INTERSTELLAR N-BODY PROBLEM

Ashvat Bansal
University of Illinois Urbana Champaign

April 3, 2019

1 Introduction

The N-Body problem is one of the most important problems used to model the motion and astrodynamics of celestial bodies and satellites, though it can be applied to any system of bodies. This paper explores the application of the N-Body problem to arguably the most important collection of celestial bodies to humans, which is the Solar System, using the governing equation:

$$m_i \frac{d^2\vec{r}}{dt^2} = \sum_{j=1, j\neq i}^{N} \frac{G m_i m_j (r_j - r_i)}{\|r_j - r_i\|^3} \tag{1}$$

The above equation is used to formulate our state space model in the form of f(u), which was used in the numerical methods for analysis.

For ease of comprehension, the Solar System problem was broken down into 2 different N-Body problems:

1.1 Inner Solar System

The first system is a 6-Body Problem analyzing the Inner Solar System and the Moon (Sun, Mercury, Venus, Earth, Mars, Moon). This system is analyzed using the Sun as the central mass, which is depicted using a 3 dimensional point at the center, while the the inner planets (and Moon) are plotted around the Sun using 3 dimensional lines.

1.2 Outer Solar System

The second N-Body System is also a 6-Body System, but it's analyzing the Outer Solar System, including Pluto (Jupiter, Saturn, Uranus, Neptune, Pluto). Again, the system is analyzed with Sun acting as the central mass, depicted by a 3 dimensional point, while the outer planets (and Pluto) are plotted around the Sun using 3 dimensional lines.

2 The Numerical Methods

The numerical methods chosen to analyze the systems were the 2-Step Adams-Bashforth method and the 4th Order Runge-Kutta method.

2.1 Description/Overview

2-Step Adams-Bashforth Method (AB2)

Adams methods are based on the idea of approximating the integrand with a polynomial within the interval (u_k, u_{k+1}) [1]. Using a kth order polynomial results in a $k+1$th order

[1] http://web.mit.edu/10.001/Web/Course$_N otes/Differential_E quations_N otes/node6.html$

method. The explicit Adams method is known as the Adams-Bashforth method, and the 2-Step Adams-Bashforth method is define as [2]:

$$u_{k+1} = u_k + \frac{\Delta t}{2}[-f(u_{k-1}, t_{k-1}) + 3f(u_k, t_k)] \tag{2}$$

The truncation error of the 2-step Adams-Bashforth method is $\mathcal{O}(\Delta t^2)$

4th Order Runge-Kutta Method (RK4)

Runge-Kutta methods are a class of methods which judiciously uses the information on the 'slope' at more than one point to extrapolate the solution to the future time step [3]. The Fourth Order Runge-Kutta method, also known as RK4, is a multi-stage method and is one of the more commonly used methods for analysis, used to solve equations of the form $\dot{y} = f(t, y)$, where y is the function that is being approximated [4]. The initial parameters t_0 and y_0 are required, and the general solution is given by:

$$u_{k+1} = u_k + \frac{1}{6}\Delta t\Big(y_1 + 2y_2 + 2y_3 + y_4\Big) \tag{3}$$

where

$$\begin{aligned} y_1 &= f(u_k, t_k) \\ y_2 &= f\left(u_k + \frac{1}{2}\Delta t y_1, t_k + \frac{1}{2}\Delta t\right) \\ y_3 &= f\left(u_k + \frac{1}{2}\Delta t y_2, t_k + \frac{1}{2}\Delta t\right) \\ y_4 &= f\left(u_k + \Delta t y_3, t_k + \Delta t\right) \end{aligned} \tag{4}$$

Since RK4 is a fourth-order method, the local truncation error is $\mathcal{O}(\Delta t^5)$, while the accumulated error is $\mathrm{O}(\Delta t^4)$

2.2 Justification for the Choice of Methods

2.2.1 2-Step Adam-Bashforth Method

The 2-Step Adam-Bashforth method builds upon the the principle of the predictor-correct of the Huen's Numerical Method. The advantage a higher order Adams-Bashforth method has over Huen is that it only used 1 more function evaulation per step to achieve a higher accuracy. This method is also suited for evaluation of problems with output at many points, which is what we're doing by plotting the orbits from our N-Body systems.[5].

The 2-step Order Adams-Bashforth has the stable region that is expected of explict methods, which is shown below:

[2] Lecture 2 26 Typed

[3] http://web.mit.edu/10.001/Web/Course$_{N}otes/Differential_{E}quations_{N}otes/node5.html$

[4] Lecture 2 24 Typed

[5] https://www.phy.ornl.gov/csep/ode/node12.html

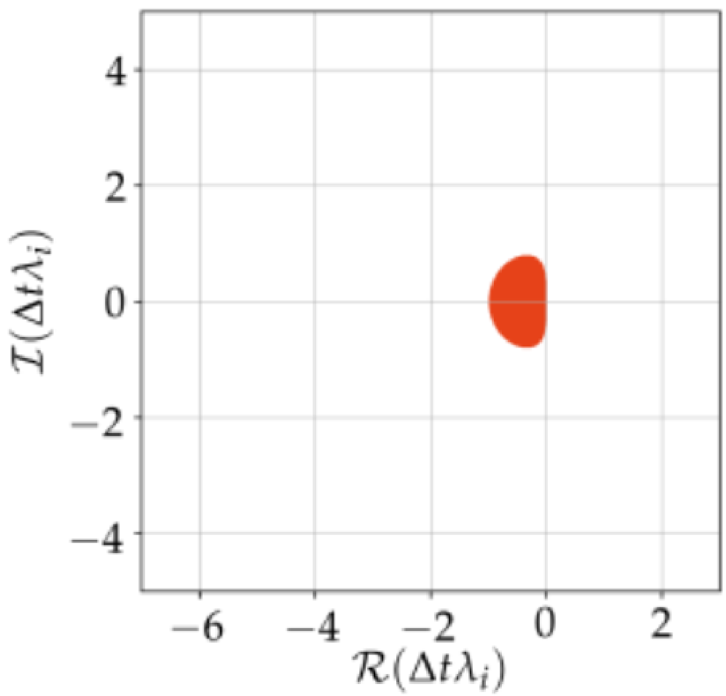

Figure 1: Stability Region for 2-Step Adams-Bashforth Method[6]

2.2.2 RK4

The RK4 Method is one of the most flexible methods tht can be used for analysis. It provides a local truncation error of $\mathcal{O}(\Delta t^5)$ and the accumulated error is $O(\Delta t^4)$, which makes it suitable for applications where not a computational power is available, but the error still needs to be minimized.

The stability region for the 4th Order Runge-Kutta method is given below:

[6] Week 7 Typed Notes

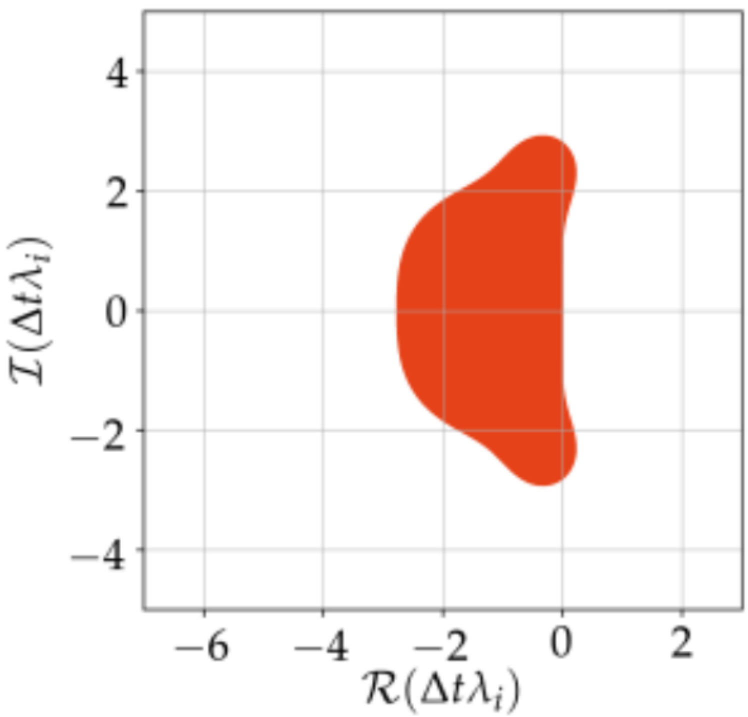

Figure 2: Stability Region for 4^{th} Order Runge-Kutta Method [7]

2.3 Demonstration of Correct Implementation

Using the Convergence plots similar to the ones we did for Homework 5, it can be shown that the both the methods (Adam-Bashforth 2 and Runge-Kutta 4) were implemented correctly for both the systems.

Inner Solar System

For the Inner Solar System, 6 different Δts were used. The Δts used were: [6.25 days, 1.25 days, 0.25 of 1 day, 0.05 of 1 day, 0.01 of 1 day, 0.002 of 1 day] (in seconds), and the total time for simulation was chosen as 1 Mars year (687 days in seconds)

[7] Week 7 Typed Notes

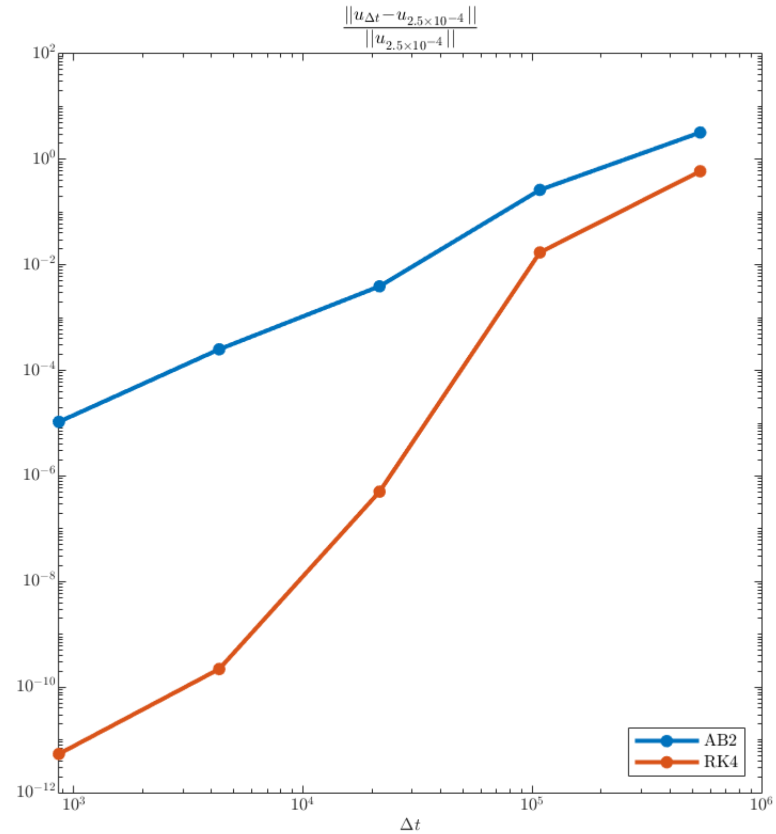

Figure 3: Convergence Plots for Inner Solar System

Outer Solar System

For the Outer Solar System, again 6 different Δts were used. The Δts used were: [781.25 days, 156.25 days, 31.25 days, 6.25 days, 1.25 days, 0.25 of 1 day] (in seconds), and the total time for simulation was chosen as 250 years (in seconds).

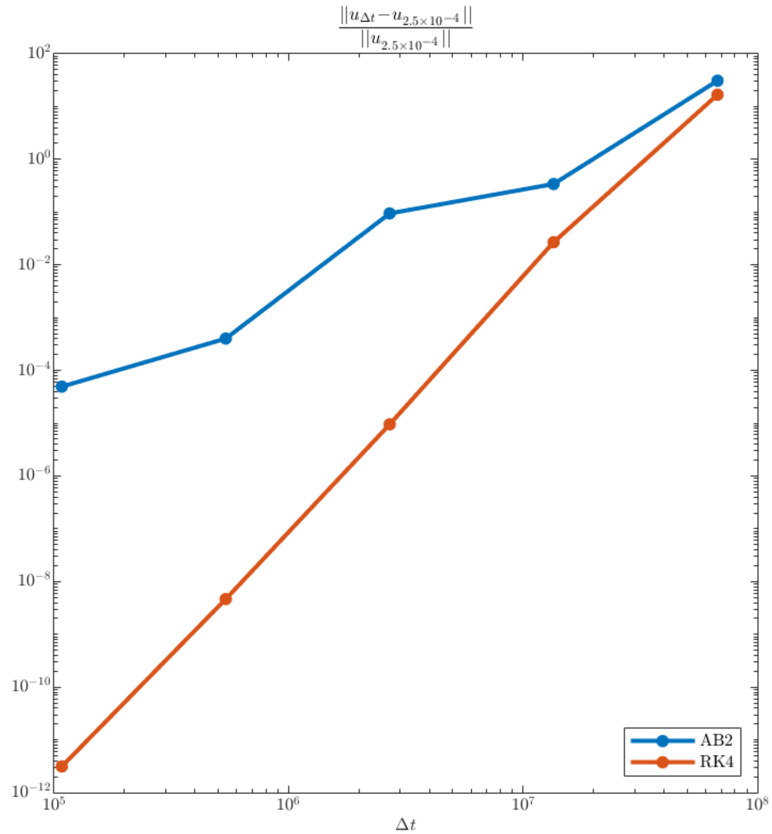

Figure 4: Convergence Plots for Inner Solar System

2.3.1 Convergence Plots Results

As it can be seen in the above plots, difference between the estimations decreases as the difference between the smallest Δt and the other Δts decreases. Furthermore, we can also see that the difference for RK4 method is much smaller than AB2, and decreases at fastest rate than AB2, which implies that RK4 method is more accurate estimation than compared to AB2, and hence can conclude that the method have been implemented correctly.

