

## Cowell's Method

This document describes several interactive MATLAB scripts that use Cowell's method to predict the long-term behavior of spacecraft and celestial objects subject to perturbations. This technique numerically integrates the equations of motion subject to the central body gravity and other external forces. Algorithms are included for predicting both geocentric and heliocentric orbital motion.

### **cowell1.m – perturbed orbital motion of Earth satellites**

This MATLAB script uses Cowell's method to propagate satellites in elliptic Earth orbits. The user can choose to model one or more of the following perturbations:

- non-spherical gravity
- aerodynamic drag
- solar radiation pressure
- point mass solar gravity
- point mass lunar gravity

After the orbit propagation is complete, this application 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
- geodetic altitude
- east longitude
- geodetic latitude

This script uses the Earth Gravity Model 1996 data file (`egm96.dat`) as its source of gravity coefficients. This can be changed by editing the call to the `readegm.m` function.

## *Orbital Mechanics with MATLAB*

The user can provide the initial orbital elements for this program interactively or by specifying the name of a 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)
6878.14

orbital eccentricity (non-dimensional)
(0 <= eccentricity < 1)
.0125

orbital inclination (degrees)
(0 <= inclination <= 180)
28.5

argument of perigee (degrees)
(0 <= argument of perigee <= 360)
270

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 cowell1.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
```

## Orbital Mechanics with MATLAB

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.

```
program cowell1

  < Earth orbital motion - Cowell's method >

initial calendar date and time

please input the calendar date
(1 <= month <= 12, 1 <= day <= 31, year = all digits!)
? 1,1,2000

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

please input the simulation period (days)
? 10

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

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 drag perturbations (y = yes, n = no)
? y

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

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

## *Orbital Mechanics with MATLAB*

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

orbital elements menu

<1> user input

<2> data file

? **1**

initial orbital elements

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

please input the true anomaly (degrees)  
(0 <= true anomaly <= 360)  
? **45**

aerodynamic drag inputs

please input the drag coefficient (non-dimensional)  
? **2**

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

please input the spacecraft mass (kilograms)  
? **2000**

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**

## Orbital Mechanics with MATLAB

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

```
please input the graphics step size (minutes)
? 60
```

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
<9> geodetic altitude
<10> east longitude
<11> geodetic latitude

? 2
```

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)
```

The following is the program output for this example.

```
program cowell1

< Earth orbital motion - Cowell's method >

initial calendar date      01-Jan-2000
initial universal time     00:00:00.000

final calendar date       11-Jan-2000
final universal time      00:00:00.000

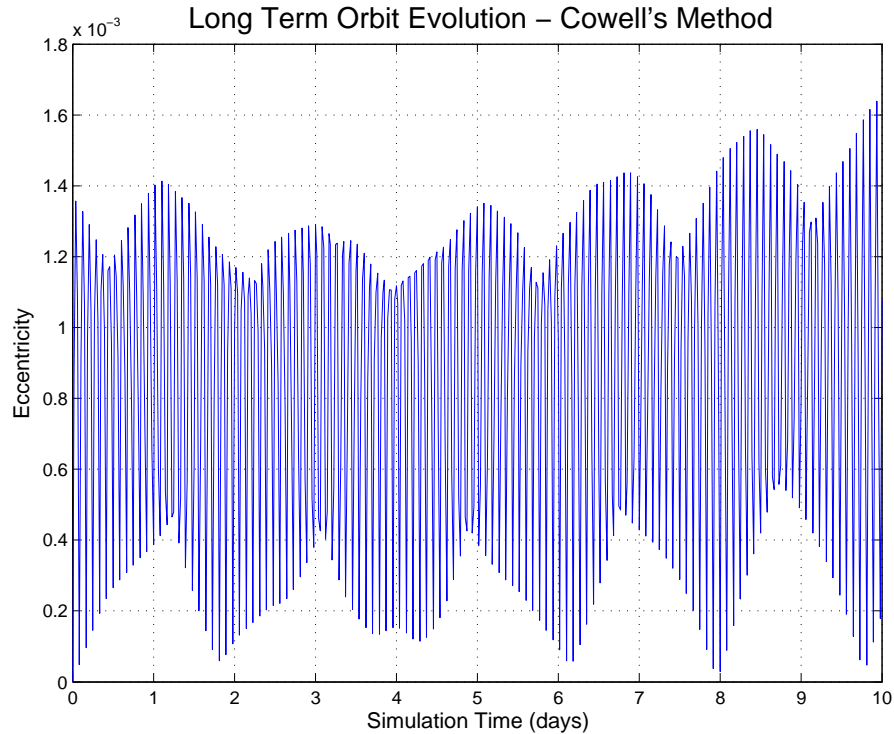
      sma (km)      eccentricity      inclination (deg)      argper (deg)
+7.99833514781320e+003 +1.07699893674950e-003 +2.84904010402501e+001 +3.06222317714564e+002

      raan (deg)      true anomaly (deg)      arglat (deg)      period (min)
+6.02338734067612e+001 +3.11514208090868e+002 +2.57736525805432e+002 +1.18647646343623e+002

degree of gravity model   4
order of gravity model    4

simulation includes solar perturbations
simulation includes lunar perturbations
simulation includes drag perturbations
simulation includes srp perturbations
```

