# Spring Motion Modeling

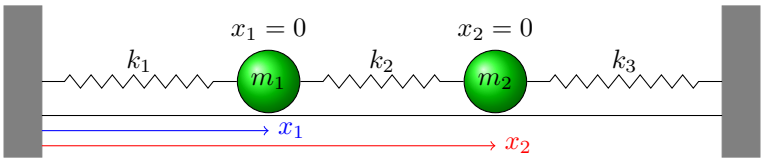
Caelan Osman

April 16, 2020

## 1 Introduction

### 1.1 Problem Definition

We will attempt to model the following system of springs. Using the equation $m\frac{d^2x}{dt^2} + b\frac{dx}{dt} + kx = f(t)$ Where $m$ is the mass of the object at the end of the spring, $b$ is the damping coefficient, $k$ is the spring stiffness, and $f(t)$ is the forcing function. We will assume all springs follow Hooke's law, We then can set up a system of second order differential equations to model the motion of the two masses $m_1$ and $m_2$.




### 1.2 Governing Equations

Given the situation at hand, assuming the walls are stationary, it would seem more natural to not have a forcing function; thus in this situation we will not consider forcing functions. Next we initialize our coordinate system to be positive to the right, we will then have $x_1$ and $x_2$ represent the positions of $m_1$ and $m_2$ respectively. The equilibrium points (where the masses naturally go after the oscillations stop) will be represented by $x_1 = 0$ and $x_2 = 0$ so any initial conditions on position will be measured from these points. To develop the equations we need to solve we come from a physics approach and use Newton's second law of motion. Notice that if the first mass moves to the left (in a negative position) the force exerted by $k_1$ on it is actually positive. so the force exerted by this spring on $m_1$ is $-k_1x_1$, if we consider the displacement of spring $k_2$ to be $(x_2 - x_1)$, then the force exerted on $m_1$ by $k_2$ is directed in the direction of the sign of the difference making $k_2(x_2 - x_1)$ the force contribution from this spring. Factoring in the damping coefficient we notice that friction always opposes motion; so if the first mass is moving to the right (a positive velocity) the damping force (friction) is actually to the left (a negative force) so the force from this piece is $-b_1x_1'$ where $x_1'$ indicates the first derivative with respect to time of the position and $b_1$ is the damping coefficient. So we have our first equation:

$$m_1x_1'' = -k_1x_1 + k_2(x_2 - x_1) - b_1x_1' = x_1(-k_1 - k_2) + k_2x_2 - b_1x_1'$$

using a similar analysis we find the equation for $m_2$ to be

$$m_2x_2'' = -k_2(x_2 - x_1) - k_3x_2 - b_2x_2' = x_2(-k_2 - k_3) + k_2x_1 - b_2x_2'$$

putting these equations into standard form we find that

$$x_1'' = -\left(\frac{k_1 + k_2}{m_1}\right) x_1 + \left(\frac{k_2}{m_1}\right) x_2 - \left(\frac{b_1}{m_1}\right) x_1' \tag{1}$$

$$x_2'' = -\left(\frac{k_2 + k_3}{m_2}\right) x_2 + \left(\frac{k_2}{m_2}\right) x_1 - \left(\frac{b_2}{m_2}\right) x_2' \tag{2}$$

# 2 Solution

## 2.1 Approach & Implementation

We will be using a Matlab designed GUI in order for solutions to equations defined by different initial conditions to be explored more readily. We will be making use of the built in ODE45 function that Matlab offers, in order to do this we need to define a new system of first order differential equations. Each second order can be decomposed into two first order equations so we will end up with a system of four first order differential equations. Defining our first order system, we will let $x_1$ and $x_2$ be what they were defined before as (the positions of $m_1$ and $m_2$ respectively). Then fix $x_3 = x_1'$ and $x_4 = x_2'$ which gives us:

$$x_1' = x_3 \tag{3}$$

$$x_2' = x_4 \tag{4}$$

$$x_3' = -\left(\frac{k_1 + k_2}{m_1}\right) x_1 + \left(\frac{k_2}{m_1}\right) x_2 - \left(\frac{b_1}{m_1}\right) x_3 \tag{5}$$

$$x_4' = -\left(\frac{k_2 + k_3}{m_2}\right) x_2 + \left(\frac{k_2}{m_2}\right) x_1 - \left(\frac{b_2}{m_2}\right) x_4 \tag{6}$$

We can now use these equations in Matlab to numerically solve for a solution. Keep in mind, we will have four initial conditions that need to be defined; along with the three spring constants, the two masses, the two damping coefficients as well as the time span desired for the solution to be determined over in the terms of an initial and final time (note that the initial time will almost always be 0). This makes a total of 13 constants that need to be defined by the user.

