

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:

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

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 k th order polynomial results in a $k + 1$ th order

¹http://web.mit.edu/10.001/Web/Course_Notes/Differential_Equations_Notes/node6.html

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

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

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:

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

where

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

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

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 principle of the predictor-corrector 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 evaluation 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 explicit methods, which is shown below:

²Lecture 2 26 Typed

³http://web.mit.edu/10.001/Web/Course_Notes/Differential_Equations_Notes/node5.html

⁴Lecture 2 24 Typed

⁵<https://www.phy.ornl.gov/csep/ode/node12.html>

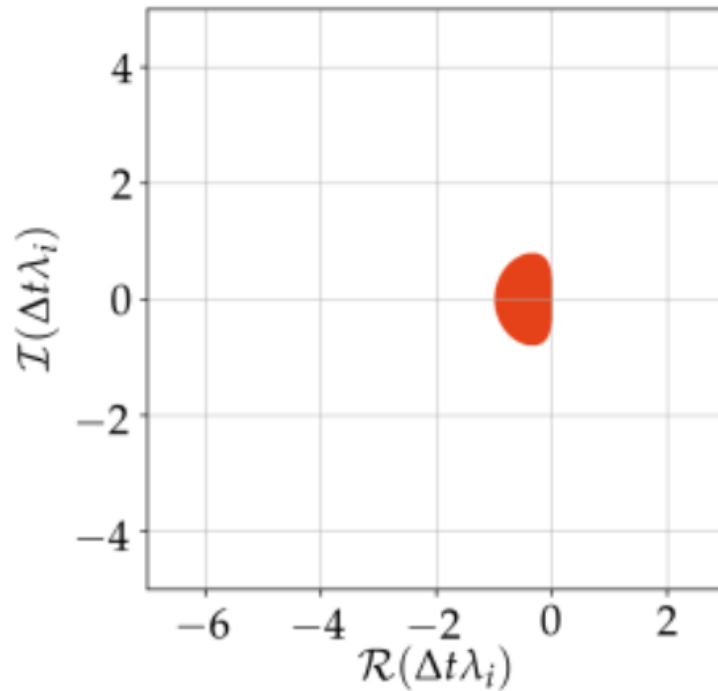


Figure 1: Stability Region for 2-Step Adams-Bashforth Method⁶

2.2.2 RK4

The RK4 Method is one of the most flexible methods that can be used for analysis. It provides a local truncation error of $\mathcal{O}(\Delta t^5)$ and the accumulated error is $\mathcal{O}(\Delta t^4)$, 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:

⁶Week 7 Typed Notes

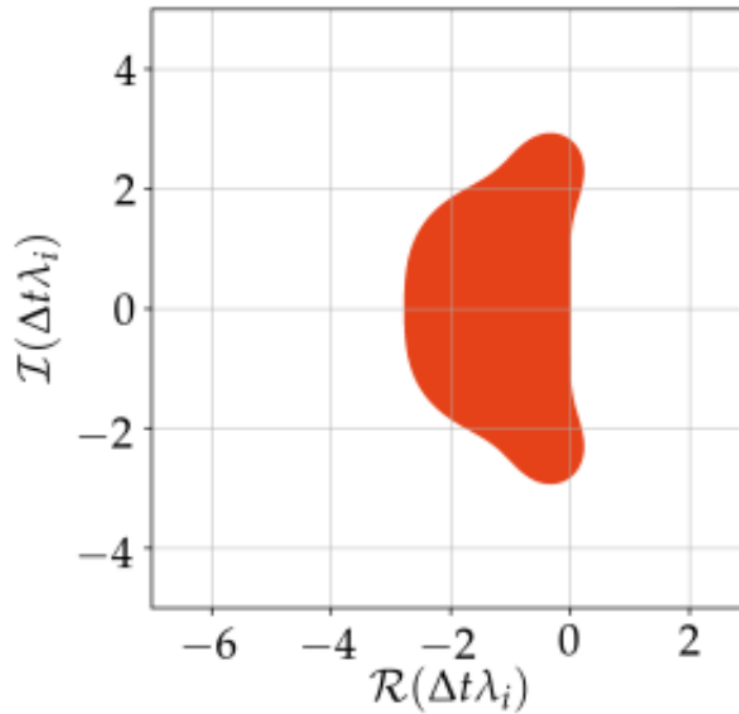


Figure 2: Stability Region for 4th Order Runge-Kutta Method ⁷

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 Δt s were used. The Δt s 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)

⁷Week 7 Typed Notes

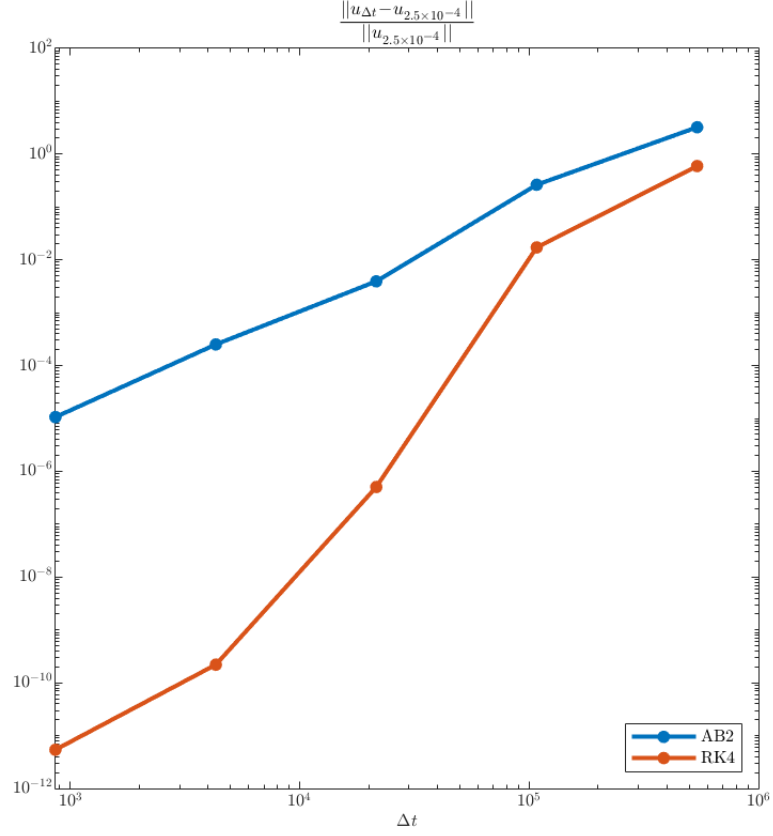


Figure 3: Convergence Plots for Inner Solar System

Outer Solar System

For the Outer Solar System, again 6 different Δt s were used. The Δt s 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).

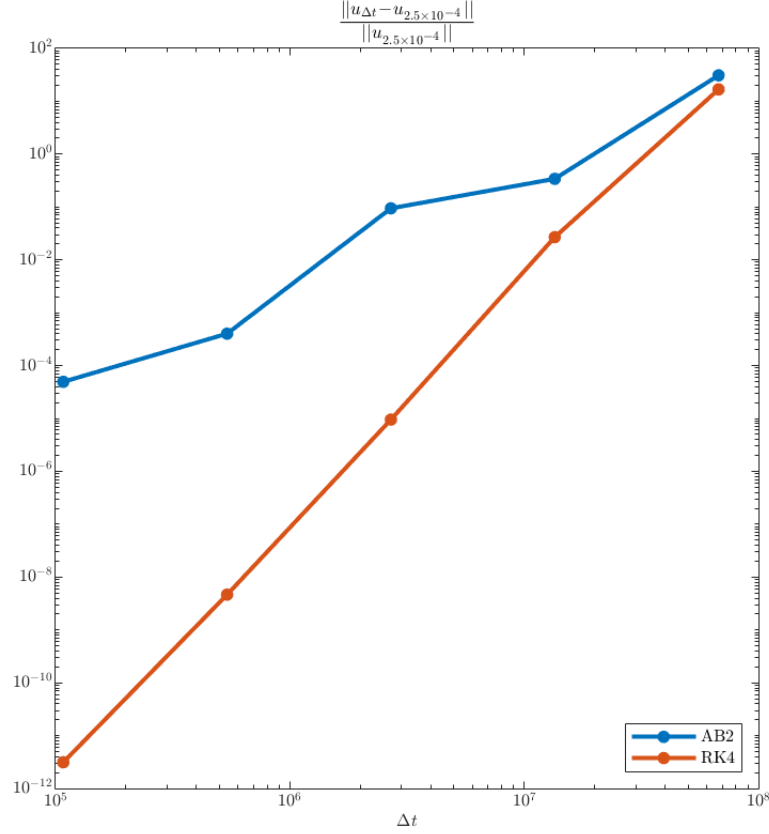


Figure 4: Convergence Plots for Inner Solar System

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 Δt s 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:

