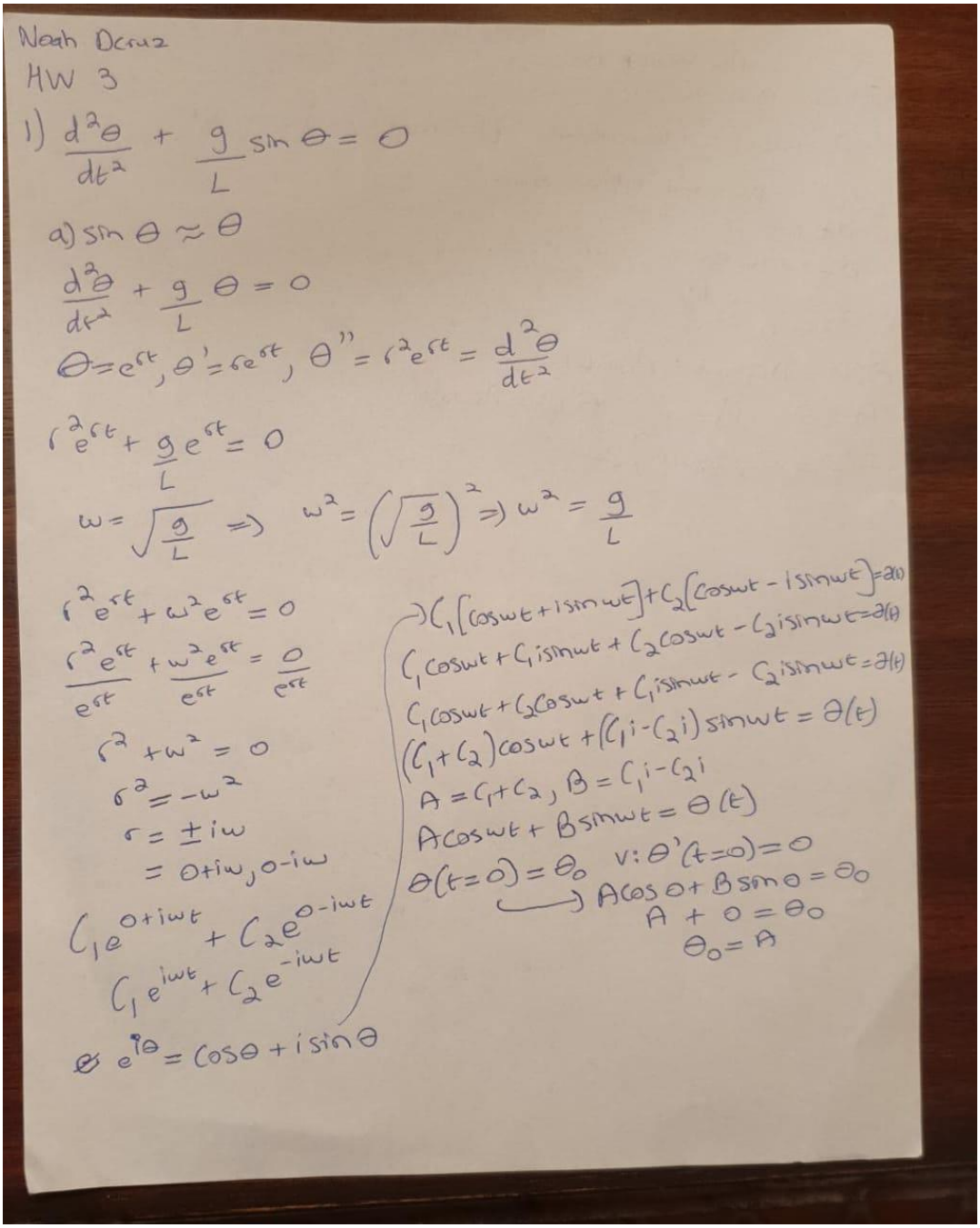
Noah Dcruz

Homework 3

MATH5131

1)

a)



$\theta'(t) = -A\omega \sin\omega t + B\omega \cos\omega t$

$\theta(t) = A\cos\omega t + B\sin\omega t$

$(A\cos\omega t)' = -A\omega \sin\omega t \quad (B\sin\omega t)' = B\omega \cos\omega t$

$\theta'(t) = -A\omega \sin\omega t + B\omega \cos\omega t$

at $t = 0$

$0 = -A\omega \sin 0 + B\omega \cos 0$

$0 = 0 + B$

$B = 0$

$\theta(t) = A\cos\omega t + B\sin\omega t$

$A = \theta_0, B = 0 \qquad \theta(t) = A\cos\omega t$

$\boxed{\theta(t) = \theta_0 \cos\omega t}$

b)

b) $r = \sqrt{\frac{g}{L}}\, t \qquad (r)^2 = \left(\sqrt{\frac{g}{L}}\, t\right)^2$

