PHYS 905 - Project 3

Terri Poxon-Pearson

March 27, 2017

INSERT ABSTRACT HERE

Contents

1 Introduction

Differential equations are one of the most important tools that physicists have for expressing physical laws. From Maxwell's equations to Einstein's field question in general relativity, differential equations are the mathematical language for almost every physical process. Outside of physics, differential equations can be

used describe systems in finance, ecosystems, and medicine. The ubiquity of these relationships demands that we have efficient and accurate algorithms for solving these problems.

In this project we will focus on one of the earliest differential equations, first expressed by Newton [1]. Newton's Second law is an example of an ordinary differential equation, where ordinary refers to the fact that it is a function of independent variable and its derivatives. Once the differential equation is solved analytically, initial values must be provided to find the exact solution. When we solve these problems numerically, we will provide initial values to begin the solver.

Specifically in this project we will be simulating the motion of the planets in the solar system. We will explore two different Algorithms for solving initial value ODEs: the Euler Algorithm and the Velocity Verlet Algorithm. We will begin by looking at the simpler two body, Sun-Earth system and expand this to the Sun-Earth-Jupiter system and, eventually, the entire solar system. The solar system will be implemented using Object Oriented (OO) programming in order to take advantage of the repetitive structure of the ODEs. Initial conditions for this system come from data provided by NASA's Jet Propulsion Laboratory. Finally, we will end with some remarks and conclusions.

2 Methods

2.1 Newton's Equations for the Earth Sun System

The motion of the planets are dictated by the gravitational force which is given by

$$F_G = \frac{GM_\odot M_{\text{Earth}}}{r^2},$$

where $M_\odot$ is the mass of the Sun and M_{Earth} is the mass of the Earth. G is the universal gravitational constant and r is the distance between the center of mass of the Earth and the Sun. The Sun's mass is over 300,000 times the mass of the Earth so, for our model, we can safely neglect any motion of the Sun. Newton's second law states

$$\frac{d^2x}{dt^2} = \frac{F_{G,x}}{M_{\text{Earth}}},$$

and

$$\frac{d^2y}{dt^2} = \frac{F_{G,y}}{M_{\text{Earth}}},$$

where $F_{G,x}$ and $F_{G,y}$ are the x and y components of the gravitational force. There is an analogous equation for z, but the orbit of the earth is, to good approximation, confined to a plane so we will neglect this for now. If we substitute in the relationship for force and make the substitution that $x = rcos(\theta)$ and $y = rsin(\theta)$, we are left with

$$\frac{d^2x}{dt^2} = -\frac{GM_\odot x}{r^3},$$

and

$$\frac{d^2y}{dt^2} = -\frac{GM_\odot y}{r^3},$$

where $r = \sqrt{x^2 + y^2}$. Now we want to rewrite these equations as first order differential equations. We do this by making the substitution that velocity is the first derivative of position. This substitution leaves us with four, first order, ODEs:

$$\begin{aligned}
\frac{dx}{dt} &= v_x \\
\frac{dy}{dt} &= v_y \\
\frac{dv_x}{dt} &= -\frac{GM_\odot x}{r^3} \\
\frac{dv_y}{dt} &= -\frac{GM_\odot y}{r^3}.
\end{aligned}$$

Finally, we can clean up these equations by our choice of units and a clever substitution. The natural unit for this system is the Astronomical Unit (AU) which is the average distance between the Sun and Earth. We will use years as our unit of time. For circular motion, we know that

$$F_G = \frac{M_{\text{Earth}} v^2}{r} = \frac{GM_\odot M_{\text{Earth}}}{r^2},$$

where v is the velocity of Earth. If the radius of the Earth's orbit is 1AU, then the velocity of the Earth is $\pi 1AU^2$. Substitution this in, the equation above can be solved to show that

$$v^2 r = GM_\odot = 4\pi^2 \text{AU}^3/\text{yr}^2.$$

Making this final substitution, the ODEs which we must solve are

$$\begin{aligned}
\frac{dx}{dt} &= v_x \\
\frac{dy}{dt} &= v_y \\
\frac{dv_x}{dt} &= -\frac{4\pi^2 x}{r^3} \\
\frac{dv_y}{dt} &= -\frac{4\pi^2 y}{r^3}.
\end{aligned}$$

