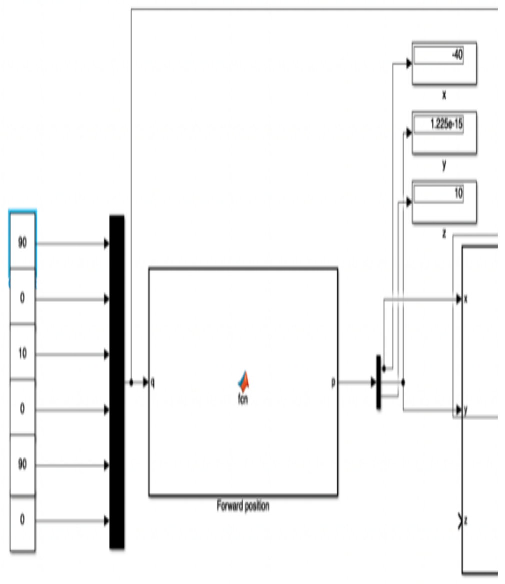
Robotics Project Report

I. Forward Position Kinematics

Results of Simulink model of the robot:

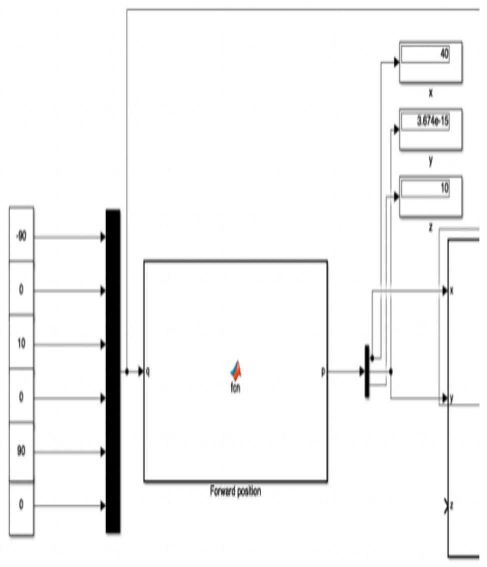
after looping over different angles to know the start position of every link and then we are able to illustrate all the outputs of the functions and visualize the movements of the robot with only inserting its inputs.

Case 1:



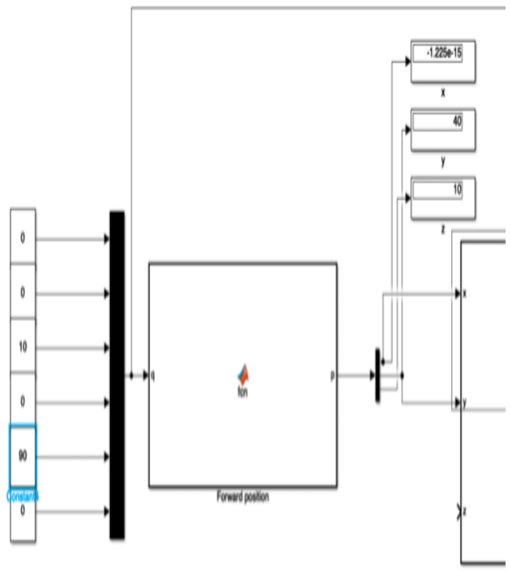
- All links are assumed to be 10 unit.
- The total lengths of the robot is 10+10+10+10 =40 , in addition to a 10 of a positive transnational movement from q3, so the total length will be 40+10=50.
- And each joint is given an input angle to make it stretched for the maximum position at x and it was 40, as it will be as (L) shaped with a height of 10 in positive Z and 40 in x.

Case 2:



The same as case (1), but the angle of the first joint (q1) will be -90 instead of (90), so the only difference is that the maximum position in x will be in positive 40.

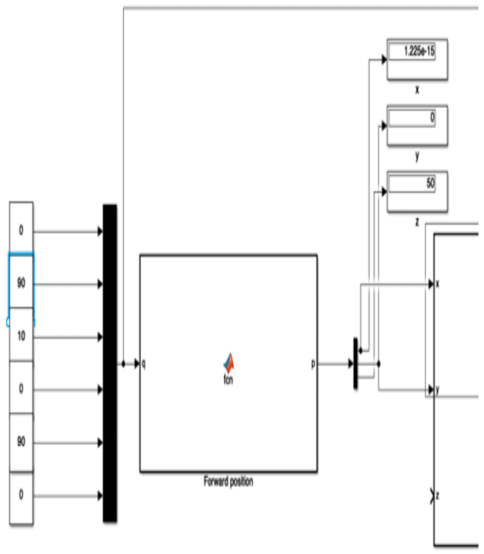
Case 3:



- All links are assumed to be 10 units.
- The total lengths of the robot is 10+10+10+10 =40 , in addition to a 10 of a positive transnational movement from q3, so the total length will be 40+10=50.
- And each joint is given an input angle to make it stretched for the maximum position at y. and it was 40, as it will be as (L) shaped with a height of 10 in Z and 40 in the positive of y.

The value in x is due to calculations in matlab but it converges to zero.

Case 4:

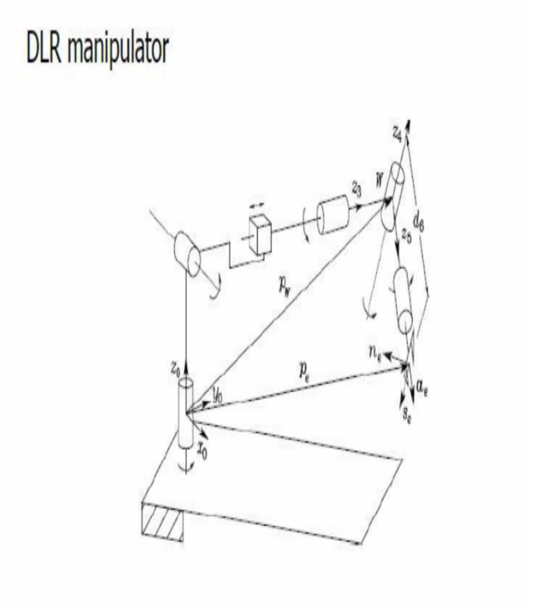


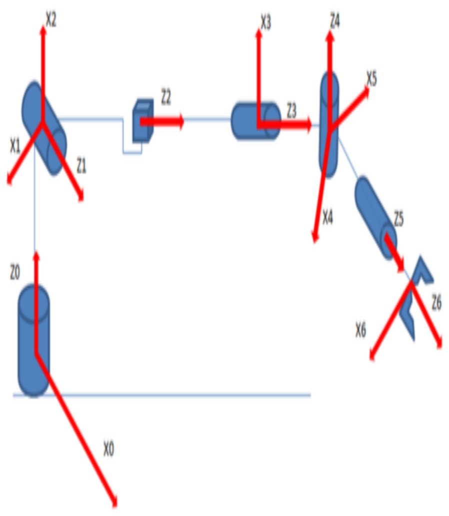
- All links are assumed to be 10 units.
- The total lengths of the robot is 10+10+10+10 =40 , in addition to a 10 of a positive transnational movement from q3, so the total length will be 40+10=50.
- And each joint is given an input angle to make it stretched for the maximum position at z. and it was 50, as it will be as straight shaped (as one line of summations of lengths) with a height of 50 in positive Z.

The value of shown x is due to calculations in matlab but it converges to zero.