$\frac{d^2\theta}{dr^2} + \sin\theta = 0 \qquad dr^2 = \frac{g}{L}(dt^2)$

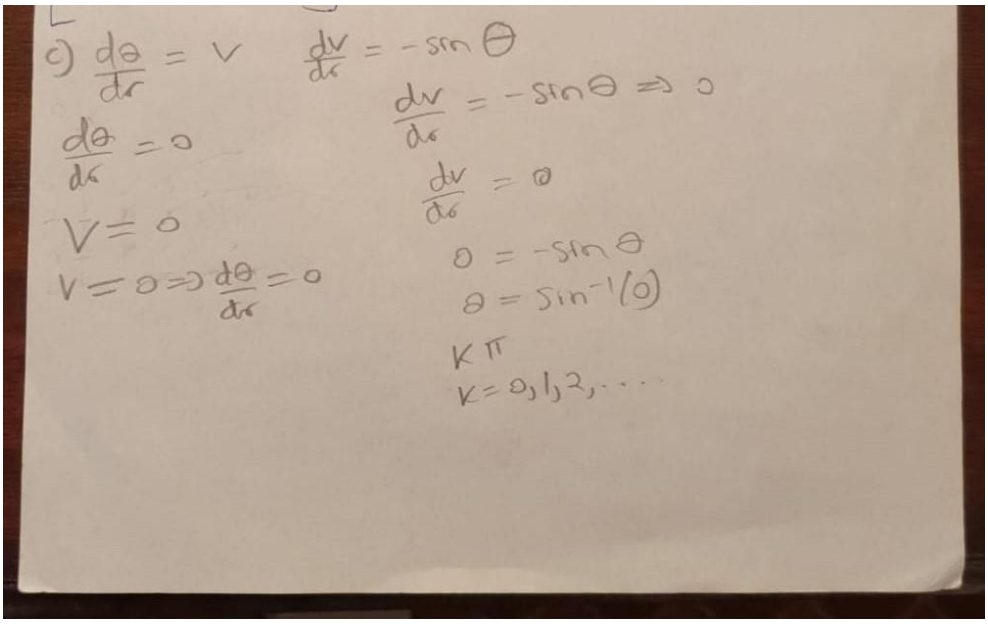
$\frac{d^2\theta}{dt^2} + \frac{g}{L}\sin\theta = 0 \qquad dt^2 = \frac{dr^2 \times L}{g}$

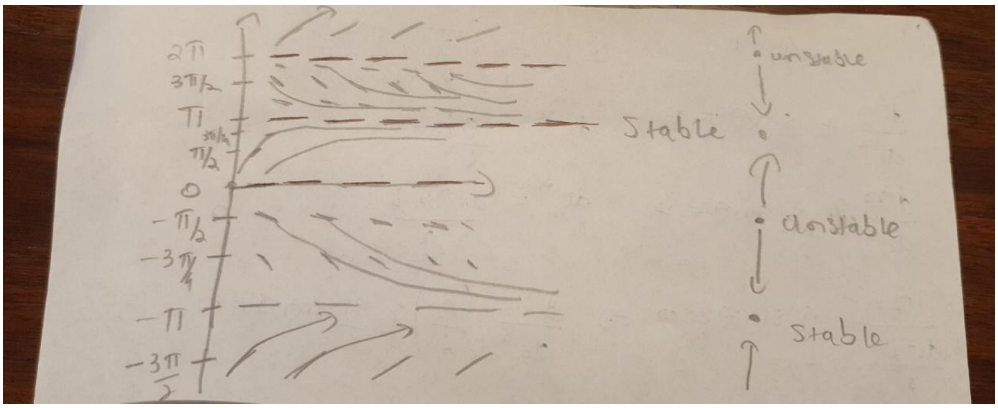
$\frac{d^2\theta}{\frac{dr^2 L}{g}} + \frac{g}{L}\sin\theta = 0 \Rightarrow \frac{d^2\theta}{dr^2}\left(\frac{g}{L}\right) + \frac{g}{L}\sin\theta = 0$

$\Rightarrow \frac{g}{L}\left(\frac{d^2\theta}{dr^2} + \sin\theta\right) = 0 \Rightarrow \frac{L}{g} \times \frac{g}{L}\left(\frac{d^2\theta}{dr^2} + \sin\theta\right) = \frac{L}{g} \times 0 \Rightarrow$

$\boxed{\frac{d^2\theta}{dr^2} + \sin\theta = 0}$

c)





d)

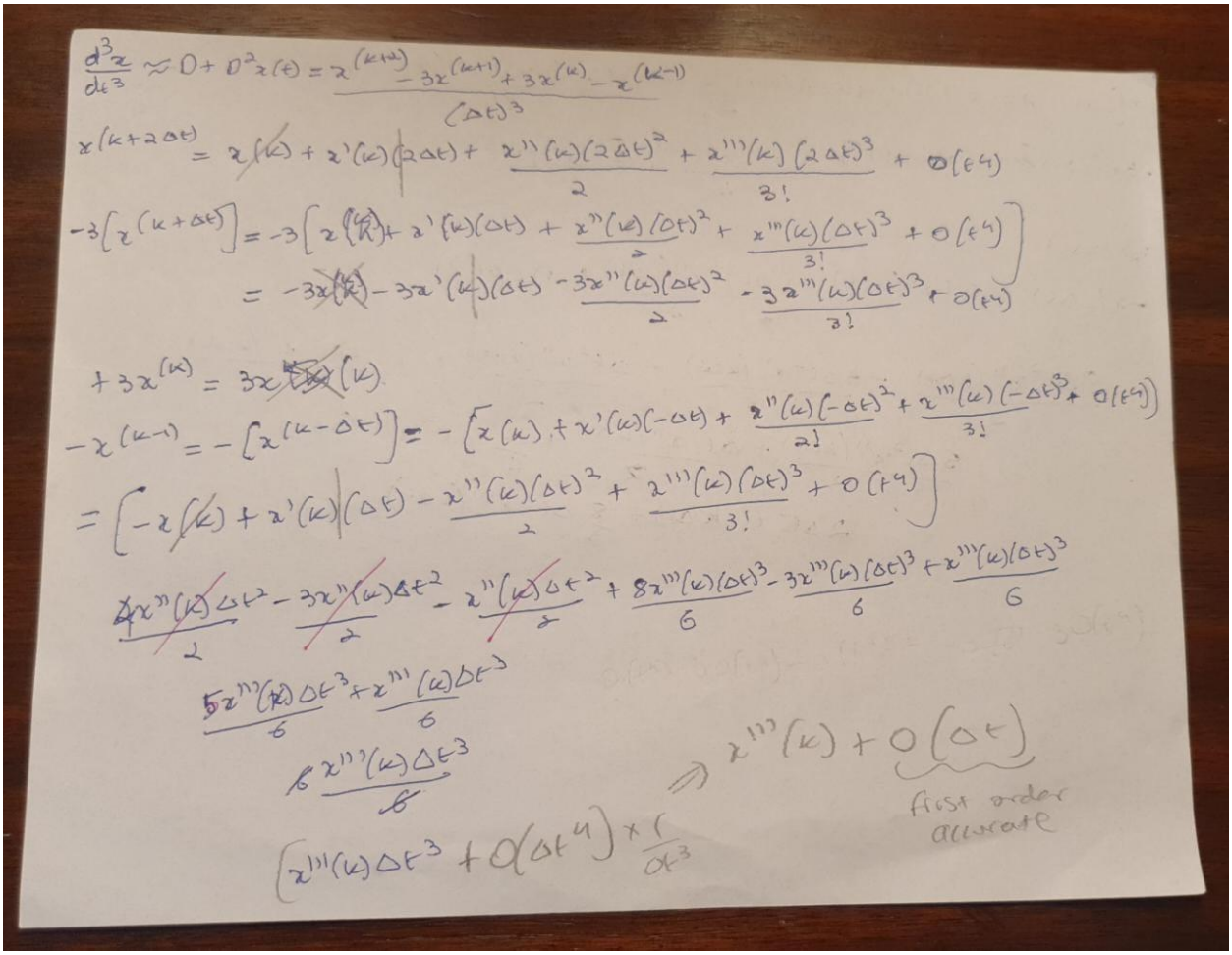
A stable equilibrium is the steady state where the pendulum remains at rest directly below with the mass hanging directly below the fixed point. The stability is that any small movement will result in a force that brings the pendulum back to its original position and the force is directed to the stable equilibrium point.