3 Results for the N-Body Problem

As mentioned before, 2 different systems were analyzed using the methods defined in Section 2. Using the N-Body governing equation and Newton's Second Law of Motion (F = ma), the following state space model was derived:

$$u = \begin{bmatrix} r_1 \\ v1 \\ \vdots \\ r_j \\ v_j \end{bmatrix}, \quad f = @(u) \begin{bmatrix} v_1; & G\frac{m_2(r_2-r_1)}{||r_2-r_1||^3} & + & G\frac{m_3(r_3-r_1)}{||r_3-r_1||^3} & + & ... & G\frac{m_j(r_j-r_1)}{||r_j-r_1||^3} \\ v_2; & G\frac{m_1(r_1-r_2)}{||r_1-r_2||^3} & + & G\frac{m_3(r_3-r_2)}{||r_3-r_2||^3} & + & ... & G\frac{m_j(r_j-r_2)}{||r_j-r_2||^3} \\ v_3; & G\frac{m_2(r_2-r_3)}{||r_2-r_3||^3} & + & G\frac{m_1(r_2-r_3)}{||r_2-r_3||^3} & + & ... & G\frac{m_j(r_j-r_1)}{||r_j-r_3||^3} \\ \vdots & \vdots & + & ... & + & ... & \vdots \\ v_i; & G\frac{m_j(r_j-r_i)}{||r_j-r_i||^3} & + & G\frac{m_j(r_j-r_i)}{||r_j-r_i||^3} & + & ... & G\frac{m_j(r_j-r_i)}{||r_j-r_i||^3} \end{bmatrix} \tag{5}$$

where

$$\vec{r_j} = \begin{bmatrix} (r_x)_i \\ (r_y)_i \\ (r_z)_i \end{bmatrix}, \quad \vec{v_j} = \begin{bmatrix} (\dot{r}_x)_i \\ (\dot{r}_y)_i \\ (\dot{r}_z)_i \end{bmatrix}, \quad \text{for i = 1,2,3...N} \tag{6}$$

Here, r_j, r_i and v_i are position and velocity vectors for the i^{th} and j^{th} body. The initial positions and velocities were obtained using the `MATLAB` function `planetEphemeris()`, which draws its data from the NASA_Horizons.

The 2 systems are talked about in more detail in the following sections:

3.1 Inner Solar System

Parameters

Since we're modelling the Inner Solar System, the parameters were chosen to replicate the real values of the celestial bodies of the Inner Solar System:

$$\begin{bmatrix} \text{1st Body} = \text{Sun} \\ \text{2nd Body} = \text{Mercury} \\ \text{3rd Body} = \text{Venus} \\ \text{4th Body} = \text{Earth} \\ \text{5th Body} = \text{Mars} \\ \text{6th Body} = \text{Moon} \end{bmatrix}, \quad \begin{bmatrix} m_1 = 1.9891 \cdot 10^{30} \\ m_2 = 3.285 \cdot 10^{23} \\ m_3 = 4.867 \cdot 10^{24} \\ m_4 = 5.97 \cdot 10^{24} \\ m_5 = 7.35 \cdot 10^{22} \\ m_6 = 7.34767 \cdot 10^{22} \end{bmatrix}, \quad \begin{bmatrix} G = 6.67 \cdot 10^{-11} \\ \text{Start Time } (t_0) = 0 \\ \text{Time Duration (T)} = 59356800(s); \\ \Delta t = 60 \cdot 60 \cdot 24 \cdot 0.002 \end{bmatrix}$$

The actual positions and velocities of these celestial bodies were gathered for the date of January 1st, 2011, using the code below:

```
% Gathering N-Body Information for the date of Jan 1, 2011
Julian = juliandate(2011,1,1);

% Position and Velocity Vectors
[r1, v1] = planetEphemeris(Julian,'Sun', 'Sun', '432t', 'km');
[r2, v2] = planetEphemeris(Julian,'Sun','Mercury','432t','km');
[r3, v3] = planetEphemeris(Julian,'Sun','Venus','432t','km');
[r4, v4] = planetEphemeris(Julian,'Sun','Earth','432t','km');
[r5, v5] = planetEphemeris(Julian,'Sun','Mars','432t','km');
[r6, v6] = planetEphemeris(Julian,'Sun','Moon','432t','km');
```

As shown above, the masses of the bodies were taken to be the real values for the masses of the celestial bodies. Since we're modelling the system till Mars only, the time duration was taken to be 687 days (in seconds), which is the time period for 1 revolution of Mars around the Sun. Δt was taken as 0.002 of a day (in seconds), to provide a small enough time step for high accuracy without making the simulation too taxing on the computer and too time consuming.

Simulation

The Simulation for the motion of the inner planets was run using both the AB2 numerical method as well as the RK4 numerical method, and the following plots were produced:

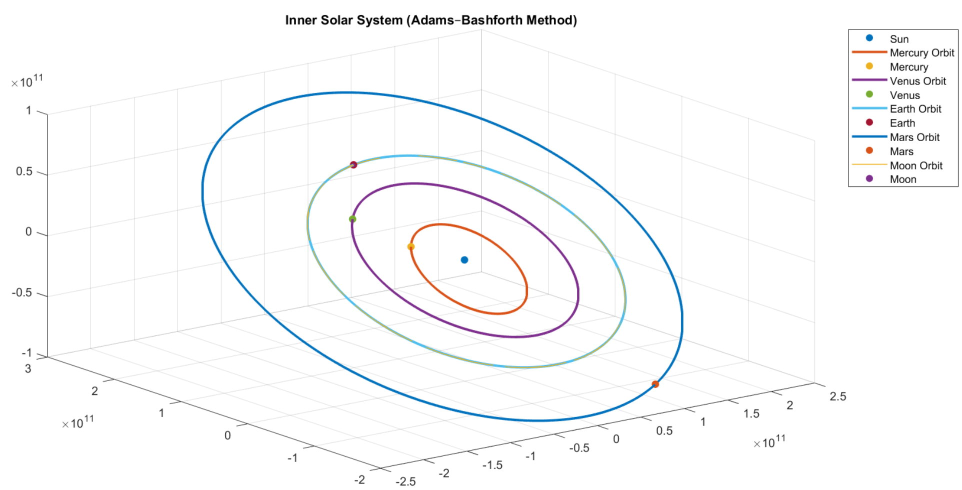

Figure 5: Inner Orbits Using AB2

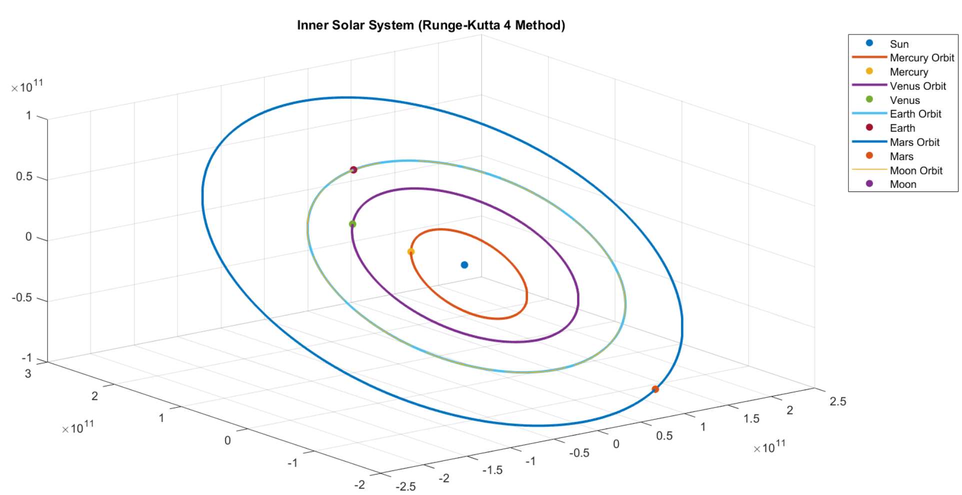

Figure 6: Inner Orbits Using RK4

Using the eye-ball test, we can see that the both the methods were able to model the orbits of the 4 planets with high accuracy. One thing that might seem out of ordinary at first is that the orbit for moon is superimposed with the orbit of the Earth, but this can easily be justified since looking from far, the Moon's orbit would look almost exactly the same as Earth's, since relatively there is negligible distance between the moon and the Earth when compared to the Solar System.

Position and Velocity Plots

At the time of simulation end, the calculated vs actual Positions and Velocities of the planets were tabulated for comparison

Planetary_Bodies	Calculated_Position_AB2			JulianDate_Position_AB2		
'Mercury'	26892815748.154	34678387802.6579	15736013382.732	24852374725.4274	35643977568.0656	16463728572.1887
'Venus'	-100178516244.953	32952294868.8072	21164967692.5706	-100606507809.529	31902535991.8289	20720221209.5169
'Earth'	83878558855.0063	111759444878.719	48449553081.4366	82959909056.5171	112313015236.478	48689084986.6857
'Mars'	84856760434.175	-175103902553.041	-82607580340.5919	85705499726.6823	-174689184516.097	-82439472766.6012
'Moon'	83890622500.4252	111420637135.783	48324178935.6857	83100955275.8817	111990719148.075	48580413849.9717

Figure 7: Position Table AB2

Planetary_Bodies	Calculated_Position_RK4			JulianDate_Position_RK4		
'Mercury'	26892941976.1258	34678333081.3997	15735971064.6381	24852374725.4274	35643977568.0656	16463728572.1887
'Venus'	-100178514954.027	32952297798.7946	21164968929.1539	-100606507809.529	31902535991.8289	20720221209.5169
'Earth'	83878560218.5983	111759444181.085	48449552789.512	82959909056.5171	112313015236.478	48689084986.6857
'Mars'	84856759685.1042	-175103902977.15	-82607580514.888	85705499726.6823	-174689184516.097	-82439472766.6012
'Moon'	83890614076.1382	111420637240.687	48324178126.2867	83100955275.8817	111990719148.075	48580413849.9717

Figure 8: Position Table RK4

Planetary_Bodies	Calculated_Final_V_AB2	JulianDate_Final_V_AB2	CalculatedMeanVelocity_AB2	KnownMeanVelocity_AB2
'Mercury'	58311.6969716765	58483.9617378529	47107.6311528898	47360
'Venus'	35232.9295888579	35231.7236180981	35014.0530873992	35020
'Earth'	30135.2585750645	30138.1692293102	29744.9622604226	29780
'Mars'	25949.0332069976	25957.0929037514	24068.4690724357	24070
'Moon'	29239.8431624951	29509.3328422083	29744.1118434967	1022

Figure 9: Velocity Table AB2

Planetary_Bodies	Calculated_Final_V_RK4	JulianDate_Final_V_RK4	CalculatedMeanVelocity_RK4	KnownMeanVelocity_RK4
'Mercury'	58311.677490386	58483.9617378529	47107.6300591996	47360
'Venus'	35232.9296089756	35231.7236180981	35014.0530825176	35020
'Earth'	30135.2587425897	30138.1692293102	29744.9622556731	29780
'Mars'	25949.0331934258	25957.0929037514	24068.4690674062	24070
'Moon'	29239.8277871024	29509.3328422083	29744.1119112235	1022

Figure 10: Velocity Table RK4

As it can be seen, the Known and actual values are very close to the calculated values, which again shows that the methods was implemented correctly

Error

To test the accuracy of methods, error between the estimation and the actual values was plotted for multiple Δt's. The real values of all positions and velocities were again obtained using `MATLAB` function `planetEphemeris` for the date of 19th November, 2012, since that is exactly 687 days away from January 1st, 2011, and then the overall error was plotted again Δt.

