

Encke's Method for Heliocentric Orbits

This *Numerit* application (`encke`) demonstrates heliocentric orbit propagation of spacecraft, asteroid or comet motion using a classical technique called Encke's method. It 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 classic form of Encke's special perturbation method solves the initial value problem (IVP) of celestial mechanics by analytically propagating a two-body "reference" orbit and numerically integrating the deviations from two-body motion. These two procedures are performed for the same time interval or step size. Since the two-body solution is "exact", the time interval used is usually a function of the numerical integration technique and the magnitude of the perturbations. After each propagation interval, the errors between the reference orbit and the perturbed orbit are evaluated. When these errors become "large", a process known as *rectification* of the orbit is performed. This involves adding the integrated deviations to the two-body orbit to produce a new reference orbit called the osculating orbit. After rectification, the algorithm is reinitialized with the elements of the new osculating orbit and the process is repeated until either the errors become large once more or the final simulation time is reached.

The numerical algorithm used in this program simplifies this technique by using a variable step-size integration method to automatically select the propagation step size. Since this type of integrator "senses" the magnitude of the perturbations, it will choose the "best" and largest step size for each propagation interval. The user can control how well this procedure is performed with the algorithm truncation error tolerance. Furthermore, the software eliminates the logic required to determine when rectification should occur by simply rectifying the orbit after each and every propagation step.

The *two-body* or *unperturbed* equations of motion of a spacecraft or celestial body are

$$\frac{d^2 \mathbf{r}_k}{dt^2} + \mathbf{m} \frac{\mathbf{r}_k(t)}{r_k^3} = 0 \quad (1)$$

where \mathbf{r}_k is the unperturbed position vector of the spacecraft at any time t and \mathbf{m} is the gravitational constant of the primary body. In this equation r_k is the scalar magnitude of the position vector \mathbf{r}_k .

The equations of motion of motion of a spacecraft subject to a point mass central body and other external perturbations are

Orbital Mechanics with Numerit

$$\frac{d^2 \mathbf{r}_p}{dt^2} + \mathbf{m} \frac{\mathbf{r}_p(t)}{r_p^3} = \mathbf{a}(\mathbf{r}_p, t) \quad (2)$$

where \mathbf{r}_p is the perturbed position vector at any time t and $\mathbf{a}(\mathbf{r}_p, t)$ is the vector of planetary perturbations or other "indirect" accelerations.

The difference between these two types of orbital motion is

$$\frac{d^2 \mathbf{r}}{dt^2} = \frac{d^2 \mathbf{r}_p}{dt^2} - \frac{d^2 \mathbf{r}_k}{dt^2} \quad (3)$$

In order to avoid numerical difficulties when subtracting small differences of vector equations, the following form of the equations of motion is used:

$$\frac{d^2 \mathbf{r}}{dt^2} = -\frac{\mathbf{m}}{r^3} [f(q) \mathbf{r}_p(t) + \mathbf{r}_p(t)] + \mathbf{a}(\mathbf{r}_p, t) \quad (4)$$

where

$$f(q) = q \left(\frac{3 + 3q + q^2}{1 + (1 + q)^{3/2}} \right) \quad (5)$$

and

$$q = \frac{(\mathbf{dr} - 2\mathbf{r}) \cdot \mathbf{dr}}{r^2} \quad (6)$$

At any time the position and velocity vectors of the true orbit are given by the vector sum of the two body and perturbed components as follows:

$$\begin{aligned} \mathbf{r}(t) &= \mathbf{r}_k(t) + \mathbf{r}_p(t) \\ \mathbf{v}(t) &= \mathbf{v}_k(t) + \mathbf{v}_p(t) \end{aligned} \quad (7)$$

The gravitational acceleration on the spacecraft, asteroid or comet due to each planet is determined from the following expression:

$$\mathbf{a}(\mathbf{r}, t) = -\mathbf{m} \left(\frac{\mathbf{r}_{p-b}}{|\mathbf{r}_{p-b}|^3} + \frac{\mathbf{r}_p}{|\mathbf{r}_p|^3} \right) \quad (8)$$

where

Orbital Mechanics with Numerit

$$\begin{aligned} \mathbf{r}_{p-b} &= \text{position vector of body relative to planet} \\ \mathbf{r}_p &= \text{position vector of body relative to the Sun} \\ \mathbf{m} &= \text{gravitational constant of planet} \end{aligned} \quad (9)$$

The software (`encke.num`) uses a slightly modified version of Stanley Shepperd's two body algorithm to provide the unperturbed solution for any simulation time. The derivation of this algorithm is described in "Universal Keplerian State Transition Matrix", *Celestial Mechanics*, **35** (1985) 129-144. The "deviation" equations of motion are integrated using a slightly modified version of the Runge-Kutta-Fehlberg 7(8) method called `rkencke`. Additional information about this algorithm can be found in the technical discussion for the `cowell` program.

Although simple and straight-forward, this approach does require special attention to the initialization at the start of each propagation step. Since there are no perturbations at the initial epoch of the osculating reference orbit, the integrator would use the initial step size defined by the user. This will cause very large errors before rectification can be performed. To avoid this problem, the software forces the algorithm to perform the first phase of the orbit propagation with a fixed step size. This is accomplished by setting the initial step size guess to a small value and the truncation error tolerance to a very large value. For the second phase of the propagation, the step size estimate is increased and the RKF algorithm is allowed to change the actual step size to the largest value based on the perturbations and user-defined error tolerance. The truncation error tolerance for this phase is set to a very small number. The "best" initial step size depends on the type of orbit propagation and the initial conditions.

Additional information about this algorithm can be found in AAS 92-196, "A Simple and Efficient Computer Implementation of Encke's Method", AAS/AIAA Spaceflight Mechanics Meeting, Colorado Springs, Colorado, February 24-26, 1992.

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

The following is a typical example for Halley's comet:

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

Orbital Mechanics with Numerit

final orbital elements

May 1, 1986

10h 20m 30s

sma (km)	eccentricity	inclination (deg)	argper (deg)
2690310721.2	0.967331947242	162.248987969	111.813339887
raan (deg)	true anomaly (deg)	arglat (deg)	period (days)
58.1306467941	107.680210212	219.493550099	27855.6935299

perihelion radius = 0.58748972956 AU

This example uses a *true-of-date, heliocentric ecliptic* orbital elements data file named `halley.dat`. The contents of this file are as follows:

```
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 lines. Please note the units of each data item.

The simulation period for this example 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.