# Cowell's Method for Earth Satellites

This document describes an interactive MATLAB script named `cowell.m` that predicts the long-term behavior of an Earth satellite subject to perturbations. This script implements a special perturbation solution of orbital motion using a variable step size Runge-Kutta-Fehlberg (RKF78) integration method to numerically solve Cowell's form of the system of differential equation subject to the central body gravity and other external forces. This is also called the orbital initial value problem (IVP).

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

- non-spherical Earth gravity
- 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

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.

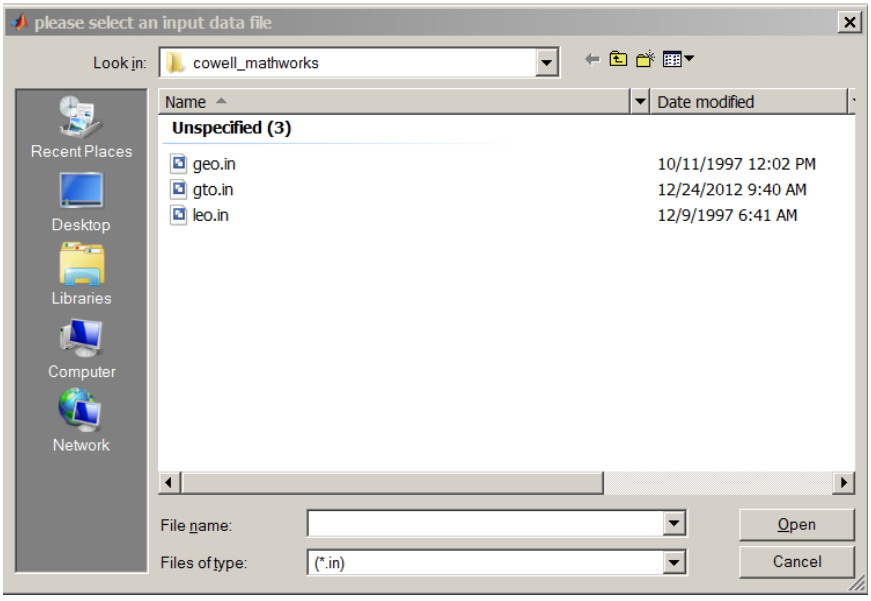
```
semimajor axis (kilometers)
(semimajor axis > 0)
6878.14

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 cowell.m

% input

%  filename = name of orbital element data file

% output

%  fid = file id

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

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



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

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

The following is a typical user interaction with this MATLAB script. Please note the units and valid range for each input. A smaller error tolerance will predict the orbit more accurately at the expense of longer run time.

```
                program cowell

  < 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-12 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