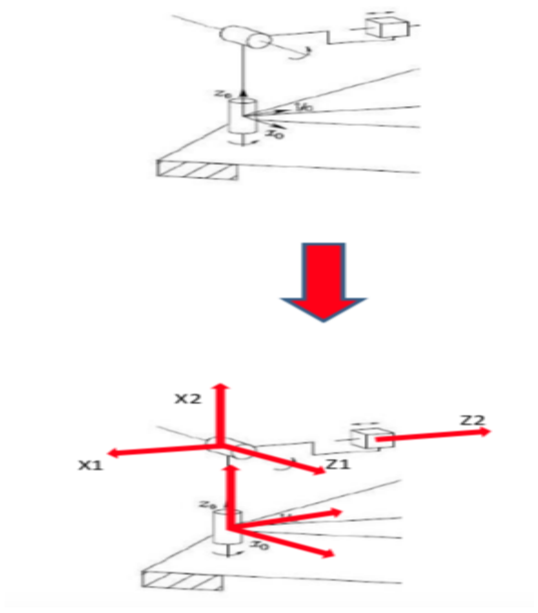
II. Inverse Position Kinematics

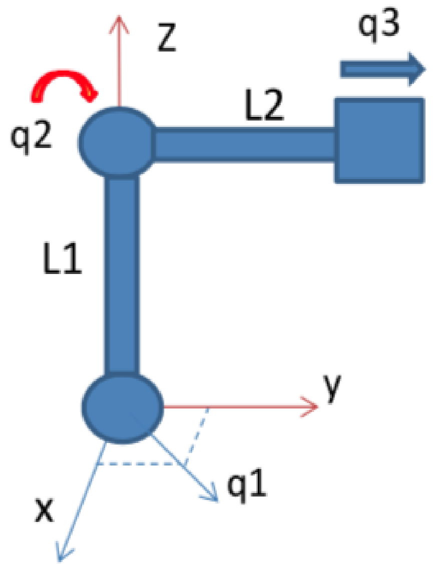
A. Trigonometric Approach





Only for the first 3 DOF (3 joints)





Solution:

$$Tan(q_1) = \frac{X}{Y} \quad (1)$$

$$Tan(q_2) = \frac{Z - L_1}{\sqrt{(X^2 - Y^2)}} \quad (2)$$

As a generalized rule if q1 has a value of

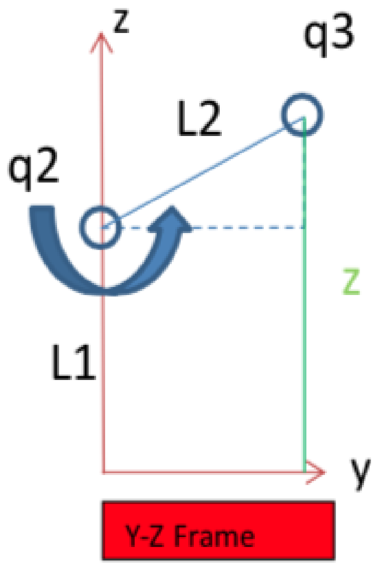
$$\theta \quad (3)$$

then the axis will be

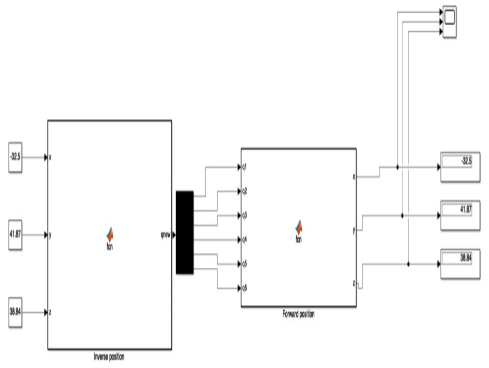
$$W = \sqrt{X^2 - Y^2} \quad (4)$$

not Y only.
At X=0;W=Y.

$$q_3 = \sqrt{X^2 - Y^2 + (Z - L_1)^2} - L_2 \quad (5)$$



B. *Newton Raphson Approach*

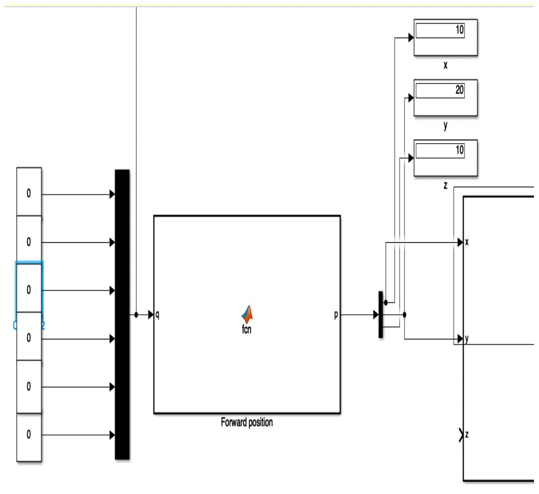


we entered the inputs of X , Y and Z into the the inverse Kinematics block then into the Forward Kinematics block , then the result were the same as the inputs so that's mean the inverse Kinematics block is working correctly.

III. Velocity and Acceleration

As we mentioned later in the position, our zero position is as follows(**after modifying the offset in the total homogeneous transformation matrix)** :
The zero position(which all the q's are zeros):
Assuming all lengths = 10.
When all the q's are zeros:

$$\begin{aligned} Z &= L_1 = 10; \\ X &= L_4 = 10; \\ Y &= L_2 + q_3 + L_3 = 20; \end{aligned} \quad (6)$$

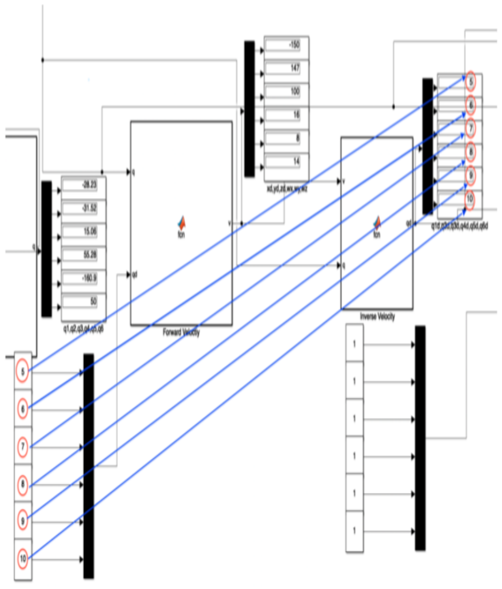


The results are as shown that the maximum positions are right! From Milestone 2 we made sure that the forward and inverse position are working well. Starting by the forward velocity analysis: By finding Jv and Jw of each joint, we have conducted the Jacobian matrix of the velocity as implemented in the matlab code:

```
Jv1= cross([0;0;1], A0T6(:,4)-[0;0;0]);
Jv2= cross(A0T1(:,3),(A0T6(:,4)-A0T1(:,4)));
Jv3= A0T3(:,3);
Jv4= cross(A0T3(:,3),(A0T6(:,4)-A0T3(:,4)));
Jv5= cross(A0T4(:,3),(A0T6(:,4)-A0T4(:,4)));
Jv6= cross(A0T5(:,3),(A0T6(:,4)-A0T5(:,4)));
Jw1=[0;0;1];
Jw2= A0T1(:,3);
Jw3=[0;0;0];
Jw4= A0T3(:,3);
Jw5= A0T4(:,3);
Jw6= A0T5(:,3);
v= [Jv1 Jv2 Jv3 Jv4 Jv5 Jv6; Jw1 Jw2 Jw3 Jw4 Jw5 Jw6]*[q1d;q2d;q3d;q4d;q5d;q6d];
```

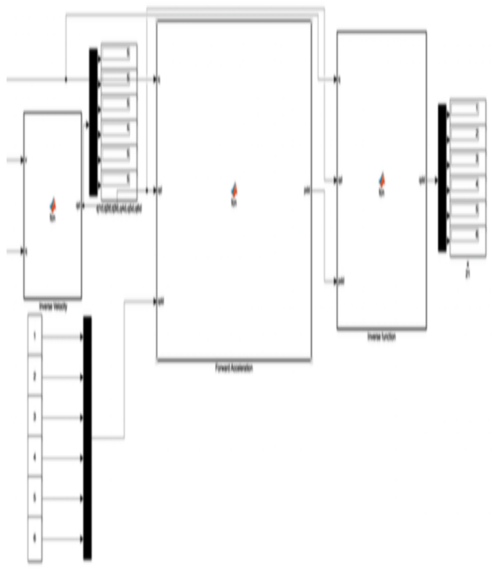
where cross is the cross product, and T is the transformation matrices.

A. *Simulink model for Forward Velocity, Inverse Velocity*



It's obvious that the input to the forward velocity is the same as the output of the Inverse, which indicates that it is correct! The input is q1d,q2d,q3d,q4d,q5d,q6d respectively.

B. *Simulink model for Forward Acceleration, Inverse Acceleration*

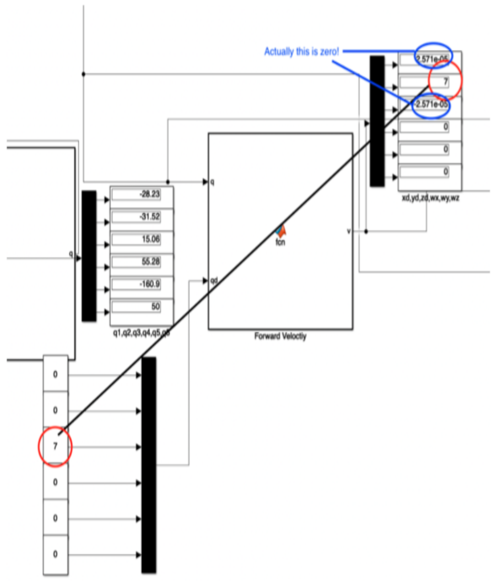


It's obvious that the input to the forward acceleration is the same as the output of the Inverse, which indicates that it is correct! The input is q1dd,q2dd,q3dd,q4dd,q5dd,q6dd respectively.

