1 - \frac{35}{4} \sin^2 i \cos^2 i \right) \right\}$$

The following is a typical user interaction with this MATLAB script.

```
program composit
< sun-synchronous, repeating ground track, frozen orbits >

please input an initial guess for the semimajor axis (kilometers)
? 7100

please input an initial guess for the eccentricity (non-dimensional)
(0 <= eccentricity < 1)
? .001

please input an initial guess for the inclination (degrees)
(90 < inclination <= 180)
? 98

please input the number of integer orbits in the repeat cycle
? 271

please input the number of integer days in the repeat cycle
? 19
```

The following is a typical output screen generated with this program:

```
program composit
< sun-synchronous, repeating ground track, frozen orbits >
mean semimajor axis           7176.6158 kilometers
mean eccentricity              0.0010308455
mean orbital inclination       98.5964 degrees
```

Orbital Mechanics with MATLAB

mean argument of perigee	90.0000	degrees
keplerian period	100.8415	minutes
nodal period	100.9594	minutes
number of orbits in repeat cycle	271.0000	
number of days in repeat cycle	19.0000	
ground trace repetition factor	14.2632	

Geosynchronous Orbits

This section describes several MATLAB scripts that are useful for performing mission design and analysis for Earth satellites in geosynchronous orbits. Geosynchronous satellites are usually in equatorial or low-inclination orbits with an orbital period very close to the time required for the Earth to complete one inertial rotation.

The algorithms implemented in these MATLAB scripts are based on the numerical techniques described in the following three articles:

“East-west Stationkeeping Requirements of Nearly Synchronous Satellites Due to Earth’s Triaxiality and Luni-Solar Effects”, A. Kamel, D. Ekman and R. Tibbitts, *Celestial Mechanics* **8** (1973) 129-148.

“On the Orbital Eccentricity Control of Synchronous Satellites”, A. Kamel and C. Wagner, *The Journal of the Astronautical Sciences*, Vol. XXX, No. 1, pp. 61-73, January-March, 1982.

“On the Propagation and Control of Geosynchronous Satellites”, C.C. Chao and J.M. Baker, *The Journal of the Astronautical Sciences*, Vol. XXXI, No. 1, pp. 99-115, January-March, 1983.

geosync1.m – equilibrium longitudes and radii of geosynchronous satellites

This MATLAB script calculates the *equilibrium* longitudes and radii of geosynchronous satellites based on the EGM96 gravity model of degree (zonals) and order (tesserals) 3. These are the four Earth-relative locations where the proper combination of geocentric radius and east longitude will minimize the longitudinal drift of satellites located at these points.

The following is the output created by this script.

```
program geosync1
< equilibrium longitudes, radii and acceleration >
```

Orbital Mechanics with MATLAB

location	east longitude (degrees)	radius (kilometers)	drift acceleration (degrees/day ²)
1	75.0602	42166.2409	-2.2287387749e-012
2	162.0816	42166.2847	7.5875859013e-012
3	255.0880	42166.2411	2.0642563986e-012
4	348.5962	42166.2811	4.3719224598e-012

drift cycle period at longitude 75.0602 degrees = 2339.2602 days

drift cycle period at longitude 255.0880 degrees = 2883.5951 days

The very small values of drift acceleration confirm that these east longitude locations are indeed equilibrium points relative to a triaxial Earth.

The fundamental astrodynamic constants of the EGM96 gravity model are:

$$r_{eq} = \text{Earth equatorial radius} = 6378.1363 \text{ km}$$

$$\mu = \text{Earth gravitational constant} = 398600.4415 \text{ km}^3 / \text{sec}^2$$

$$\omega = \text{Earth inertial rotation rate} = 7.292115 \cdot 10^{-5} \text{ radians/sec}$$

The spherical harmonic and longitude coefficients of the EGM96 gravity model are determined from the following expressions:

$$J_{nm} = -\sqrt{C_{nm}^2 + S_{nm}^2}$$

$$\lambda_{nm} = \frac{1}{m} \tan^{-1} \left(\frac{S_{nm}}{C_{nm}} \right)$$

where n is the degree and m is the order of these terms.

