# Kepler Problem

Numerical Ordinary Differential Equations

**Jonathon D'Arcy**
S1607860

December 7, 2021

## 1 Kepler Problem

To the right we have the position graph of the Kepler problem using the verlet method with the initial conditions of $q_0 = (1, 0)$ and $p_0 = (0.4, 0.6)$, where $q_0$ and $p_0$ denote the starting position and momentum respectively. We can clearly see through the four graphs that altering the step size of the verlet method causes an offset between starting positions of each orbit, causing in large step sizes the Spirograph like images. We also can note that increasing the step size allows us to draw more orbit cycles (Assuming we keep the same iteration number). This can be seen by comparing the subplots corresponding to the step size 0.001, and 0.04. In the first subplot move through about 1.5 orbits, while in the subplot for step size 0.04 we make over 110 full rotations. It should be noted that the offset is not created by the increase in orbit count. We know this to be true because if we increase the step count of subplot 1 (step size 0.001) to 101 times it current step count, we still do not achieve a similar plot to the fourth subplot. Finally we must note that the step size of 0.001 is not achieving no offset, its offset is much smaller. We can calculate this off set by taking the Euclidean norm of the difference between the starting position and it closest return point. This occurs at step 6981 and the value of the Euclidean norm is $2 * 10^{-4}$.

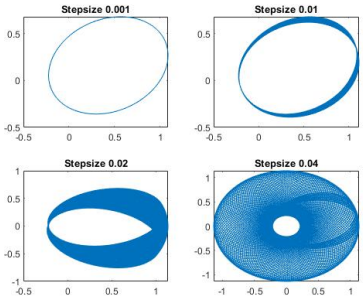


Figure 1: Position of Kepler Problem at Varying Step Sizes

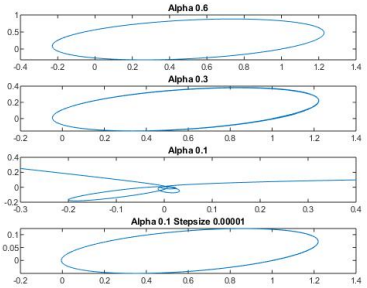


Figure 2: Effects of varying starting Momentum

As well as manipulating our step sizes and counts we can change our inputs to position and momentum variables. The section deals with the manipulation of the starting momentum of the planet. In figure 2 we create a representation of the changes incurred to the orbit of a planets path when altering the amplitude of y-directional momentum. More specifically we vary the momentum through the values $p_0 = (0.6, 0.6)$, $p_0 = (0.6, 0.3)$, and $p_0 = (0.6, 0.1)$. After observing the outputs and noting the respective axes, we see that as we decrease the magnitude of the initial impulse in the y-axis we increase the eccentricity of the typical orbits of the planet. This is directly due to the decrease magnitude of momentum, as it will take the same time for the gravity to flip the x-axis momentum and having a lower upwards momentum will thus cause tighter ellipses.

We also notice in the third graph a phenomena which occurs when the verlet method lands the moving planet very close to stable planet. Here due to the very close proximity of the planets the force enacted on the moving planet becomes very large, thrusting the planet away with a lot of momentum. While this may appear to make

the planet leave forever eventually with a large enough step count it will return. This is due to the fact that the planets, no matter how small the attraction, are always attracted to each other and only each other, thus will eventually cause the return of the moving planet at a very eccentric orbit. This all however is because of an error of the verlet method and is occurring because our step size is too large, if we consult our fourth subplot we can fix this by decreasing our step size to 0.00001 and our step count to 1000000. What we learn here is that for models which depict close approaches to planets we need to enlarge our step size, because of this an application of a method with variable step size may be useful.

## 2 Two Fixed Centres Problem

To create the code for the two fixed centres problem (TFC) we must first derive the force field function, $F(q)$, from the given potential energy function $U(q)$ defined as

$$U(q) = -\frac{\gamma}{\|q - q_1\|_2} - \frac{\gamma}{\|q - q_2\|_2}. \tag{1}$$

We are given that $F(q)$ is equal to the negative gradient of U(q), thus

$$F(q) = -\nabla U(q) = -\nabla(-\frac{\gamma}{\|q - q_1\|_2} - \frac{\gamma}{\|q - q_2\|_2}). \tag{2}$$

Since $\nabla$ is linear we can rearrange this to

$$F(q) = \gamma(\nabla\frac{1}{\|q - q_1\|_2} + \nabla\frac{1}{\|q - q_2\|_2}). \tag{3}$$

