EAS4510
Homework #0

Samantha Harris

January 17, 2020

1 Question 1

Consider the following second-order ordinary differential equation:

$$\ddot{x}(t) + \omega_n^2 x(t) = 0$$

(a) Derive the analytic solution given the initial conditions $(x(0), \dot{x}(0)) = (x_0, v_0)$.

Step 1. General Solution.
The functions $cos(\omega_n t)$ and $sin(\omega_n t)$ are solutions of the homogeneous linear ordinary differential equation. This can be shown by differentiation and substitution. By moving the second term of the differential equation to the other side, we can see the following is true,

$$\ddot{x}(t) = -\omega_n^2 x(t).$$

Therefore, any solution to this differential equation must follow the above format.

Showing $x(t) = cos(\omega_n t)$ is a solution to the differential equation,

$$x(t) = cos(\omega_n t)$$

$$\dot{x}(t) = -\omega_n sin(\omega_n t)$$

$$\ddot{x}(t) = -\omega_n^2 cos(\omega_n t)$$

$$\therefore\ \ddot{x}(t) = -\omega_n^2 x(t).$$

Showing $x(t) = sin(\omega_n t)$ is a solution to the differential equation,

$$x(t) = sin(\omega_n t)$$

$$\dot{x}(t) = \omega_n cos(\omega_n t)$$

$$\ddot{x}(t) = -\omega_n^2 sin(\omega_n t)$$

$$\therefore\ \ddot{x}(t) = -\omega_n^2 x(t).$$

Step 2. Particular Solution.
Due to the Fundamental Theorem for the Homogeneous Linear ODE, which states,

"For a homogeneous linear ODE (2), any linear combination of two solutions on an open interval I is again a solution of (2) on I. In particular, for such an equation, sums and constant multiples of solutions are again solutions."

the solution can be put in the form, $x(t) = c_1 cos(\omega_n t) + c_2 sin(\omega_n t)$. By using our initial conditions, we can find the constants c_1 and c_2.

$$x_0 = x(0) = c_1 cos(0) + c_2 sin(0)$$

$$c_1 = x_0$$

$$v_0 = \dot{x}(0) = -c_1 \omega_n sin(0) + c_2 \omega_n cos(0)$$

$$v_0 = c_2 \omega_n$$

$$c_2 = \frac{v_0}{\omega_n}$$

Therefore, the solution to the second-order ordinary differential equation is,

$$x(t) = x_0 cos(\omega_n t) + \frac{v_o}{\omega_n} sin(\omega_n t).$$

To check our answer, we can use the same process as in Step 1, using differentiation and substitution,

$$x(t) = x_0 cos(\omega_n t) + \frac{v_o}{\omega_n} sin(\omega_n t)$$

$$\dot{x}(t) = -\omega_n x_0 sin(\omega_n t) + v_0 cos(\omega_n t)$$

$$\ddot{x}(t) = -\omega_n^2 x_0 cos(\omega_n t) - \omega_n v_0 sin(\omega_n t)$$

$$\ddot{x}(t) = -\omega_n^2 (x_0 cos(\omega_n t) + \frac{v_o}{\omega_n} sin(\omega_n t))$$

$$\therefore \ \ddot{x}(t) = -\omega_n^2 x(t).$$

Used *Advanced Engineering Mathematics* by Erwin Kreyszig pg 49 as a reference.

(b) Set up the differential equation as a system of two first-order equations using $x(t)$ and $\dot{x}(t) = v(t)$ as state variables.

Using $x(t)$ and $\dot{x}(t) = v(t)$ as state variables, the state equations are:

$$\dot{x}(t) = v(t)$$

$$\dot{v}(t) = -\omega_n^2 x(t)$$

Used *Optimal Control Theory An Introduction* By Donald E Kirk pg 5 as a reference.

(c) The MATLAB ordinary differential equation solver ODE113 was used to solve the differential equation on the time interval $[0,\, 2\pi/\omega_n]$ with the initial conditions

$$(x(0), \dot{x}(0)) = (x_0, v_0) = (1, 0)$$

$$(x(0), \ddot{x}(0)) = (x_0, v_0) = (0, 1)$$

The ODE113 function integrated the system of differential equations (state equations) from the initial time to the final time using the initial conditions. The column vector labeled 'tout' in my code[1], holds the time steps starting from 0 and ending at $2\pi/\omega_n$. The 'pout' matrix corresponds to the x and v values at the time steps of 'tout'. Where x(t) and v(t) are position and velocity. The results were plotted as position vs. time on one figure, and velocity vs. time on another for each set of initial conditions. By plugging in $\omega_n = 1$ and the initial conditions for $x(t)$, we can determine whether our plots make sense according to our analytic solution.

After plugging in the initial conditions $(x_0, v_0) = (1, 0)$ to our position function,

$$x(t) = (1)cos(t) + (0)sin(t)$$
$$x(t) = cos(t)$$

we can see that $x(t)$ can be described by a cosine wave. This is seen in figure 1. Using the same logic, we can understand the position vs time plot with initial conditions $(x_0, v_0) = (0, 1)$. By using these initial conditions,

$$x(t) = (0)cos(t) + (1)sin(t)$$
$$x(t) = sin(t)$$

we can see that $x(t)$ can be described by a sine wave. This is seen in figure 2.

