

Cowell's Method for Heliocentric Orbits

This *Numerit* program (cowell12) demonstrates heliocentric orbit propagation of spacecraft, asteroid or comet motion using a classic technique called Cowell's method. Cowell's method numerically integrates the orbital equations of motion. This program includes the point mass gravity perturbations due to the planets Venus, Earth, Mars, Jupiter, and Saturn. The planetary 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 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

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

where

- \mathbf{m}_s = gravitational constant of the Sun
- \mathbf{m}_{p_i} = gravitational constant of planet i
- \mathbf{r}_{p_i} = position vector from the Sun to planet i
- \mathbf{r}_{s-b} = position vector from the Sun to the body
- $\mathbf{r}_{(p-b)_i}$ = position vector from planet i to the body

These position vectors are related according to

$$\mathbf{r}_{s-b} = \mathbf{r}_p + \mathbf{r}_{p-b} \quad (2)$$

This *Numerit* program uses a Runge-Kutta-Fehlberg 7(8) method to numerically integrate the first-order form of the orbital equations of motion. This is a variable step size method of order 7 with an 8th order error estimate which is used to dynamically change the integration step size during the simulation. The user can control how well the equations are solved by specifying a truncation error tolerance.

The syntax of this *Numerit* function is as follows:

```
function rkf78 (neq, ti, tf, h, tetol, x, xout)
` solve first order system of differential equations
` Runge-Kutta-Fehlberg 7(8) method
` input
` neq = number of differential equations
```

Orbital Mechanics with Numerit

```
` ti      = initial simulation time
` tf      = final simulation time
` h       = initial guess for integration step size
` tetol   = truncation error tolerance (non-dimensional)
` x       = integration vector at time = ti

` output

` xout    = integration vector at time = tf
```

This algorithm requires the following statement the first time it is called.

```
rkcoef = 1
```

This statement is placed in the main program and is used to initialize the integration coefficients for this numerical method. The value of `rkcoef` is passed to the `rkf78` function using a common `rkcoef` statement in the main program. The `rkf78` function will then set this to 0 after the coefficients are calculated.

This function also requires a second function which defines the user's system of first-order differential equations. The format of this function is given by

```
function orbeqm (time, y, ydot)

` first order equations of orbital motion

` cowell's method

` input

` time = simulation time (seconds)
` y    = state vector

` output

` ydot = integration vector
```

where `orbeqm` is the actual name of the user-coded function. The function parameter list should be exactly as shown here. The main program uses "rerouting" to tell the `rkf78` function which equations of motion function to use. For this example the code is

```
orbeqm -> ceqm
```

The `cowell` program will prompt you for the name of an orbital elements data file, including the file name extension. It will also prompt you for the initial calendar date and universal time. The final request asks for the simulation duration in days.

The orbital elements data file contains *true-of-date, heliocentric ecliptic* classical orbital elements. It is a simple ascii text file created with a text editor. The format and contents of a typical data file are as follows:

Orbital Mechanics with Numerit

```
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)
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
```

You can change the explanatory text and data in this file, but do not change the number of lines or the type of information in each line. The software expects to find exactly 27 total lines. This data file is called `halley.dat` and contains elements for Halley's comet.

The following is a typical example for Halley's comet using this data file:

```
program cowell12
< Cowell's method for heliocentric orbits >

initial orbital elements

January 1, 1986

10h 20m 30s

      sma (km)      eccentricity      inclination (deg)      argper (deg)
2690014320.33      0.967329      162.2486      111.8114

      raan (deg)      true anomaly (deg)      arglat (deg)      period (days)
58.1291      279.035113226      30.8465132264      27851.0902191

final orbital elements

May 1, 1986

10h 20m 30s

      sma (km)      eccentricity      inclination (deg)      argper (deg)
```

Orbital Mechanics with Numerit

2690311083.86 0.967331951657 162.248987975 111.813339888

raan (deg)	true anomaly (deg)	arglat (deg)	period (days)
58.1306470436	107.680222659	219.493562547	27855.6991624

perihelion radius = 0.58748972935 AU

The initial calendar date and universal time for this example were January 1, 1986 at 0 hours UT. The simulation period was 120 days and the RKF truncation tolerance was $1.0e-8$. This tolerance is "hard-wired" in the code (`tetol`) and can be easily changed by the user. Smaller values of `tetol` will predict the orbit more accurately at the expense of longer run times.