## 2.2 The GUI

We came up with the following design for the GUI and the function code is in the figure that follows.

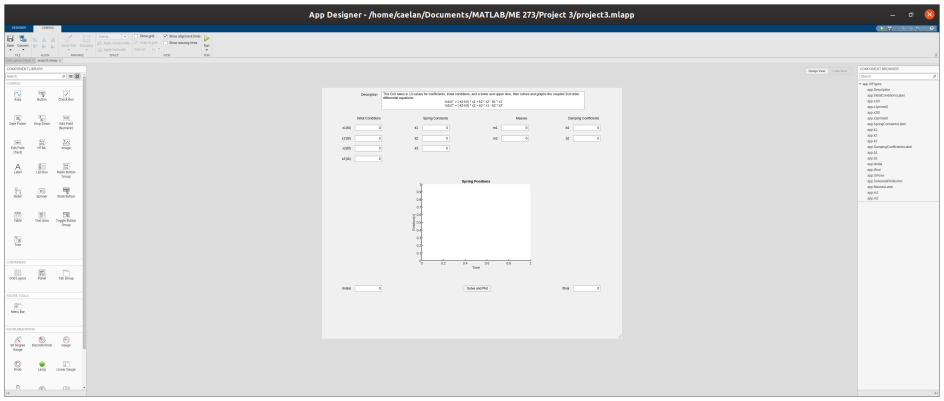



Figure 1: The GUI Design

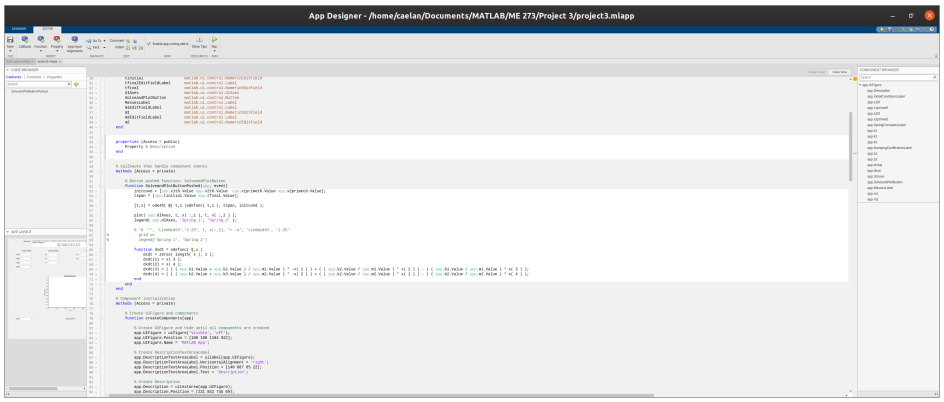



Figure 2: The Code

## 2.3 Results & Conclusion

We ran and solved the following ODE. This behaved as expected. As we can see all the initial conditions match, both springs start at position 1, spring 1 has a positive initial slope and spring 2 has a negative initial slope. Furthermore, with zero damping, we expect the behavior to last indefinitely.

