Cameron Veit

Co-Authored: Connor Veit

11/26/2021

Dr. Zvi Roth

Robotic Applications

EEL 5661 HW 3

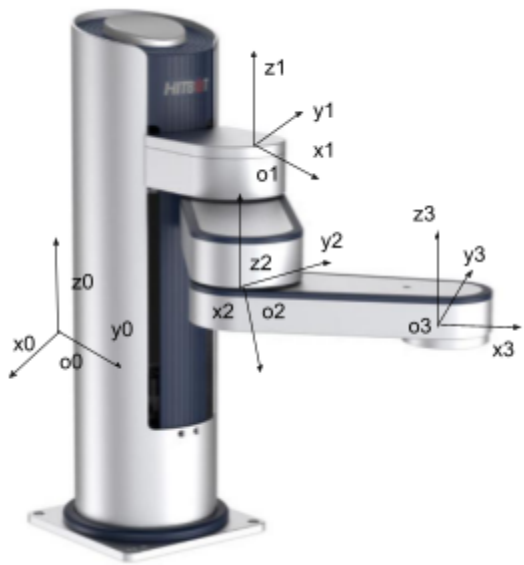
1.1) Joint Types - Joint 1: Prismatic, Joint 2: Revolute, Joint 3: Revolute, Joint 4: Revolute. The combination of Prismatic, Revolute, Revolute, Revolute, means that the 4-axis ZARM 1632 robot is a 1P3R robotic arm. The joint order starts from the base and goes to the end of the arm.

1.2) The joint axes of motion can be seen below, labeled joint 1, 2, 3, and 4. From these axes, the a_n and d_n can be found. d_1 is variable d1 because joint 1 is a prismatic joint. a_1 is an unknown value which will be denoted as a1 for now because joint 1 and joint 2 do not intersect. d_2 is 0 because joint 1 and joint 2 are parallel. a_2 is again an unknown value which we will denote as a2 because joint 2 and joint 3 do not intersect. Now, with joint 3, d_3 is 0 because joint 2 and 3 are parallel, while a_3 is an unknown value denoted as a3 because joint 3 and joint 4 do not intersect. Lastly, with joint 4, d_4 is 0 because joint 3 and joint 4 are parallel, while a_4 is 0 because joint 4 is the end of the arm, and as such there is no link extending off joint 4 to have an a_4. It should be noted that all the non zero a_n values should have a length of 160mm based on the specs of the model from the manufacturer website: https://www.automationar.com/product/z-arm-1632-4-axis-collaborative-robot-with-a-better-price-than-traditional-scara.html

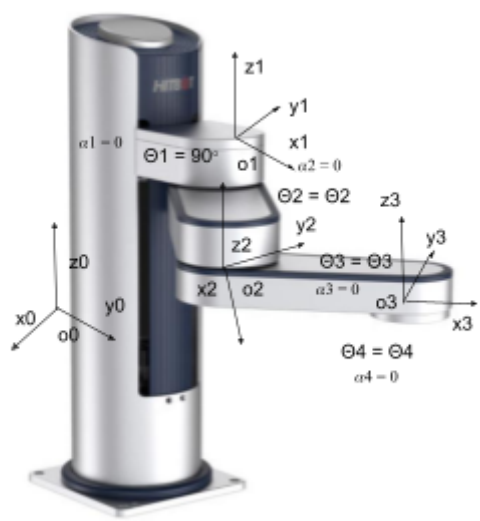
The ZARM 1632 Image With The Joints Of Motion Labeled Can Be Seen Below The Specs.

Model			Z-Arm 1632C collaborative
Basic infomation	J1-axis	Arm reach	160mm
		Rotation angle	+/-90°
	J2-axis	Arm reach	160mm
		Rotation angle	+/-143°
	Z-axis	Stroke	160mm
	R-axis	Rotation angle	+/-1080°
Linear velocity			1017mm(0.5kg payload)
Repeatiblity			+/-0.02mm
Rated payload			0.5kg
Max. Payload			1kg
Degree of freedom			4
Power			220V/110V 50~60Hz
			Adapter to 24VDC
Communication			Wifi / Ethernet
Extensibility			To build-in motion controller,provide 6 I/O
I/O interface	Digtial input		3 channel
	Digtial output		3 channel
Height			480 mm
Weight			19kg
Base installation	Overall size		200mm*200mm*8mm
	Mounting hole		160mm*160mm
			with 4*M5*12 screw
Safety-related monitored stop			Avaiable
Guide programming			Avaiable

1.3) The Basic Origin 0 or o_0 is at the base, at this point the coordinate frame is normally shaped similar to an unaltered world frame and z_0 is pointing upwards because z_0 corresponds to joint 1 which is prismatic and is vertical. At o_1, z_1 points upwards because it is a rotation about the basic z axis via a revolute joint. Furthermore, x_1 points in the direction that a_1, the link between joints 1 and 2 is facing because that is the same direction as z_0 to z_1 and because joint 1 and 2 are parallel. At o_2, z_2 again points upwards because it rotates in the same orientation as the world z axis, that is it is vertical. Similarly, x_2 points in the same direction as a_2, because the link points in the same direction as z_1 to z_2 and because joint 2 and 3 are parallel. At the final local frame of o_3, z_3 once again points upwards because the revolute joint rotates in the same orientation as the world z axis, that is vertical. Similarly, because joint 3 and 4 are parallel and z_2 and z_3 are parallel, the x_3 points in the same direction as a_3, the link between joint 3 and 4 that also points in the same direction as z_2 to z_3. The image with its local coordinate frames can be seen below.



1.4) Because Joint 1 is prismatic it has a constant angle θ_1. Furthermore, We Can See That By Rotating x_0 By 90° around $z_0 = z_1$, we align x_0 and x_1, therefore giving Us $\theta_1 = 90°$, it should also be noted that frame 0 and frame 1 will have different origin points. Additionally, Given That Joints 2, 3, and 4 are revolute, they all have variable θ, Meaning That θ_2, θ_3, and θ_4 are all variable. Next, we find the twist angles α. For starters, because frame 0 and frame 1 have the same orientation with a difference in origin, $\alpha_1 = 0$. In a similar fashion, since all the Z axes are pointed in the same direction upwards, they are parallel and α_2, α_3, and $\alpha_4 = 0$.



1.5)

$$\begin{pmatrix} \cos(90) & -\sin(90) & 0 & 0 \\ \sin(90) & \cos(90) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & a1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(0) & -\sin(0) & 0 \\ 0 & \sin(0) & \cos(0) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$A1 = \begin{pmatrix} \cos(90) & -\sin(90)\cos(0) & \sin(90)\sin(0) & a1\cos(90) \\ \sin(90) & \cos(90)\cos(0) & -\cos(90)\sin(0) & a1\sin(90) \\ 0 & \sin(0) & \cos(0) & d1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\boxed{A1 = \begin{pmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & a1 \\ 0 & 0 & 1 & d1 \\ 0 & 0 & 0 & 1 \end{pmatrix}}$$

$$\begin{pmatrix} \cos(\theta_2) & -\sin(\theta_2) & 0 & 0 \\ \sin(\theta_2) & \cos(\theta_2) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & a2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(0) & -\sin(0) & 0 \\ 0 & \sin(0) & \cos(0) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$A2 = \begin{pmatrix} \cos(\theta_2) & -\sin(\theta_2)\cos(0) & \sin(\theta_2)\sin(0) & a2\cos(\theta_2) \\ \sin(\theta_2) & \cos(\theta_2)\cos(0) & -\cos(\theta_2)\sin(0) & a2\sin(\theta_2) \\ 0 & \sin(0) & \cos(0) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\boxed{A2 = \begin{pmatrix} \cos(\theta_2) & -\sin(\theta_2) & 0 & a2\cos(\theta_2) \\ \sin(\theta_2) & \cos(\theta_2) & 0 & a2\sin(\theta_2) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}}$$

$$\begin{pmatrix} \cos(\theta_3) & -\sin(\theta_3) & 0 & 0 \\ \sin(\theta_3) & \cos(\theta_3) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & a3 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(0) & -\sin(0) & 0 \\ 0 & \sin(0) & \cos(0) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$A3 = \begin{pmatrix} \cos(\theta_3) & -\sin(\theta_3)\cos(0) & \sin(\theta_3)\sin(0) & a3\cos(\theta_3) \\ \sin(\theta_3) & \cos(\theta_3)\cos(0) & -\cos(\theta_3)\sin(0) & a3\sin(\theta_3) \\ 0 & \sin(0) & \cos(0) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\boxed{A3 = \begin{pmatrix} \cos(\theta_3) & -\sin(\theta_3) & 0 & a3\cos(\theta_3) \\ \sin(\theta_3) & \cos(\theta_3) & 0 & a3\sin(\theta_3) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}}$$

$$\begin{pmatrix} \cos(\theta_4) & -\sin(\theta_4) & 0 & 0 \\ \sin(\theta_4) & \cos(\theta_4) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(0) & -\sin(0) & 0 \\ 0 & \sin(0) & \cos(0) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$A4 = \begin{pmatrix} \cos(\theta_4) & -\sin(\theta_4)\cos(0) & \sin(\theta_4)\sin(0) & 0\cos(\theta_4) \\ \sin(\theta_4) & \cos(\theta_4)\cos(0) & -\cos(\theta_4)\sin(0) & 0\sin(\theta_4) \\ 0 & \sin(0) & \cos(0) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\boxed{A4 = \begin{pmatrix} \cos(\theta_4) & -\sin(\theta_4) & 0 & 0 \\ \sin(\theta_4) & \cos(\theta_4) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}}$$