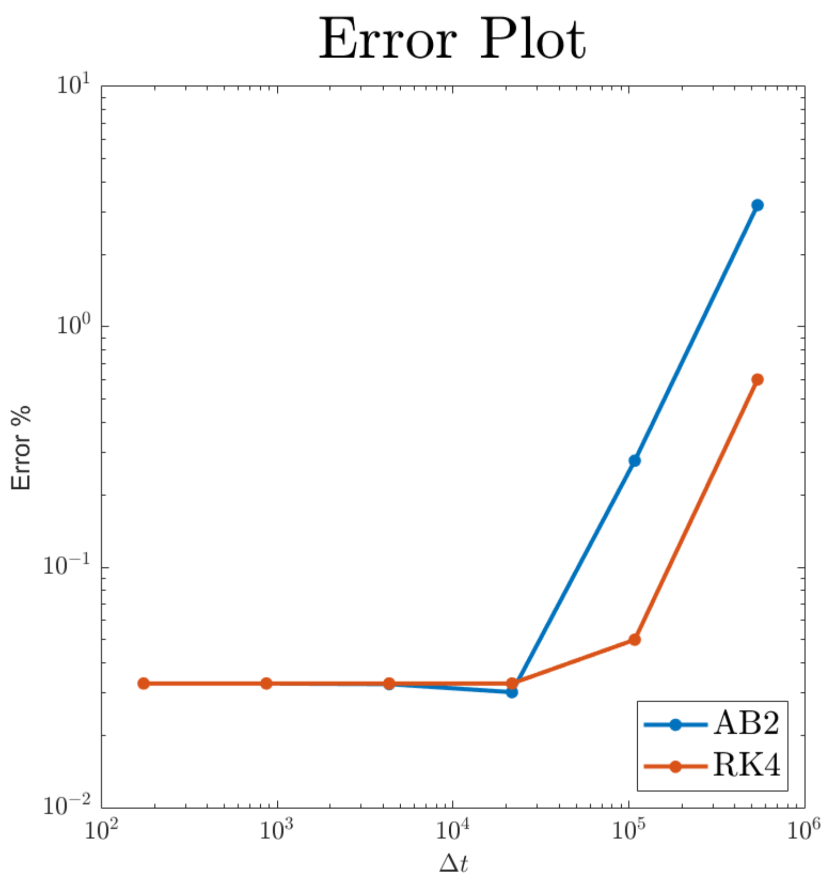

Figure 11: Error Plot for Inner Orbits

As it can be seen above, the overall error from both the methods is just above 3%, which I believe is an acceptable margin of error. After running the error plot though, it can be seen that the value of Δt chosen could have been higher than what was chosen, since the error stabilizes at Δt's much bigger than the chosen one, but I had wanted to get the results as accurate as possible, hence I had chosen such a small Δt.

3.2 Outer Solar System

Parameters

Since the second system is the model of the Outer Solar System, the parameters were chosen to replicate the real values of the celestial bodies in the Outer Solar System:

$$\begin{bmatrix} \text{1st Body} = \text{Sun} \\ \text{2nd Body} = \text{Jupiter} \\ \text{3rd Body} = \text{Saturn} \\ \text{4th Body} = \text{Uranus} \\ \text{5th Body} = \text{Neptune} \\ \text{6th Body} = \text{Pluto} \end{bmatrix}, \begin{bmatrix} m1 = 1.9891 \cdot 10^{30} \\ m2 = 1.898 \cdot 10^{27} \\ m3 = 5.683 \cdot 10^{26} \\ m4 = 8.681 \cdot 10^{25} \\ m5 = 1.024 \cdot 10^{26} \\ m6 = 1.309 \cdot 10^{22} \end{bmatrix}, \begin{bmatrix} G = 6.67 \cdot 10^{-11} \\ \text{Start Time } (t_0) = 0 \\ \text{Time Duration (T)} = 7884000000(s); \\ \Delta t = 60 \cdot 60 \cdot 24 \cdot 0.25 \end{bmatrix}$$

Again, the actual positions and velocities of these celestial bodies were gathered for the date of January 1st, 2011, using the code below:

```
% Gathering N-Body Information for the date of Jan 1, 2011
Julian = juliandate(2011,1,1);

% Position and Velocity Vectors
[r1, v1] = planetEphemeris(Julian,'Sun', 'Sun', '432t', 'km');
[r2, v2] = planetEphemeris(Julian,'Sun','Jupiter','432t','km');
[r3, v3] = planetEphemeris(Julian,'Sun','Saturn','432t','km');
[r4, v4] = planetEphemeris(Julian,'Sun','Uranus','432t','km');
[r5, v5] = planetEphemeris(Julian,'Sun','Neptune','432t','km');
[r6, v6] = planetEphemeris(Julian,'Sun','Pluto','432t','km');
```

As shown above, the masses of the bodies were taken to be the real values for the masses of the celestial bodies. Since we're modelling the system till Pluto, the time duration was taken to be 250 years (in seconds), which is just over the time period for 1 revolution of Pluto around the Sun (1 Pluto Year is 248 Earth Years). Δt was taken to a quarter of a day (in seconds), to provide a small enough time step for high accuracy without making the simulation too taxing on the computer and too time consuming. The time step in this case was increased compared to the previous system because the total run time of simulation was much higher than before, hence a suitable time tep was chosen

Simulation

The Simulation for the motion of the outer planets was run using both the AB2 numerical method as well as the RK4 numerical method, and the following plots were produced:

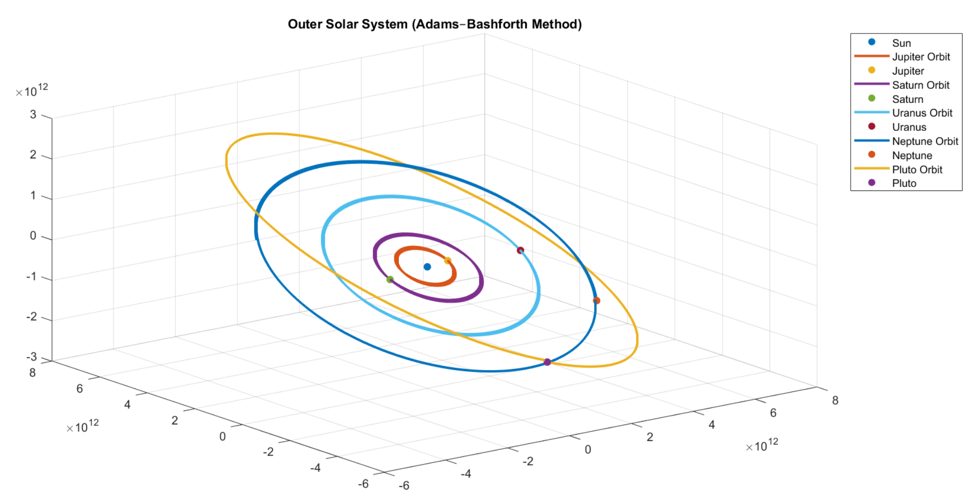

Figure 12: Inner Orbits Using AB2

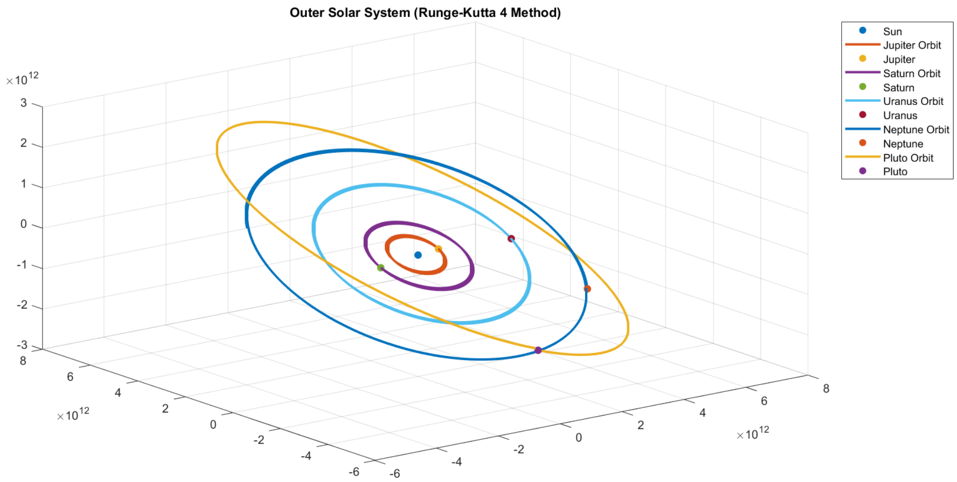

Figure 13: Inner Orbits Using RK4

Using the eye-ball test, we can see that the both the methods were able to model the orbits of the 5 planets with high accuracy. The Orbit for Jupiter looks a little off, which I believe is due to the high run time. The Accuracy could maybe have been improved using a smaller time step, but then it would take too long to run, and hence I chose to stick with the current time step.

Position and Velocity Plots

At the time of simulation end, the calculated vs actual Positions and Velocities of the planets were tabulated for comparison

Planetary_Bodies	Calculated_Position_AB2			JulianDate_Position_AB2		
'Jupiter'	6.5412e+11	3.8406e+11	1.4899e+11	5.9301e+11	4.1631e+11	1.6402e+11
'Saturn'	1.4115e+12	-1.253e+11	-1.1186e+11	1.4287e+12	-1.1391e+11	-1.0912e+11
'Uranus'	2.9578e+12	-3.5994e+11	-1.9971e+11	2.9775e+12	-3.8175e+11	-2.0919e+11
'Neptune'	-4.1105e+12	1.7887e+12	8.3625e+11	-4.1194e+12	1.6678e+12	7.8524e+11
'Pluto'	6.9945e+11	-4.4194e+12	-1.5838e+12	7.6913e+11	-4.4924e+12	-1.6339e+12

Figure 14: Outer OrbitsPosition Table AB2

Planetary_Bodies	Calculated_Position_RK4			JulianDate_Position_RK4		
'Jupiter'	6.5413e+11	3.8405e+11	1.4899e+11	5.9301e+11	4.1631e+11	1.6402e+11
'Saturn'	1.4115e+12	-1.253e+11	-1.1186e+11	1.4287e+12	-1.1391e+11	-1.0912e+11
'Uranus'	2.9578e+12	-3.5994e+11	-1.9971e+11	2.9775e+12	-3.8175e+11	-2.0919e+11
'Neptune'	-4.1105e+12	1.7887e+12	8.3625e+11	-4.1194e+12	1.6678e+12	7.8524e+11
'Pluto'	6.9945e+11	-4.4194e+12	-1.5838e+12	7.6913e+11	-4.4924e+12	-1.6339e+12

Figure 15: Position Table RK4

Planetary_Bodies	Calculated_Final_V_AB2	JulianDate_Final_V_AB2	CalculatedMeanVelocity_AB2	KnownMeanVelocity_AB2
'Jupiter'	13702	13672	13041	13060
'Saturn'	9554.8	9599.1	9617.2	9680
'Uranus'	6495.1	6496.9	6801.2	6800
'Neptune'	5405.3	5407.9	5444	5430
'Pluto'	5701.1	5684.5	4675.2	4670

Figure 16: Velocity Table AB2

Planetary_Bodies	Calculated_Final_V_RK4	JulianDate_Final_V_RK4	CalculatedMeanVelocity_RK4	KnownMeanVelocity_RK4
'Jupiter'	13702	13672	13041	13060
'Saturn'	9554.8	9599.1	9617.2	9680
'Uranus'	6495.1	6496.9	6801.2	6800
'Neptune'	5405.3	5407.9	5444	5430
'Pluto'	5701.1	5684.5	4675.2	4670

Figure 17: Velocity Table RK4

As it can be seen, the Known and actual values are very close to the calculated values, which again shows that the methods was implemented correctly

Error

To test the accuracy of methods, error between the estimation and the actual values was again plotted for multiple Δt's. The real values of all positions and velocities were obtained using `MATLAB` function `planetEphemeris` for the date of 31st December, 2060, since that is exactly 250 year in the future from January 1st, 2011, and then the overall error was plotted again Δt.

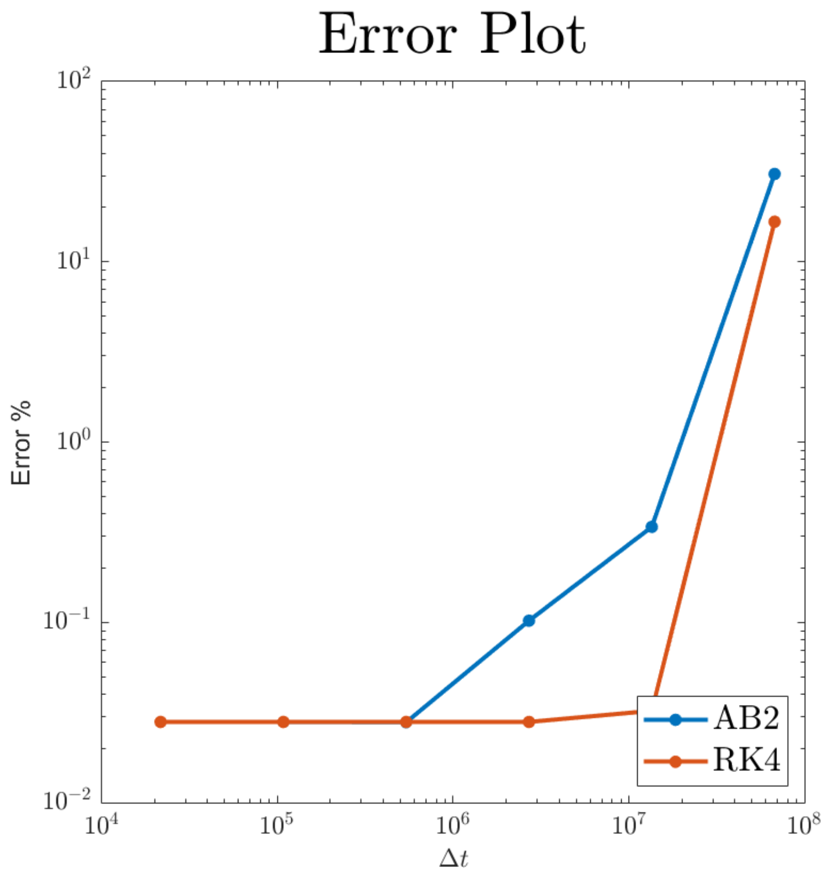

Figure 18: Error Plot for Inner Orbits

As it can be seen above, the overall error from both the methods is just below 3%, which I believe is an acceptable margin of error. After running the error plot though, it can be seen that the value of Δt chosen again could have been higher than what was chosen, since the error stabilizes at Δt's much bigger than the chosen one, but I had wanted to get the results as accurate as possible, hence I had chosen such a small Δt.