The following is a typical graphics image created with this program.



### 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

$$\mathbf{a}(\mathbf{r}, \mathbf{v}, t) = \ddot{\mathbf{r}}(\mathbf{r}, \dot{\mathbf{r}}, t) = \mathbf{a}_g(\mathbf{r}) + \mathbf{a}_d(\mathbf{r}, \mathbf{v}, t) + \mathbf{a}_{sm}(\mathbf{r}, t) + \mathbf{a}_{srp}(\mathbf{r}, t)$$

where

- $t$  = Universal time
- $\mathbf{r}$  = inertial position vector of the satellite
- $\mathbf{v}$  = inertial velocity vector of the satellite
- $\mathbf{a}_g$  = acceleration due to gravity
- $\mathbf{a}_d$  = acceleration due to atmospheric drag
- $\mathbf{a}_{sm}$  = acceleration due to the Sun and Moon
- $\mathbf{a}_{srp}$  = acceleration due to solar radiation pressure

The satellite's 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.

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

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

where  $\phi$  is the geocentric latitude of the satellite,  $\lambda$  is the geocentric east longitude of the satellite and  $r = |\mathbf{r}| = \sqrt{x^2 + y^2 + z^2}$  is the geocentric distance of the satellite. 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 satellite's 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  $\Phi$ , the inertial rectangular cartesian components of the satellite's 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  $r, \phi, \lambda$  are given by

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

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

where

$R$  = radius of the Earth

$r$  = geocentric distance of the satellite

$S_n^m, C_n^m$  = harmonic coefficients

$\phi$  = geocentric latitude of the satellite =  $\arcsin\left(\frac{z}{r}\right)$

$\lambda$  = longitude of the satellite =  $\alpha - \alpha_g$

$\alpha$  = right ascension of the satellite =  $\arctan\left(\frac{y}{x}\right)$

$\alpha_g$  = right ascension of Greenwich

Right ascension is measure 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:

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

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

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

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

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

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

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

## Orbital Mechanics with MATLAB

The acceleration experienced by the satellite due to atmospheric drag is computed using the following vector expression:

$$\mathbf{a}_d(\mathbf{r}, \mathbf{v}, t) = -\frac{1}{2} \rho(\mathbf{r}, t) |\mathbf{v}_r| \mathbf{v}_r \frac{C_d A}{m}$$

where

- $\mu$  = gravitational constant of the Earth
- $\mathbf{v}_r$  = satellite velocity vector relative to the atmosphere
- $\rho$  = atmospheric density
- $C_d$  = drag coefficient of the satellite
- $A$  = reference area of the satellite
- $m$  = mass of the satellite

During orbit propagation the software uses constant values for the mass, drag coefficient and the reference area of the satellite. Please note that the reference area is measured perpendicular to the relative velocity vector.