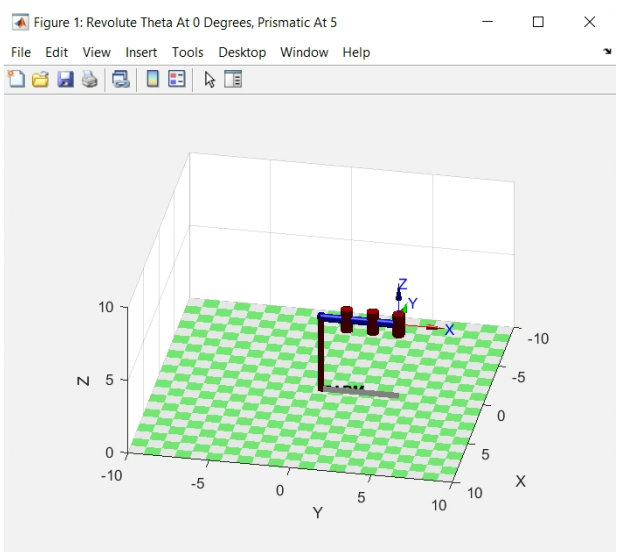
$$Tn = A1(q1)A2(q2)A3(q3)A4(q4)$$

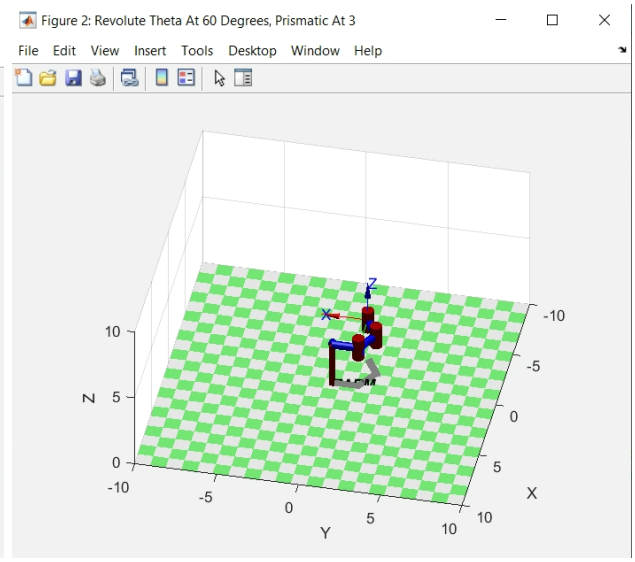
$$Tn = \begin{pmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & a1 \\ 0 & 0 & 1 & q1 \\ 0 & 0 & 0 & 1 \end{pmatrix} * \begin{pmatrix} \cos(q2) & -\sin(q2) & 0 & a2\cos(q2) \\ \sin(q2) & \cos(q2) & 0 & a2\sin(q2) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} *$$

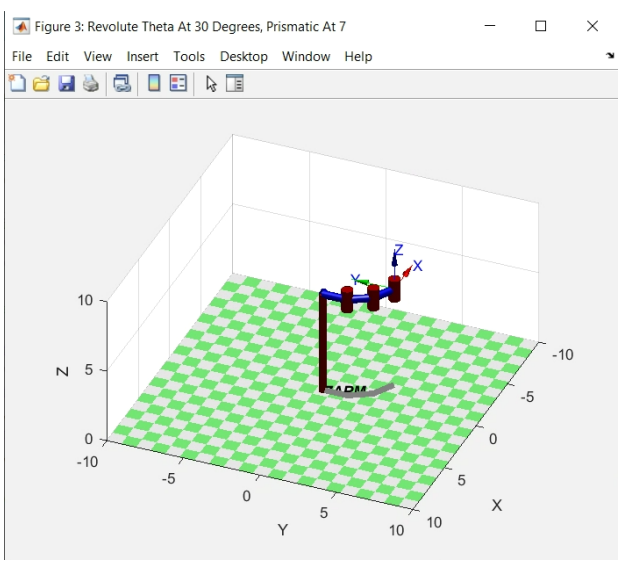
$$* \begin{pmatrix} \cos(q3) & -\sin(q3) & 0 & a3\cos(q3) \\ \sin(q3) & \cos(q3) & 0 & a3\sin(q3) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} * \begin{pmatrix} \cos(q4) & -\sin(q4) & 0 & 0 \\ \sin(q4) & \cos(q4) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

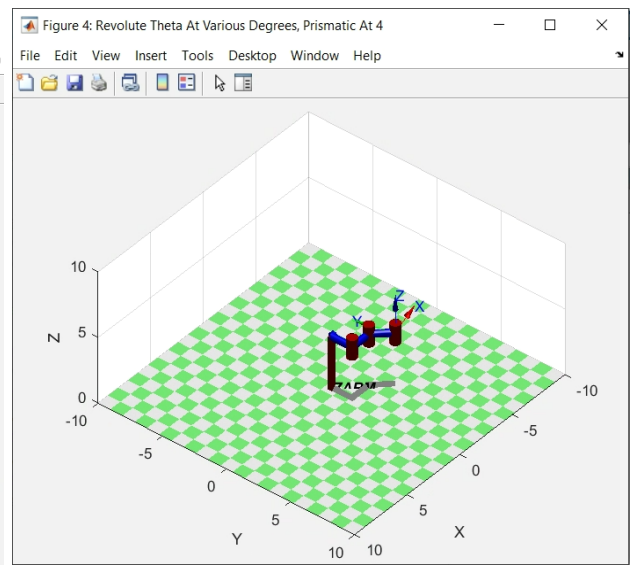
$$Tn = \begin{pmatrix} -\cos(q4)*(\cos(q2)*\sin(q3) + \cos(q3)*\sin(q2)) - \sin(q4)*(\cos(q2)*\cos(q3) - \sin(q2)*\sin(q3)), & \sin(q4)*(\cos(q2)*\sin(q3) + \cos(q3)*\sin(q2)) - \cos(q4)*(\cos(q2)*\cos(q3) - \sin(q2)*\sin(q3)), & 0, & -a2*\sin(q2) - a3*\cos(q2)*\sin(q3) - a3*\cos(q3)*\sin(q2) \\ \cos(q4)*(\cos(q2)*\cos(q3) - \sin(q2)*\sin(q3)) - \sin(q4)*(\cos(q2)*\sin(q3) + \cos(q3)*\sin(q2)), & -\cos(q4)*(\cos(q2)*\sin(q3) + \cos(q3)*\sin(q2)) - \sin(q4)*(\cos(q2)*\cos(q3) - \sin(q2)*\sin(q3)), & 0, & a1 + a2*\cos(q2) + a3*\cos(q2)*\cos(q3) - a3*\sin(q2)*\sin(q3) \\ 0, & 0, & 1, & q1 \\ 0, & 0, & 0, & 1 \end{pmatrix}$$

1.6) It should be noted that only one configuration can be viewed at a time because the SerialLink.plot() function moves the robot to the set configuration no matter which figure is viewed. I completed 1.6 by simply entering the discovered values into a SerialLink function and then creating four separate configurations using SerialLink.plot() that can be seen below. I commented out and in each configuration one at a time to properly capture them all.







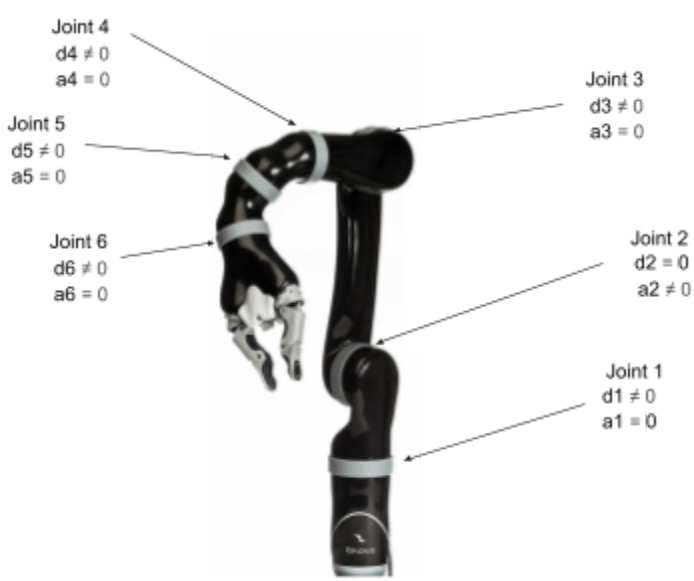


1.6 Matlab Code:

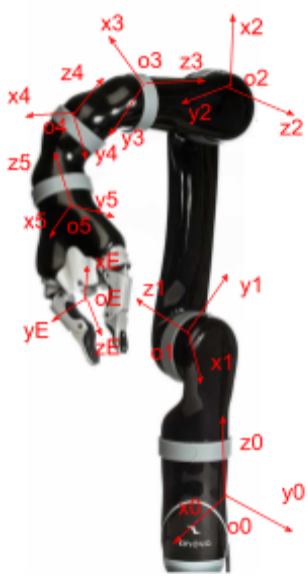
HW31.m

```
robot=SerialLink([Prismatic('a', 1.6, 'theta', pi/2, 'alpha', 0)
    Revolute('a', 1.6, 'd', 0, 'alpha', 0)
    Revolute('a', 1.6, 'd', 0, 'alpha', 0)
    Revolute('a', 0, 'd', 0, 'alpha', 0)], 'name', 'ZARM'); %Create The ZARM 1632 Robotic Arm
%Only One Configuration Can Be Viewed At A Time, Otherwise The Last
%Configuration Overwrites The Prior Configurations
figure('Name', 'Revolute Theta At 0 Degrees, Prismatic At 5') %Straight Robotic Arm Plot
robot.plot([5, 0, 0, 0], 'deg', 'workspace', [-10 10 -10 10 0 10]);
figure('Name', 'Revolute Theta At 60 Degrees, Prismatic At 3') %60 Degree Robotic Arm Plot
robot.plot([3, pi/3, pi/3, pi/3], 'deg', 'workspace', [-10 10 -10 10 0 10]); %pi/3 = 60 degrees
figure('Name', 'Revolute Theta At 30 Degrees, Prismatic At 7') %30 Degree Robotic Arm Plot
robot.plot([7, pi/6, pi/6, pi/6], 'deg', 'workspace', [-10 10 -10 10 0 10]); %pi/6 = 30 degrees
figure('Name', 'Revolute Theta At Various Degrees, Prismatic At 5') %Various Degree Robotic Arm Plot
robot.plot([5, pi/2, -pi/4, pi/3], 'deg', 'workspace', [-10 10 -10 10 0 10]);
```

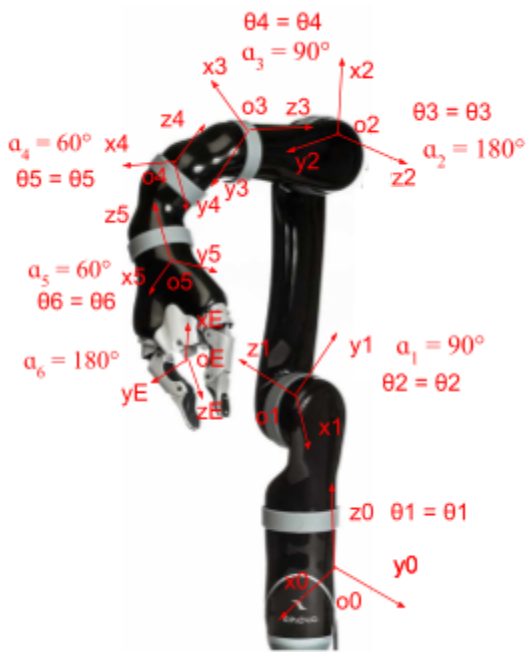
2.1) For joint 1, because the link is not perpendicular to joint 1 and 2, the common normal and $a_1 = 0$, on the other hand, d_1 has an unidentified value we will call d1 because there is a difference between the current joint and the base. Next we have joint 2, the link of joint 2 is in a perpendicular direction to joints 2 and 3, as such the link has a valid common normal and a_2 is an unidentified value called a2, on the other hand there is no distance between link 1 and 2 along the axis of joint 2, and that $d_2 = 0$. For joint 3, there is no perpendicular link between joint 3 and 4, so $a_3 = 0$, and the joints 2 and 3 are not perfectly parallel with a slight change in distance along the axis of joint 3, so d_3 has an unidentified value we will call d3. For joint 4, because the link is not perpendicular to both joints, there is no common normal so $a_4 = 0$, and because joint 3 and 4 are not parallel and there is a change in distance along the axis of joint 4, d_4 is an unidentified value called d4. For joint 5, because there is no perpendicular link, $a_5 = 0$, and since joint 4 and 5 are not parallel and the distance along joint 5 away from joint 4 exists, d_5 has an unknown value called d5. Lastly, for joint 6, because there is no perpendicular link between the two joints $a_6 = 0$, and because there is a distance between joint 6 and joint 5 along the axis of joint 6, d_6 exists with an unknown value called d6.