Now looking at the velocity vs time plot with initial conditions $(x_0, v_0) = (1, 0)$, and $\omega_n = 1$,

$$v(t) = -(1)sin(t) + (0)cos(t)$$
$$v(t) = -sin(t)$$

we can see that $v(t)$ can be described by a negative sine wave. This is seen in figure 3. Using the same logic, we can understand the velocity vs time plot with initial conditions $(x_0, v_0) = (0, 1)$. Using these initial conditions,

$$v(t) = -(0)sin(t) + (1)cos(t)$$

[1]The code is shown on page 8.

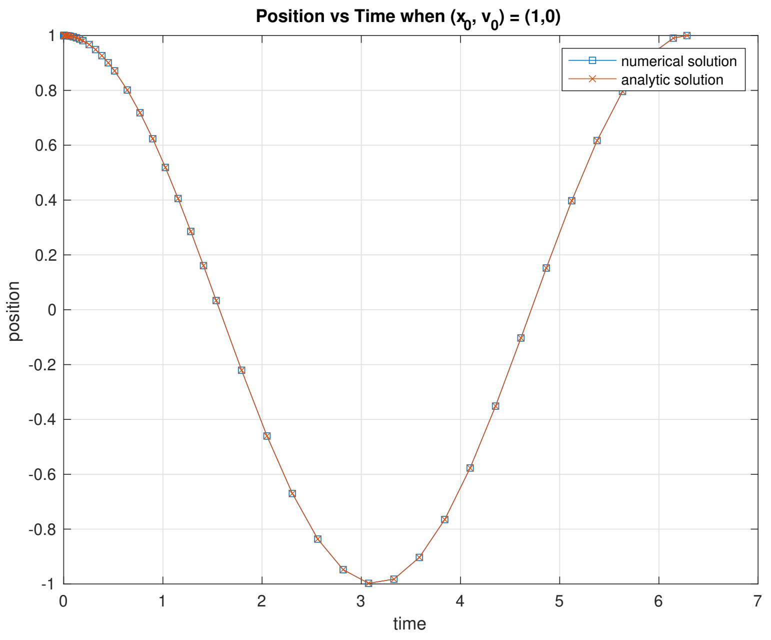

Figure 1: Plot of position vs time when $(x_0, v_0) = (1, 0)$.

$$v(t) = cos(t)$$

we can see that $v(t)$ can be described by a cosine wave. This is seen in figure 4.

Note: The $\frac{1}{\omega_n}$ coefficient doesn't seem to have an effect on the function, only because ω_n was assigned to equal 1. If $\omega_n \neq 1$ the frequency of oscillation would be effected.

(d) On each graph, the analytic solution is plotted alongside the numerical solution. It looks as if each solution is the same, but when zoomed in it can be seen that they are slightly off. This is because the numerical solution is an approximation to the solution, it is not an exact solution. Since the function was fairly simple, we were able to get very close to the exact solution, which is why it looks as if they are the same function.

2 Question 2

The following relationship,

$$\ddot{x} = \dot{x}\frac{d\dot{x}}{dx}$$

can be shown by,

$$\ddot{x} = \frac{d\dot{x}}{dt} = \frac{d\dot{x}}{dx}\frac{dx}{dt}$$

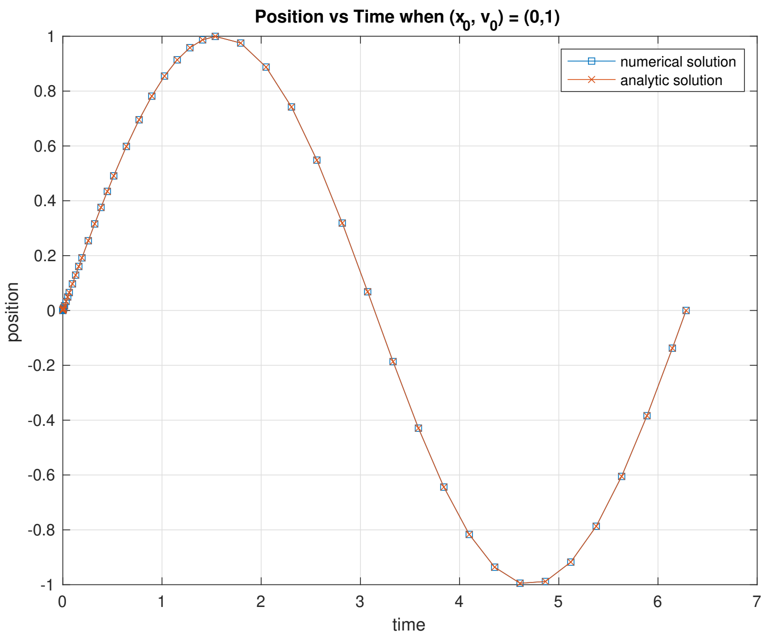

Figure 2: Plot of position vs time when $(x_0, v_0) = (0, 1)$.