The first few unnormalized gravity coefficients are given by

$$C_{22} = 1.57446037456 \cdot 10^{-6}$$

$$S_{22} = -9.03803806639 \cdot 10^{-7}$$

⋮

$$C_{31} = 2.19263852917 \cdot 10^{-6}$$

$$S_{31} = 2.68424890297 \cdot 10^{-7}$$

The geocentric radius of a synchronous satellite is given by

$$a_s = a_{sk} + \delta a_s$$

where a_{sk} is the Keplerian or unperturbed geosynchronous semimajor axis given by

$$a_{sk} = \left(\frac{\mu}{\omega_e^2} \right)^{1/3}$$

In this equation μ is the gravitational constant of the Earth and ω_e is the inertial rotation rate of the Earth. The “correction” of the Keplerian semimajor axis due to the triaxial shape of the Earth and a gravity model of degree and order 3 for a satellite located at an east longitude denoted by λ_s is given by the expression

$$\begin{aligned} \delta a_s = & 2J_{20}a_{sk} \left(\frac{r_{eq}}{a_{sk}} \right)^2 + 12J_{22}a_{sk} \left(\frac{r_{eq}}{a_{sk}} \right)^2 \cos 2(\lambda_s - \lambda_{22}) \\ & - 8J_{31}a_{sk} \left(\frac{r_{eq}}{a_{sk}} \right)^3 \cos 3(\lambda_s - \lambda_{31}) + 80J_{33}a_{sk} \left(\frac{r_{eq}}{a_{sk}} \right)^3 \cos 3(\lambda_s - \lambda_{33}) \end{aligned}$$

The equilibrium longitudes are the points at which the ratio g_1/g_2 is equal to zero. The “g” factors g_1 and g_2 are determined from the following two equations:

$$\begin{aligned} g_1 = & 6J_{22} \left(\frac{r_{eq}}{a_{sk}} \right)^2 \sin 2(\lambda_s - \lambda_{22}) - \frac{3}{2}J_{31} \left(\frac{r_{eq}}{a_{sk}} \right)^3 \sin(\lambda_s - \lambda_{31}) \\ & + 45J_{33} \left(\frac{r_{eq}}{a_{sk}} \right)^3 \sin 3(\lambda_s - \lambda_{33}) \\ g_2 = & \frac{\partial g_1}{\partial \lambda_s} = 12J_{22} \left(\frac{r_{eq}}{a_{sk}} \right)^2 \cos 2(\lambda_s - \lambda_{22}) - \frac{3}{2}J_{31} \left(\frac{r_{eq}}{a_{sk}} \right)^3 \cos(\lambda_s - \lambda_{31}) \\ & + 135J_{33} \left(\frac{r_{eq}}{a_{sk}} \right)^3 \cos 3(\lambda_s - \lambda_{33}) \end{aligned}$$

The *normalized* longitudinal acceleration can be determined from the following expression:

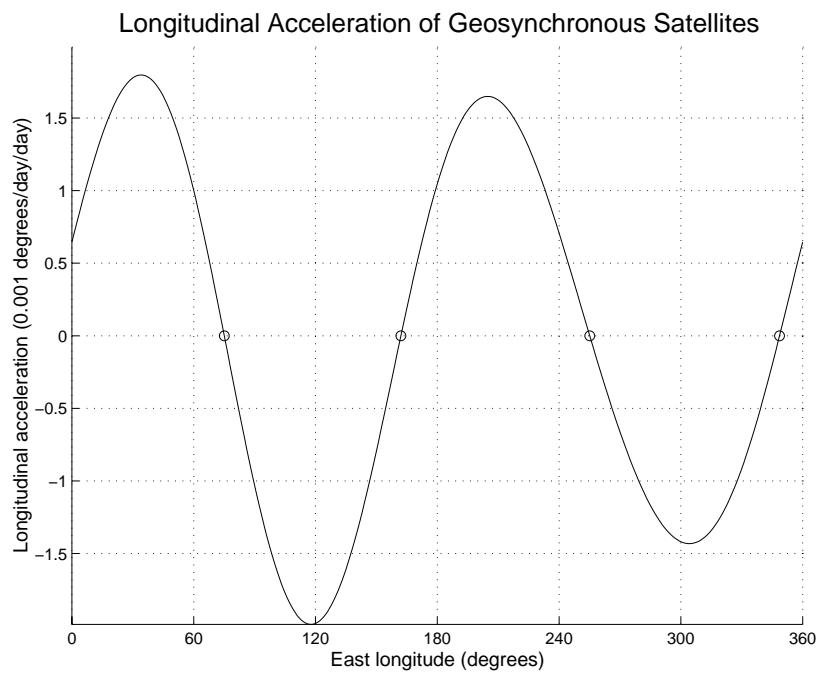
$$\ddot{\lambda} = 18 \{ c_{22} \sin 2(\lambda_s - \lambda_{22}) - 0.25c_{31} \sin(\lambda_s - \lambda_{31}) + 7.5c_{33} \sin 3(\lambda_s - \lambda_{33}) \}$$

In this equation $c_{nm} = J_{nm} \left(r_{eq}/a_s \right)^n$. The dimensional longitude acceleration is $\ddot{\lambda} \omega_e^2$. The equilibrium longitudes are determined using Brent’s root-finding algorithm.

