Project 2

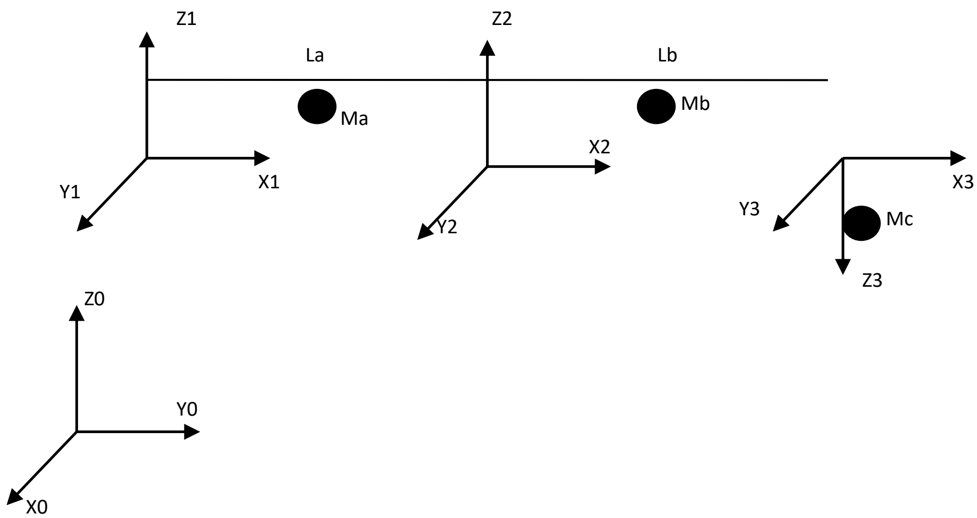
RBE 500
Prepared for Prof. Flickinger

By

Abedin Sherifi

12/13/2019

Tower of Hanoi Frame Configuration:



DH Table:

Link	α_i	a_i	d_i	θ_i
1	0	La	0	θ_1
2	180°	Lb	0	θ_2
3	0	0	D3	0

$$A_i = \begin{bmatrix} c_{\theta_i} & -s_{\theta_i}c_{\alpha_i} & s_{\theta_i}s_{\alpha_i} & a_i c_{\theta_i} \\ s_{\theta_i} & c_{\theta_i}c_{\alpha_i} & -c_{\theta_i}s_{\alpha_i} & a_i s_{\theta_i} \\ 0 & s_{\alpha_i} & c_{\alpha_i} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_1^0 = \begin{bmatrix} c_1 & -s_1 & 0 & L_a c_1 \\ s_1 & c_1 & 0 & L_b s_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_2^1 = \begin{bmatrix} c_2 & s_2 & 0 & L_b c_2 \\ s_2 & -c_2 & 0 & L_b s_2 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_3^2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & D_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_3^0 = \begin{bmatrix} c_1c_2 - s_1s_2 & c_1s_2 + c_2s_1 & 0 & L_ac_1 + L_bc_1c_2 - L_bs_1s_2 \\ c_1s_2 + c_2s_1 & s_1s_2 - c_1c_2 & 0 & L_as_1 + L_bc_1s_2 + L_bs_1c_2 \\ 0 & 0 & -1 & -D_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\theta_1 = atan2(P_y, P_x) - atan2(L_bs_2, L_a + L_bc_2)$$

$$\theta_2 = atan2(\pm\sqrt{1 - c_2}, c_2)$$

$$P_z = -D_3$$

$$c_2 = \frac{P_x^2 + P_y^2 - L_a^2 - L_b^2}{2L_aL_b}$$

$$J(q) = \begin{bmatrix} -L_as_1 - L_bs_1s_2 & -L_bs_1s_2 & 0 \\ L_ac_1 + L_bc_1c_2 & L_bc_1c_2 & 0 \\ 0 & 0 & -1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Manipulator Link Specifications:

Z0 Offset	0.5 m
Z1 Offset	0.02 m
Link 1 Properties	0.3 m length, 0.05 m x 0.06 m rectangular prism, 6 kg
Link 2 Properties	0.3 m, length, 0.05 m x 0.06 m rectangular prism, 6 kg
Link 3 Length	0.6 m x 0.05 m cylinder, 5 kg (zero position at midpoint, +z downward)
La	0.300 m
Lb	0.300 m
Lc	0.600 m
la	La/2 m
lb	Lb/2 m
lc	Lc/2 m
Ma	6 kg
Mb	6 kg
Mc	5 kg

Peg Locations:

Peg 1	[-0.2, 0.4, 0.0] m
Peg 2	[0.1, 0.4, 0.0] m
Peg 3	[0.4, 0.4, 0.0] m

Forward Kinematics:

$\dot{x}_1 = -\dot{\theta}_1 L_a \sin(\theta_1)$

$\dot{y}_1 = \dot{\theta}_1 L_a \cos(\theta_1)$

$\dot{z}_1 = 0$

Total Kinetic Energy:

K.E 1 = $\frac{1}{2}M_a * v_1^T v_1 + \frac{1}{2}J_1 * \dot{\theta}_1^2$

K.E 1 = $\frac{1}{2}M_a * l_a^2\dot{\theta}_1^2 + \frac{1}{2}J_1 * \dot{\theta}_1^2$

K.E 2 = $\frac{1}{2} * M_b[L_a^2 * \dot{\theta}_1^2 + l_b^2 * \left(\dot{\theta}_1 + \dot{\theta}_2\right)^2 + \ 2 * L_a * l_b * \dot{\theta}_1 * \left(\dot{\theta}_1 + \dot{\theta}_2\right) * \cos\left(\dot{\theta}_2\right)] + \frac{1}{2}J_2 * \left(\dot{\theta}_1 + \dot{\theta}_2\right)^2$

K.E 3 = $\frac{1}{2}J_3 * \left(\dot{D}_3\right)^2 + \ \frac{1}{2} * M_c[L_a^2 * \dot{\theta}_1^2 + l_c^2 * \dot{D}_3^2 + \ l_c^2 * \left(\dot{\theta}_1 + \dot{\theta}_2\right)^2 * \sin(D_3)^2 + \ \left(\dot{\theta}_1 + \dot{\theta}_2\right)^2 * L_b^2 - 2 * \dot{D}_3 * \left(\dot{\theta}_1 + \dot{\theta}_2\right)L_b * l_c * \cos(D_3) - 2 * \dot{\theta}_1 * \dot{D}_3 * \cos(\theta_2) * \cos(D_3) + 2 * \dot{\theta}_1 * \left(\dot{\theta}_1 + \dot{\theta}_2\right) * L_a * l_c * \sin(\theta_2) * \sin(D_3) + \ 2 * \dot{\theta}_1 * \left(\dot{\theta}_1 + \dot{\theta}_2\right) * L_a * L_b * \cos(\theta_2)\ ]$

K.E Total = K.E 1 + K.E 2 + K.E 3

Total Potential Energy:

P.E 1 = P.E 2 = 0

P.E = $P.E\ 3 = \ M_c * g * D_3$

Lagrange Equation:

