**ESE 446 Final Project: Simulation and Control of the 3R Planar Robot**

**Done by:** Manohar Pradhan

: Billy Gilbert Habimana Cyusa

**Submitted to :** Prof. Dennis Mell

**Date:** 5/2/2019

**Overview**

**Goal:** The goal of this project to demonstrate the understanding of the Jacobian, robot dynamics, trajectory, and manipulator control of a 3R planar robot as shown below:

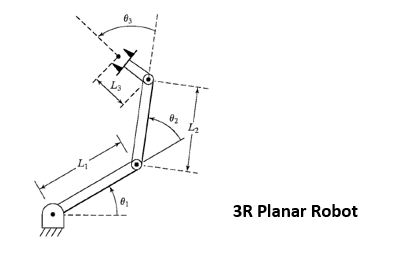


Fig 1. Showing the 3R Planar Robot with Links L1,L2 and L3.

**Approach:** The entire project consists of three parts, divided into 3 tasks which are summarized below:

1. **Task 1**: Robot Simulation - We first derived equations that describe the dynamics of the robotic manipulator and calculated the acceleration by solving the dynamic equations. Then, the specified initial link lengths, angles, masses were used to demonstrate the dynamics of the robotics through Simulink.
2. **Task 2**: Robot Simulation with Control Partitioning: A closed loop system was designed by creating a control partition and the robot dynamics was investigated.
3. **Task 3**: Robot Simulation with Path Trajectory: Spline fit curves were created for each joint, meaning 3 curves were created and the trajectories were fed to the 3R manipulator and simulated.

**Task 1**

The position of the robotic links were described by the following DH parameter table:

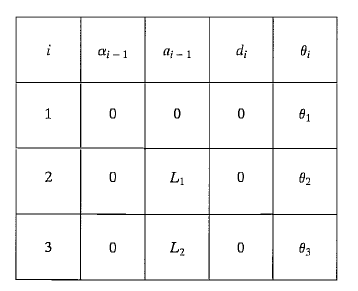
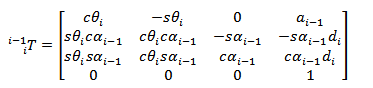


Table 1. Showing the DH parameters that describe the position of the manipulator

First, we found the transformation matrix that represented the position of the manipulator with respect to the previous frame. The transformation matrix going from frame(i-1) to (i) is given below:



In order to find the final frame with respect to the world frame (initial), we multiply all the transformation matrices from getting from the first to the last frame, ie, in our 3R planar robot, the end effector frame(3) can be represented in the initial frame by the following equation:



**Mass Matrix**: We first found the Jacobian matrices for the velocity and angular velocity using the following equations:

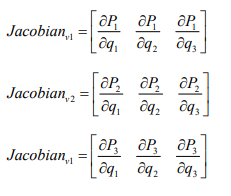


Fig 2. Showing Jacobian calculation for the velocity ‘v’.

Where [q1.q2,q3] are the positions of the links

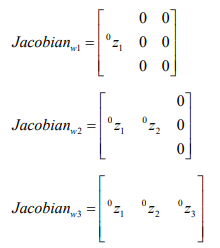


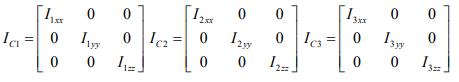
Fig 3. Showing Jacobian calculation for the angular velocity ‘w’.

Once the Jacobians were found, the following equation was used to find the mass matrix:



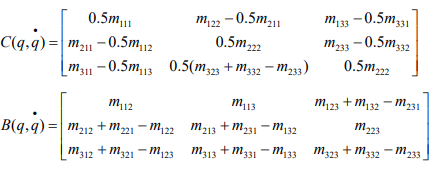
The ‘*I’* matrix is the inertial matrix that represents the mass distribution for each joint.

Thus,



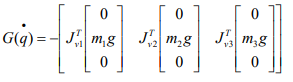
For example, I1xy represents the product of the moment of inertia about the x and y axis for link 1.

**Centrifugal and Coriolis Matrix:** The centrifugal(C) and coriolis matrix(B) represent the centrifugal and coriolis forces experienced by the manipulator links. The matrices are as follows:



Where mijk is derivative of mij with respect to qk.