4 Conclusion

In Conclusion, we can say that we were successful in implementing both the chosen methods to get both the models of Solar System, though from the simulations and above plots, it can be seen that RK4 was a much better implementation than AB2, since it RK4 converges quicker and would have given good estimation with higher values of time step as compared to AB2.

These calculations can further be improved upon if we use a higher order Runge-Kutta method, though it is not really necessary, since 4th Order Runge-Kutta provides a good estimation already. Another way to improve the analysis is by using a more accurate numerical method, such as Bulirsch Stoer method, which is considered one of the best numerical methods. These methods can further be extended into systems with 10s and 100s of bodies, which would improve the estimation of the dynamics of the bodies, but much better computational power would be required to make such calculations.

Like mentioned above, with right computational power and method, this problem can be extended to much more complex problems, such as modelling the Solar System with the asteroid belt present between Mars and Jupiter, or mapping the motion of all spacecrafts/satellites orbiting the Earth. There is a huge number of applications for this problem.

5 Appendix : Code

```
%% N-Body Problem where N = 6
clear all, close all, clc

% Nomenclature
% 1= Sun
% 2 = Mercury
% 3 = Venus
% 4 = Earth
% 5 = Mars
% 6 = Moon

% Orbital Velocities of Planets in km/s (Data from NASA Fact Sheet)
% Mercury
Mercury_Min = 38.86; Mercury_Max = 58.98; Mercury_Mean = 47.36;
% Venus
Venus_Min = 34.79; Venus_Max = 35.26; Venus_Mean = 35.02;
% Earth
Earth_Min = 29.29; Earth_Max = 30.29; Earth_Mean = 29.78;
% Mars
Mars_Min = 21.97; Mars_Max = 26.50; Mars_Mean = 24.07;
% Moon
Moon_Min = 0.970; Moon_Max = 1.082; Moon_Mean = 1.022;

% Gathering N-Body Information for the date of Jan 1, 2011
Julian = juliandate(2011,1,1);

% Position and Velocity Vectors
[r1, v1] = planetEphemeris(Julian,'Sun', 'Sun', '432t', 'km');
[r2, v2] = planetEphemeris(Julian,'Sun','Mercury','432t','km');
[r3, v3] = planetEphemeris(Julian,'Sun','Venus','432t','km');
[r4, v4] = planetEphemeris(Julian,'Sun','Earth','432t','km');
[r5, v5] = planetEphemeris(Julian,'Sun','Mars','432t','km');
[r6, v6] = planetEphemeris(Julian,'Sun','Moon','432t','km');

```

```
% Defining u0 (in meters)
u0 = 10^3 * [r1'; v1'; r2'; v2'; r3'; v3'; r4'; v4'; r5'; v5'; r6'; v6'];

% Masses (in kg)
m1 = 1.9891 * 10^30;
m2 = 3.285 * 10^23;
m3 = 4.867 * 10^24;
m4 = 5.97 * 10^24;
m5 = 7.35 * 10^22;
m6 = 7.34767 * 10^22;

% Defining the Parameters
G = 6.67*10^(-11); % Gravitational Constant
t0 = 0; % Inital Start Time
T = 60*60*24*687*1; % Final Time (1 Mars Revolution in Seconds)
dt = 60*60*24*0.002; % Time step (0.002 Days in Seconds)

% State-Space Form Vectors (For Reference)
%r1 = u(1:3); v1 = u(4:6);
%r2 = u(7:9); v2 = u(10:12);
%r3 = u(13:15); v3 = u(16:18);
%r4 = u(19:21); v4 = u(22:24);
%r5 = u(25:27); v5 = u(28;30);
%r6 = u(31:33); v6 = u(34:36);

% State-Space Model
f = @(u) [ u(4:6); (G*m2*(u(7:9)-u(1:3)))./(norm((u(7:9)-u(1:3))).^3) + ...
    (G*m3*(u(13:15)-u(1:3)))./(norm((u(13:15)-u(1:3))).^3) + ...
    (G*m4*(u(19:21)-u(1:3)))./(norm((u(19:21)-u(1:3))).^3) + ...
    (G*m5*(u(25:27)-u(1:3)))./(norm((u(25:27)-u(1:3))).^3) + ...
    (G*m6*(u(31:33)-u(1:3)))./(norm((u(31:33)-u(1:3))).^3); ...
        u(10:12); (G*m1*(u(1:3)- u(7:9)))./(norm((u(1:3)-u(7:9))).^3) + ...
            (G*m3*(u(13:15)-u(7:9)))./(norm((u(13:15)-u(7:9))).^3) + ...
            (G*m4*(u(19:21)-u(7:9)))./(norm((u(19:21)-u(7:9))).^3) + ...
            (G*m5*(u(25:27)-u(7:9)))./(norm((u(25:27)-u(7:9))).^3) + ...
            (G*m6*(u(31:33)-u(7:9)))./(norm((u(31:33)-u(7:9))).^3); ...
        u(16:18); (G*m1*(u(1:3)- u(13:15)))./(norm((u(1:3)-u(13:15))).^3) + ...
            (G*m2*(u(7:9)-u(13:15)))./(norm((u(7:9)-u(13:15))).^3) + ...
            (G*m4*(u(19:21)-u(13:15)))./(norm((u(19:21)-u(13:15))).^3) + ...
            (G*m5*(u(25:27)-u(13:15)))./(norm((u(25:27)-u(13:15))).^3) + ...
            (G*m6*(u(31:33)-u(13:15)))./(norm((u(31:33)-u(13:15))).^3); ...
        u(22:24); (G*m1*(u(1:3)- u(19:21)))./(norm((u(1:3)-u(19:21))).^3) + ...
            (G*m2*(u(7:9)-u(19:21)))./(norm((u(7:9)-u(19:21))).^3) + ...
            (G*m3*(u(13:15)-u(19:21)))./(norm((u(13:15)-u(19:21))).^3) + ...
            (G*m5*(u(25:27)-u(19:21)))./(norm((u(25:27)-u(19:21))).^3) + ...
            (G*m6*(u(31:33)-u(19:21)))./(norm((u(31:33)-u(19:21))).^3); ...
```

```
        u(28:30); (G*m1*(u(1:3)- u(25:27)))./(norm((u(1:3)-u(25:27))).^3) + ...
            (G*m2*(u(7:9)-u(25:27)))./(norm((u(7:9)-u(25:27))).^3) + ...
            (G*m3*(u(13:15)-u(25:27)))./(norm((u(13:15)-u(25:27))).^3) + ...
            (G*m4*(u(19:21)-u(25:27)))./(norm((u(19:21)-u(25:27))).^3) + ...
            (G*m6*(u(31:33)-u(25:27)))./(norm((u(31:33)-u(25:27))).^3); ...
        u(34:36); (G*m1*(u(1:3)- u(31:33)))./(norm((u(1:3)-u(31:33))).^3) + ...
            (G*m2*(u(7:9)-u(31:33)))./(norm((u(7:9)-u(31:33))).^3) + ...
            (G*m3*(u(13:15)-u(31:33)))./(norm((u(13:15)-u(31:33))).^3) + ...
            (G*m4*(u(19:21)-u(31:33)))./(norm((u(19:21)-u(31:33))).^3) + ...
            (G*m5*(u(25:27)-u(31:33)))./(norm((u(25:27)-u(31:33))).^3)] ;

%%%%%% AB2 Numerical Analysis %%%%%%%

tic

tk = 0; %starting time

%initialize iterates for AB2
u_AB2_k = u0;
u_AB2_km1 = u0;

%initialize vector that stores approximate soln at various times
u_AB2_approx = zeros( 36,T/dt );

%Initialize vector that stores the magnitude of velocities at T
u_AB2_vel = zeros(6,T/dt);

%advance to final time T
for i = 1 : T/dt

    if i < 2

        %advance with Heun's for 1st time step
        u_AB2_approx(:,i) = (u_AB2_k + 1/2*dt*(f(u_AB2_k) + ...
            f(u_AB2_k+dt*f(u_AB2_k))));

    else

        u_AB2_approx(:,i) = u_AB2_k + 1/2*dt*(-1*f(u_AB2_km1) + 3*f(u_AB2_k));

    end

    %update iterates
    tk = tk + dt;
    u_AB2_km1 = u_AB2_k;
    u_AB2_k = u_AB2_approx(:,i);