L = K.E – P.E = $\frac{1}{2} * \dot{\theta}_1^2[J_1 + M_a * l_a^2] + \frac{1}{2} * J_2 * \left(\dot{\theta}_1 + \dot{\theta}_2\right)^2 + \frac{1}{2} * M_2\left[L_a^2 * \dot{\theta}_1^2 + l_b^2 * \left(\dot{\theta}_1 + \dot{\theta}_2\right)^2 + 2 * L_a * l_b * \dot{\theta}_1 * \left(\dot{\theta}_1 + \dot{\theta}_2\right) * \cos\left(\dot{\theta}_2\right)\right] + \frac{1}{2}J_3 * \left(\dot{D}_3\right)^2 + \frac{1}{2} * M_c[L_a^2 * \dot{\theta}_1^2 + l_c^2 * \dot{D}_3^2 + \ l_c^2 * \left(\dot{\theta}_1 + \dot{\theta}_2\right)^2 * \sin(D_3)^2 + \ \left(\dot{\theta}_1 + \dot{\theta}_2\right)^2 * L_b^2 - 2 * \dot{D}_3 * \left(\dot{\theta}_1 + \dot{\theta}_2\right)L_b * l_c * \cos(D_3) - 2 * \dot{\theta}_1 * \dot{D}_3 * \cos(\theta_2) * \cos(D_3) + 2 * \dot{\theta}_1 * \left(\dot{\theta}_1 + \dot{\theta}_2\right) * L_a * l_c * \sin(\theta_2) * \sin(D_3) + \ 2 * \dot{\theta}_1 * \left(\dot{\theta}_1 + \dot{\theta}_2\right) * L_a * L_b * \cos(\theta_2)] - M_c * g * D_3$

Dynamic Equations:

$$\tau_{\theta 1} = \ \frac{d}{dt}\left[\frac{\delta L}{\delta \dot{\theta}_1}\right] - \ \frac{\delta L}{\delta \theta_1}$$

$$\tau_{\theta 2} = \ \frac{d}{dt}\left[\frac{\delta L}{\delta \dot{\theta}_2}\right] - \ \frac{\delta L}{\delta \theta_2}$$

$$0 = \ \frac{d}{dt}\left[\frac{\delta L}{\delta \dot{D}_3}\right] - \ \frac{\delta L}{\delta D_3}$$

$$\tau = D(\theta)\ddot{\theta} + C\left(\theta, \dot{\theta}\right)\dot{\theta} + G(\theta)$$

Joint Displacement:

$$q = \begin{bmatrix}\theta_1 \\ \theta_2 \\ D_3\end{bmatrix}$$

Mass Inertia:

$$D_{11} = J_1 + J_2 + M_a * l_a^2 + M_b * L_a^2 + M_b * l_b^2 + M_c * L_a^2 + M_c * L_b^2 + 2 * M_b * L_a * l_b * \cos(\theta_2) + M_c * l_c^2 * \sin(D_3)^2 + 2 * M_c * L_a * l_c * \sin(\theta_2) * \sin(D_3) + 2 * M_c * L_a * L_b * cos(\theta_2)$$

$$D_{12} = J_2 + M_b * l_b^2 + M_c * L_b^2 + M_b * L_a * l_b * \cos(\theta_2) + M_c * L_a * l_c * \sin(\theta_2) * \sin(D_3) + M_c * L_a * L_b * cos(\theta_2) + M_c * l_c^2 * \sin(D_3)^2$$

$$D_{13} = -M_c * L_b * l_c * \cos(D_3) - M_c * L_a * l_c * cos(\theta_2) * \cos(D_3)$$

$$D_{21} = D_{12}$$

$$D_{22} = J_2 + M_b * l_b^2 + M_c * L_b^2 + M_c * l_c^2 * \sin(D_3)^2$$

$$D_{23} = -M_c * L_b * l_c * \cos(D_3)$$

$$D_{31} = D_{13}$$

$$D_{32} = D_{23}$$

$$D_{33} = J_3 + M_c * l_c^2$$

Coriolis Force:

$$C_{11} = -2 * \dot{\theta}_2 * M_b * L_a * l_b * \sin(\theta_2) + \dot{D}_3 * M_c * l_c^2 * \sin(2 * \theta_2) + 2 * \dot{\theta}_2 * M_c * L_a * l_c * \cos(\theta_2) * \sin(D_3) + 2 * \dot{D}_3 * M_c * L_a * l_c * sin(\theta_2) * \cos(D_3) - 2 * \dot{\theta}_2 * M_c * L_a * L_b * \sin(\theta_2)$$

$$C_{12} = -\dot{\theta}_2 * M_b * L_a * l_b * \sin(\theta_2) + \dot{D}_3 * M_c * l_c^2 * \sin(2 * D_3) + 2 * \dot{D}_3 * M_c * L_a * l_c * \sin(\theta_2) * \cos(D_3) + \dot{\theta}_2 * M_c * L_a * l_c * cos(\theta_2) * \sin(D_3) - \dot{\theta}_2 * M_c * L_a * L_b * \sin(\theta_2)$$

$$C_{13} = \dot{D}_3 * M_c * l_c * L_b * \sin(D_3) + \dot{D}_3 * M_c * L_a * l_c * \cos(\theta_2) * \sin(D_3)$$

$$C_{21} = -3 * \dot{\theta}_2 * M_b * L_a * l_b * \sin(\theta_2) + \dot{D}_3 * M_c * l_c^2 * \sin(2 * D_3) - 2 * \dot{\theta}_1 * M_b * L_a * l_b * \sin(\theta_2) - \dot{\theta}_1 * M_c * L_a * l_c * sin(D_3) * \cos(\theta_2) + \dot{\theta}_1 * M_c * L_a * L_b * \sin(\theta_2)$$

$$C_{22} = \dot{D}_3 * M_c * l_c^2 * \sin(2 * D_3)$$

$$C_{23} = \dot{D}_3 * M_c * L_b * l_c * \sin(D_3)$$

$$C_{31} = \dot{D}_3 * M_c * l_c * L_a * \cos(\theta_2) * \cos(D_3) + \frac{1}{2} * \dot{\theta}_1 * M_c * l_c^2 * \sin(2 * D_3) + \dot{\theta}_2 * M_c * l_c^2 * \sin(2 * \theta_2) - \dot{\theta}_1 * M_c * l_c * L_a * sin(D_3) * \cos(\theta_2) - \dot{\theta}_1 * M_c * l_c * L_a * \sin(\theta_2) * \sin(D_3)$$

Gravitational Force:

$G_1 = 0$
$G_2 = 0$
$G_3 = -M_c * g * lc * \sin(D_3)$

Joint Driving Torque:

$$\tau = \begin{bmatrix} \tau_{\theta 1} \\ \tau_{\theta 2} \\ 0 \end{bmatrix} = \begin{bmatrix} (D_{11} * \ddot{\theta}_1 + D_{12} * \ddot{\theta}_2 + D_{13} * \ddot{D}_3 + C_{11} * \dot{\theta}_1 + C_{12} * \dot{\theta}_2 + C_{13} * \dot{D}_3 + G_1) \\ (D_{21} * \ddot{\theta}_1 + D_{22} * \ddot{\theta}_2 + D_{23} * \ddot{D}_3 + C_{21} * \dot{\theta}_1 + C_{22} * \dot{\theta}_2 + C_{23} * \dot{D}_3 + G_2) \\ (D_{31} * \ddot{\theta}_1 + D_{32} * \ddot{\theta}_2 + D_{33} * \ddot{D}_3 + C_{31} * \dot{\theta}_1 + C_{32} * \dot{\theta}_2 + C_{33} * \dot{D}_3 + G_3) \end{bmatrix}$$