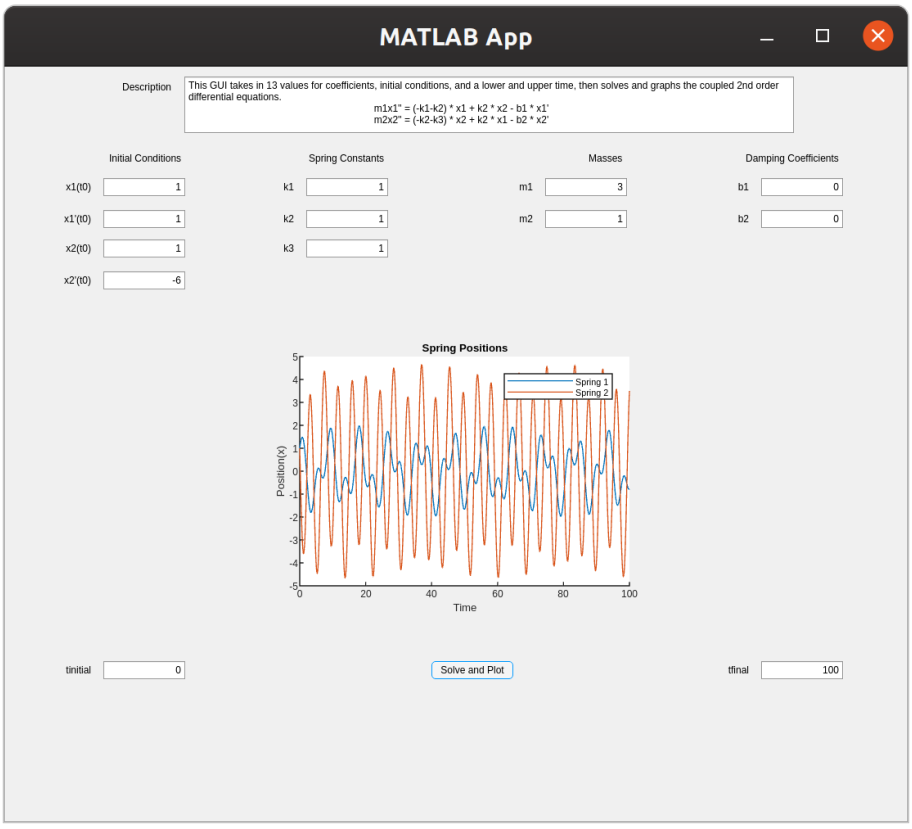



Figure 3: Example ODE

In the following example we keep everything the same but we add a little bit of damping.

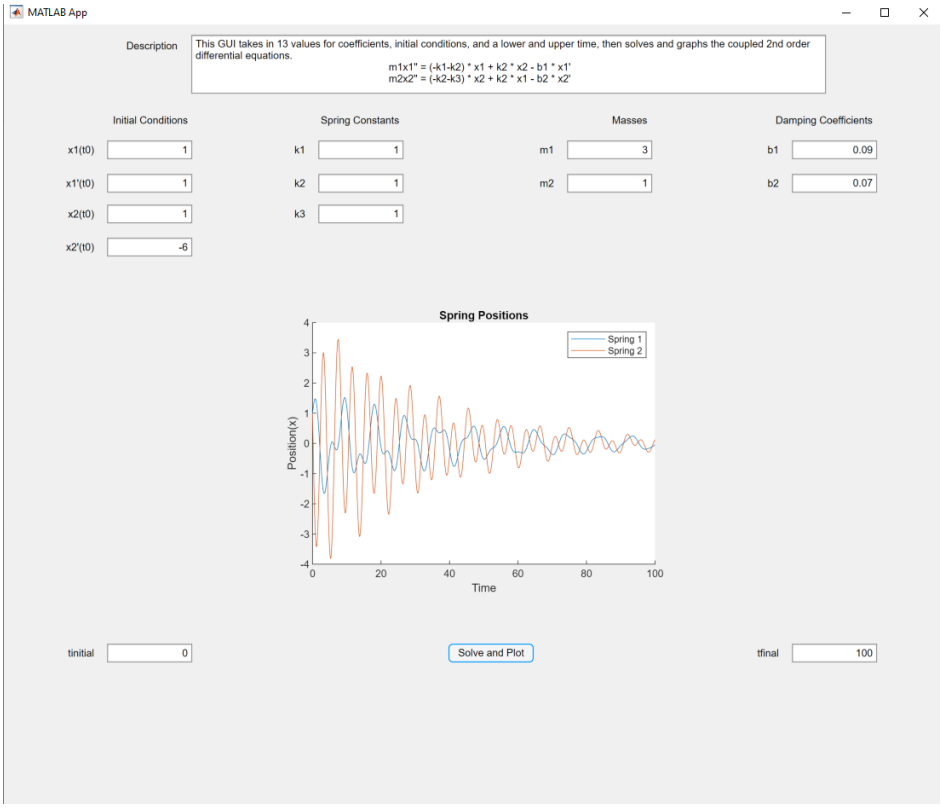



Figure 4: With Damping

As expected, the initial plot is close to the same but as time increases, the springs come closer and closer to equilibrium. I thoroughly enjoyed this project and although the code was not the most complicated I think this may be something to include in my portfolio and resume. The spring problem itself was pretty complex and shows that I can figure out more difficult problems and even more that instead of solving the problems by hand I will use a more efficient method (using a computer) which in the end will save the company I end up working for money. As my career in school continues I can continue to add more and more complex projects to my portfolio which will help show my progression which can foreshadow the progression I may have at a company.