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

where

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

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:

```

1 % Gathering N-Body Information for the date of Jan 1, 2011
2 Julian = juliandate(2011,1,1);
3
4 % Position and Velocity Vectors
5 [r1, v1] = planetEphemeris(Julian, 'Sun', 'Sun', '432t', 'km');
6 [r2, v2] = planetEphemeris(Julian, 'Sun', 'Mercury', '432t', 'km');
7 [r3, v3] = planetEphemeris(Julian, 'Sun', 'Venus', '432t', 'km');
8 [r4, v4] = planetEphemeris(Julian, 'Sun', 'Earth', '432t', 'km');
9 [r5, v5] = planetEphemeris(Julian, 'Sun', 'Mars', '432t', 'km');
10 [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:

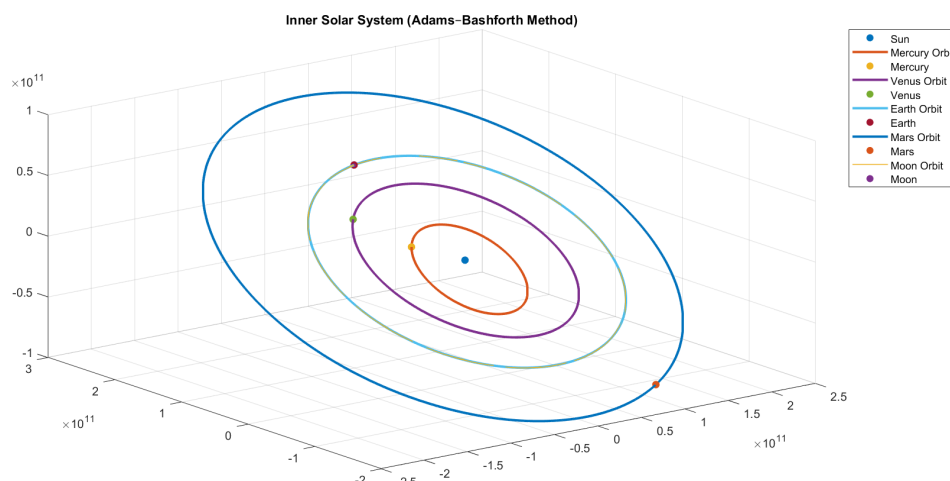


Figure 5: Inner Orbits Using AB2

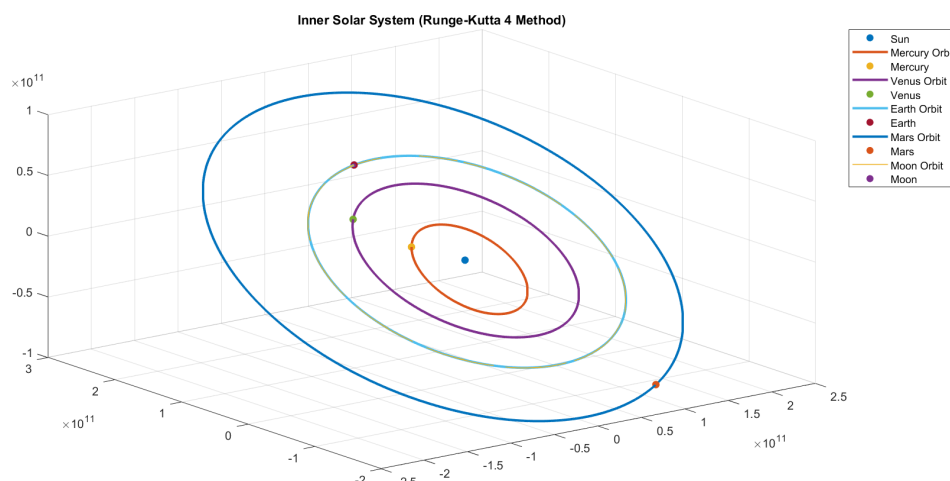


Figure 6: Inner Orbits Using RK4

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 .

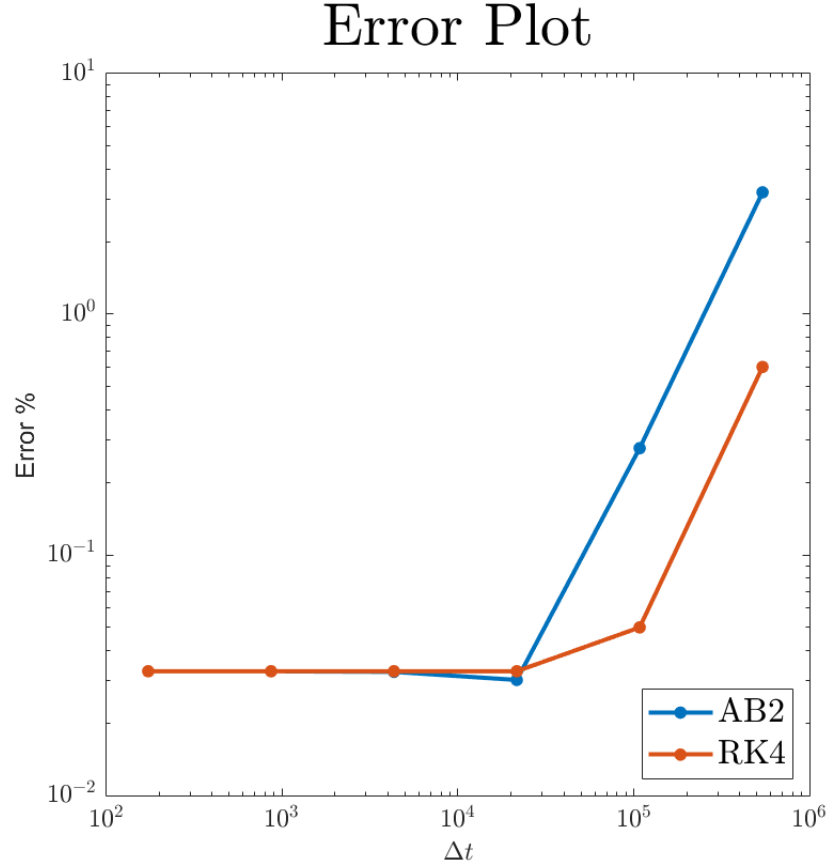


Figure 11: Error Plot for Inner Orbits

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:

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

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

```

1 % Gathering N-Body Information for the date of Jan 1, 2011
2 Julian = juliandate(2011,1,1);
3
4 % Position and Velocity Vectors
5 [r1, v1] = planetEphemeris(Julian,'Sun', 'Sun', '432t', 'km');
6 [r2, v2] = planetEphemeris(Julian,'Sun','Jupiter','432t','km');
7 [r3, v3] = planetEphemeris(Julian,'Sun','Saturn','432t','km');
8 [r4, v4] = planetEphemeris(Julian,'Sun','Uranus','432t','km');
9 [r5, v5] = planetEphemeris(Julian,'Sun','Neptune','432t','km');
10 [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 step 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:

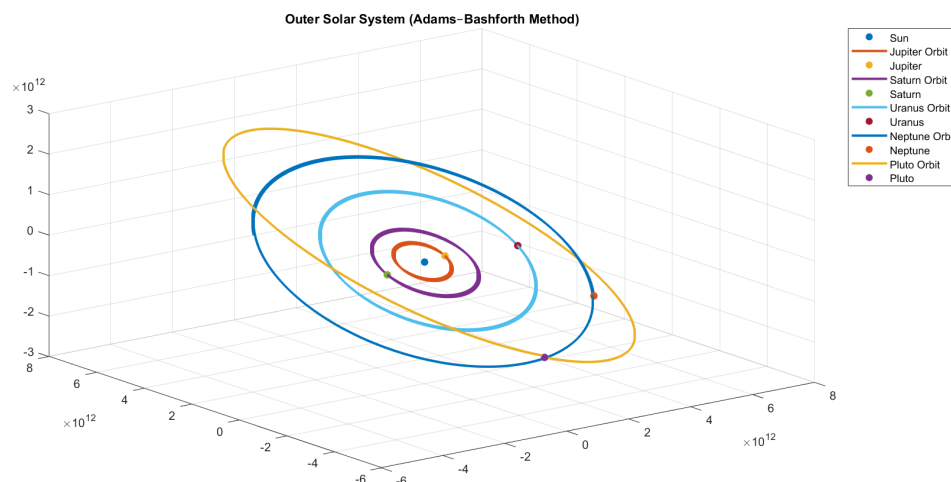


Figure 12: Inner Orbits Using AB2

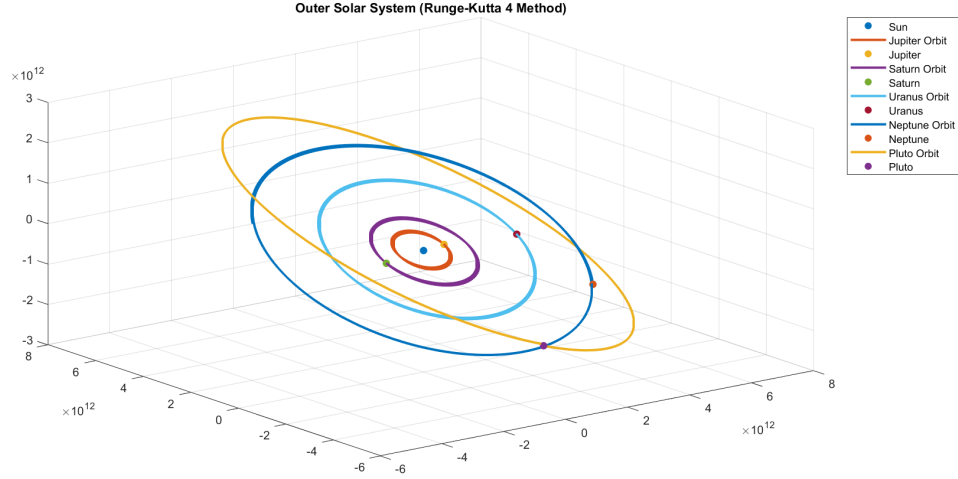


Figure 13: Inner Orbits Using RK4

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 .

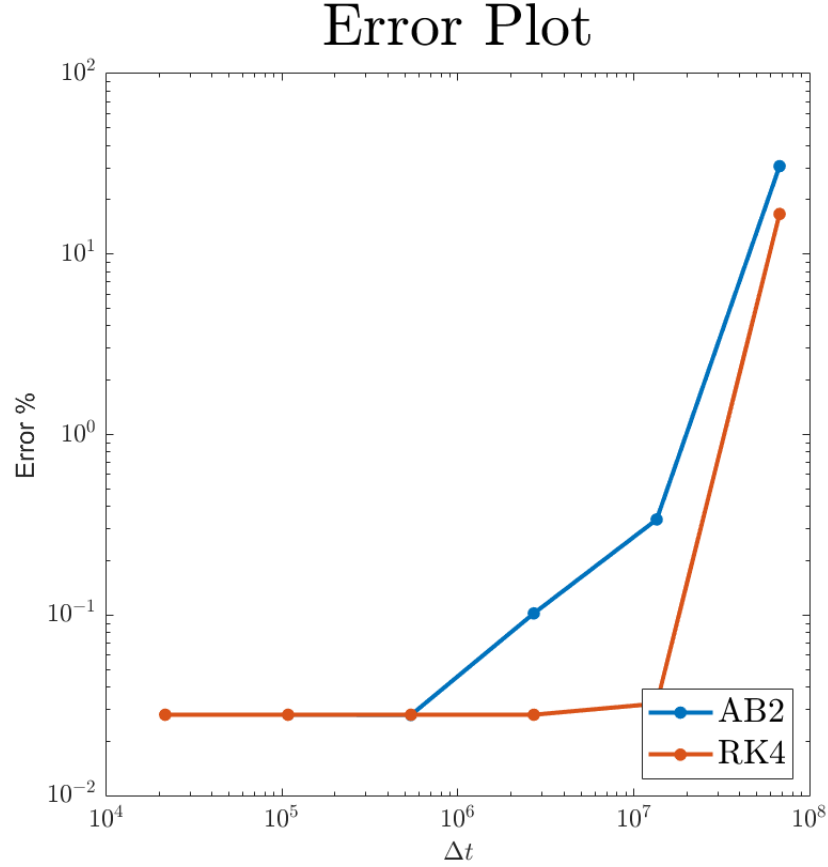


Figure 18: Error Plot for Inner Orbits

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

```

1 %% N-Body Problem where N = 6
2 clear all, close all, clc
3
4 % Nomenclature
5 % 1= Sun
6 % 2 = Mercury
7 % 3 = Venus
8 % 4 = Earth
9 % 5 = Mars
10 % 6 = Moon
11
12 % Orbital Velocities of Planets in km/s (Data from NASA Fact Sheet)
13 % Mercury
14 Mercury_Min = 38.86; Mercury_Max = 58.98; Mercury_Mean = 47.36;
15 % Venus
16 Venus_Min = 34.79; Venus_Max = 35.26; Venus_Mean = 35.02;
17 % Earth
18 Earth_Min = 29.29; Earth_Max = 30.29; Earth_Mean = 29.78;
19 % Mars
20 Mars_Min = 21.97; Mars_Max = 26.50; Mars_Mean = 24.07;
21 % Moon
22 Moon_Min = 0.970; Moon_Max = 1.082; Moon_Mean = 1.022;
23
24 % Gathering N-Body Information for the date of Jan 1, 2011
25 Julian = juliandate(2011,1,1);
26
27 % Position and Velocity Vectors
28 [r1, v1] = planetEphemeris(Julian, 'Sun', 'Sun', '432t', 'km');
29 [r2, v2] = planetEphemeris(Julian, 'Sun', 'Mercury', '432t', 'km');
30 [r3, v3] = planetEphemeris(Julian, 'Sun', 'Venus', '432t', 'km');
31 [r4, v4] = planetEphemeris(Julian, 'Sun', 'Earth', '432t', 'km');
32 [r5, v5] = planetEphemeris(Julian, 'Sun', 'Mars', '432t', 'km');
33 [r6, v6] = planetEphemeris(Julian, 'Sun', 'Moon', '432t', 'km');
34

```

```

35 % Defining u0 (in meters)
36 u0 = 10^3 * [r1'; v1'; r2'; v2'; r3'; v3'; r4'; v4'; r5'; v5'; r6'; v6'];
37
38 % Masses (in kg)
39 m1 = 1.9891 * 10^30;
40 m2 = 3.285 * 10^23;
41 m3 = 4.867 * 10^24;
42 m4 = 5.97 * 10^24;
43 m5 = 7.35 * 10^22;
44 m6 = 7.34767 * 10^22;
45
46 % Defining the Parameters
47 G = 6.67*10^(-11); % Gravitational Constant
48 t0 = 0; % Initial Start Time
49 T = 60*60*24*687*1; % Final Time (1 Mars Revolution in Seconds)
50 dt = 60*60*24*0.002; % Time step (0.002 Days in Seconds)
51
52 % State-Space Form Vectors (For Reference)
53 %r1 = u(1:3); v1 = u(4:6);
54 %r2 = u(7:9); v2 = u(10:12);
55 %r3 = u(13:15); v3 = u(16:18);
56 %r4 = u(19:21); v4 = u(22:24);
57 %r5 = u(25:27); v5 = u(28:30);
58 %r6 = u(31:33); v6 = u(34:36);
59
60 % State-Space Model
61 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); ...
62     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); ...
63     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); ...
64     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); ...

```

```

65     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); ...
66     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)] ;
67
68 %%%%% AB2 Numerical Analysis %%%%%
69
70 tic
71
72 tk = 0; %starting time
73
74 %initialize iterates for AB2
75 u_AB2_k = u0;
76 u_AB2_km1 = u0;
77
78 %initialize vector that stores approximate soln at various times
79 u_AB2_approx = zeros( 36,T/dt );
80
81 %Initialize vector that stores the magnitude of velocities at T
82 u_AB2_vel = zeros(6,T/dt);
83
84 %advance to final time T
85 for i = 1 : T/dt
86
87     if i < 2
88
89         %advance with Heun's for 1st time step
90         u_AB2_approx(:,i) = (u_AB2_k + 1/2*dt*(f(u_AB2_k) + ...
            f(u_AB2_k+dt*f(u_AB2_k))));
91
92     else
93
94         u_AB2_approx(:,i) = u_AB2_k + 1/2*dt*(-1*f(u_AB2_km1) + 3*f(u_AB2_k));
95
96     end
97
98     %update iterates
99     tk = tk + dt;
100    u_AB2_km1 = u_AB2_k;
101    u_AB2_k = u_AB2_approx(:,i);
102

```

```

103     %Calculating and storing the velocities of each body at T
104     u_AB2_vel(1,i) = norm(norm(u_AB2_approx(4:6,i)));
105     u_AB2_vel(2,i) = norm(norm(u_AB2_approx(10:12,i)));
106     u_AB2_vel(3,i) = norm(norm(u_AB2_approx(16:18,i)));
107     u_AB2_vel(4,i) = norm(norm(u_AB2_approx(22:24,i)));
108     u_AB2_vel(5,i) = norm(norm(u_AB2_approx(28:30,i)));
109     u_AB2_vel(6,i) = norm(norm(u_AB2_approx(34:36,i)));
110
111 end
112
113 t_AB2 = toc
114
115 % Gathering N-Body Information for 687 days in future from Jan 1, 2011 (T)
116 Julian2 = juliandate(2012,11,19);
117
118 % Position and Velocity Vectors at future position
119 [r1, v1] = planetEphemeris(Julian2,'Sun', 'Sun', '432t', 'km');
120 [r2, v2] = planetEphemeris(Julian2,'Sun','Mercury','432t','km');
121 [r3, v3] = planetEphemeris(Julian2,'Sun','Venus','432t','km');
122 [r4, v4] = planetEphemeris(Julian2,'Sun','Earth','432t','km');
123 [r5, v5] = planetEphemeris(Julian2,'Sun','Mars','432t','km');
124 [r6, v6] = planetEphemeris(Julian2,'Sun','Moon','432t','km');
125
126 % Calculating the Average, Minimum and Maximum Velocity of Each Planet
127 AB2_vel_avg = mean(u_AB2_vel,2);
128 AB2_vel_max = max(u_AB2_vel,[],2);
129 AB2_vel_min = min(u_AB2_vel,[],2);
130
131 % Assigning Variable Names
132 Planetary_Bodies = {'Mercury';'Venus';'Earth';'Mars';'Moon'};
133 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)];
134 JulianDate_Position_AB2 = 10^3 * [r2;r3;r4;r5;r6];
135 % 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)])];
136 % JulianDate_Position = 10^3 * [norm(r2);norm(r3);norm(r4);norm(r5);norm(r6)];
137 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)]]);
138 JulianDate_Final_V_AB2 = 10^3 * [norm(v2);norm(v3);norm(v4);norm(v5);norm(v6)];
139 CalculatedMeanVelocity_AB2 = [AB2_vel_avg(2); AB2_vel_avg(3); ...
    AB2_vel_avg(4); AB2_vel_avg(5); AB2_vel_avg(6)];
140 KnownMeanVelocity_AB2 = 10^3 * [Mercury_Mean; Venus_Mean; Earth_Mean; ...
    Mars_Mean; Moon_Mean];
141
142 % Defining Tables
143 Table1 = table(Planetary_Bodies, Calculated_Position_AB2, ...
    JulianDate_Position_AB2);
144 Table2 = table(Planetary_Bodies, Calculated_Final_V_AB2, ...
    JulianDate_Final_V_AB2, CalculatedMeanVelocity_AB2, KnownMeanVelocity_AB2);
145
146 % Get the table in string form.
147 TString = evalc('disp(Table1)');
148 % Use TeX Markup for bold formatting and underscores.
149 TString = strrep(TString,'<strong>','\bf');
150 TString = strrep(TString,'</strong>','\rm');
151 TString = strrep(TString,'_','\_');
152 % Get a fixed-width font.
153 FixedWidth = get(0,'FixedWidthFontName');
154 % Output the table using the annotation command.
155 figure(100)
156 annotation(gcf,'Textbox','String',TString,'Interpreter','Tex','FontName',...
157     FixedWidth,'Units','Normalized','Position',[0 0 1 1]);
158
159 % Get the table in string form.
160 TString = evalc('disp(Table2)');
161 % Use TeX Markup for bold formatting and underscores.
162 TString = strrep(TString,'<strong>','\bf');
163 TString = strrep(TString,'</strong>','\rm');
164 TString = strrep(TString,'_','\_');
165 % Get a fixed-width font.
166 FixedWidth = get(0,'FixedWidthFontName');
167 % Output the table using the annotation command.
168 figure(200)
169 annotation(gcf,'Textbox','String',TString,'Interpreter','Tex','FontName',...
170     FixedWidth,'Units','Normalized','Position',[0 0 1 1]);
171
172 % Plotting the Orbits Estimated
173 figure(1)
174 scatter4(0,0,0,40,'filled')

```

```

175 grid on
176 hold on
177 plot3(u_AB2_approx(7,:), u_AB2_approx(8,:), u_AB2_approx(9,:), 'LineWidth', 2)
178 scatter4(u_AB2_approx(7,1), u_AB2_approx(8,1), u_AB2_approx(9,1), 40, 'filled')
179 plot3(u_AB2_approx(13,:), u_AB2_approx(14,:), ...
        u_AB2_approx(15,:), 'LineWidth', 2)
180 scatter4(u_AB2_approx(13,1), u_AB2_approx(14,1), u_AB2_approx(15,1), 40, 'filled')
181 plot3(u_AB2_approx(19,:), u_AB2_approx(20,:), ...
        u_AB2_approx(21,:), 'LineWidth', 2)
182 scatter4(u_AB2_approx(19,1), u_AB2_approx(20,1), u_AB2_approx(21,1), 40, 'filled')
183 plot3(u_AB2_approx(25,:), u_AB2_approx(26,:), ...
        u_AB2_approx(27,:), 'LineWidth', 2)
184 scatter4(u_AB2_approx(25,1), u_AB2_approx(26,1), u_AB2_approx(27,1), 40, 'filled')
185 plot3(u_AB2_approx(31,:), u_AB2_approx(32,:), ...
        u_AB2_approx(33,:), 'LineWidth', 0.75)
186 scatter4(u_AB2_approx(31,1), u_AB2_approx(32,1), u_AB2_approx(33,1), 10, 'filled')
187 title('Inner Solar System (Adams Bashforth Method)')
188 legend('Sun', 'Mercury Orbit', 'Mercury', 'Venus Orbit', 'Venus', 'Earth ...
        Orbit', 'Earth', 'Mars Orbit', 'Mars', 'Moon Orbit', 'Moon')
189
190
191 %%%%% RK4 Numerical Analysis %%%%%
192
193 tk = 0; %starting time
194
195 tic
196
197 %initialize iterates for Huens
198 u_rk_k = u0;
199
200 %initialize vector that stores approximate soln at various times
201 u_rk_approx = zeros( 36,T/dt );
202
203 %Initialize vector that stores the magnitude of velocities at T
204 u_rk_vel = zeros(6,T/dt);
205
206 %advance to final time T
207 for i = 1 : T/dt
208
209     %advance with RK4 for 1st time step
210     y1 = f(u_rk_k);
211     y2 = f(u_rk_k + 1/2*dt*y1);
212     y3 = f(u_rk_k + (1/2)*dt*y2);
213     y4 = f(u_rk_k + dt*y3);
214     u_rk_approx(:,i) = u_rk_k + 1/6*dt*(y1 + 2*y2 + 2*y3 + y4);
215
216     %update iterates

```



```

217     tk = tk + dt;
218     u_rk_k = u_rk_approx(:,i);
219
220     %Calculating and storing the velocities of each body at T
221     u_rk_vel(1,i) = norm(norm(u_rk_approx(4:6,i)));
222     u_rk_vel(2,i) = norm(norm(u_rk_approx(10:12,i)));
223     u_rk_vel(3,i) = norm(norm(u_rk_approx(16:18,i)));
224     u_rk_vel(4,i) = norm(norm(u_rk_approx(22:24,i)));
225     u_rk_vel(5,i) = norm(norm(u_rk_approx(28:30,i)));
226     u_rk_vel(6,i) = norm(norm(u_rk_approx(34:36,i)));
227
228 end
229
230 t_r = toc
231
232 % Gathering N-Body Information for 687 days in future from Jan 1, 2011 (T)
233 Julian2 = juliandate(2012,11,19);
234
235 % Position and Velocity Vectors at future position
236 [r1, v1] = planetEphemeris(Julian2,'Sun', 'Sun', '432t', 'km');
237 [r2, v2] = planetEphemeris(Julian2,'Sun','Mercury','432t','km');
238 [r3, v3] = planetEphemeris(Julian2,'Sun','Venus','432t','km');
239 [r4, v4] = planetEphemeris(Julian2,'Sun','Earth','432t','km');
240 [r5, v5] = planetEphemeris(Julian2,'Sun','Mars','432t','km');
241 [r6, v6] = planetEphemeris(Julian2,'Sun','Moon','432t','km');
242
243 % Calculating the Average Velocity of Each Planet
244 rk_vel_avg = mean(u_rk_vel,2);
245 rk_vel_max = max(u_rk_vel,[],2);
246 rk_vel_min = min(u_rk_vel,[],2);
247
248 % Assigning Variable Names
249 Planetary_Bodies = {'Mercury','Venus','Earth','Mars','Moon'};
250 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)];
251 JulianDate_Position_RK4 = 10^3 * [r2;r3;r4;r5;r6];
252 % 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)])];
253 % JulianDate_Position = 10^3 * [norm(r2);norm(r3);norm(r4);norm(r5);norm(r6)];

```

```

254 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)])];
255 JulianDate_Final_V_RK4 = 10^3 * [norm(v2);norm(v3);norm(v4);norm(v5);norm(v6)];
256 CalculatedMeanVelocity_RK4 = [rk_vel_avg(2); rk_vel_avg(3); rk_vel_avg(4); ...
    rk_vel_avg(5); rk_vel_avg(6)];
257 KnownMeanVelocity_RK4 = 10^3 * [Mercury_Mean; Venus_Mean; Earth_Mean; ...
    Mars_Mean; Moon_Mean];
258
259 % Defining Tables
260 Table1 = table(Planetary_Bodies, Calculated_Position_RK4, ...
    JulianDate_Position_RK4);
261 Table2 = table(Planetary_Bodies, Calculated_Final_V_RK4, ...
    JulianDate_Final_V_RK4, CalculatedMeanVelocity_RK4, KnownMeanVelocity_RK4);
262
263 % Get the table in string form.
264 TString = evalc('disp(Table1)');
265 % Use TeX Markup for bold formatting and underscores.
266 TString = strrep(TString, '<strong>', '\bf');
267 TString = strrep(TString, '</strong>', '\rm');
268 TString = strrep(TString, '_', '\_');
269 % Get a fixed-width font.
270 FixedWidth = get(0, 'FixedWidthFontName');
271 % Output the table using the annotation command.
272 figure(300)
273 annotation(gcf, 'Textbox', 'String', TString, 'Interpreter', 'Tex', 'FontName', ...
274     FixedWidth, 'Units', 'Normalized', 'Position', [0 0 1 1]);
275
276 % Get the table in string form.
277 TString = evalc('disp(Table2)');
278 % Use TeX Markup for bold formatting and underscores.
279 TString = strrep(TString, '<strong>', '\bf');
280 TString = strrep(TString, '</strong>', '\rm');
281 TString = strrep(TString, '_', '\_');
282 % Get a fixed-width font.
283 FixedWidth = get(0, 'FixedWidthFontName');
284 % Output the table using the annotation command.
285 figure(400)
286 annotation(gcf, 'Textbox', 'String', TString, 'Interpreter', 'Tex', 'FontName', ...
287     FixedWidth, 'Units', 'Normalized', 'Position', [0 0 1 1]);
288
289 % Plotting the Orbits Estimated
290 figure(2)
291 scatter4(0,0,0,40, 'filled')

```

```

292 grid on
293 hold on
294 plot3(u_rk_approx(7,:), u_rk_approx(8,:), u_rk_approx(9,:), 'LineWidth', 2)
295 scatter4(u_rk_approx(7,1), u_rk_approx(8,1), u_rk_approx(9,1), 40, 'filled')
296 plot3(u_rk_approx(13,:), u_rk_approx(14,:), u_rk_approx(15,:), 'LineWidth', 2)
297 scatter4(u_rk_approx(13,1), u_rk_approx(14,1), u_rk_approx(15,1), 40, 'filled')
298 plot3(u_rk_approx(19,:), u_rk_approx(20,:), u_rk_approx(21,:), 'LineWidth', 2)
299 scatter4(u_rk_approx(19,1), u_rk_approx(20,1), u_rk_approx(21,1), 40, 'filled')
300 plot3(u_rk_approx(25,:), u_rk_approx(26,:), u_rk_approx(27,:), 'LineWidth', 2)
301 scatter4(u_rk_approx(25,1), u_rk_approx(26,1), u_rk_approx(27,1), 40, 'filled')
302 plot3(u_rk_approx(31,:), u_rk_approx(32,:), u_rk_approx(33,:), 'LineWidth', 0.75)
303 scatter4(u_rk_approx(31,1), u_rk_approx(32,1), u_rk_approx(33,1), 10, 'filled')
304 title('Inner Solar System (Runge-Kutta 4 Method)')
305 legend('Sun', 'Mercury Orbit', 'Mercury', 'Venus Orbit', 'Venus', 'Earth ...
        Orbit', 'Earth', 'Mars Orbit', 'Mars', 'Moon Orbit', 'Moon')
306
307
308 %% N-Body Problem where N = 6 where Sun lost 25% of its mass
309 clear all, close all, clc
310
311 % Nomenclature
312 % 1 = Sun
313 % 2 = Jupiter
314 % 3 = Saturn
315 % 4 = Uranus
316 % 5 = Neptune
317 % 6 = Pluto
318
319 % Orbital Velocities of Planets in km/s (Data from NASA Fact Sheet)
320 % Jupiter
321 Jupiter_Min = 12.44; Jupiter_Max = 13.72; Jupiter_Mean = 13.06;
322 % Saturn
323 Saturn_Min = 9.09; Saturn_Max = 10.18; Saturn_Mean = 9.68;
324 % Uranus
325 Uranus_Min = 6.49; Uranus_Max = 7.11; Uranus_Mean = 6.80;
326 % Neptune
327 Neptune_Min = 5.37; Neptune_Max = 5.50; Neptune_Mean = 5.43;
328 % Pluto
329 Pluto_Min = 3.71; Pluto_Max = 6.10; Pluto_Mean = 4.67;
330
331 % Gathering N-Body Information for the date of Jan 1, 2011
332 Julian = juliandate(2011,1,1);
333
334 % Position and Velocity Vectors
335 [r1, v1] = planetEphemeris(Julian, 'Sun', 'Sun', '432t', 'km');
336 [r2, v2] = planetEphemeris(Julian, 'Sun', 'Jupiter', '432t', 'km');
337 [r3, v3] = planetEphemeris(Julian, 'Sun', 'Saturn', '432t', 'km');

```

```

338 [r4, v4] = planetEphemeris(Julian, 'Sun', 'Uranus', '432t', 'km');
339 [r5, v5] = planetEphemeris(Julian, 'Sun', 'Neptune', '432t', 'km');
340 [r6, v6] = planetEphemeris(Julian, 'Sun', 'Pluto', '432t', 'km');
341
342
343 % Defining u0 (in meters)
344 u0 = 10^3 * [r1'; v1'; r2'; v2'; r3'; v3'; r4'; v4'; r5'; v5'; r6'; v6'];
345
346 % Masses (in kg)
347 m1 = 1.9891 * 10^30;
348 m2 = 1.898 * 10^27;
349 m3 = 5.683 * 10^26;
350 m4 = 8.681 * 10^25;
351 m5 = 1.024 * 10^26;
352 m6 = 1.309 * 10^22;
353
354
355 % Defining the Parameters
356 G = 6.67*10^(-11); % Gravitational Constant
357 t0 = 0; % Initial Start Time
358 T = 60*60*24*365*250; % Final Time (1 Pluto Years)
359 dt = 60*60*24*0.25; % Time step (0.01 Days in Seconds)
360
361 % State-Space Form Vectors (For Reference)
362 %r1 = u(1:3); v1 = u(4:6);
363 %r2 = u(7:9); v2 = u(10:12);
364 %r3 = u(13:15); v3 = u(16:18);
365 %r4 = u(19:21); v4 = u(22:24);
366 %r5 = u(25:27); v5 = u(28:30);
367 %r6 = u(31:33); v6 = u(34:36);
368
369 % State-Space Model
370 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); ...
371     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); ...
372     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); ...

```

```

373     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); ...
374     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); ...
375     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)] ;

376
377 %%%%% AB2 Numerical Analysis %%%%%
378
379 tic
380
381 tk = 0; %starting time
382
383 %initialize iterates for AB2
384 u_AB2_k = u0;
385 u_AB2_km1 = u0;
386
387 %initialize vector that stores approximate soln at various times
388 u_AB2_approx = zeros( 36,T/dt );
389
390 %Initialize vector that stores the magnitude of velocities at T
391 u_AB2_vel = zeros(6,T/dt);
392
393 %advance to final time T
394 for i = 1 : T/dt
395
396     if i < 2
397
398         %advance with Heun's for 1st time step
399         u_AB2_approx(:,i) = (u_AB2_k + 1/2*dt*(f(u_AB2_k) + ...
            f(u_AB2_k+dt*f(u_AB2_k))));
400
401     else
402
403         u_AB2_approx(:,i) = u_AB2_k + 1/2*dt*(-1*f(u_AB2_km1) + 3*f(u_AB2_k));
404
405     end
406

```

```

407     %update iterates
408     tk = tk + dt;
409     u_AB2_km1 = u_AB2_k;
410     u_AB2_k = u_AB2_approx(:,i);
411
412     %Calculating and storing the velocities of each body at T
413     u_AB2_vel(1,i) = norm(norm(u_AB2_approx(4:6,i)));
414     u_AB2_vel(2,i) = norm(norm(u_AB2_approx(10:12,i)));
415     u_AB2_vel(3,i) = norm(norm(u_AB2_approx(16:18,i)));
416     u_AB2_vel(4,i) = norm(norm(u_AB2_approx(22:24,i)));
417     u_AB2_vel(5,i) = norm(norm(u_AB2_approx(28:30,i)));
418     u_AB2_vel(6,i) = norm(norm(u_AB2_approx(34:36,i)));
419
420 end
421
422 t_AB2 = toc
423
424 % Gathering N-Body Information for 249 years in future from Jan 1, 2011 (T)
425 Julian2 = juliandate(2260,12,31);
426
427 % Position and Velocity Vectors at future position
428 [r1, v1] = planetEphemeris(Julian2,'Sun', 'Sun', '432t', 'km');
429 [r2, v2] = planetEphemeris(Julian2,'Sun','Jupiter','432t','km');
430 [r3, v3] = planetEphemeris(Julian2,'Sun','Saturn','432t','km');
431 [r4, v4] = planetEphemeris(Julian2,'Sun','Uranus','432t','km');
432 [r5, v5] = planetEphemeris(Julian2,'Sun','Neptune','432t','km');
433 [r6, v6] = planetEphemeris(Julian2,'Sun','Pluto','432t','km');
434
435 % Calculating the Average, Minimum and Maximum Velocity of Each Planet
436 AB2_vel_avg = mean(u_AB2_vel,2);
437 AB2_vel_max = max(u_AB2_vel,[],2);
438 AB2_vel_min = min(u_AB2_vel,[],2);
439
440 % Assigning Variable Names
441 Planetary_Bodies = {'Jupiter';'Saturn';'Uranus';'Neptune';'Pluto'};
442 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)];
443 JulianDate_Position_AB2 = 10^3 * [r2;r3;r4;r5;r6];
444 % 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)]));
445 % JulianDate_Position = 10^3 * [norm(r2);norm(r3);norm(r4);norm(r5);norm(r6)];
446 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)])];
447 JulianDate_Final_V_AB2 = 10^3 * [norm(v2);norm(v3);norm(v4);norm(v5);norm(v6)];
448 CalculatedMeanVelocity_AB2 = [AB2_vel_avg(2); AB2_vel_avg(3); ...
        AB2_vel_avg(4); AB2_vel_avg(5); AB2_vel_avg(6)];
449 KnownMeanVelocity_AB2 = 10^3 * [Jupiter_Mean; Saturn_Mean; Uranus_Mean; ...
        Neptune_Mean; Pluto_Mean];
450
451 % Defining Tables
452 Table1 = table(Planetary_Bodies, Calculated_Position_AB2, ...
        JulianDate_Position_AB2);
453 Table2 = table(Planetary_Bodies, Calculated_Final_V_AB2, ...
        JulianDate_Final_V_AB2, CalculatedMeanVelocity_AB2, KnownMeanVelocity_AB2);
454
455 % Get the table in string form.
456 TString = evalc('disp(Table1)');
457 % Use TeX Markup for bold formatting and underscores.
458 TString = strrep(TString,'<strong>','\bf');
459 TString = strrep(TString,'</strong>','\rm');
460 TString = strrep(TString,'_','\_');
461 % Get a fixed-width font.
462 FixedWidth = get(0,'FixedWidthFontName');
463 % Output the table using the annotation command.
464 figure(100)
465 annotation(gcf,'Textbox','String',TString,'Interpreter','Tex','FontName',...
466         FixedWidth,'Units','Normalized','Position',[0 0 1 1]);
467
468 % Get the table in string form.
469 TString = evalc('disp(Table2)');
470 % Use TeX Markup for bold formatting and underscores.
471 TString = strrep(TString,'<strong>','\bf');
472 TString = strrep(TString,'</strong>','\rm');
473 TString = strrep(TString,'_','\_');
474 % Get a fixed-width font.
475 FixedWidth = get(0,'FixedWidthFontName');
476 % Output the table using the annotation command.
477 figure(200)
478 annotation(gcf,'Textbox','String',TString,'Interpreter','Tex','FontName',...

```

```

479     FixedWidth,'Units','Normalized','Position',[0 0 1 1]);
480
481 % Plotting the Orbits Estimated
482 figure(1)
483 scatter4(0,0,0,40,'filled')
484 grid on
485 hold on
486 plot3(u_AB2_approx(7,:), u_AB2_approx(8,:), u_AB2_approx(9,:), 'LineWidth', 2)
487 scatter4(u_AB2_approx(7,1), u_AB2_approx(8,1), u_AB2_approx(9,1),40,'filled')
488 plot3(u_AB2_approx(13,:), u_AB2_approx(14,:), ...
        u_AB2_approx(15,:), 'LineWidth', 2)
489 scatter4(u_AB2_approx(13,1), u_AB2_approx(14,1), u_AB2_approx(15,1),40,'filled')
490 plot3(u_AB2_approx(19,:), u_AB2_approx(20,:), ...
        u_AB2_approx(21,:), 'LineWidth', 2)
491 scatter4(u_AB2_approx(19,1), u_AB2_approx(20,1), u_AB2_approx(21,1),40,'filled')
492 plot3(u_AB2_approx(25,:), u_AB2_approx(26,:), ...
        u_AB2_approx(27,:), 'LineWidth', 2)
493 scatter4(u_AB2_approx(25,1), u_AB2_approx(26,1), u_AB2_approx(27,1),40,'filled')
494 plot3(u_AB2_approx(31,:), u_AB2_approx(32,:), ...
        u_AB2_approx(33,:), 'LineWidth', 2)
495 scatter4(u_AB2_approx(31,1), u_AB2_approx(32,1), u_AB2_approx(33,1),40,'filled')
496 title('Outer Solar System ( Adams Bashforth Method)')
497 legend('Sun','Jupiter Orbit','Jupiter','Saturn Orbit','Saturn','Uranus ...
        Orbit','Uranus','Neptune Orbit','Neptune','Pluto Orbit','Pluto')
498
499
500
501 %%%%% RK4 Numerical Analysis %%%%%
502
503 tk = 0; %starting time
504
505 tic
506
507 %initialize iterates for Huens
508 u_rk_k = u0;
509
510 %initialize vector that stores approximate soln at various times
511 u_rk_approx = zeros( 36,T/dt );
512
513 %Initialize vector that stores the magnitude of velocities at T
514 u_rk_vel = zeros(6,T/dt);
515
516 %advance to final time T
517 for i = 1 : T/dt
518
519     %advance with RK4 for 1st time step
520     y1 = f(u_rk_k);

```



```

521     y2 = f(u_rk_k + 1/2*dt*y1);
522     y3 = f(u_rk_k + (1/2)*dt*y2);
523     y4 = f(u_rk_k + dt*y3);
524     u_rk_approx(:,i) = u_rk_k + 1/6*dt*(y1 + 2*y2 + 2*y3 + y4);
525
526     %update iterates
527     tk = tk + dt;
528     u_rk_k = u_rk_approx(:,i);
529
530     %Calculating and storing the velocities of each body at T
531     u_rk_vel(1,i) = norm(norm(u_rk_approx(4:6,i)));
532     u_rk_vel(2,i) = norm(norm(u_rk_approx(10:12,i)));
533     u_rk_vel(3,i) = norm(norm(u_rk_approx(16:18,i)));
534     u_rk_vel(4,i) = norm(norm(u_rk_approx(22:24,i)));
535     u_rk_vel(5,i) = norm(norm(u_rk_approx(28:30,i)));
536     u_rk_vel(6,i) = norm(norm(u_rk_approx(34:36,i)));
537
538 end
539
540 t_r = toc
541
542 % Gathering N-Body Information for 249 years in future from Jan 1, 2011 (T)
543 Julian2 = juliandate(2260,12,31);
544
545 % Position and Velocity Vectors at future position
546 [r1, v1] = planetEphemeris(Julian2,'Sun', 'Sun', '432t', 'km');
547 [r2, v2] = planetEphemeris(Julian2,'Sun','Jupiter','432t','km');
548 [r3, v3] = planetEphemeris(Julian2,'Sun','Saturn','432t','km');
549 [r4, v4] = planetEphemeris(Julian2,'Sun','Uranus','432t','km');
550 [r5, v5] = planetEphemeris(Julian2,'Sun','Neptune','432t','km');
551 [r6, v6] = planetEphemeris(Julian2,'Sun','Pluto','432t','km');
552
553 % Calculating the Average Velocity of Each Planet
554 rk_vel_avg = mean(u_rk_vel,2);
555 rk_vel_max = max(u_rk_vel,[],2);
556 rk_vel_min = min(u_rk_vel,[],2);
557
558 % Assigning Variable Names
559 Planetary_Bodies = {'Jupiter';'Saturn';'Uranus';'Neptune';'Pluto'};
560 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)];
561 JulianDate_Position_RK4 = 10^3 * [r2;r3;r4;r5;r6];

```

```

562 % 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)])];
563 % JulianDate_Position = 10^3 * [norm(r2);norm(r3);norm(r4);norm(r5);norm(r6)];
564 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)])];
565 JulianDate_Final_V_RK4 = 10^3 * [norm(v2);norm(v3);norm(v4);norm(v5);norm(v6)];
566 CalculatedMeanVelocity_RK4 = [rk_vel_avg(2); rk_vel_avg(3); rk_vel_avg(4); ...
    rk_vel_avg(5); rk_vel_avg(6)];
567 KnownMeanVelocity_RK4 = 10^3 * [Jupiter_Mean; Saturn_Mean; Uranus_Mean; ...
    Neptune_Mean; Pluto_Mean];
568
569 % Defining Tables
570 Table1 = table(Planetary_Bodies, Calculated_Position_RK4, ...
    JulianDate_Position_RK4);
571 Table2 = table(Planetary_Bodies, Calculated_Final_V_RK4, ...
    JulianDate_Final_V_RK4, CalculatedMeanVelocity_RK4, KnownMeanVelocity_RK4);
572
573 % Get the table in string form.
574 TString = evalc('disp(Table1)');
575 % Use TeX Markup for bold formatting and underscores.
576 TString = strrep(TString, '<strong>', '\bf');
577 TString = strrep(TString, '</strong>', '\rm');
578 TString = strrep(TString, '_', '\_');
579 % Get a fixed-width font.
580 FixedWidth = get(0, 'FixedWidthFontName');
581 % Output the table using the annotation command.
582 figure(300)
583 annotation(gcf, 'Textbox', 'String', TString, 'Interpreter', 'Tex', 'FontName', ...
    FixedWidth, 'Units', 'Normalized', 'Position', [0 0 1 1]);
584
585
586 % Get the table in string form.
587 TString = evalc('disp(Table2)');
588 % Use TeX Markup for bold formatting and underscores.
589 TString = strrep(TString, '<strong>', '\bf');
590 TString = strrep(TString, '</strong>', '\rm');
591 TString = strrep(TString, '_', '\_');
592 % Get a fixed-width font.
593 FixedWidth = get(0, 'FixedWidthFontName');
594 % Output the table using the annotation command.

```

```

595 figure(400)
596 annotation(gcf,'Textbox','String',TString,'Interpreter','Tex','FontName',...
597     FixedWidth,'Units','Normalized','Position',[0 0 1 1]);
598
599 % Plotting the Orbits Estimated
600 figure(2)
601 scatter4(0,0,0,40,'filled')
602 grid on
603 hold on
604 plot3(u_rk_approx(7,:), u_rk_approx(8,:), u_rk_approx(9,:), 'LineWidth', 2)
605 scatter4(u_rk_approx(7,1), u_rk_approx(8,1), u_rk_approx(9,1),40,'filled')
606 plot3(u_rk_approx(13,:), u_rk_approx(14,:), u_rk_approx(15,:), 'LineWidth', 2)
607 scatter4(u_rk_approx(13,1), u_rk_approx(14,1), u_rk_approx(15,1),40,'filled')
608 plot3(u_rk_approx(19,:), u_rk_approx(20,:), u_rk_approx(21,:), 'LineWidth', 2)
609 scatter4(u_rk_approx(19,1), u_rk_approx(20,1), u_rk_approx(21,1),40,'filled')
610 plot3(u_rk_approx(25,:), u_rk_approx(26,:), u_rk_approx(27,:), 'LineWidth', 2)
611 scatter4(u_rk_approx(25,1), u_rk_approx(26,1), u_rk_approx(27,1),40,'filled')
612 plot3(u_rk_approx(31,:), u_rk_approx(32,:), u_rk_approx(33,:), 'LineWidth', 2)
613 scatter4(u_rk_approx(31,1), u_rk_approx(32,1), u_rk_approx(33,1),40,'filled')
614 title('Outer Solar System (Runge-Kutta 4 Method)')
615 legend('Sun','Jupiter Orbit','Jupiter','Saturn Orbit','Saturn','Uranus ...
        Orbit','Uranus','Neptune Orbit','Neptune','Pluto Orbit','Pluto')
616
617
618 %% Convergence Test
619
620 % AB2
621 %--simulation params
622 % 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
623 dtvect = 60*60*24 * [781.25, 156.25, 31.25, 6.25, 1.25, 0.25]; % For Outer Orbits
624 %--
625
626 %initialize vector that stores approximate soln at T for various dt
627 u_AB2_keep = zeros( 36,length( dtvect ) );
628 tic
629 %advance in time
630 for j = 1 : length(dtvect)
631
632     %current dt
633     dt = dtvect(j);
634
635     tk = t0; %initialize time iterate
636
637     %initialize iterates for various methods
638     u_AB2_k = u0;
639     u_AB2_km1 = u0;

```

```

640
641
642 %run to final time T
643 for jj = 1 : T/dt
644
645     %AB2
646     if jj < 2
647
648         %advance with Heun's for 1st time step
649         u_AB2_kp1 = (u_AB2_k + 1/2*dt*(f(u_AB2_k) + f(u_AB2_k + ...
            dt*f(u_AB2_k))));
650
651     else
652         u_AB2_kp1 = u_AB2_k + 1/2*dt*(-1*f(u_AB2_km1) + 3*f(u_AB2_k));
653
654     end
655
656     %update iterates
657     u_AB2_km1 = u_AB2_k;
658     u_AB2_k = u_AB2_kp1;
659
660     %update time
661     tk = tk + dt;
662
663
664 end
665
666 %store soln at T
667 u_AB2_keep( :,j ) = u_AB2_kp1;
668
669 end
670
671 %compute difference between solution at smallest dt and the other dts
672
673 %initialize vector
674 u_AB2_diff = zeros( length( dtvect )-1,1 );
675
676 for j = 1 : length( dtvect )-1
677
678     %AB2
679     u_AB2_diff(j) = norm( u_AB2_keep(:,j) - u_AB2_keep(:,end) )/ ...
680         norm( u_AB2_keep(:,end) );
681
682 end
683 toc
684 figure(50)
685 %AB2

```

```

686 loglog( dtvect(1:end-1), u_AB2_diff, '.-', 'markersize', 20, 'linewidth', 2 )
687
688 set( gca, 'fontsize', 16, 'ticklabelinterpreter', 'latex' )
689
690 leg = legend( 'AB2' );
691 set( leg, 'fontsize', 12, 'interpreter', 'latex', 'location', 'southeast' )
692 xlabel('$\Delta t$', 'fontsize', 12, 'interpreter', 'latex')
693 title('$\frac{||u_{\Delta t} - u_{2.5 \times 10^{-4}}||}{|| \dots$
        $u_{2.5 \times 10^{-4}} ||}$', 'fontsize', 28, 'interpreter', 'latex' )
694 set(gcf, 'PaperPositionMode', 'manual')
695 set(gcf, 'Color', [1 1 1])
696 set(gca, 'Color', [1 1 1])
697 set(gcf, 'PaperUnits', 'centimeters')
698 set(gcf, 'PaperSize', [15 15])
699 set(gcf, 'Units', 'centimeters' )
700 set(gcf, 'Position', [0 0 15 15])
701 set(gcf, 'PaperPosition', [0 0 15 15])
702
703 svnm = 'ConvergencePlots1';
704 print( '-dpng', svnm, '-r300' )
705
706
707 % RK4
708 %--simulation params
709 % 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
710 dtvect = 60*60*24 * [781.25, 156.25, 31.25, 6.25, 1.25, 0.25]; % For Outer Orbits
711 %--
712
713 %initialize vector that stores approximate soln at T for various dt
714 u_r_keep = zeros( 36,length( dtvect ) );
715 tic
716 %advance in time
717 for j = 1 : length(dtvect)
718
719     %current dt
720     dt = dtvect(j);
721
722     tk = t0; %initialize time iterate
723
724     %initialize iterates for various methods
725     u_r_k = u0;
726
727
728     %run to final time T
729     for jj = 1 : T/dt
730

```

```

731         %RK4
732         y1 = f(u_r_k);
733         y2 = f(u_r_k + 1/2*dt*y1);
734         y3 = f(u_r_k + (1/2)*dt*y2);
735         y4 = f(u_r_k + dt*y3);
736         u_r_kp1 = u_r_k + 1/6*dt*(y1 + 2*y2 + 2*y3 + y4);
737
738         %update iterates
739         u_r_k = u_r_kp1;
740
741         %update time
742         tk = tk + dt;
743
744
745     end
746
747     %store soln at T
748     u_r_keep( :,j ) = u_r_kp1;
749
750 end
751
752 %compute difference between solution at smallest dt and the other dts
753
754 %initialize vector
755 u_r_diff = zeros( length( dtvect )-1,1 );
756
757 for j = 1 : length( dtvect )-1
758
759     %RK4
760     u_r_diff(j) = norm( u_r_keep(:,j) - u_r_keep(:,end) )/ ...
761         norm( u_r_keep(:,end) );
762
763 end
764 toc
765 figure(50), hold on
766 %RK4
767 loglog( dtvect(1:end-1), u_r_diff, '.-', 'markersize', 20, 'linewidth', 2 )
768
769 set( gca, 'fontsize', 12, 'ticklabelinterpreter', 'latex' )
770
771 leg = legend( 'AB2', 'RK4' );
772 set( leg, 'fontsize', 16, 'interpreter', 'latex', 'location', 'southeast' )
773 xlabel('$\Delta t$', 'fontsize', 12, 'interpreter', 'latex')
774 title('$\frac{||u_{\Delta t} - u_{2.5\times 10^{-4}}||}{|| ...$' , 'fontsize', 28, 'interpreter', 'latex' )
775 set(gcf, 'PaperPositionMode', 'manual')
776 set(gcf, 'Color', [1 1 1])

```

```

777 set(gca, 'Color', [1 1 1])
778 set(gcf, 'PaperUnits', 'centimeters')
779 set(gcf, 'PaperSize', [15 15])
780 set(gcf, 'Units', 'centimeters' )
781 set(gcf, 'Position', [0 0 15 15])
782 set(gcf, 'PaperPosition', [0 0 15 15])
783
784 svnm = 'ConvergencePlots2';
785 print( '-dpng', svnm, '-r300' )
786
787
788 % Euler
789 %--simulation params
790 % dtvect = [60*60*24*6.25, 60*60*24*1.25, 60*60*24*0.25, 60*60*24*0.05, ...
791           60*60*24*0.01, 60*60*24*0.002]; % For Inner Orbits
791 dtvect = 60*60*24 * [781.25, 156.25, 31.25, 6.25, 1.25, 0.25]; % For Outer Orbits
792 %--
793
794 %initialize vector that stores approximate soln at T for various dt
795 u_eu_approx = zeros( 36,length( dtvect ) );
796
797 %advance in time
798 for j = 1 : length(dtvect)
799
800     %current dt
801     dt = dtvect(j);
802
803     tk = t0; %initialize time iterate
804
805     %initialize iterates for various methods
806     u_eu_k = u0;
807
808
809     %run to final time T
810     for jj = 1 : T/dt
811
812         %Euler
813         u_eu_kp1 = u_eu_k + dt*f(u_eu_k);
814
815         %update iterates
816         u_eu_k = u_eu_kp1;
817
818         %update time
819         tk = tk + dt;
820
821
822     end

```

```

823
824     %store soln at T
825     u_eu_approx( :,j ) = u_eu_kp1;
826
827 end
828
829 %compute difference between solution at smallest dt and the other dts
830
831 %initialize vector
832 u_eu_diff = zeros( length( dtvect )-1,1 );
833
834 for j = 1 : length( dtvect )-1
835
836     %Euler
837     u_eu_diff(j) = norm( u_eu_approx(:,j) - u_eu_approx(:,end) )/ ...
838         norm( u_eu_approx(:,end) );
839
840 end
841
842 figure(50), hold on
843 %Euler
844 loglog( dtvect(1:end-1), u_eu_diff, '.-', 'markersize', 20, 'linewidth', 2 )
845
846 set( gca, 'fontsize', 16, 'ticklabelinterpreter', 'latex' )
847
848 leg = legend( 'AB2', 'RK4', 'Euler' );
849 set( leg, 'fontsize', 12, 'interpreter', 'latex', 'location', 'southeast' )
850 xlabel('$\Delta t$', 'fontsize', 12, 'interpreter', 'latex')
851 title('$\frac{||u_{\Delta t} - u_{2.5 \times 10^{-4}}||}{|| \dots$'
852     $u_{2.5 \times 10^{-4}} ||}$', 'fontsize', 28, 'interpreter', 'latex' )
853
852 set(gcf, 'PaperPositionMode', 'manual')
853 set(gcf, 'Color', [1 1 1])
854 set(gcf, 'Color', [1 1 1])
855 set(gcf, 'PaperUnits', 'centimeters')
856 set(gcf, 'PaperSize', [15 15])
857 set(gcf, 'Units', 'centimeters' )
858 set(gcf, 'Position', [0 0 15 15])
859 set(gcf, 'PaperPosition', [0 0 15 15])
860
861 svnm = 'error_q2_Euler';
862 print( '-dpng', svnm, '-r200' )
863
864
865 % Heun's
866 %--simulation params
867 % dtvect = [60*60*24*6.25, 60*60*24*1.25, 60*60*24*0.25, 60*60*24*0.05, ...
868     60*60*24*0.01, 60*60*24*0.002]; % For Inner Orbits

```



```

868 dtvect = 60*60*24 * [781.25, 156.25, 31.25, 6.25, 1.25, 0.25]; % For Outer Orbits
869 %--
870
871 %initialize vector that stores approximate soln at T for various dt
872 u_h_keep = zeros( 36,length( dtvect ) );
873
874 %advance in time
875 for j = 1 : length(dtvect)
876
877     %current dt
878     dt = dtvect(j);
879
880     tk = t0; %initialize time iterate
881
882     %initialize iterates for various methods
883     u_h_k = u0;
884
885
886     %run to final time T
887     for jj = 1 : T/dt
888
889         %Huens
890         u_h_kp1 = u_h_k + 1/2*dt*(f(u_h_k) + f(u_h_k + dt*f(u_h_k)));
891
892         %update iterates
893         u_h_k = u_h_kp1;
894
895         %update time
896         tk = tk + dt;
897
898
899     end
900
901     %store soln at T
902     u_h_keep( :,j ) = u_h_kp1;
903
904 end
905
906 %compute difference between solution at smallest dt and the other dts
907
908 %initialize vector
909 u_h_diff = zeros( length( dtvect )-1,1 );
910
911 for j = 1 : length( dtvect )-1
912
913     %Huens
914     u_h_diff(j) = norm( u_h_keep(:,j) - u_h_keep(:,end) )/ ...

```

```

915         norm( u_h_keep(:,end) );
916
917     end
918
919     figure(50), hold on
920     %Heun's
921     loglog( dtvect(1:end-1), u_h_diff, '.-', 'markersize', 20, 'linewidth', 2 )
922
923     set( gca, 'fontsize', 16, 'ticklabelinterpreter', 'latex' )
924
925     leg = legend( 'AB2', 'RK4', 'Euler', 'Huens' );
926     set( leg, 'fontsize', 12, 'interpreter', 'latex', 'location', 'southeast' )
927     xlabel('$\Delta t$', 'fontsize', 12, 'interpreter', 'latex')
928     title('$\frac{||u_{\Delta t} - u_{2.5 \times 10^{-4}}||}{||...
          u_{2.5 \times 10^{-4}} ||}$', 'fontsize', 28, 'interpreter', 'latex' )
929     set(gcf, 'PaperPositionMode', 'manual')
930     set(gcf, 'Color', [1 1 1])
931     set(gca, 'Color', [1 1 1])
932     set(gcf, 'PaperUnits', 'centimeters')
933     set(gcf, 'PaperSize', [15 15])
934     set(gcf, 'Units', 'centimeters' )
935     set(gcf, 'Position', [0 0 15 15])
936     set(gcf, 'PaperPosition', [0 0 15 15])
937
938     svnm = 'error_q2_Huens';
939     print( '-dpng', svnm, '-r200' )
940
941
942     %% Error Plots
943
944     % % Gathering N-Body Information for 687 days in future from Jan 1, 2011 (T)
945     % Julian2 = juliandate(2012,11,19);
946
947     % Gathering N-Body Information for 249 years in future from Jan 1, 2011 (T)
948     Julian2 = juliandate(2260,12,31);
949
950     % % Position and Velocity Vectors at future position
951     % [r1, v1] = planetEphemeris(Julian2,'Sun', 'Sun', '432t', 'km');
952     % [r2, v2] = planetEphemeris(Julian2,'Sun','Mercury','432t','km');
953     % [r3, v3] = planetEphemeris(Julian2,'Sun','Venus','432t','km');
954     % [r4, v4] = planetEphemeris(Julian2,'Sun','Earth','432t','km');
955     % [r5, v5] = planetEphemeris(Julian2,'Sun','Mars','432t','km');
956     % [r6, v6] = planetEphemeris(Julian2,'Sun','Moon','432t','km');
957
958
959     % Position and Velocity Vectors at future position
960     [r1, v1] = planetEphemeris(Julian2,'Sun', 'Sun', '432t', 'km');

```

```

961 [r2, v2] = planetEphemeris(Julian2, 'Sun', 'Jupiter', '432t', 'km');
962 [r3, v3] = planetEphemeris(Julian2, 'Sun', 'Saturn', '432t', 'km');
963 [r4, v4] = planetEphemeris(Julian2, 'Sun', 'Uranus', '432t', 'km');
964 [r5, v5] = planetEphemeris(Julian2, 'Sun', 'Neptune', '432t', 'km');
965 [r6, v6] = planetEphemeris(Julian2, 'Sun', 'Pluto', '432t', 'km');
966
967 [u_real] = 10^3 * [r1'; v1'; r2'; v2'; r3'; v3'; r4'; v4'; r5'; v5'; r6'; v6'];
968
969 % AB2
970 %--simulation params
971 % dtvect = [60*60*24*6.25, 60*60*24*1.25, 60*60*24*0.25, 60*60*24*0.05, ...
972             60*60*24*0.01, 60*60*24*0.002];
973 dtvect = 60*60*24 * [781.25, 156.25, 31.25, 6.25, 1.25, 0.25];
974 %--
975 %initialize vector that stores approximate soln at T for various dt
976 u_AB2_keep = zeros( 36,length( dtvect ) );
977 tic
978 %advance in time
979 for j = 1 : length(dtvect)
980
981     %current dt
982     dt = dtvect(j);
983
984     tk = t0; %initialize time iterate
985
986     %initialize iterates for various methods
987     u_AB2_k = u0;
988     u_AB2_km1 = u0;
989
990
991     %run to final time T
992     for jj = 1 : T/dt
993
994         %AB2
995         if jj < 2
996
997             %advance with Heun's for 1st time step
998             u_AB2_kp1 = (u_AB2_k + 1/2*dt*(f(u_AB2_k) + f(u_AB2_k + ...
999                 dt*f(u_AB2_k))));
1000
1001         else
1002             u_AB2_kp1 = u_AB2_k + 1/2*dt*(-1*f(u_AB2_km1) + 3*f(u_AB2_k));
1003
1004         end
1005
1006         %update iterates

```

```

1006         u_AB2_km1 = u_AB2_k;
1007         u_AB2_k = u_AB2_kp1;
1008
1009         %update time
1010         tk = tk + dt;
1011
1012
1013     end
1014
1015     %store soln at T
1016     u_AB2_keep( :,j ) = u_AB2_kp1;
1017
1018 end
1019
1020 %compute difference between solution at smallest dt and the other dts
1021
1022 %initialize vector
1023 u_AB2_error = zeros( length( dtvect ),1 );
1024
1025 for j = 1 : length( dtvect )
1026
1027     %AB2
1028     u_AB2_error(j) = norm( u_AB2_keep(:,j) - u_real(:) )/ ...
1029         norm( u_real(:) );
1030
1031 end
1032 toc
1033 figure(50)
1034 %AB2
1035 loglog( dtvect(1:end), u_AB2_error, '-.', 'markersize', 20, 'linewidth', 2 )
1036
1037 set( gca, 'fontsize', 16, 'ticklabelinterpreter', 'latex' )
1038
1039 leg = legend( 'AB2' );
1040 set( leg, 'fontsize', 12, 'interpreter', 'latex', 'location', 'southeast' )
1041 xlabel('$\Delta t$', 'fontsize', 12, 'interpreter', 'latex')
1042 ylabel('Error %', 'fontsize', 12, 'interpreter', 'latex')
1043 title('Error Plot', 'fontsize', 28, 'interpreter', 'latex' )
1044 set(gcf, 'PaperPositionMode', 'manual')
1045 set(gcf, 'Color', [1 1 1])
1046 set(gca, 'Color', [1 1 1])
1047 set(gcf, 'PaperUnits', 'centimeters')
1048 set(gcf, 'PaperSize', [15 15])
1049 set(gcf, 'Units', 'centimeters' )
1050 set(gcf, 'Position', [0 0 15 15])
1051 set(gcf, 'PaperPosition', [0 0 15 15])
1052

```

```

1053 svnm = 'ErrorPlots1';
1054 print( '-dpng', svnm, '-r300' )
1055
1056
1057
1058 % RK4
1059 %--simulation params
1060 % 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];
1061 dtvect = 60*60*24 * [781.25, 156.25, 31.25, 6.25, 1.25, 0.25];
1062 %--
1063
1064 %initialize vector that stores approximate soln at T for various dt
1065 u_r_keep = zeros( 36,length( dtvect ) );
1066 tic
1067 %advance in time
1068 for j = 1 : length(dtvect)
1069
1070     %current dt
1071     dt = dtvect(j);
1072
1073     tk = t0; %initialize time iterate
1074
1075     %initialize iterates for various methods
1076     u_r_k = u0;
1077
1078
1079     %run to final time T
1080     for jj = 1 : T/dt
1081
1082         %RK4
1083         y1 = f(u_r_k);
1084         y2 = f(u_r_k + 1/2*dt*y1);
1085         y3 = f(u_r_k + (1/2)*dt*y2);
1086         y4 = f(u_r_k + dt*y3);
1087         u_r_kp1 = u_r_k + 1/6*dt*(y1 + 2*y2 + 2*y3 + y4);
1088
1089         %update iterates
1090         u_r_k = u_r_kp1;
1091
1092         %update time
1093         tk = tk + dt;
1094
1095
1096     end
1097
1098     %store soln at T

```

```

1109     u_r_keep( :,j ) = u_r_kp1;
1100
1101 end
1102
1103 %compute the error
1104
1105 %initialize vector
1106 u_r_error = zeros( length( dtvect ),1 );
1107
1108 for j = 1 : length( dtvect )
1109
1110     %RK4
1111     u_r_error(j) = (norm( u_r_keep(:,j) - u_real(:) )/ ...
1112         norm( u_real(:) ));
1113
1114 end
1115 toc
1116 figure(50), hold on
1117 %RK4
1118 loglog( dtvect(1:end), u_r_error, '.-', 'markersize', 20, 'linewidth', 2 )
1119
1120 set( gca, 'fontsize', 12, 'ticklabelinterpreter', 'latex' )
1121
1122 leg = legend( 'AB2', 'RK4' );
1123 set( leg, 'fontsize', 16, 'interpreter', 'latex', 'location', 'southeast' )
1124 xlabel('$\Delta t$', 'fontsize', 12, 'interpreter', 'latex')
1125 ylabel('Error %', 'fontsize', 12, 'interpreter', 'latex')
1126 title('Error Plot', 'fontsize', 28, 'interpreter', 'latex' )
1127 set(gcf, 'PaperPositionMode', 'manual')
1128 set(gcf, 'Color', [1 1 1])
1129 set(gca, 'Color', [1 1 1])
1130 set(gcf, 'PaperUnits', 'centimeters')
1131 set(gcf, 'PaperSize', [15 15])
1132 set(gcf, 'Units', 'centimeters' )
1133 set(gcf, 'Position', [0 0 15 15])
1134 set(gcf, 'PaperPosition', [0 0 15 15])
1135
1136 svnm = 'ErrorPlots2';
1137 print( '-dpng', svnm, '-r300' )

```