2.2 Euler's Method

The first method we will implement for solving the Sun-Earth system is Euler's Method. This method is derived from a Taylor expansion which can be expressed as

$$f(x+h) = \sum_{i=0}^{\infty} \frac{h^i}{i!} f^{(i)}(x)$$

where $f^{(i)}$ is the ith derivative and h is a step in the x variable. If we keep only the first two terms, we are left with

$$f(x+h) = f(x) + hf^{(1)} + O(h^2).$$

Applying this relationship to our four differential equation, the algorithm for Euler's method is

$$\begin{aligned} x_{i+1} &= x_i + v_{xi} + O(h^2) \\ y_{i+1} &= y_i + v_{yi} + O(h^2) \\ v_{x_i+1} &= v_{x_i} - \frac{4\pi^2}{r_i^3} x_i h + O(h^2) \\ v_{y_i+1} &= v_{y_i} - \frac{4\pi^2}{r_i^3} y_i h + O(h^2). \end{aligned}$$

In this case, h is a step in time so $h = \frac{t_{max} - t_{min}}{N}$ where N is the number of time steps used.

2.3 Velocity Verlet Algorithm

The Verlet method also begins with a Taylor expansion, this time to second order, gving us

$$x(t+h) = x(t) + hx^{(1)}(t) + \frac{h^2}{2} x^{(2)}(t) + O(h^3).$$

We know from Newton's law that the second derivative position is equatl to the acceleration. We can then add the cooresponding equation for the Taylor expansion of $x(t-h)$

$$x(t-h) = x(t) - hx^{(1)}(t) + \frac{h^2}{2} x^{(2)}(t) + O(h^3).$$

After this addition and our usual discritization of the expressions, we obtain

$$x_{i+1} = 2x_i - x_{i-1} + h^2 x_i^{(2)} + O(h^4).$$

This expression is nice because the turncation error in position goes as $O(h^4)$, but it also has an obvious flaw. If we set an initial value for x_0, it is unclear how

to handle the x_{i-1} term. This means the position is not "self-starting." We can deal with this by introducing, instead, the Velocity Verlet method.

We can start again with 2nd order Taylor expansions for position and velocity

$$x_{i+1} = x_i + hx_i^{(1)} + \frac{h^2}{2}x_i^{(2)} + O(h^3)$$

and

$$v_{i+1} = v_i + hv_i^{(1)} + \frac{h^2}{2}v_i^{(2)} + O(h^3).$$

We know that the first derivative of position is velocity and Newton's law gives us an expression for the first derivative of velocity and the second derivative of position

$$v_i^{(1)} = \frac{d^2x}{dt^2}|_i = \frac{f(x_i, t_i)}{m}$$

Newton's law, however, does not provide an expression for the second derivative of velocity, but we can obtain this through the expanding the first derivative of velocity

$$v_{i+1}^{(1)} = v_i^{(1)} + hv_i^{(2)} + O(h^2).$$

Solving this expression for $v_i^{(2)}$ gives us a value to plug into our Taylor expansion. After making that substitution, our final Velocity Verlet algorithm expressions are

$$x_{i+1} = x_i + hv_i + \frac{h^2}{2}v_i^{(1)} + O(h^3)$$

and

$$v_{i+1} = v_i + \frac{h}{2}\left(v_{i+1}^{(1)} + v_i^{(1)}\right) + O(h^3).$$

These expressions are self-starting, but require the position at time t_{i+1} is calculated becfore the velocity at that point can be calculated.

2.4 Expanding Equations to Solar System

It is fairly straightforward to expand the expressions for the Sun-Earth system to include the entire Solar System. As an example, we can think about adding Jupiter to the system, giving us a three-body problem. Jupiter is a natural choice becuase it is the heaviest planet in the solar syster, about 1000 times smaller than the Sun. The gravitational force between the Earth and Jupiter is

$$F_{\text{Earth}-\text{Jupiter}} = \frac{GM_{\text{Jupiter}}M_{\text{Earth}}}{r^2_{\text{Earth}-\text{Jupiter}}},$$