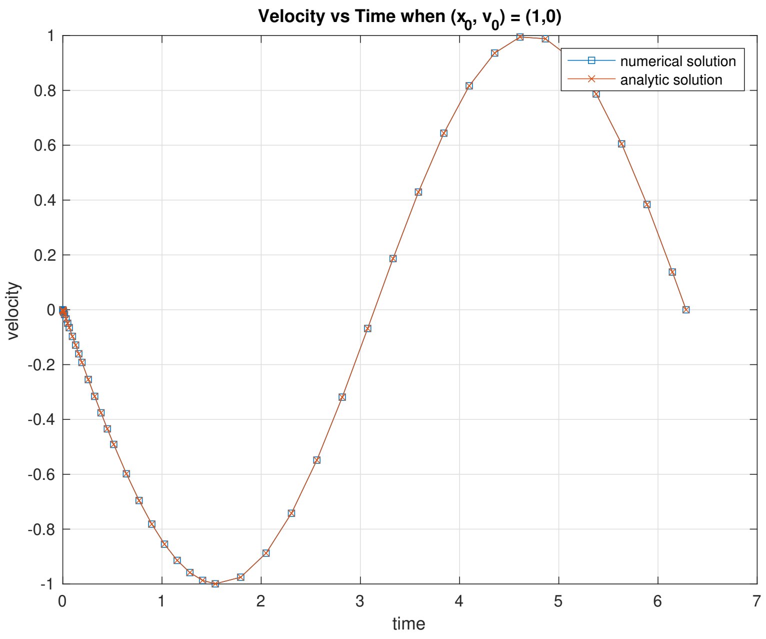

Figure 3: Plot of velocity vs time when $(x_0, v_0) = (1, 0)$.

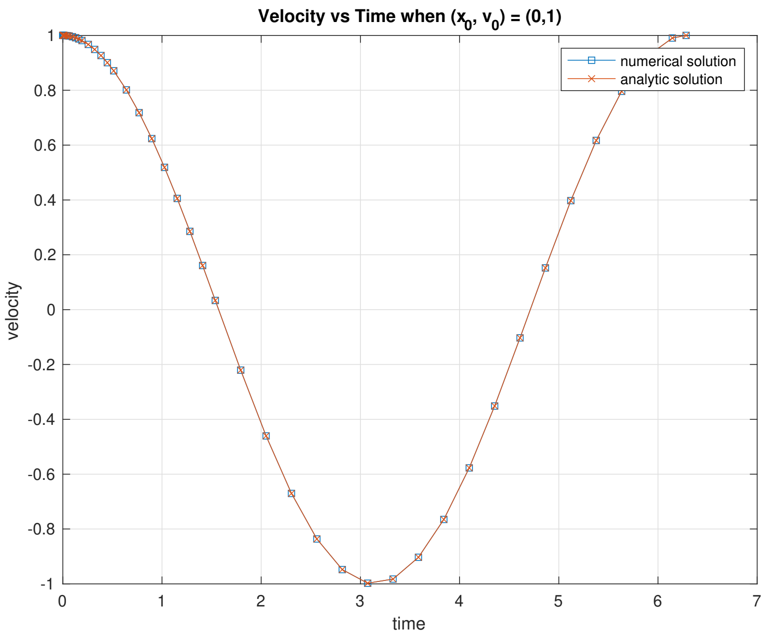

Figure 4: Plot of velocity vs time when $(x_0, v_0) = (0, 1)$.

$$\ddot{x} = \dot{x}\frac{d\dot{x}}{dx}.$$

We can rewrite the differential equation that was in Question 1, into the above format,

$$\dot{x}\frac{d\dot{x}}{dx} + \omega_n^2 x = 0.$$

By solving for $\dot{x}$ as a function of x we get,

$$\dot{x}\frac{d\dot{x}}{dx} = -\omega_n^2 x$$

$$\dot{x}d\dot{x} = -\omega_n^2 x(t)dx$$

$$\int \dot{x}d\dot{x} = -\omega_n^2 \int x(t)dx$$

$$\frac{\dot{x}^2}{2} = -\omega_n^2 \frac{x^2}{2} + C$$

$$\dot{x}^2 = -\omega_n^2 x^2 + C.$$

Using the initial conditions

$$(x(0), \dot{x}(0)) = (x_0, v_0)$$

to solve for C,

$$C = v_0^2 + \omega_n^2 x_0^2.$$

As we can see, the above equation is the equation of an ellipse. In this example, $\omega_n = 1$, so graphing $v(t)$ against $x(t)$ will be a circle, as seen in figures 5 and 6. In both cases, the circle has a radius of 1. This is because for each set of initial conditions, $C = 1$. The square root of C determines the radius, and the square root of 1 is 1. Due to the fact that they are both circles with radius 1, both plots should be the same. Therefore, when plotted on top of each other, they line up as seen in figure 7.

