Adrian Loekman
UID: 105785820
CEE/MAE M20
October 22, 2021

HOMEWORK 4

1. Resonance Problem: Mass-Spring System

1.1. Introduction

The goal in this problem is to solve a mass-spring system. Given a second order differential equation, we can solve the said differential equation using semi-implicit Euler's method and the help of MATLAB. MATLAB can help by using its for loop and array to store data for each time step. Using all the given values and equations for the force applied, F(t), we can plot the graph of position (x) and time (t) for different values of angular velocity (omega).

1.2. Model and Methods

For this problem, we will start by discretizing the second order differential equation. Since it is a second order, we will have 2 equations. The first will be the derivative of the main variable, in this problem will be x (position), and the second derivative of x, or the derivative of v (velocity). So, we have an equation $m\frac{d^2x}{dt^2} + kx = F(t)$. First, we find the first derivative, so we get $\frac{dx}{dt} = v$. Thus, for the second equation we get $\frac{d^2x}{dt^2} = \frac{dv}{dt}$. By rearranging the main equation, we obtain $\frac{d^2x}{dt^2} = \frac{dv}{dt} = \frac{F(t)-kx}{m}$. Note that x and v are both a function of time. Then, we can calculate each value of v and x for each time step using semi-implicit Euler's method. In the problem, it is stated to use k and k+1 but since k is already a variable put for the spring constant, I will be using i instead. So, the first equation would look like $\frac{x(i+1)-x(i)}{h} = v(i+1)$ and $\frac{v(i+1)-v(i)}{h} = \frac{F(t)-kx(i)}{m}$. After rearranging the equations, we could get an equation that MATLAB will be able to read and calculate. The equations are $x(i+1) = x(i) + v(i+1) * h$ and $v(i+1) = v(i) + \frac{F(t)-kx(i)}{m} * h$. This are the discretized equations governing equations using semi-implicit Euler's method.

The next step is to write the code. As usual, we use `clc; clear all; close all;` to clear any previous codes or plots. We define the parameters mass (m), spring constant (k), force applied (fc), 2 values for omega (w), step size (h), and time (t). We will create an array for time t with step size h, which will be t from 0 to 80 seconds, and initializing arrays for F(t) = fc*cos(wt), x, and v with zeros that goes to the n number of elements from t = 0 to 80s. Next we put our initial conditions where v(1) and x(1) are both 0, "cart is static at x=0 initially".

Next is the for loop. Since we have two values for omega, we will create a for loop that will first use the first value of omega and the next loop will use the second.

```
for j = 1:2
    omega = w(j);
```

This for loop will have an end at the very end of the for loop. Inside this for loop, we will calculate the value of F(t), x, and v. The for loop will be initiated from t = 1 to t = n-1 since this will calculate the three values for each iteration.

```
ft(i,j) = fc*cos(omega*t(i));
v(i+1,j) = v(i,j) + ((ft(i,j) - k*x(i,j))/m) * h;
x(i+1,j) = x(i,j) + h*(v(i+1,j));
```

This code calculates ft for both omega 1 and omega 2, thus we have the j which will print the values in different columns according to the value of omega. Notice the t has only (i) because it doesn't have a second column. V(i+1,j) is calculating the next value of v. j = 1 is for the first omega, and j = 2 will be for the second omega. Lastly, we calculate x. This x will use the new value of v (i.e. the value of v(i+1)) since this uses the semi-implicit Euler's method. Then, we end the loop for the time iteration and the omega values.

So, now we have 2 values of x, one from omega = 3 and another from omega = sqrt(10). The plot for x (with omega = 3) with t and x (with omega = sqrt(10)) with t will be plotted in the same figure window, but in a different subplot.

```
subplot(2,1,1) % 1st plot in figure 1
plot(t,x(:,1),'b-')
axis([0 80 -150 150])
```

subplot is used to locate the first plot. t is the time and x(:,1) is all x values in the first column (which corresponds to the value of every x with omega = 3). I also put axis so that the first subplot has the same axis as the second subplot. This way will make it easier to make observations and comparations. Similarly, the second subplot will be

```
subplot(2,1,2) % 2nd plot in figure 1
plot(t,x(:,2),'r-')
```

Now the subplot has a different position. This will create a subplot that is below the first subplot. In addition to the main data on the plot, I also added a grid and box to help observe better. Nevertheless, it is essential to put xlabel, ylabel, title, and legend to each plot so observers can get a better visual on the data.

1.3. Calculations and Results

When the code is run, we will get 1 figure as an output with two subplots.