An unstable equilibrium is a steady state where the pendulum is at rest at the highest position possible. The stability of this steady state is that any small movement causes a force that drives the pendulum away from its original position.

e)

When a pendulum system is affected by friction, it will eventually lose its energy and the magnitude of its motion will decrease over time. The unstable equilibrium point of the pendulum system will disappear, leaving only one stable equilibrium point.

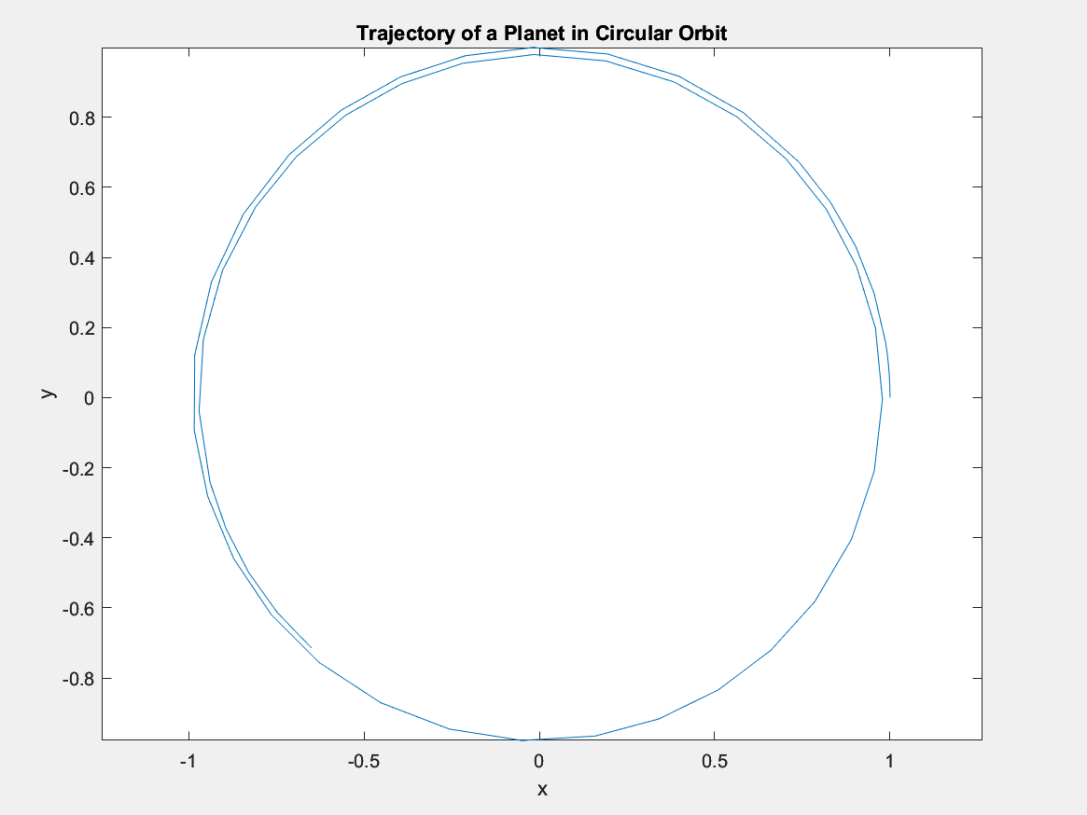
2)