For every single joint, the z axis is on the joint axis of the revolute joint. For frame 0, the orientation is the same as the world frame. For frame 1, z1 points in the general direction of the world frame's -y direction, the direction of joint 2, and x1 points between the world frame's x and y direction, because they are technically intersecting, so the right hand rule is followed. For frame 2, z2 points in the opposite direction of z1, the world frame's general y direction, because z1 and z2 are almost parallel but in opposite directions, x2 follows the a2 link, that is in the world frame's general z direction. For frame 3, the z follows the same direction as the joint's axis of rotation, which is between the world frame's -x and y directions, because z2 and z3 are not parallel, they are technically intersecting, causing the right hand rule to take place and putting x3 in a similar direction to the world frame's -y direction. For frame 4, z4 follows the joint, giving a similar direction to the world frame -x direction, and because z3 and z4 are technically intersecting because they are not parallel, x4 follows right hand rule and points in a similar direction to the world frame's -y direction. Lastly, for frame 5, z5 follows the joint's direction, meaning it points in the general z world frame direction, the x5 direction follows right hand rule because z4 and z5 are technically intersecting, so x5 points in the world frame x direction. It should be noted that these point away from the gripper because the base z direction is upwards instead of downwards like the connection between the gripper and joint 6 meaning that any rotation turns counter clockwise, this also applies to frames 3 and 4 as well. This means that frame end (frame E), which follows the coordinate frame of a basic gripper has opposite directions to frame 5, so zE follows the world frame's -z direction, and xE follows the right hand rule, meaning that it points in the world frame's general -x direction.



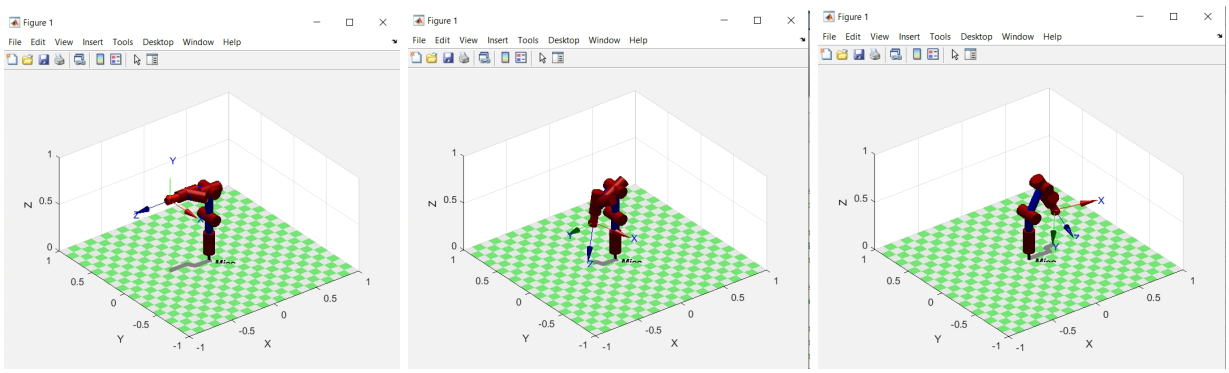
Because every joint is revolute, every θ value is variable. Because of this, we now need to find the α angles for every joint. First we find α_1, if x0 and x1 are pointing in the same direction, then we need to rotate z0 around x1 about 90° or pi/2 in order to get z1, so $\alpha_1 = 90°$. Next, we find α_2, this one is simple to find as the z axes point in opposite directions of one another, so if z1 is rotated about x2, then it needs to be rotate 180° or pi to reach z2, so $\alpha_2 = 180°$. Continuing onwards, we need to find α_3, this can be found by again assuming x2 and x3 are lined up. Then, z2 needs to be rotated about 90° or pi/2 in order to get z3, so that $\alpha_3 = 90°$. Then, α_4, is found by following a similar strategy. Assuming that x3 and x4 are aligned, then by rotating z3 around x4 by 60° or pi/3, we can get z4, so $\alpha_4 = 60°$. Next up, we find α_5, this can be found by aligning x4 and x5 and then rotating z4 by around 60° or pi/3 to get z5, meaning that $\alpha_5 = 60°$. Lastly, we need to find α_6, which is based on the frame of the gripper, or frame end (frame 6), because z6 points in the opposite direction of z5, we can assume that when x6 and x5 are aligned, we need to rotate by 180 degrees in order to flip the direction of the z axis, giving $\alpha_6 = 180°$.



The parameters I found were identical to the parameters mentioned in the model table. This final solution however, took two attempts until I realized that the Z axis starting at frame 3 (joint 4) needed to be pointing back towards the prior joint, joint 3. As such, it was determined that the Z axis for the following joints, joints 5 and 6, needed to point in a similar direction, that is away from the link of the current joint. The model table can be seen below.

```
mico =

Mico [Kinova]:: 6 axis, RRRRRR, stdDH, slowRNE
+---+-----------+-----------+-----------+-----------+-----------+
| j |     theta |         d |         a |     alpha |    offset |
+---+-----------+-----------+-----------+-----------+-----------+
|  1|         q1|     0.2755|          0|     1.5708|          0|
|  2|         q2|          0|       0.29|    3.14159|    -1.5708|
|  3|         q3|     -0.007|          0|     1.5708|     1.5708|
|  4|         q4|  -0.166082|          0|     1.0472|          0|
|  5|         q5| -0.0855633|          0|     1.0472|   -3.14159|
|  6|         q6|  -0.202782|          0|    3.14159|     1.5708|
+---+-----------+-----------+-----------+-----------+-----------+
```

2.2) I was able to successfully run fkine without any issue whatsoever, all it required was a list of 6 different input theta values for each of the 6 revolute joints. On the other hand, it was quite difficult to implement the ikine results. In order to make ikine function properly, it was necessary to both set an initial position of [0, pi, pi/6, 0, 0, 0] as well as disable the joint values that do not change from the initial position to the end position using a mask, despite the fact that the mask is supposed to be used for robotic arms with a DoF below 6. Once both of these conditions were set, the ikine function could properly create an output based on its corresponding fkine function that was used earlier. Additionally, I was unable to get ikine6s working because the mico does not have a spherical wrist. The images of fkine and ikine implementation as well as the plotted configurations for each can be found below.




```
fkine1 =
         0         0        -1   -0.4116
        -1         0         0    0.0671
         0         1         0    0.7655
         0         0         0         1
```

```
fkine2 =
    0.2834   -0.9470   -0.1510   -0.2019
   -0.9573   -0.2888    0.0150   0.06762
   -0.0578    0.1403   -0.9884     0.444
         0         0         0         1
```

```
fkine3 =
    0.6124   -0.6124    0.5000    0.3508
   -0.7071   -0.7071         0    0.0671
    0.3536   -0.3536   -0.8660    0.3702
         0         0         0         1
```

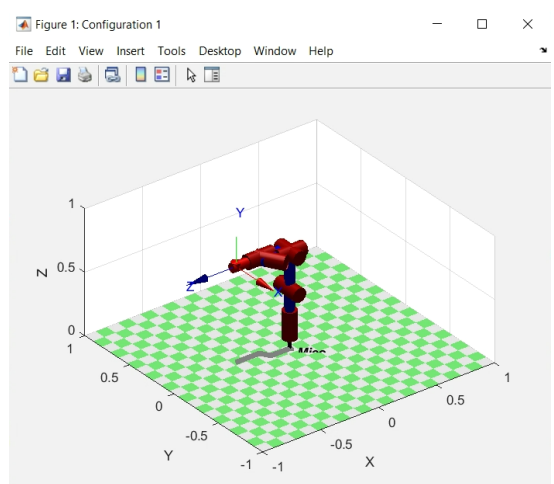
```
ikine1 =

   -0.1181    2.7086    0.5236   -0.0000    0.0000         0


ikine2 =

    0.1909    2.8507    0.7539   -0.2980    1.0442    0.0000


ikine3 =

   -0.0302    2.5454   -0.8024    0.0848    0.5683    0.4032
```

2.2 Matlab Code:

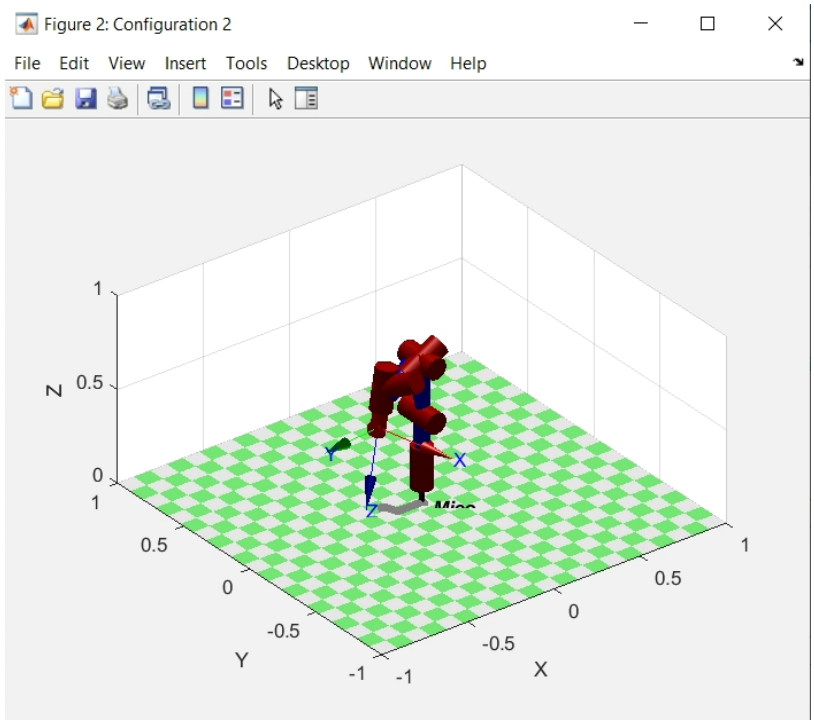
HW32.m

```
mdl_mico
mico
mico.base = SE3(0, 0, 0.2);
%fkine commands take the angles for each joint and display the end position and coordinate frame
mico.plot([0, pi, pi/2, 0, 0, 0], 'deg', 'workspace', [-1 1 -1 1 0 1]) %plots first configuration
fkine1 = mico.fkine([0, pi, pi/2, 0, 0, 0]) %fkine for joint 2 pointing up and joint 3 making a right angle
mico.plot([0, pi, pi/3, -pi/12, pi/3, 0], 'deg', 'workspace', [-1 1 -1 1 0 1]) %plots second configuration
fkine2 = mico.fkine([0, pi, pi/3, -pi/12, pi/3, 0]) %fkine for joint 2 pointed up and joint 3, 4, and 5 slanted down
mico.plot([0, pi/1.2, -pi/3, 0, 0, pi/4], 'deg', 'workspace', [-1 1 -1 1 0 1]) %plots third configuration
fkine3 = mico.fkine([0, pi/1.2, -pi/3, 0, 0, pi/4]) %fkine with joint 2 at 150, joint 3 at -60, and joint 6 at 45
%ikine commands take the angles for each joint and display the end position and coordinate frame, to have it
%properly function a mask must be implemented wherin the values that don't change from q0 have a 0 in the mask
%otherwise the ikine function gets stuck in an infinite loop at a specific error value based on the inputs
ikine1 = mico.ikine(fkine1, 'q0', [0, pi, pi/6, 0, 0, 0], 'mask', [0 0 1 0 0 0]) %ikine for fkine1
ikine2 = mico.ikine(fkine2, 'q0', [0, pi, pi/6, 0, 0, 0], 'mask', [0 0 1 1 1 0]) %ikine for fkine2
ikine3 = mico.ikine(fkine3, 'q0', [0, pi, pi/6, 0, 0, 0], 'mask', [0 1 1 0 0 1]) %ikine for fkine3
%ikine6s does not work for the Kinova Mico. This is because it does not have a spherical wrist and
%rather has a normal gripper.
```