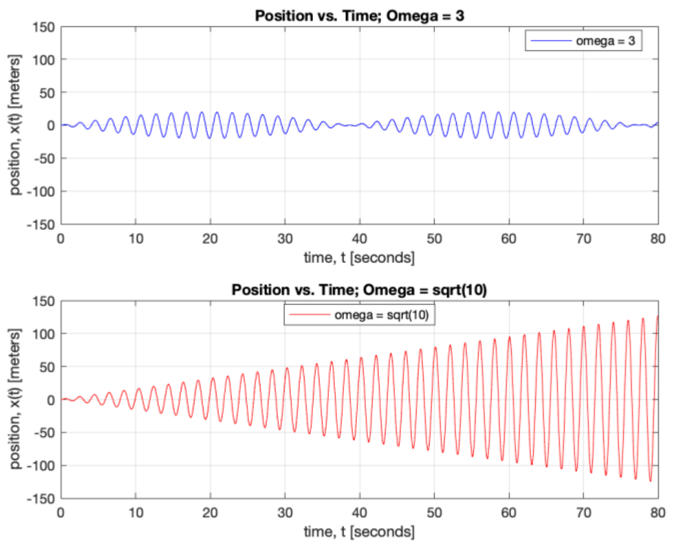

Figure 1 The graph of position vs. time with omega = 3 and sqrt(10)

We can see in figure 1 above that the two distinct looking plots by using different values for omega. Note that the value of 3 and sqrt(10) is very close. Thus, a small change can have a very large impact especially for this problem.

1.4. Discussion

From the graph presented above, we can analyze and get plenty observations. We can see that the variation of omega is very close. The first omega is 3 and the second is sqrt(10), which is approximately 3.1623. However, we can see a significant change in our plots. This is because omega is included in a cosine such that it will change the plot dramatically although only with a difference of a very small value. Both subplots have some kind of sinusoidal wave type.

In the first subplot, we can see that the position of the object moves through the equilibrium position and back to the equilibrium position, to the other side, and back to the equilibrium position. The position will continue to increase until it reaches a maximum amplitude before decreasing again and become close to the equilibrium position. Only then it will also create the same pattern with increasing amplitude and will decrease again. We are only observing this for the first 80 seconds, but I believe this pattern will go on as long as all conditions holds true. I am fairly sure these are called beats.

On the other hand, the position of the object with omega sqrt(10) has an amplitude that will continue to increase. This will keep increasing until it reaches a maximum amplitude of infinity. This maybe because the frequency of the function is equal to the natural frequency of the system which is sqrt(k/m). Since we know the value is sqrt(10), that value is equal to the natural frequency of the system. Thus, when the omega is set to sqrt(10), the maximum amplitude would go to infinity as it increases with each time step.

All in all, this problem lets us evaluate second order differential equations using semi-implicit Euler's method. I found it a great step to let me analyze and discretize the equations by hand before typing it in MATLAB. It also has a visualization of the data that makes observers get a better interpretation of the data.

2. Resonance Problem: Mass-Spring-Damper System

2.1. Introduction

This problem is similar to the previous problem and has the same goal as the previous problem. The difference in this problem is now the problem is more realistic where it involves a damper, thus a new variable is introduced, c. This is called the damping coefficient. Using the same known values from the previous problem, we can solve the differential equation using the semi-implicit Euler method and plot the graph of x and t.

2.2. Model and Methods

We would need to discretize the governing equations of the second order differential equation. This process is similar to what have been done in the first problem. However, we add a new variable c, damping coefficient. So now, our discretized governing equations that will be coded into MATLAB are $x(i+1) = x(i) + v(i+1) * h$ and $v(i+1) = v(i) + \frac{F(t) - cv(i) - kx(i)}{m} * h$.

Next, we can proceed with the code. The parameters are exactly identical as the parameters in the first problem. We now use the time up to 50s instead of 80s and we have 3 values of omega now: 3, 1, sqrt(10). Using the same time step 0.01, we can create an array for time and count the number of elements in that array, then set that to n. Now, we can create an array for ft, x, and v. Since we have 3 values for omega, then we're going to need 3 columns for 3 different ft, x, and v values.

```
ft = zeros(n,3); % Force Applied, 3 columns for 3 different omega
x = zeros(n,3); % Position, 3 columns from 3 different omega
v = zeros(n,3);
```

This way, we are creating a n row times 3 columns of zeros which we will calculate. We can set the initial conditions, but for this problem it would be unnecessary because the initial conditions are zero. The time marching loop is similar to what we've created for problem 1. The only difference is now we add a new variable c, or the damping coefficient. The for loop will proceed from time 1 to n-1 and omega from 1 to 3

```
ft(i,j) = fc*cos(omega*t(i)); % find ft for each time and omega
        v(i+1,j) = ((ft(i,j)-c*v(i,j) - k*x(i,j))/m) * h +
v(i,j); %velocity (explicit euler)
        x(i+1,j) = v(i+1,j)*h + x(i,j);
```