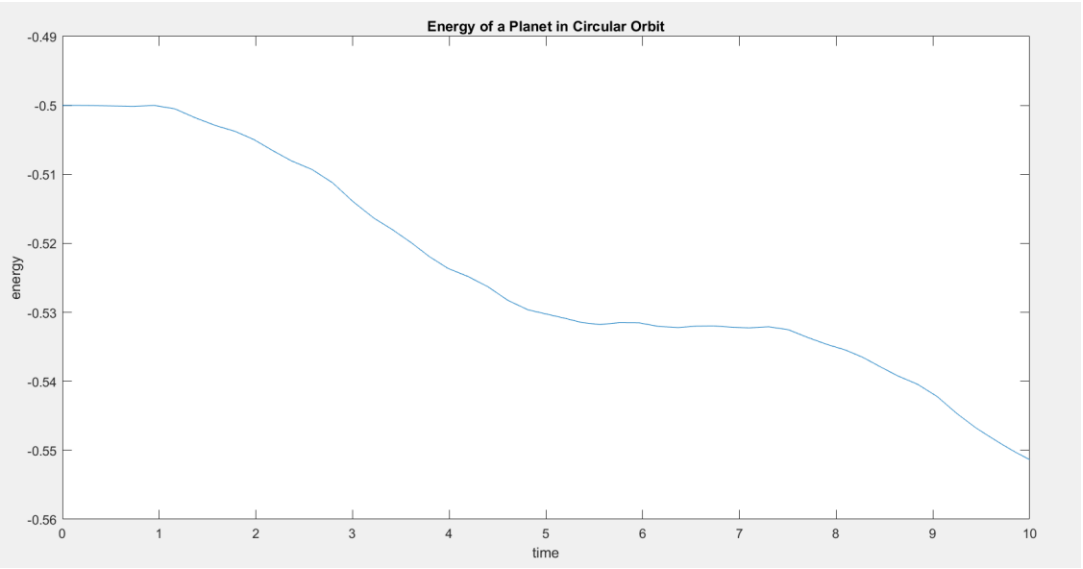

$\frac{d^3x}{dt^3} \approx D_+ D^2 x(t) = \frac{x^{(k+2)} - 3x^{(k+1)} + 3x^{(k)} - x^{(k-1)}}{(\Delta t)^3}$
$x(k+2\Delta t) = x(k) + x'(k)(2\Delta t) + \frac{x''(k)(2\Delta t)^2}{2} + \frac{x'''(k)(2\Delta t)^3}{3!} + O(t^4)$
$-3[x(k+\Delta t)] = -3\left[x(k) + x'(k)(\Delta t) + \frac{x''(k)(\Delta t)^2}{2} + \frac{x'''(k)(\Delta t)^3}{3!} + O(t^4)\right]$
$= -3x(k) - 3x'(k)(\Delta t) - \frac{3x''(k)(\Delta t)^2}{2} - \frac{3x'''(k)(\Delta t)^3}{3!} + O(t^4)$
$+3x^{(k)} = 3x(k)$
$-x^{(k-1)} = -[x(k-\Delta t)] = -\left[x(k) + x'(k)(-\Delta t) + \frac{x''(k)(-\Delta t)^2}{2!} + \frac{x'''(k)(-\Delta t)^3}{3!} + O(t^4)\right]$
$= \left[-x(k) + x'(k)(\Delta t) - \frac{x''(k)(\Delta t)^2}{2} + \frac{x'''(k)(\Delta t)^3}{3!} + O(t^4)\right]$
$\frac{4x''(k)\Delta t^2}{2} - \frac{3x''(k)\Delta t^2}{2} - \frac{x''(k)\Delta t^2}{2} + \frac{8x'''(k)(\Delta t)^3}{6} - \frac{3x'''(k)(\Delta t)^3}{6} + \frac{x'''(k)(\Delta t)^3}{6}$
$\frac{5x'''(k)\Delta t^3 + x'''(k)\Delta t^3}{6}$
$\frac{6x'''(k)\Delta t^3}{6}$
$\Rightarrow x'''(k) + O(\Delta t)$
first order accurate
$\left(x'''(k)\Delta t^3 + O(\Delta t^4)\right) \times \frac{1}{\Delta t^3}$

3) Comments in code to explain

a)

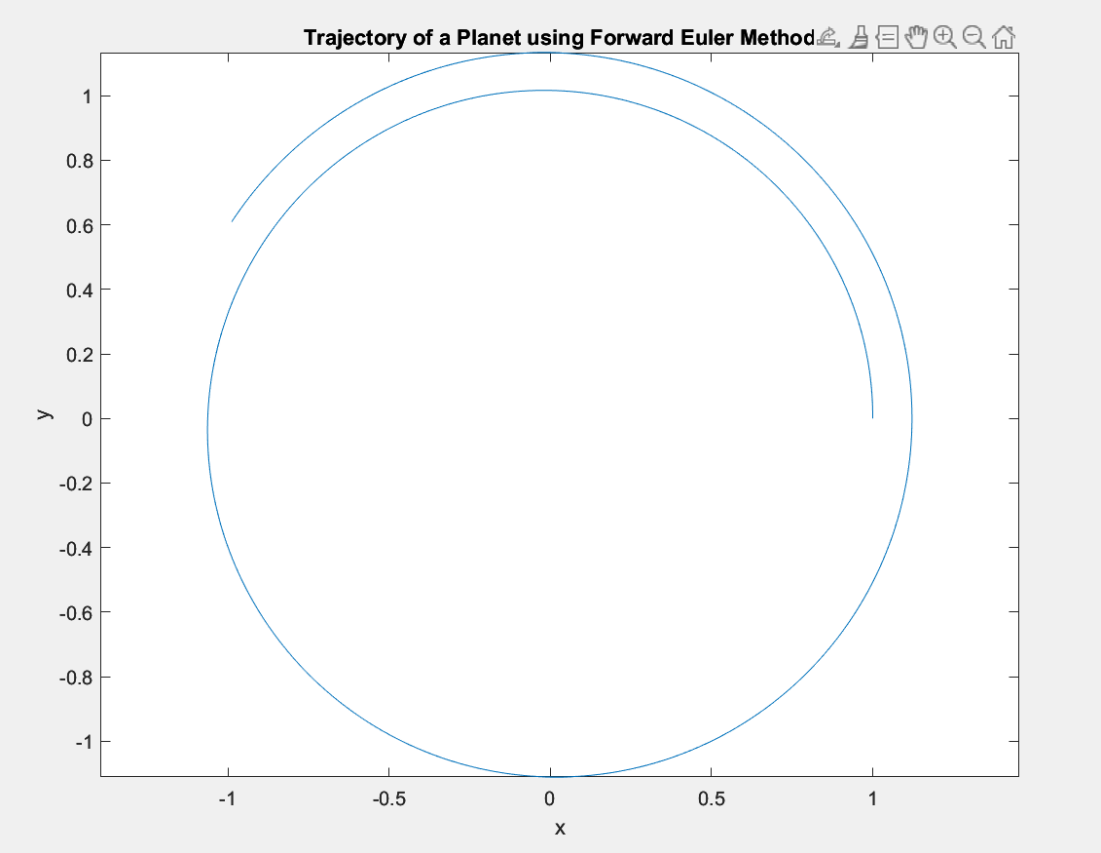
```
% Gravity, mass and position
G = 1;
m0 = 1;
r0 = [0 0];

% Initial position, velocity and circular orbit
r_init = [1 0];
v_init = [0 1];
v_init = norm(r_init)*[0 1];
y0 = [r_init v_init];

% span
tspan = [0 10];

% ODE
orbitODE = @(t,y) [y(3:4); -G*m0*y(1:2)/norm(y(1:2))^3];

% ode45
[t, y] = ode45(orbitODE, tspan, y0);

% Position and velocity
r = y(:, 1:2);
v = y(:, 3:4);

% Energy
E = -G*m0./vecnorm(r, 2, 2).^2 + 0.5*vecnorm(v, 2, 2).^2;

% Trajectory plot
plot(r(:,1), r(:,2));
xlabel('x');
ylabel('y');
title('Trajectory of a Planet in Circular Orbit');
axis equal;

% Energy plot
figure;
plot(t, E);
xlabel('time');
ylabel('energy');
title('Energy of a Planet in Circular Orbit');
```