We can apply the same transformations as before and obtain the following expression for the motion of the Earth

$$\frac{dv_x^E}{dt} = -\frac{GM_\odot}{r^3}x_E - \frac{GM_J}{r_{EJ}^3}(x_E - x_J).$$

If we normalize everything to the orbit and mass of the earth, we obtain

$$\frac{dv_x^E}{dt} = -\frac{4\pi^2}{r^3}x_E - \frac{4\pi^2 M_J/M_\odot}{r_{EJ}^3}(x_E - x_J).$$

There is an analogous equation for motion in the y and z position. In this report we seperately study the Earth-Sun, Sun-Earth-Jupiter, and Solar Systems. To expand these expressions to the entire solar system, one just needs to add an additional force term corresponding to each two body interaction. These forces can all be summed into one total force and used in the same Velocity Verlet expressions from the previous section. The masses and distances to the Sun for the solar system are given in the table below.

Planet	Mass in kg	Distance to sun in AU
Earth	$M_{\text{Earth}} = 6 \times 10^{24}$ kg	1AU
Jupiter	$M_{\text{Jupiter}} = 1.9 \times 10^{27}$ kg	5.20 AU
Mars	$M_{\text{Mars}} = 6.6 \times 10^{23}$ kg	1.52 AU
Venus	$M_{\text{Venus}} = 4.9 \times 10^{24}$ kg	0.72 AU
Saturn	$M_{\text{Saturn}} = 5.5 \times 10^{26}$ kg	9.54 AU
Mercury	$M_{\text{Mercury}} = 3.3 \times 10^{23}$ kg	0.39 AU
Uranus	$M_{\text{Uranus}} = 8.8 \times 10^{25}$ kg	19.19 AU
Neptun	$M_{\text{Neptun}} = 1.03 \times 10^{26}$ kg	30.06 AU
Pluto	$M_{\text{Pluto}} = 1.31 \times 10^{22}$ kg	39.53 AU

3 Code and Implementation

All of the programs, results, and benchmarks for this work can be found in my GIT repository (https://github.com/poxonpea/PHYS905). All codes for this project were written in FORTRAN.

3.1 Implementing Euler and Verlet Algorithms

The code containing the most important elements of Euler's method for the Sun-Earth is shown below.

```
!Initial Conditions from JPS at Midnight on 3/16/17

  x(1)=-0.988251
  y(1)=0.08499779
  vx(1)=(365.25*(-.0068024))
  vy(1)=(365.25*(-0.01719988))

! Euler Method Implementation

  do i=1,Num
     x(i+1) = x(i) + h*vx(i)
     y(i+1) = y(i) + h*vy(i)
     vx(i+1) = vx(i)-((h*4*PI*PI*x(i))/(r(x(i),y(i))**3.d0))
     vy(i+1) = vy(i)-((h*4*PI*PI*y(i))/(r(x(i),y(i))**3.d0))

  end do
```

Figure 1: The code used to implement Euler's method for the Sun Earth system.

While this algorithm is simple, it is also unstable. The plot below shows the trajectory of the earth over 10 years in 1000 time steps using initial conditions from the Jet Propulsion Labratory [2] at midnight on March 16,2017.

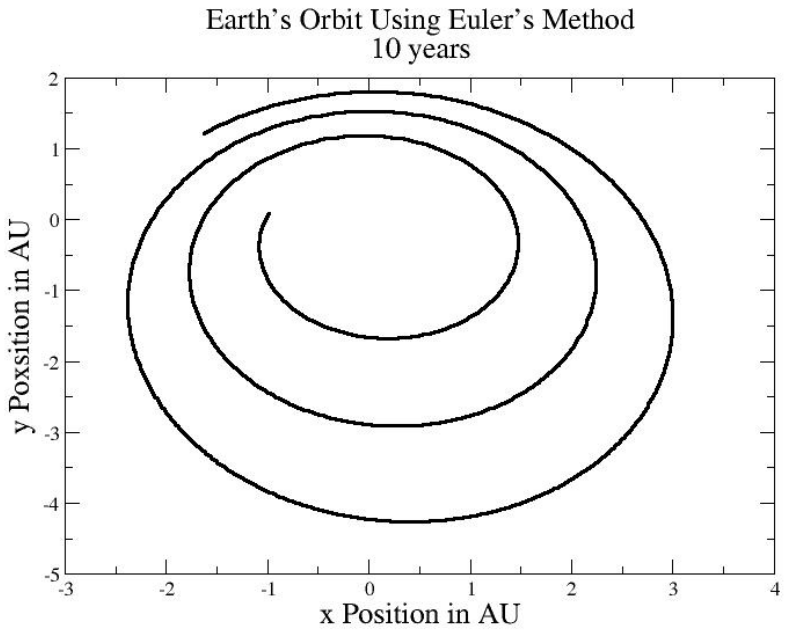

Figure 2: The trajectory of the Earth in the Sun-Earth system produced by Euler's method