Joint Acceleration:

$$\ddot{\theta} = D(\theta)^{-1}[-C(\theta,\dot{\theta}) - G(\theta)] + \tau$$

Error signals:

$$e(\theta_1) = \theta_{1f} - \theta_1$$
$$e(\theta_2) = \theta_{2f} - \theta_2$$
$$e(D_3) = D_{3f} - D_3$$

PID Control Design:

$$\tau_M = K_p * e + K_d * \dot{e} + K_i \int e\, dt$$

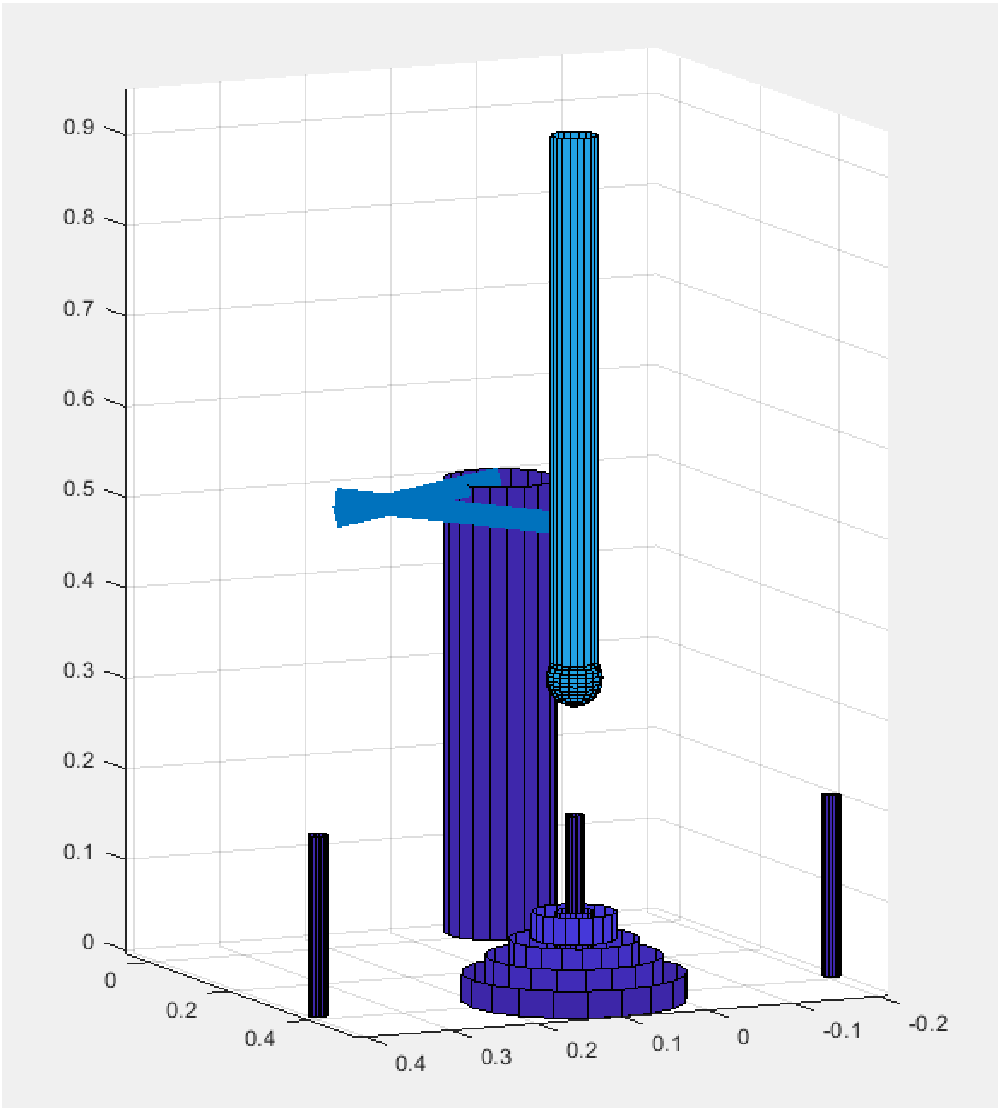
$$\tau_1 = K_{p1} * (\theta_{1f} - \theta_1) - K_{d1} * \dot{\theta}_1 + K_{i1} \int e(\theta_1)\, dt$$
$$\tau_2 = K_{p2} * (\theta_{2f} - \theta_2) - K_{d2} * \dot{\theta}_2 + K_{i2} \int e(\theta_2)\, dt$$
$$0 = K_{p3} * (D_{3f} - D_3) - K_{d3} * \dot{D}_3 + K_{i3} \int e(D_3)\, dt$$

$$x_1 = \int e(\theta_1)\, dt => \dot{x}_1 = \theta_{1f} - \theta_1$$
$$x_2 = \int e(\theta_2)\, dt => \dot{x}_2 = \theta_{2f} - \theta_2$$
$$x_3 = \int e(D_3)\, dt => \dot{x}_3 = D_{3f} - D_2$$

PID parameter effect to system dynamics:

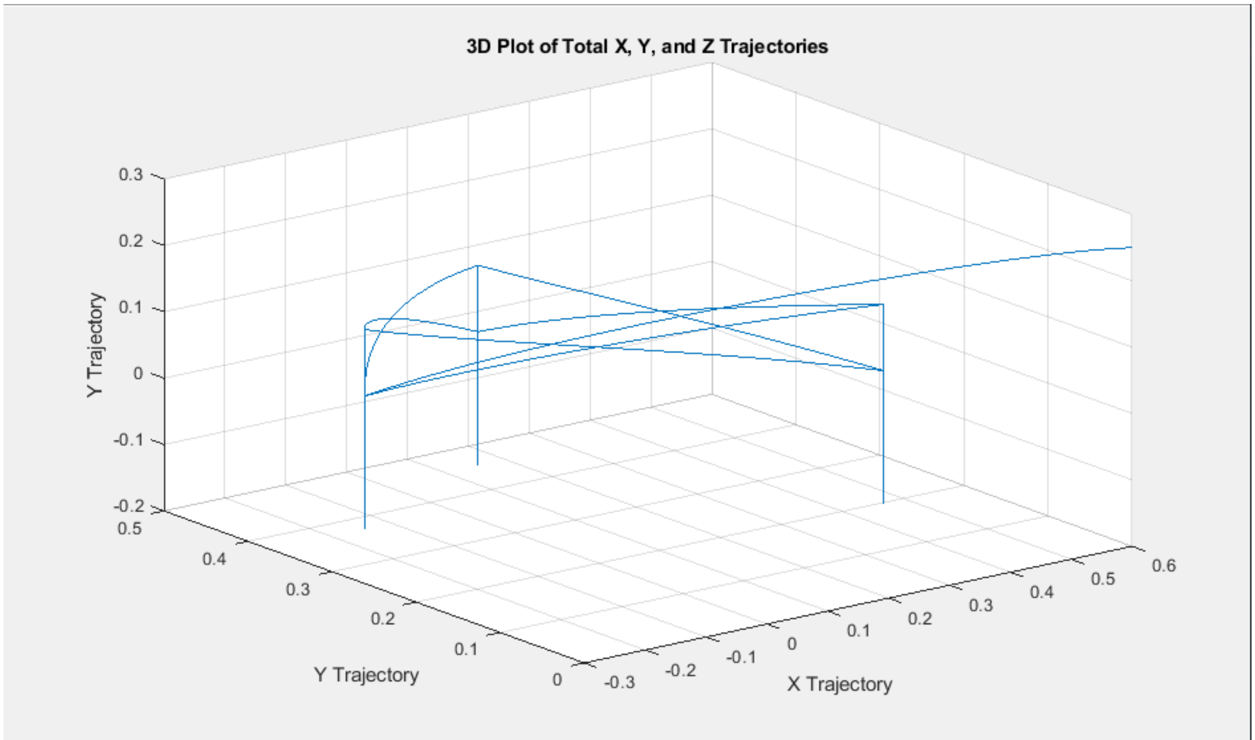
Response	Rise Time	Overshoot	Settling Time	S-S Error
Kp	Decrease	Increase	Minor Change	Decrease
Ki	Decrease	Increase	Increase	Eliminate
Kd	Minor Change	Decrease	Decrease	Minor Change