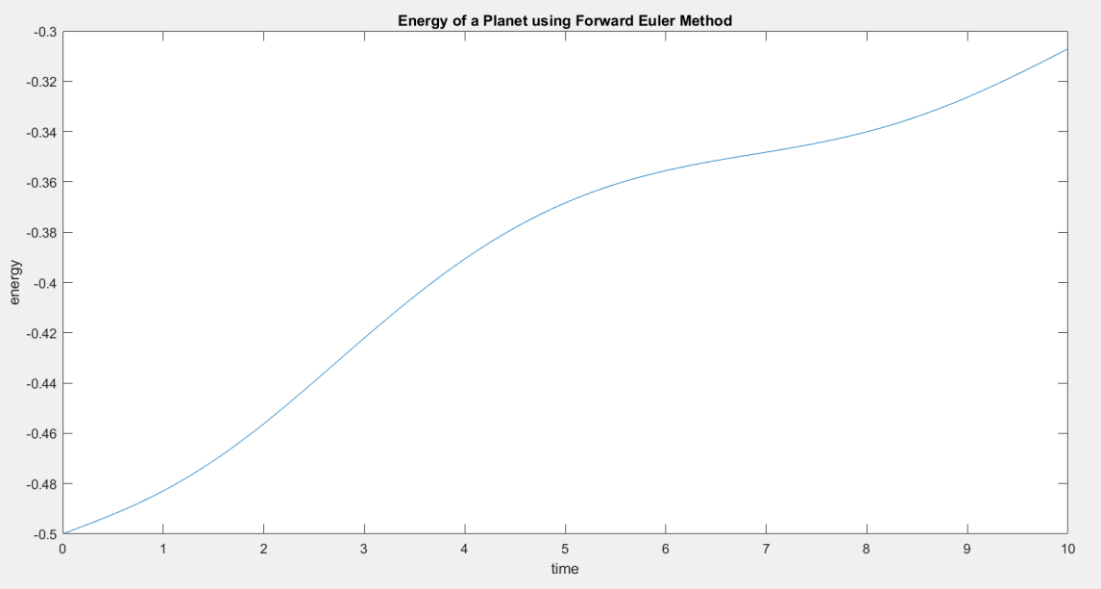

Trajectory of a Planet in Circular Orbit
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
y
-1
-0.5
0
0.5
1
x


Energy of a Planet in Circular Orbit
-0.49
-0.5
-0.51
-0.52
-0.53
-0.54
-0.55
-0.56
energy
0
1
2
3
4
5
6
7
8
9
10
time

b)

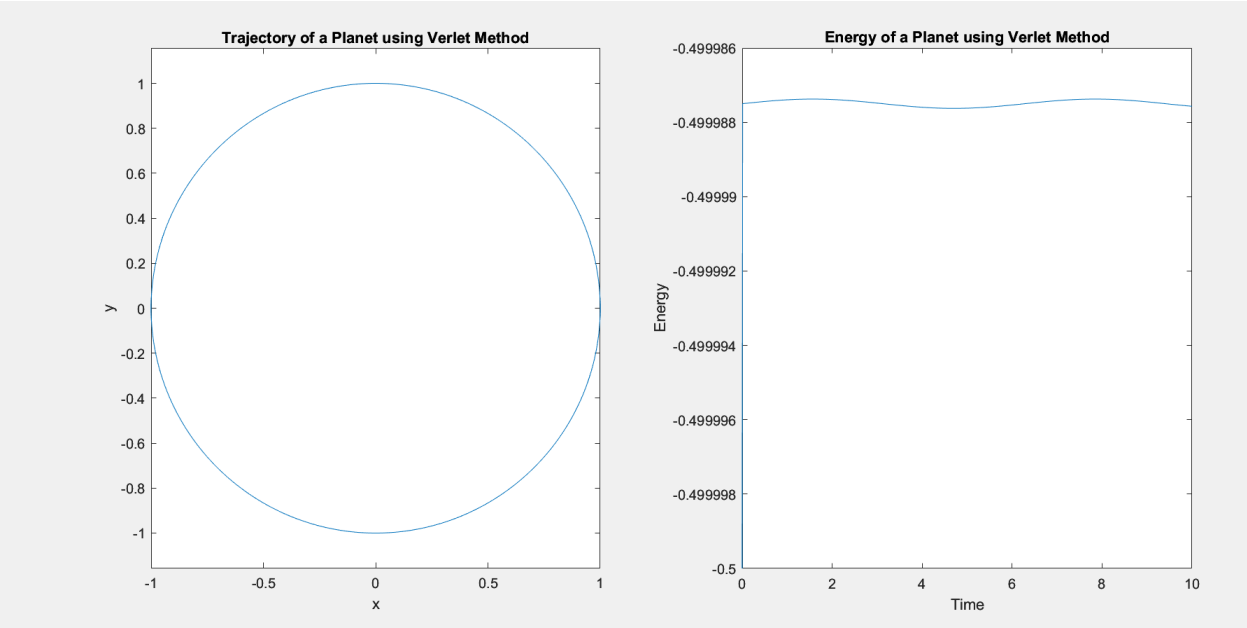
```
% Forward Euler
% Gravity, mass and position
G = 1;
m0 = 1;
r0 = [0 0];

% Initial position and velocity
r_init = [1 0];
v_init = [0 1];
y0 = [r_init v_init];

% span and time step
dt = 0.01;
tspan = 0:dt:10;

% Store
r = zeros(length(tspan), 2);
v = zeros(length(tspan), 2);

% Position and velocity
r(1,:) = r_init;
v(1,:) = v_init;

% Euler method
for k = 1:length(tspan)-1
    alpha = 1;
    r(k+1,:) = r(k,:) + v(k,:) * dt;
    v(k+1,:) = v(k,:) + alpha * dt * (-G * m0 * r(k,:) / norm(r(k,:))^3);
end

% Energy
E = -G*m0./vecnorm(r, 2, 2).^2 + 0.5*vecnorm(v, 2, 2).^2;

% Trajectory plot
plot(r(:,1), r(:,2));
xlabel('x');
ylabel('y');
title('Trajectory of a Planet using Forward Euler Method');
axis equal;

% Energy plot
figure;
plot(tspan, E);
xlabel('time');
ylabel('energy');
title('Energy of a Planet using Forward Euler Method');
```





