Project 1:

Control of a Nonlinear Model of a Robotic Arm with a Flexible Joint via the Linearization Technique

Written and Conducted By:

Keeyan Haghshenas

October 20th, 2019

Electrical and Computer Engineering Department
School of Engineering
Rutgers University, Piscataway, NJ 08854

1. Introduction:

Project Problem Formation:

This project is based around the linearized dynamics of a single-linked robotic manipulator with a flexible joint. The robotic manipulator nonlinear mathematical model is given as follows,

$$I\ddot{\Theta}_1 + mgl\sin\Theta_1 + k(\Theta_1 - \Theta_2) = 0$$

$$I\ddot{\Theta}_2 - k(\Theta_1 - \Theta_2) = u$$

where Θ_1 and Θ_2 are angular positions, I and J are moments of inertia, m and l are, respectively, the link's mass and length, and k is the link's spring constant. The angular positions are defined by their respective locations in reference to their surrounding, where Θ_1 is the angle between the joint to base section and the vertical axis and Θ_2 is the angle between the joint to hand section of the arm. The nonlinear mathematical model describes the motion and position of the robotic arm and the purpose of the project is to, linearize the mathematical model through the state space approach by defining and designing the state space variables and equations that can govern the control system where it is desired to control the robot arm to hold any reference (nominal-, operating-, trim-, steady-state- set-point) angle in the entire range $[0, \frac{\pi}{2}]$ as well as to design a full-state feedback control law for this system using said linearization about a set point..

2. Methodology Used:

To begin the process of achieving the purpose of the project we must first understand the theory and mathematics behind the linearization of the nonlinear system, then define and design our variables, matrices and equations.

Theoretical Linearization of the Nonlinear System:

Derivation of Steady State Equations: The mathematical model used to describe the system is presented below but also can be manipulated algebraically also be represented in the boxed equations.

System: System Rewritten:

$$I\ddot{\theta}_1 + mgl\sin(\theta_1) + k(\theta_1 - \theta_2) \longrightarrow \ddot{\theta}_1 = -\frac{mgl}{I}\sin(\theta_1) - \frac{k}{I}(\theta_1 - \theta_2) \quad (1)$$

$$J\ddot{\theta}_2 - k(\theta_1 - \theta_2) = u \qquad \ddot{\theta}_2 = \frac{k}{J}(\theta_1 - \theta_2) + \frac{u}{J} \quad (2)$$

Where equation (1) is the equation that models the robotic arm and equation (2) is the equation that models the joint.

Now it is time to define the state space variables to use in our linearization, in this case, the variables m, g, l, I, J, and k are constants though we will change Θ_1 and Θ_2 and their derivatives to state space variables "x".

$$\begin{bmatrix} x_1 = \theta_1 \\ x_2 = \dot{\theta}_1 \\ x_3 = \theta_2 \\ x_4 = \dot{\theta}_2 \end{bmatrix}$$

It is evident here that x1, x2, x3, and x4 are the state space variables and the thetas are the original angles used in our control system despite their apparent derivates which have been assigned to x2 and x4.

$$F(x,u) = \dot{x} = \begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \end{bmatrix} = \begin{bmatrix} \dot{x_1} = \dot{\theta}_1 \\ \dot{x_2} = \ddot{\theta}_1 = [-\frac{mgl}{I}\sin(\theta_1) - \frac{k}{I}] + (\theta_1 - \theta_2) \\ \dot{x_3} = \dot{\theta}_2 \\ \dot{x_4} = \ddot{\theta}_2 = \frac{k}{J}(\theta_1 - \theta_2) + \frac{u}{J} \end{bmatrix}$$

When we find our first derivative then we can define the system as shown above. Where $\ddot{\Theta}_1$ is directly from the robotic arm equation and $\ddot{\Theta}_2$ is directly from the joint equation. From here we can replace the angles present in the system to x's because we previously defined them as state space variables.

$$F = \begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \end{bmatrix} = \begin{bmatrix} \dot{x_1} = x_2 \\ \dot{x_2} = [-\frac{mgl}{I}\sin(x_1) - \frac{k}{I}] + (x_1 - x_3) \\ \dot{x_3} = x_4 \\ \dot{x_4} = \frac{k}{J}(x_1 - x_3) + \frac{u}{J} \end{bmatrix}$$

What is important to understand is that we did not change anything but the representation of the information in the system. The previous representation was the original representation of the system where inputs were angles and the representation given above is in a state space form.

With the system in the current representation it is in, we are now ready to define the state space equations and their matrices, the state space equations are given by,

$$\dot{x} = Ax + By$$

$$y = Cx + Du$$

Where the matrices A,B,C,D are defined below and are also evaluated at the steady state values of x and u which are defined as, x = x_{ss} and u = u_{ss}.

$$A = \frac{\partial F}{\partial x}\Big|_{x,u=x_{ss},u_{ss}} = \begin{bmatrix} \frac{\partial F_1}{\partial x_1} & \frac{\partial F_1}{\partial x_2} & \frac{\partial F_1}{\partial x_3} & \frac{\partial F_1}{\partial x_4} \\ \frac{\partial F_2}{\partial x_1} & \frac{\partial F_2}{\partial x_2} & \frac{\partial F_2}{\partial x_3} & \frac{\partial F_2}{\partial x_4} \\ \frac{\partial F_3}{\partial x_1} & \frac{\partial F_3}{\partial x_2} & \frac{\partial F_3}{\partial x_3} & \frac{\partial F_3}{\partial x_4} \\ \frac{\partial F_4}{\partial x_1} & \frac{\partial F_4}{\partial x_2} & \frac{\partial F_4}{\partial x_3} & \frac{\partial F_4}{\partial x_4} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -\frac{mgl}{I}\cos x_{1ss} - \frac{k}{I} & 0 & \frac{k}{I} & 0 \\ 0 & 0 & 0 & 1 \\ \frac{k}{J} & 0 & -\frac{k}{J} & 0 \end{bmatrix}$$