Tower of Hanoi Final Configuration:



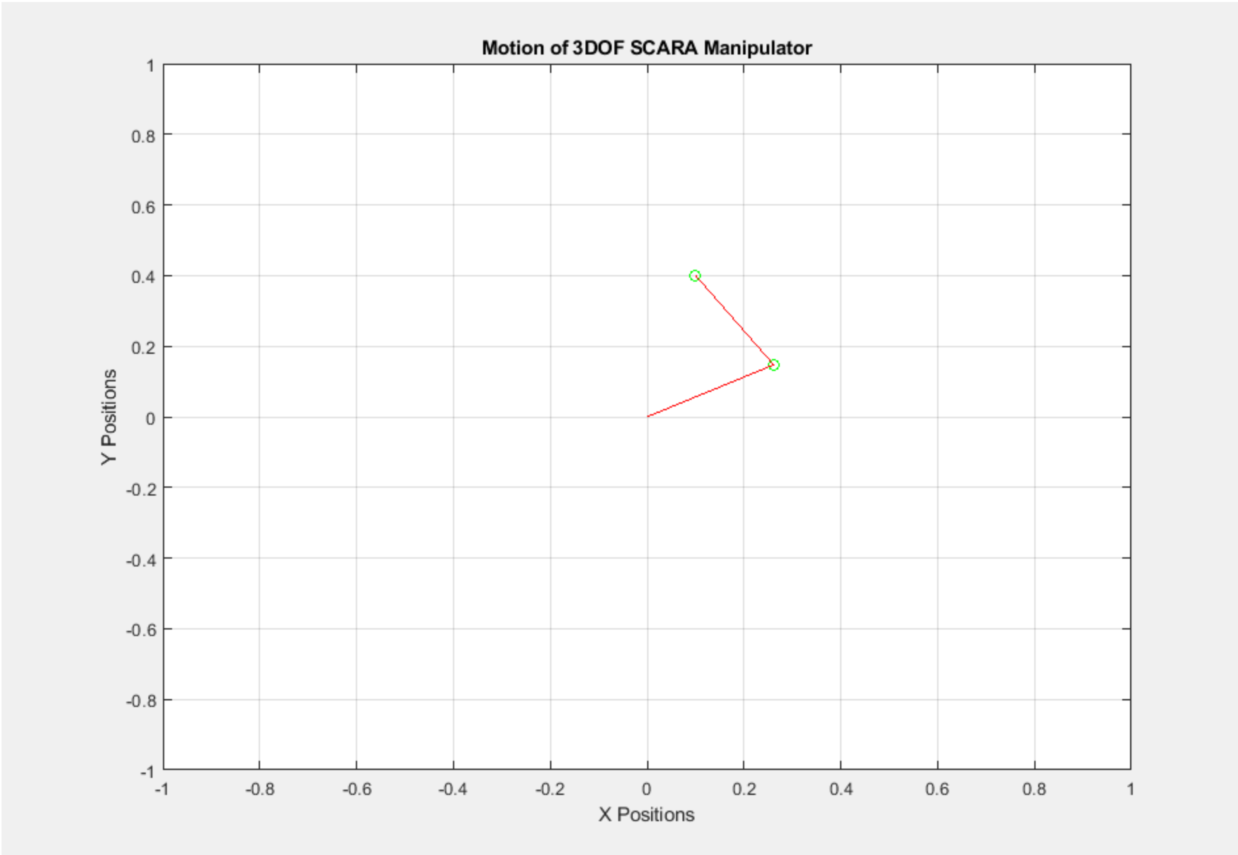
The figure above depicts the final configuration of the Tower of Hanoi. All the disks from the right peg (peg 1) are moved to the middle peg (peg 2). The same disks could be moved to the leftmost peg (peg 3) with just few programming changes. It takes a minimum of 15 primitive steps to accomplish the whole move of disks considering that smaller disks are not allowed to be located below bigger disks. A video showing the whole simulation will be included separately with this report.

3D Plot of Total X, Y and Z Components of the Full Trajectory:



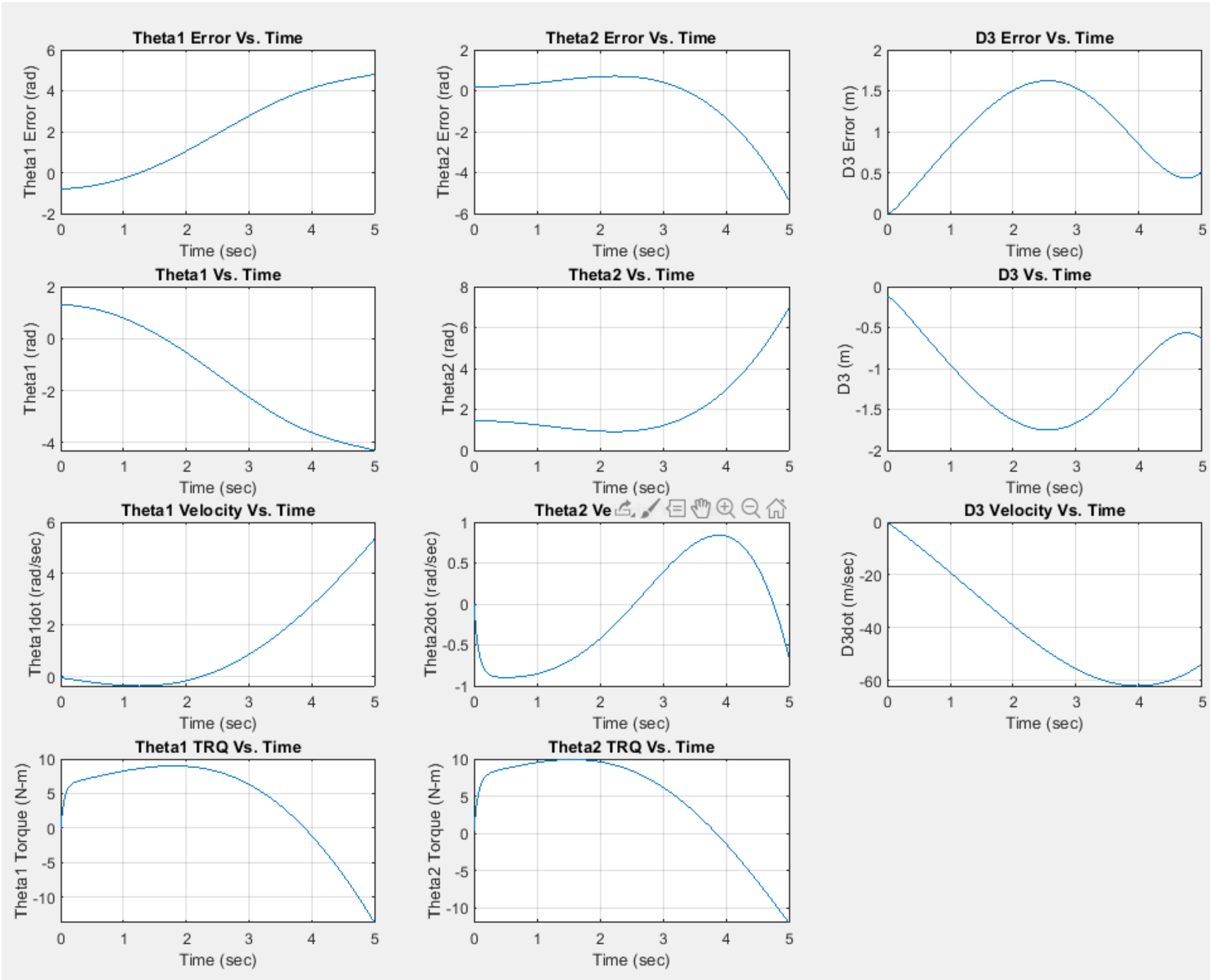
The figure above plots the x, y and z components of the full trajectory of the manipulator. As it can be observed from the figure above that the end effector is moved to all three different pegs.

Full Animation of the 3-DOF SCARA Manipulator Implementation by the Tower of Hanoi:



The figure shown above displays a 2D animation of just the RR section of the SCARA (RRP) manipulator. A video is going to be included separately to this report to show the full animation. It is easily seen that the manipulator moves the end effector to three distinct positions which represent the three different pegs.

Error, Joint Variable Positions, Velocities, and Torque Plots:



The figure above depicts multiple plots. The first set of plots show joint variable errors versus time. It is of importance to note that PID controller was implemented only for one distinct move of the end effector (between two pegs only). Due to time limitations, the PID controller was not implemented throughout the whole trajectory. Additional PID tuning work is required in order to bring the steady state error approach zero since in our situation that is not the case. The next sets of plots are the joint variable positions versus time. Here, all three joint variable position plots correctly depict the move of the manipulator from the initial peg to the goal peg. The third row of plots represents the joint variable velocities versus time. Again, these are correctly displayed and it can be easily verified by the increase in velocity for all three joint variables. The last row depicts the torque plots. Only two torque plots are shown since the third torque working on the vertical prismatic joint is assumed to be zero.