After this step, we should get x values that correspond to each time step. Since we have all the x values, we can plot the graph of x (position) vs time. In this problem, there are two figure windows. The first figure will plot the graph of x vs t with w = 3. The second figure plots 3 x vs t graphs on top of each other. In other words, we are plotting these plots on the same horizontal and vertical axis. This step is exactly the same as the first problem. Some aspects of the figure are xlabel, ylabel, title, and legend. In this particular problem, I also use grid on and box on to make a better visualization. Also, I changed the axis of the first figure so it's the same axis with the second figure. I also distinguished each plot with different colors so it may be easier to see which plot and its corresponding function.

2.3. Calculations and Results

When the MATLAB code is run, we get an output of two figures showing the position x vs t on the various values of omega (w).

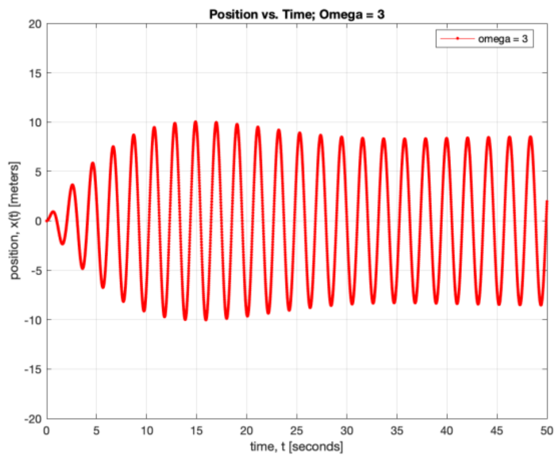

Figure 2 Mass-Spring-Damper position vs time with omega = 3

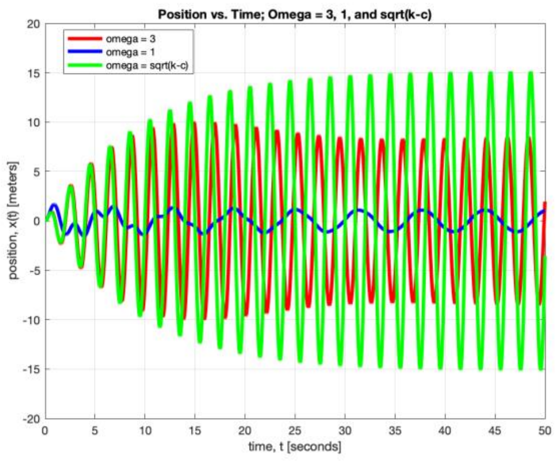

Figure 3 Position vs time with omega = 1,3, and sqrt(10)

From the two figures above, figure 2 and figure 3, we can see that the red plot is the plot of x vs t where omega is equal to 3, while the blue is when omega equals to two and green is when omega is equal to sqrt(k-c) = sqrt(10-0.2) = 3.3105. Though these graphs have different values, there are some similarities between them. The more details will be discussed in 2.4.

2.4. Discussion

When I plot these functions, I assume that they are at least similar because they have a pattern for each time step and position. These function looks like sinusoidal functions. This may be caused by the cosine that is in the applied force function. In the first figure, we can see that when t is less than 15s. the amplitude seems to get larger for every turnaround position. Then, the phenomena that I see is that it becomes a sinusoidal wave with a constant wavelength and amplitude. Thus, it also has a constant frequency and period.

A similar trend also applies to the second figure. We can see that when each function has a time range where the wave is not constant. This depends on the value of omega. As omega gets larger, the time for the function to become constant is larger. Another observation that I made is the amplitude. This is also proportional to the value of omega. As omega gets larger, the amplitude gets to a larger value. This graph doesn't show much about the period since it shows that the period for small value of omega is larger than the large value of omega but at a certain point it looks to have a very similar period.

This problem gives a more realistic touch to the mass-spring system. I can learn about how each component of the differential equation plays a part in determining how an object's position corresponds to each parameter for each time step. It's also interesting to learn about how varying a parameter, even a very small value, can impact the plot so much.

3. Resonance Problem: Implicit Euler

3.1. Introduction

The goal of this problem is very similar to the previous problem. In fact, the goal is exactly the same, but this problem wants to be solved using a different approach. This problem uses implicit Euler method and Newton method to solve the second order differential equation. The main goal of this problem is use implicit Euler method and Newton method to solve for v (velocity) and x (position) and plot the graph.