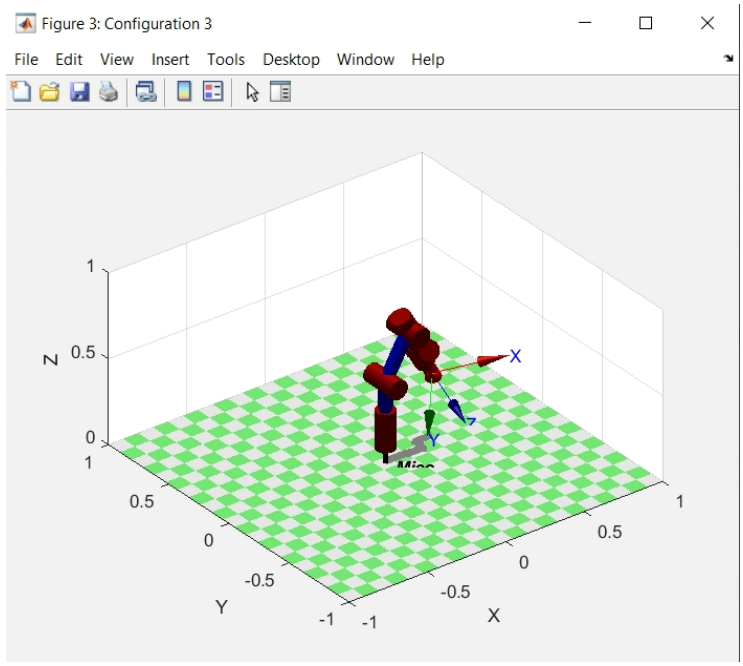
2.3) It should be noted that to get a specific configuration, you must comment out all mico.plot functions after the plot function of the desired configuration. Additionally, 2.3 was simple to complete, only requiring the use of an unmodified jacob0 and rank function as well as a rounding function to keep the size of the numbers within four digits to the right of the decimal point.




```
J1 =

   -0.0671   -0.2900         0         0         0         0
   -0.4116         0         0         0         0         0
         0   -0.4116    0.4116    0.0741   -0.1756         0
         0         0         0    1.0000    0.5000    1.0000
         0   -1.0000    1.0000         0   -0.8660         0
    1.0000         0         0         0         0         0


Rank1 =

     5
```




```
J2 =

   -0.0676    0.0315   -0.3215   -0.0373    0.1663         0
   -0.2019         0         0    0.1774    0.0507         0
         0   -0.2019    0.2019    0.0646   -0.0246         0
         0         0         0    0.8660    0.3209    0.1510
         0   -1.0000    1.0000         0   -0.8365   -0.0150
    1.0000         0         0    0.5000    0.4441    0.9884


Rank2 =

     6
```




```
J3 =

   -0.0671    0.1053   -0.3565   -0.0642    0.1521         0
    0.3508         0         0         0         0         0
         0    0.3508   -0.2058   -0.0371    0.0878         0
         0         0         0   -0.5000   -0.2500   -0.5000
         0   -1.0000    1.0000         0   -0.8660         0
    1.0000         0         0    0.8660    0.4330    0.8660


Rank3 =

     5
```

2.3 Matlab Code:

HW32_2.m

```
mdl_mico
mico.base = SE3(0, 0, 0.2);
figure('name', 'Configuration 1') %Figure for first configuration
mico.plot([0, pi, pi/2, 0, 0, 0], 'deg', 'workspace', [-1 1 -1 1 0 1]); %plots first configuration
J1 = mico.jacob0([0, pi, pi/2, 0, 0, 0]); %Get rank and jacobian matrix of configuration 1
J1 = round(J1, 4) %Round J1 to the four digits
Rank1 = rank(J1)
figure('name', 'Configuration 2') %Figure for second configuration
mico.plot([0, pi, pi/3, -pi/12, pi/3, 0], 'deg', 'workspace', [-1 1 -1 1 0 1]) %plots second configuration
J2 = mico.jacob0([0, pi, pi/3, -pi/12, pi/3, 0]); %Get rank and jacobian matrix of configuration 2
J2 = round(J2, 4) %Round J2 to the four digits
Rank2 = rank(J2)
figure('name', 'Configuration 3') %Figure for third configuration
mico.plot([0, pi/1.2, -pi/3, 0, 0, pi/4], 'deg', 'workspace', [-1 1 -1 1 0 1]) %plots third configuration
J3 = mico.jacob0([0, pi/1.2, -pi/3, 0, 0, pi/4]); %Get rank and jacobian matrix of configuration 3
J3 = round(J3, 4) %Round J3 to the four digits
Rank3 = rank(J3)
```

3.1) For starters, it is necessary to calculate the maximum and optimal velocities as well as Δq.

$$\left|\Delta q_X\right| = \left|q(T_m) - q(0)\right| = |-2 - 4| \Rightarrow \left|\Delta q_X\right| = 6 \;||\; \left|\Delta q_Z\right| = \left|q(T_m) - q(0)\right| = |3.5 - 1| \Rightarrow \left|\Delta q_Z\right| = 2.5$$

$$V_{X,opt} = \sqrt{a_{x,m}\left|\Delta q_X\right|} = \sqrt{2.9 * 6} \Rightarrow V_{X,opt} = 4.1713 \;||\; V_{Z,opt} = \sqrt{a_{z,m}\left|\Delta q_Z\right|} = \sqrt{2.1 * 2.5} \Rightarrow V_{X,opt} = 2.2913$$

Because $V_{X,opt} > V_{X,max} = 0.8$ and $V_{Z,opt} > V_{Z,max} = 0.4$, T_m is based on trapezoidal velocity.

$$T_{m,X} = \frac{V_X}{a_X} + \frac{\left|\Delta q_X\right|}{V_X} = \frac{0.8}{2.9} + \frac{6}{0.8} \Rightarrow T_{m,X} = 7.7759 \;||\; T_{m,Z} = \frac{V_Z}{a_Z} + \frac{\left|\Delta q_Z\right|}{V_Z} = \frac{0.4}{2.1} + \frac{2.5}{0.4} \Rightarrow T_{m,Z} = 6.4405$$

The X joint is slower because it takes longer, making the synchronized motion time T = 7.7759.

Additionally, because the maximum time is used, the velocity profile is trapezoidal for both joints.

$$V_X = \frac{a_{x,max}T_{max} - \sqrt{a_{X,max}{}^2 T_{max}{}^2 - 4a_{X,max}\left|\Delta q_X\right|}}{2} = \frac{2.9*7.7759 - \sqrt{2.9^2 7.7759^2 - 4*2.9*6}}{2} \Rightarrow V_X = 0.8$$

$$V_Z = \frac{a_{z,max}T_{max} - \sqrt{a_{z,max}{}^2 T_{max}{}^2 - 4a_{z,max}\left|\Delta q_z\right|}}{2} = \frac{2.1*7.7759 - \sqrt{2.1^2 7.7759^2 - 4*2.1*2.5}}{2} \Rightarrow V_Z = 0.3281$$

Given these specific velocities over the course of the time motion from (4, 1) to (-2, 3.5). We get:

Velocity Profile Of The X Joint, Including Acceleration And Velocity:

$0 \le t \le 0.277[s]: \; l(t) = \frac{1}{2} \cdot \frac{0.8}{(0.8/2.9)} t^2 = 1.45t^2$ [m] [Acceleration]

$0.8/2.9 \le t \le 7.22\,[s]: l(t) = 0.8(t - (0.8/2.9)) = 0.8t - 0.64/2.9$ [m]

$7.22 \le t \le 7.7759[s]: \; a_t = 0.8 - \frac{0.8}{(0.8/2.9)}(t - 7.22) = -\frac{0.8}{2.9}t + 21.738\,[m/sec]\,[Deceleration]$

$7.22 \le t \le 7.7759[s]: \; l(t) = -\frac{1}{2} \cdot \frac{0.8}{(0.8/2.9)} t^2 + 2.14508t + l(7.22) = -1.45t^2 + 21.738t + 5.555[m]$

Velocity Profile Of The Z Joint, Including Acceleration And Velocity:

$0 \le t \le (0.3281/2.1)\,[s]: \; l(t) = \frac{1}{2} \cdot \frac{0.3281}{(0.3281/2.1)} t^2 = 1.05t^2$ [m] [Acceleration]

$(0.3281/2.1) \le t \le 7.3762\,[s]: l(t) = 0.3281(t - (0.3281/2.1)) = 0.3281\,t - 0.0513$