What is important to note above is that the resulting matrix is a result of taking the derivative of F with respect to x. While what is shown below is a result of taking the derivative of F with respect to u.

$$B = \frac{\partial F}{\partial u}\Big|_{x,u=x_{ss},u_{ss}} = \begin{bmatrix} \frac{\partial F_1}{\partial u} \\ \frac{\partial F_2}{\partial u} \\ \frac{\partial F_3}{\partial u} \\ \frac{\partial F_4}{\partial u} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \frac{1}{J} \end{bmatrix}$$

As given from the reference material in Example 1.3 the remain matrices are defined as follows,

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \text{ and } D = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

What is important to take note of here are the dimensions of the matrices because it aids in the understanding of what they are, notice how C carries the dimensions of the “x” matrix while D carries the dimensions of “u” matrix.

From here we have now defined our matrices for the state space equations, which will now allow us to define the state space equations themselves to which I will do below,

<u>State Space Equations:</u>

$$F = \dot{x} = Ax + By = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -\frac{mgl}{I}\cos x_{1ss} - \frac{k}{I} & 0 & \frac{k}{I} & 0 \\ 0 & 0 & 0 & 1 \\ \frac{k}{J} & 0 & -\frac{k}{J} & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ \frac{1}{J} \end{bmatrix} u$$

And

$$y = Cx = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

<u>Obtaining the Steady State Values:</u> We can obtain the steady state values by setting F = 0 at steady state and due to the fact that not only is x_1 the only angle we desire to have set but also the fact that $x_1 = \theta_1$. Therefore we can say that y_1 will be our main output and it will also equal our desired angle θ_1 at steady state.

$$F = \begin{bmatrix} x_{2ss} \\ -\frac{mgl}{I}\sin(x_1ss) - \frac{k}{I}(x_{1ss} - x_{3ss}) \\ x_{4ss} \\ \frac{k}{J}(x_{1ss} - x_{3ss}) + \frac{u_{ss}}{J} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

What is important to note here is that this matrix is the original F matrix before taking any derivates and that since we need to set the system to zero, we create a zero matrix and set it equal to the system. Furthermore the steady state output y_1 is the desired value for θ_1 implying that, $y_{1ss} = y_{desired} = x_{1ss} = \theta_{1(desired)}$. From here we can solve for the other steady state variables at steady state yielding, the following values.

$$x_{1ss} = \theta_{1(desired)}\, , x_{2ss} = 0$$

$$x_{3ss} = \frac{mgl}{k} sin x_{1ss} + x_{1ss}\, , x_{4ss} = 0$$

$$u_{ss} = mglsin(x_{1ss})$$

Now with the state space equations, variables, and matrices defined along with the steady state points also being defined, we can now introduce Simulink and Matlab to simulate our linearized system.

Simulation

The first step in simulating the dynamics of the linearized system using Simulink and Matlab would be defining the matrices A,B,C, and D as well as creating F with the using the command "F = place(A, B , lambda_desired). Then the next step would be selecting initial conditions x(t_0) for $x_1, x_2, x_3, and x_4$ that are relatively close to their steady state values as well initial condition s for $\Delta x_1, \Delta x_2, \Delta x_3, and \Delta x_4$ where $\Delta x(t_0) = x(t_0) - x_{ss}$.

Selecting Certain Eigenvalues: The next step in the process is then the choice of the eigenvalues. Such that $\lambda_{desired}$ has a negative real part and is also somewhat large as well as also considering that if we were to use a complex value eigenvalue the complex conjugate should also be included; these restrictions give a range in which the eigenvalues can be in, and that range is somewhere between -5 and -20. The selection of the eigenvalues have an affect on the settling time and overshoot of the resulting graphs, where settling time is defined as the time of transience from the beginning until we reach within 5% of our steady state values which were previously determined. Overshoot is then defined as the initially large spike the system response will have during the change from the initial values during transience, or more specifically the difference in the maximum value of the spike and the steady state value.

To then explicitly defined the effect λ has on both settling time and overshoot we can use the following equations,

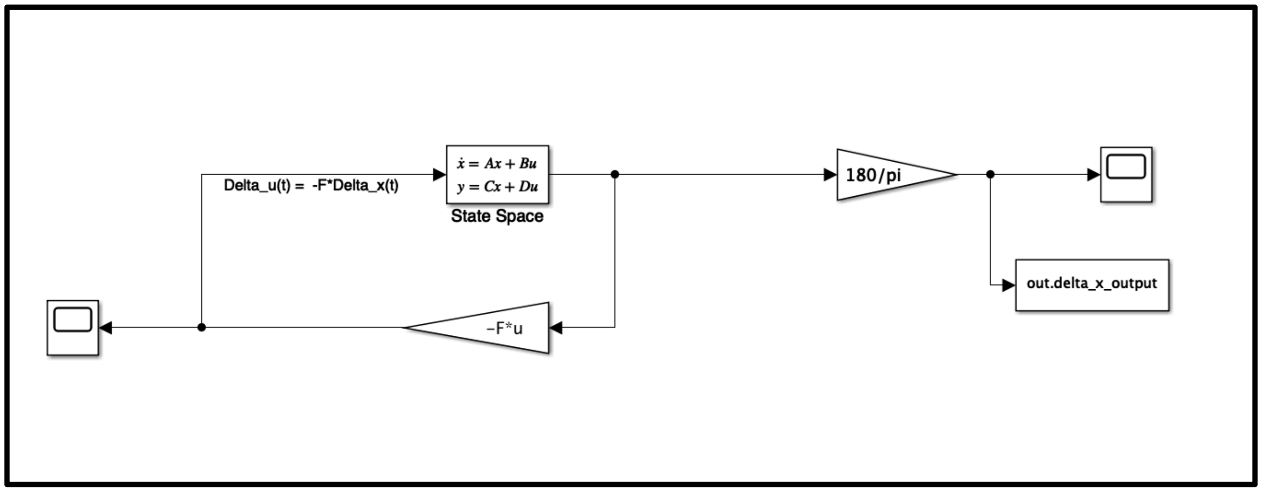
Settling Time:

$$\text{Real}(\lambda) \approx \frac{1}{\tau}\text{ ; where } \tau \text{ is the time constant related to settling time.}$$

Overshoot:

$$\text{OS} = e^{-\frac{\tau\pi}{\sqrt{1-\tau^2}}}$$

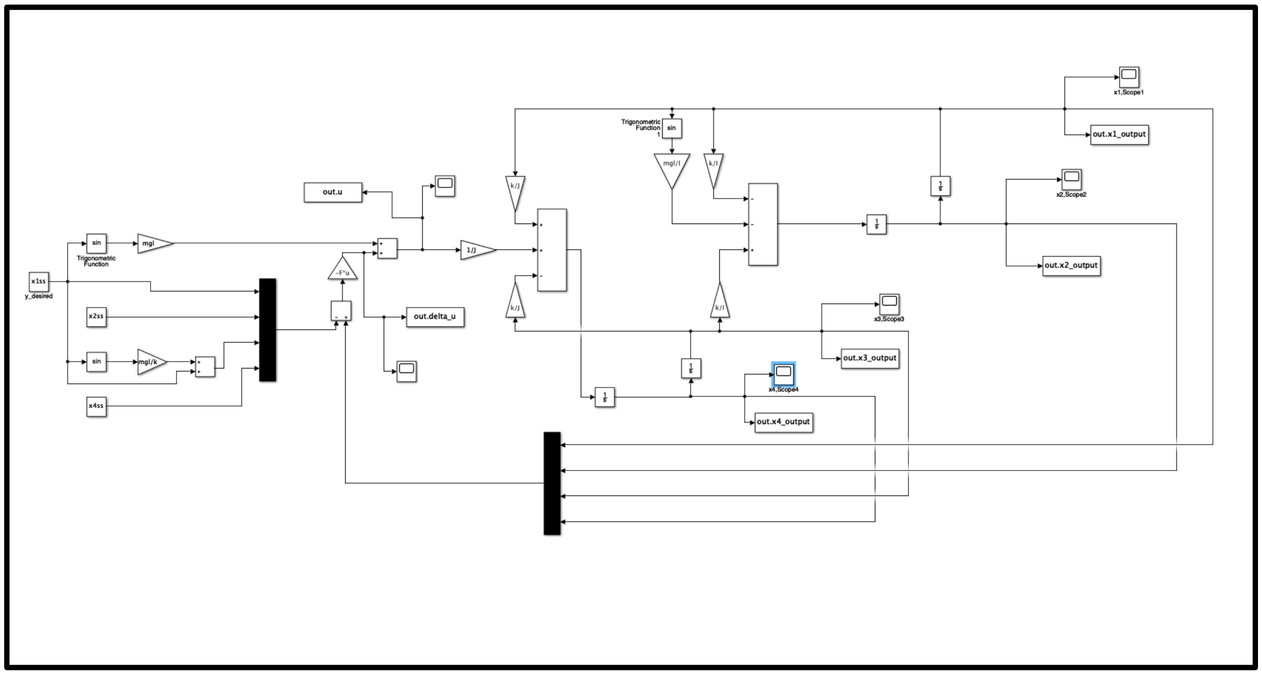
The relationship between the settling time and overshoot revolves around both τ and λ. To define it explicitly we can see that the larger the λ the resulting τ will be smaller, in turn this will create a smaller settling time but then create a larger overshoot (meaning more error) during the transient period. Inversely, we can see that the smaller the λ the resulting τ will be larger meaning a larger settling time and a smaller overshoot. Therefore in the project, we must test eigenvalues until there is an appropriate settling time and overshoot. We can use the block diagram below to test eigenvalues, and select the ones that give us small overshoot, rise, and settling times.



To describe what is shown above, it is the a model that shows the linearized system dynamics or in other words, it shows how the system will respond when we select a certain output meaning the depiction of the system moving from its initial conditions towards the certain output we have selected. The few blocks present in the diagram do very specific things pertinent to what needs to be done, where the state space block emulates the state space equations we derived earlier that are defined in a Matlab workspace code and gives a certain output depending on the input as well as having the initial conditions defined that were chosen to be close to the steady state values of x. While the -F is the feedback controller that takes the output produced by the state space block and creates a new input, to which is fed back into the state space equation, with every iteration of the loop that is occurring we get closer and closer to the desired steady state output values.