3 Code for Questions 1 and 2

The script that covers questions 1 and 2 is below,

```
%-------------------------------------------------------------
%                                Question 1
%
%Use the matlab ordinary differential equation solver ODE113 to solve
%         x"(t) + (omega^2)*x(t) = 0
%
% omega = 1
% t_initial = 0
% t_final = 2*pi/omega
%
% initial conditions:
%  (i)  (x(0), x'(0)) = (xo, vo) = (1,0)
%  (ii) (x(0), x'(0)) = (xo, vo) = (0,1)
```

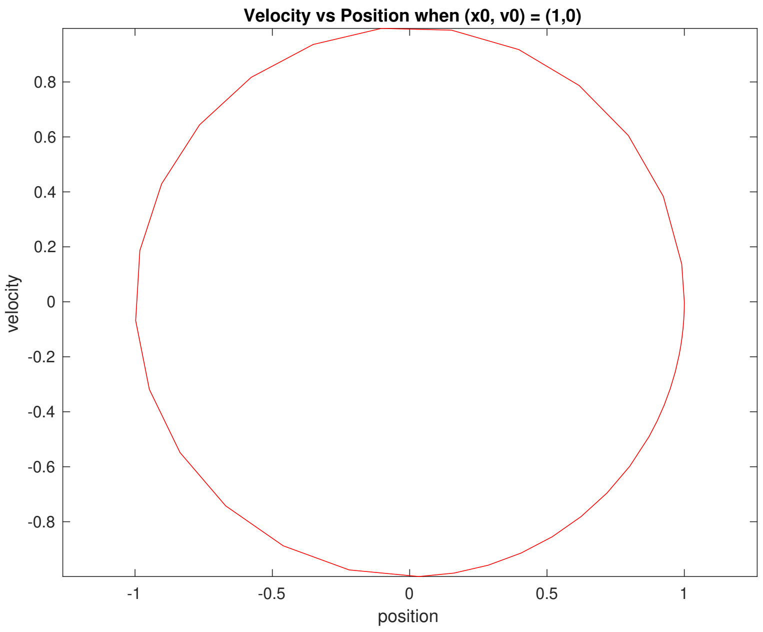

Figure 5: Velocity vs position plot for the initial conditions $(x_0, v_0) = (0, 1)$.

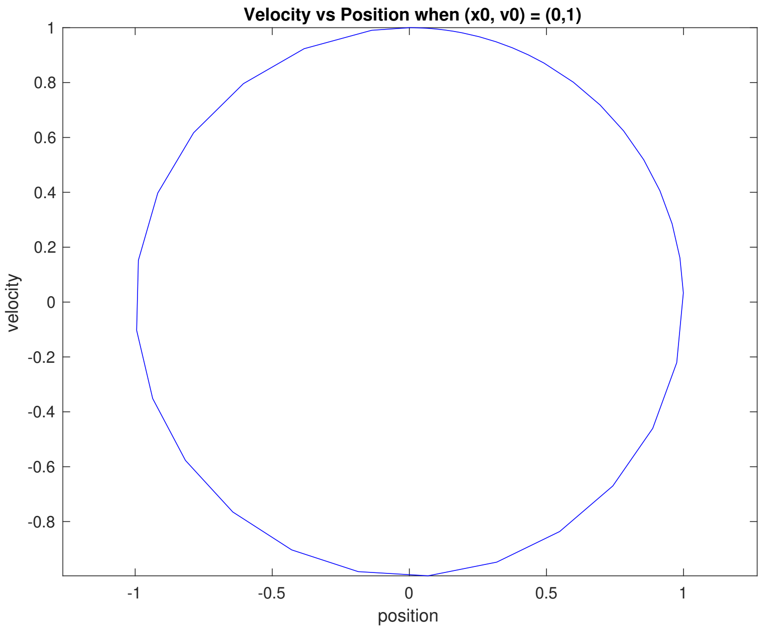

Figure 6: Velocity vs. position plot for the initial conditions $(x_0, v_0) = (1, 0)$.