**Gravity matrix:** The gravity matrix is given by:



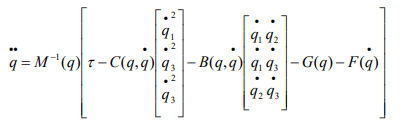
**Friction Matrix:** The friction is represented as follows:



**Tau**: Tau is the applied torque(force) on the system given by:



The acceleration of the links can be found by rearranging the equation for Tau, which is:



**Demonstrations with video clips**

The results from our simulation, presented below, was sped up for the purposes of a concise presentation.

1. **No actuator torque, no gravity**

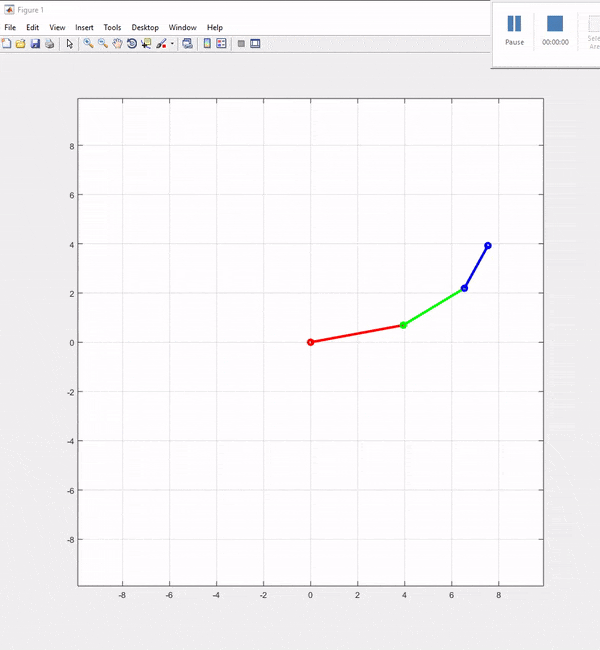
****

Figure 1.1.1: Simulation of Robot arm

****

Figure 1.1.2: Graphs of x, y and alpha vs time

**2. No actuator torque, with gravity**

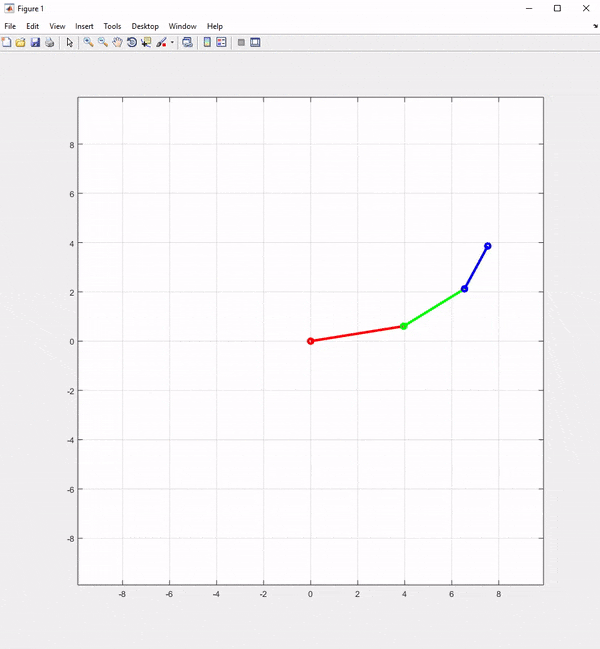
****

Figure 1.2.1: Simulation of Robot arm

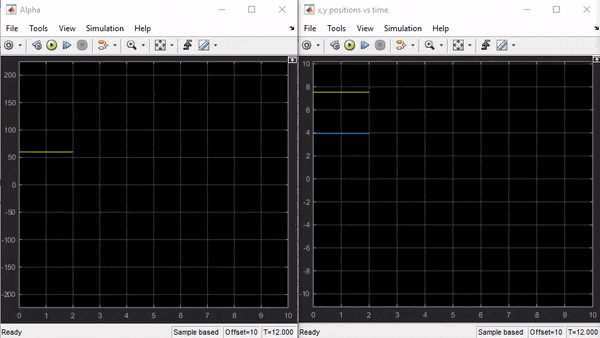
****

Figure 1.2.2: Graphs of x, y and alpha vs time

**3. Joint 1 actuator torque, with gravity**

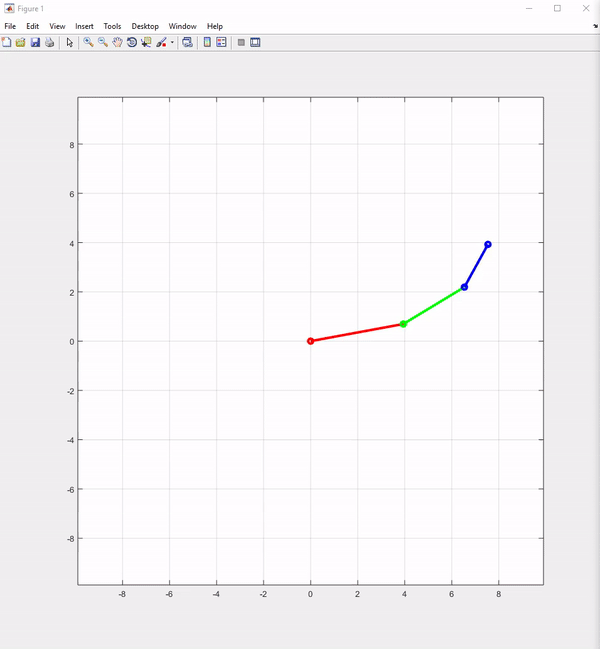
****

Figure 1.3.1: Simulation of Robot arm

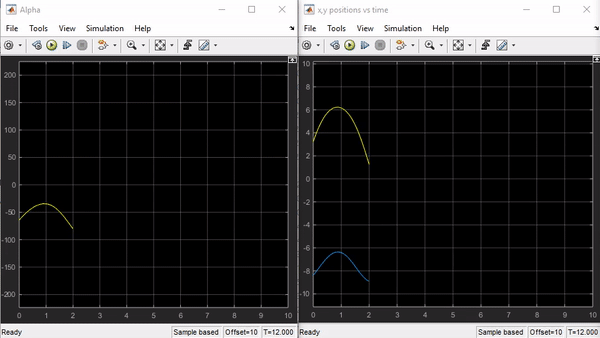
****

Figure 1.3.2: Graphs of x, y and alpha vs time

**4. Joint 1 Joint 2 actuator torque, with gravity**

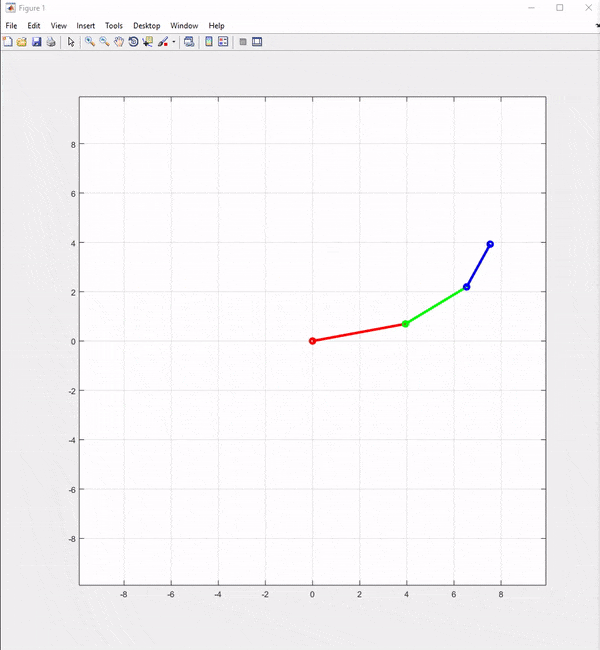