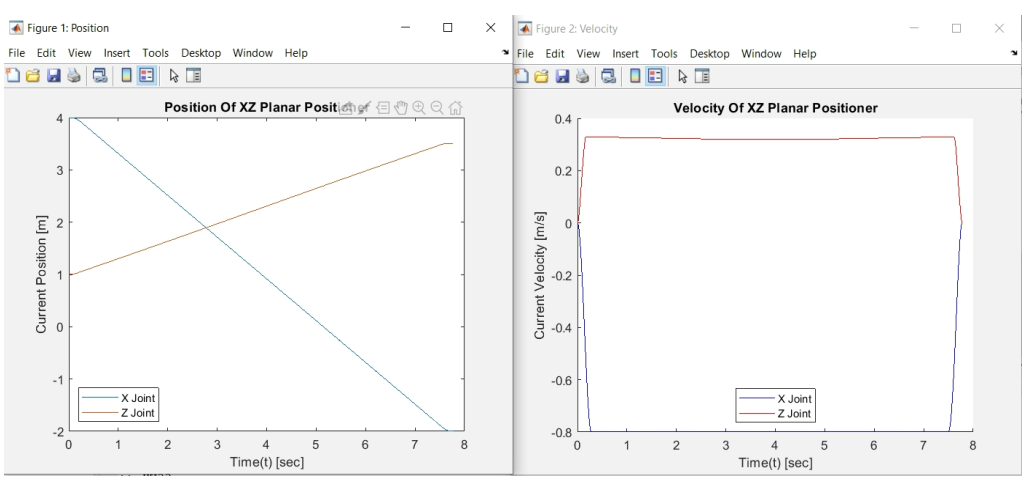
[m]

$7.3762 \le t \le 7.7759[s]: \; a_t = 0.3281 - \frac{0.3281}{(0.3281/2.1)}(t - 7.3762) = -2.1t + 15.8181\,[m/sec]\,[Deceleration]$

$7.3762 \leq t \leq 7.7759[s]: \; l(t) = -\frac{1}{2} \cdot \frac{0.3281}{2.1} t^2 + 1.0899t + l(7.3762) = -1.05t^2 + 15.8181t + 2.3688\,[m]$

It should be noted that the chosen acceleration was always the maximum acceleration for any given joint, and the chosen velocity was calculated using $V_X = \frac{a_{x,max} T_{max} - \sqrt{a_{X,max}{}^2 T_{max}{}^2 - 4a_{X,max}|\Delta q_X|}}{2}$ for each joint, giving the velocities: $V_X = 0.8$ and $V_Z = 0.3281$.

3.2) I was able to successfully create a working trapezoidal velocity profile for the planar positioner. During this process I encountered a few issues, one such issue was the fact that the Z axis accelerated to full speed faster than the X axis, as such an extra jtraj had to be created to take into account the change where acceleration stops for the velocity plot of the Z axis. Additionally, there was an issue with having a proper acceleration with the jtraj, as it overcompensated because it gradually increased the acceleration instead of having a full out maximum acceleration or no acceleration. As such, the jtraj acceleration starts at 0, goes above the stated maximum, and goes back towards 0 whenever the velocity changes. As this is incorrect and I found no way to prevent this, the acceleration profile has been excluded. Much effort was taken to manuever past this acceleration issue and the resulting graphs are as close to accurate as possible, they act as if the acceleration starts at the stated maximum and stops accelerating when required.

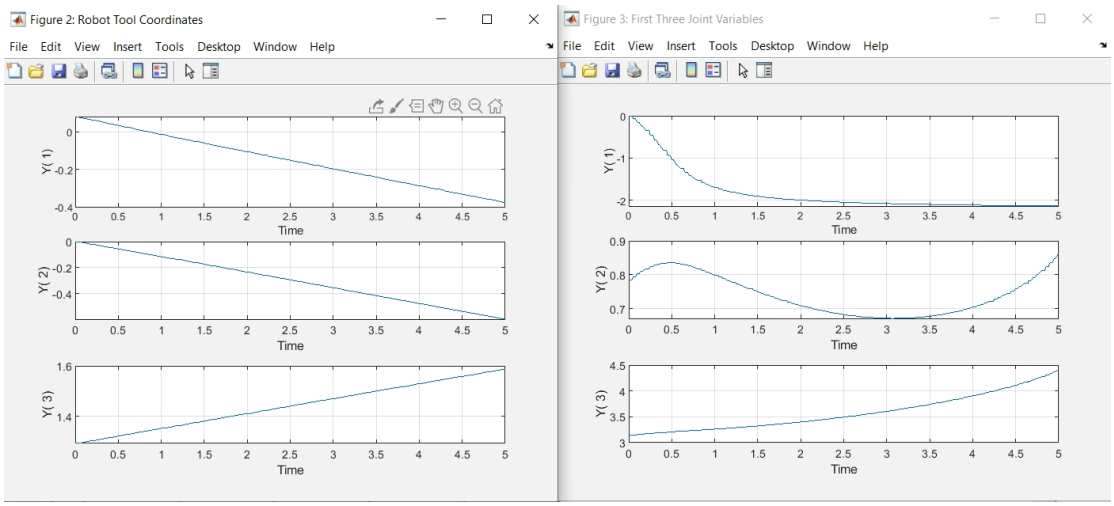


3.2 Matlab Code:

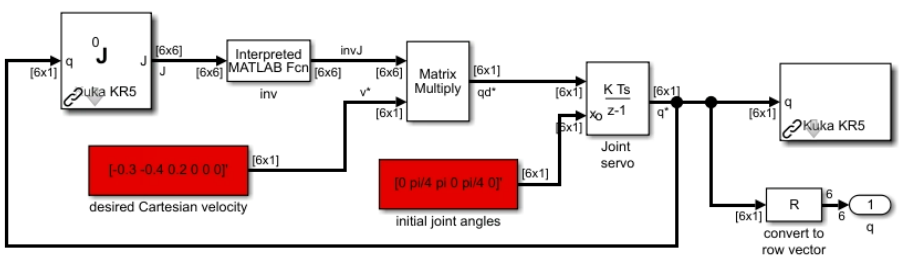
```
HW33.m
Start = [4;1];
Accel = [3.8894;1.064971639985959];
Mid = [-1.8807; 3.4743];
End = [-2;3.5]; %Grab positions for when accelerating starts and stops as well as the start and end position
VStart = [0;0];
VMid = [-0.8;0.3281];
VEnd = [0;0];
TimeAccel = 0:0.0001:0.278;
TimeMid = 0:0.0001:7.22; %Grab time values for when the acceleration is active and inactive.
TimeEnd = 0:0.0001:0.278;
Time = 0:0.0001:7.7762;
[q, qd, qdd] = jtraj(Start, Accel, TimeAccel, VStart, VMid);
[q2, qd2, qdd2] = jtraj(Accel, Mid, TimeMid, VMid, VMid); %Use jtraj to grab outpit values.
[q3, qd3, qdd3] = jtraj(Mid, End, TimeEnd, VMid, VEnd);
Idx = find(round(qd, 4) == 0.3281, 1) - size(qd, 1) - 1;
TimeAccelZ = 0:0.0001:(Idx/10000);
TimeMidZ = 0:0.0001:7.22+((Idx)/7000); %Create jtraj for Z axis which accelerates to maximum faster.
[qZ2, qZd2, qZdd2] = jtraj(Accel, Mid, TimeMidZ, VMid, VMid);
Pos = [q;q2;q3];
XV = [qd;qd2;qd3]; %Put all outputs into one output which will be plotted.
ZV = [qd(1:Idx+1, :);qZd2;qd3(1:Idx+4, :)];
figure('name', 'Position')
plot(Time, Pos); %Plot position of XZ Planar Positioner
title('Position Of XZ Planar Positioner');
xlabel('Time(t) [sec]');
ylabel('Current Position [m]'); %Add Plot Descriptions
legend({'X Joint','Z Joint'},'Location','southwest');
figure('name', 'Velocity')
hold on
plot(Time, XV(:, 1), 'b'); %Plot velocity of XZ Planar Positioner
plot(Time, ZV(:, 2), 'r');
title('Velocity Of XZ Planar Positioner');
xlabel('Time(t) [sec]');
ylabel('Current Velocity [m/s]'); %Add Plot Descriptions
legend({'X Joint','Z Joint'},'Location','south');
```

4.1) For 4.1, I used the Kuka KR5 Robotic Arm. Additionally I changed the desired Caretsian velocity to (-0.3 -0.4, 0.2, 0, 0, 0). However, the desired result for 4.1 was not achieved immediately. First off, I tried using the Kinova Mico Robotic Arm. The Mico arm however had the issue that it was unable to properly reach its goal ever, always glitching out no matter the entered T or K values. After experimenting with a few additional robots, I ended up at the Kuka KR5 Robotic Arm. From there, I started calculating the proper T and K values. I found that the basic T value of 0.05 works with the KR5 and this specific Cartesian Velocity, on the other hand, lower values don't allow the robot to move far and higher values cause the robot to glitch out and make the arm teleport to undesired locations. Additionally, I found the K value to have similar effects when too large or too small, with the optimal K value being 0.3 or below. Once I got the

RRMC Simulink working, I then built the matlab program. This program ended up achieving its desired purpose and displayed the coordinate values of both the robotic arm's tool as well as the first three joint variables, which can all be seen below.



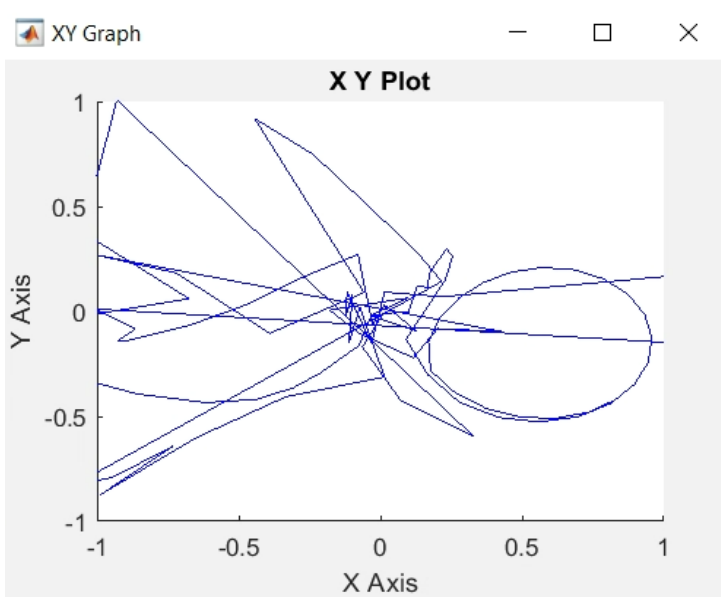
4.1 Simulink:



4.1 Matlab Code:

```
HW34.m
mdl_KR5 %Choses Kuka KR5 Robotic Arm
KR5 %Displays Kuka KR5 Robotic Arm Model Table
HW34rrmc %Grabs RRMC
simulation = sim('HW34rrmc'); %Simulates RRMC
endEffector = simulation.find('tout'); %Gets End Effector
output = simulation.find('yout'); %Gets Output
fkineO = KR5.fkine(output); %Performs Fkine On Output To Get Position Of Gripper
coordinates = transl(fkineO); %Find Position Of Gripper From Basic Position
figure('Name', 'Robot Tool Coordinates') %Plot Coordinates Of Robot's Tool
mplot(endEffector, coordinates(:,1:3));
figure('Name', 'First Three Joint Variables'); %Plot The First 3 Joint Variables
mplot(endEffector, output(:,1:3));
```

4.2) For 4.2, I used the Kuka KR5 Robotic Arm. The process to create a proper circle with this robotic arm required a large amount of trial and error. Initially, both the radius of the circle and the frequency were set to 0.5. From there, the Kp was changed until it seemed the most orderly of the various Kp values, this more orderly value ended up being 0.3. From there the T was tested until a T that formed a circle was found, this T was 0.1. Any deviation from these values will cause a circle to be unable to form at all. Through a vast amount of tests, it was determined that a large amount of the other possible values will fail to result in a circle no matter the combination entered. It was also noted that by increasing the length of T, the length of the line in the X Y plot containing the circle shrunk. Additionally, the X Y plot became more flat the lower the value of Kp was, being almost completely flat when Kp was set to 1. Lastly, the radius simply designates the radius of the circle and the frequency changes how fast the line moves, with higher frequencies meaning the line turns at a much faster pace. The results of the entered variables can be seen in the graphs below.