This leave us with two calculations of the gradient of a Euclidean norm. To do this we look to the calculation of the gradient of a general p-norm which is known to be

$$\frac{\partial\|q\|_p}{\partial q} = \frac{q \circ |\ q\ |^{p-2}}{\|q\|_p^{p-1}}, \quad \text{where } \circ \text{ is the Hadamard product.} \tag{4}$$

We now set $p = 2$ which corresponds to the Euclidean norm. Here the term $|\ q\ |^{p-2}$ simplifies down to $|\ q\ |^0$ which equals the ones vector of the same dimension. The Hadamard product of q and x, where x is the ones vector of identical dimension of q is always equal to q. Thus for the Euclidean norm line (4) reduces to the form

$$\frac{\partial\|q\|_2}{\partial q} = \frac{q}{\|q\|_2}. \tag{5}$$

We can then implement this form into (3) with the chain rule in mind to gather

$$F(q) = \gamma(-\frac{q - q_1}{\|q - q_1\|_2^3} - \frac{q - q_2}{\|q - q_2\|_2^3}). \tag{6}$$

With this force field function acquired we can now create the code for the TFC problems, these can be viewed in the Mat lab code section under TFCInit and TFCForce. Note when viewing these that TFCInit does not include the initial positions of the two fixed masses. This is purposefully do as the positions vary from question to question and rather than creating a function of TFCInit it was elect to just rewritten the 2 lines each time. To validate the accuracy of this code we can plot an output of the model where both the fixed objects are placed at the origin. This will allow us to validate what we have because as $q_1 = q_2 = 0$ equation (6) will reduce to the form that $F(q)$ took in the Kepler problem with twice the value of $\gamma$. This means if we run the two method with the same initial conditions apart from a doubled value of $\gamma$ we should output the same results. This can be seen as true in Figure 3.

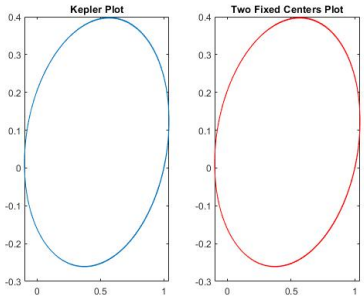


Figure 3: Position of Kepler Problem at Varying Step Sizes

Now that we have validated our code for the two fixed centres problem we can implement it to the problem where we have two fixed planets at $(0,1)$ and $(0,-1)$ respectively, and the moving planet with initial conditions $\gamma = 1$, $q_0 = (0,0)$, and $p_0 = (0.45, 0.25)$. If we run these condition through the Verlet method over the time frame $[0,10]$ we can compare step sizes of 0.0001 and 0.001 we see this in Figure 4. Comparing these outputs we see that the larger output diverges from the smaller output, by the time that we pass the bottom planet. This is due to what we discussed earlier as the larger step size is approaching the stable planet closer than it should creating a larger impulse of momentum drawing it further away from the true solution. We notice however that the trajectory of the smaller step size does not appear to be periodic and that if left to continue it will have a different path through the planets as it has a different initial position on its next orbit. We can find through running this through the time frame [0,100] that the even the smaller step size will destabilize as it gets to near a planet and shoots off in error.

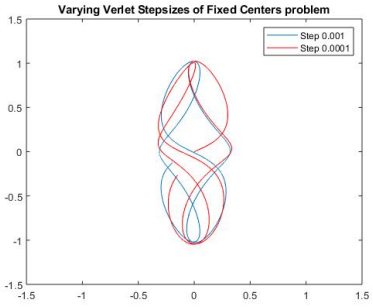


Figure 4: Position of Moving Planet in the Fixed Centres Problem at Varying Step Sizes of the Verlet Method

We now focus on the smaller step size but this time we increase the duration of the time frame to [0,20] and plot as well the RK4 method of the simulation over the same interval with a step size of 0.04. The result of these models is given in figure 5. In the first subplot of figure 5 we can see that the Verlet method as stated before has a new trajectory in its second pass. We can also see that the RK4 method has approached too close to the upper planet shooting off. The second subplot gives us a better idea of what happened to the RK4 model as we see that it began to tightly orbit the one point as the force of the planet that it was near was so much greater than that of the one further away. What we can extrapolate through this is that even if we had a model that varied it step size as we discussed before if the model fell into an inward winding orbit like this we would have to take progressively smaller and smaller step sizes. As this would get computationally very heavy it may be better in a model like this to have the ability to end a model wen a moving planet gets trapped and classify it as a collision. This is important especially for modelling planets as this phenomenon is true in Accretion disks. This implementation should be carefully done as the opposite effect can happen, for instance our own moon is slowly moving away from the Earth.