IV. Testing different input joint velocities and accelerations:

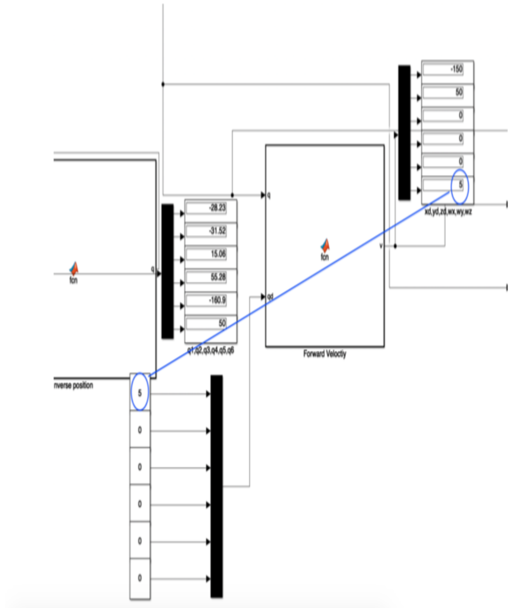
A. *Velocity*

In our robot, q3 is the only prismatic joint. So, If we make an input velocity to q3 with a certain value and assume that the other velocities q's are equal to zero, with fixing our robot in its initial position, it should result **Only** linear velocity in y direction.



This result of the simulation is as we expected!

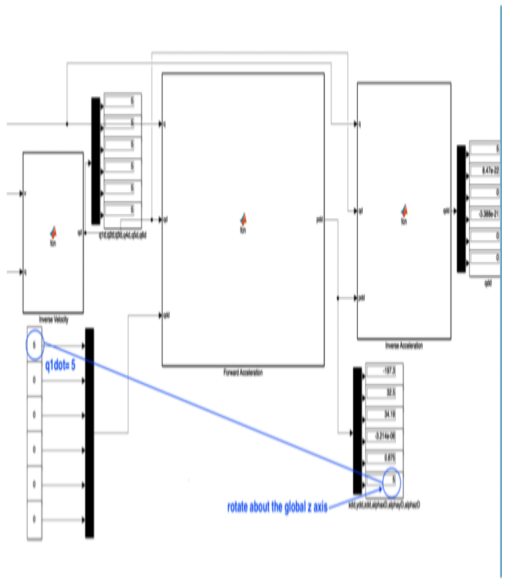
As q1 is rotation about the global Z so it is very easy to be deduced from the output angular velocity, as an example if joint(q1)'s velocity has a certain value, it should result **Only** velocity in x and y directions and angular velocity about global z.



So, The result is as expected, the output is only velocity in x,y and angular velocity about Z.

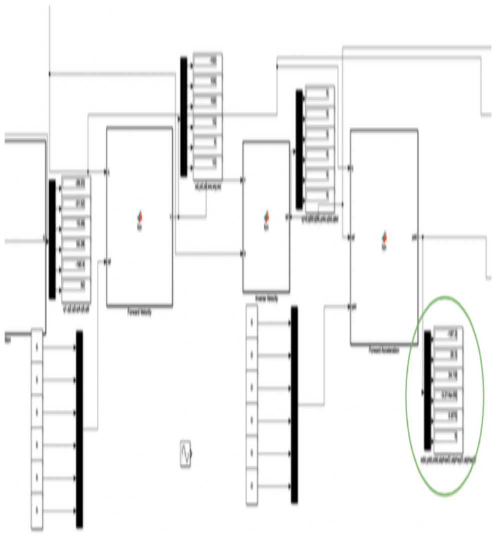
B. Acceleration

With the same concept we used in the velocity analysis, As q1 is rotation about the global z so it is very easy to be deduced from the output acceleration around z, with keeping the velocity of 5 5 rad/sec in every joint, so q1 double dot = certain value, with keeping the other other joint accelerations= 0, this certain value should be the same output in the angular acceleration about z.

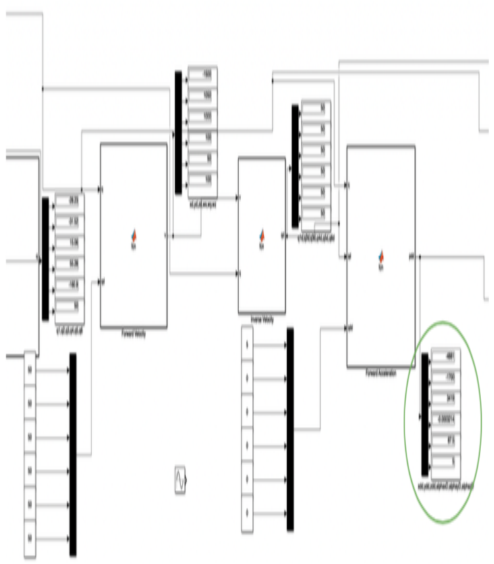


To test the effect of changing the speed only on the acceleration, by increasing the velocity of the q's the output angular acceleration should increase as well as, by keeping the acceleration's input to the joints to zero:

The angular velocities to the joints are all of 5 rad/sec:



The angular velocities to the joints are all of 50 rad/sec:

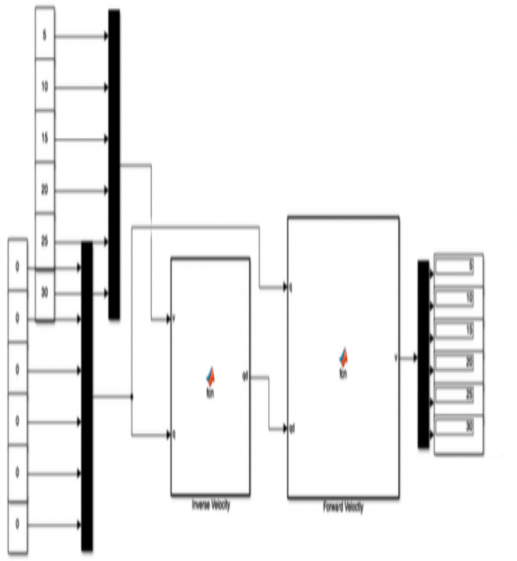


It is pretty obvious the effect of increasing the input angular velocities to the joints on the angular acceleration of the end effector.

V. Testing different input angular velocities and accelerations to the end effector

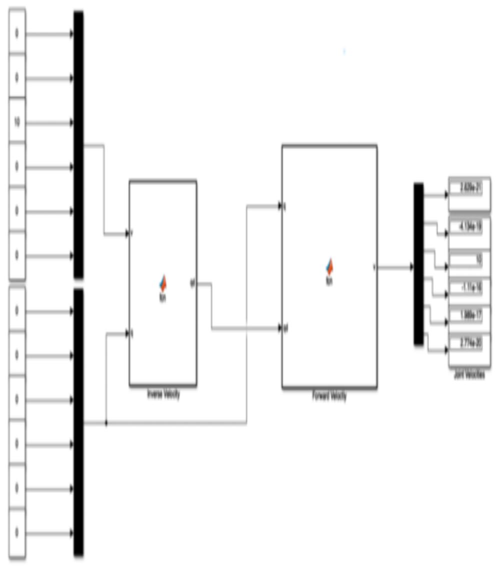
A. Velocity

Validating the input angular velocities to end effector (by passing it to the forward once again) :

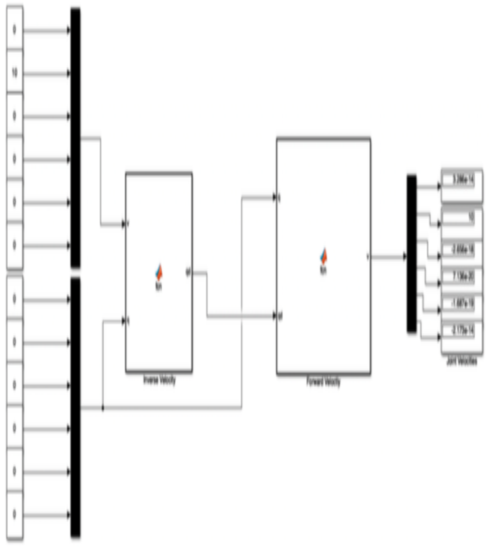


As shown, the input is the same as the output by passing by the inverse then the forward blocks.

