CSX/ECE 6730 Project 1: Orbit Modeling

Team 57: Blaine Costello, Caitlyn Caggia, Lakshmi Raju

March 1, 2019

INPUT SETUP

Additional files: planetData.m

Mass Data

Masses of the Sun and 9 planets (Mercury, Venus, Earth, Moon, Mars, Jupiter, Saturn, Uranus, Neptune, Pluto, respectively) courtesy of NASA (page 49). Mass is saved as a 1xN vector as GM in au^3/day^2.

Position and Velocity Data

Initial position and velocity vectors were pulled from NASA data for June 28, 1969 (page 47).

The barycenter of the Solar System is used as the observation point of reference. Distances are in au and velocities are in au/day. Positions and velocities are saved in 3xN vectors.

Expected Values

Actual positions and velocities for June 28, 1970 were pulled from NASA JPL data through Matlab's built-in `planetEphemeris()` function. These values have less significant figures than the data for 1969 above, but are useful to compare each model's calculations to the real position of each body.

Note that 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) are needed to use this function. These packages use data from NASA's Jet Propulsion Laboratory (JPL) ephemeris database.

```
clear all; close all;
addpath('Data');
run planetData.m;
```

COWELL'S METHOD

Additional files: cowells.m, cowellsExample.avi

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*

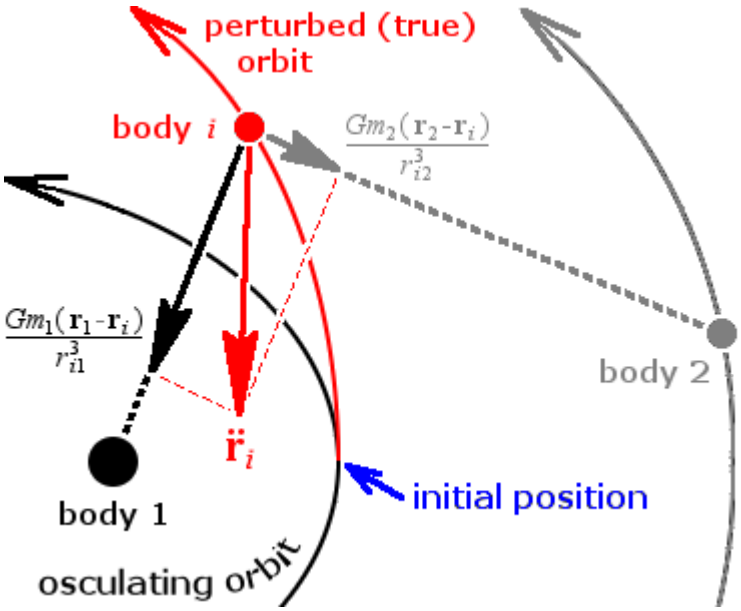
G is the gravitational constant ($6.674 \times 10^{-11}\ \mathrm{m^3 kg^{-1} s^{-2}}$)

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.



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.

Cowell's Method Example

The situation to be simulated is the orbital movement of a system of 10 bodies around the Sun: Mercury, Venus, Earth, Moon, Mars, Jupiter, Saturn, Uranus, Neptune, Pluto. To do this, Cowell's Method is used to estimate the perturbations of the bodies on each other and model the movement of the system.