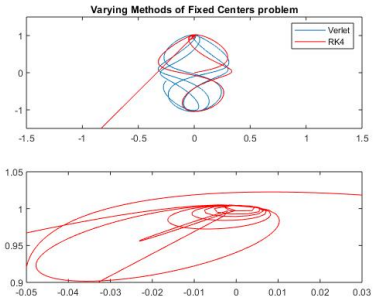


Figure 5: Position of Moving Planet in the Fixed Centers Problem at Varying Step Sizes of the Verlet Method

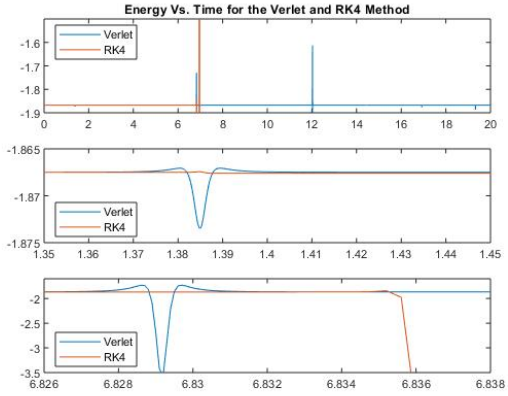


Figure 6: Energy Vs. Time For Verlet and RK4 Methods

In Figure 6 we plot tight axes at specific times, including the collision we had earlier encountered. This time however we plot not the location of the moving planet but rather the energy possessed by the planet verses the time we are at. In the first graph we can see a massive spike at a time of about 7, this is the prior discussed wrapping into the planet, we also notice two major spikes under gone by the Verlet method, this is where the planet passes by the planets on the tight side of its orbit. Further inspecting the subplot 2 of figure 6 we see that the energy first increases then decreases then increases again to a symmetric high before returning to its stable line. This is however what we expect from the nature of an orbiting planet however. We can see through the third subplot that we take this same form only scaled higher, if we were to zoom on the 12 time frame we would see this again. The reason for this increase is due to the verlets method previously stated off set and each time it is moving into a trajectory which brings it closer to a planet, eventually it will get too close and hit its own error, making it to appear to have left as the RK4 graph does. Both of these however will eventually return as stated prior and will continue this expanding oscillation but as the planet is so far away now for this to take effect will be a very long time frame extension, and each subsequent one after that will grow too. It is because of this we can say that neither method works better that the other if we alter the step sizes to be equal, however from a computational stand point since RK4 is roughly four time as expensive as the Verlet method we say that the Verlet method is better.

## Matlab Code

### KeplerProb1

```
KeplerInit
figure(1)
for i=1:4
    hvect=[0.001,0.01,0.02,0.04];
    stepvect=[10000,10000,10000,10000];
    [T,Q,P,H]=HamSolver(q0,p0, stepvect(i),hvect(i),@KeplerForce, 'Verlet', Pars);
    subplot(2,2,i)
    plot(Q(1,:),Q(2,:))
    title(['Stepsize ',num2str(hvect(i))]);
    KeplerInit
end
```

### KeplerProb2

```
KeplerintiP2
alpha=[0.6,0.3,0.1];
figure(3);
for i=1:3
    hvect=[0.001,0.001,0.001];
    stepvect=[5000,5000,5000,5000,5000,5000];
    p0=[0.6;alpha(i)];
    [T,Q,P,H]=HamSolver(q0,p0, stepvect(i),hvect(i),@KeplerForce, 'Verlet', Pars);
    subplot(3,1,i)
    plot(Q(1,:),Q(2,:))
    if i==3
        axis([-0.3 0.4 -0.2 0.4]);
    end
    title(['Alpha ',num2str(alpha(i))]);
    KeplerintiP2
end
```

### TFCInit

```
q0 = [1;0]; p0=[0.4;0.6];
Pars.Mass = [1 1]'; Pars.gamma = 1;
```

## TFCForce

```
function [F,U] = TFCForce(q,Pars)
q1=Pars.q1;
q2=Pars.q2;
r1 = norm(q-q1);
r2 = norm(q-q2);
U = -(Pars.gamma/r1)-(Pars.gamma/r2);
F = (-Pars.gamma*(q-q1)*r1^(-3)-Pars.gamma*(q-q2)*r2^(-3));
```

## TFCProb1

```
TFCInit
Pars.q1 = [0;0] ; Pars.q2 = [0;0];
[T,Q,P,H]=HamSolver(q0,p0, 10000,0.001,@TFCForce, 'Verlet', Pars);
figure(1)
plot(Q(1,:),Q(2,:))
figure(2)
KeplerInit
[T,Q,P,H]=HamSolver(q0,p0, 10000,0.001,@KeplerForce, 'Verlet', Pars);
plot(Q(1,:),Q(2,:),'r')
```