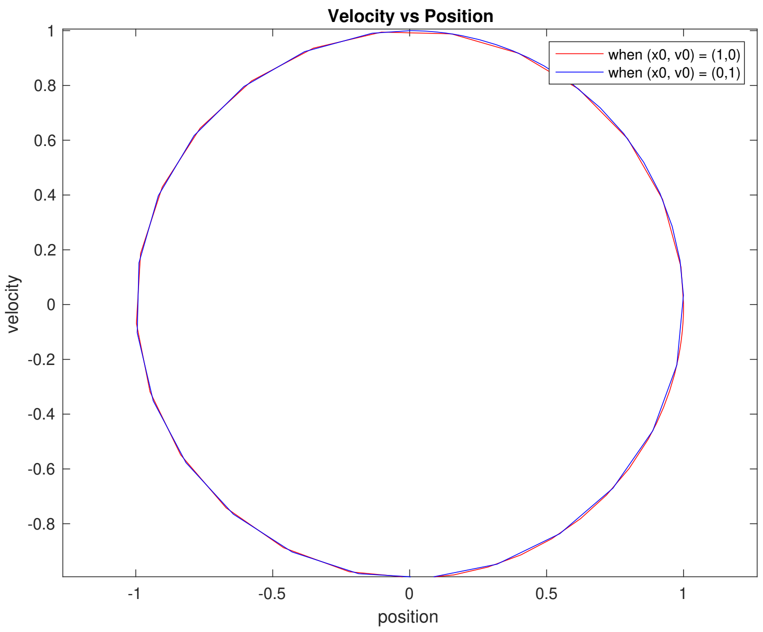

Figure 7: Velocity vs Position for initial conditions $(x_0, v_0) = (0, 1)$ and $(x_0, v_0) = (1, 0)$.

```
%
% Plot the results obtained from the initial conditions above using matlab.
% When making the plots, put x(t) vs t on the figure and put x(t) vs t on a
% second figure
%
% Plot the results alongside the analytic solution
%
%                               Question 2
%
% Plot the solution x' as a function of x for the following initial
% conditions
%
%   (a)  (x(0), x'(0)) = (x0, v0) = (1,0)
%   (b)  (x(0), x'(0)) = (x0, v0) = (0,1)
%
% Make a single plot that captures both sets of initial conditions given in
% (a) and (b). Use the MATLAB legend command to label each plot with the
% appropriate set of initial conditions
%-------------------------------------------------------------

    %delete all previously produced graphs
    close all
    clear all
    clc
```

```
%value of omegan (natural frequency)
omegan = 1;

%value of time
t = 1;

%start and end times
t_initial = 0;
t_final = 2*pi;
%bounds of integration
tspan = [t_initial, t_final];

%set the relative error tolerance
options = odeset('reltol', 1e-8);

%(i)
    %initial conditions of x0 = 1 and v0 = 0
    x0 = 1;
    v0 = 0;
    p0 = [x0;v0];

    % In this code, it is shown how to find the Analytic solution using
    % a function handle, or a function. The use of a function is
    % implemented in the code, but the way in which to use of a function
    % handle is included to provide an example for future reference.
    %-------------------
    % Notes about using the function handle:
    % This function computes the analytic solution of
    %       x"(t) + (omega^2)*x(t) = 0
    %
    %   inputs
    %       t = time
    %       x0 = initial condition x(0)
    %       v0 = initial condition v(0)
    %       omegan = natural frequency
    %
    %       where
    %           x = function that describes position
    %           v = function that describes velocity
    %   outputs
    %       xout = value of position at a certain time
    %       vout = value of velocity at a certain time
    %-------------------
    xAnalytic = @(t,x0,v0,omegan) x0*cos(omegan*t) + (v0/omegan)*sin(omegan*t);
```

```
   vAnalytic = @(t,x0,v0,omegan) -x0*omegan*sin(omegan*t) + v0*cos(omegan*t);

   %Numerical Solution
   [tout,pout] = ode113(@harmonicOscillator, tspan, p0, options, omegan);

   %Analytic Solution using handle function
   %using the function handles xAnalytic and vAnalytic
   %xout is the value of x at each time step in tout
   xout = xAnalytic(tout, x0, v0, omegan);
   %vout is the value of v at each time step in tout
   vout = vAnalytic(tout, x0, v0, omegan);

   %Analytic Soliution using function
   pExact1 = exactSolution(tout,p0,omegan);

   % Plot of x(t) vs t when initial conditions are [1,0]
   % if we were to use the analytic solution using the handle function
   % we would replace pExact1(:1) with xout and pExact1(:2) with vout
   figure;
   plot(tout,pout(:,1),'-s',tout,pExact1(:,1),'-x')
   title('Position vs Time when (x_0, v_0) = (1,0) ');
   xlabel('time');
   ylabel('position');
   legend('numerical solution','analytic solution');
   grid on



   % v(t) vs t when initial conditions are [1,0]
   figure;
   plot(tout,pout(:,2),'-s',tout,pExact1(:,2),'-x')
   title('Velocity vs Time when (x_0, v_0) = (1,0)');
   xlabel('time');
   ylabel('velocity');
   legend('numerical solution','analytic solution');
   grid on

   %keyboard (stops code at this point)

%(ii)
   %initial conditions of x0 = 0 and v0 = 1
   x0 = 0;
   v0 = 1;
   p0 = [x0;v0];
```

```
%Numerical Solution
[tout,pout] = ode113(@harmonicOscillator, tspan, p0, options, omegan);

%Analytic Solution
pExact2 = exactSolution(tout,p0,omegan);

% x(t) vs. t when initial conditions are [1,0]
figure;
plot(tout,pout(:,1),'-s',tout,pExact2(:,1),'-x')
title('Position vs Time when (x_0, v_0) = (0,1) ');
xlabel('time');
ylabel('position');
legend('numerical solution','analytic solution');
grid on

% v(t) vs t when initial conditions are [1,0]
figure;
plot(tout,pout(:,2),'-s',tout,pExact2(:,2),'-x')
title('Velocity vs Time when (x_0, v_0) = (0,1)');
xlabel('time');
ylabel('velocity');
legend('numerical solution','analytic solution');
grid on


% Plot the solution x' as a function of x for the initial
% conditions  (x(0), x'(0)) = (x0, v0) = (1,0)
figure;
plot(pExact1(:,1), pExact1(:,2), 'r')
title('Velocity vs Position when (x0, v0) = (1,0)');
xlabel('position');
ylabel('velocity');
axis equal

% Plot the solution x' as a function of x for the initial
% conditions  (x(0), x'(0)) = (x0, v0) = (0,1)
figure;
plot(pExact2(:,1), pExact2(:,2), 'b')
title('Velocity vs Position when (x0, v0) = (0,1)');
xlabel('position');
ylabel('velocity');
axis equal

%A single plot that captures x' vs x for both sets of initial conditions
figure;
```

```
        plot(pExact1(:,1), pExact1(:,2), 'r', pExact2(:,1), pExact2(:,2), 'b')
        title('Velocity vs Position');
        xlabel('position');
        ylabel('velocity');
        legend('when (x0, v0) = (1,0)','when (x0, v0) = (0,1)');
        axis equal
```