Almost immediately, the trajectory begins to spin out away from the Sun. Because of this, there is no point pursuing this method for larger systems.

Instead, we will use the Velocity Verlet method. The code containing the most important elements of te Velocity Verlet method for the Sun-Earth is shown below.

```
!Initial Conditions from JPS at Midnight on 3/16/17

  x(1)=-0.988251
  y(1)=0.08499779
  vx(1)=(365.25*(-.0068024))
  vy(1)=(365.25*(-0.01719988))

  do i=1,Num
     x(i+1)=x(i)+h*vx(i)-(h*h/2)*((4.d0*pi*pi*x(i))/(r(x(i),y(i))**3))
     y(i+1)=y(i)+h*vy(i)-(h*h/2)*((4.d0*pi*pi*y(i))/(r(x(i),y(i))**3))
     vx(i+1)=vx(i)-(h/2)*((4.d0*pi*pi*x(i+1))/(r(x(i+1),y(i+1))**3))-(h/2)*((4.d0*pi*pi*x(i))/(r(x(i),y(i))**3))
     vy(i+1)=vy(i)-(h/2)*((4.d0*pi*pi*y(i+1))/(r(x(i+1),y(i+1))**3))-(h/2)*((4.d0*pi*pi*y(i))/(r(x(i),y(i))**3))
  end do
```

Figure 3: The code used to implement the Velocity Verlet method for the Sun Earth system.

The plot below shows the trajectory of the earth over 10 years in 1000 time steps using initial conditions from the Jet Propulsion Labratory at midnight on March 16,2017.

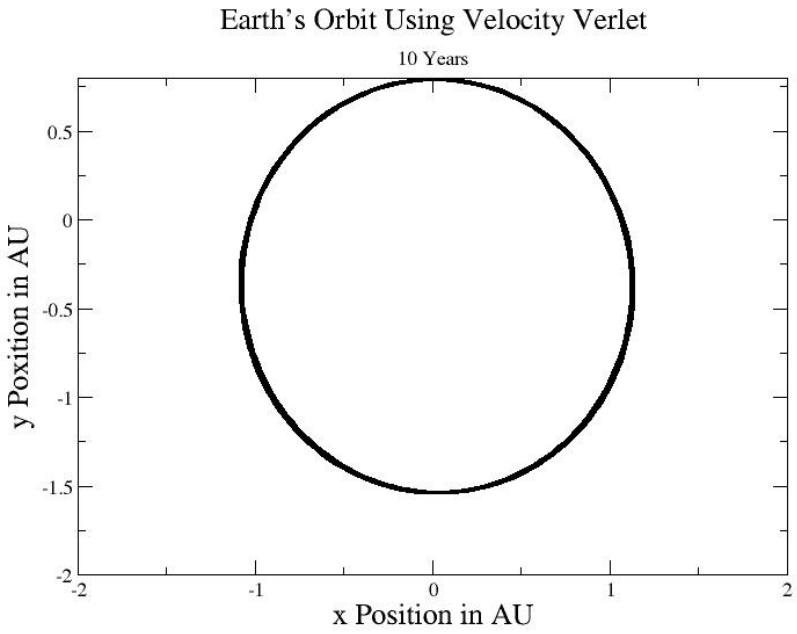

Figure 4: The trajectory of the Earth in the Sun-Earth system produced by the Velocity Verlet method

Unlike Euler's method, the Velocity Verlet algorithm produces a stable, almost circular orbit. We can investigate to see how much more computationally expensive the Velocity Verlet method is. If we examine the algorithm for Euler's method and we precalculate as many factors as possible to decrease the number of FLOPS in the main algorithm, Euler's method in two dimensions needs 16N FLOPS where N is the number of time steps taken. We we apply a similar analysis to the Velocity Verlet algorithm, we get almost 30N FLOPS in 2 dimensions.