Equilibrium longitudes with negative g_1 values are stable while those with positive g_1 values are unstable. The equilibrium longitudes located at 75.0602° and 255.088° are stable points with drift cycle periods given by

$$\tau = \frac{\pi}{\sqrt{-3g_2}}$$

After this MATLAB script has calculated the characteristics of the equilibrium points, it will ask if you would like to create and display a plot of the longitudinal acceleration as a function of east longitude of the satellite. This information can be used to graphically verify the calculations described in this section. The following is the graphics display for this program option. The location of each equilibrium point computed by this script is marked by a small circle.



geosync2.m – required osculating semimajor axis

This MATLAB script can be used to estimate the initial osculating semimajor axis required for an Earth satellite in geosynchronous orbit. The calculations include perturbations due to the point-mass gravity of the Sun and Moon and non-spherical Earth gravity. The user can also elect to include the effect of solar radiation pressure in the calculations. The script allows the user to specify the degree and order of the Earth gravity model to use during the solution.

The numerical method implemented in this script uses a combination of one-dimensional root-finding and numerical integration of the orbital equations of motion to determine the initial osculating semimajor axis that will result in a satellite orbit with a geosynchronous period subject to the perturbations mentioned above.

The one-dimensional objective function used during the root-finding calculations is

$$f(t) = \lambda_m - \lambda_{m_0}$$

where λ_m is the mean east longitude of the satellite's subpoint at any time t and λ_{m_0} is mean east longitude at the initial time.

The true east longitude of the satellite subpoint at any time is given by

$$\lambda_t(t) = \omega + \Omega + \theta - \alpha_g$$

and the mean east longitude is

$$\lambda_m(t) = \omega + \Omega + M - \alpha_g$$

In these two equations ω is the argument of perigee, Ω is the right ascension of the ascending node, θ is the satellite's true anomaly, M is the mean anomaly and α_g is the Greenwich sidereal time at any simulation time t .

The relationship between mean and true anomaly is given by Kepler's equation for elliptic orbits,

$$M = E - e \sin E$$

via the following intermediate calculations for eccentric anomaly:

$$\sin E = \sin \theta \sqrt{1 - e^2}$$

$$\cos E = e + \cos \theta$$

$$E = \tan^{-1}(\sin E, \cos E)$$

The inverse tangent calculation in the last equation is a four quadrant operation.

The orbital period of a geosynchronous satellite in seconds is given by

$$\tau = \frac{2\pi}{\omega_e}$$

where ω_e is the inertial rotation rate of the Earth in radians per second. The algorithm numerically integrates the equations of motion for one or more orbital periods, evaluates the mean east longitude objective function and uses Brent's root-finding method to drive the difference to a small tolerance. The user can specify how many orbital periods to use during the solution process. More orbital periods will provide a better semimajor axis solution at the expense of longer computation times. A value between 1 and 5 orbits is recommended.

Orbital Mechanics with MATLAB

During the solution the algorithm uses a bracketing interval for the required osculating semimajor axis a given by

$$a_0 - 100 \leq a \leq a_0 + 100$$

where a_0 is the initial semimajor axis guess (in kilometers) provided by the user.

The `geosync2.m` script will begin by prompting the user for the initial calendar date and universal time. It will then ask for the degree and order of the gravity model and which perturbations to include in the simulation. Finally, the script will prompt the user for the initial orbital elements of the satellite. If the initial orbit is not equatorial (orbital inclination > 0), the software assumes that the true east longitude input by the user is at the ascending node.

The following is a typical user interaction with this MATLAB script.

```
program geosync2

  < geosynchronous osculating semimajor axis >

initial calendar date and time

please input the calendar date
(1 <= month <= 12, 1 <= day <= 31, year = all digits!)
? 1,1,1998

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

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

orbital perturbations

would you like to include solar perturbations (y = yes, n = no)
? y

would you like to include lunar perturbations (y = yes, n = no)
? y

would you like to include srp perturbations (y = yes, n = no)
? y
```

Orbital Mechanics with MATLAB

solar radiation pressure inputs

please input the reflectivity constant (non-dimensional)
? 1.85

please input the cross-sectional area (square meters)
? 10

please input the spacecraft mass (kilograms)
? 2000

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

orbital elements menu

<1> user input

<2> data file

? 1

initial orbital elements

please input the semimajor axis (kilometers)
(semimajor axis > 0)
? 42160

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

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

please input the satellite's east longitude (degrees)
(0 <= east longitude <= 360)
? 45

please input the number of orbits to model
(integer >= 1)
? 5

The software will display the initial and final east longitude and geodetic latitude. This information indicates how well the orbit “closed” at the solution. The following is the program output created for this example.