```

```
    %Calculating and storing the velocities of each body at T
    u_AB2_vel(1,i) = norm(norm(u_AB2_approx(4:6,i)));
    u_AB2_vel(2,i) = norm(norm(u_AB2_approx(10:12,i)));
    u_AB2_vel(3,i) = norm(norm(u_AB2_approx(16:18,i)));
    u_AB2_vel(4,i) = norm(norm(u_AB2_approx(22:24,i)));
    u_AB2_vel(5,i) = norm(norm(u_AB2_approx(28:30,i)));
    u_AB2_vel(6,i) = norm(norm(u_AB2_approx(34:36,i)));

end

t_AB2 = toc

% Gathering N-Body Information for 687 days in future from Jan 1, 2011 (T)
Julian2 = juliandate(2012,11,19);

% Position and Velocity Vectors at future position
[r1, v1] = planetEphemeris(Julian2,'Sun', 'Sun', '432t', 'km');
[r2, v2] = planetEphemeris(Julian2,'Sun','Mercury','432t','km');
[r3, v3] = planetEphemeris(Julian2,'Sun','Venus','432t','km');
[r4, v4] = planetEphemeris(Julian2,'Sun','Earth','432t','km');
[r5, v5] = planetEphemeris(Julian2,'Sun','Mars','432t','km');
[r6, v6] = planetEphemeris(Julian2,'Sun','Moon','432t','km');

% Calculating the Average, Minimum and Maximum Velocity of Each Planet
AB2_vel_avg = mean(u_AB2_vel,2);
AB2_vel_max = max(u_AB2_vel,[],2);
AB2_vel_min = min(u_AB2_vel,[],2);

% Assigning Variable Names
Planetary_Bodies = {'Mercury';'Venus';'Earth';'Mars';'Moon'};
Calculated_Position_AB2 = [u_AB2_approx(7,T/dt), u_AB2_approx(8,T/dt), ...
    u_AB2_approx(9,T/dt); u_AB2_approx(13,T/dt), u_AB2_approx(14,T/dt), ...
    u_AB2_approx(15,T/dt); u_AB2_approx(19,T/dt), u_AB2_approx(20,T/dt), ...
    u_AB2_approx(21,T/dt); u_AB2_approx(25,T/dt), u_AB2_approx(26,T/dt), ...
    u_AB2_approx(27,T/dt); u_AB2_approx(31,T/dt), u_AB2_approx(32,T/dt), ...
    u_AB2_approx(33,T/dt)];
JulianDate_Position_AB2 = 10^3 * [r2;r3;r4;r5;r6];
% Calculated_Position = [norm([u_AB2_approx(7,T/dt), u_AB2_approx(8,T/dt), ...
    u_AB2_approx(9,T/dt)]); norm([u_AB2_approx(13,T/dt), ...
    u_AB2_approx(14,T/dt), u_AB2_approx(15,T/dt)]); ...
    norm([u_AB2_approx(19,T/dt), u_AB2_approx(20,T/dt), ...
    u_AB2_approx(21,T/dt)]); norm([u_AB2_approx(25,T/dt), ...
    u_AB2_approx(26,T/dt), u_AB2_approx(27,T/dt)]); ...
    norm([u_AB2_approx(31,T/dt), u_AB2_approx(32,T/dt), u_AB2_approx(33,T/dt)])];
% JulianDate_Position = 10^3 * [norm(r2);norm(r3);norm(r4);norm(r5);norm(r6)];
Calculated_Final_V_AB2 = [norm([u_AB2_approx(10,T/dt), ...
    u_AB2_approx(11,T/dt), u_AB2_approx(12,T/dt)]); ...
```

```
    norm([u_AB2_approx(16,T/dt), u_AB2_approx(17,T/dt), ...
    u_AB2_approx(18,T/dt)]); norm([u_AB2_approx(22,T/dt), ...
    u_AB2_approx(23,T/dt), u_AB2_approx(24,T/dt)]); ...
    norm([u_AB2_approx(28,T/dt), u_AB2_approx(29,T/dt), ...
    u_AB2_approx(30,T/dt)]); norm([u_AB2_approx(34,T/dt), ...
    u_AB2_approx(35,T/dt), u_AB2_approx(36,T/dt)])];
JulianDate_Final_V_AB2 = 10^3 * [norm(v2);norm(v3);norm(v4);norm(v5);norm(v6)];
CalculatedMeanVelocity_AB2 = [AB2_vel_avg(2); AB2_vel_avg(3); ...
    AB2_vel_avg(4); AB2_vel_avg(5); AB2_vel_avg(6)];
KnownMeanVelocity_AB2 = 10^3 * [Mercury_Mean; Venus_Mean; Earth_Mean; ...
    Mars_Mean; Moon_Mean];

% Defining Tables
Table1 = table(Planetary_Bodies, Calculated_Position_AB2, ...
    JulianDate_Position_AB2);
Table2 = table(Planetary_Bodies, Calculated_Final_V_AB2, ...
    JulianDate_Final_V_AB2, CalculatedMeanVelocity_AB2, KnownMeanVelocity_AB2);

% Get the table in string form.
TString = evalc('disp(Table1)');
% Use TeX Markup for bold formatting and underscores.
TString = strrep(TString,'<strong>','\bf');
TString = strrep(TString,'</strong>','\rm');
TString = strrep(TString,'_','\_');
% Get a fixed-width font.
FixedWidth = get(0,'FixedWidthFontName');
% Output the table using the annotation command.
figure(100)
annotation(gcf,'Textbox','String',TString,'Interpreter','Tex','FontName',...
     FixedWidth,'Units','Normalized','Position',[0 0 1 1]);

% Get the table in string form.
TString = evalc('disp(Table2)');
% Use TeX Markup for bold formatting and underscores.
TString = strrep(TString,'<strong>','\bf');
TString = strrep(TString,'</strong>','\rm');
TString = strrep(TString,'_','\_');
% Get a fixed-width font.
FixedWidth = get(0,'FixedWidthFontName');
% Output the table using the annotation command.
figure(200)
annotation(gcf,'Textbox','String',TString,'Interpreter','Tex','FontName',...
     FixedWidth,'Units','Normalized','Position',[0 0 1 1]);

% Plotting the Orbits Estimated
figure(1)
scatter4(0,0,0,40,'filled')
```

```
grid on
hold on
plot3(u_AB2_approx(7,:), u_AB2_approx(8,:), u_AB2_approx(9,:),'LineWidth', 2)
scatter4(u_AB2_approx(7,1), u_AB2_approx(8,1), u_AB2_approx(9,1),40,'filled')
plot3(u_AB2_approx(13,:), u_AB2_approx(14,:), ...
    u_AB2_approx(15,:),'LineWidth', 2)
scatter4(u_AB2_approx(13,1), u_AB2_approx(14,1), u_AB2_approx(15,1),40,'filled')
plot3(u_AB2_approx(19,:), u_AB2_approx(20,:), ...
    u_AB2_approx(21,:),'LineWidth', 2)
scatter4(u_AB2_approx(19,1), u_AB2_approx(20,1), u_AB2_approx(21,1),40,'filled')
plot3(u_AB2_approx(25,:), u_AB2_approx(26,:), ...
    u_AB2_approx(27,:),'LineWidth', 2)
scatter4(u_AB2_approx(25,1), u_AB2_approx(26,1), u_AB2_approx(27,1),40,'filled')
plot3(u_AB2_approx(31,:), u_AB2_approx(32,:), ...
    u_AB2_approx(33,:),'LineWidth', 0.75)
scatter4(u_AB2_approx(31,1), u_AB2_approx(32,1), u_AB2_approx(33,1),10,'filled')
title('Inner Solar System (Adams Bashforth Method)')
legend('Sun','Mercury Orbit','Mercury','Venus Orbit','Venus','Earth ...
    Orbit','Earth','Mars Orbit','Mars','Moon Orbit','Moon')


%%%%%% RK4 Numerical Analysis %%%%%%%

tk = 0; %starting time

tic

%initialize iterates for Huens
u_rk_k = u0;

%initialize vector that stores approximate soln at various times
u_rk_approx = zeros( 36,T/dt );

%Initialize vector that stores the magnitude of velocities at T
u_rk_vel = zeros(6,T/dt);

%advance to final time T
for i = 1 : T/dt

    %advance with RK4 for 1st time step
    y1 = f(u_rk_k);
    y2 = f(u_rk_k + 1/2*dt*y1);
    y3 = f(u_rk_k + (1/2)*dt*y2);
    y4 = f(u_rk_k + dt*y3);
    u_rk_approx(:,i) = u_rk_k + 1/6*dt*(y1 + 2*y2 + 2*y3 + y4);

    %update iterates
```

```
    tk = tk + dt;
    u_rk_k = u_rk_approx(:,i);

    %Calculating and storing the velocities of each body at T
    u_rk_vel(1,i) = norm(norm(u_rk_approx(4:6,i)));
    u_rk_vel(2,i) = norm(norm(u_rk_approx(10:12,i)));
    u_rk_vel(3,i) = norm(norm(u_rk_approx(16:18,i)));
    u_rk_vel(4,i) = norm(norm(u_rk_approx(22:24,i)));
    u_rk_vel(5,i) = norm(norm(u_rk_approx(28:30,i)));
    u_rk_vel(6,i) = norm(norm(u_rk_approx(34:36,i)));

end

t_r = toc

% Gathering N-Body Information for 687 days in future from Jan 1, 2011 (T)
Julian2 = juliandate(2012,11,19);

% Position and Velocity Vectors at future position
[r1, v1] = planetEphemeris(Julian2,'Sun', 'Sun', '432t', 'km');
[r2, v2] = planetEphemeris(Julian2,'Sun','Mercury','432t','km');
[r3, v3] = planetEphemeris(Julian2,'Sun','Venus','432t','km');
[r4, v4] = planetEphemeris(Julian2,'Sun','Earth','432t','km');
[r5, v5] = planetEphemeris(Julian2,'Sun','Mars','432t','km');
[r6, v6] = planetEphemeris(Julian2,'Sun','Moon','432t','km');

% Calculating the Average Velocity of Each Planet
rk_vel_avg = mean(u_rk_vel,2);
rk_vel_max = max(u_rk_vel,[],2);
rk_vel_min = min(u_rk_vel,[],2);

% Assigning Variable Names
Planetary_Bodies = {'Mercury';'Venus';'Earth';'Mars';'Moon'};
Calculated_Position_RK4 = [u_rk_approx(7,T/dt), u_rk_approx(8,T/dt), ...
    u_rk_approx(9,T/dt); u_rk_approx(13,T/dt), u_rk_approx(14,T/dt), ...
    u_rk_approx(15,T/dt); u_rk_approx(19,T/dt), u_rk_approx(20,T/dt), ...
    u_rk_approx(21,T/dt); u_rk_approx(25,T/dt), u_rk_approx(26,T/dt), ...
    u_rk_approx(27,T/dt); u_rk_approx(31,T/dt), u_rk_approx(32,T/dt), ...
    u_rk_approx(33,T/dt)];
JulianDate_Position_RK4 = 10^3 * [r2;r3;r4;r5;r6];
% Calculated_Position = [norm([u_rk_approx(7,T/dt), u_rk_approx(8,T/dt), ...
    u_rk_approx(9,T/dt)]); norm([u_rk_approx(13,T/dt), u_rk_approx(14,T/dt), ...
    u_rk_approx(15,T/dt)]); norm([u_rk_approx(19,T/dt), u_rk_approx(20,T/dt), ...
    u_rk_approx(21,T/dt)]); norm([u_rk_approx(25,T/dt), u_rk_approx(26,T/dt), ...