Appendix A

(Matlab Code)

```
% Abedin Sherifi
% RBE 500
% Project 2
% 12/14/2019

function Tower_of_Hanoi()
cp_data = [0, 0, -.25, 0];

%% Move #1
%Grasp disc 4 from Peg 1 and ungrasp on Peg 3 in disc position 1
[th1_goal, th2_goal, d3_goal, cp_data] = grasp_disc(0.0, 0.0, -0.25, 1.0,
4.0, cp_data);
[th1_goal, th2_goal, d3_goal, cp_data] = ungrasp_disc(th1_goal, th2_goal,
d3_goal, 3.0, 1.0, cp_data);

%% Move #2
%Grasp disc 3 from Peg 1 and ungrasp on Peg 2 in disc position 1
[th1_goal, th2_goal, d3_goal, cp_data] = grasp_disc(th1_goal, th2_goal,
d3_goal, 1.0, 3.0, cp_data);
[th1_goal, th2_goal, d3_goal, cp_data] = ungrasp_disc(th1_goal, th2_goal,
d3_goal, 2.0, 1.0, cp_data);

%% Move #3
%Grasp disc 1 from Peg 3, and ungrasp on Peg 2, in disc postion 2
[th1_goal, th2_goal, d3_goal, cp_data] = grasp_disc(th1_goal, th2_goal,
d3_goal, 3.0, 1.0, cp_data);
[th1_goal, th2_goal, d3_goal, cp_data] = ungrasp_disc(th1_goal, th2_goal,
d3_goal, 2.0, 2.0, cp_data);

%% Move #4
%Grasp disc 2 from Peg 1, and ungrasp on Peg 3, in disc postion 1
[th1_goal, th2_goal, d3_goal, cp_data] = grasp_disc(th1_goal, th2_goal,
d3_goal, 1.0, 2.0, cp_data);
[th1_goal, th2_goal, d3_goal, cp_data] = ungrasp_disc(th1_goal, th2_goal,
d3_goal, 3.0, 1.0, cp_data);

%% Move #5
%Grasp disc 2 from Peg 2, and ungrasp on Peg 1, in disc postion 2
[th1_goal, th2_goal, d3_goal, cp_data] = grasp_disc(th1_goal, th2_goal,
d3_goal, 2.0, 2.0, cp_data);
[th1_goal, th2_goal, d3_goal, cp_data] = ungrasp_disc(th1_goal, th2_goal,
d3_goal, 1.0, 2.0, cp_data);

%% Move #6
%Grasp disc 1 from Peg 2, and ungrasp on Peg 3, in disc postion 2
[th1_goal, th2_goal, d3_goal, cp_data] = grasp_disc(th1_goal, th2_goal,
d3_goal, 2.0, 1.0, cp_data);
[th1_goal, th2_goal, d3_goal, cp_data] = ungrasp_disc(th1_goal, th2_goal,
d3_goal, 3.0, 2.0, cp_data);

%% Move #7
%Grasp disc 2 from Peg 1, and ungrasp on Peg 3, in disc postion 3
```

```
[th1_goal, th2_goal, d3_goal, cp_data] = grasp_disc(th1_goal, th2_goal,
d3_goal, 1.0, 2.0, cp_data);
[th1_goal, th2_goal, d3_goal, cp_data] = ungrasp_disc(th1_goal, th2_goal,
d3_goal, 3.0, 3.0, cp_data);

%% Move #8
%Grasp disc 1 from Peg 1, and ungrasp on Peg 2, in disc postion 1
[th1_goal, th2_goal, d3_goal, cp_data] = grasp_disc(th1_goal, th2_goal,
d3_goal, 1.0, 1.0, cp_data);
[th1_goal, th2_goal, d3_goal, cp_data] = ungrasp_disc(th1_goal, th2_goal,
d3_goal, 2.0, 1.0, cp_data);

%% Move #9
%Grasp disc 3 from Peg 3, and ungrasp on Peg 2, in disc postion 2
[th1_goal, th2_goal, d3_goal, cp_data] = grasp_disc(th1_goal, th2_goal,
d3_goal, 3.0, 3.0, cp_data);
[th1_goal, th2_goal, d3_goal, cp_data] = ungrasp_disc(th1_goal, th2_goal,
d3_goal, 2.0, 2.0, cp_data);

%% Move #10
%Grasp disc 2 from Peg 3, and ungrasp on Peg 1, in disc postion 1
[th1_goal, th2_goal, d3_goal, cp_data] = grasp_disc(th1_goal, th2_goal,
d3_goal, 3.0, 2.0, cp_data);
[th1_goal, th2_goal, d3_goal, cp_data] = ungrasp_disc(th1_goal, th2_goal,
d3_goal, 1.0, 1.0, cp_data);

%% Move #11
%Grasp disc 2 from Peg 2, and ungrasp on Peg 1, in disc postion 2
[th1_goal, th2_goal, d3_goal, cp_data] = grasp_disc(th1_goal, th2_goal,
d3_goal, 2.0, 2.0, cp_data);
[th1_goal, th2_goal, d3_goal, cp_data] = ungrasp_disc(th1_goal, th2_goal,
d3_goal, 1.0, 2.0, cp_data);

%% Move #12
%Grasp disc 1 from Peg 3, and ungrasp on Peg 2, in disc postion 2
[th1_goal, th2_goal, d3_goal, cp_data] = grasp_disc(th1_goal, th2_goal,
d3_goal, 3.0, 1.0, cp_data);
[th1_goal, th2_goal, d3_goal, cp_data] = ungrasp_disc(th1_goal, th2_goal,
d3_goal, 2.0, 2.0, cp_data);

%% Move #13
%Grasp disc 2 from Peg 1, and ungrasp on Peg 3, in disc postion 1
[th1_goal, th2_goal, d3_goal, cp_data] = grasp_disc(th1_goal, th2_goal,
d3_goal, 1.0, 2.0, cp_data);
[th1_goal, th2_goal, d3_goal, cp_data] = ungrasp_disc(th1_goal, th2_goal,
d3_goal, 3.0, 1.0, cp_data);

%% Move #14
%Grasp disc 1 from Peg 1, and ungrasp on Peg 2, in disc postion 3
[th1_goal, th2_goal, d3_goal, cp_data] = grasp_disc(th1_goal, th2_goal,
d3_goal, 1.0, 1.0, cp_data);
[th1_goal, th2_goal, d3_goal, cp_data] = ungrasp_disc(th1_goal, th2_goal,
d3_goal, 2.0, 3.0, cp_data);

%% Move #15
```

```
%Grasp disc 1 from Peg 3, and ungrasp on Peg 2, in disc postion 4
[th1_goal, th2_goal, d3_goal, cp_data] = grasp_disc(th1_goal, th2_goal,
d3_goal, 3.0, 1.0, cp_data);
[th1_goal, th2_goal, d3_goal, cp_data] = ungrasp_disc(th1_goal, th2_goal,
d3_goal, 2.0, 4.0, cp_data);

% Write data to text file
dlmwrite("sim_data.txt", cp_data, " ");
end
```

```
% Abedin Sherifi
% RBE 500
% Project 2
% 12/14/2019

function output=manipulator_3dof_scara_project2(t,x,th_goal,manip_det,PIDs)
output=zeros(10,1);

%% Goal positions
th1_f=th_goal(1);
th2_f=th_goal(2);
d3_f=th_goal(3);

%% Manipulator Details
Ma=manip_det(4);
Mb=manip_det(5);
Mc=manip_det(6);
La=manip_det(1);
Lb=manip_det(2);
Lc=manip_det(3);
la = La/2;
lb = Lb/2;
lc = Lc/2;
g=9.8;

%% Jacobian Values
J1 = 2.000*10^(-4);
J2 = 5.000*10^(-5);
J3 = 10.000*10^(-4);

%% Inertia Matrix
d11=J1 + J2 +
Ma*la^2+Mb*La^2+Mc*la^2+Mc*Lb^2+2*Mb*La*lb*cos(th2_f)+Mc*lc^2*(sin(d3_f))^2+2
*Mc*La*lc*sin(th2_f)*sin(d3_f)+2*Mc*La*Lb*cos(th2_f);
d12=J2 +
Mb*lb^2+Mc*Lb^2+Mb*La*lb*cos(th2_f)+Mc*La*lc*sin(th2_f)*sin(d3_f)+Mc*La*Lb*co
s(th2_f)+Mc*lc^2*(sin(d3_f))^2;
d13=-Mc*Lb*lc*cos(d3_f)-Mc*La*lc*cos(th2_f)*cos(d3_f);
d21 = d12;
d22 = J2 + Mb*lb^2+Mc*Lb^2+ Mc*lc^2*(sin(d3_f))^2;
d23 = -Mc*Lb*lc*cos(d3_f);
d31 = d13;
d32 = d23;
d33 = J3 + Mc*lc^2;
Dq=[d11 d12 d13;d21 d22 d23;d31 d32 d33];

%% Coriolis Matrix
th1d = x(5); % Theta1 dot
th2d = x(6); % Theta2 dot
d3d = x(7);  % D3 dot

c11 = -
2*th2d*Mb*La*lb*sin(th2_f)+d3d*Mc*lc^2*sin(2*th2_f)+2*th2d*Mc*La*lc*cos(th2_f
)*sin(d3_f)+...
    2*d3d*Mc*La*lc*sin(th2_f)*cos(d3_f)-2*th2d*Mc*La*Lb*sin(th2_f);
```

```
c12 = -
