An Object Oriented Model of the Solar System with Explorations in Gravitational Assists

Austin Sagan

May 11, 2015

1 Background

While traversing the solar system, space probes approach planets to capture images, take data and use the gravity to alter their speed and trajectory, an effect known as the gravitational slingshot[2]. This effect has been used to assist several space probes, such as Mariner 10, Voyager 1 and 2 and many others. The Galileo spacecraft and the Ulysses probe both used gravitational slingshots when there were problems with the original trajectories, and there was not enough fuel to completely correct it.

Consider the scenario shown in figure 1. There is a planet initially moving with velocity u_i having mass M, and a space probe with velocity v_i with mass m. The space probe approaches the planet and gets turned around by gravity. Let the final velocity of the planet and space probe be u_f and v_f respectively. Conserving momentum and energy yields the following equations [1]

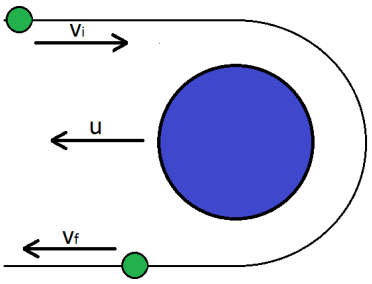

Figure 1:

$$\frac{1}{2}mv_i^2 + \frac{1}{2}Mu_i^2 = \frac{1}{2}mv_f^2 + \frac{1}{2}Mu_f^2$$

$$-mv_i + Mu_i = mv_f + Mu_f$$

Eliminating the variable u_f and solving for v_f yields

$$v_f = \frac{(1 - \frac{m}{M})v_i + 2u_i}{1 + \frac{m}{M}}$$

and because the mass of the space probe is negligible in comparison to the planet, we are left with

$$v_f = v_i + 2u_i$$

In practice, however, it is not easy to replicate this exact scenario.

2 Program Design

The heart of the program in the "body.m" class. Each body in the simulation is an instance of the body class. Here, all information about the object is stored. The properties are as follows:

x0	3d vector containing initial position in m
v0	3d vector containing initial velocity in m/s
x	3 x n matrix containing all positions. x list(:,i) gives the position at time i in m.
v	3 x n matrix containing all velocities. v list(:,i) gives the velocity at time i in m/s.
name	name of object, e.g. "Earth", "Moon"
id	Integer assigned to object.
mass	Mass of object in kg
i	number of current iteration.

The ID numbers for objects are consistent through out all programs and are as follows:

Object	ID number
Sun	1
Mercury	2
Venus	3
Earth	4
Mars	5
Jupiter	6
Saturn	7
Uranus	8
Neptune	9
Moon	10
Voyager	11

Additionally, the class has several methods. The most important one being "update". This method is called to make to object progress one time step in time. This function calls one of the ODE solving functions and returns the object one time step forward in time. It also has a function "init" to initialize all arrays and a function to get the acceleration that all other objects have on it.

The program "main.m" takes a list of body objects that have not been initialized and have no x or v data yet, iterates through all times and return a list of objects that have the position and velocity for the given time span. This program is run, the results are saved and then analyzed with a suite of programs.

Because of the hashtables used while creating the lists of objects, the objects are not in

any order. The program "get_object.m" takes the list and a desired object ID and returns that object.

There are 4 scripts that generate the list of objects and initial values that then gets given to the main program. The scripts "create_sling_shot1.m" and "create_sling_shot2.m" create the gravitational assist example in section 3 and "make_solar_system.m" create all planets in the solar system and our moon starting on January 5, 1915 using initial conditions from JPL's Horizons database.

The programs "compare.m", "distance_between_objects.m", "period_of_moon.m", "plot_objects.m", "plot_plot_total_energy.m", and "gravitational_assist.m" are all designed to analyze the output of the main program.

The program compare.m compares the results from a simulation and the data from JPL's Horizons database for a specific object. It takes the output from "make_solar_system_JPL.m" and from "main.m", the desired object ID and the time step of the simulation as inputs. It calculates the error, plots it and returns it.

When studying the Moon traveling around the Earth, it is useful and interesting to look at the distance between them. That is what "distance_between_objects.m" was designed to do. It takes the output from the main program, the ID number of the two objects, and the time step of the simulation as inputs. It returns the list of the distance over time and plots it. We are also interested in the orbital period of the moon. The program "period_of_moon.m" takes the output from main and returns the period of the planet.

We want to create a way to easily visualize the data. The program "plot_objects.m" takes the output from the main program, a file to write to, a time step, and the end time and creates a video of the planets.

Because energy conservation is a concern, the program "plot_total_energy" was created as an easy test of this. It take the output from main and returns the energy over time and plots it. Ideally, this is a strait line.

Finally, the program "gravitational_assist" is a script to call previously mentioned functions to simulate a space probe flying by a planet, plots the distance between them, the space probes velocity and the total energy over time.

3 Investigations

The program was used initially to test its reliability by simulating the entire solar system and comparing the results with those acquired from JPL's Horizons database. For each case, the simulation was run for a total to 10,000 days. The figure 2 shows the relative error in the position of the earth as a function of time for Runge Kutta with time steps of 1 and 0.1 days and for Euler Cromer with a time step of 1 day. See the supplementary material for video output of the simulation for each of these cases. As the graph indicates, the Euler Cromer method provides a good estimate while in the Runge Kutta method error accumulates and the planets leave the system. The smaller the time step is for the Euler Cromer simulations, the smaller the oscillations in the error are. However, the error seems to still be growing lin-

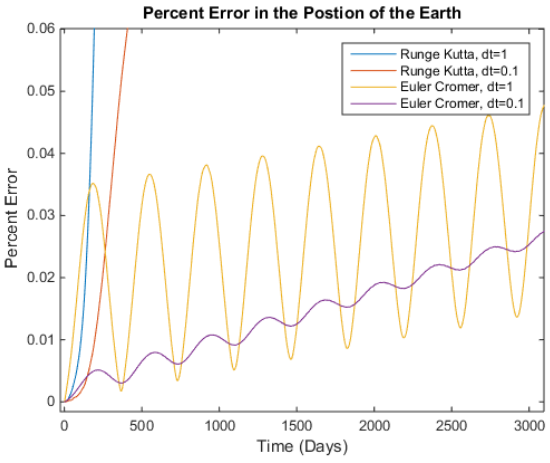

Figure 2:

early. This would indicate that there are

other relevant forces that are not being accounted for in this simulation. This may have to do with the spin of the planets, which is not in the simulation. See the videos "solar_system_RK1.avi", "solar_system_RK0dot1.avi", "solar_system_EC1.avi", "solar_system_EC0dot1.avi" and "solar_system_JPL.avi" for the video output form Runge Kutta with time step of 1, Runge Kutta with time step of 0.1, Euler Cromer with time step of 1, Euler Cromer with time step of 0.1, and from data from the Horizons database.

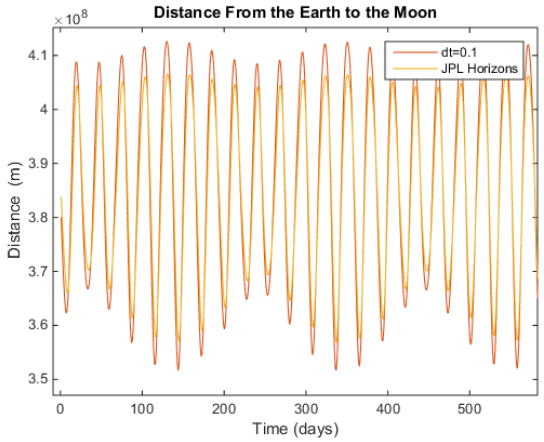

Figure 3:

The distance between the earth and the moon and the moon's orbital period were also measured. The accepted value oscillates between $3.63 * 10^8$ and $4.05 * 10^8$ meters[3]. These numbers where obtained by measuring the time it takes a laser to go from an observatory on Earth to the moon where it reflects off of the retroreflectors placed on the moon as part of the Apollo 11 mission [4]. Figure 3 shows a plot of the distance from the Earth to the moon for Euler Cromer with a time step of 0.1 days and the data obtained from Horizons. The shape of the curve is identical, however the simulation is still not measuring the amplitudes of this completely accurately. See "moon_RK1.avi", "moon_RK0dot1.avi", "moon_EC1.avi", "moon_EC0dot1.avi" and "moon_JPL.avi" for the video output form Runge Kutta with time step of 1, Runge Kutta with time step of 0.1, Euler Cromer with time step of 1, Euler Cromer with time step of 0.1, and from data from the Horizons database.

Simulation	Minimum (m)	Maximum (m)	Mean (m)
Euler Cromer with time step of 1 day	3.1317	4.8536	4.0501
Euler Cromer with time step of 0.1 days	3.5165	4.1280	3.8566
JPL's Horizons Database	3.5684	4.067	3.8500
Measurements	3.63	4.05	3.844

The period of the Moon's orbit was measured by calculating the distance between successive peaks using the program "period_of_orbit.m". The following table shows results from a few simulations

Simulation	Period (days)
Euler Cromer with time step of 1 day	28.6871
Euler Cromer with time step of 0.1 days	27.3637
JPL Horizons Database	27.3211

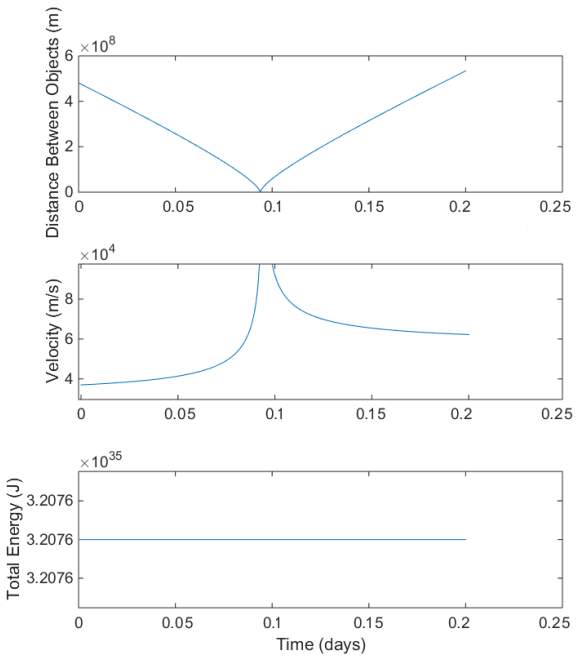

Figure 4:

Here we see that the simulation was able to accurately calculate the orbital speed of the moon with a reasonable time step.

The next step was to investigate gravitational assists. We attempt to replicate the scenario described in part 1 using Jupiter and Voyager 1. See the video 'assist1.avi' in the supplementary materials. Voyager was launched with a speed of 37 km/s towards Jupiter with a speed of 13 km/s in the opposite direction, and the simulation had a time

step of 10^{-6} days. The video shows that the planet was not launched in the exact direction that it was initially traveling in. This was hard to experiment with because of the very small time step needed for when the two objects where very close. This could have been made better with an adaptive step size routine. As seen in figure 4, the final speed of Voyager was 62.2 km/s. In the scenario described in part 1, the final velocity would be 37 km/s + 2*13 km/s= 63 km/s. We would expect these two values to be closer if the satellite where to return in a direction parallel to which it came.

Another scenario to investigate is using gravitational assists to slow down the space craft. Here, we present the same scenario as before with the planet moving in the opposite direction. Figure 5 shows the initial velocity of the probe was 111 km/s, and the final velocity was 95.39 km/s. The expected result was 85 km/s. As seen in the video 'assist2.avi', the space probe is not behaving exactly as described in part 1, possibly explaining the difference.

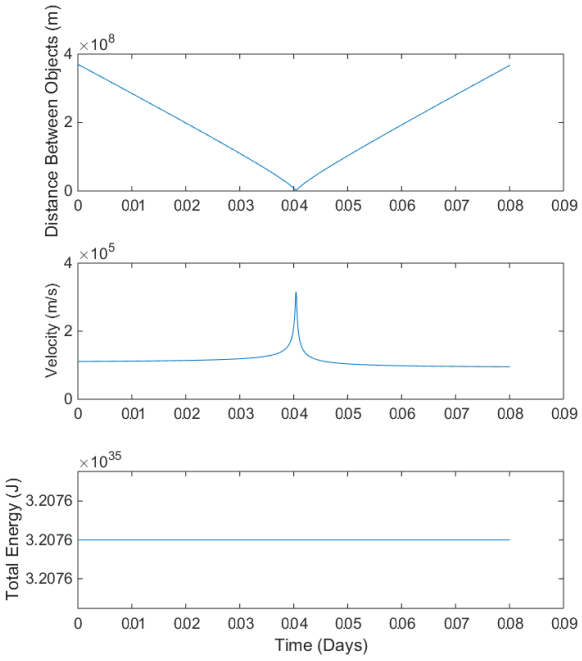

Figure 5:

4 Conclusion

In conclusion, we successfully created ab object oriented model of solar system. This model was used to take some measurements, and for the most part, it's accuracy was confirmed.

However, the way the error from Euler Cromer with a time step of 1 day accumulates linearly and how error from Euler Cromer with a time step of 0.1 days accumulates in the same linear way, just with less oscillations indicated that there might be important forces acting on the system in reality that are not being represented in this model. One way to improve this is to do more research and determine what this might be. Additionally, the ODE solver used was not ideal. An attempt to implement Verlet was unsuccessful.

Additionally, the model was able to be used to experiment with gravitational assists of space probes. The results where as expected from the analytical solutions, showing this model could be used somewhat reliably to replicate other scenarios with gravitational assists. Further exploration would have included replicating the Voyager 1 mission.

References

[1] Gravitational Slingshot. (n.d.). Retrieved May 11, 2015, from http://www.mathpages.com/home/kmath114/kmath114.htm

[2] A Gravity Assist Primer. (n.d.). Retrieved May 11, 2015, from http://www2.jpl.nasa.gov/basics/grav/primer.php

[3] Our Solar System: Overview. (n.d.). Retrieved May 11, 2015, from http://solarsystem.nasa.gov/planets/profile.cfm?Display=Facts&Object=Moon

[4] Apollo 11 Mission. (n.d.). Retrieved May 11, 2015, from http://www.lpi.usra.edu/lunar/missions/apollo/apollo_11/experiments/lrr/