Orbital Mechanics with MATLAB

program geosync2

```
< geosynchronous osculating semimajor axis >

initial calendar date      01-Jan-1998
initial universal time     00:00:00

initial mean east longitude 45.000000 degrees
final mean east longitude  45.000000 degrees

initial geodetic latitude  0.000000 degrees
final geodetic latitude    0.272835 degrees

reflectivity constant      1.850000

cross-sectional area       10.000000  sqr meters

spacecraft mass            2000.000000 kilograms

degree of gravity model    4.0
order of gravity model     4.0

number of orbits modeled   5.0

initial orbital elements

      sma (km)      eccentricity      inclination (deg)      argper (deg)
4.2166908088e+004  0.0000000000e+000  0.0000000000e+000  0.0000000000e+000

      raan (deg)      true anomaly (deg)      arglat (deg)      period (min)
0.0000000000e+000  1.4544413882e+002  1.4544413882e+002  1.4362080817e+003
```

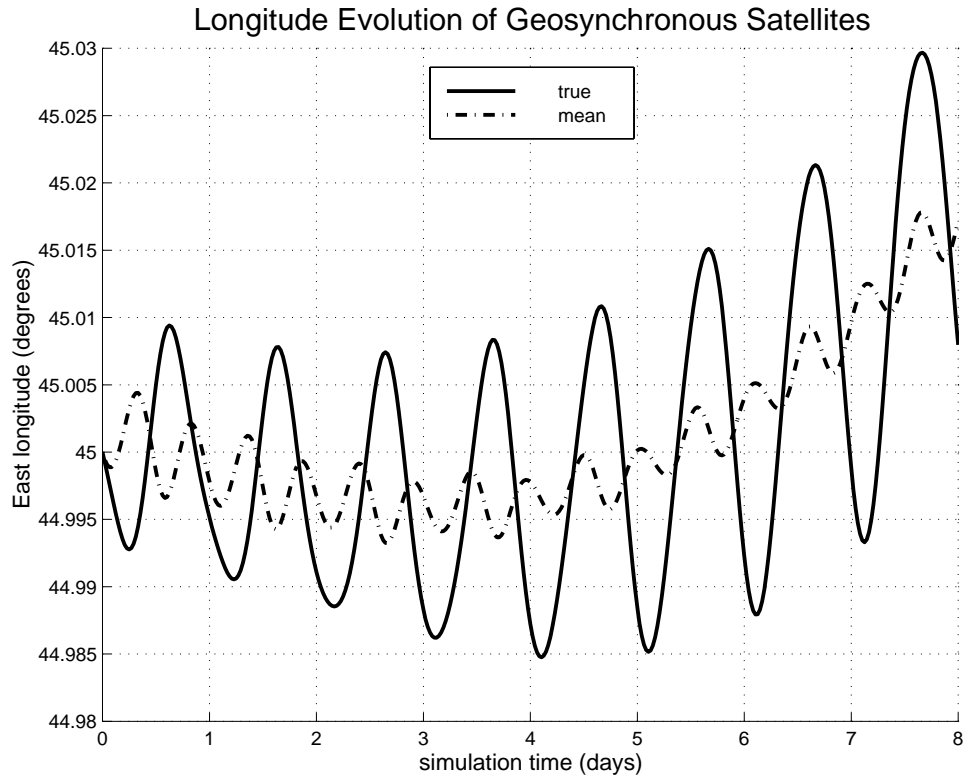
After the script solves the problem it will ask the user if he or she would like to create a graphics display of the evolution of the satellite's east longitude. A typical user interaction with this program option is as follows:

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

please input the simulation period (days)
? 8

please input the graphics step size (minutes)
? 30
```

The following is a plot of the long-term evolution of both the true and mean east longitude for this example. It illustrates that the secular variation of the east longitude is small and only starts to “drift” after about 5 days which is the prediction period used in this example.



geosync3.m – repositioning a geosynchronous satellite

This MATLAB script determines the impulsive maneuvers required to reposition a geosynchronous satellite. Repositioning is the process of moving a geosynchronous satellite in longitude to the east or west relative to some initial location. This orbit modification is performed by creating an elliptical *drift* orbit with one impulsive maneuver and then applying a second impulse that re-establishes a circular geosynchronous orbit at the “target” longitude.

The semimajor axis of the elliptical drift orbit is given by

$$a_d = a_s \left(1 + \frac{D}{360^\circ} \right)^{2/3}$$

where a_s is the semimajor axis of the initial geosynchronous circular orbit.

The drift rate D is given by

$$D = \Delta\lambda/n_d$$

In this equation $\Delta\lambda$ is the longitude change (+ west, – east) relative to the initial location, and n_d is the integer number of drift orbits that the spacecraft travels during the change.

The longitude drift during one drift orbit is given by

Orbital Mechanics with MATLAB

$$\Delta\lambda = \omega_e \tau_d - \omega_s \tau_d$$

In this equation ω_e is the inertial rotation rate of the Earth, ω_s is the orbital rate of the drift orbit and τ_d is the orbital period of the drift orbit. The longitude drift is caused by the difference between the Earth rotation and orbital rates during the drift period.

For a longitude change to the west, the orbital eccentricity of the drift orbit is determined from

$$e_d = 1 - \frac{a_s}{a_d}$$

The perigee velocity of the drift orbit is given by

$$V_p = \sqrt{\frac{2\mu}{a_s} - \frac{\mu}{a_d}}$$

The magnitude of the delta-v that creates the elliptical drift orbit is

$$\Delta V = V_p - V_{lc}$$

where V_{lc} is the local circular velocity of the initial geosynchronous orbit, $V_{lc} = \sqrt{\mu/a_s}$. This delta-v is applied in the direction of orbital motion and creates an elliptical orbit that has a longer period than the original circular orbit. After the satellite has drifted for n_d orbits, a delta-v of equal magnitude but opposite direction is applied. This delta-v re-circularizes the elliptical drift orbit and creates a geosynchronous orbit at the new longitude.

For a longitude change to the east, the orbital eccentricity of the drift orbit is determined from

$$e_d = \frac{a_s}{a_d} - 1$$

The apogee velocity of the drift orbit is given by

$$V_a = \sqrt{\frac{2\mu}{a_s} - \frac{\mu}{a_d}}$$

The magnitude of the delta-v that creates the elliptical drift orbit is

$$\Delta V = V_{lc} - V_a$$

This delta-v is applied opposite to the direction of orbital motion and creates an elliptical orbit that has a shorter period than the original circular orbit. As before a delta-v of equal magnitude

Orbital Mechanics with MATLAB

but opposite direction is applied to create a circular geosynchronous orbit at the new user-defined east longitude.

The following is a typical user interaction with this MATLAB script.

```
program geosync3

< repositioning maneuvers for gso satellites >

please input the semimajor axis (kilometers)
? 42165

please input the longitude change
(+ west, - east; degrees)
? -30

please input the number of drift orbits
? 10

semimajor axis          42165.000000  kilometers
delta-longitude        -30.000000  degrees
number of orbits        10.0
drift period            237.357151  hours

drift orbit characteristics

semimajor axis          41930.423442  kilometers
eccentricity (nd)       0.005594
perigee altitude        35317.709884  kilometers
apogee altitude         35786.863000  kilometers
keplerian period        1424.142906  minutes
total delta-v           17.224908  meters/second
```

geosync4.m – east-west stationkeeping of geosynchronous satellites

This MATLAB script can be used to determine the impulsive delta-v and drift cycle period required for east-west stationkeeping of geosynchronous satellites in equatorial orbits. The east-west stationkeeping requirement is specified by a longitude “deadband” centered about the nominal east longitude of the satellite.

A “drift” orbit is established by biasing the initial semimajor axis such that the satellite moves from this initial condition to the edge of the deadband. After reaching the edge of the deadband

Orbital Mechanics with MATLAB

the satellite then drifts toward the other side. When the satellite reaches the other end of the deadband, a single impulsive maneuver is performed to create an elliptical orbit with an apogee equal to the original semimajor axis of the drift orbit. Once the satellite reaches this new apogee, another single maneuver is performed to circularize the satellite's orbit at this radius.

The following is a typical user interaction with this script.

```
program geosync4

< east-west stationkeeping of geosynchronous satellites >

initial east longitude and deadband

please input the satellite's east longitude (degrees)
(0 <= east longitude <= 360)
? 45

please input the total deadband constraint (degrees)
? 1
```

The following is the program output for this example.