****

Figure 1.4.1: Simulation of Robot arm

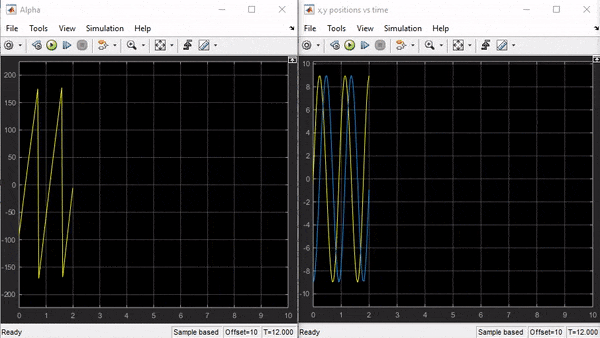


Figure 1.4.2: Graphs of x, y and alpha vs time

**5. Joint-1 Joint-2 Joint-3 actuator torque**

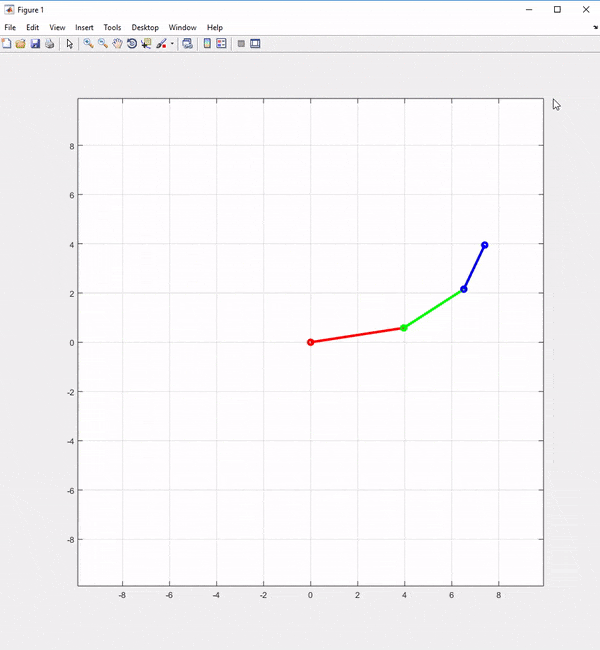
****

Figure 1.5.1: Simulation of Robot arm

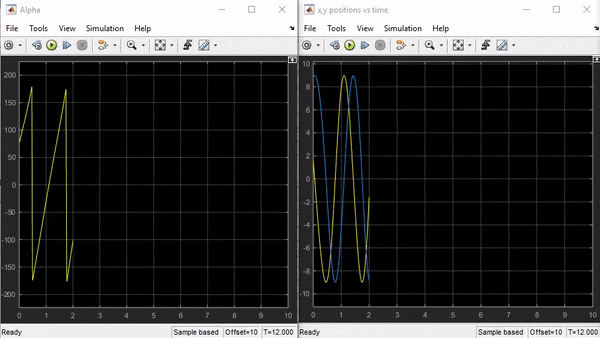


Figure 1.5.2: Graphs of x, y and alpha vs time

**Task 2**

**Robot Simulation with Control Partitioning**

A closed loop system was designed by creating a control partition and the robot dynamics was investigated. The following diagram shows the position and velocity controlled closed loop:

**Controller Design**: The commanded input, the position and velocity is fed into the gain blocks, Kp and Kv where they are multiplied respectively. Then, both these signals are fed into the summation of the mass, centrifugal, coriolis, friction and gravity matrix in order to calculate Tau and then again the tau is used as the input to find the velocity and acceleration. The velocity and acceleration is fed into the integrator to get actual position and velocity and then sent back to check with the commanded input where the error is again fed