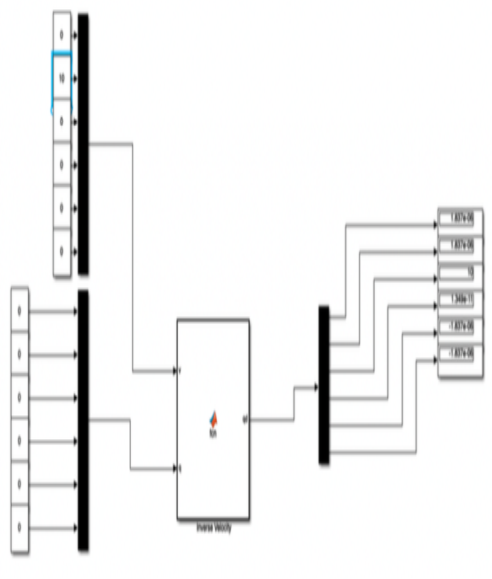
The output is also correct for the next two simulations: (linear velocity in z direction = 10)



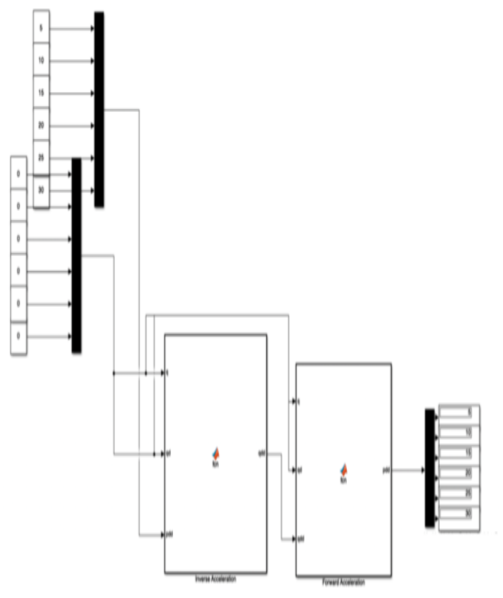
(linear velocity in y direction = 10)



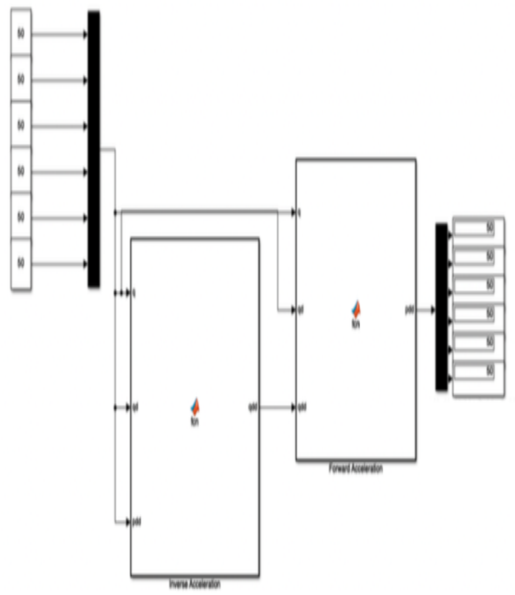
If we require a linear velocity in the y direction by 10 unit/sec, so q3 will be equal to 10 unit/sec, which makes sense as it is the only prismatic joint we have.



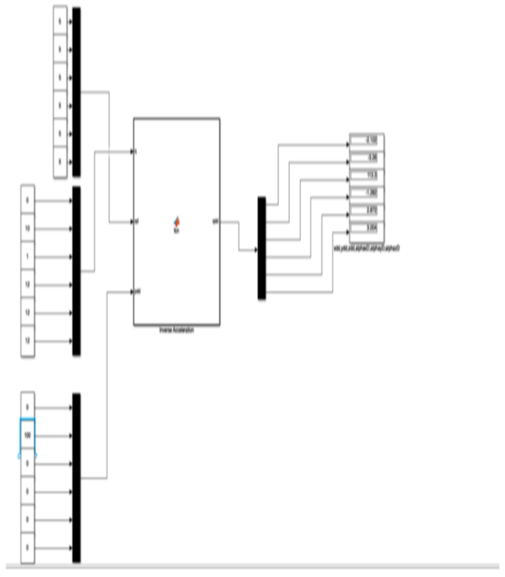
B. Acceleration



This also for another input:



For Input to end effector:
If our robot is moving with a constant velocities in the joints, and it is required to make the end effector to move by linear acceleration in y by 100 rad/sec2, so the required acceleration required will be the largest at q3, which is the only prismatic joint in our robot. So, yes it is logical!



VI. Task-Space Trajectory Planning

A. Straight Line Trajectory:

Initial point (3,5,7) at time = 0
Destination point (-1,2,4) at time = 8
Time(t)=8

$$x(t) = x(0) + \alpha t$$
$$-1 = 3 + \alpha(8)$$
$$\alpha = -4/8 = -0.5 \tag{7}$$

x(t)=3–0.5t

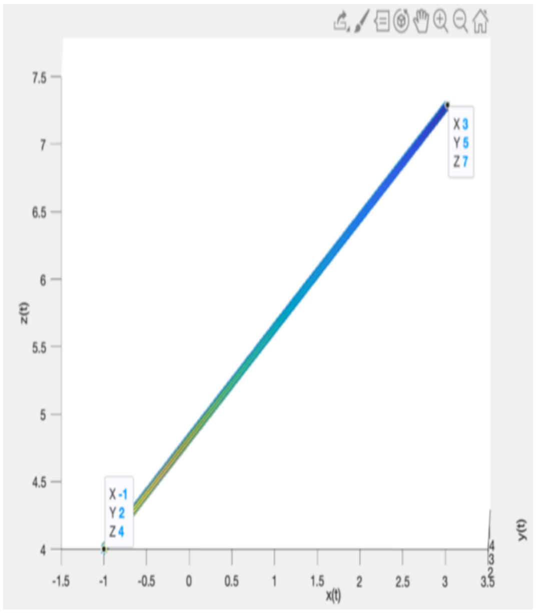
$$y(t) = y(0) + \alpha t$$
$$2 = 5 + \alpha(8)$$
$$\alpha = -3/8 = -0.5 \tag{8}$$

y(t) = 5 – 0.375t

$$z(t) = z(0) + \alpha t$$
$$4 = 7 + \alpha(8)$$
$$\alpha = -3/8 = -0.5 \tag{9}$$

z(t) = 7 – 0.375t

All lengths are always assumed to be 10. (L1 = L2 = L3 = L4 = 10)

line.png 

B. Cone Trajectory:

Initial point (20,y(0),10) at time = 0
Destination point (1,y(8),40) at time = 8
Time(t)=8
Rounds=4

$$\theta = w * t \Rightarrow 4 * 2 * \pi = w * (8) \Rightarrow w = \pi \tag{10}$$

1)

$$t = 0$$
$$x(t) = (r - k * t)cos(w * t)$$
$$20 = r * cos(0) \Rightarrow r = 20$$

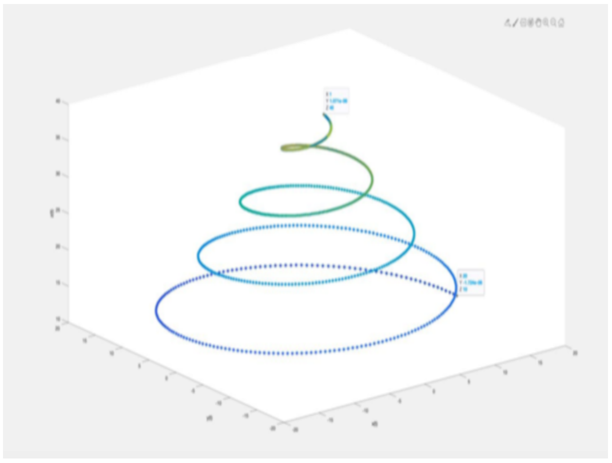
$$y(t) = (r - kt)sin(\pi t)\ can't\ be\ used\ to\ do\ sin(0) = 0$$