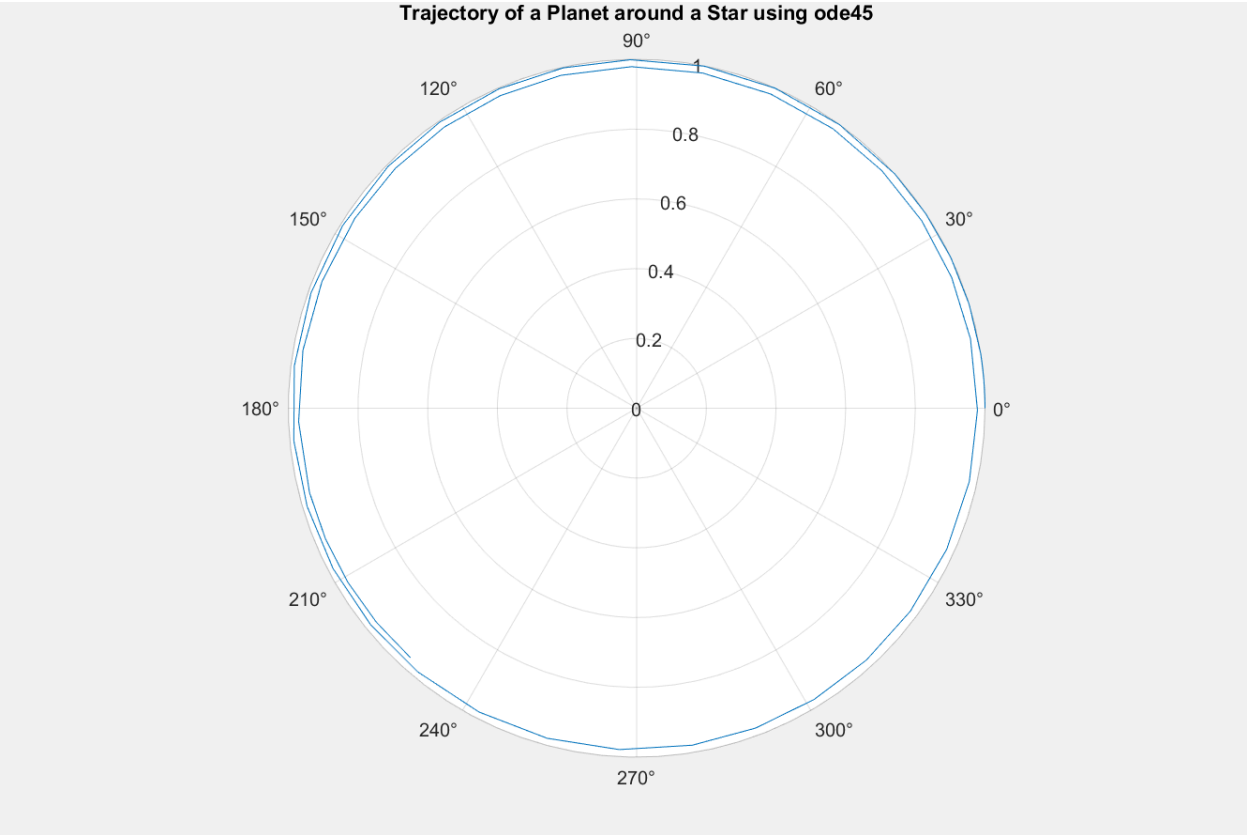
The forward Euler produces unstable orbits