to the Kp and Kv blocks. The closed loop control tries to reduce the error to zero. Kp is a measure of how fast the error settles whereas Kv checks the damping behaviour as the error is reduced.

Critical Damping: The condition for critical damping is as follows: Kv = 2p. We tried different sets of Kp and Kv in order to run our simulations and investigate the rise time and the overshoot. The following content summarizes our results:

**Demonstration of simulation with varying values of Kp and Kv**

Below are simulations when we applied step inputs to the command angles, one joint at a time.

Table 1: Variation of Kv and Rise Timeto changes in Kp

|  |  |  |
| --- | --- | --- |
| Kp | Kv | Rise Time(s) |
| 1 | 2 | 1.2 |
| 10 | 6.3246 | 0.6 |
| 100 | 20 | 0.3 |
| 1000 | 63.2456 | 0.15 |

1. **Kp = 1; Kv = 2**



Figure 2.1.1: Simulation of Robot arm Figure 2.1.2: Graph of error signal vs time

1. **Kp = 10; Kv = 6.325**



Figure 2.1.3: Simulation of Robot arm Figure 2.1.4: Graph of error signal vs time

1. **Kp = 100; Kv = 20**

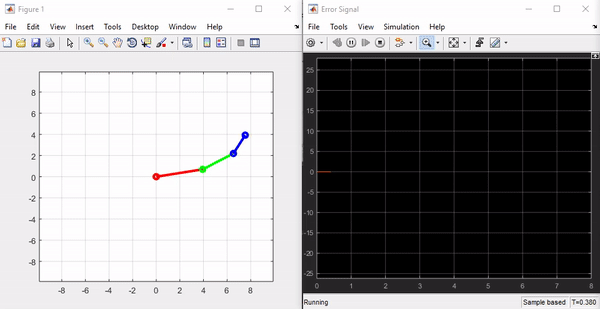


Figure 2.1.5: Simulation of Robot arm Figure 2.1.6: Graph of error signal vs time