The code for the function harmonicOscillator is seen below,

```
function pdot = harmonicOscillator(t,p, omegan)

%This code computes the right hand side of the system of differential equations
%           x'(t) = v(t)
%           v'(t) = -omegan^2 * x(t)
%   inputs
%       t = time
%       p = [x;v]
%       omegan = natural frequency
%   where
%       x = position
%       v = velocity
%

%pulling out the position and velocity from the column vector p
x = p(1);
v = p(2);
%state equations
xdot = v;
vdot = -omegan^2 * x;
%the state equations as a column vector
pdot = [xdot;vdot];
end
```

The code for the function exactSolution is seen below,

```
function p = exactSolution(t, p0,omegan)

%This code computes the exact solution to the differential equation
%           x''(t) + omega^2 * x(t) = 0
```

```
%
%   inputs
%       t = time
%       p0 = [x;v]
%       omegan = natural frequency
%   where
%       x = position
%       v = velocity
%

% pulling out the initial conditions of x and v from the column vector p0
x0 = p0(1);
v0 = p0(2);

%filling in the first column of p with the value of x
p(:,1) = x0*cos(omegan*t) + (v0/omegan)*sin(omegan*t);
%filling in the second column of p with the value of v
p(:,2) = -x0*omegan*sin(omegan*t) + v0*cos(omegan*t);
end
```

4 Question 3

In this question, the goal was to solve the nonlinear algebraic equation,

$$x - asin(x) + C = 0$$

Where a and C are constants and $0 \leq a < 1$. The tables below show the equation being solved numerically using Newton's method (Tables 1-4) and fixed point iteration (Tables 5-8) for a set of initial guesses. After looking at each table, it is clear that Newton's Method converges in less iterations than fixed point iteration. This shows us that Newton's method is a quicker method to finding the solution to this particular equation.

4.1 Newton's Method

Newton's method produces a sequence of iterates that are meant to converge to the root of an equation. To start, we need an initial guess that is assumed to be in the vicinity of the root. Since the initial guess is assumed to be close to the root, we can approximate the curve of $x - asin(x) + C$ in the vicinity of its root by constructing its tangent line at the initial guess. Then use the root of this tangent line to get a new approximation of the root of our equation. The iterations were found using,

$$x^{(k+1)} = x^k - g(x^{(k)})$$

k	x^k	x^{k+1}
0	0.00000000	6.61318552
1	6.61318552	-1.66844728
2	-1.66844728	1.94535669
3	1.94535669	3.04525517
4	3.04525517	2.88527842
5	2.88527842	2.88456509
6	2.88456509	2.88456507
7	2.88456507	2.88456507
8	2.88456507	2.88456507
9	2.88456507	2.88456507

Table 1: Newton's method with the Initial guess of $x = 0$. The value of x that solves this equation is 2.8845650701.

k	x^k	x^{k+1}
0	1.57079633	3.32186476
1	3.32186476	2.88334132
2	2.88334132	2.88456514
3	2.88456514	2.88456507
4	2.88456507	2.88456507
5	2.88456507	2.88456507
6	2.88456507	2.88456507
7	2.88456507	2.88456507
8	2.88456507	2.88456507
9	2.88456507	2.88456507

Table 2: Newton's method with the initial guess of $\pi/2$. The value of x that solves this equation is 2.8845650701.

Where

$$g(x^{(k)}) = \frac{f(x^{(k)})}{f'(x^{(k)})}$$

and

$$f(x^{(k)}) = g(x^{(k)}).$$

4.2 Fixed Point Iteration

The fixed point iteration method will also produce a sequence of iterates that are meant to converge to the root of the equation. The iterative procedure for this method has the form,

k	x^k	x^{k+1}
0	3.14159265	2.88560766
1	2.88560766	2.88456512
2	2.88456512	2.88456507
3	2.88456507	2.88456507
4	2.88456507	2.88456507
5	2.88456507	2.88456507
6	2.88456507	2.88456507
7	2.88456507	2.88456507
8	2.88456507	2.88456507
9	2.88456507	2.88456507

Table 3: Newton's method with the initial guess of π. The value of x that solves this equation is 2.8845650701.

k	x^k	x^{k+1}
0	4.71238898	2.14915427
1	2.14915427	2.96499773
2	2.96499773	2.88480795
3	2.88480795	2.88456507
4	2.88456507	2.88456507
5	2.88456507	2.88456507
6	2.88456507	2.88456507
7	2.88456507	2.88456507
8	2.88456507	2.88456507
9	2.88456507	2.88456507

Table 4: Newton's method with the initial guess of $3\pi/2$. The value of x that solves this equation is 2.8845650701.

k	x^k	x^{k+1}
0	0.00000000	2.73550952
1	2.73550952	2.96712807
2	2.96712807	2.83728957
3	2.83728957	2.91119817
4	2.91119817	2.86941054
5	2.86941054	2.89314166
6	2.89314166	2.87969592
7	2.87969592	2.88732455
8	2.88732455	2.88299962
9	2.88299962	2.88545264