th2d*Mb*La*lb*sin(th2_f)+d3d*Mc*lc^2*sin(2*d3_f)+2*d3d*Mc*La*lc*sin(th2_f)*co
s(d3_f)+...
    th2d*Mc*La*lc*cos(th2_f)*sin(d3_f)-th2d*Mc*La*Lb*sin(th2_f);
c13 = d3d*Mc*lc*Lb*sin(d3_f)+d3d*Mc*La*lc*cos(th2_f)*sin(d3_f);
c21 = -3*th2d*Mb*La*lb*sin(th2_f)+d3d*Mc*lc^2*sin(2*d3_f)-
2*th1d*Mb*La*lb*sin(th2_f)-...
    th1d*Mc*La*lc*sin(d3_f)*cos(th2_f)+th1d*Mc*La*Lb*sin(th2_f);
c23 = d3d*Mc*Lb*lc*sin(d3_f);
c22 = d3d*Mc*lc^2*sin(2*d3_f);
c31 =
d3d*Mc*lc*La*cos(th2_f)*cos(d3_f)+0.5*th1d*Mc*lc^2*sin(2*d3_f)+th2d*Mc*lc^2*s
in(2*th2_f)-...
    th1d*Mc*lc*La*sin(d3_f)*cos(th2_f)-th1d*Mc*lc*La*sin(th2_f)*sin(d3_f);
c32  =0;
c33 =0;
Cq=[c11 c12 c13;c21 c22 c23;c31 c32 c33];

%% Gravity Matrix
g1=0;
g2=0;
g3 = -Mc*g*lc*sin(d3_f);
Gq=[g1;g2;g3];

%% PID parameters for theta 1 and 2
Kp1=PIDs(1);
Kd1=PIDs(2);
Ki1=PIDs(3);
Kp2=PIDs(4);
Kd2=PIDs(5);
Ki2=PIDs(6);

%% Control input
trq1=Kp1*(th1_f-x(3))-Kd1*x(6)+Ki1*(x(1));
trq2=Kp2*(th2_f-x(4))-Kd2*x(7)+Ki2*(x(2));
trq=[trq1;trq2;0];
trqf=Dq*trq;

%% System states
output(1)=(th1_f-x(3));
output(2)=(th2_f-x(4));

%% Theta 1, Theta 2, and D3 positions
output(3)=x(5); % Theta1
output(4)=x(6); % Theta2
output(5)=x(7); % D3

%% Dynamic equation and joint variable velocities
q2dot=inv(Dq)*(-Cq-Gq+trqf);
output(6)=q2dot(1); %Theta1 dot
output(7)=q2dot(2); %Theta2 dot
output(8)=q2dot(3); %D3 dot

%% Torque 1 and 2 Returns
output(9)=trqf(1);
```

```
output(10)=trqf(2);
end
```

```
% Abedin Sherifi
% RBE 500
% Project 2
% 12/14/2019

close all
clear all
clc

%% Manipulator Details
La=0.300;                                                   %Link A in meters
Lb=0.300;                                                   %Link B in meters
Lc=0.600;                                                   %Link C in meters
Ma=6;                                                       %Mass A in Kg
Mb=6;                                                       %Mass B in Kg
Mc=5;                                                       %Mass C in Kg
manip_det=[La Lb Lc Ma Mb Mc];

%% Initial setup and known variables
th_initial=[1.3077 1.4535 -0.13];                          %Initial positions
th_goal=[0.51341 1.6248 -0.13];                            %Goal positions
initial=[0 0 th_initial 0 0 0 0 0];                        %Initial state values

%% Time duration
t0 = 0.0;
tspan = t0:0.01:5;

%% PID parameters for theta 1 and 2 based on manual pid tunning
Kp1=1;
Kd1=20;
Ki1=10;
Kp2=1;
Kd2=15;
Ki2=10;
PIDs=[Kp1 Kd1 Ki1 Kp2 Kd2 Ki2];

%% ODE45 implemented in solving the ordinary differential equations
[T,X] = ode45(@(t,x)
manipulator_3dof_scara_project2(t,x,th_goal,manip_det,PIDs),tspan,initial);

%% Output joint positions and velocities
th1=X(:,3);                     %Theta1 position
th2=X(:,4);                     %Theta2 position
d3=X(:,5);                      %Theta1 position
th1dot=X(:,6);                  %Theta1 velocity
th2dot=X(:,7);                  %Theta2 velocity
d3dot=X(:,8);

%% Torque equations while taking into consideration the 15 N-m stall torque
trq1=diff(X(:,9))./diff(T);
trq2=diff(X(:,10))./diff(T);
ttime=0:(T(end)/(length(trq1)-1)):T(end);

%% Error plot for Theta1
figure
```

```
subplot(4,3,1)
plot(T,th_goal(1)-th1)
grid
title('Theta1 Error Vs. Time')
ylabel('Theta1 Error (rad)')
xlabel('Time (sec)')

%% Error plot for Theta2
subplot(4,3,2)
plot(T,th_goal(2)-th2)
grid
title('Theta2 Error Vs. Time')
ylabel('Theta2 Error (rad)')
xlabel('Time (sec)')

%% Error plot for D3
subplot(4,3,3)
plot(T,th_goal(3)-d3)
grid
title('D3 Error Vs. Time')
ylabel('D3 Error (m)')
xlabel('Time (sec)')

%% Plot for Theta1 Position
subplot(4,3,4)
plot(T,th1)
grid
title('Theta1 Vs. Time')
ylabel('Theta1 (rad)')
xlabel('Time (sec)')

%% Plot for Theta2 Position
subplot(4,3,5)
plot(T,th2)
grid
title('Theta2 Vs. Time')
ylabel('Theta2 (rad)')
xlabel('Time (sec)')

%% Plot for D3 Position
subplot(4,3,6)
plot(T,d3)
grid
title('D3 Vs. Time')
ylabel('D3 (m)')
xlabel('Time (sec)')

%% Plot for Theta1 Velocity
subplot(4,3,7)
plot(T,th1dot)
grid
title('Theta1 Velocity Vs. Time')
ylabel('Theta1dot (rad/sec)')
xlabel('Time (sec)')
```

```
%% Plot for Theta2 Velocity
subplot(4,3,8)
plot(T,th2dot)
grid
title('Theta2 Velocity Vs. Time')
ylabel('Theta2dot (rad/sec)')
xlabel('Time (sec)')

%% Plot for D3 Velocity
subplot(4,3,9)
plot(T,d3dot)
grid
title('D3 Velocity Vs. Time')
ylabel('D3dot (m/sec)')
xlabel('Time (sec)')

%% Plot for Torque 1
subplot(4,3,10)
plot(ttime,trq1)
grid
title('Theta1 TRQ Vs. Time')
ylabel('Theta1 Torque (N-m)')
xlabel('Time (sec)')

%% Plot for Torque 2
subplot(4,3,11)
plot(ttime,trq2)
grid
title('Theta2 TRQ Vs. Time')
ylabel('Theta2 Torque (N-m)')
xlabel('Time (sec)')
```

```matlab
% Abedin Sherifi
% RBE 500
% Project 2
% 12/14/2019

function [th1_goal, th2_goal, d3_goal, cp_data] = grasp_disc(th1_init,
th2_init, d3_init, peg, disc_pos, cp_data)
%% Peg location, Peg Specifications, Disk Specifications, Link Specifications
pegs = [-0.2, 0.4, 0.0; 0.1, 0.4, 0.0; 0.4, 0.4, 0.0];
peg_Height = 0.2;
z0_offset = 0.5;
z1_offset = 0.02;
Link_d = 0.3;
Link3_Length = 0.6;
Disk_Height = 0.03;

%% Inverse Kinematics
ikine_th2 = @(x, y, l1, l2) (atan2(sqrt(1-(x^2 + y^2 - l1^2 - l2^2) / (2 * l1
* l2)), (x^2 + y^2 - l1^2 - l2^2) / (2 * l1 * l2)));
ikine_th1 = @(x, y, l1, l2, th2) (atan2(y, x) - atan2(l2 * sin(th2), l1 + l2
* cos(th2)));

% number of steps for each path segment
num_steps = 20;

%% Define first segment
th2_start = th2_init;
th2_goal = ikine_th2(pegs(peg,1), pegs(peg,2), Link_d, Link_d);

th1_start = th1_init;
th1_goal = ikine_th1(pegs(peg,1), pegs(peg,2), Link_d, Link_d, th2_goal);

%% set end point above peg:
above_peg = -0.05 + z0_offset + z1_offset - peg_Height - Link3_Length * .5;
above_disk = z0_offset + z1_offset - disc_pos * Disk_Height - Link3_Length *