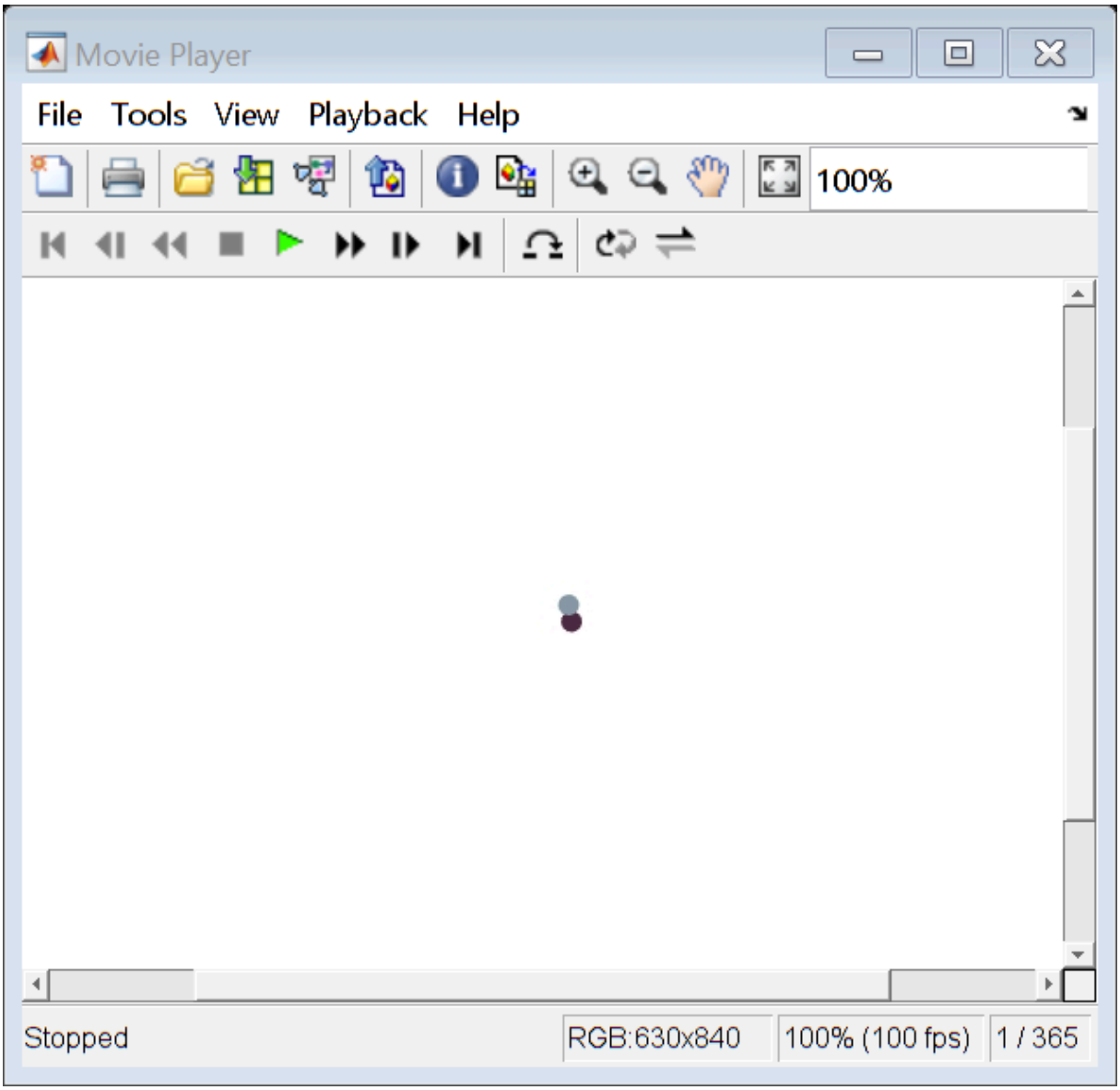
Run the simulation for one full year (on Earth).

```
%[finalPos, finalVel] = cowells('cowellsExample.avi', 365, 100, true);
implay('cowellsExample.avi')
```

To get an accurate representation of the orbit, very small time steps are required. We tested three timesteps (0.1, 0.01, and 0.001 days) running for a total of 1000 days. Although the time steps seems small, they are actually still not small enought to accurately simulate the moon's trajectory. We are limited by computations, running a more accurate simulation for three years would require over 10GB of data, and take a lot longer than practical with our resources. The resulting videos are too large to host in github, so they are in a google drive: https://drive.google.com/open?id=1y0ewrl9TxiLYopDY_j2Ry0_34iph4zjs.

The simulation was also run to simulate just the Moon around the Earth. The Cowell's Method code allows for N bodies, simply changing the constants in the planets.mat file which is loaded in line 10 in the cowells.m function to include the wanted bodies, allows for various simulations.

```
load planets.mat masses positions velocities N;
masses11 = masses; positions11 = positions; velocities11 = velocities; N11 = N; %Saving old dat
masses = masses11(4:5); positions = positions11(:,4:5); velocities = velocities11(:,4:5); N = 2
save('planets', 'masses', 'positions', 'velocities', 'N');
%[finalPos2, finalVel2] = cowells('cowellsExample2.avi', 365, 100, true);
implay('cowellsExample2.avi')
```




```
masses = masses11; positions = positions11; velocities = velocities11; N = N11;
save('planets', 'masses', 'positions', 'velocities', 'N'); %Reverting planets.mat back to 11 bo
```

Cowell's Method Error

The calculated values were compared against actual measurements with a simple percent error calculation for position and velocity.

```
load planetsExpected.mat expectedPos expectedVel names;
errorPos = (finalPos-expectedPos)./expectedPos .* 100;
errorVel = (finalVel-expectedVel)./expectedVel .* 100;
cowellError = [num2cell(errorPos); num2cell(errorVel)];
rownames = [{'% error in x-position'}; {'% error in y-position'}; ...
    {'% error in z-position'}; {'% error in x-velocity'}; ...
    {'% error in y-velocity'}; {'% error in z-velocity'}];
cowellTable = cell2table(cowellError);
cowellTable.Properties.RowNames = rownames;
```

```
cowellTable.Properties.VariableNames = names;
display(cowellTable)
```

```
cowellTable = 6×11 table
```

...