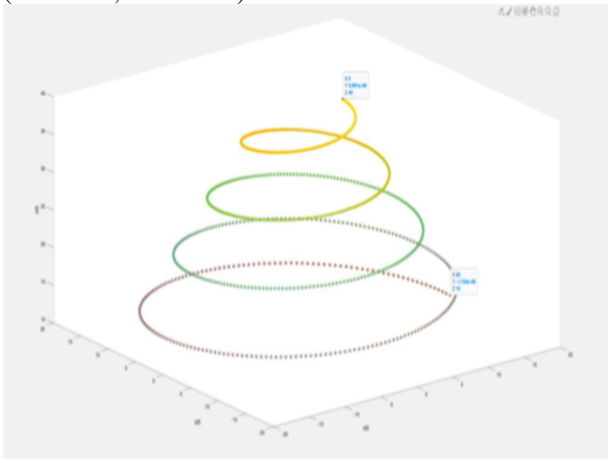
$$z(t) = L_1 + ct10 = L_1 + 0 \Rightarrow L1 = 10 \tag{11}$$

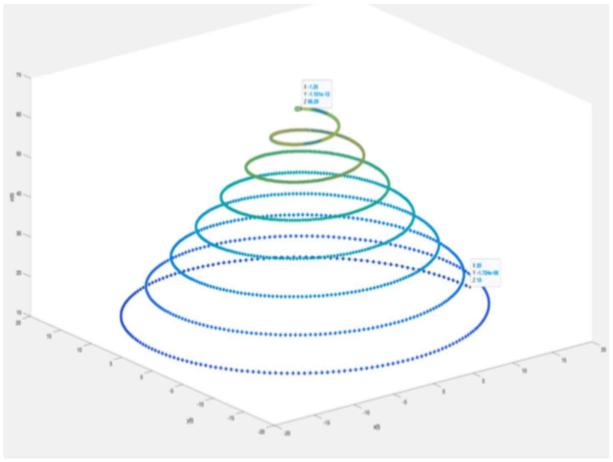
2)

$$t = 8$$
$$x(8) = (20–8k)cos(8 * \pi) = 1 \Rightarrow k = 2.375 \tag{12}$$
$$z(8) = 10 + 8c = 40 \Rightarrow c = 3.75$$

$$x(t) = (20–2.375t)cos(\pi t)$$
$$y(t) = (20–2.375t)sin(\pi t) \tag{13}$$
$$z(t) = 10 + 3.75t$$


(Xdes = 1, zdes = 40)


(Xdes=5, zdes=40)



Changing parameters in order to reach the desired full path.

C. Spiral Ellipse Trajectory:

Initial point (20,y(0),10) at time = 0
Destination point (18,y(8),14) at time = 8
Time(t)=8
Rounds=4

$$\theta = w * t \Rightarrow 4 * 2 * \pi = w * (8) \Rightarrow w = \pi \qquad (14)$$

1)

$$t = 0$$
$$x(t) = (r - k * t)cos(w * t)$$
$$20 = r * cos(0) \Rightarrow r = 20$$

$$y(t) = (r - kt)sin(\pi t)\ can't\ be\ used\ to\ do\ sin(0) = 0$$

$$z(t) = L_1 + ct10 = L_1 + 0 \Rightarrow L1 = 10 \qquad (15)$$

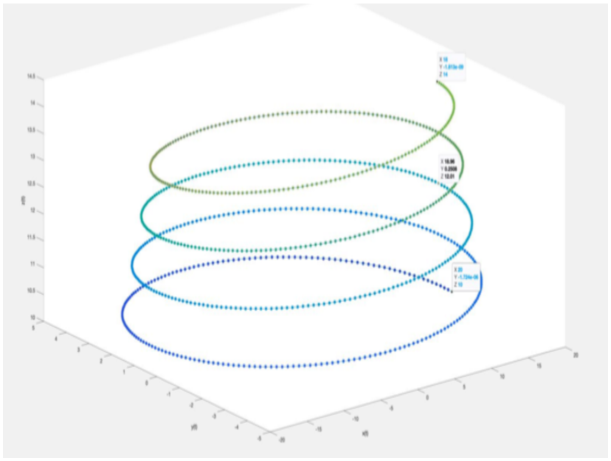
2)

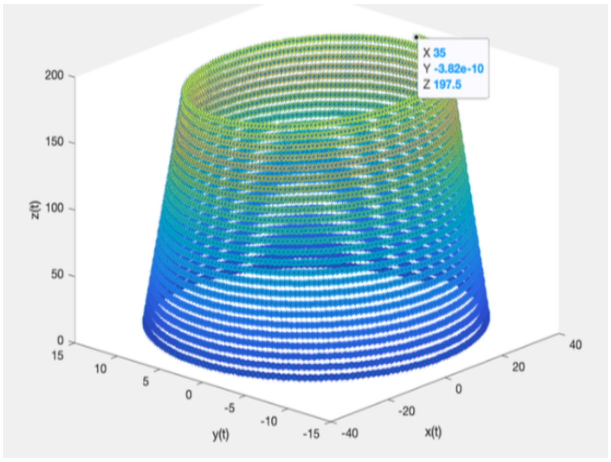
$$t = 8$$
$$x(8) = (20–8k)cos(8 * \pi) = 18 \Rightarrow k = 0.25 \qquad (16)$$
$$z(8) = 10 + 8c = 14 \Rightarrow c = 0.5$$

$$x(t) = (20–0.25t)cos(\pi t)$$
$$y(t) = (5–0.25)sin(\pi t) \qquad (17)$$
$$z(t) = 10 + 0.5t$$

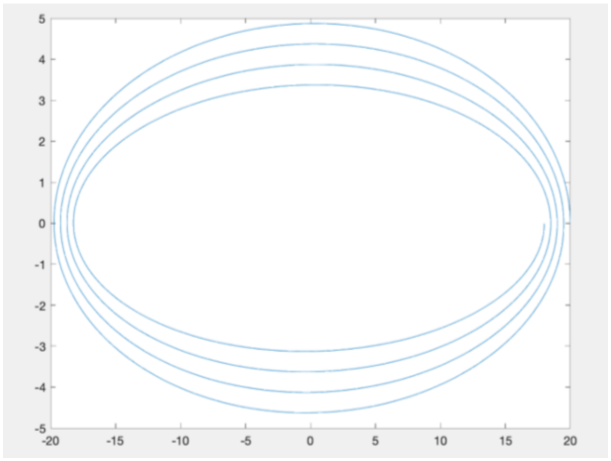
Note: To make the spiral ellipse, K must be ¡1 so that the difference between the desired and the initial is constrained by time.

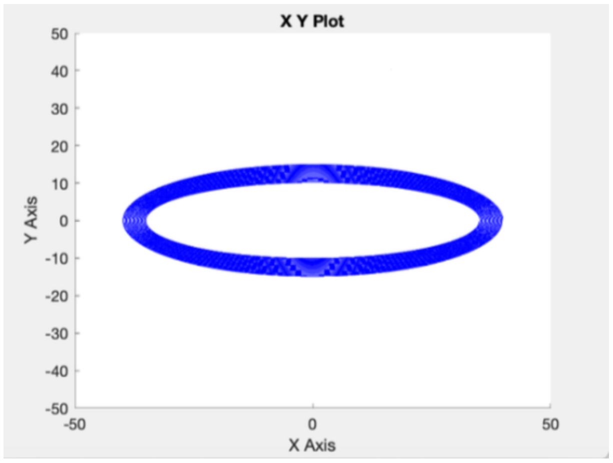
3-Dimensional:

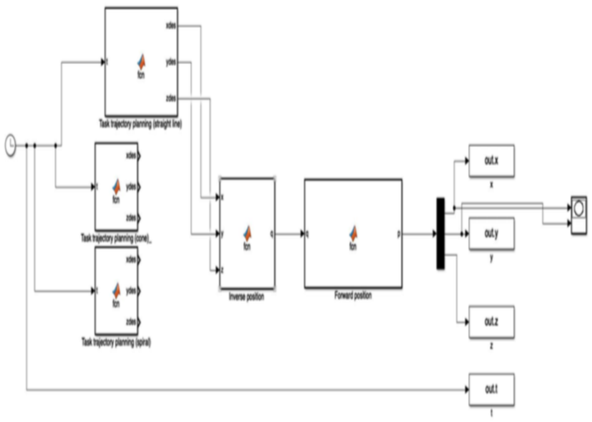




2-Dimensional:





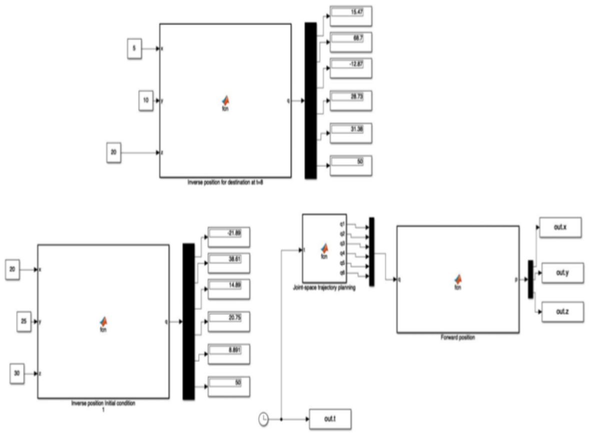


All results are as expected, desired positions were reached at desired time through the expected and planned trajectories.