    u_rk_approx(27,T/dt)]); norm([u_rk_approx(31,T/dt), u_rk_approx(32,T/dt), ...
    u_rk_approx(33,T/dt)])];
% JulianDate_Position = 10^3 * [norm(r2);norm(r3);norm(r4);norm(r5);norm(r6)];
```

```
Calculated_Final_V_RK4 = [norm([u_rk_approx(10,T/dt), u_rk_approx(11,T/dt), ...
    u_rk_approx(12,T/dt)]); norm([u_rk_approx(16,T/dt), u_rk_approx(17,T/dt), ...
    u_rk_approx(18,T/dt)]); norm([u_rk_approx(22,T/dt), u_rk_approx(23,T/dt), ...
    u_rk_approx(24,T/dt)]); norm([u_rk_approx(28,T/dt), u_rk_approx(29,T/dt), ...
    u_rk_approx(30,T/dt)]); norm([u_rk_approx(34,T/dt), u_rk_approx(35,T/dt), ...
    u_rk_approx(36,T/dt)])];
JulianDate_Final_V_RK4 = 10^3 * [norm(v2);norm(v3);norm(v4);norm(v5);norm(v6)];
CalculatedMeanVelocity_RK4 = [rk_vel_avg(2); rk_vel_avg(3); rk_vel_avg(4); ...
    rk_vel_avg(5); rk_vel_avg(6)];
KnownMeanVelocity_RK4 = 10^3 * [Mercury_Mean; Venus_Mean; Earth_Mean; ...
    Mars_Mean; Moon_Mean];

% Defining Tables
Table1 = table(Planetary_Bodies, Calculated_Position_RK4, ...
    JulianDate_Position_RK4);
Table2 = table(Planetary_Bodies, Calculated_Final_V_RK4, ...
    JulianDate_Final_V_RK4, CalculatedMeanVelocity_RK4, KnownMeanVelocity_RK4);

% Get the table in string form.
TString = evalc('disp(Table1)');
% Use TeX Markup for bold formatting and underscores.
TString = strrep(TString,'<strong>','\bf');
TString = strrep(TString,'</strong>','\rm');
TString = strrep(TString,'_','\_');
% Get a fixed-width font.
FixedWidth = get(0,'FixedWidthFontName');
% Output the table using the annotation command.
figure(300)
annotation(gcf,'Textbox','String',TString,'Interpreter','Tex','FontName',...
    FixedWidth,'Units','Normalized','Position',[0 0 1 1]);

% Get the table in string form.
TString = evalc('disp(Table2)');
% Use TeX Markup for bold formatting and underscores.
TString = strrep(TString,'<strong>','\bf');
TString = strrep(TString,'</strong>','\rm');
TString = strrep(TString,'_','\_');
% Get a fixed-width font.
FixedWidth = get(0,'FixedWidthFontName');
% Output the table using the annotation command.
figure(400)
annotation(gcf,'Textbox','String',TString,'Interpreter','Tex','FontName',...
    FixedWidth,'Units','Normalized','Position',[0 0 1 1]);

% Plotting the Orbits Estimated
figure(2)
scatter4(0,0,0,40,'filled')
```

```
grid on
hold on
plot3(u_rk_approx(7,:), u_rk_approx(8,:), u_rk_approx(9,:),'LineWidth', 2)
scatter4(u_rk_approx(7,1), u_rk_approx(8,1), u_rk_approx(9,1),40,'filled')
plot3(u_rk_approx(13,:), u_rk_approx(14,:), u_rk_approx(15,:),'LineWidth', 2)
scatter4(u_rk_approx(13,1), u_rk_approx(14,1), u_rk_approx(15,1),40,'filled')
plot3(u_rk_approx(19,:), u_rk_approx(20,:), u_rk_approx(21,:),'LineWidth', 2)
scatter4(u_rk_approx(19,1), u_rk_approx(20,1), u_rk_approx(21,1),40,'filled')
plot3(u_rk_approx(25,:), u_rk_approx(26,:), u_rk_approx(27,:),'LineWidth', 2)
scatter4(u_rk_approx(25,1), u_rk_approx(26,1), u_rk_approx(27,1),40,'filled')
plot3(u_rk_approx(31,:), u_rk_approx(32,:), u_rk_approx(33,:),'LineWidth', 0.75)
scatter4(u_rk_approx(31,1), u_rk_approx(32,1), u_rk_approx(33,1),10,'filled')
title('Inner Solar System (Runge-Kutta 4 Method)')
legend('Sun','Mercury Orbit','Mercury','Venus Orbit','Venus','Earth ...
    Orbit','Earth','Mars Orbit','Mars','Moon Orbit','Moon')


%% N-Body Problem where N = 6 where Sun lost 25% of its mass
clear all, close all, clc

% Nomenclature
% 1 = Sun
% 2 = Jupiter
% 3 = Saturn
% 4 = Uranus
% 5 = Neptune
% 6 = Pluto

% Orbital Velocities of Planets in km/s (Data from NASA Fact Sheet)
% Jupiter
Jupiter_Min = 12.44; Jupiter_Max = 13.72; Jupiter_Mean = 13.06;
% Saturn
Saturn_Min = 9.09; Saturn_Max = 10.18; Saturn_Mean = 9.68;
% Uranus
Uranus_Min = 6.49; Uranus_Max = 7.11; Uranus_Mean = 6.80;
% Neptune
Neptune_Min = 5.37; Neptune_Max = 5.50; Neptune_Mean = 5.43;
% Pluto
Pluto_Min = 3.71; Pluto_Max = 6.10; Pluto_Mean = 4.67;

% Gathering N-Body Information for the date of Jan 1, 2011
Julian = juliandate(2011,1,1);

% Position and Velocity Vectors
[r1, v1] = planetEphemeris(Julian,'Sun', 'Sun', '432t', 'km');
[r2, v2] = planetEphemeris(Julian,'Sun','Jupiter','432t','km');
[r3, v3] = planetEphemeris(Julian,'Sun','Saturn','432t','km');
```

```
[r4, v4] = planetEphemeris(Julian,'Sun','Uranus','432t','km');
[r5, v5] = planetEphemeris(Julian,'Sun','Neptune','432t','km');
[r6, v6] = planetEphemeris(Julian,'Sun','Pluto','432t','km');


% Defining u0 (in meters)
u0 = 10^3 * [r1'; v1'; r2'; v2'; r3'; v3'; r4'; v4'; r5'; v5'; r6'; v6'];

% Masses (in kg)
m1 = 1.9891 * 10^30;
m2 = 1.898 * 10^27;
m3 = 5.683 * 10^26;
m4 = 8.681 * 10^25;
m5 = 1.024 * 10^26;
m6 = 1.309 * 10^22;


% Defining the Parameters
G = 6.67*10^(-11); % Gravitational Constant
t0 = 0; % Inital Start Time
T = 60*60*24*365*250; % Final Time (1 Pluto Years)
dt = 60*60*24*0.25; % Time step (0.01 Days in Seconds)

% State-Space Form Vectors (For Reference)
%r1 = u(1:3); v1 = u(4:6);
%r2 = u(7:9); v2 = u(10:12);
%r3 = u(13:15); v3 = u(16:18);
%r4 = u(19:21); v4 = u(22:24);
%r5 = u(25:27); v5 = u(28;30);
%r6 = u(31:33); v6 = u(34:36);

% State-Space Model
f = @(u) [ u(4:6); (G*m2*(u(7:9)-u(1:3)))./(norm((u(7:9)-u(1:3))).^3) + ...
    (G*m3*(u(13:15)-u(1:3)))./(norm((u(13:15)-u(1:3))).^3) + ...
    (G*m4*(u(19:21)-u(1:3)))./(norm((u(19:21)-u(1:3))).^3) + ...
    (G*m5*(u(25:27)-u(1:3)))./(norm((u(25:27)-u(1:3))).^3) + ...
    (G*m6*(u(31:33)-u(1:3)))./(norm((u(31:33)-u(1:3))).^3); ...
        u(10:12); (G*m1*(u(1:3)- u(7:9)))./(norm((u(1:3)-u(7:9))).^3) + ...
            (G*m3*(u(13:15)-u(7:9)))./(norm((u(13:15)-u(7:9))).^3) + ...
            (G*m4*(u(19:21)-u(7:9)))./(norm((u(19:21)-u(7:9))).^3) + ...
            (G*m5*(u(25:27)-u(7:9)))./(norm((u(25:27)-u(7:9))).^3) + ...
            (G*m6*(u(31:33)-u(7:9)))./(norm((u(31:33)-u(7:9))).^3); ...
        u(16:18); (G*m1*(u(1:3)- u(13:15)))./(norm((u(1:3)-u(13:15))).^3) + ...
            (G*m2*(u(7:9)-u(13:15)))./(norm((u(7:9)-u(13:15))).^3) + ...
            (G*m4*(u(19:21)-u(13:15)))./(norm((u(19:21)-u(13:15))).^3) + ...
            (G*m5*(u(25:27)-u(13:15)))./(norm((u(25:27)-u(13:15))).^3) + ...
            (G*m6*(u(31:33)-u(13:15)))./(norm((u(31:33)-u(13:15))).^3); ...
```

```
        u(22:24); (G*m1*(u(1:3)- u(19:21)))./(norm((u(1:3)-u(19:21))).^3) + ...
            (G*m2*(u(7:9)-u(19:21)))./(norm((u(7:9)-u(19:21))).^3) + ...
            (G*m3*(u(13:15)-u(19:21)))./(norm((u(13:15)-u(19:21))).^3) + ...
            (G*m5*(u(25:27)-u(19:21)))./(norm((u(25:27)-u(19:21))).^3) + ...
            (G*m6*(u(31:33)-u(19:21)))./(norm((u(31:33)-u(19:21))).^3); ...
        u(28:30); (G*m1*(u(1:3)- u(25:27)))./(norm((u(1:3)-u(25:27))).^3) + ...
            (G*m2*(u(7:9)-u(25:27)))./(norm((u(7:9)-u(25:27))).^3) + ...
            (G*m3*(u(13:15)-u(25:27)))./(norm((u(13:15)-u(25:27))).^3) + ...
            (G*m4*(u(19:21)-u(25:27)))./(norm((u(19:21)-u(25:27))).^3) + ...
            (G*m6*(u(31:33)-u(25:27)))./(norm((u(31:33)-u(25:27))).^3); ...
        u(34:36); (G*m1*(u(1:3)- u(31:33)))./(norm((u(1:3)-u(31:33))).^3) + ...
            (G*m2*(u(7:9)-u(31:33)))./(norm((u(7:9)-u(31:33))).^3) + ...
            (G*m3*(u(13:15)-u(31:33)))./(norm((u(13:15)-u(31:33))).^3) + ...
            (G*m4*(u(19:21)-u(31:33)))./(norm((u(19:21)-u(31:33))).^3) + ...
            (G*m5*(u(25:27)-u(31:33)))./(norm((u(25:27)-u(31:33))).^3)] ;

%%%%%% AB2 Numerical Analysis %%%%%%%

tic

tk = 0; %starting time

%initialize iterates for AB2
u_AB2_k = u0;
u_AB2_km1 = u0;

%initialize vector that stores approximate soln at various times
u_AB2_approx = zeros( 36,T/dt );

%Initialize vector that stores the magnitude of velocities at T
u_AB2_vel = zeros(6,T/dt);

%advance to final time T
for i = 1 : T/dt

    if i < 2

        %advance with Heun's for 1st time step
        u_AB2_approx(:,i) = (u_AB2_k + 1/2*dt*(f(u_AB2_k) + ...
            f(u_AB2_k+dt*f(u_AB2_k))));

    else

        u_AB2_approx(:,i) = u_AB2_k + 1/2*dt*(-1*f(u_AB2_km1) + 3*f(u_AB2_k));

    end