## TFCProb2

```
q0 = [0;0]; p0=[0.45;0.25];
Pars.Mass = [1 1]'; Pars.gamma = 1;
Pars.q1 = [0;-1] ; Pars.q2 = [0;1];
[T,Q,P,H]=HamSolver(q0,p0, 10000,0.001,@TFCForce, 'Verlet', Pars);
figure(1)
plot(Q(1,:),Q(2,:))
hold on
clear
q0 = [0;0]; p0=[0.45;0.25];
Pars.Mass = [1 1]'; Pars.gamma = 1;
Pars.q1 = [0;-1] ; Pars.q2 = [0;1];
[T,Q,P,H]=HamSolver(q0,p0, 100000,0.0001,@TFCForce, 'Verlet', Pars);
plot(Q(1,:),Q(2,:),'r')
axis([-1.5 1.5 -1.5 1.5]);
legend('Step 0.001','Step 0.0001')
title('Stepsizes of Fixed Centers problem')
hold off
```

## TFCProb3

```
q0 = [0;0]; p0=[0.45;0.25];
Pars.Mass = [1 1]'; Pars.gamma = 1;
Pars.q1 = [0;-1] ; Pars.q2 = [0;1];
[T,Q,P,H]=HamSolver(q0,p0, 200000,0.0001,@TFCForce, 'Verlet', Pars);
figure(1)
plot(Q(1,:),Q(2,:))
hold on
clear
q0 = [0;0]; p0=[0.45;0.25];
Pars.Mass = [1 1]'; Pars.gamma = 1;
Pars.q1 = [0;-1] ; Pars.q2 = [0;1];
[T,Q,P,H]=HamSolver(q0,p0, 50000,0.0004,@TFCForce, 'RK4', Pars);
plot(Q(1,:),Q(2,:),'r')
axis([-1.5 1.5 -1.5 1.5]);
legend('Verlet','RK4')
title('Methods of Fixed Centers problem')
hold off
```

## TFCProb4

```
q0 = [0;0]; p0=[0.45;0.25];
Pars.Mass = [1 1]'; Pars.gamma = 1;
Pars.q1 = [0;-1] ; Pars.q2 = [0;1];
[T,Q,P,H]=HamSolver(q0,p0, 200000,0.0001,@TFCForce, 'Verlet', Pars);
figure(1)
subplot(3,1,1)
plot(T(1,:),H(1,:))
hold on
axis([0 20 -1.9 -1.5]);
clear
q0 = [0;0]; p0=[0.45;0.25];
Pars.Mass = [1 1]'; Pars.gamma = 1;
Pars.q1 = [0;-1] ; Pars.q2 = [0;1];
[T,Q,P,H]=HamSolver(q0,p0, 50000,0.0004,@TFCForce, 'RK4', Pars);
plot(T(1,:),H(1,:))
hold off
clear

q0 = [0;0]; p0=[0.45;0.25];
Pars.Mass = [1 1]'; Pars.gamma = 1;
Pars.q1 = [0;-1] ; Pars.q2 = [0;1];
[T,Q,P,H]=HamSolver(q0,p0,200000,0.0001,@TFCForce, 'Verlet', Pars);
subplot(3,1,2)
plot(T(1,:),H(1,:))
axis([1.35 1.45 -1.875 -1.865]);
hold on
clear
q0 = [0;0]; p0=[0.45;0.25];
Pars.Mass = [1 1]'; Pars.gamma = 1;
Pars.q1 = [0;-1] ; Pars.q2 = [0;1];
[T,Q,P,H]=HamSolver(q0,p0, 50000,0.0004,@TFCForce, 'RK4', Pars);
plot(T(1,:),H(1,:))
hold off
clear

q0 = [0;0]; p0=[0.45;0.25];
Pars.Mass = [1 1]'; Pars.gamma = 1;
Pars.q1 = [0;-1] ; Pars.q2 = [0;1];
[T,Q,P,H]=HamSolver(q0,p0, 200000,0.0001,@TFCForce, 'Verlet', Pars);
subplot(3,1,3)
plot(T(1,:),H(1,:))
axis([6.825 6.837 -3.5 -1.6]);
hold on
clear
q0 = [0;0]; p0=[0.45;0.25];
Pars.Mass = [1 1]'; Pars.gamma = 1;
Pars.q1 = [0;-1] ; Pars.q2 = [0;1];
[T,Q,P,H]=HamSolver(q0,p0, 50000,0.0004,@TFCForce, 'RK4', Pars);
plot(T(1,:),H(1,:))
hold off
```