	Sun	Mercury	Venus	Earth
1 % error in x-position	-4.9190	-706.2148	-193.3463	-751.0916
2 % error in y-position	-4.3812	-375.2005	-523.0428	-5.1985
3 % error in z-position	-5.2283	-299.7625	-1.0906e+03	-5.1953
4 % error in x-velocity	14.6120	-92.0968	-407.4205	-29.2665
5 % error in y-velocity	-7.9214	-173.1120	-161.0833	-628.5896
6 % error in z-velocity	-8.7778	-153.5519	-169.0159	-628.7740

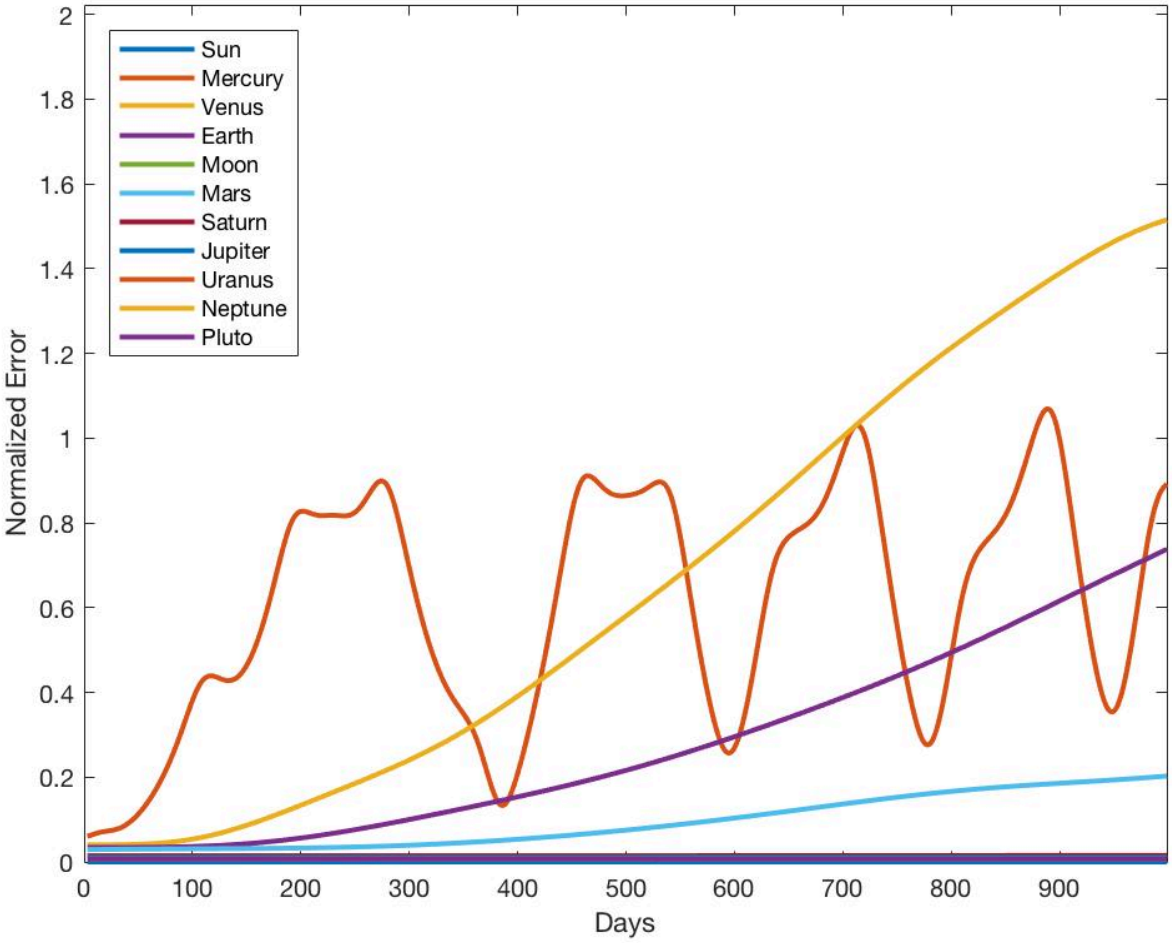
```
% Excluding the Moon...
noMoonPos = [errorPos(:,1:4) errorPos(:,6:end)];
noMoonVel = [errorVel(:,1:4) errorVel(:,6:end)];
avgPosError = sum(sum(noMoonPos)) / (length(noMoonPos));
avgVelError = sum(sum(noMoonVel)) / (length(noMoonVel));
sprintf(['Average Position Error: %4.1f%%\n'...
    'Average Velocity Error: %4.1f%%'], avgPosError, avgVelError)
```

```
ans =
    'Average Position Error: -396.2%
     Average Velocity Error: -253.9%'
```

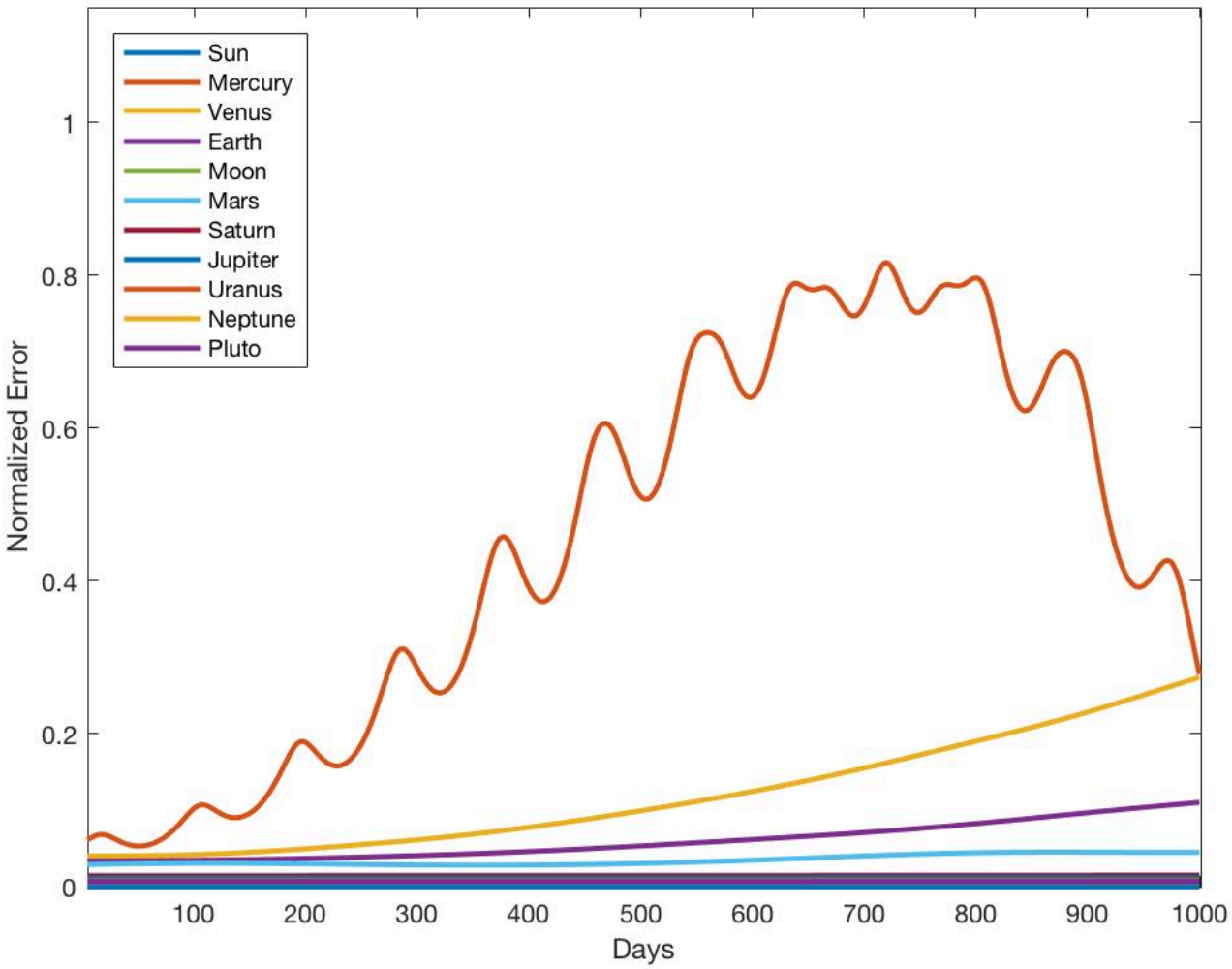
The reason the moon's orbit results in such large error is due to the size of the timesteps. The smaller scale of the moon's orbit in relation to the planetary orbits requires more precision.

The normalized error for the 11 body system at the three timesteps (0.1, 0.01, 0.001 days) is shown below. The error is normalized dividing by the magnitude of error at each timestep by the radius of the orbit for each planet.

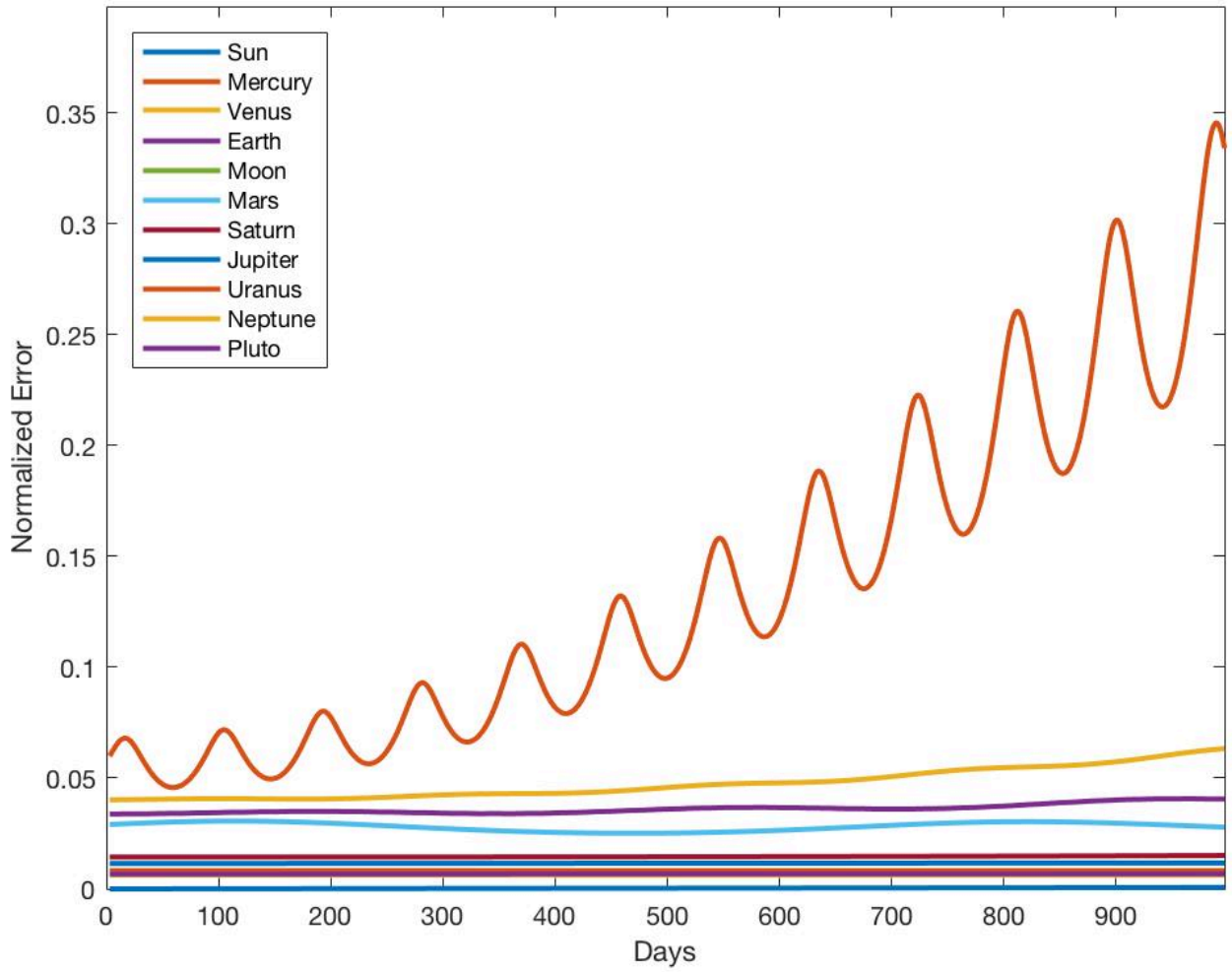
Timestep: 0.1 days:



Timestep: 0.01 days:



Timestep: 0.001 days:



ENCKE'S METHOD

Additional files: enckes.m, enckesExample.avi

Encke's Method Background

details

Encke's Method Example

```
% [finalPos, finalVel] = enckes('enckesExample.avi', 365, 10);
% implay('enckesExample.avi')
```

Encke's Method Error

Division of Labor

All three team members worked on the project equally. The Cowell's Method was primarily written by Blaine Costello and the Encke's Method was primarily written by Caitlyn Caggia. The written tutorial was completed by Lakshmi Raju. All three members debugged code, and finalized the project together.