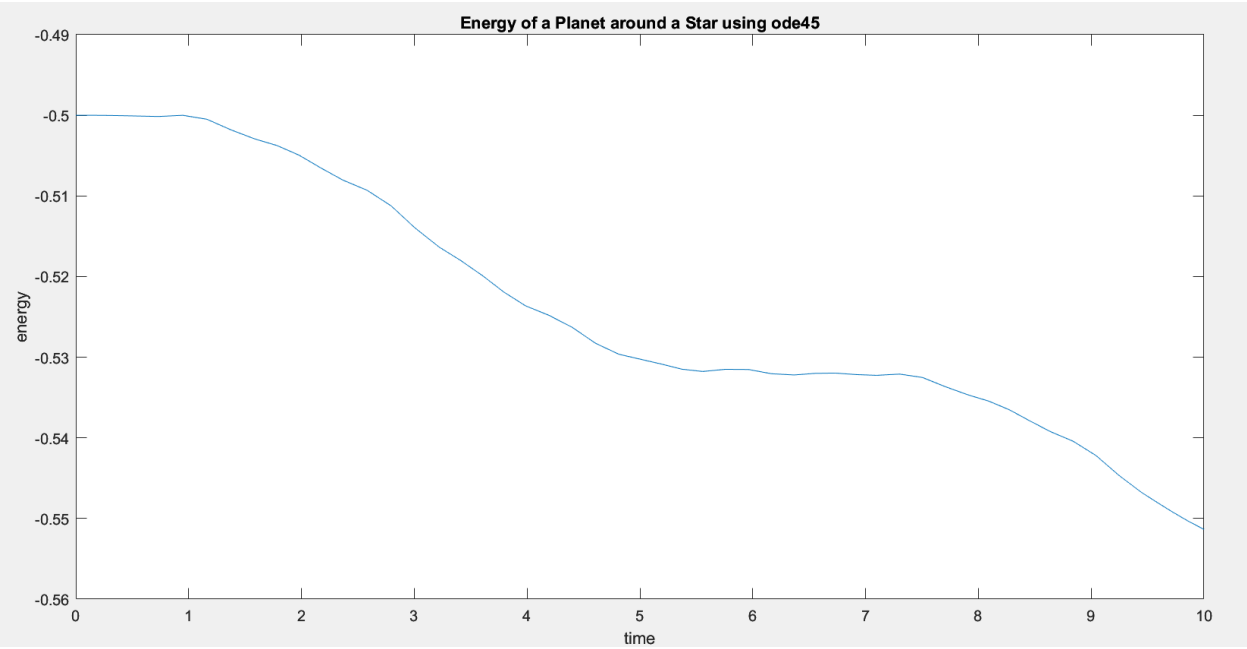
```
% Verlet
% Gravity, mass and position
G = 1;
m0 = 1;
r0 = [0 0];

% Initial position and velocity
r_init = [1 0];
v_init = [0 1];
y0 = [r_init v_init];

% span time step
dt = 0.01;
tspan = 0:dt:10;

% store
r = zeros(length(tspan), 2);
v = zeros(length(tspan), 2);

% Position and velocity
r(1,:) = r_init;
v(1,:) = v_init;


% Verlet's method
for k = 1:length(tspan)-1
    if k == 1
        r(k+1,:) = r(k,:) + dt*v(k,:) + dt^2/2 * (-G * m0 * r(k,:) / norm(r(k,:))^3);
    else
        r(k+1,:) = 2*r(k,:) - r(k-1,:) + dt^2 * (-G * m0 * r(k,:) / norm(r(k,:))^3);
    end
    v(k+1,:) = (r(k+1,:) - r(k,:)) / dt;
end

% Energy
E = zeros(length(tspan), 1);
for k = 1:length(tspan)
    r_k = r(k,:);
    v_k = v(k,:);
    E(k) = 0.5 * norm(v_k)^2 - G * m0 / norm(r_k);
end

% Plot planet and energy
subplot(1,2,1)
plot(r(:,1), r(:,2));
xlabel('x');
ylabel('y');
title('Trajectory of a Planet using Verlet Method');
axis equal;

subplot(1,2,2)
plot(tspan, E);
xlabel('Time');
ylabel('Energy');
title('Energy of a Planet using Verlet Method');
```



The Verlet method produces more stable orbits but the energy fluctuates around a constant value.

```
% Runge Kutta
% Gravity, mass and position
G = 1;
m0 = 1;
r0 = [0 0];

% Initial position and velocity
r_init = [1 0];
v_init = [0 1];
y0 = [r_init v_init];

% span
tspan = [0 10];

% ODE system
orbitODE = @(t,y) [y(3:4); -G*m0*y(1:2)/norm(y(1:2))^3];

% ode45
[t, y] = ode45(orbitODE, tspan, y0);

% position and velocity
r = y(:, 1:2);
v = y(:, 3:4);

% trajectory plot
polarplot(atan2(r(:,2), r(:,1)), vecnorm(r, 2, 2));
title('Trajectory of a Planet around a Star using ode45');

% energy plot
figure;
plot(t, -G*m0./vecnorm(r, 2, 2).^2 + 0.5*vecnorm(v, 2, 2).^2);
xlabel('time');
ylabel('energy');
title('Energy of a Planet around a Star using ode45');
```





Based on these graphs, I would use the adaptive Runge-Kutta method (ode45) for this problem, as it produces stable, accurate orbits and energy plots.

c)

```
% initial condition
N = 5;
mass = rand(N,1);
pos = rand(N,2);
vel = rand(N,2);

% integration parameters
tspan = [0 100];
dt = 0.01;

% animation
figure('Position',[100 100 800 600]);
ax = gca;
axis equal;
xlim([-1 1]);
ylim([-1 1]);
set(gca,'FontSize',20);
set(gcf,'color','w');
box on;
starsize = 200;
star = scatter(0,0,starsize,'*','markerfacecolor','y');
planetsize = 50;
planet = scatter(pos(:,1),pos(:,2),planetsize,mass,'filled');

% video writer
vidObj = VideoWriter('planetary_systempartc.mp4','MPEG-4');
vidObj.Quality = 100;
vidObj.FrameRate = 30;
open(vidObj);

% Integrate animate
for t = tspan(1):dt:tspan(2)
    % new positions of planets
    pos = pos + vel*dt + 0.5*accel(pos,mass)*dt^2;
    vel = vel + 0.5*(accel(pos,mass) + accel(pos + vel*dt,mass))*dt;

    % positions of objects
    set(planet,'XData',pos(:,1),'YData',pos(:,2));

    % new video frame
    currFrame = getframe(gcf);
    writeVideo(vidObj,currFrame);
end

close(vidObj);

function a = accel(pos,mass)
G = 1;
N = size(pos,1);
a = zeros(N,2);
for i = 1:N
    for j = 1:N
        if i ~= j
```

```
            r = pos(j,:) - pos(i,:);
            a(i,:) = a(i,:) + G*mass(j)/norm(r)^3*r;
        end
    end
end
end
```