VII. Joint- Space Trajectory Planning

Polynomial constants of qi(t) equations:

Initial point (position)(20,25,30) at time = 0
Destination point (5,10,20) at time = 8
Time(t)=8
Using inverse and forward Kinematics from previous miles



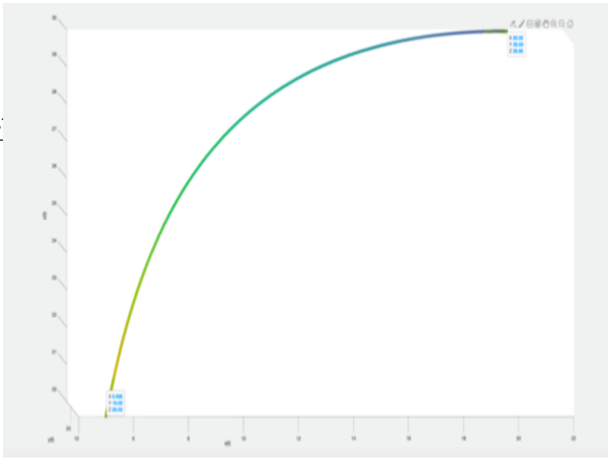
Our robot has 6 qs:

$$q_i(t) = c_0 + c_{1i}t + c_{2i}t^2 + c_{3i}t^3$$
$$q_i dot(t) = c_{1i}t + 2c_{2i}t + 3c_{3i}t^2 \quad (18)$$

since, qi dot(t)=0, at t=0 and t=8

Therefore c1i=0 for all qi.

Joint number	qi(0)	qi(8)	c0i,c2i,c3i
i=1	q1(0)=-21.89	q1(8)=15.47	c01=-21.89 c21=1.75125 c31=0.416
i=2	q2(0)=38.61	q2(8)=68.7	c02=38.61 c22=1.415 c32=-0.118
i=3	q3(0)=14.89	q3(8)=-12.87	c03=14.89 c23=-1.30125 c33=0.1084375
i=4	q4(0)=20.75	q4(8)=28.73	c04=20.75 c24=0.3740625 c34=-0.031172
i=5	q5(0)=8.891	q5(8)=31.38	c05=8.891 c25=1.0542 c35=-0.0878
i=6	q6(0)=0	q6(8)=0	c06=0 c26=0 c36=0



All results are as expected, desired positions were reached at desired time through the expected and planned trajectories.
However due to the inverse kinematics block (shown in

ms2) reaching accuracy = 0.1 , so it gives results (q initial) with error around 0.1 which could be handled to be better and more accurate values by changing the accuracy of the block but it might take more computation time and iterations.

VIII. PID Control

A. Based on Joint- Space Trajectory Planning

Initial point(Position) (20,25,30) at time = 0
Destination point (5,10,20) at time = 8
Time(t)=8

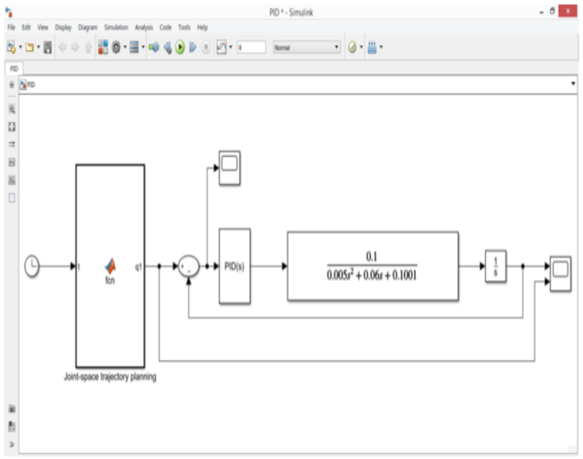
In Quiz 2, using inverse and forward Kinematics from previous milestones, qs are obtained then based on this milestone requirements

$$q_1(t) = -21.89 + 1.75125t^2 - 0.146t^3 \tag{19}$$

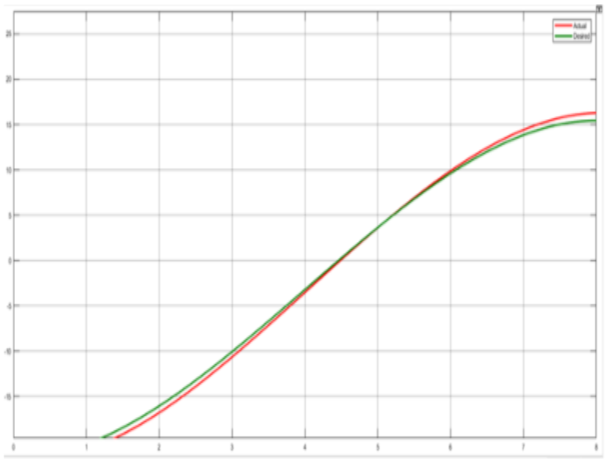
was used to control the first (base) joint through determining the reference (q desired) values.

DC-motor Transfer function(Used as a plant):

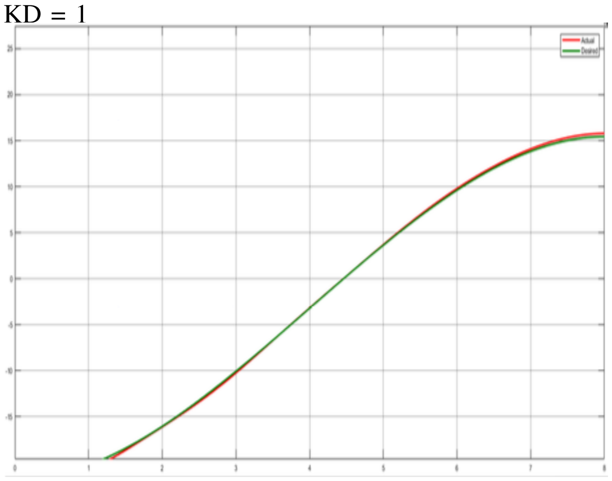
$$\frac{0.1}{0.005s^2 + 0.06s + 0.1001} \tag{20}$$


PID * - Simulink
fcn
q1
PID(s)
$\frac{0.1}{0.005s^2+0.06s+0.1001}$
$\frac{1}{s}$
Joint-space trajectory planning

Different Responses based on changing the PID gains :
Kp = 5
KI = 3
KD = 2

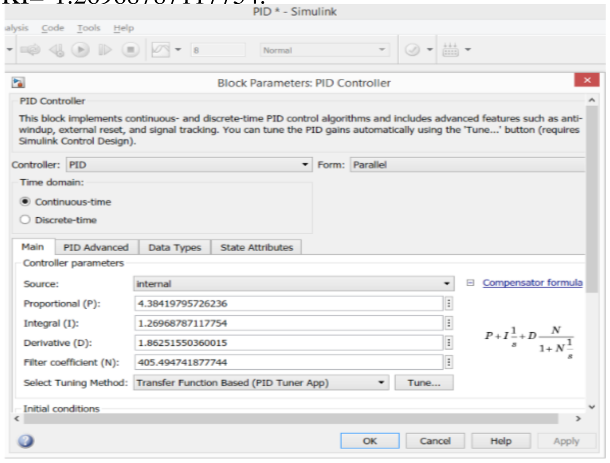


Kp = 7
KI = 10
KD = 1



The final Response reached after the Auto-tune of the PID.

Kp= 4.38419795726236.
Kd= 1.86251550360015.
KI= 1.26968787117754.