```

```
    %update iterates
    tk = tk + dt;
    u_AB2_km1 = u_AB2_k;
    u_AB2_k = u_AB2_approx(:,i);

    %Calculating and storing the velocities of each body at T
    u_AB2_vel(1,i) = norm(norm(u_AB2_approx(4:6,i)));
    u_AB2_vel(2,i) = norm(norm(u_AB2_approx(10:12,i)));
    u_AB2_vel(3,i) = norm(norm(u_AB2_approx(16:18,i)));
    u_AB2_vel(4,i) = norm(norm(u_AB2_approx(22:24,i)));
    u_AB2_vel(5,i) = norm(norm(u_AB2_approx(28:30,i)));
    u_AB2_vel(6,i) = norm(norm(u_AB2_approx(34:36,i)));

end

t_AB2 = toc

% Gathering N-Body Information for 249 years in future from Jan 1, 2011 (T)
Julian2 = juliandate(2260,12,31);

% Position and Velocity Vectors at future position
[r1, v1] = planetEphemeris(Julian2,'Sun', 'Sun', '432t', 'km');
[r2, v2] = planetEphemeris(Julian2,'Sun','Jupiter','432t','km');
[r3, v3] = planetEphemeris(Julian2,'Sun','Saturn','432t','km');
[r4, v4] = planetEphemeris(Julian2,'Sun','Uranus','432t','km');
[r5, v5] = planetEphemeris(Julian2,'Sun','Neptune','432t','km');
[r6, v6] = planetEphemeris(Julian2,'Sun','Pluto','432t','km');

% Calculating the Average, Minimum and Maximum Velocity of Each Planet
AB2_vel_avg = mean(u_AB2_vel,2);
AB2_vel_max = max(u_AB2_vel,[],2);
AB2_vel_min = min(u_AB2_vel,[],2);

% Assigning Variable Names
Planetary_Bodies = {'Jupiter';'Saturn';'Uranus';'Neptune';'Pluto'};
Calculated_Position_AB2 = [u_AB2_approx(7,T/dt), u_AB2_approx(8,T/dt), ...
    u_AB2_approx(9,T/dt); u_AB2_approx(13,T/dt), u_AB2_approx(14,T/dt), ...
    u_AB2_approx(15,T/dt); u_AB2_approx(19,T/dt), u_AB2_approx(20,T/dt), ...
    u_AB2_approx(21,T/dt); u_AB2_approx(25,T/dt), u_AB2_approx(26,T/dt), ...
    u_AB2_approx(27,T/dt); u_AB2_approx(31,T/dt), u_AB2_approx(32,T/dt), ...
    u_AB2_approx(33,T/dt)];
JulianDate_Position_AB2 = 10^3 * [r2;r3;r4;r5;r6];
% Calculated_Position = [norm([u_AB2_approx(7,T/dt), u_AB2_approx(8,T/dt), ...
    u_AB2_approx(9,T/dt)]); norm([u_AB2_approx(13,T/dt), ...
    u_AB2_approx(14,T/dt), u_AB2_approx(15,T/dt)]); ...
    norm([u_AB2_approx(19,T/dt), u_AB2_approx(20,T/dt), ...
    u_AB2_approx(21,T/dt)]); norm([u_AB2_approx(25,T/dt), ...
```

```
    u_AB2_approx(26,T/dt), u_AB2_approx(27,T/dt)]); ...
    norm([u_AB2_approx(31,T/dt), u_AB2_approx(32,T/dt), u_AB2_approx(33,T/dt)])];
% JulianDate_Position = 10^3 * [norm(r2);norm(r3);norm(r4);norm(r5);norm(r6)];
Calculated_Final_V_AB2 = [norm([u_AB2_approx(10,T/dt), ...
    u_AB2_approx(11,T/dt), u_AB2_approx(12,T/dt)]); ...
    norm([u_AB2_approx(16,T/dt), u_AB2_approx(17,T/dt), ...
    u_AB2_approx(18,T/dt)]); norm([u_AB2_approx(22,T/dt), ...
    u_AB2_approx(23,T/dt), u_AB2_approx(24,T/dt)]); ...
    norm([u_AB2_approx(28,T/dt), u_AB2_approx(29,T/dt), ...
    u_AB2_approx(30,T/dt)]); norm([u_AB2_approx(34,T/dt), ...
    u_AB2_approx(35,T/dt), u_AB2_approx(36,T/dt)])];
JulianDate_Final_V_AB2 = 10^3 * [norm(v2);norm(v3);norm(v4);norm(v5);norm(v6)];
CalculatedMeanVelocity_AB2 = [AB2_vel_avg(2); AB2_vel_avg(3); ...
    AB2_vel_avg(4); AB2_vel_avg(5); AB2_vel_avg(6)];
KnownMeanVelocity_AB2 = 10^3 * [Jupiter_Mean; Saturn_Mean; Uranus_Mean; ...
    Neptune_Mean; Pluto_Mean];

% Defining Tables
Table1 = table(Planetary_Bodies, Calculated_Position_AB2, ...
    JulianDate_Position_AB2);
Table2 = table(Planetary_Bodies, Calculated_Final_V_AB2, ...
    JulianDate_Final_V_AB2, CalculatedMeanVelocity_AB2, KnownMeanVelocity_AB2);

% Get the table in string form.
TString = evalc('disp(Table1)');
% Use TeX Markup for bold formatting and underscores.
TString = strrep(TString,'<strong>','\bf');
TString = strrep(TString,'</strong>','\rm');
TString = strrep(TString,'_','\_');
% Get a fixed-width font.
FixedWidth = get(0,'FixedWidthFontName');
% Output the table using the annotation command.
figure(100)
annotation(gcf,'Textbox','String',TString,'Interpreter','Tex','FontName',...
     FixedWidth,'Units','Normalized','Position',[0 0 1 1]);

% Get the table in string form.
TString = evalc('disp(Table2)');
% Use TeX Markup for bold formatting and underscores.
TString = strrep(TString,'<strong>','\bf');
TString = strrep(TString,'</strong>','\rm');
TString = strrep(TString,'_','\_');
% Get a fixed-width font.
FixedWidth = get(0,'FixedWidthFontName');
% Output the table using the annotation command.
figure(200)
annotation(gcf,'Textbox','String',TString,'Interpreter','Tex','FontName',...
```

```
    FixedWidth,'Units','Normalized','Position',[0 0 1 1]);

% Plotting the Orbits Estimated
figure(1)
scatter4(0,0,0,40,'filled')
grid on
hold on
plot3(u_AB2_approx(7,:), u_AB2_approx(8,:), u_AB2_approx(9,:),'LineWidth', 2)
scatter4(u_AB2_approx(7,1), u_AB2_approx(8,1), u_AB2_approx(9,1),40,'filled')
plot3(u_AB2_approx(13,:), u_AB2_approx(14,:), ...
    u_AB2_approx(15,:),'LineWidth', 2)
scatter4(u_AB2_approx(13,1), u_AB2_approx(14,1), u_AB2_approx(15,1),40,'filled')
plot3(u_AB2_approx(19,:), u_AB2_approx(20,:), ...
    u_AB2_approx(21,:),'LineWidth', 2)
scatter4(u_AB2_approx(19,1), u_AB2_approx(20,1), u_AB2_approx(21,1),40,'filled')
plot3(u_AB2_approx(25,:), u_AB2_approx(26,:), ...
    u_AB2_approx(27,:),'LineWidth', 2)
scatter4(u_AB2_approx(25,1), u_AB2_approx(26,1), u_AB2_approx(27,1),40,'filled')
plot3(u_AB2_approx(31,:), u_AB2_approx(32,:), ...
    u_AB2_approx(33,:),'LineWidth', 2)
scatter4(u_AB2_approx(31,1), u_AB2_approx(32,1), u_AB2_approx(33,1),40,'filled')
title('Outer Solar System (Adams Bashforth Method)')
legend('Sun','Jupiter Orbit','Jupiter','Saturn Orbit','Saturn','Uranus ...
    Orbit','Uranus','Neptune Orbit','Neptune','Pluto Orbit','Pluto')



%%%%%% RK4 Numerical Analysis %%%%%%%

tk = 0; %starting time

tic

%initialize iterates for Huens
u_rk_k = u0;

%initialize vector that stores approximate soln at various times
u_rk_approx = zeros( 36,T/dt );

%Initialize vector that stores the magnitude of velocities at T
u_rk_vel = zeros(6,T/dt);

%advance to final time T
for i = 1 : T/dt

    %advance with RK4 for 1st time step
    y1 = f(u_rk_k);
```

```
    y2 = f(u_rk_k + 1/2*dt*y1);
    y3 = f(u_rk_k + (1/2)*dt*y2);
    y4 = f(u_rk_k + dt*y3);
    u_rk_approx(:,i) = u_rk_k + 1/6*dt*(y1 + 2*y2 + 2*y3 + y4);

    %update iterates
    tk = tk + dt;
    u_rk_k = u_rk_approx(:,i);

    %Calculating and storing the velocities of each body at T
    u_rk_vel(1,i) = norm(norm(u_rk_approx(4:6,i)));
    u_rk_vel(2,i) = norm(norm(u_rk_approx(10:12,i)));
    u_rk_vel(3,i) = norm(norm(u_rk_approx(16:18,i)));
    u_rk_vel(4,i) = norm(norm(u_rk_approx(22:24,i)));
    u_rk_vel(5,i) = norm(norm(u_rk_approx(28:30,i)));
    u_rk_vel(6,i) = norm(norm(u_rk_approx(34:36,i)));

end

t_r = toc

% Gathering N-Body Information for 249 years in future from Jan 1, 2011 (T)
Julian2 = juliandate(2260,12,31);

% Position and Velocity Vectors at future position
[r1, v1] = planetEphemeris(Julian2,'Sun', 'Sun', '432t', 'km');
[r2, v2] = planetEphemeris(Julian2,'Sun','Jupiter','432t','km');
[r3, v3] = planetEphemeris(Julian2,'Sun','Saturn','432t','km');
[r4, v4] = planetEphemeris(Julian2,'Sun','Uranus','432t','km');
[r5, v5] = planetEphemeris(Julian2,'Sun','Neptune','432t','km');
[r6, v6] = planetEphemeris(Julian2,'Sun','Pluto','432t','km');

% Calculating the Average Velocity of Each Planet
rk_vel_avg = mean(u_rk_vel,2);
rk_vel_max = max(u_rk_vel,[],2);
rk_vel_min = min(u_rk_vel,[],2);

% Assigning Variable Names
Planetary_Bodies = {'Jupiter';'Saturn';'Uranus';'Neptune';'Pluto'};
Calculated_Position_RK4 = [u_rk_approx(7,T/dt), u_rk_approx(8,T/dt), ...
    u_rk_approx(9,T/dt); u_rk_approx(13,T/dt), u_rk_approx(14,T/dt), ...
    u_rk_approx(15,T/dt); u_rk_approx(19,T/dt), u_rk_approx(20,T/dt), ...
    u_rk_approx(21,T/dt); u_rk_approx(25,T/dt), u_rk_approx(26,T/dt), ...
    u_rk_approx(27,T/dt); u_rk_approx(31,T/dt), u_rk_approx(32,T/dt), ...
    u_rk_approx(33,T/dt)];
JulianDate_Position_RK4 = 10^3 * [r2;r3;r4;r5;r6];
```

```
% Calculated_Position = [norm([u_rk_approx(7,T/dt), u_rk_approx(8,T/dt), ...
    u_rk_approx(9,T/dt)]); norm([u_rk_approx(13,T/dt), u_rk_approx(14,T/dt), ...
    u_rk_approx(15,T/dt)]); norm([u_rk_approx(19,T/dt), u_rk_approx(20,T/dt), ...
    u_rk_approx(21,T/dt)]); norm([u_rk_approx(25,T/dt), u_rk_approx(26,T/dt), ...
    u_rk_approx(27,T/dt)]); norm([u_rk_approx(31,T/dt), u_rk_approx(32,T/dt), ...
    u_rk_approx(33,T/dt)])];
% JulianDate_Position = 10^3 * [norm(r2);norm(r3);norm(r4);norm(r5);norm(r6)];
Calculated_Final_V_RK4 = [norm([u_rk_approx(10,T/dt), u_rk_approx(11,T/dt), ...
    u_rk_approx(12,T/dt)]); norm([u_rk_approx(16,T/dt), u_rk_approx(17,T/dt), ...
    u_rk_approx(18,T/dt)]); norm([u_rk_approx(22,T/dt), u_rk_approx(23,T/dt), ...
    u_rk_approx(24,T/dt)]); norm([u_rk_approx(28,T/dt), u_rk_approx(29,T/dt), ...
    u_rk_approx(30,T/dt)]); norm([u_rk_approx(34,T/dt), u_rk_approx(35,T/dt), ...
    u_rk_approx(36,T/dt)])];
JulianDate_Final_V_RK4 = 10^3 * [norm(v2);norm(v3);norm(v4);norm(v5);norm(v6)];
CalculatedMeanVelocity_RK4 = [rk_vel_avg(2); rk_vel_avg(3); rk_vel_avg(4); ...
    rk_vel_avg(5); rk_vel_avg(6)];
KnownMeanVelocity_RK4 = 10^3 * [Jupiter_Mean; Saturn_Mean; Uranus_Mean; ...
    Neptune_Mean; Pluto_Mean];

% Defining Tables
Table1 = table(Planetary_Bodies, Calculated_Position_RK4, ...
    JulianDate_Position_RK4);
Table2 = table(Planetary_Bodies, Calculated_Final_V_RK4, ...
    JulianDate_Final_V_RK4, CalculatedMeanVelocity_RK4, KnownMeanVelocity_RK4);

% Get the table in string form.
TString = evalc('disp(Table1)');
% Use TeX Markup for bold formatting and underscores.
TString = strrep(TString,'<strong>','\bf');
TString = strrep(TString,'</strong>','\rm');
TString = strrep(TString,'_','\_');
% Get a fixed-width font.
FixedWidth = get(0,'FixedWidthFontName');
% Output the table using the annotation command.
figure(300)
annotation(gcf,'Textbox','String',TString,'Interpreter','Tex','FontName',...
     FixedWidth,'Units','Normalized','Position',[0 0 1 1]);

% Get the table in string form.
TString = evalc('disp(Table2)');
% Use TeX Markup for bold formatting and underscores.
TString = strrep(TString,'<strong>','\bf');
TString = strrep(TString,'</strong>','\rm');
TString = strrep(TString,'_','\_');
% Get a fixed-width font.
FixedWidth = get(0,'FixedWidthFontName');
% Output the table using the annotation command.
```

```
figure(400)
annotation(gcf,'Textbox','String',TString,'Interpreter','Tex','FontName',...
    FixedWidth,'Units','Normalized','Position',[0 0 1 1]);

% Plotting the Orbits Estimated
figure(2)
scatter4(0,0,0,40,'filled')
grid on
hold on
plot3(u_rk_approx(7,:), u_rk_approx(8,:), u_rk_approx(9,:),'LineWidth', 2)
scatter4(u_rk_approx(7,1), u_rk_approx(8,1), u_rk_approx(9,1),40,'filled')
plot3(u_rk_approx(13,:), u_rk_approx(14,:), u_rk_approx(15,:),'LineWidth', 2)
scatter4(u_rk_approx(13,1), u_rk_approx(14,1), u_rk_approx(15,1),40,'filled')
plot3(u_rk_approx(19,:), u_rk_approx(20,:), u_rk_approx(21,:),'LineWidth', 2)
scatter4(u_rk_approx(19,1), u_rk_approx(20,1), u_rk_approx(21,1),40,'filled')
plot3(u_rk_approx(25,:), u_rk_approx(26,:), u_rk_approx(27,:),'LineWidth', 2)
scatter4(u_rk_approx(25,1), u_rk_approx(26,1), u_rk_approx(27,1),40,'filled')
plot3(u_rk_approx(31,:), u_rk_approx(32,:), u_rk_approx(33,:),'LineWidth', 2)
scatter4(u_rk_approx(31,1), u_rk_approx(32,1), u_rk_approx(33,1),40,'filled')
title('Outer Solar System (Runge-Kutta 4 Method)')
legend('Sun','Jupiter Orbit','Jupiter','Saturn Orbit','Saturn','Uranus ...
    Orbit','Uranus','Neptune Orbit','Neptune','Pluto Orbit','Pluto')


%% Convergence Test

% AB2
%--simulation params
% dtvect = [60*60*24*6.25, 60*60*24*1.25, 60*60*24*0.25, 60*60*24*0.05, ...
    60*60*24*0.01, 60*60*24*0.002]; % For Inner Orbits
dtvect = 60*60*24 * [781.25, 156.25, 31.25, 6.25, 1.25, 0.25]; % For Outer Orbits
%--

%initialize vector that stores approximate soln at T for various dt
u_AB2_keep = zeros( 36,length( dtvect ) );
tic
%advance in time
for j = 1 : length(dtvect)

    %current dt
    dt = dtvect(j);

    tk = t0; %initialize time iterate

    %initialize iterates for various methods
    u_AB2_k = u0;
    u_AB2_km1 = u0;
```

```
    %run to final time T
    for jj = 1 : T/dt

        %AB2
        if jj < 2

            %advance with Heun's for 1st time step
            u_AB2_kp1 = (u_AB2_k + 1/2*dt*(f(u_AB2_k) + f(u_AB2_k + ...
                dt*f(u_AB2_k))));

        else
            u_AB2_kp1 = u_AB2_k + 1/2*dt*(-1*f(u_AB2_km1) + 3*f(u_AB2_k));

        end

        %update iterates
        u_AB2_km1 = u_AB2_k;
        u_AB2_k = u_AB2_kp1;

        %update time
        tk = tk + dt;


    end

    %store soln at T
    u_AB2_keep( :,j ) = u_AB2_kp1;

end

%compute difference between solution at smallest dt and the other dts

%initialize vector
u_AB2_diff = zeros( length( dtvect )-1,1 );

for j = 1 : length( dtvect )-1

    %AB2
    u_AB2_diff(j) = norm( u_AB2_keep(:,j) - u_AB2_keep(:,end) )/ ...
        norm( u_AB2_keep(:,end) );

end
toc
figure(50)
%AB2
```

```
loglog( dtvect(1:end-1), u_AB2_diff, '.-', 'markersize', 20, 'linewidth', 2 )

set( gca, 'fontsize', 16, 'ticklabelinterpreter', 'latex' )

leg = legend( 'AB2' );
set( leg, 'fontsize', 12, 'interpreter', 'latex', 'location', 'southeast' )
xlabel('$\Delta t$', 'fontsize', 12, 'interpreter' , 'latex')
title('$\frac{||u_{\Delta t} - u_{2.5\times10^{-4}}||}{|| ...
    u_{2.5\times10^{-4}} ||}$', 'fontsize', 28,  'interpreter' , 'latex' )
set(gcf, 'PaperPositionMode', 'manual')
set(gcf, 'Color', [1 1 1])
set(gca, 'Color', [1 1 1])
set(gcf, 'PaperUnits', 'centimeters')
set(gcf, 'PaperSize', [15 15])
set(gcf, 'Units', 'centimeters' )
set(gcf, 'Position', [0 0 15 15])
set(gcf, 'PaperPosition', [0 0 15 15])

svnm = 'ConvergencePlots1';
print( '-dpng', svnm, '-r300' )


% RK4
%--simulation params
% dtvect = [60*60*24*6.25, 60*60*24*1.25, 60*60*24*0.25, 60*60*24*0.05, ...
    60*60*24*0.01, 60*60*24*0.002]; % For Inner Orbits
dtvect = 60*60*24 * [781.25, 156.25, 31.25, 6.25, 1.25, 0.25]; % For Outer Orbits
%--

%initialize vector that stores approximate soln at T for various dt
u_r_keep = zeros( 36,length( dtvect ) );
tic
%advance in time
for j = 1 : length(dtvect)

    %current dt
    dt = dtvect(j);

    tk = t0; %initialize time iterate

    %initialize iterates for various methods
    u_r_k = u0;


    %run to final time T
    for jj = 1 : T/dt