The aerodynamic drag algorithm assumes that the atmosphere rotates at the same angular speed as the Earth. With this assumption the relative velocity vector is given by

$$\mathbf{v}_r = \mathbf{v} - \mathbf{w} \times \mathbf{r}$$

where  $\mathbf{w}$  is the inertial rotation vector of the Earth. The angular velocity vector of the Earth is

$$\mathbf{w} = \omega_e \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

where  $\omega_e = 7.2921151467\text{E-}5$  radians per second.

The cross product expansion of the previous equation gives the three components of the relative velocity vector as follows:

$$\mathbf{v}_r = \begin{bmatrix} v_x + \omega_e r_y \\ v_y - \omega_e r_x \\ v_z \end{bmatrix}$$

The calculation of atmospheric density in this script is based on the 1976 U.S. Standard atmosphere.

The acceleration contribution of the Sun and Moon represented by *point masses* is given by

## Orbital Mechanics with MATLAB

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

where

$\mu_m$  = gravitational constant of the Moon

$\mu_s$  = gravitational constant of the Sun

$\mathbf{r}_{m-b}$  = position vector from the Moon to the satellite

$\mathbf{r}_{s-b}$  = position vector from the Sun to the satellite

$\mathbf{r}_{e-m}$  = position vector from the Earth to the Moon

$\mathbf{r}_{e-s}$  = position vector from the Earth to the Sun

The solar and lunar ephemerides used in this program are computer implementations of the numerical method 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. The values of the gravitational constants are as follows:

$$\mu_m = 4902.800076 \frac{\text{km}^3}{\text{sec}^2} \qquad \mu_s = 132712440040.944 \frac{\text{km}^3}{\text{sec}^2}$$

We can define a *solar radiation constant* for any satellite as a function of its size, mass and surface reflective properties according to the equation:

$$C_{srp} = \gamma P_s a^2 \frac{A}{m}$$

where

$\gamma$  = reflectivity constant

$P_s$  = solar radiation constant

$a$  = astronomical unit

$A$  = surface area normal to the incident radiation

$m$  = mass of the satellite

The reflectivity constant is a dimensionless number between 0 and 2. For a perfectly absorbent body  $\gamma = 1$ , for a perfectly reflective body  $\gamma = 2$ , and for a translucent body  $\gamma < 1$ . For example, the reflectivity constant for an aluminum surface is approximately 1.96.

The value of the solar radiation pressure on a perfectly absorbing satellite surface at a distance of one Astronomical Unit from the Sun is

$$P_s = 4.4 \times 10^{-3} \frac{\text{kg}}{\text{km} - \text{sec}^2} = \frac{1358 \text{ watts}}{c \text{ m}^2}$$

where  $c$  is the speed of light.

The acceleration vector of the satellite due to solar radiation pressure is given by:

$$\mathbf{a}_{srp} = C_{srp} \frac{\mathbf{r}_{sat-to-Sun}}{r_{sat-to-Sun}^3}$$

where

$$\mathbf{r}_{sat-to-Sun} = \mathbf{r}_{sat} - \mathbf{r}_{Earth-to-Sun}$$

$\mathbf{r}_{sat}$  = geocentric, inertial position vector of the satellite

$\mathbf{r}_{Earth-to-Sun}$  = geocentric, inertial position vector of the Sun

During the integration process, the software must determine if the satellite is in Earth shadow or sunlight. Obviously, there can be no solar radiation perturbation during Earth eclipse of the satellite orbit. The software makes use of a *shadow parameter* to determine eclipse conditions. This parameter is defined by the following expression:

$$\varphi = -\frac{|\mathbf{r}_{sc} \times \mathbf{r}_{es}|}{|\mathbf{r}_{es}|} \text{sign}(\mathbf{r}_{sc} \bullet \mathbf{r}_{es})$$

where  $\mathbf{r}_{sc}$  = is the geocentric, inertial position vector of the satellite and  $\mathbf{r}_{es}$  = is the geocentric, inertial position vector of the Sun relative to the satellite.