PID * - Simulink

Block Parameters: PID Controller

PID Controller

This block implements continuous- and discrete-time PID control algorithms and includes advanced features such as anti-windup, external reset, and signal tracking. You can tune the PID gains automatically using the 'Tune...' button (requires Simulink Control Design).

Controller: PID   Form: Parallel

Time domain:
(●) Continuous-time
( ) Discrete-time

Main | PID Advanced | Data Types | State Attributes

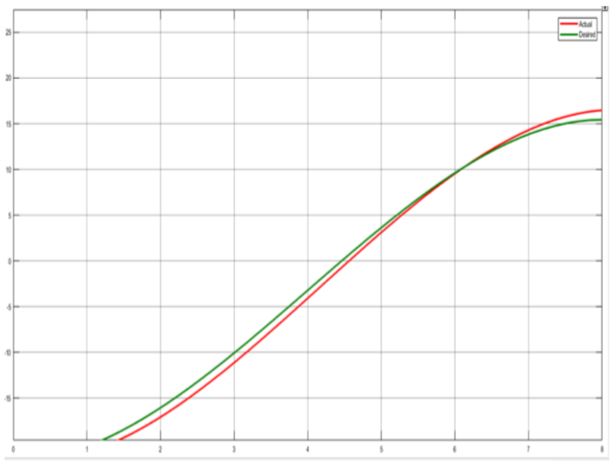
Controller parameters

Source: internal
Proportional (P): 4.38419795726236
Integral (I): 1.26968787117754
Derivative (D): 1.86251550360015
Filter coefficient (N): 405.494741877744
Select Tuning Method: Transfer Function Based (PID Tuner App)

Compensator formula

$$P + I\frac{1}{s} + D\frac{N}{1+N\frac{1}{s}}$$

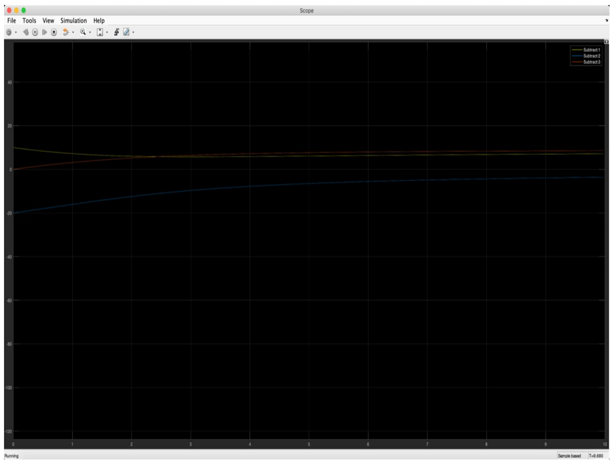
Initial conditions

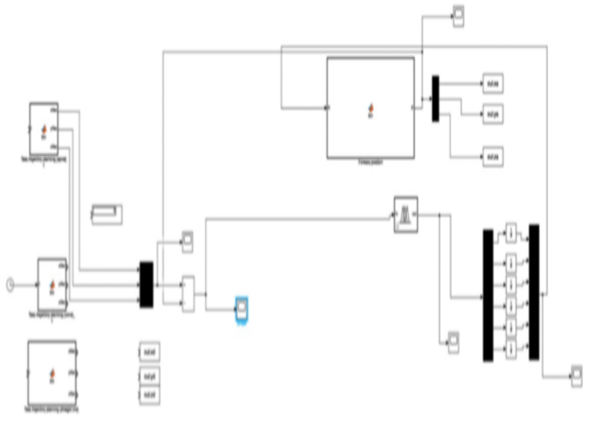


By having these actual responses vs the desired one we can decide that these gains from the auto- tune and Kp =7 KI = 10,Kd=1 for the PID are suitable and accepted in order to have the expected and satisfying control over the joint motion.

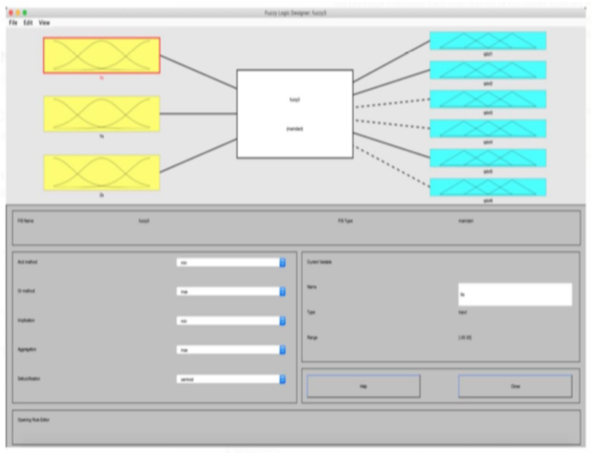
IX. Fuzzy Logic Control

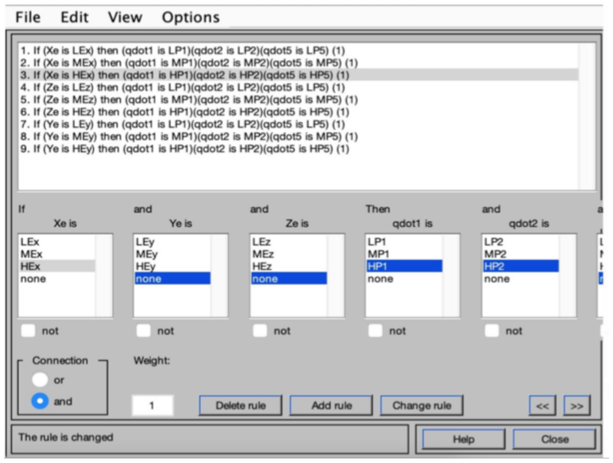
- In the high level control the fuzzy control was used to to control and y positions. In the Matlab block for the fuzzy controller for the 2 inputs x and y and z 3 member ship functions were made and 6 outputs for the 6 joints in the robot.
- For the control loop the desired values are from the task space made from the previous milestone then then the fuzzy controller controls the output for each motor for each joint weather its high speed, medium speed or low speed then the actual position of the joints is compared with the desired, the response of the control loop is shown below.





- The range of x and y is form -20 to 20 and the range of z is from 10 to 50 and the qdot is from -50 to 50





1. If (Xe is LEx) then (qdot1 is LP1)(qdot2 is LP2)(qdot5 is LP5) (1)
2. If (Xe is MEx) then (qdot1 is MP1)(qdot2 is MP2)(qdot5 is MP5) (1)
3. If (Xe is HEx) then (qdot1 is HP1)(qdot2 is HP2)(qdot5 is HP5) (1)
4. If (Ze is LEz) then (qdot1 is LP1)(qdot2 is LP2)(qdot5 is LP5) (1)
5. If (Ze is MEz) then (qdot1 is MP1)(qdot2 is MP2)(qdot5 is MP5) (1)
6. If (Ze is HEz) then (qdot1 is HP1)(qdot2 is HP2)(qdot5 is HP5) (1)
7. If (Ye is LEy) then (qdot1 is LP1)(qdot2 is LP2)(qdot5 is LP5) (1)
8. If (Ye is MEy) then (qdot1 is MP1)(qdot2 is MP2)(qdot5 is MP5) (1)
9. If (Ye is HEy) then (qdot1 is HP1)(qdot2 is HP2)(qdot5 is HP5) (1)

If Xe is	and Ye is	and Ze is	Then qdot1 is	and qdot2 is
LEx	LEy	LEz	LP1	LP2
MEx	MEy	MEz	MP1	MP2
HEx	HEy	HEz	[✓ HP1]	[✓ HP2]
none	[✓ none]	[✓ none]	none	none
not	not	not	not	not

Connection: or [✓ and]

Weight: 1

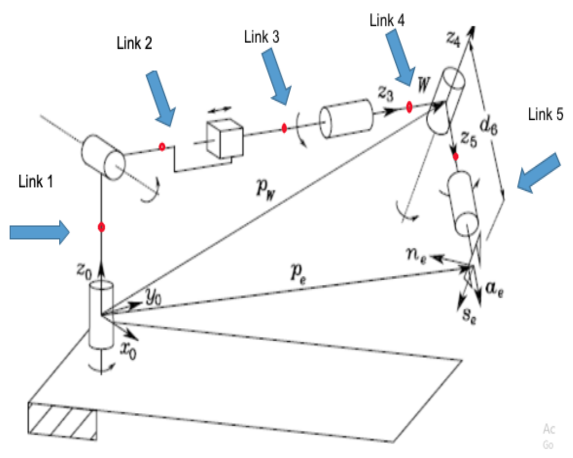
Delete rule | Add rule | Change rule | << | >>