```
satellite east longitude          45.0000 degrees
total longitude deadband          1.0000 degrees
initial drift rate                 0.0575 degrees/day
single maneuver delta-v           0.3262 meters/second
total annual delta-v              1.7113 meters/second/year
drift cycle period                 69.6248 days

geosynchronous semimajor axis     42166.2534 kilometers
drift cycle semimajor axis        42170.7272 kilometers
delta semimajor axis              4.4738 kilometers
```

After the script has solved the problem it will ask the user if he or she would like to create a graphics display of the satellite's orbital motion during the stationkeeping. The following is a typical user response to this option:

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

initial calendar date and time

please input the calendar date
(1 <= month <= 12, 1 <= day <= 31, year = all digits!)
? 1,1,1998
```

Orbital Mechanics with MATLAB

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

please input the graphics step size (days)
? 1

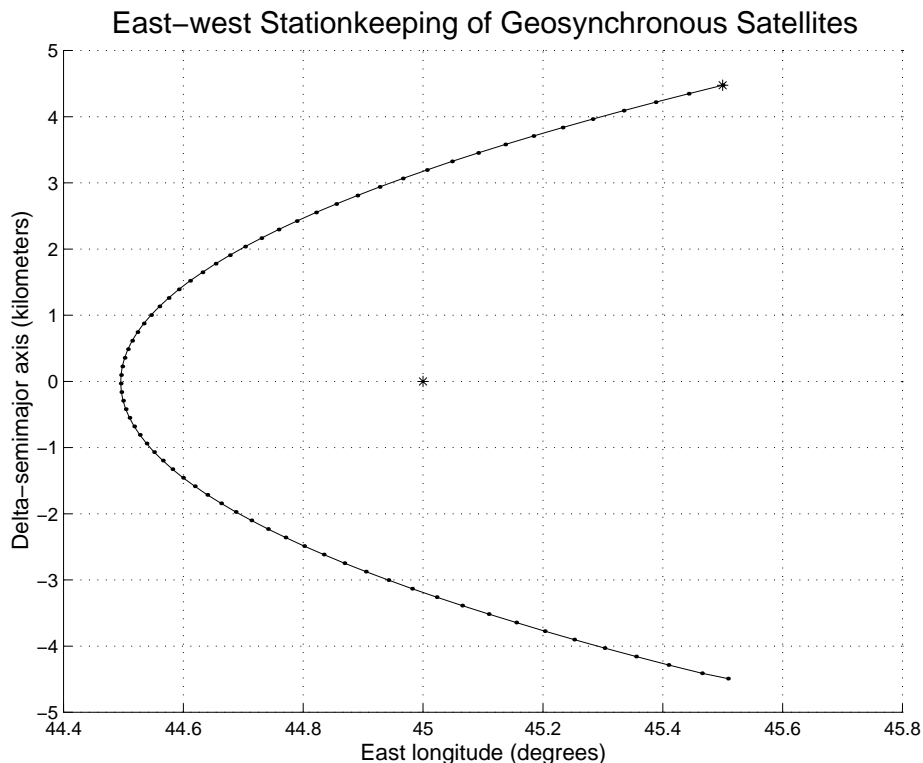
orbital perturbations

would you like to include solar perturbations (y = yes, n = no)
? n

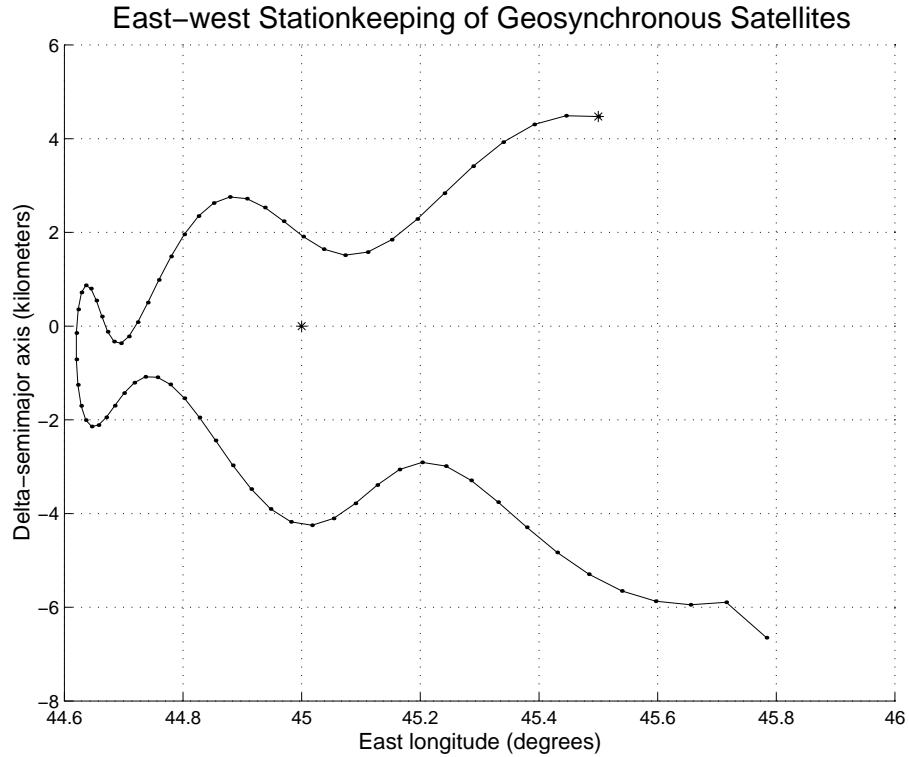
would you like to include lunar perturbations (y = yes, n = no)
? n