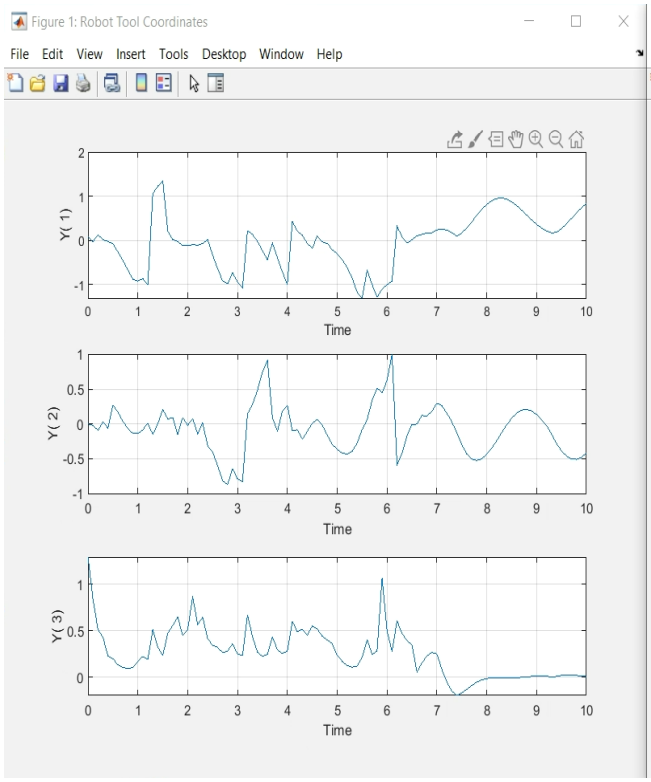


Figure 1: Robot Tool Coordinates
File Edit View Insert Tools Desktop Window Help
Y(1)
Y(2)
Y(3)
Time

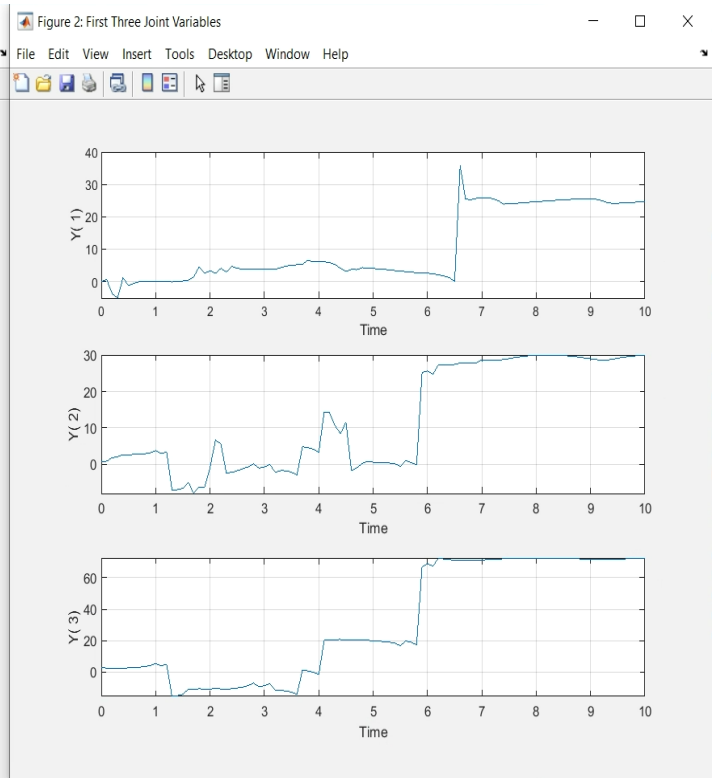


Figure 2: First Three Joint Variables
File Edit View Insert Tools Desktop Window Help
Y(1)
Y(2)
Y(3)
Time

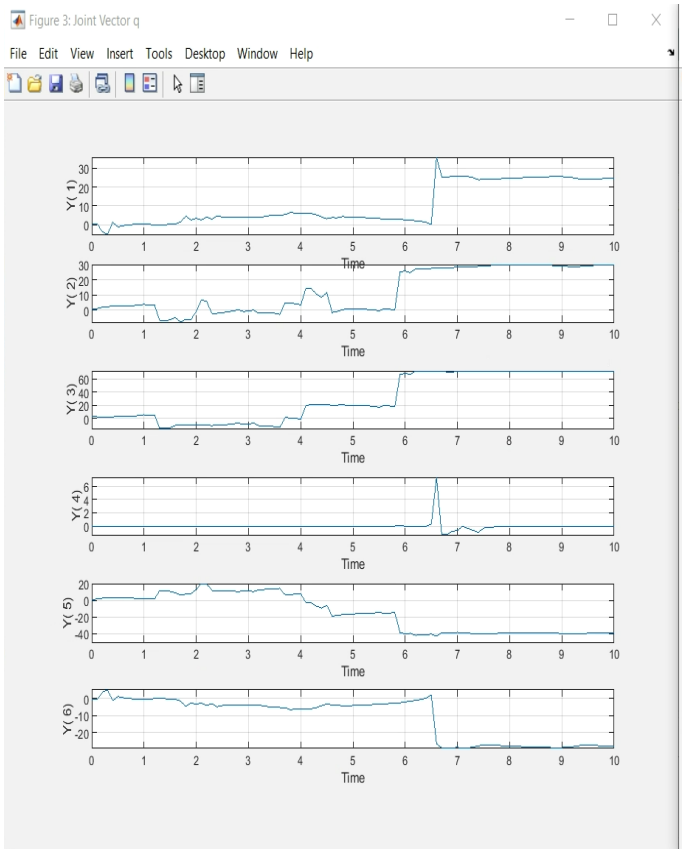


Figure 3: Joint Vector q
File Edit View Insert Tools Desktop Window Help
Y(1)
Y(2)
Y(3)
Y(4)
Y(5)
Y(6)
Time

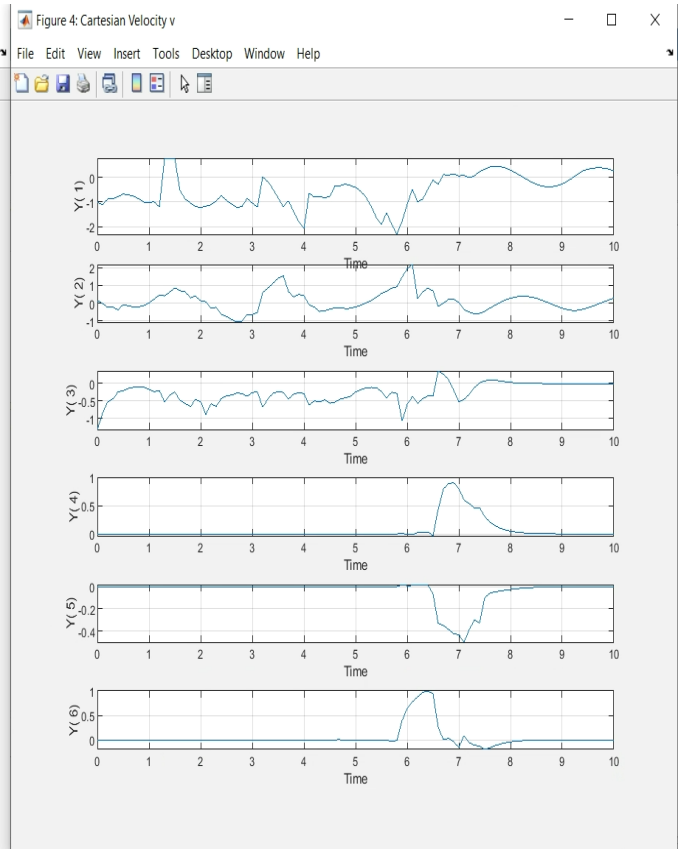
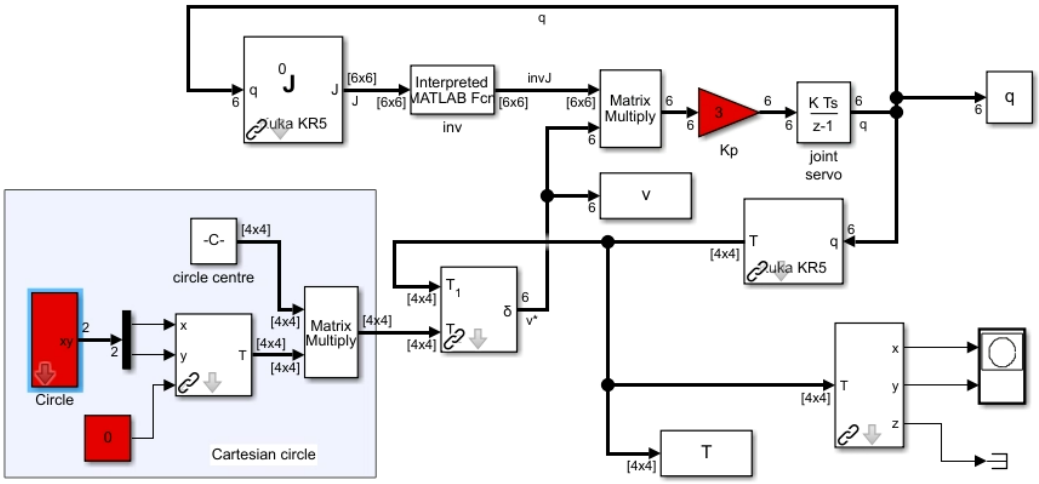


Figure 4: Cartesian Velocity v
File Edit View Insert Tools Desktop Window Help
Y(1)
Y(2)
Y(3)
Y(4)
Y(5)
Y(6)
Time