Once we have acquired values for A,B,C,D, $\lambda_{desired}$ and F we can look to creating a model for the robotic arm, while designing the model to keep θ_{1ss} at a desired steady state, in the projects case it is $\frac{\pi}{3}$. The general ideology is to have the steady state values $x_{ss} and u_{ss}$ as inputs into the steady state system block and then to have a feedback controller modify the output to correct the input until we reach our desired steady state values. The model for the robotic arm is shown below,

Schematic:



3: Results and Conclusion:

3a. Linearized Dynamics

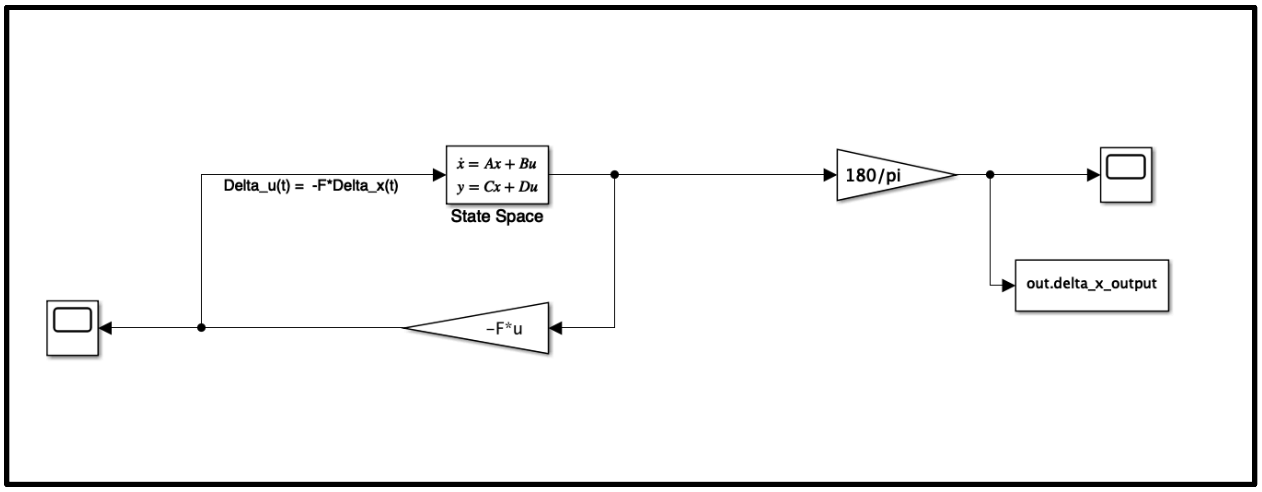
I will now present the results of the simulations I have described above, including the order in which I have conducted them, as well as initial conditions and constant values. There are a few constants that are set inside our dynamic system which are as follows, mgl = 5, I = 1, J = 1, and k = 0.08. From here we can also set our $x_{1ss} = \frac{\pi}{3} = 1.0472$, $x_{2ss} = 0$, and $x_{4ss} = 0$. Then we can obtain $x_{3ss} = \frac{mgl}{k} sin x_{1ss} + x_{1ss}$ = 55.174 radians and u_{ss} = 4.3301. This results in the matrices,

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -2.58 & 0 & 0.08 & 0 \\ 0 & 0 & 0 & 1 \\ 0.08 & 0 & -0.08 & 0 \end{bmatrix}, B = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}, C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, D = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

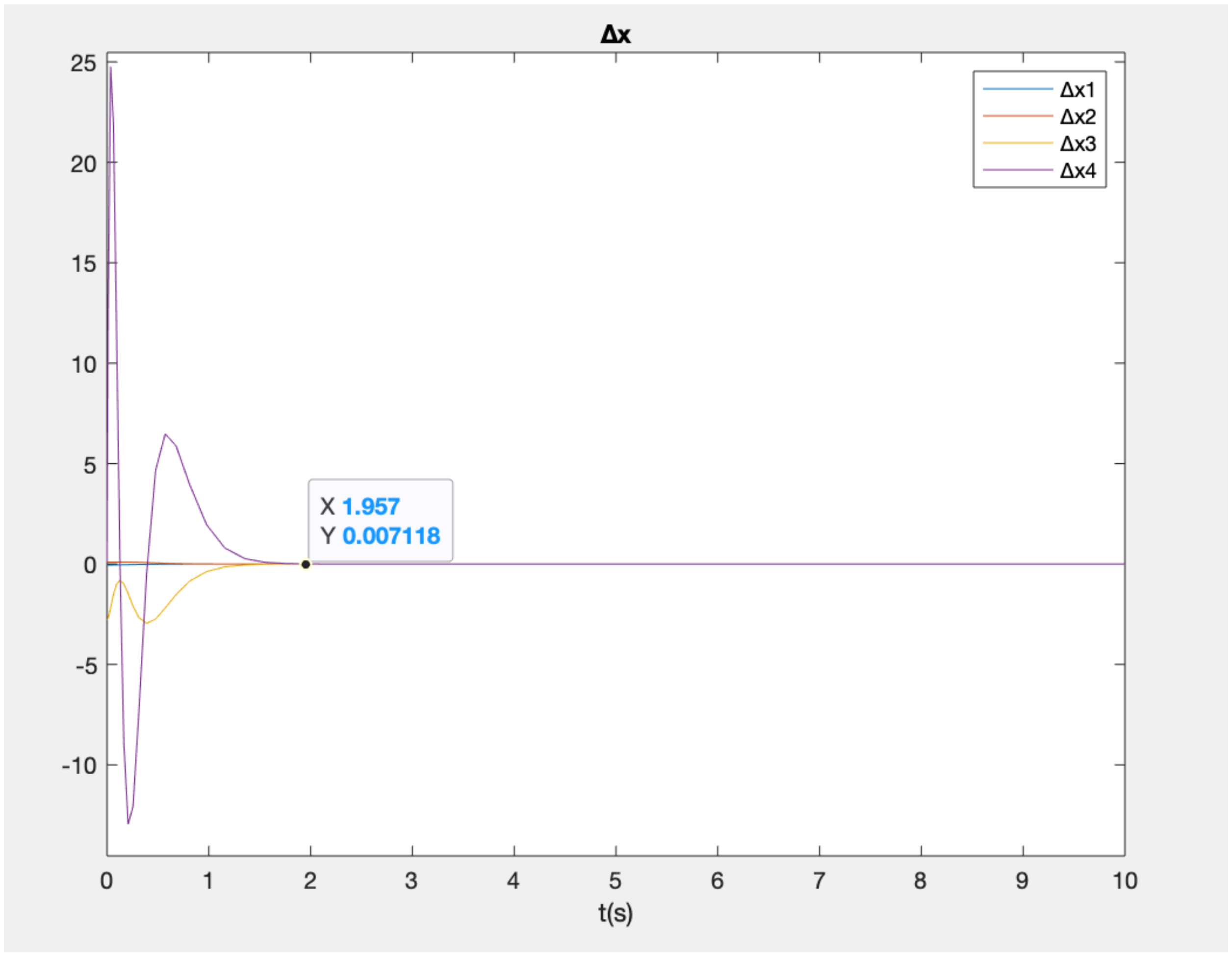
From here I was able to select my eigenvalues to be, λ = [-12-j0.2, -12+j0.2, -7-j0.2, -7+j0.2] due to the relatively low to moderate level of overshoot and the dynamics settling around 1.5 - 2 seconds.