CODE

3)

a)

```
% Gravity, mass and position
G = 1;
m0 = 1;
r0 = [0 0];

% Initial position, velocity and circular orbit
r_init = [1 0];
v_init = [0 1];
v_init = norm(r_init)*[0 1];
y0 = [r_init v_init];

% span
tspan = [0 10];

% ODE
orbitODE = @(t,y) [y(3:4); -G*m0*y(1:2)/norm(y(1:2))^3];

% ode45
[t, y] = ode45(orbitODE, tspan, y0);

% Position and velocity
r = y(:, 1:2);
v = y(:, 3:4);

% Energy
E = -G*m0./vecnorm(r, 2, 2).^2 + 0.5*vecnorm(v, 2, 2).^2;

% Trajectory plot
plot(r(:,1), r(:,2));
xlabel('x');
ylabel('y');
title('Trajectory of a Planet in Circular Orbit');
axis equal;

% Energy plot
figure;
plot(t, E);
xlabel('time');
ylabel('energy');
title('Energy of a Planet in Circular Orbit');
```

b)

```
% Forward Euler
% Gravity, mass and position
G = 1;
m0 = 1;
r0 = [0 0];

% Initial position and velocity
r_init = [1 0];
v_init = [0 1];
y0 = [r_init v_init];

% span and time step
dt = 0.01;
tspan = 0:dt:10;

% Store
r = zeros(length(tspan), 2);
v = zeros(length(tspan), 2);

% Position and velocity
r(1,:) = r_init;
v(1,:) = v_init;

% Euler method
for k = 1:length(tspan)-1
    alpha = 1;
    r(k+1,:) = r(k,:) + v(k,:) * dt;
    v(k+1,:) = v(k,:) + alpha * dt * (-G * m0 * r(k,:) / norm(r(k,:))^3);
end

% Energy
E = -G*m0./vecnorm(r, 2, 2).^2 + 0.5*vecnorm(v, 2, 2).^2;

% Trajectory plot
plot(r(:,1), r(:,2));
xlabel('x');
ylabel('y');
title('Trajectory of a Planet using Forward Euler Method');
axis equal;

% Energy plot
figure;
plot(tspan, E);
xlabel('time');
ylabel('energy');
title('Energy of a Planet using Forward Euler Method');
```

```
% Verlet
% Gravity, mass and position
G = 1;
m0 = 1;
r0 = [0 0];

% Initial position and velocity
r_init = [1 0];
v_init = [0 1];
y0 = [r_init v_init];

% span time step
dt = 0.01;
tspan = 0:dt:10;

% store
r = zeros(length(tspan), 2);
v = zeros(length(tspan), 2);

% Position and velocity
r(1,:) = r_init;
v(1,:) = v_init;


% Verlet's method
for k = 1:length(tspan)-1
    if k == 1
        r(k+1,:) = r(k,:) + dt*v(k,:) + dt^2/2 * (-G * m0 * r(k,:) / norm(r(k,:))^3);
    else
        r(k+1,:) = 2*r(k,:) - r(k-1,:) + dt^2 * (-G * m0 * r(k,:) / norm(r(k,:))^3);
    end
    v(k+1,:) = (r(k+1,:) - r(k,:)) / dt;
end

% Energy
E = zeros(length(tspan), 1);
for k = 1:length(tspan)
    r_k = r(k,:);
    v_k = v(k,:);
    E(k) = 0.5 * norm(v_k)^2 - G * m0 / norm(r_k);
end

% Plot planet and energy
subplot(1,2,1)
plot(r(:,1), r(:,2));
xlabel('x');
ylabel('y');
title('Trajectory of a Planet using Verlet Method');
axis equal;

subplot(1,2,2)
plot(tspan, E);
xlabel('Time');
ylabel('Energy');
title('Energy of a Planet using Verlet Method');
```

```
% Runge Kutta
% Gravity, mass and position
G = 1;
m0 = 1;
r0 = [0 0];

% Initial position and velocity
r_init = [1 0];
v_init = [0 1];
y0 = [r_init v_init];

% span
tspan = [0 10];

% ODE system
orbitODE = @(t,y) [y(3:4); -G*m0*y(1:2)/norm(y(1:2))^3];

% ode45
[t, y] = ode45(orbitODE, tspan, y0);

% position and velocity
r = y(:, 1:2);
v = y(:, 3:4);

% trajectory plot
polarplot(atan2(r(:,2), r(:,1)), vecnorm(r, 2, 2));
title('Trajectory of a Planet around a Star using ode45');

% energy plot
figure;
plot(t, -G*m0./vecnorm(r, 2, 2).^2 + 0.5*vecnorm(v, 2, 2).^2);
xlabel('time');
ylabel('energy');
title('Energy of a Planet around a Star using ode45');
```

```
% initial condition
N = 5;
mass = rand(N,1);
pos = rand(N,2);
vel = rand(N,2);

% integration parameters
tspan = [0 100];
dt = 0.01;

% animation
figure('Position',[100 100 800 600]);
ax = gca;
axis equal;
xlim([-1 1]);
ylim([-1 1]);
set(gca,'FontSize',20);
set(gcf,'color','w');
box on;
starsize = 200;
star = scatter(0,0,starsize,'*','markerfacecolor','y');
planetsize = 50;
planet = scatter(pos(:,1),pos(:,2),planetsize,mass,'filled');

% video writer
vidObj = VideoWriter('planetary_systempartc.mp4','MPEG-4');
vidObj.Quality = 100;
vidObj.FrameRate = 30;
open(vidObj);

% Integrate animate
for t = tspan(1):dt:tspan(2)
    % new positions of planets
    pos = pos + vel*dt + 0.5*accel(pos,mass)*dt^2;
    vel = vel + 0.5*(accel(pos,mass) + accel(pos + vel*dt,mass))*dt;

    % positions of objects
    set(planet,'XData',pos(:,1),'YData',pos(:,2));

    % new video frame
    currFrame = getframe(gcf);
    writeVideo(vidObj,currFrame);
end

close(vidObj);

function a = accel(pos,mass)
G = 1;
N = size(pos,1);
a = zeros(N,2);
for i = 1:N
    for j = 1:N
        if i ~= j
            r = pos(j,:) - pos(i,:);
            a(i,:) = a(i,:) + G*mass(j)/norm(r)^3*r;
```

```
        end
    end
end
end
```