Table 5: Fixed point method with the initial guess 0. The value of x that solves this equation is 2.8854526440

k	x^k	x^{k+1}
0	1.57079633	3.32186476
1	3.32186476	2.63037762
2	2.63037762	3.02237637
3	3.02237637	2.80524714
4	2.80524714	2.92902996
5	2.92902996	2.85921031
6	2.85921031	2.89889414
7	2.89889414	2.87642413
8	2.87642413	2.88917676
9	2.88917676	2.88194824

Table 6: Fixed point method with the initial guess of $\pi/2$. The value of x that solves this equation is 2.8819482380.

$$x^{(k+1)} = g(x^{(k)})$$

where

$$g(x^{(k)}) = asin(x) - C.$$

This process is started with an initial guess of what value of x would satisfy the equation $x = asin(x) + C$. Plugging the initial guess into the right side of the equation results in the next x iteration. This process continues until the two sides of the equation converge. This point of convergence is the value of x that satisfies the equation.

Using *An Introduction to Numerical Analysis* by Kendall E. Atkinson as a reference.

4.3 Code For Question 3

The script that includes Question 3 is as follows,

k	x^k	x^{k+1}
0	3.14159265	2.73550952
1	2.73550952	2.96712807
2	2.96712807	2.83728957
3	2.83728957	2.91119817
4	2.91119817	2.86941054
5	2.86941054	2.89314166
6	2.89314166	2.87969592
7	2.87969592	2.88732455
8	2.88732455	2.88299962
9	2.88299962	2.88545264

Table 7: Fixed point method with the initial guess π. The value of x that solves this equation is 2.8854526440.

k	x^k	x^{k+1}
0	4.71238898	2.14915427
1	2.14915427	3.22650077
2	3.22650077	2.68578300
3	2.68578300	2.99361691
4	2.99361691	2.82195957
5	2.82195957	2.91975304
6	2.91975304	2.86452205
7	2.86452205	2.89590063
8	2.89590063	2.87812731
9	2.87812731	2.88821277

Table 8: Fixed point method with the initial guess $3\pi/2$. The value of x that solves this equation is 2.8882127679.

```
%-----------------------------------------------------------
% Question 3
%
% Determine the procedure to solve the nonlinear algebraic equation
%       x - asin(x) + C = 0
% using (a) Newton's Method of the form
%       x^(k^1) = x^(k) + g((x^k))
% and (b) fixed point iteration of the form
%       x^(k+1) = g(x^(k))
%
%For Newton's method and a fixed point iteration, write MATLAB codes that
%solve the algebraic equation for x with the values
%   a = 0.5863552428728520
%   C = -2.735509517401657
% and the following initial guesses
%   x(0) = (0,pi/2,pi,3pi/2)
%
% For each initial guess tabulate the first ten iterations of your results
% and display in a separate output line the values of x that solves the
% algebraic equation
%-------------------------------------------------------------

%values for the constants
a = 0.5863552428728520;
C = -2.735509517401657;

%number of iterations we are taking
N = 10;

%we wish to find a value x so that x = asinx - C
g = @(x) a*sin(x) - C

%for Newton's method we need the function and it's derivative
algebraEq = @(x) x - a*sin(x) + C;
algebraEqDerivative = @(x) 1 - a*cos(x);

%   THIS FOR LOOP IS FOR NEWTON'S METHOD

%array of initial guesses
initGuess = [0, pi/2 , pi, 3*pi/2];

fprintf('NEWTWON''S METHOD')
%this for loop goes through each initial guess
for i = 1:length(initGuess)
    x = initGuess(i);
    initialGuess = x
    % this for loop goes through 10 iterations to find the root of the equation
    % using Newton's method

    for m = 0:N-1
        xprevious = x;
        x = x - algebraEq(x)/algebraEqDerivative(x);
        %print the iteration number, the input x and the output x
        fprintf('%12i \t %12.8f \t %12.8f\n',m,xprevious,x);

    end

    %print the x value of the last iteration
    fprintf('\nThe value of x that solves this equation is %6.10f\n', x);
end


%   THIS FOR LOOP IS FOR A FIXED POINT ITERATION


fprintf('\n')
fprintf('\nFIXED POINT ITERATION')


%this loop goes through each initial guess
initGuess = [0, pi/2 , pi, 3*pi/2];
for i = 1:length(initGuess)
    x = initGuess(i);
    initialGuess = x
    %this loop goes through 10 iterations to find the root of the equation
    %using fixed point iteration
    for m = 0:N-1
        xprevious = x;
        x = g(x);
        %print the iteration number, the input x and the output x
        fprintf('%12i \t %12.8f \t %12.8f\n',m,xprevious,x);
    end
    %print the x value of the last iteration
    fprintf('\nThe value of x that solves this equation is %6.10f\n', x);
end
```

5 Question 4

The following second-order ordinary differential equation,

$$\ddot{\mathbf{r}} + \frac{\mu}{||\mathbf{r}||^3}\mathbf{r} = 0$$

can be put into first-order form using $\mathbf{r}(t)$ and $\dot{\mathbf{r}}(t) = \mathbf{v}(t)$ as,

$$\dot{\mathbf{r}}(t) = \mathbf{v}(t)$$

$$\dot{\mathbf{v}}(t) = \frac{\mu}{||\mathbf{r}||^3}\mathbf{r}$$

The first-order representation is solved numerically from $t = 0$ to $t = 38032$ using the MATLAB ordinary differential equation solver ode113 with a relative error tolerance of 10^{-6} using $\mu = 398600$ and the following initial conditions:

$$\mathbf{r}^T(0) = [6997.56 \quad -34108.00 \quad 20765.49]$$
$$\mathbf{v}^T(0) = [0.15599 \quad 0.25517 \quad 1.80763]$$