Eigenvalue Testing/Selection Results:

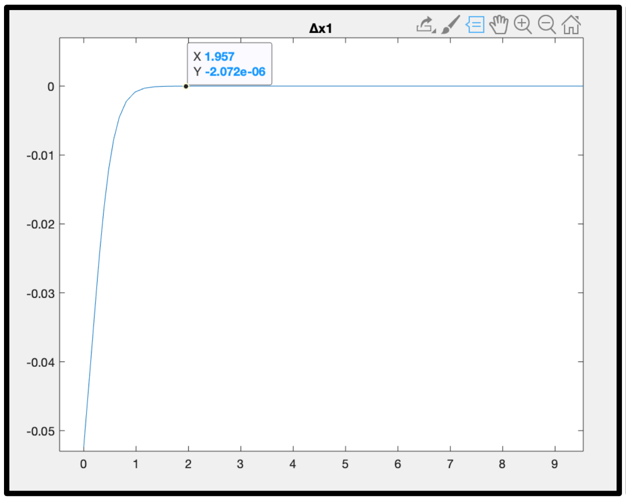
System Model:

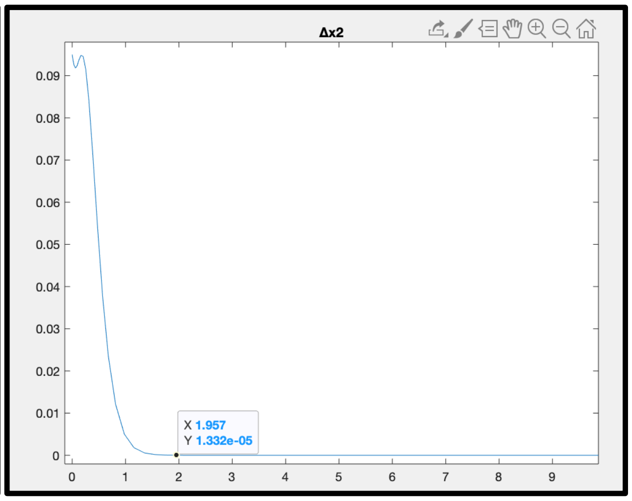


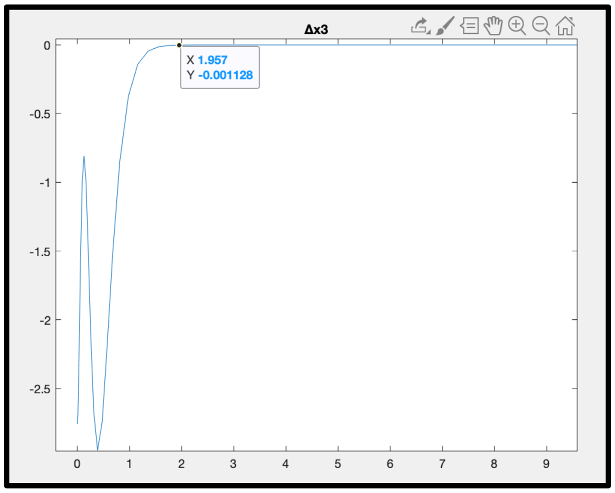
Overall Graph:

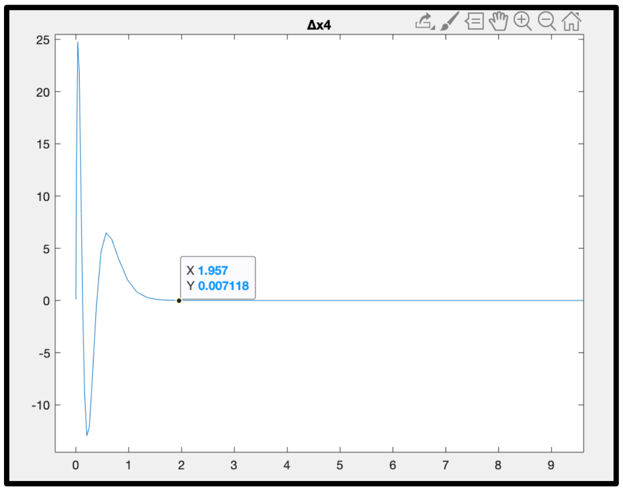


Individual Graphs:









Overshoot Values:

Δx1	0
Δx2	0.85
Δx3	2.95
Δx4	22.12

Comment on The Results:

Looking for a 5% settling time, I plotted the signals of interest and used the cursor tool (Tool>Data Tips) to drag a cursor around and eyeball the data until I felt it reached the appropriate point and stays within +/- 5% of the steady state value. Which resulted in the estimated time of transience being 1.957 seconds. I have listed the overshoot values in the table above and now will compare them to the steady state values.

Steady State Values: (all variables are steady state values)

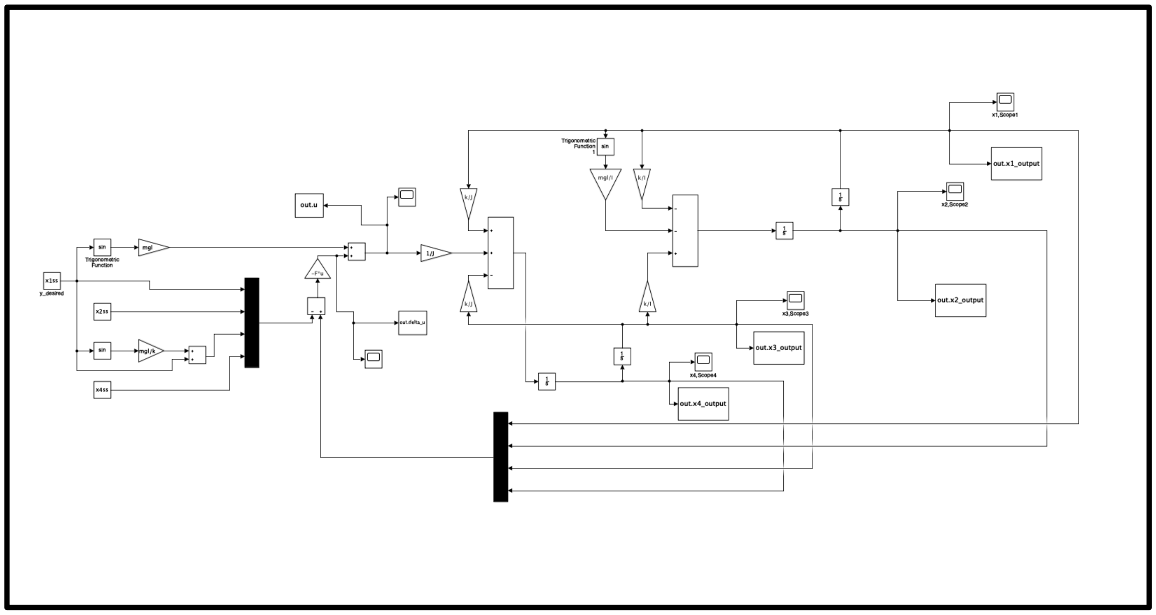
x1	1.0472 radians
x2	0 radians
x3	55.174 radians
x4	0 radians

It is clear to see that the x_{4ss} variable has the largest overshoot. x_{4ss} is the representation to the change in angular position otherwise known as velocity of θ_2. Therefore it is expected because of the movement of the arm from its initial position to the requested position. Though expected, problems with the arm could arise, like instability for example, however the trade off is that with a higher instability there is a lower transient period meaning we reach our steady state faster. The remainder of the variables seems to have reasonable overshoots to the point where I do not see any problems with my eigenvalue selection.

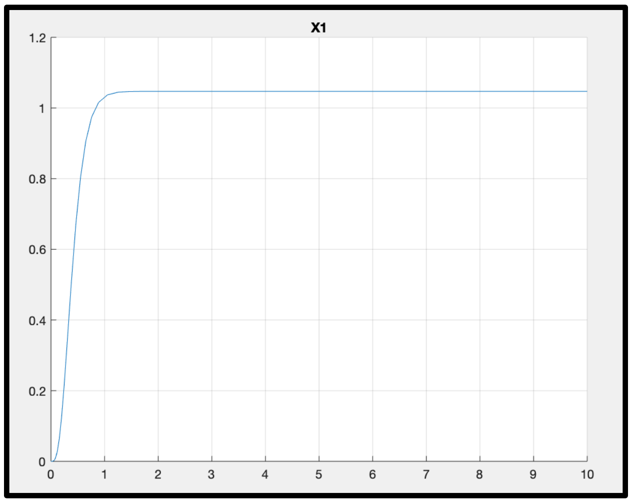
3b. Robotic Arm Simulation

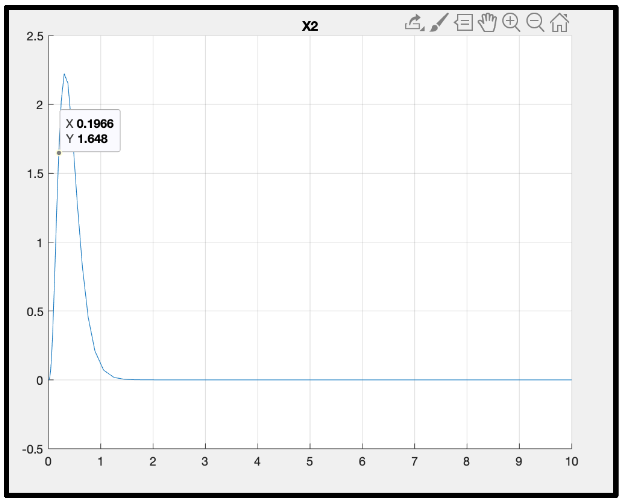
Using the afore mentioned values for A,B,C,D and steady state x and u, I designed and simulated the robotic arm in Simulink and present the results below.