please input the algorithm error tolerance
(a value of 1.0e-8 is recommended)
? 1e-8
```

The following is a plot of delta-semimajor axis versus east longitude of the satellite. The delta-semimajor axis of this plot is the difference between the instantaneous semimajor axis and the synchronous semimajor axis a_s at the nominal east longitude.



The following is a plot of this example that includes the point-mass gravity perturbation of the sun and moon.



In both of these plots the “nominal” east longitude of the satellite is marked with an asterisk.

After the graphics are complete the software will display the following information based on the numerical integration of the satellite’s orbital motion.

```

program geosync4

< east-west stationkeeping of geosynchronous satellites >

integrated solution

initial calendar date      01-Jan-1998
initial universal time     00:00:00

final calendar date       11-Mar-1998
final universal time      14:59:40

final semimajor axis      42161.7620 kilometers
final eccentricity        0.000041
final east longitude      45.508808 degrees
first delta-v             0.163431 meters/second
second delta-v            0.163422 meters/second
total delta-v             0.326853 meters/second
    
```

This information can be used to verify the calculations described in the following section.

Technical Discussion

If g_1 is positive, the initial east longitude of the satellite is given by

$$\lambda_0 = \lambda_s + \frac{1}{2} \Delta\lambda$$

If g_1 is negative, the initial east longitude of the satellite is given by

$$\lambda_0 = \lambda_s - \frac{1}{2} \Delta\lambda$$

where $\Delta\lambda$ is the *total* deadband longitude constraint. The calculation of g_1 is described under the discussion for the `geosync1.m` MATLAB script.

The drift cycle period is determined with the following expression:

$$P_D \leq \frac{2}{\omega_e} \sqrt{\left| \frac{2\Delta\lambda}{\ddot{\lambda}} \right|}$$

In this equation ω_e is the inertial rotation rate of the Earth and $\ddot{\lambda}$ is the normalized longitudinal acceleration given by

$$\ddot{\lambda} = 18 \{ c_{22} \sin 2(\lambda_s - \lambda_{22}) - 0.25 c_{31} \sin(\lambda_s - \lambda_{31}) + 7.5 c_{33} \sin 3(\lambda_s - \lambda_{33}) \}$$

In this equation $c_{nm} = J_{nm} (r_{eq}/a_s)^n$. The dimensional longitude acceleration is $\ddot{\lambda} \omega_e^2$.

The spherical harmonic and longitude coefficients of the EGM96 gravity model are determined from the following expressions:

$$J_{nm} = -\sqrt{C_{nm}^2 + S_{nm}^2}$$

$$\lambda_{nm} = \frac{1}{m} \tan^{-1} \left(\frac{S_{nm}}{C_{nm}} \right)$$

where n is the degree (zonals) and m is the order (tesserals) of these terms.

The delta-v required for each two-impulse Hohmann orbit transfer is given by

$$\Delta V_D = \frac{1}{3} \omega_e V_{syn} P_D \ddot{\lambda}$$

The total annual ΔV required for east-west stationkeeping is given by

$$\Delta V_D = \frac{1}{3} \omega_e V_{syn} P_D \ddot{\lambda} \left(\frac{365.25}{P_D} \right)$$

The initial semimajor axis required for the drifting motion can be determined from

$$a_0 = a_s (1 + \eta)$$

where $\eta = \pm \frac{4}{3} \sqrt{3g_1 \text{sign}(g_1) \Delta\lambda}$ and a_s is the “reference” synchronous radius at the satellite’s nominal east longitude λ_s . The equations used to determine this radius are described in the technical discussion for the `geosync1.m` script. The positive sign in the expression for η should be used whenever g_1 is positive.

The initial drift rate of the satellite is determined from

$$\Delta\dot{\lambda}_0 = \mp 2 \sqrt{3g_1 \text{sign}(g_1) \Delta\lambda_0}$$

where the negative sign of this equation is used whenever $g_1 > 0$ and the positive sign is used whenever $g_1 < 0$.

geosync5.m – north-south stationkeeping of geosynchronous satellites

This MATLAB script can be used to graphically display the long-term orbital inclination behavior of geosynchronous satellites. It can also be used to calculate the impulsive delta-v required to correct these inclination changes. The inclination perturbation of geosynchronous satellites is caused mainly by the gravitational attraction of the sun and moon.

This script numerically integrates the Cartesian form of the equations of orbital motion. These equations include the point-mass gravity of the sun and moon and the J_2 gravity effect of the Earth. The orbital conditions at the time of a north-south correction maneuver are computed using a one-dimensional root-finder based on Brent’s method.

The delta-v required to correct the inclination change is applied at either the ascending or descending node. The scalar magnitude of this impulsive maneuver is given by

$$\Delta V = 2V \sin\left(\frac{\Delta i}{2}\right)$$

where V is the speed of the spacecraft prior to the maneuver and Δi is the inclination change.

Orbital Mechanics with MATLAB

The following is a typical user interaction with the graphics option of this script.

