CSX/ECE 6730 Project 1: Cowell's Method

Team 57: Blaine Costello, Caitlyn Caggia, Lakshmi Raju

Spring 2019

Cowell's Method Background

Cowell's method for orbital simulation simply sums the Newtonian forces on a body of *i* from the other bodies *j:*

$$\ddot{\mathbf{r}}_i = \sum_{j=1,\, j\neq i}^{n} \frac{Gm_j(\mathbf{r}_j - \mathbf{r}_i)}{r_{ij}^3}$$

where

$\ddot{\mathbf{r}}_i$ is the acceleration vector of body *i*

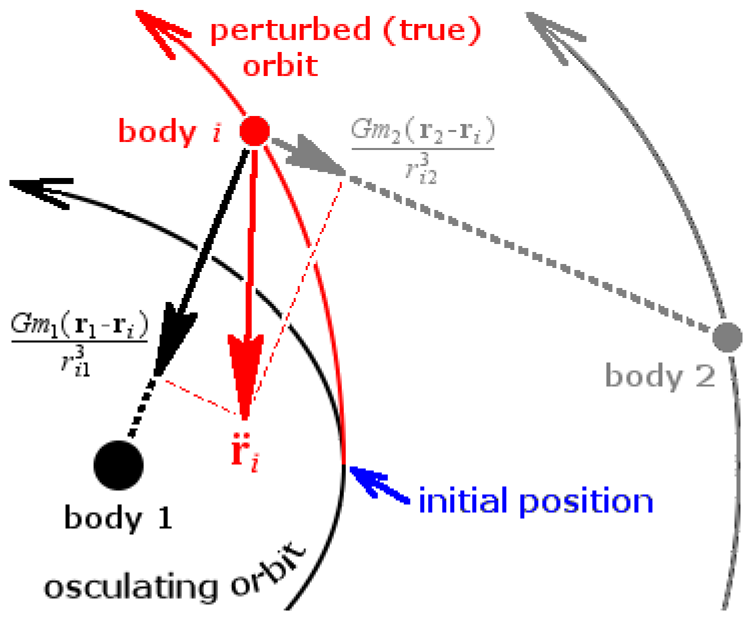
G is the gravitational constant ($6.674 \times 10^{-11}\,\mathrm{m^3\,kg^{-1}\,s^{-2}}$)

m_j is the mass of body j

$\mathbf{r}_i$ and $\mathbf{r}_j$ are the position vectors of objects *i* and *j*

r_{ij} is the distance from object *i* to object *j*

with all vectors relative to the barycenter of the system.


perturbed (true) orbit
body i
$\frac{Gm_2(\mathbf{r}_2-\mathbf{r}_i)}{r_{i2}^3}$
$\frac{Gm_1(\mathbf{r}_1-\mathbf{r}_i)}{r_{i1}^3}$
body 2
$\ddot{\mathbf{r}}_i$
initial position
body 1
osculating orbit

Forces from preturbing bodies (black and gray) are summed to form the total force on the body of interest (red). Thus the true (red) orbit deviates from the original unpreturbed orbit (black). Image courtesy of Wikimedia.

While this method is the among the simplest to use, it has high error when handling large perturbations (for example, when two objects nearly miss each other) and requires a large degree of arithmetic precision when the mass of the bodies varies greatly.

Input Setup

Mass Data

Masses of the Sun and 9 planets (Mercury, Venus, Earth, Moon, Mars, Jupiter, Saturn, Uranus, Neptune, Pluto, respectively) courtesy of NASA. Mass is saved as a 1xN vector in kilograms.

```
% Saving these in case we need easier numbers to try solving at first...
% distances = [0 57.9 108.2 149.6 227.9 778.6 1433.5 2872.5 4495.1 5906.4]; % [km]
% velocities = [0 47.4 35.0 29.8 24.1 13.1 9.7 6.8 5.4 4.7]; % [km/s]

% We will keep masses regardless...
                   % Sun  Merc  V    E    Moon  Mars  J    S   U    N   P
 masses = 10^24.*[1988500 0.330 4.87 5.97 0.073 0.642 1898 568 86.8 102 0.0146]; % [kg]
 N = length(masses); % Number of bodies
```

Position and Velocity Data

Note that to use this data, you need Matlab's Aerospace Toolbox (available with the Georgia Tech's Matlab licenses for free) as well as the Ephemeris Data add-on available through Mathworks for free. These packages use data from NASA's Jet Propulsion Laboratory (JPL) ephemeris database.

The barycenter of the Solar System is used as the observation point of reference as of January 1, 2000. Distances are in km and velocities are in km/s. Positions and velocities are saved in 3xN vectors.

```
[posSun, velSun] = planetEphemeris(juliandate(2000,1,1),'SolarSystem','Sun');
[posMerc, velMerc] = planetEphemeris(juliandate(2000,1,1),'SolarSystem','Mercury');
[posV, velV] = planetEphemeris(juliandate(2000,1,1),'SolarSystem','Venus');
[posE, velE] = planetEphemeris(juliandate(2000,1,1),'SolarSystem','Earth');
[posMoon, velMoon] = planetEphemeris(juliandate(2000,1,1),'SolarSystem','Moon');
[posMars, velMars] = planetEphemeris(juliandate(2000,1,1),'SolarSystem','Mars');
[posJ, velJ] = planetEphemeris(juliandate(2000,1,1),'SolarSystem','Jupiter');
[posS, velS] = planetEphemeris(juliandate(2000,1,1),'SolarSystem','Saturn');
[posU, velU] = planetEphemeris(juliandate(2000,1,1),'SolarSystem','Uranus');
[posN, velN] = planetEphemeris(juliandate(2000,1,1),'SolarSystem','Neptune');
[posP, velP] = planetEphemeris(juliandate(2000,1,1),'SolarSystem','Pluto');

positions = [posSun(:) posMerc(:) posV(:) posE(:) posMoon(:) posMars(:) posJ(:) ...
    posS(:) posU(:) posN(:) posP(:)]; % [km]
velocities = [velSun(:) velMerc(:) velV(:) posE(:) velMoon(:) velMars(:) velJ(:) ...
```

```
    velS(:) velU(:) velN(:) velP(:)]; % [km/s]
```

Vector Solver

[Description to be added]

```
G = 6.674e-11; % Gravitational constant
accel = 0;

% Influence of the other 10 bodies on the Earth
for tstep = 1:T
    accel = zeros(3,N);
    for i = 1:N
        for j = 1:N
            if i ~= j
                d = sqrt((positions(1,j)-positions(1,i))^2 + ...
                    (positions(2,j)-positions(2,i))^2 + ...
                    (positions(3,j)-positions(3,i))^2);
                accel(1,i) = (accel(1,i)+(G*masses(j)*(positions(1,j)...
                    -positions(1,i))) / (d.^3));
                accel(2,i) = (accel(2,i)+(G*masses(j)*(positions(2,j)...
                    -positions(2,i))) / (d.^3));
                accel(3,i) = (accel(3,i)+(G*masses(j)*(positions(3,j)...
                    -positions(3,i))) / (d.^3));
            end
        end
    end
end
```

Plot Generator

[Description to be added]

```
%Code to be added
```