0.5 - .02;

d3_start = d3_init;
d3_goal = above_peg;

%% Generate data
th1 = th1_start:(th1_goal - th1_start)/num_steps:th1_goal;
th2 = th2_start:(th2_goal - th2_start)/num_steps:th2_goal;
d3 = d3_start:(d3_goal - d3_start)/num_steps:d3_goal;
grip = ones(1, num_steps+1);

cp_data = [cp_data; th1', th2', d3', grip'];

%% Define second segment
d3_start = d3_goal;
d3_goal = above_disk;

th1 = th1_goal * ones(1, num_steps+1);
th2 = th2_goal * ones(1, num_steps+1);
d3 = d3_start:(d3_goal - d3_start)/num_steps:d3_goal;
```

```
grip = ones(1, num_steps+1);


cp_data = [cp_data; th1', th2', d3', grip'];

%% Define third segment
d3_start = d3_goal;
d3_goal = above_peg - 0.1;

d3 = d3_start:(d3_goal - d3_start)/num_steps:d3_goal;

cp_data = [cp_data; th1', th2', d3' grip'];
end
```

```
% Abedin Sherifi
% RBE 500
% Project 2
% 12/14/2019

function [th1_goal, th2_goal, d3_goal, cp_data] = ungrasp_disc(th1_init,
th2_init, d3_init, peg, disc, cp_data)
%% Peg location, Peg Specifications, Disk Specifications, Link Specifications
pegs = [-0.2, 0.4, 0.0; 0.1, 0.4, 0.0; 0.4, 0.4, 0.0];
peg_Height = 0.2;
z0_offset = 0.5;
z1_offset = 0.02;
Link_d = 0.3;
Link3_Length = 0.6;
Disk_Height = 0.03;

%% Inverse Kinematics
ikine_th2 = @(x, y, l1, l2) (atan2(sqrt(1-(x^2 + y^2 - l1^2 - l2^2) / (2 * l1
* l2)), (x^2 + y^2 - l1^2 - l2^2) / (2 * l1 * l2)));
ikine_th1 = @(x, y, l1, l2, th2) (atan2(y, x) - atan2(l2 * sin(th2), l1 + l2
* cos(th2)));

% number of steps for each path segment
num_steps = 20;

%% Define fourth segment
th1_start = th1_init;
th2_start = th2_init;
d3_start = d3_init;

th2_goal = ikine_th2(pegs(peg,1), pegs(peg, 2), Link_d, Link_d);
th1_goal = ikine_th1(pegs(peg,1), pegs(peg, 2), Link_d, Link_d, th2_goal);

%% set end point above peg:
above_peg = -0.05 + z0_offset + z1_offset - peg_Height - Link3_Length * .5;
above_disk = z0_offset + z1_offset - disc * Disk_Height - Link3_Length * 0.5
- .02;

d3_goal = above_peg;

th1 = th1_start:(th1_goal - th1_start)/num_steps:th1_goal;
th2 = th2_start:(th2_goal - th2_start)/num_steps:th2_goal;
d3 = d3_start:(d3_goal - d3_start)/num_steps:d3_goal;
grip = ones(1, num_steps+1);

cp_data = [cp_data; th1', th2', d3' grip'];


%% Define fifth segment
d3_start = d3_goal;
d3_goal = above_disk;

th1 = th1_goal * ones(1, num_steps+1);
th2 = th2_goal * ones(1, num_steps+1);
```

```
d3 = d3_start:(d3_goal - d3_start)/num_steps:d3_goal;
grip = ones(1, num_steps+1);

cp_data = [cp_data; th1', th2', d3', grip'];

%% Define sixth segment
d3_start = d3_goal;
d3_goal = above_peg - 0.1;

d3 = d3_start:(d3_goal - d3_start)/num_steps:d3_goal;
grip = zeros(1, num_steps+1);

cp_data = [cp_data; th1', th2', d3' grip'];
end
```

```
% Abedin Sherifi
% RBE 500
% Project 2
% 12/14/2019

%Animation of the manipulator
function anim(fname)

fdata = dlmread(fname);          % Read sim_data file

d = 5;                           % Step Size
T = size(fdata,1);               % Time size
j = 1:d:T;                       % j is a vector with values starting with 1
and with end value T incrementing by d steps

% Link specifications
La=0.300;
Lb=0.300;

%%
    % Assign size of fdata to idx in order to get th1, th2, and d3
    % numbers for the whole trajectory.
    for idx = 1:size(fdata,1)

        % set robot states from fdata
        th1 = fdata(idx,1);                     % Theta1 total trajectory
values
        th2 = fdata(idx,2);                     % Theta2 total trajectory
values

        y1(idx)=La.*sin(th1);                   % Y1 position for whole traj
        x1(idx)=La.*cos(th1);                   % X1 position for whole traj
        y2(idx)=La.*sin(th1)+Lb.*sin(th1+th2);  % Y2 position for whole traj
        x2(idx)=La.*cos(th1)+Lb.*cos(th1+th2);  % X2 position for whole traj
    end

%%
% 2D animation generation
figure

for i=1:length(j)-1
hold off
plot([x1(j(i)) x2(j(i))],[y1(j(i)) y2(j(i))],'g--o',[0 x1(j(i))],[0
y1(j(i))],'r',[x1(j(i)) x2(j(i))],[y1(j(i)) y2(j(i))],'r');
title('Motion of 3DOF SCARA Manipulator')
xlabel('X Positions')
ylabel('Y Positions')
axis([-1 1 -1 1]);
grid
hold on
animation(i)=getframe(gcf);
end
drawnow;
end
```

```
% Abedin Sherifi
% RBE 500
% Project 2
% 12/14/2019

% Full trajectory generation plots of x,y, and z positions
function traj_gen(fname)

        % Read Data File
        fdata = dlmread(fname);

        % La and Lb specifications
        La=0.300;                                                           %
Link A in meters
        Lb=0.300;                                                           %
Link B in meters


    % Assign size of fdata to idx in order to get th1, th2, and d3
    % numbers for the whole trajectory.
    for idx = 1:size(fdata,1)

        th1 = fdata(idx,1);                                                 %
Theta1 total trajectory values
        th2 = fdata(idx,2);                                                 %
Theta2 total trajectory values
        d3 = fdata(idx,3);                                                  %
D3 total trajectory values

        % Calculating x,y, and z values for the total iterations of
        % th1,th2, and d3 throughout the whole trajectory.
        x(idx) = La*cos(th1)+Lb*cos(th1)*cos(th2)-Lb*sin(th1)*sin(th2);     %
X total position trajectory
        y(idx) = La*sin(th1)+Lb*cos(th1)*sin(th2)-Lb*cos(th2)*sin(th1);     %
Y total position trajectory
        z(idx) = -d3;                                                       %
Z total position trajectory
    end

    % Plot
    figure
    plot3(x,y,z)
    grid
    title('3D Plot of Total X, Y, and Z Trajectories')
    ylabel('Y Trajectory')
    xlabel('X Trajectory')
    zlabel('Y Trajectory')
end
```

Appendix B
(Simulation Data File)

Simulation data file to be included separately in final report.