The *critical* values of the shadow parameter for the penumbra (subscript  $p$ ) and umbra part (subscript  $u$ ) of the shadow are given by:

$$\varphi_p = |\mathbf{r}_{sc}| \sin \psi_p$$

$$\varphi_u = |\mathbf{r}_{sc}| \sin \psi_u$$

The penumbra and umbra shadow angles are found from:

$$\psi_p = \eta + \theta_p$$

$$\psi_u = \eta - \theta_u$$

These are the angles between the geocentric anti-Sun vector and the vector to a satellite at the time of shadow entrance or exit.

If we represent the shadow as a cylinder, the shadow angle is given by:

$$\eta = \sin^{-1} \left( \frac{r_e}{r_{sc}} \right)$$

## Orbital Mechanics with MATLAB

The corresponding penumbra and umbra *cone* angles are as follows:

$$\theta_p = \sin^{-1} \left( \frac{r_s + r_e}{r_{es}} \right)$$

$$\theta_u = \sin^{-1} \left( \frac{r_s - r_e}{r_{es}} \right)$$

where

$r_e$  = radius of the Earth

$r_s$  = radius of the Sun

$r_{es}$  = distance from the Earth to the Sun

If the condition  $\varphi_u < \varphi \leq \varphi_p$  is true, the satellite is in the penumbra part of the Earth's shadow, and if the inequality  $0 \leq \varphi \leq \varphi_u$  is true, the satellite is in the umbra part of the shadow. If the absolute value of the shadow parameter is larger than the penumbra value, the satellite is in full sunlight. The shadow calculations also assume that the Earth's atmosphere increases the radius of the Earth by two percent.

### cowell2.m – heliocentric orbit propagation

This application uses Cowell's method to predict the motion of spacecraft, comets and other celestial bodies in heliocentric elliptic orbits. It includes the point mass gravity perturbations due to the planets Venus, Earth, Mars, Jupiter, Saturn and the Sun.

The user can provide the initial orbital elements for this program interactively or by specifying the name of a data file. 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.

The following data file defines a simulation for Halley's comet. Please note that the angular orbital elements are with respect to the true-of-date ecliptic and equinox.

```
calendar date of perihelion passage
(1 <= month <= 12, 1 <= day <= 31, year = all digits!)
2,9,1986

universal time of perihelion passage
(0 <= hours <= 24, 0 <= minutes <= 60, 0 <= seconds <= 60)
15,52,14.592

perihelion radius (Astronomical Units)
(perihelion radius > 0)
0.587478

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

## Orbital Mechanics with MATLAB

```
0.967329

orbital inclination (degrees)
(0 <= inclination <= 180)
162.2486

argument of perigee (degrees)
(0 <= argument of perigee <= 360)
111.8114

right ascension of the ascending node (degrees)
(0 <= raan <= 360)
58.1291
```

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

```
function [fid, oev1, jdpp] = readoe2(filename)

% read orbital elements data file

% required by encke.m and cowell2.m

% input

% filename = name of orbital element data file

% output

% fid    = file id
% jdpp   = julian date of perihelion passage
% 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
```

The following is a typical user interaction with this script.

```
program cowell2

< heliocentric orbit propagation using Cowell's method >

simulation data menu

  <1> user input
  <2> data file

? 2

please input the simulation data file name
(be sure to include the file name extension)
```

## Orbital Mechanics with MATLAB

? halley.dat

initial calendar date and universal time

please input the calendar date

(1 <= month <= 12, 1 <= day <= 31, year = all digits!)

? 1,1,1986

please input the universal time

(0 <= hours <= 24, 0 <= minutes <= 60, 0 <= seconds <= 60)

? 0,0,0

please input the simulation period (days)

? 120

please input the algorithm error tolerance

(a value between 1.0e-8 and 1.0e-10 is recommended)

? 1e-10

The following is the output created for this example. The classical orbital elements are provided in the true-of-date ecliptic and equinox coordinate system.