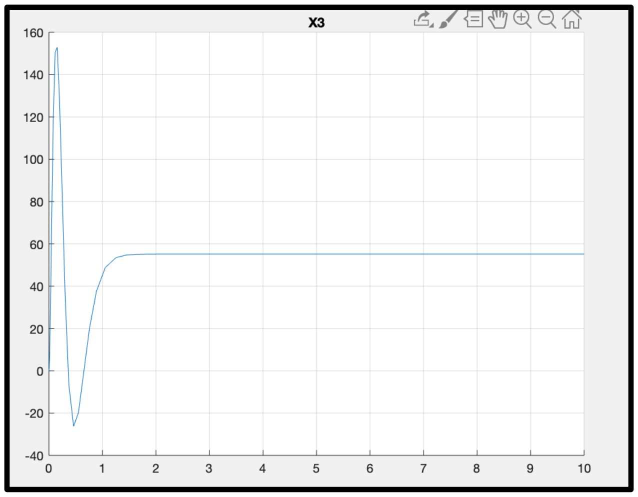
System Model:

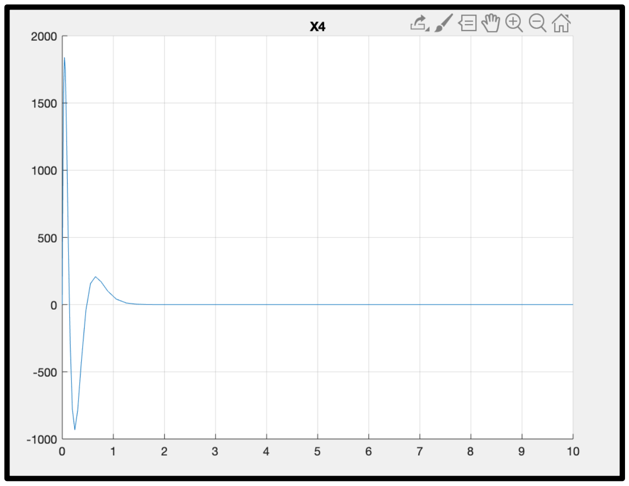


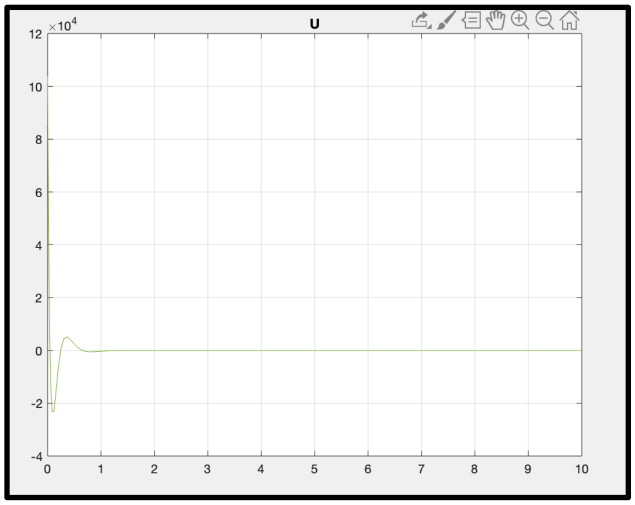
Individual Graphs:

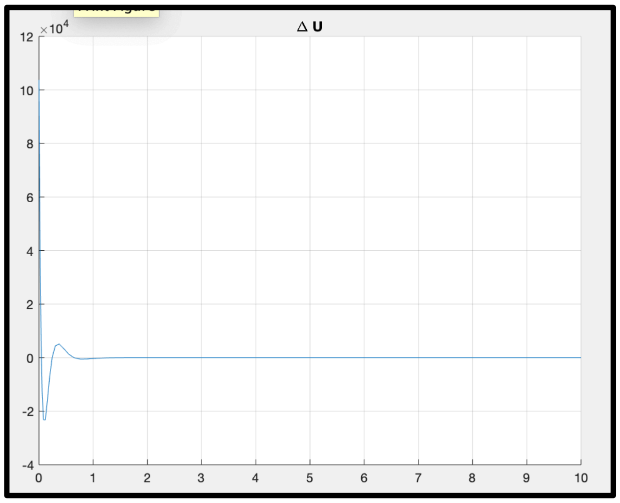












Comment on The Results:

As you can see there is a lot of overshoot but we knew that when we simulated because we chose lower transient time for larger overshoot. X4 and X3 have the largest overshoots which can be expected because their delta overshoots being relatively large as well. Furthermore this means that our θ_2 will have some unstable motion but we sacrificed that for lower transient time. θ_1 has the smoothest motion because it converges to the steady state relatively quickly. One thing to note is how the graphs do not start at their initial conditions meaning there must be some problem within the simulation, due to my unfamiliarity with Simulink and Matlab I was unable to debug the solution before the date of the project, meaning majority of the graphs represented above are skewed by their initial positions. The good thing however is that all of the graphs mimic the delta value shapes meaning the model is running correctly however the communication between Simulink and Matlab with the initial conditions is incorrect. I will hope to remedy this problem in future projects.

When looking our inputs u(t) to delta u(t) we can see the input to the function itself has a high overshoot and converges to the steady state value in the settling time we estimated. Between the two graphs we can see the difference as well because $u(t) = u_{ss} + \Delta u(t)$.

