

Sun-synchronous Orbit Design

This document describes three MATLAB scripts that can be used to design and analyze sun-synchronous Earth 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 heliocentric 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 based on Kozai's J_2 solution.

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 the orbital 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 1.0×10^{-8} in this script or the algorithm exceeds 100 iterations.

The following is a typical user interaction with this program. Inputs are in bold font.

```

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 Earth 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

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

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

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

The following is a typical user interaction with this MATLAB script. Inputs are in bold font.

```

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 the script 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 Earth gravity model to use during the simulation. Optionally, 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 true anomaly to the ascending node ($\theta = -\omega$).

This script uses Brent’s root-finding method to calculate the inclination. The bracketing interval for the required osculating orbital inclination i is equal to

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

where i_0 is the initial inclination guess provided by the user. An initial guess can be determined by running either the `sunsync1` or `sunsync2` MATLAB script.

The main nonlinear equation solved by this script is given by

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

where the quantity in the braces is the “averaged” *finite-difference numerical* derivative of RAAN. The denominator is the total simulation duration. 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. Here r_z is the z-component of the inertial position vector and v_z is the z-component of the inertial velocity vector.

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. The initial epoch and osculating orbital elements correspond to the situation at the initial ascending node. Inputs are shown in bold font.

```
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 Universal Coordinated Time (UTC)
(0 <= hours <= 24, 0 <= minutes <= 60, 0 <= seconds <= 60)
? 0,0,0

please input the simulation duration (nodal periods)
? 10

please input the degree of the Earth gravity model (zonals)
(2 <= zonals <= 18)
? 8

please input the order of the Earth gravity model (tesserals)
(0 <= tesserals <= 18)
? 8

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

please input the semimajor axis (kilometers)
(semimajor axis > 0)
? 7000

please input the orbital eccentricity (non-dimensional)
(0 <= eccentricity < 1)
? 0.015

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

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

please input the right ascension of the ascending node (degrees)
(0 <= raan <= 360)
? 100
```

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

```

program sunsync3

< sun-synchronous orbits - integrated solution >

semimajor axis          7000.000000  kilometers
eccentricity            0.01500000
inclination             97.846179  degrees
argument of perigee     270.000000  degrees
initial raan            100.000000  degrees
average nodal period    97.070006  minutes
desired raan_dot        0.98564733  degrees/day
predicted raan_dot      0.98564737  degrees/day
degree of gravity model      8
order of gravity model      8
number of nodal periods    10

simulation includes the point-mass gravity of the Sun

```

In this display, average nodal period is the nodal period averaged over the user-defined number of nodal periods and predicted raan_dot is the time rate-of-change of RAAN predicted by the numerical integration also averaged over the total duration of all nodal periods. The desired raan_dot is the nodal regression rate computed from the expression 360 degrees/365.2422.

Modeling the orbital motion

The sunsync3 MATLAB script implements a *special perturbation* technique which solves the vector system of second-order, nonlinear differential equations of orbital motion given by

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

where

t = dynamical time

\mathbf{r} = inertial position vector

\mathbf{a}_g = acceleration due to Earth gravity

\mathbf{a}_s = acceleration due to the Sun

Geocentric acceleration due to non-spherical Earth gravity

The `sunsync3` MATLAB script implements a *spherical harmonic* representation of the Earth's geopotential function given by

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

where ϕ is the geocentric latitude, λ is the geocentric east longitude and $r = |\vec{r}| = \sqrt{x^2 + y^2 + z^2}$ is the geocentric distance. In this expression the S 's and C 's are 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 nonspherical 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:

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

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

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

where

R = radius of the Earth

r = geocentric distance

S_n^m, C_n^m = harmonic coefficients

ϕ = geocentric declination = $\sin^{-1}\left(\frac{z}{r}\right)$

λ = longitude = $\alpha - \alpha_g$

α = right ascension = $\tan^{-1}\left(\frac{y}{x}\right)$

α_g = right ascension of Greenwich

The right ascension is measure positive east of the vernal equinox, longitude is measured positive east of Greenwich, and declination 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:

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

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

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

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

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

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

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

These gravity model data files are simple space delimited ASCII data files. The following is a portion of a typical gravity model data file. In this file, column one is the degree index, column two is the model order index, and columns three and four are the corresponding *un-normalized* gravity coefficients (zonals and tesserals, respectively).

2	0	-0.10826300D-02	0.00000000D+00
3	0	0.25321531D-05	0.00000000D+00
4	0	0.16109876D-05	0.00000000D+00
5	0	0.23578565D-06	0.00000000D+00
6	0	-0.54316985D-06	0.00000000D+00

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.

This MATLAB script is “hard-wired” to use an Earth gravity model data file named `egm96.dat`. The user can use a different data file by editing the following line of the source code.

```
% read gravity model coefficients
[cccoef, scoef] = readgm('egm96.dat');
```

Geocentric acceleration due to the point-mass gravity of the Sun

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

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

where

- μ_s = gravitational constant of the Sun
- \mathbf{r}_{s-sc} = position vector from the Sun to the trajectory
- \mathbf{r}_{e-s} = position vector from the Earth to the Sun

Order reduction

The first-order system of equations required by this computer program can be created from the second-order system by the method of *order reduction*. With the following definitions,

$$\begin{array}{lll} y_1 = r_x & y_2 = r_y & y_3 = r_z \\ y_4 = v_x & y_5 = v_y & y_6 = v_z \end{array}$$

where v_x, v_y, v_z are the velocity vector components, the first-order system of differential equations is given by

$$\begin{array}{lll} \dot{y}_1 = v_x & \dot{y}_2 = v_y & \dot{y}_3 = v_z \\ \dot{y}_4 = a_{x-e} + a_{x-s} & \dot{y}_5 = a_{y-e} + a_{y-s} & \dot{y}_6 = a_{z-e} + a_{z-s} \end{array}$$

In these equations, a_{x-e}, a_{y-e} and a_{z-e} are the x, y and z gravitational contributions due to non-spherical Earth gravity, and a_{x-s}, a_{y-s} and a_{z-s} are the x, y and z gravitational contributions of the point-mass gravity of the Sun.