```
program cowell2
  < heliocentric orbital motion - Cowell's method >

final conditions
calendar date      01-May-1986
universal time     00:00:00.000

      sma (km)      eccentricity      inclination (deg)      argper (deg)
2.6903159700e+009  9.6733201395e-001  1.6224898626e+002  1.1181332498e+002

      raan (deg)    true anomaly (deg)    arglat (deg)          period (days)
5.8130641792e+001  1.0750909424e+002  2.1932241922e+002  2.7855775049e+004

perihelion radius  5.8748967600e-001 AU
```

### Technical Discussion

The second-order heliocentric equations of motion of a satellite or celestial body subject to the point mass gravitational attraction of the Sun and planets are given by

$$\ddot{\mathbf{r}} = \frac{d^2 \mathbf{r}}{dt^2}(\mathbf{r}, t) = -\mu_s \frac{\mathbf{r}_{s-b}}{|\mathbf{r}_{s-b}|^3} - \sum_{i=1}^9 \mu_{p_i} \left( \frac{\mathbf{r}_{(p-b)_i}}{|\mathbf{r}_{(p-b)_i}|^3} + \frac{\mathbf{r}_{p_i}}{|\mathbf{r}_{p_i}|^3} \right)$$

where

$\mu_s$  = gravitational constant of the Sun

$\mu_{p_i}$  = gravitational constant of planet  $i$

$\mathbf{r}_p$  = position vector from the Sun to planet

$\mathbf{r}_{s-b}$  = position vector from the Sun to the body

$\mathbf{r}_{p-b}$  = position vector from the planet to the body

These position vectors are related according to

$$\mathbf{r}_{s-b} = \mathbf{r}_p + \mathbf{r}_{p-b}$$

### cowell3.m – drag perturbed orbital motion

This MATLAB script determines the long-term orbit evolution of satellites due to the  $J_2$  contribution of Earth oblateness and aerodynamic drag using Cowell's solution of the cartesian form of differential equations of motion. The atmospheric density required for the drag calculations is determined using the Jacchia 1970 atmosphere model.

This script reads and linearly interpolates a solar activity data file named `solar.dat` and the user can select the "level" of solar activity to use (5%, 50% or 95%). The format of this ASCII data file is also described in Section 31 under the description for the `jatmos70.m` function.

The user can provide the initial orbital elements for this program interactively or by specifying the name of a 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)
6878.14

orbital eccentricity (non-dimensional)
(0 <= eccentricity < 1)
.0125

orbital inclination (degrees)
(0 <= inclination <= 180)
28.5

argument of perigee (degrees)
(0 <= argument of perigee <= 360)
270

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

## Orbital Mechanics with MATLAB

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

The following is a typical user interaction with this script.

```
program cowell13

< drag perturbed orbital motion >

initial calendar date and time

please input the calendar date
(1 <= month <= 12, 1 <= day <= 31, year = all digits!)
? 1,1,2001

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

please input the simulation period (days)
? 5

please input the algorithm 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)
? 6578.14

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

## *Orbital Mechanics with MATLAB*

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

aerodynamic drag inputs

```
please input the drag coefficient (non-dimensional)
? 2
```

```
please input the cross-sectional area (square meters)
? 5
```

```
please input the spacecraft mass (kilograms)
? 2000
```

```
please input the type of solar activity data to use
(1 = 95%, 2 = 50%, 3 = 5%)
? 2
```

The user can also elect to plot the long-term evolution of the satellite's orbital elements by responding to the following two requests:

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

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

The orbital elements that can be displayed are defined in the following interactive menu:

```
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
<9> geodetic altitude
<10> east longitude
<11> geodetic latitude

? 9
```

The program will display the final orbital elements as follows:

```
program cowell3
< drag perturbed orbital motion >
initial calendar date      01-Jan-2001
```