3.2. Model and Method

In this problem, we will start by defining parameters and arrays. Since these values are identical to problem 2, we will just code the same thing. We also do not necessarily need to state the initial conditions for x and v because they're both zero and we're using zeros as arrays. Since we're using implicit Euler now, we need to pay attention to a different detail than the previous approach. We're using two for loops so that we can take into account for each value of omega. Then for each time step we will calculate the value of ft. This step is identical to previous problems. Next, we need to define a guess for our values of x and v for the next time step. We can use 0 for both because of initial conditions.

```
q = [x(i,j), v(i,j)]; %initial guess at t(i+1)
```

These initial guesses will be updated with each time step. Then, we need to define a 2x1 vector which is the function when the time step +1. The first row is for x and the second for v. This is gotten from the discretized governing equations in problem 2.

```
funct = [m * ((q(2) - v(i,j))/h) + c*q(2) + k*q(1) - ft(i,j);
                   q(2) - ((q(1)-x(i,j))/h)]; % discretized function
from f1 and f2
        J = [k, m/h + c; -1/h, 1]; % jacobian 2x2 matrix
```

The values for q(1) and q(2) are x and v in the k+1 time step. The Jacobian matrix J, is the partial differential of the function f1 and f2 with respect to x and v. After this, we can proceed with the while loop. Set an error so that the loop goes on as long as the error is less than 1e-6. The error is actually the norm of the vector fk+1, which is the square of the first term added with the square of the second term. Next, calculate dq using the Jacobian and f(k+1).

```
        dq = J\funct;
```

This division is called left division and is very useful to calculate matrix multiplication instead of using the inverse and then multiply. The new value of q is q – dq. Calculate the function f(k+1) and calculate the error. The while loop will keep calculating until the error tolerance is met. Then, the value of x is gotten from q(1) and v from q(2). Do this for all time steps and then for each value of omega.

Lastly, we will plot the same graph like in problem 2. There are 2 figure windows where the first one shows the plot of x vs t with omega = 3 and the second shows x vs t with 3 values of omega.

3.3. Calculations and Results

When the code is run, the outputs would be 2 figure windows, the first one has 1 plot and the second has 3 plots.

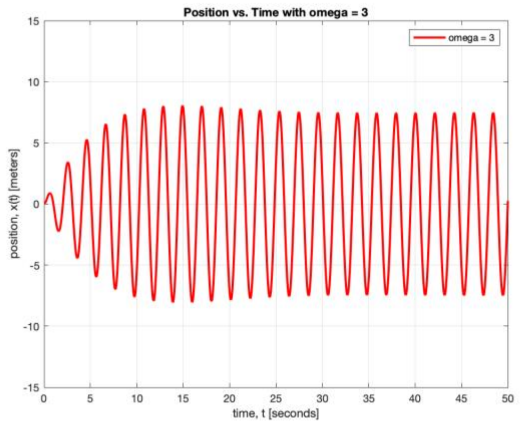

Figure 4 Implicit Euler for mass-spring-damper with omega =3

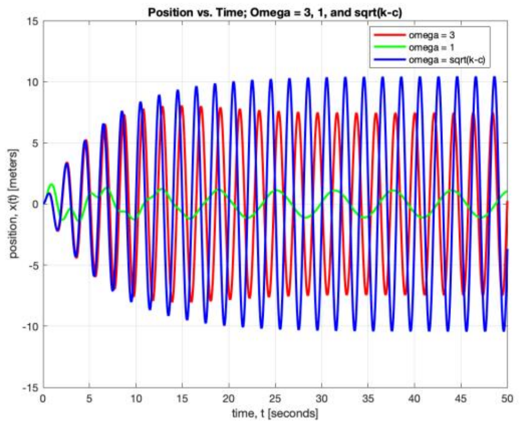

Figure 5 Position vs time with omega = 1,3, and sqrt(10) using implicit Euler method

If we look closely, the graph for both are very similar to what we found in problem 2. In fact, it's actually identical. However, if we closely look at the values of x for each t, we can see that the values are actually different.

3.4. Discussion

Looking at the output of problem 2 and 3 at a glance, we can see that the pattern is clearly identical. But, when looking more closely, we can see that the maximum amplitude of the graph is actually different especially for omega values 3 and sqrt(k-c). Using the implicit Euler, we actually find out that the amplitude is smaller than semi-implicit. Other than that, we find the exact same graph.

This problem is a challenge because the approach is very different than the first two. However, it is not quite difficult one we can understand the pseudocode given. This problem tells use the position of the object (i.e. how far the object goes from the equilibrium point). Then, we can analyze how different angular velocities can affect those positions.