The plot for $\mathbf{r}(t) = (x(t), y(t), z(t))$ is seen in figure 8. The equation,

$$\ddot{\mathbf{r}} + \frac{\mu}{||\mathbf{r}||^3}\mathbf{r} = 0$$

is called the two-body differential equation. The two-body problem consists of a spacecraft in motion relative to a planet. The solution to this equation describes the motion of a spacecraft under the influence of central-body gravitational force from the planet. The solution in of the two-body problem tells us the position of the spacecraft as a function of time. Due to the fact that position as a function of time is now known, we can then take it's derivatives and find velocity and acceleration of the spacecraft. The points on figure 8 show the possible positions of the spacecraft under the influence of a central-body gravitation and the arrows show the direction of the velocity of the spacecraft at a certain position.

5.1 Code for Question 4

The script that includes question 4 is shown below.

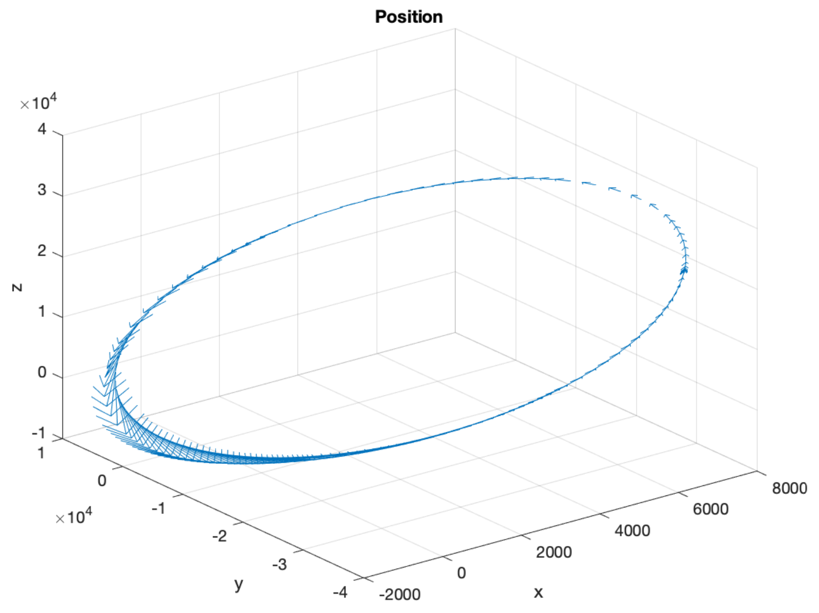

Figure 8: Graph describing the motion of a spacecraft under the influence of a central-body gravitational force from the earth.

```
%---------------------------
%                      Question 4
% Solve the first order representation of the equation
%
%       r'' + (mu/(norm(r)^3))r = 0
%       where r and 0 are vectors
%
%using the matlab ordinary differential equation solver ode113 with
%initial conditions:
%       r(0) = [6997.56; -34108.00; 20765.49]
%       v(0) = [0.15599; 0.25517; 1.80763]
%
% plot your solutions for r(t) = (x(t), y(t), z(t)) on a single graph using
% the matlab command plot3. On the same plot, use matlab command quiver3 to
% show v(t) = (vx(t), vy(t), vz(t)) at each point on the plot.
%----------------------------
%the value of mu (gravitational parameter of the earth)
mu = 398600;
t = 1;
%start and end times
t_initial = 0;
t_final = 38032;
%bounds of integration
```

```
tspan = [t_initial, t_final];

%set the relative error tolerance
options = odeset('reltol', 1e-6);

%initial conditions
r0 = [6997.56; -34108.00; 20765.49];
v0 = [0.15599; 0.25517; 1.80763];
p0 = [r0;v0];

%Numerical Solution
[tout,pout] = ode113(@twoBody, tspan, p0, options, mu);

%Plot of r(t) = (x(t), y(t), z(t)) with initial conditions
figure;
quiver3(pout(:,1), pout(:,2), pout(:,3), pout(:,4), pout(:,5), pout(:,6), 1.25);
title('Position');
xlabel('x');
ylabel('y');
zlabel('z');
```

The code for the function twoBody is shown below.

```
function pdot = twoBody(t,p, mu)

%This code computes the right hand side of the system of differential equations
%           r'(t) = v(t)
%           v'(t) = mu/(norm(r)^3) * r(t)
%   inputs
%       t = time
%       p = [r;v]
%       mu =
%   where
%       r = position
%       v = velocity
%

%rx ry rz
r = p(1:3);

%vx vy vz
v = p(4:6);
```

```
%state equation
rdot = v;
vdot = -mu/(norm(r)^3) * r;

%the state equations as a column vector
pdot = [rdot;vdot];
end
```

Using *Orbital Mechanics: An Introduction* by Anil Rao page 7 as a reference.