## Orbital Mechanics with MATLAB

```
initial universal time      00:00:00.000

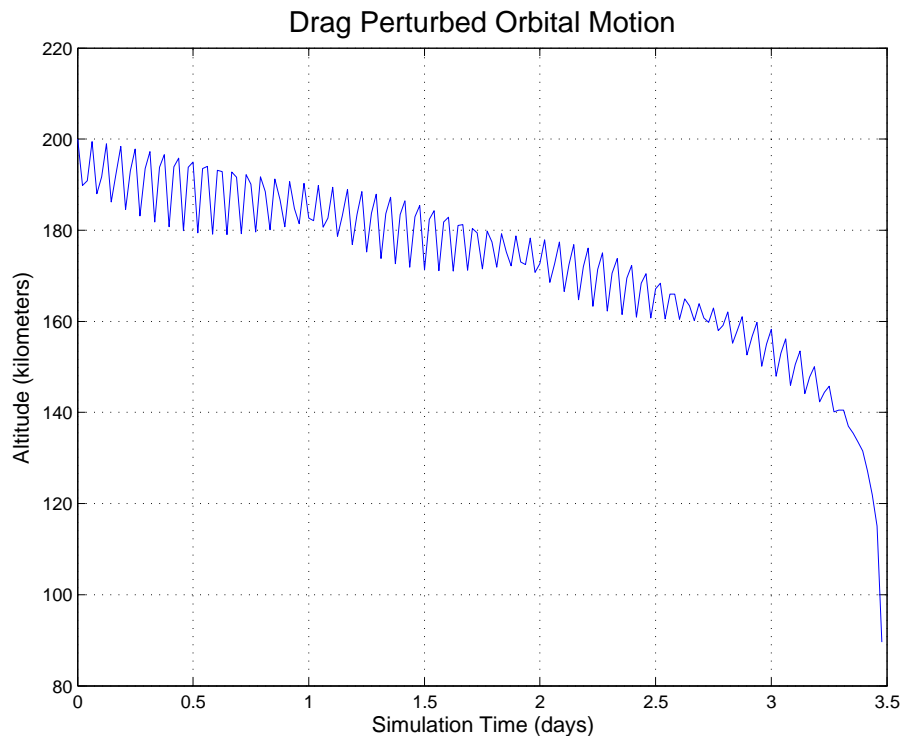
final calendar date        04-Jan-2001
final universal time       11:23:04.284

      sma (km)      eccentricity      inclination (deg)      argper (deg)
+6.40781533395313e+003 +1.09147692464397e-002 +2.84843665751203e+001 +1.54829940796242e+002

      raan (deg)      true anomaly (deg)      arglat (deg)      period (min)
+3.32284614244865e+002 +2.10751952664877e+002 +5.58189346111888e+000 +8.50794579312798e+001

drag coefficient           2.000000000
cross-sectional area      5.000000000 square meters
spacecraft mass          2000.000000000 kilograms
solar activity            50 percent
```

The companion graphics display for this example is shown below. For this example the satellite actually decays and re-enters the Earth's atmosphere. This script will terminate whenever the altitude falls below 90 kilometers.



### **cowell4.m – modified equinoctial equations of motion**

This MATLAB script determines the long-term orbit evolution of Earth satellites using Cowell's solution of the modified equinoctial orbital elements form of the differential equations of motion. The user can elect to include the non-spherical gravity effects of the Earth, aerodynamic drag using a US 1976 atmosphere model, solar radiation pressure perturbations, and the point-mass gravity of the Sun and Moon in these predictions.

## Orbital Mechanics with MATLAB

The user can provide the initial orbital elements for this program interactively or by specifying the name of a 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)
6878.14

orbital eccentricity (non-dimensional)
(0 <= eccentricity < 1)
.0125

orbital inclination (degrees)
(0 <= inclination <= 180)
28.5

argument of perigee (degrees)
(0 <= argument of perigee <= 360)
270

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

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

The following is a typical user interaction with this script.

```
program cowell14

< Earth orbital motion - Cowell's method >

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

please input the simulation period (days)
? 30

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

## *Orbital Mechanics with MATLAB*

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 drag perturbations (y = yes, n = no)  
? **y**

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

orbital elements menu

<1> user input

<2> data file

? **2**

please input the orbital elements data file name  
(be sure to include the file name extension)  
? **leo.dat**

aerodynamic drag inputs

please input the drag coefficient (non-dimensional)  
? **2**

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

please input the spacecraft mass (kilograms)  
? **2000**

solar radiation pressure inputs

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

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

## Orbital Mechanics with MATLAB

please input the spacecraft mass (kilograms)  
? 2000

The following is the data display for this example.