The rule is changed

Help | Close

X. Dynamics

First of all we had to get the positions of the center of masses of each link



To get these points we had to get the forth column of each transformation matrix, however we had to divide the length of the link we are studying by half like of example in the first link
The forth column was

$$\begin{vmatrix} 0 \\ 0 \\ L1 \end{vmatrix}$$

therefore, our first center of mass equation is

$$\begin{vmatrix} 0 \\ 0 \\ L1/2 \end{vmatrix}$$

and so on:

We ended up having these position matrices

LM1=[0;0;L1/2];

LM2=[cos((pi*(q1(t) - 90))/180)*sin((pi*(q2(t) - 90))/180)*(L2/2);sin((pi*(q1(t) - 90))/180)*sin((pi*(q2(t) - 90))/180)*(L2/2); L1 + cos((pi*(q2(t) - 90))/180)*(L2/2)];

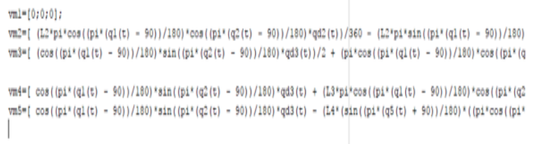
LM3=[cos((pi*(q1(t) - 90))/180)*sin((pi*(q2(t) - 90))/180)*(L2 + q3(t)/2);sin((pi*(q1(t) - 90))/180)*sin((pi*(q2(t) - 90))/180)*(L2 + q3(t)/2); L1 + cos((pi*(q2(t) - 90))/180)*(L2 + q3(t)/2)];

LM4=[cos((pi*(q1(t) - 90))/180)*sin((pi*(q2(t) - 90))/180)*(L2 + q3(t)) + (L3*cos((pi*(q1(t) - 90))/180)*sin((pi*(q2(t) - 90))/180))/2; sin((pi*(q1(t) - 90))/180)*sin((pi*(q2(t) - 90))/180)*(L2 + q3(t)) + (L3*sin((pi*(q1(t) - 90))/180)*sin((pi*(q2(t) - 90))/180))/2;L1 + cos((pi*(q2(t) - 90))/180)*(L2 + q3(t)) + (L3*cos((pi*(q2(t) - 90))/180))/2];

LM5=[cos((pi*(q1(t) - 90))/180)*sin((pi*(q2(t) - 90))/180)*(L2 + q3(t)) - (L4*(sin((pi*(q5(t) + 90))/180)*(sin((pi*(q1(t) - 90))/180)*sin((pi*(q4(t) + 90))/180) - cos((pi*(q1(t) - 90))/180)*cos((pi*(q2(t) - 90))/180)*cos((pi*(q4(t) + 90))/180)) + cos((pi*(q1(t) - 90))/180)*cos((pi*(q5(t) + 90))/180)*sin((pi*(q2(t) - 90))/180)))/2 + L3*cos((pi*(q1(t) - 90))/180)*sin((pi*(q2(t) - 90))/180);(L4*(sin((pi*(q5(t) + 90))/180)*(cos((pi*(q1(t) - 90))/180)*sin((pi*(q4(t) + 90))/180) + cos((pi*(q2(t) - 90))/180)*cos((pi*(q4(t) + 90))/180)*sin((pi*(q1(t) - 90))/180)) - cos((pi*(q5(t) + 90))/180)*sin((pi*(q1(t) - 90))/180)*sin((pi*(q2(t) - 90))/180)))/2 + sin((pi*(q1(t) - 90))/180)*sin((pi*(q2(t) - 90))/180)*(L2 + q3(t)) + L3*sin((pi*(q1(t) - 90))/180)*sin((pi*(q2(t) - 90))/180);L1 - (L4*(cos((pi*(q2(t) - 90))/180)*cos((pi*(q5(t) + 90))/180) + cos((pi*(q4(t) + 90))/180)*sin((pi*(q2(t) - 90))/180)*sin((pi*(q5(t) + 90))/180)))/2 + cos((pi*(q2(t) - 90))/180)*(L2 + q3(t)) + L3*cos((pi*(q2(t) - 90))/180)];

As we can see each Centre of Mass position of each link depends directly at the previous links as you take the length of the previous link(s) into consideration while calculating the position.

After that we differentiate the matrices with respect to time in order to get the velocity of each link so we end up with more complicated and much longer matrices



```
vm1=[0;0;0];
vm2=[ (L2*pi*cos((pi*(q1(t) - 90))/180)*cos((pi*(q2(t) - 90))/180)*qd2(t))/360 - (L2*pi*sin((pi*(q1(t) - 90))/180)
vm3=[ (cos((pi*(q1(t) - 90))/180)*sin((pi*(q2(t) - 90))/180)*qd3(t))/2 + (pi*cos((pi*(q1(t) - 90))/180)*cos((pi*(q

vm4=[ cos((pi*(q1(t) - 90))/180)*sin((pi*(q2(t) - 90))/180)*qd3(t) + (L3*pi*cos((pi*(q1(t) - 90))/180)*cos((pi*(q2
vm5=[ cos((pi*(q1(t) - 90))/180)*sin((pi*(q2(t) - 90))/180)*qd3(t) - (L4*(sin((pi*(q5(t) + 90))/180)*((pi*cos((pi*
```

That's a screenshot of how the matrices are placed in the m.file

After that we calculated the angular velocities of the links. In order to get that we got the jw of each link which was calculated in the inverse velocity procedures. They were calculated by getting the third column of the transformations matrices:

Jw1=[0;0;1];
Jw2= A0T1(:,3);
Jw3=[0;0;0];
Jw4= A0T3(:,3);
Jw5= A0T4(:,3);
Jw6= A0T5(:,3);

Then to get the angular velocities we made the following calculations

w1=qd1(t)*Jw1;
w2=w1+qd2(t)*Jw2;
w3=w2+qd3(t)*Jw3;
w4=w3+qd4(t)*Jw4;
w5=w4+qd5(t)*Jw5;

Then The Kinetic energies were calculated:

$$T_{i,trans} = \frac{1}{2} m_i \; v_{CMi}^T \; v_{CMi} \tag{21}$$

In our code it was as follows:

```
TM1=0.5*m1*transpose(vm1)*vm1+0.5*transpose(w1)*I1*w1;
TM2=0.5*m2*transpose(vm2)*vm2+0.5*transpose(w2)*I2*w2;
TM3=0.5*m3*transpose(vm3)*vm3+0.5*transpose(w3)*I3*w3;
TM4=0.5*m4*transpose(vm4)*vm4+0.5*transpose(w4)*I4*w4;
TM5=0.5*m5*transpose(vm5)*vm5+0.5*transpose(w5)*I5*w5;
```

After that the potential energy was calculated:

$$U_i = m_i \; g \; P_{CMi} \tag{22}$$

In our code it was as follows:

```
g=[0,0,-9.8];
U1=m1*g*LM1;
U2=m2*g*LM2;
U3=m3*g*LM3;
U4=m4*g*LM4;
U5=m5*g*LM5;
```

To get the lagrange equations we do the following:
L=TM1+TM2+TM3+TM4+TM5-U1-U2-U3-U4-U5;

Then we had to differentiate the Lagrange equation with respect to (q1,q2,q3,q4 and q5) in order to get

$$\frac{\partial L}{\partial q_1}, \frac{\partial L}{\partial q_2}, \frac{\partial L}{\partial q_3}, \frac{\partial L}{\partial q_4}, \frac{\partial L}{\partial q_5} \tag{23}$$

After that we had to operate

$$\frac{\partial L}{\partial \dot{q}_1}, \frac{\partial L}{\partial \dot{q}_2}, \frac{\partial L}{\partial \dot{q}_3}, \frac{\partial L}{\partial \dot{q}_4}, \frac{\partial L}{\partial \dot{q}_5} \tag{24}$$

After that we had to also differentiate the last equations with respect to time so the sequence of operations was as follows: Lq1d=diff(L,q1dot)
Lq1dt=diff(Lq1d,t)
Lq1=diff(L,q1)
And so on with the other links

To get the torques:
tow1=Lq1dt-Lq1;
tow2=Lq2dt-Lq2;
tow3=Lq3dt-Lq3;
tow4=Lq4dt-Lq4;
tow5=Lq5dt-Lq5;
If we see the matrices we will find that tow 1 had all the masses and the velocities into account, this is because each torque affect the link and each following link(s) connected to it.