1. **Kp = 1000; Kv = 63.245**

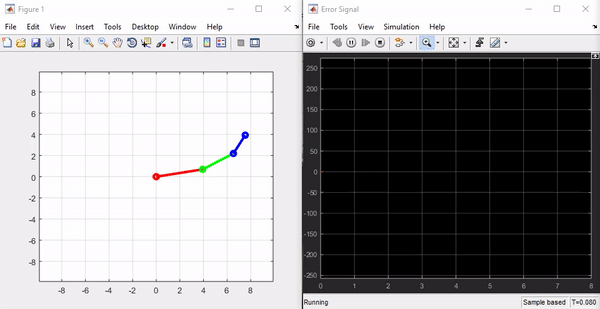


Figure 2.1.7: Simulation of Robot arm Figure 2.1.8: Graph of error signal vs time

**Observations:** From Figures 2.1.1 through 2.1.8, we observe that there is a pattern in the response of the system to varying values of Kp and Kv. As Kp increases, the amplitude of the error for each joint increases as well and the system attains steady-state faster. It is important to note that because a simulated environment assumes perfect conditions, the graphs of the error signal are extremely smooth. This might not be the case when working with a real system; we would see much more noisy hence realistic results.

**Rise Time Changes:**

Kp also affects the rise time of the system. The rise time is the time that it takes the response of system to rise from 10% to 90% of the steady-state value. Therefore, because as Kp increases the system reaches steady-state faster, the rise time of the response of the system decreases as Kp increases. In other words, the higher Kp, the lower the rise time.

**Overshoot:**

Furthermore, we could not get the gain to give overshoot.

Additionally, the system does not need the value of Kp to be different for each joint. A single value for all three joints was enough to get the system to settle relatively fast.

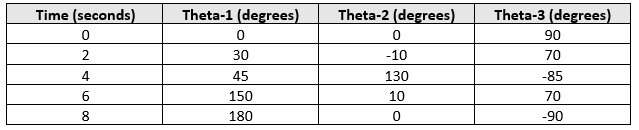
**Does the step response look like a second order system?**

Finally, the system seems to behave like a second order system when Kv takes up smaller values.

**Task 3**

**Robot Simulation with Path Trajectory**

Spline fit curves were created using the MATLAB function spline for each joint, meaning 3 curves were created and the trajectories were fed to the 3R manipulator and simulated. A teach pendant was used to maneuver to some desired point/orientation and then the joint’s coordinate values were captured. The given timeline was as follows:



The initial angles of the system were set to match those at time zero and then simulation was performed for 8 seconds.

**Figure Showing the spline curves:**

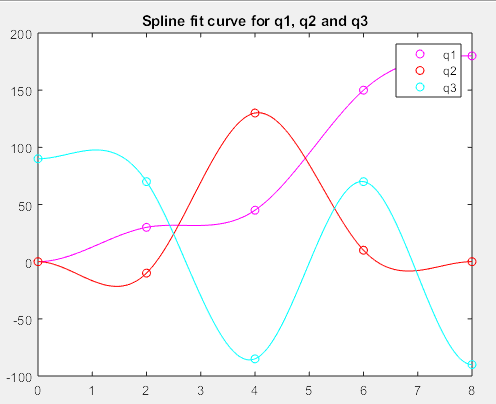
****

Figure 3.0: Spline Curves

**Simulation using the spline matlab function**

Note: The simulations have been sped up for the purposes of a concise presentation. The same speeding multiplier was applied to all videos to effectively demonstrate how changes in Kp affect the trajectory of the robot arm.