```
program cowell4

< Earth orbital motion - Cowell's method >

initial calendar date      01-Jan-1998
initial universal time     00:00:00.000

final calendar date       31-Jan-1998
final universal time      00:00:00.000

      sma (km)      eccentricity      inclination (deg)      argper (deg)
+6.88156536909356e+003 +1.24627707493235e-002 +2.85293919589642e+001 +2.33542703417814e+002

      raan (deg)      true anomaly (deg)      arglat (deg)      period (min)
+2.02518759110216e+002 +3.06894933311517e+002 +1.80437636729331e+002 +9.46870507777942e+001

degree of gravity model   42
order of gravity model    42

simulation includes solar perturbations
simulation includes lunar perturbations
simulation includes drag perturbations
simulation includes srp perturbations
```

The user can also elect to plot the long-term evolution of the satellite's orbital elements by responding to the following request:

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

please input the graphics step size (minutes)
? 60
```

The orbital elements that can be displayed are defined in the following menu:

```
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
<9> geodetic altitude
<10> east longitude
<11> geodetic latitude
```

### Technical Discussion

The modified equinoctial orbital elements are a set of orbital elements that are useful for trajectory analysis and optimization. They are valid for circular, elliptic, and hyperbolic orbits.

## Orbital Mechanics with MATLAB

These equations exhibit no singularity for zero eccentricity and orbital inclinations equal to 0 and 90 degrees. However, two components of the orbital element set are singular for an orbital inclination of 180 degrees.

The relationship between modified equinoctial orbital elements and classical orbital elements is

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

$$f = e \cos(\omega + \Omega)$$

$$g = e \sin(\omega + \Omega)$$

$$h = \tan(i/2) \cos \Omega$$

$$k = \tan(i/2) \sin \Omega$$

$$L = \Omega + \omega + \theta$$

where

$p$  = semiparameter

$a$  = semimajor axis

$e$  = orbital eccentricity

$i$  = orbital inclination

$\omega$  = argument of perigee

$\Omega$  = right ascension of the ascending node

$\theta$  = true anomaly

$L$  = true longitude

The relationship between classical and modified equinoctial orbital elements is summarized as follows:

*semimajor axis*

$$a = \frac{p}{1 - f^2 - g^2}$$

*orbital eccentricity*

$$e = \sqrt{f^2 + g^2}$$

*orbital inclination*

$$i = 2 \tan^{-1} \left( \sqrt{h^2 + k^2} \right)$$

*argument of periapsis*

$$\omega = \tan^{-1}(g/f) - \tan^{-1}(k/h)$$

*right ascension of the ascending node*

$$\Omega = \tan^{-1}(k/h)$$

*true anomaly*

$$\theta = L - (\Omega + \omega) = L - \tan^{-1}(g/f)$$

The mathematical relationships between an inertial state vector and the corresponding modified equinoctial elements are summarized as follows:

*position vector*

$$\mathbf{r} = \begin{bmatrix} \frac{r}{s^2} (\cos L + \alpha^2 \cos L + 2hk \sin L) \\ \frac{r}{s^2} (\sin L - \alpha^2 \sin L + 2hk \cos L) \\ \frac{2r}{s^2} (h \sin L - k \cos L) \end{bmatrix}$$

*velocity vector*

$$\mathbf{v} = \begin{bmatrix} -\frac{1}{s^2} \sqrt{\frac{\mu}{p}} (\sin L + \alpha^2 \sin L - 2hk \cos L + g - 2fhk + \alpha^2 g) \\ -\frac{1}{s^2} \sqrt{\frac{\mu}{p}} (-\cos L + \alpha^2 \cos L + 2hk \sin L - f + 2ghk + \alpha^2 f) \\ \frac{2}{s^2} \sqrt{\frac{\mu}{p}} (h \cos L + k \sin L + fh + gk) \end{bmatrix}$$

where

$$\alpha^2 = h^2 - k^2$$

$$s^2 = 1 + h^2 + k^2$$

$$r = \frac{p}{w}$$

$$w = 1 + f \cos L + g \sin L$$

The differential equations of orbital motion expressed in terms of modified equinoctial orbital elements are as follows:

$$\dot{p} = \frac{dp}{dt} = \frac{2p}{w} \sqrt{\frac{p}{\mu}} \Delta_t$$

$$\dot{f} = \frac{df}{dt} = \sqrt{\frac{p}{\mu}} \left[ \Delta_r \sin L + [(w+1) \cos L + f] \frac{\Delta_t}{w} - (h \sin L - k \cos L) \frac{g \Delta_n}{w} \right]$$

$$\dot{g} = \frac{dg}{dt} = \sqrt{\frac{p}{\mu}} \left[ -\Delta_r \cos L + [(w+1) \sin L + g] \frac{\Delta_t}{w} + (h \sin L - k \cos L) \frac{g \Delta_n}{w} \right]$$

$$\dot{h} = \frac{dh}{dt} = \sqrt{\frac{p}{\mu}} \frac{s^2 \Delta_n}{2w} \cos L$$

$$\dot{k} = \frac{dk}{dt} = \sqrt{\frac{p}{\mu}} \frac{s^2 \Delta_n}{2w} \sin L$$

$$\dot{L} = \frac{dL}{dt} = \sqrt{\mu p} \left( \frac{w}{p} \right)^2 + \frac{1}{w} \sqrt{\frac{p}{\mu}} (h \sin L - k \cos L) \Delta_n$$

where  $\Delta_r, \Delta_t, \Delta_n$  are external *non-two-body* perturbations in the radial, tangential and orbit normal directions, respectively.

$$\mathbf{P} = \Delta_r \hat{\mathbf{i}}_r + \Delta_t \hat{\mathbf{i}}_t + \Delta_n \hat{\mathbf{i}}_n$$

where  $\hat{\mathbf{i}}_r, \hat{\mathbf{i}}_t$  and  $\hat{\mathbf{i}}_n$  are unit vectors in the radial, tangential and normal directions given by

$$\hat{\mathbf{i}}_r = \frac{\mathbf{r}}{\|\mathbf{r}\|}$$

$$\hat{\mathbf{i}}_n = \frac{\mathbf{r} \times \mathbf{v}}{\|\mathbf{r} \times \mathbf{v}\|}$$

$$\hat{\mathbf{i}}_t = \hat{\mathbf{i}}_n \times \hat{\mathbf{i}}_r$$

The equations of orbital motion can also be expressed in vector form as follows:

$$\dot{\mathbf{y}} = \frac{d\mathbf{y}}{dt} = \mathbf{A}(\mathbf{y})\mathbf{P} + \mathbf{b}$$

where

$$\mathbf{A} = \begin{pmatrix} 0 & \frac{2p}{q} \frac{\sqrt{p}}{\sqrt{\mu}} & 0 \\ \sqrt{\frac{p}{\mu}} \sin L & \sqrt{\frac{p}{\mu}} \frac{1}{q} \{(q+1) \cos L + f\} & -\sqrt{\frac{p}{\mu}} \frac{g}{q} \{h \sin L - k \cos L\} \\ -\sqrt{\frac{p}{\mu}} \cos L & \sqrt{\frac{p}{\mu}} \{(q+1) \sin L + g\} & \sqrt{\frac{p}{\mu}} \frac{f}{q} \{h \sin L - k \cos L\} \\ 0 & 0 & \sqrt{\frac{p}{\mu}} \frac{s^2 \cos L}{2q} \\ 0 & 0 & \sqrt{\frac{p}{\mu}} \frac{s^2 \sin L}{2q} \\ 0 & 0 & \sqrt{\frac{p}{\mu}} \{h \sin L - k \cos L\} \end{pmatrix}$$

and

$$\mathbf{b} = \left\{ 0 \ 0 \ 0 \ 0 \ 0 \ \sqrt{\mu p} \left( \frac{q}{p} \right)^2 \right\}^T$$

For *unperturbed* two-body motion,  $\mathbf{P} = 0$  and the first five equations of motion are simply  $\dot{p} = \dot{f} = \dot{g} = \dot{h} = \dot{k} = 0$ . Therefore, for two-body motion these modified equinoctial orbital elements are constant. The true longitude  $L$  is often called the *fast variable* of this orbital element set.