4.Conclusion:

In the end, the purpose of the project and the overarching goals behind the project were achieved. The robot arm simulation was a success when compared with a certain selection of eigenvalues and really solidified the relationship between the eigenvalues settling time and overshoot. Within a certain degree, the linearization of the robot arm model including, variables, feedback controller, eigenvalue testing, state space block implementation, and full robot model implemented in Simulink yielded results. It was the unfamilityie with the new software and the communication between the two softwares that skewed some of the restulsts towards the end. I hope to remedy this gap in knowledge for the remaining projects and look forward to further analysis of such interesting concepts.

5: References:

Gajic, Zoran, and M. Lelic. Modern Control Systems Engineering. 1st ed., Prentice Hall, 1996.

Control System Design class notes

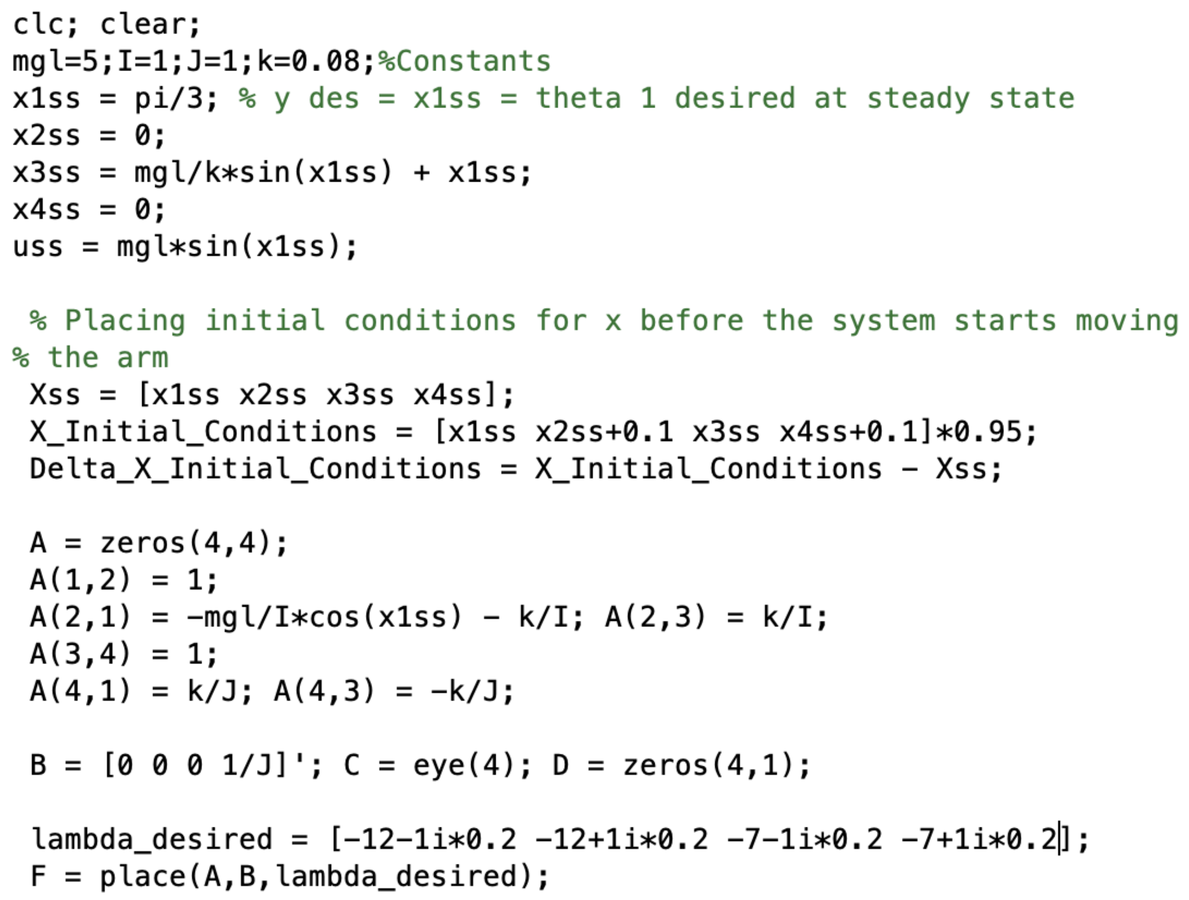
Matlab Code

Matlab Website

Simulink Manuel

5 Appendix

MATLAB Code



```
clc; clear;
mgl=5;I=1;J=1;k=0.08;%Constants
x1ss = pi/3; % y des = x1ss = theta 1 desired at steady state
x2ss = 0;
x3ss = mgl/k*sin(x1ss) + x1ss;
x4ss = 0;
uss = mgl*sin(x1ss);

 % Placing initial conditions for x before the system starts moving
% the arm
 Xss = [x1ss x2ss x3ss x4ss];
 X_Initial_Conditions = [x1ss x2ss+0.1 x3ss x4ss+0.1]*0.95;
 Delta_X_Initial_Conditions = X_Initial_Conditions - Xss;

 A = zeros(4,4);
 A(1,2) = 1;
 A(2,1) = -mgl/I*cos(x1ss) - k/I; A(2,3) = k/I;
 A(3,4) = 1;
 A(4,1) = k/J; A(4,3) = -k/J;

 B = [0 0 0 1/J]'; C = eye(4); D = zeros(4,1);

 lambda_desired = [-12-1i*0.2 -12+1i*0.2 -7-1i*0.2 -7+1i*0.2];
 F = place(A,B,lambda_desired);
```