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

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

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

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 script output for this example.

```
                    program cowell

    < Earth orbital motion - Cowell's method >

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

initial orbital elements and state vector

        sma (km)              eccentricity          inclination (deg)         argper (deg)
 +8.00000000000000e+003  +0.00000000000000e+000  +2.85000000000000e+001  +0.00000000000000e+000

       raan (deg)          true anomaly (deg)         arglat (deg)            period (min)
 +1.00000000000000e+002  +4.50000000000000e+001  +4.50000000000000e+001  +1.18684693004297e+002
```

```
        rx (km)                  ry (km)                  rz (km)                 rmag (km)
 -5.87811692006444e+003  +4.70764973572722e+003  +2.69921756065708e+003  +8.00000000000000e+003

        vx (kps)                 vy (kps)                 vz (kps)                vmag (kps)
 -3.45303184300316e+000  -5.67710577460503e+000  +2.38161632146162e+000  +7.05868650582387e+000


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

final orbital elements and state vector

        sma (km)              eccentricity          inclination (deg)         argper (deg)
 +7.99833355048314e+003  +1.07701539221320e-003  +2.84903999305853e+001  +3.06224456694258e+002

       raan (deg)          true anomaly (deg)         arglat (deg)            period (min)
 +6.02338556218530e+001  +3.11518112139396e+002  +2.57742568833654e+002  +1.18647610801331e+002

        rx (km)                  ry (km)                  rz (km)                 rmag (km)
 +5.11640582833442e+003  -4.88096421005727e+003  -3.72565790990581e+003  +7.99261828700196e+003

        vx (kps)                 vy (kps)                 vz (kps)                vmag (kps)
 +4.56789825042465e+000  +5.34162549550074e+000  -7.12776460211125e-001  +7.06446796600049e+000


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 simulation summary screen display contains the following information:

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

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

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

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

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

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

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

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

**sma (km)** = semimajor axis in kilometers

**eccentricity** = orbital eccentricity (non-dimensional)

**inclination (deg)** = orbital inclination in degrees

**argper (deg)** = argument of perigee in degrees

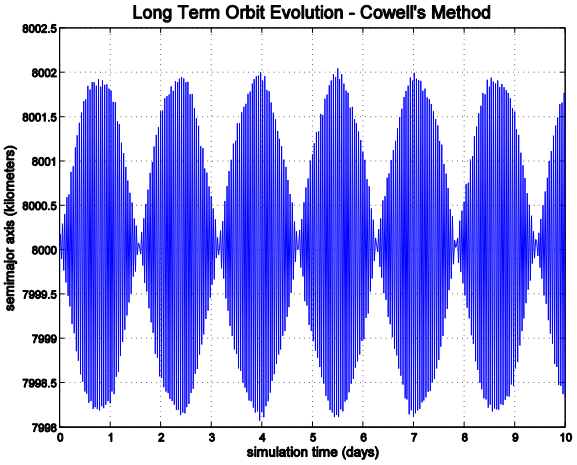
**raan (deg)** = right ascension of the ascending node in degrees

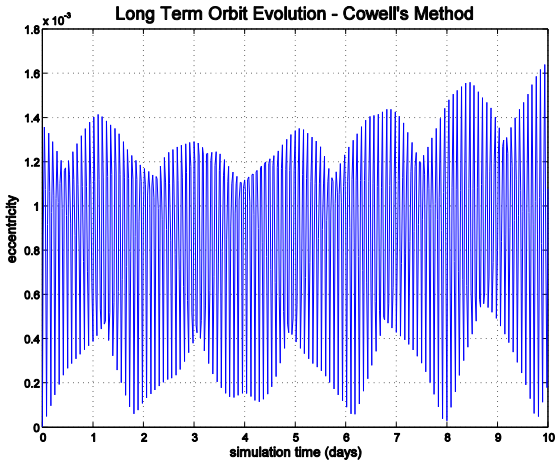
**true anomaly (deg)** = true anomaly in degrees

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

**period (min)** = orbital period in minutes

The following are two graphics images for this example.



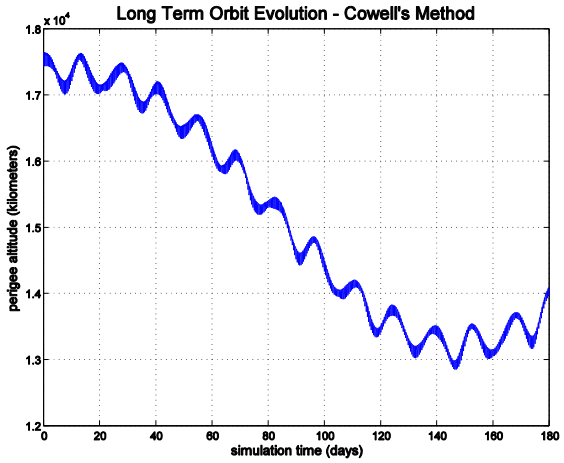

Here's the user interaction and script output for an example that models a geosynchronous transfer orbit (GTO). This example reads a file name `gto.in` which contains the classical orbital elements.

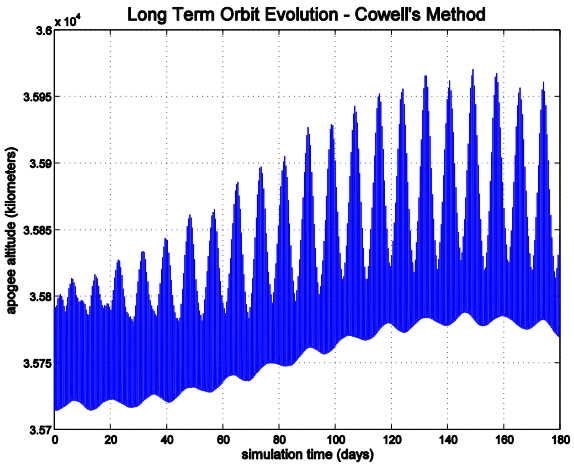
```
                program cowell1

  < 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!)
? 12,1,2012

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

please input the simulation period (days)
? 180

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


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)
? n

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


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


please input the graphics step size (minutes)
? 30


 orbital elements menu

   <1> user input

   <2> data file

? 2


                    program cowell

    < Earth orbital motion - Cowell's method >

initial calendar date       01-Dec-2012
initial universal time      00:00:00.000

initial orbital elements and state vector

        sma (km)              eccentricity          inclination (deg)         argper (deg)
 +2.442123630000000e+004  +7.265438900000000e-001  +2.629980000000000e+001  +2.700000000000000e+002

       raan (deg)          true anomaly (deg)         arglat (deg)            period (min)
 +1.000000000000000e+002  +0.000000000000000e+000  +2.700000000000000e+002  +6.330109156903620e+002

        rx (km)                 ry (km)                rz (km)                rmag (km)
 +5.895914892933970e+003  +1.039608871585120e+003  -2.958868895964040e+003  +6.678136279988790e+003

        vx (kps)                vy (kps)               vz (kps)               vmag (kps)
 -1.762788832205550e+000  +9.997272255912500e+000  -8.262331512094680e-016  +1.015149629492060e+001


final calendar date         30-May-2013
final universal time        00:00:00.000

final orbital elements and state vector

        sma (km)              eccentricity          inclination (deg)         argper (deg)
 +2.438492626175890e+004  +7.283781046660820e-001  +2.687851698361990e+001  +2.250642155365620e+001

       raan (deg)          true anomaly (deg)         arglat (deg)            period (min)
 +3.257443080749960e+001  +1.708404406138730e+002  +1.933468621675290e+002  +6.315996784807130e+002

        rx (km)                 ry (km)                rz (km)                rmag (km)
 -2.889677298618380e+004  -2.841980803870100e+004  -4.253191369469510e+003  +4.075289701109590e+004

        vx (kps)                vy (kps)               vz (kps)               vmag (kps)
 +6.118422090540990e-001  -1.483375864385660e+000  -8.005525370868910e-001  +1.793220345763000e+000


degree of gravity model     4
order of gravity model      4
```

```
simulation includes solar perturbations
simulation includes lunar perturbations
```

Here are graphic displays of the behavior of the geodetic perigee and apogee altitudes for this example.




*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.

*Acceleration due to non-spherical Earth gravity*

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

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

where $\phi$ 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 non-spherical terms in the Earth's geopotential. In terms of the Earth's geopotential $\Phi$, the inertial rectangular cartesian components of the spacecraft's acceleration vector are as follows:

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

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

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

The three partial derivatives of the geopotential with respect to $r,\phi,\lambda$ are given by

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

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

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

where

$R =$ radius of the Earth

$r =$ geocentric distance of the spacecraft

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

$\phi =$ geocentric latitude of the spacecraft $= \sin^{-1}\left(z/r\right)$

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

$\alpha =$ right ascension of the spacecraft $= \tan^{-1}\left(r_y/r_x\right)$

$\alpha_g =$ right ascension of Greenwich

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

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

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

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

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

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

where the first few associated Legendre functions are given by

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

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

The trigonometric arguments are determined from expansions given by

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

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

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

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

```
% read gravity model coefficients

gmfile = 'egm96.dat';
```

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

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

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

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

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

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

*Acceleration due to atmospheric drag*

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

$\mathbf{v}_r =$ 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 the 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 **w** is the inertial rotation vector of the Earth. The angular velocity vector of the Earth is $\mathbf{w} = \omega_e \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$ where $\omega_e$ is the scalar rotation rate.

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 MATLAB script is based on the 1976 U.S. Standard atmosphere. The tabular data for this model is contained in a simple data file named `atmos76.dat`. This script uses linear interpolation in the `atmos76.m` function to evaluate the density as a function of geodetic altitude.

*Acceleration due to the sun and moon*

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

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

where

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

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

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

where

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

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

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

where

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

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

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

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

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

*Acceleration due to solar radiation pressure*

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

$$P_s = \frac{1358}{c}\frac{watts}{m^2}$$

where $c$ is the speed of light in meters/second.

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:

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

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 the following formulas:

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

The penumbra and umbra shadow angles are found from:

$$\psi_p = \eta + \theta_p \qquad \psi_u = \eta - \theta_u$$

These are the angles between the geocentric anti-Sun vector and the geocentric 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:

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

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

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

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 $\varphi_u < \varphi \le \varphi_p$ is true, the satellite is in the penumbra part of the Earth's shadow, and if the inequality $0 \le \varphi \le \varphi_u$ 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 used in this MATLAB script also assume that the Earth's atmosphere increases the radius of the Earth by two percent.

*Fundamental constants*

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

```
function om_constants

% astrodynamic and utility constants

% Orbital Mechanics with Matlab

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

global dtr rtd emu mmu smu omega req flat j2 aunit

dtr = pi / 180.0;

rtd = 180.0 / pi;

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

emu = 398600.436233;

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

mmu = 4902.800076;

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

smu = 132712440040.944;

% earth inertial rotation rate (radians/second)

omega = 7.292115486e-5;

% earth equatorial radius (kilometers)

req = 6378.1363;
```

```
% earth flattening factor (non-dimensional)

flat = 1.0 / 298.257;

% earth oblateness gravity coefficient (non-dimensional)

j2 = 0.00108263;

% astronomical unit (kilometers)

aunit = 149597870.691;
```