```

```
        %RK4
        y1 = f(u_r_k);
        y2 = f(u_r_k + 1/2*dt*y1);
        y3 = f(u_r_k + (1/2)*dt*y2);
        y4 = f(u_r_k + dt*y3);
        u_r_kp1 = u_r_k + 1/6*dt*(y1 + 2*y2 + 2*y3 + y4);

        %update iterates
        u_r_k = u_r_kp1;

        %update time
        tk = tk + dt;


    end

    %store soln at T
    u_r_keep( :,j ) = u_r_kp1;

end

%compute difference between solution at smallest dt and the other dts

%initialize vector
u_r_diff = zeros( length( dtvect )-1,1 );

for j = 1 : length( dtvect )-1

    %RK4
    u_r_diff(j) = norm( u_r_keep(:,j) - u_r_keep(:,end) )/ ...
        norm( u_r_keep(:,end) );

end
toc
figure(50), hold on
%RK4
loglog( dtvect(1:end-1), u_r_diff, '.-', 'markersize', 20, 'linewidth', 2 )

set( gca, 'fontsize', 12, 'ticklabelinterpreter', 'latex' )

leg = legend( 'AB2', 'RK4' );
set( leg, 'fontsize', 16, 'interpreter', 'latex', 'location', 'southeast' )
xlabel('$\Delta t$', 'fontsize', 12, 'interpreter' , 'latex')
title('$\frac{||u_{\Delta t} - u_{2.5\times10^{-4}}||}{|| ...
    u_{2.5\times10^{-4}} ||}$', 'fontsize', 28,  'interpreter' , 'latex' )
set(gcf, 'PaperPositionMode', 'manual')
set(gcf, 'Color', [1 1 1])
```

```
set(gca, 'Color', [1 1 1])
set(gcf, 'PaperUnits', 'centimeters')
set(gcf, 'PaperSize', [15 15])
set(gcf, 'Units', 'centimeters' )
set(gcf, 'Position', [0 0 15 15])
set(gcf, 'PaperPosition', [0 0 15 15])

svnm = 'ConvergencePlots2';
print( '-dpng', svnm, '-r300' )


% Euler
%--simulation params
% dtvect = [60*60*24*6.25, 60*60*24*1.25, 60*60*24*0.25, 60*60*24*0.05, ...
    60*60*24*0.01, 60*60*24*0.002]; % For Inner Orbits
dtvect = 60*60*24 * [781.25, 156.25, 31.25, 6.25, 1.25, 0.25]; % For Outer Orbits
%--

%initialize vector that stores approximate soln at T for various dt
u_eu_approx = zeros( 36,length( dtvect ) );

%advance in time
for j = 1 : length(dtvect)

    %current dt
    dt = dtvect(j);

    tk = t0; %initialize time iterate

    %initialize iterates for various methods
    u_eu_k = u0;


    %run to final time T
    for jj = 1 : T/dt

        %Euler
        u_eu_kp1 = u_eu_k + dt*f(u_eu_k);

        %update iterates
        u_eu_k = u_eu_kp1;

        %update time
        tk = tk + dt;


    end
```

```
    %store soln at T
    u_eu_approx( :,j ) = u_eu_kp1;

end

%compute difference between solution at smallest dt and the other dts

%initialize vector
u_eu_diff = zeros( length( dtvect )-1,1 );

for j = 1 : length( dtvect )-1

    %Euler
    u_eu_diff(j) = norm( u_eu_approx(:,j) - u_eu_approx(:,end) )/ ...
        norm( u_eu_approx(:,end) );

end

figure(50), hold on
%Euler
loglog( dtvect(1:end-1), u_eu_diff, '.-', 'markersize', 20, 'linewidth', 2 )

set( gca, 'fontsize', 16, 'ticklabelinterpreter', 'latex' )

leg = legend( 'AB2', 'RK4', 'Euler' );
set( leg, 'fontsize', 12, 'interpreter', 'latex', 'location', 'southeast' )
xlabel('$\Delta t$', 'fontsize', 12, 'interpreter' , 'latex')
title('$\frac{||u_{\Delta t} - u_{2.5\times10^{-4}}||}{|| ...
    u_{2.5\times10^{-4}} ||}$', 'fontsize', 28,  'interpreter' , 'latex' )
set(gcf, 'PaperPositionMode', 'manual')
set(gcf, 'Color', [1 1 1])
set(gca, 'Color', [1 1 1])
set(gcf, 'PaperUnits', 'centimeters')
set(gcf, 'PaperSize', [15 15])
set(gcf, 'Units', 'centimeters' )
set(gcf, 'Position', [0 0 15 15])
set(gcf, 'PaperPosition', [0 0 15 15])

svnm = 'error_q2_Euler';
print( '-dpng', svnm, '-r200' )


% Heun's
%--simulation params
% dtvect = [60*60*24*6.25, 60*60*24*1.25, 60*60*24*0.25, 60*60*24*0.05, ...
    60*60*24*0.01, 60*60*24*0.002]; % For Inner Orbits
```

```
dtvect = 60*60*24 * [781.25, 156.25, 31.25, 6.25, 1.25, 0.25]; % For Outer Orbits
%--

%initialize vector that stores approximate soln at T for various dt
u_h_keep = zeros( 36,length( dtvect ) );

%advance in time
for j = 1 : length(dtvect)

    %current dt
    dt = dtvect(j);

    tk = t0; %initialize time iterate

    %initialize iterates for various methods
    u_h_k = u0;


    %run to final time T
    for jj = 1 : T/dt

        %Huens
        u_h_kp1 = u_h_k + 1/2*dt*(f(u_h_k) + f(u_h_k + dt*f(u_h_k)));

        %update iterates
        u_h_k = u_h_kp1;

        %update time
        tk = tk + dt;


    end

    %store soln at T
    u_h_keep( :,j ) = u_h_kp1;

end

%compute difference between solution at smallest dt and the other dts

%initialize vector
u_h_diff = zeros( length( dtvect )-1,1 );

for j = 1 : length( dtvect )-1

    %Huens
    u_h_diff(j) = norm( u_h_keep(:,j) - u_h_keep(:,end) )/ ...
```

```
        norm( u_h_keep(:,end) );

end

figure(50), hold on
%Heun's
loglog( dtvect(1:end-1), u_h_diff, '.-', 'markersize', 20, 'linewidth', 2 )

set( gca, 'fontsize', 16, 'ticklabelinterpreter', 'latex' )

leg = legend( 'AB2', 'RK4', 'Euler', 'Huens' );
set( leg, 'fontsize', 12, 'interpreter', 'latex', 'location', 'southeast' )
xlabel('$\Delta t$', 'fontsize', 12, 'interpreter' , 'latex')
title('$\frac{||u_{\Delta t} - u_{2.5\times10^{-4}}||}{|| ...
    u_{2.5\times10^{-4}} ||}$', 'fontsize', 28,  'interpreter' , 'latex' )
set(gcf, 'PaperPositionMode', 'manual')
set(gcf, 'Color', [1 1 1])
set(gca, 'Color', [1 1 1])
set(gcf, 'PaperUnits', 'centimeters')
set(gcf, 'PaperSize', [15 15])
set(gcf, 'Units', 'centimeters' )
set(gcf, 'Position', [0 0 15 15])
set(gcf, 'PaperPosition', [0 0 15 15])

svnm = 'error_q2_Huens';
print( '-dpng', svnm, '-r200' )


%% Error Plots

% % Gathering N-Body Information for 687 days in future from Jan 1, 2011 (T)
% Julian2 = juliandate(2012,11,19);

% Gathering N-Body Information for 249 years in future from Jan 1, 2011 (T)
Julian2 = juliandate(2260,12,31);

% % Position and Velocity Vectors at future position
% [r1, v1] = planetEphemeris(Julian2,'Sun', 'Sun', '432t', 'km');
% [r2, v2] = planetEphemeris(Julian2,'Sun','Mercury','432t','km');
% [r3, v3] = planetEphemeris(Julian2,'Sun','Venus','432t','km');
% [r4, v4] = planetEphemeris(Julian2,'Sun','Earth','432t','km');
% [r5, v5] = planetEphemeris(Julian2,'Sun','Mars','432t','km');
% [r6, v6] = planetEphemeris(Julian2,'Sun','Moon','432t','km');


% Position and Velocity Vectors at future position
[r1, v1] = planetEphemeris(Julian2,'Sun', 'Sun', '432t', 'km');
```

```
[r2, v2] = planetEphemeris(Julian2,'Sun','Jupiter','432t','km');
[r3, v3] = planetEphemeris(Julian2,'Sun','Saturn','432t','km');
[r4, v4] = planetEphemeris(Julian2,'Sun','Uranus','432t','km');
[r5, v5] = planetEphemeris(Julian2,'Sun','Neptune','432t','km');
[r6, v6] = planetEphemeris(Julian2,'Sun','Pluto','432t','km');

[u_real] = 10^3 * [r1'; v1'; r2'; v2'; r3'; v3'; r4'; v4'; r5'; v5'; r6'; v6'];

% AB2
%--simulation params
% dtvect = [60*60*24*6.25, 60*60*24*1.25, 60*60*24*0.25, 60*60*24*0.05, ...
    60*60*24*0.01, 60*60*24*0.002];
dtvect = 60*60*24 * [781.25, 156.25, 31.25, 6.25, 1.25, 0.25];
%--

%initialize vector that stores approximate soln at T for various dt
u_AB2_keep = zeros( 36,length( dtvect ) );
tic
%advance in time
for j = 1 : length(dtvect)

    %current dt
    dt = dtvect(j);

    tk = t0; %initialize time iterate

    %initialize iterates for various methods
    u_AB2_k = u0;
    u_AB2_km1 = u0;


    %run to final time T
    for jj = 1 : T/dt

        %AB2
        if jj < 2

            %advance with Heun's for 1st time step
            u_AB2_kp1 = (u_AB2_k + 1/2*dt*(f(u_AB2_k) + f(u_AB2_k + ...
                dt*f(u_AB2_k))));

        else
            u_AB2_kp1 = u_AB2_k + 1/2*dt*(-1*f(u_AB2_km1) + 3*f(u_AB2_k));

        end

        %update iterates
```

```
        u_AB2_km1 = u_AB2_k;
        u_AB2_k = u_AB2_kp1;

        %update time
        tk = tk + dt;


    end

    %store soln at T
    u_AB2_keep( :,j ) = u_AB2_kp1;

end

%compute difference between solution at smallest dt and the other dts

%initialize vector
u_AB2_error = zeros( length( dtvect ),1 );

for j = 1 : length( dtvect )

    %AB2
    u_AB2_error(j) = norm( u_AB2_keep(:,j) - u_real(:) )/ ...
        norm( u_real(:) );

end
toc
figure(50)
%AB2
loglog( dtvect(1:end), u_AB2_error, '.-', 'markersize', 20, 'linewidth', 2 )

set( gca, 'fontsize', 16, 'ticklabelinterpreter', 'latex' )

leg = legend( 'AB2' );
set( leg, 'fontsize', 12, 'interpreter', 'latex', 'location', 'southeast' )
xlabel('$\Delta t$', 'fontsize', 12, 'interpreter' , 'latex')
ylabel('Error %', 'fontsize', 12, 'interpreter' , 'latex')
title('Error Plot', 'fontsize', 28,  'interpreter' , 'latex' )
set(gcf, 'PaperPositionMode', 'manual')
set(gcf, 'Color', [1 1 1])
set(gca, 'Color', [1 1 1])
set(gcf, 'PaperUnits', 'centimeters')
set(gcf, 'PaperSize', [15 15])
set(gcf, 'Units', 'centimeters' )
set(gcf, 'Position', [0 0 15 15])
set(gcf, 'PaperPosition', [0 0 15 15])
```

```
svnm = 'ErrorPlots1';
print( '-dpng', svnm, '-r300' )



% RK4
%--simulation params
% dtvect = [60*60*24*6.25, 60*60*24*1.25, 60*60*24*0.25, 60*60*24*0.05, ...
    60*60*24*0.01, 60*60*24*0.002];
dtvect = 60*60*24 * [781.25, 156.25, 31.25, 6.25, 1.25, 0.25];
%--

%initialize vector that stores approximate soln at T for various dt
u_r_keep = zeros( 36,length( dtvect ) );
tic
%advance in time
for j = 1 : length(dtvect)

    %current dt
    dt = dtvect(j);

    tk = t0; %initialize time iterate

    %initialize iterates for various methods
    u_r_k = u0;


    %run to final time T
    for jj = 1 : T/dt

        %RK4
        y1 = f(u_r_k);
        y2 = f(u_r_k + 1/2*dt*y1);
        y3 = f(u_r_k + (1/2)*dt*y2);
        y4 = f(u_r_k + dt*y3);
        u_r_kp1 = u_r_k + 1/6*dt*(y1 + 2*y2 + 2*y3 + y4);

        %update iterates
        u_r_k = u_r_kp1;

        %update time
        tk = tk + dt;


    end

    %store soln at T
```

```
    u_r_keep( :,j ) = u_r_kp1;

end

%compute the error

%initialize vector
u_r_error = zeros( length( dtvect ),1 );

for j = 1 : length( dtvect )

    %RK4
    u_r_error(j) = (norm( u_r_keep(:,j) - u_real(:) )/ ...
        norm( u_real(:) ));

end
toc
figure(50), hold on
%RK4
loglog( dtvect(1:end), u_r_error, '.-', 'markersize', 20, 'linewidth', 2 )

set( gca, 'fontsize', 12, 'ticklabelinterpreter', 'latex' )

leg = legend( 'AB2', 'RK4' );
set( leg, 'fontsize', 16, 'interpreter', 'latex', 'location', 'southeast' )
xlabel('$\Delta t$', 'fontsize', 12, 'interpreter' , 'latex')
ylabel('Error %', 'fontsize', 12, 'interpreter' , 'latex')
title('Error Plot', 'fontsize', 28,  'interpreter' , 'latex' )
set(gcf, 'PaperPositionMode', 'manual')
set(gcf, 'Color', [1 1 1])
set(gca, 'Color', [1 1 1])
set(gcf, 'PaperUnits', 'centimeters')
set(gcf, 'PaperSize', [15 15])
set(gcf, 'Units', 'centimeters' )
set(gcf, 'Position', [0 0 15 15])
set(gcf, 'PaperPosition', [0 0 15 15])

svnm = 'ErrorPlots2';
print( '-dpng', svnm, '-r300' )
```