We can also examine this difference by looking at the time it takes to run the algorithm. Because both methods are very fast, I ran them for the case of 100 years with 100,000 time steps. This is well above the necessary precision, but will allow for the clocks to accumulate a longer computation time. The Euler method took 0.3359 seconds to complete that computation, while the Velocity Verlet method required 0.444 seconds. This shows that the Velocity Verlet algorithm maintains a good balance of precision and computational speed. Because of this, we will use the Velocity Verlet algorithm to extend our code to the Sun-Earth-Jupiter and Solar Stystems. Because all of the equations for this system follow a similar format, we will switch to an OO structure for our code, which will be discussed in more detail in the following section.

3.2 Object Oriented Code

I delveloped an object oriented version of the Velocity Verlet method to solve the Earth-Sun, Earth-Sun-Jupiter, and Solar systems. This involved defiining a type, which I called "solver" that contained all of the masses, positions, and velocities of the system that you are solving. This solver also conatins a series of subroutines that can be used to calculate relative positions, forces, and recalculate a planet's position. amoung other things. This declaration can be seen below.

```
type solver
   integer::Bodies
   real(8),dimension(2) :: mass
   real(8),dimension(2,2) :: positio
   real(8),dimension(2,2) :: velocit
contains
  procedure ::relative_position
  procedure ::forces
  procedure ::calc_position
  procedure ::calc_velocities
  procedure ::kinetic_energy
  procedure ::potential_energy
  procedure ::angular_momentum

end type solver
```

Figure 5: Definition of Solver type in OO code.

This module is accompanies by a function which makes calls to the procedures in "solver" to apply the Velocity verlet method. This program also prints the results to file so they can be plotted. The heart of this code is shown below.

```
h=Tmax/Num_Steps
do m=1,Num_Steps

   call relative_position(earthsun,numbodies,relposition)
   call forces(earthsun,numbodies,relposition,relforce)
   call calc_position(earthsun,numbodies,relforce,h)
   call relative_position(earthsun,numbodies,updaterel)
   call forces(earthsun,numbodies,updaterel,updatedforce)
   call calc_velocities(earthsun,numbodies,relforce,updatedforce,h)
   call kinetic_energy(earthsun,numbodies,kinetic)
   call potential_energy(earthsun,numbodies,updaterel,potential)
   call angular_momentum(earthsun,numbodies,updaterel,angular)

   write(4,*), kinetic(Numbodies+1), potential(numbodies+1)
   write(5,*), angular(Numbodies+1)

write(3,*),earthsun%position(2,1), earthsun%position(2,2)

end do
```

Figure 6: Heart of the main function for the OO program.

This portion of the code does not need to be modified at all when the user switches betweem systems. All that needs to be changed is the size of arrays in "solver" and the additional intial conditions need to be defined.

3.3 Tests of Code

4 Results and Discussion

4.1 Escape Velocity of the Sun-Earth System

If an object's kinetic energy exceeds the potential energy keeping it in orbit, it can escape from the system. This escape velocity can be found by

$$-\frac{Gm_1M_2}{R} = \frac{1}{2}m_1v^2$$

so

$$v_{esc} = \sqrt{\frac{2GM_2}{R}}.$$

The normal orbital velocity can be found by setting the gravitational force equal to the centripetal force. Solving this leaves us with an orbital velocity of $v_{orb} = \sqrt{\frac{GM_2}{R}}$ so that

$$v_{escape} = \sqrt{2}v_{orbit}.$$

For the Earth, the velocity is 2π AU/year, so the escape velocity should be roughtly $\sqrt{2}2\pi$ AU/year.

4.2 The Three Body Problem

4.3 Solar System Model

5 Conclusions

6 Appendices

6.1 Appendix A

References

[1] Hjorth-Jensen, Mortehn. Computational Physics, Lecture Notes Fall 2015. August 2015.

[2] Jet Propulsion Laboratory . HORIZONS Web-Interface. http://ssd.jpl.nasa.gov/horizons.cgi.