1. **Kp = 1; Kv = 2**

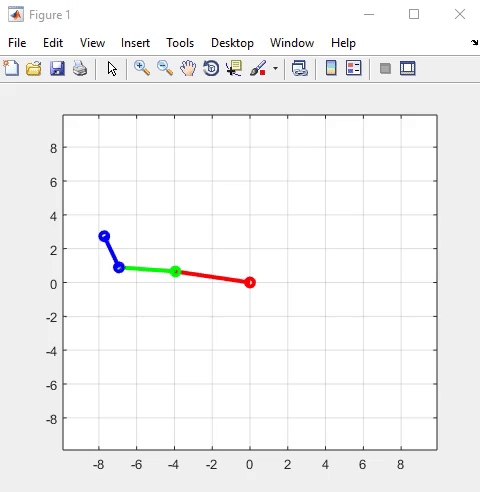
****

Figure 3.1.1: Simulation of Robot arm

1. **Kp = 10; Kv = 6.325**

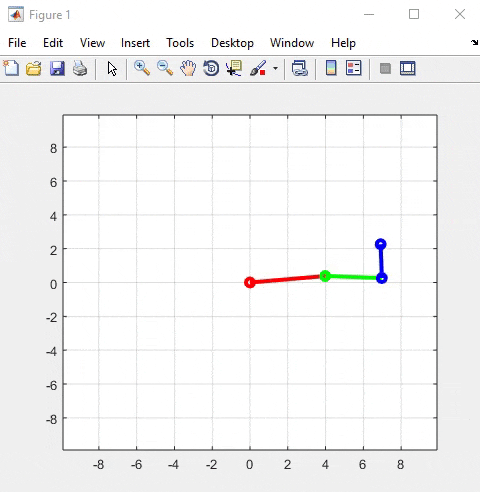
****

Figure 3.1.2: Simulation of Robot arm

Figure 3.1.2

**2. Kp = 100; Kv = 20**

****

Figure 3.1.3: Simulation of Robot arm

**3. Kp = 1000; Kv = 63.245**

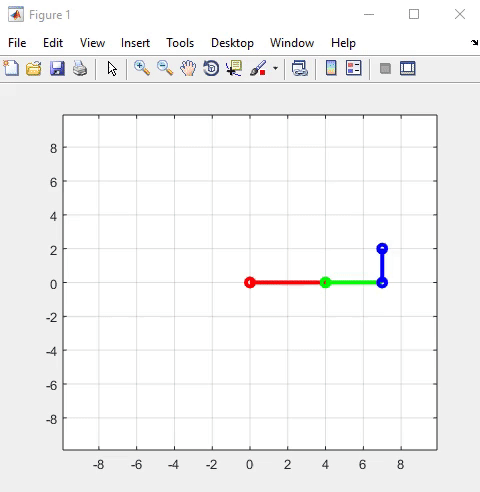
****

Figure 3.1.4: Simulation of Robot arm

**Observations:**

**Does the simulations make it to the end?**

Yes except for case 1, where Kp = 1 and Kv= 2. For all other 3 cases that we tried, the simulation did indeed make it to the end.

**Does the value of the selected gains changes the simulation?**

Yes! In fact, we noticed that as the values of Kp and, consequently Kv increase, the longer it takes the simulation to reach the final angles. Figures 3.1.1 through 3.1.4 demonstrate the changing speeds at which the robot arm follows the trajectory.

**Does the actual path follow the designed trajectory?**

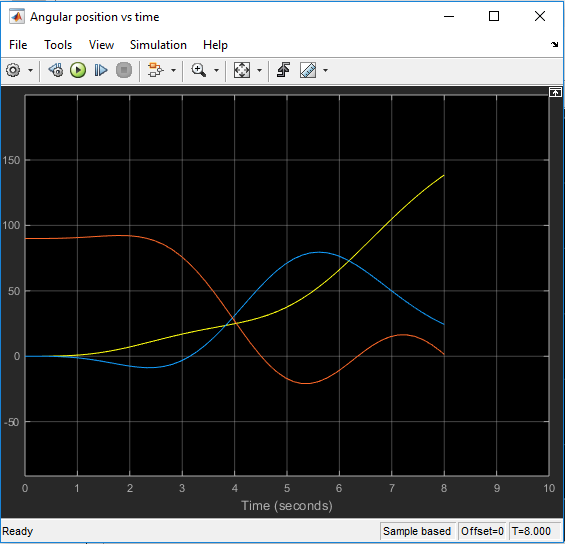
****

Figure3.2: Angular position of the Robot arm joints (1, 2, 3) vs time

After observing the scope shown in Figure 3.2, we can see that the desired angles are different than what we want them to be at the specific times. However, tuning the controller, or in other words choosing the right Kp and Kv will help us achieve our requirements better.