Encke's Method

This document describes two MATLAB scripts that demonstrate how to use Encke's method to propagate geocentric and heliocentric orbits. 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 orbital elements of the new osculating orbit and the process is repeated until either the errors become large once more or the final time is reached.

The numerical algorithm used in this script 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.

Technical Discussion

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

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

where $\mathbf{r}_k$ is the unperturbed position vector of the spacecraft at any time t and μ is the gravitational constant of the primary body.

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

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

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

The difference between these two types of orbital motion is

$$\frac{\delta^2\mathbf{r}}{dt^2} = \frac{d^2\mathbf{r}_p}{dt^2} - \frac{d^2\mathbf{r}_k}{dt^2}$$

In order to avoid numerical difficulties when subtracting small differences of vector equations, the following form of the equations of motion that use Battin's *f(q)* function is used:

$$\frac{\delta^2 \mathbf{r}}{dt^2} = -\frac{\mu}{\rho^3}\left[f(q)\mathbf{r}_p(t) + \mathbf{r}_p(t)\right] + \mathbf{a}(\mathbf{r}_p, t)$$

where

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

and

$$q = \frac{(\delta\mathbf{r} - 2\mathbf{r}) \bullet \delta\mathbf{r}}{r^2}$$

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:

$$\mathbf{r}(t) = \mathbf{r}_k(t) + \mathbf{r}_p(t)$$

$$\mathbf{v}(t) = \mathbf{v}_k(t) + \mathbf{v}_p(t)$$

The software uses a slightly modified version of Shepperd's two body algorithm to provide the unperturbed solution for any simulation time. This MATLAB file is called `twobody4.m` and has for its input and output the entire state vector in a single array instead of individual position and velocity arrays. The "deviation" equations of motion are integrated using a slightly modified version of the Runge-Kutta-Fehlberg 7(8) method called `rkencke.m`.

Although simple and straightforward, 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 numerical 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.

encke1.m – Encke's method for geocentric orbits

This MATLAB script demonstrates geocentric orbit propagation using Encke's method. It includes the central body point mass gravity perturbation of the Earth and the perturbation due to Earth oblateness. The software can be modified to include additional perturbations such as aerodynamic drag, solar-lunar perturbations and so forth.

The position vector perturbations due to the J_2 oblateness effect only are given by

$$\ddot{r}_x = -\mu \frac{r_x}{r^3} \left\{ \frac{3}{2} \frac{J_2 r_{eq}^2}{r^2} \left(1 - \frac{5 r_z^2}{r^2} \right) \right\}$$

$$\ddot{r}_y = -\mu \frac{r_y}{r^3} \left\{ \frac{3}{2} \frac{J_2 r_{eq}^2}{r^2} \left(1 - \frac{5 r_z^2}{r^2} \right) \right\}$$

$$\ddot{r}_z = -\mu \frac{r_z}{r^3} \left\{ \frac{3}{2} \frac{J_2 r_{eq}^2}{r^2} \left(3 - \frac{5 r_z^2}{r^2} \right) \right\}$$

where $r = \sqrt{r_x^2 + r_y^2 + r_z^2}$. In these equations μ and r_{eq} are the gravitational constant and equatorial radius of the Earth, respectively.

The information required for this script can either be manually input by the user or the software can read a simple data file of orbital elements. For both options the program will always ask the user for the initial calendar date and Universal Time of the simulation, the simulation duration in days, and the algorithm error tolerance. A tolerance of `1.0e-8` is recommended.

The following is a typical user interaction with this program.

```
                program encke1

  < geocentric orbit propagation using Encke's method >


  initial calendar date and universal 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)
  ? 10,20,30

  please input the simulation period (days)
  ? 1

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

```
   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)
   ? leo.dat
```

This program uses the `readoe1` function to process the input data file. The following is the screen display created by this script for this example.

```
                    program encke1

< geocentric orbit propagation using Encke's method >

initial calendar date       01-Jan-2001
initial universal time       10:20:30.000

final calendar date         02-Jan-2001
final universal time         10:20:30.000

        sma (km)              eccentricity           inclination (deg)        argper (deg)
 +6.88246100825183e+003   +1.20352973737767e-002   +2.85337139238701e+001  +2.86590530109220e+002

       raan (deg)            true anomaly (deg)         arglat (deg)           period (min)
 +3.82678124572323e+001   +7.72067788625648e+001   +3.79730897178449e+000  +9.47055373551450e+001
```

Here are the contents of the `leo.dat` data file used in this example.

```
   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
```

encke2.m – Encke's method for heliocentric orbits

This MATLAB application demonstrates heliocentric orbit propagation using Encke's method. It includes the point mass gravity perturbations due to the planets Venus, Earth, Mars, Jupiter, Saturn and the Sun.

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

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

where

μ_s = gravitational constant of the Sun

μ_{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

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

The information required for this script can either be manually input by the user or the software can read a simple data file of orbital elements. For both options the program will always ask the user for the initial calendar date and Universal Time of the simulation, the simulation duration in days, and the algorithm error tolerance. A tolerance of `1.0e-8` is recommended.

The following is a typical user interaction with this program.

```
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)
? 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 of 1.0e-8 is recommended)
? 1e-8
```

This program uses the `readoe2` function to process the input data file. The following is the screen display created by this software for this example.

```
                    program encke2

 < heliocentric orbit propagation using Encke's method >


 final conditions

 calendar date       01-May-1986

 universal time      00:00:00.000

      sma (km)          eccentricity      inclination (deg)     argper (deg)
 2.6903152122e+009   9.6733200476e-001   1.6224898622e+002   1.1181332521e+002

      raan (deg)       true anomaly (deg)    arglat (deg)        period (days)
 5.8130641930e+001   1.0750909439e+002   2.1932241960e+002   2.7855763280e+004


 perihelion radius    5.8748967582e-001 AU
```

Here are the contents of the `halley.dat` data file used in this example.

```
 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)
58.129
```