```
program geosync5

< north-south stationkeeping of geosynchronous satellites >

(1) create and display graphics
(2) predict time and delta-v

? 1

please input the integration error tolerance
(a value of 1.0e-8 is recommended)
? 1e-8

initial calendar date and time

please input the calendar date
(1 <= month <= 12, 1 <= day <= 31, year = all digits!)
? 1,1,2003

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

initial orbital elements

please input the semimajor axis (kilometers)
(semimajor axis > 0)
? 42164

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

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

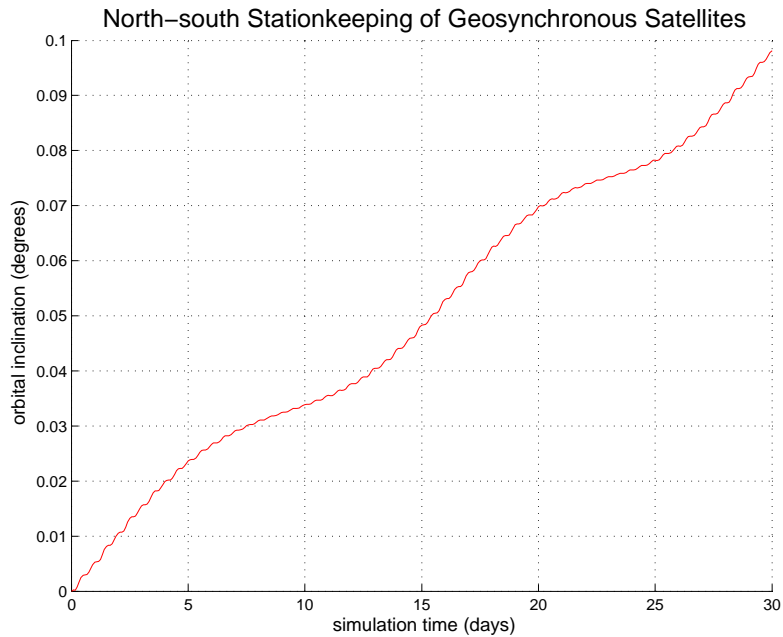
please input the satellite's east longitude (degrees)
(0 <= east longitude <= 360)
? 45

please input the simulation period (days)
? 30

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

Orbital Mechanics with MATLAB

The following is the graphics display for this example.



The following is a typical user interaction with the “predict time and delta-v” option of this script.

```
please input the integration error tolerance
(a value of 1.0e-8 is recommended)
? 1e-8
```

```
please input the root-finding error tolerance
(a value between 1.0e-4 and 1.0e-6 is recommended)
? 1e-4
```

```
initial calendar date and time
```

```
please input the calendar date
(1 <= month <= 12, 1 <= day <= 31, year = all digits!)
? 1,1,2003
```

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

```
initial orbital elements
```

```
please input the semimajor axis (kilometers)
(semimajor axis > 0)
? 42164
```

Orbital Mechanics with MATLAB

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

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

```
please input the satellite's east longitude (degrees)
(0 <= east longitude <= 360)
? 45
```

```
please input the final orbital inclination (degrees)
(0 <= inclination <= +180)
? .05
```

The following is the script output for this example.

```
final calendar date      16-Jan-2003
final universal time     09:56:22
mission elapsed time     15.4141 (days)
delta-v                  2.6834 (mps)

      sma (km)      eccentricity      inclination (deg)      argper (deg)
4.2165306913e+004  1.4971447098e-004  5.0000000000e-002  1.6714095246e+002

      raan (deg)      true anomaly (deg)      arglat (deg)      period (min)
1.0309105956e+002  3.9520052673e+001  2.0666100513e+002  1.4361262783e+003
```