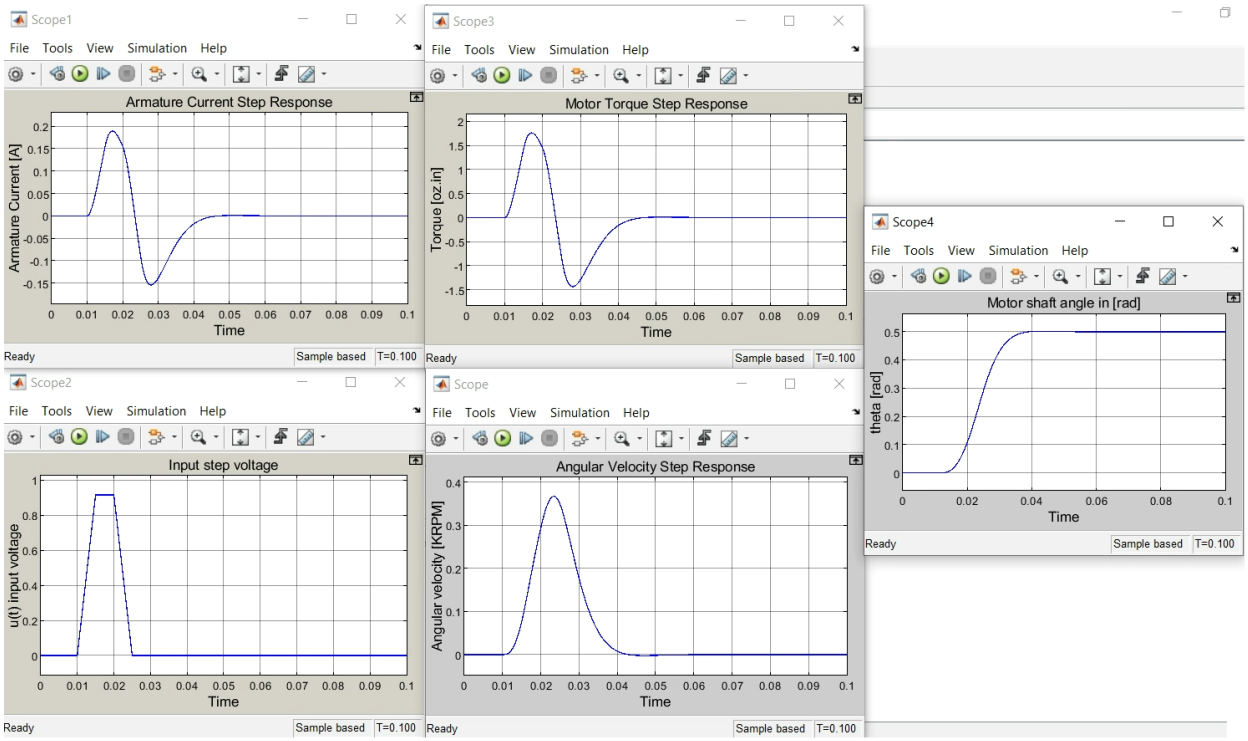
4.1 Simulink:



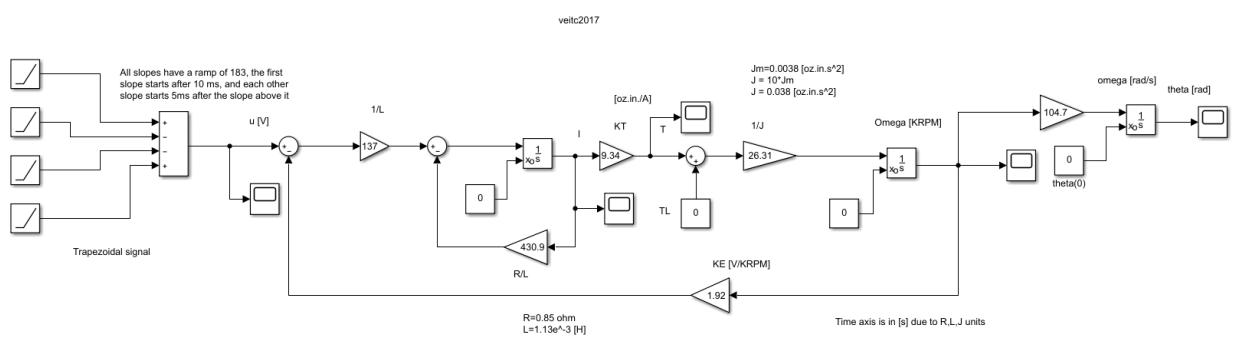
4.1 Matlab Code:

```
HW34_2.m
mdl_KR5 %Choses Kuka KR5 Robotic Arm
KR5 %Displays Kuka KR5 Robotic Arm Model Table
HW34rrmc2 %Grabs RRMC2
simulation = sim('HW34rrmc2'); %Simulates RRMC2
time = [0:0.1:10]; %Grab time value to be used as x for graphs
fkineO = KR5.fkine(q); %Performs Fkine On Output To Get Position Of Gripper
coordinates = transl(fkineO); %Find Position Of Gripper From Basic Position
figure('Name', 'Robot Tool Coordinates') %Plot Coordinates Of Robot's Tool
mplot(time, coordinates(:,1:3));
figure('Name', 'First Three Joint Variables'); %Plot The First 3 Joint Variables
mplot(time, q(:,1:3));
figure('Name', 'Joint Vector q') %Plot Joint Vector q
mplot(time, q);
figure('Name', 'Cartesian Velocity v'); %Plot Cartesian Velocity v
mplot(time, v);
```

5.1) To solve 5.1, I implemented parts of the PID velocity diagram into the DC motor diagram. Specifically, I took the starting ramps and the end section of PID velocity. Once I added these sections, I changed the parameters of the motor to match those of the E-510C motor and changed the theta value near the end to 0 to set the starting angle to 0. I then altered the values of the ramps, setting the slopes of each to 183 and having each activate 5ms after the above ramp, with the top ramp starting at 10ms into the simulation. These alterations allowed me to change the motor's angle from 0 to 0.5 in 40ms, starting at 10ms and ending at 50ms, as can be seen below.

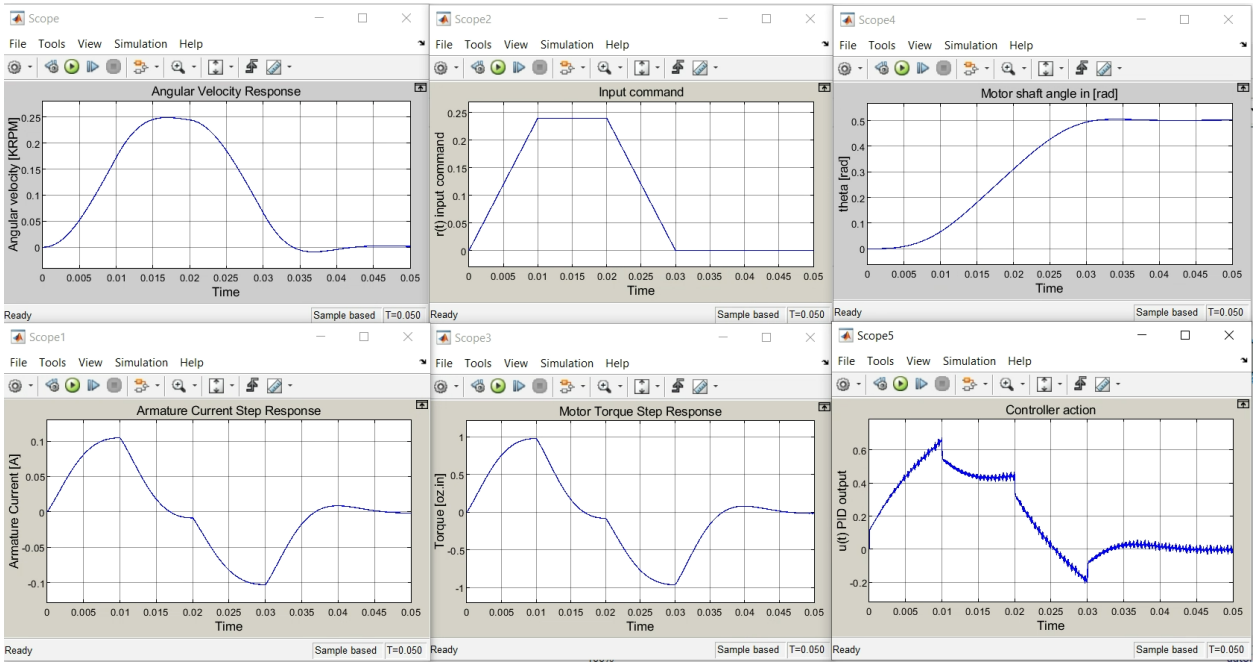


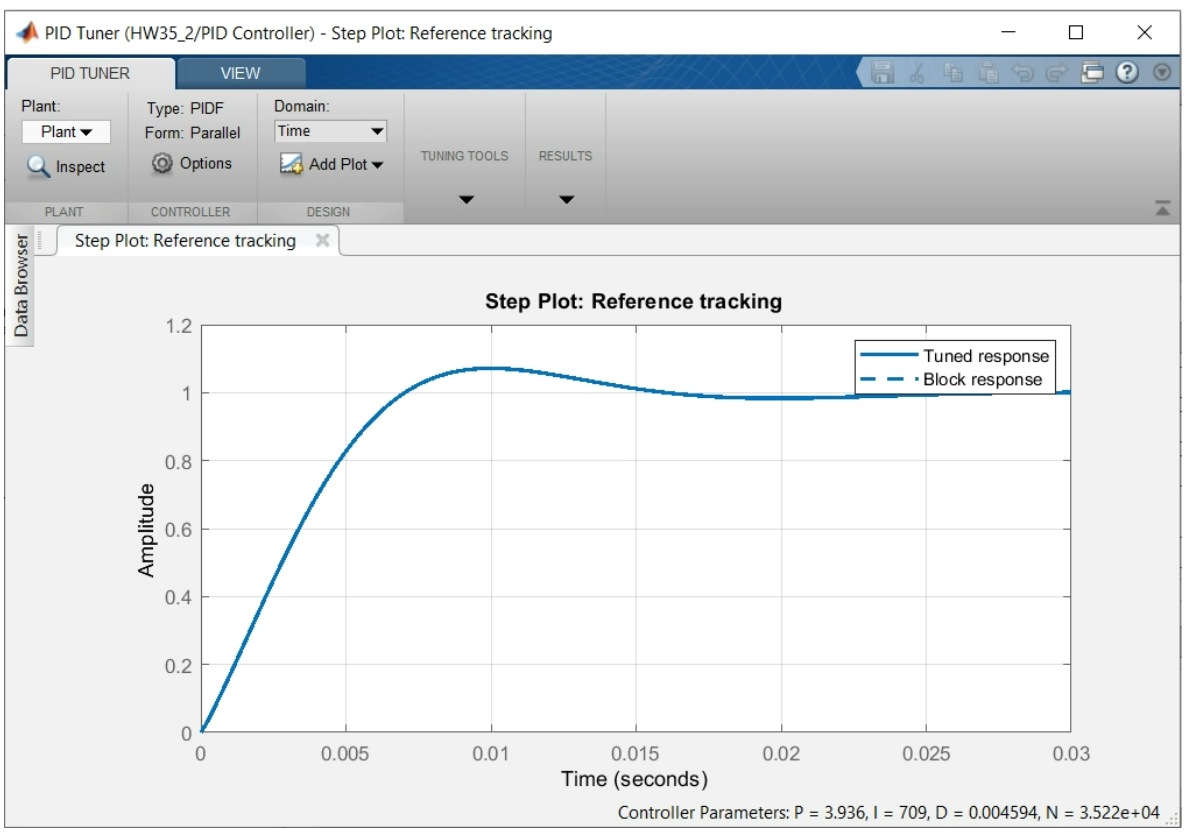
5.1 Simulink Model:



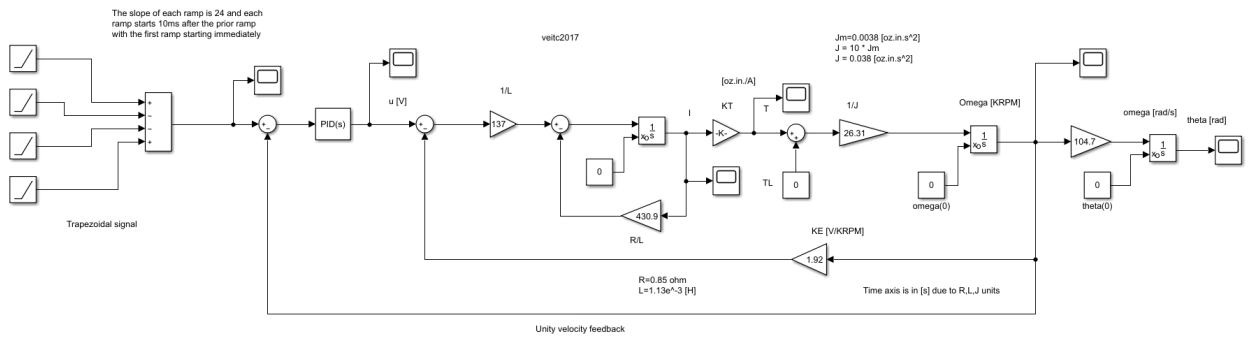
5.2) To solve 5.2 I removed the step input from the basic PID model. I then set the theta value at the end to 0 to match the initial angle of 0. Next, I changed the slope of all the ramps to 24 each and made them all start 10ms after the prior ramp, with the first ramp starting immediately. Once I edited the ramps, I changed the motor parameters in the diagram to the parameters of the E-510C motor. Lastly, I tuned the PID to prevent the output from being changed by the looped

signal. These alterations allowed me to change the motor's angle from 0 to 0.5 rads in 40ms, starting at 10ms and ending at 50ms, as can be seen below alongside the PID tuner.

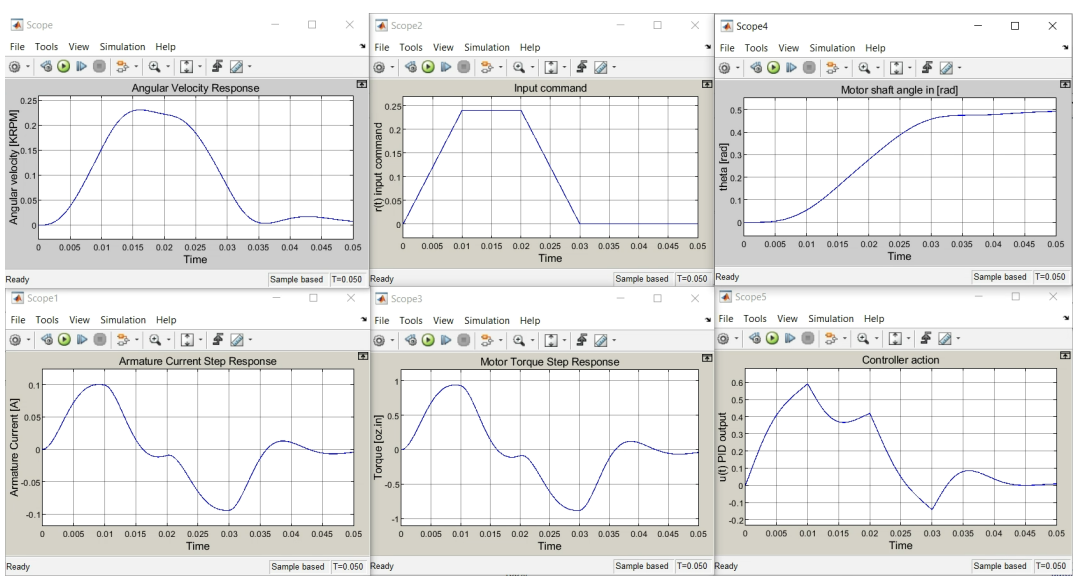


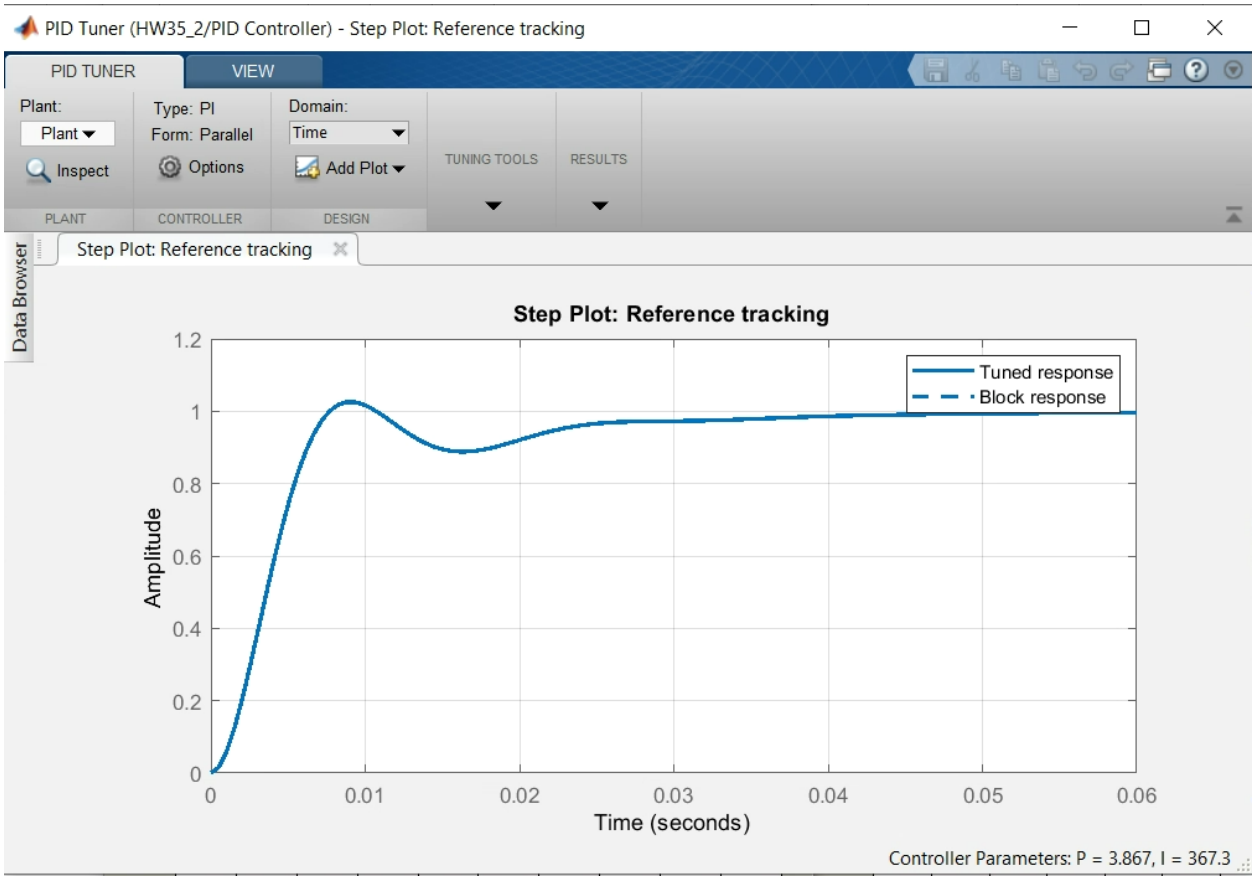
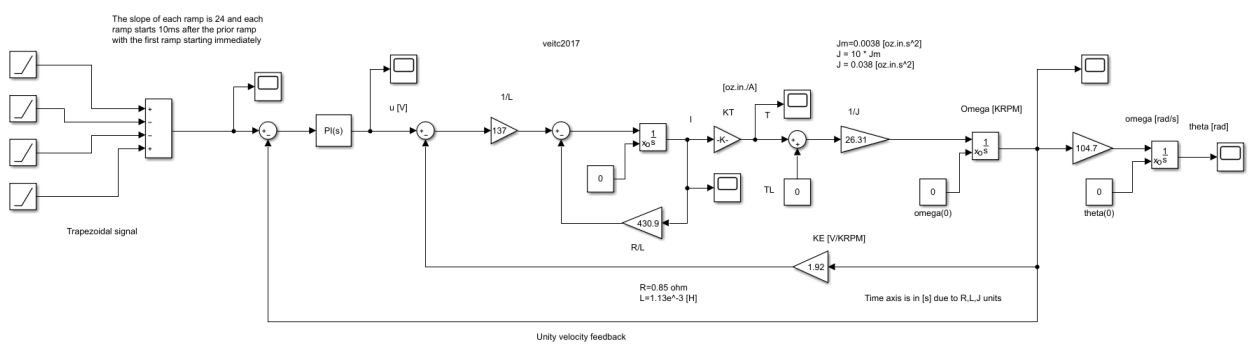


5.2 Simulink Model:



5.3) The D gain in 5.2 was very small, only around #, leading me to try and remove it in 5.3. The consequences of this change directly altered the output of the model, leading to an overall decrease in performance as is evident by the various changes in the values. This is evident in the fact that the model does not reach the goal of 0.5 rads turned in 40ms. This is because changing from PID to PI greatly affects the looped signal, because even though P only lowers by 0.07, the I drops by about half its original value. This lowering in the I value changes the looped signal and prevents the model from reaching the end goal by lowering most scope results by a minor amount, despite stabilizing the controller action. As such, removing the D gain is a negative change, preventing the motor's angle from changing 0.5 rads in 40ms as can be seen below.





5.3 Simulink Model: