1. OBJECTIVE:

The main objectives of this coursework is to perform the following tasks and findings

- To implement PUMA560 robot using robotics toolbox with right dh parameters and be able to animate it using matlab.
- To run the simulation of independent joint control of a motor and analyse the actual and desired joint speed and errors.
- To analyze the performance of the motor for different values of Kv and Kp
- To perform the root locus analysis of vloop and analyze the maximum possible gain for the proportional gain
- To analyze the effect of speed on torque and integral controller

2.1 PUMA560

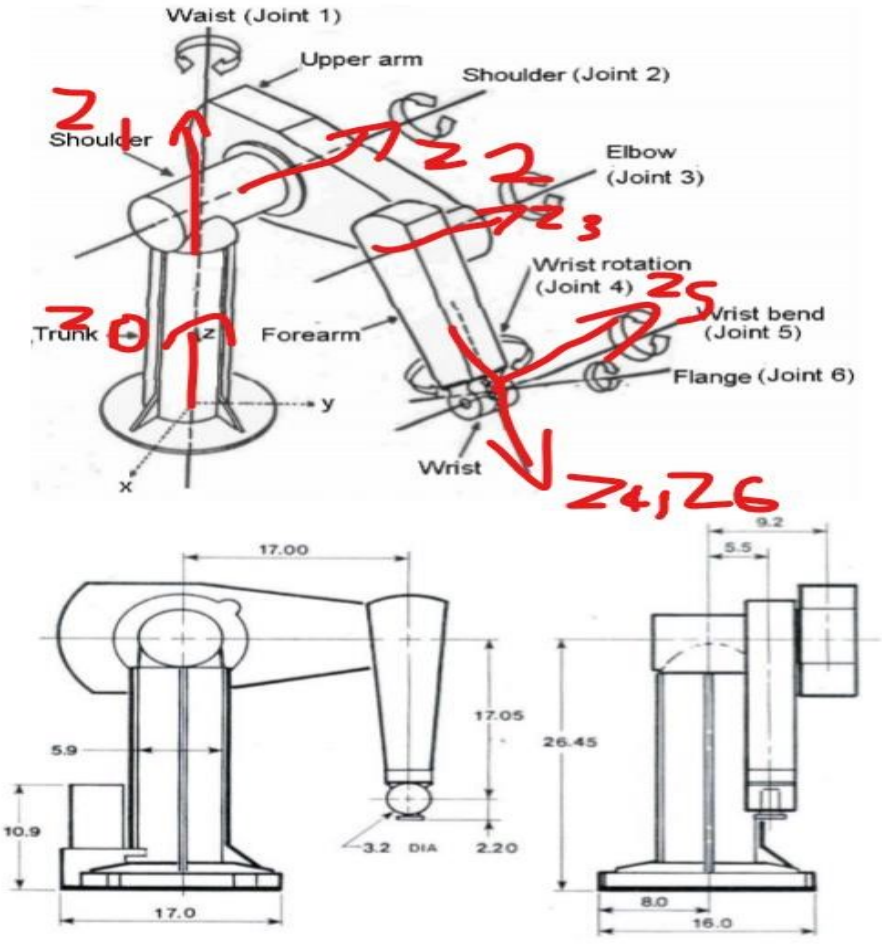


Fig: Puma560 with frames attached

The generated DH PARAMETERS and 3d layout.

```
robo =

noname:: 6 axis, RRRRRR, stdDH, slowRNE
+---+-----------+-----------+-----------+-----------+-----------
| j |     theta |         d |         a |     alpha |    offset
+---+-----------+-----------+-----------+-----------+-----------
|  1|         q1|      26.45|          0|     1.5708|          0
|  2|         q2|       -9.2|         17|          0|          0
|  3|         q3|        3.7|         17|     1.5708|          0
|  4|         q4|      17.05|          0|     1.5708|          0
|  5|         q5|          0|          0|    -1.5708|          0
|  6|         q6|        2.2|          0|          0|          0
+---+-----------+-----------+-----------+-----------+-----------
```

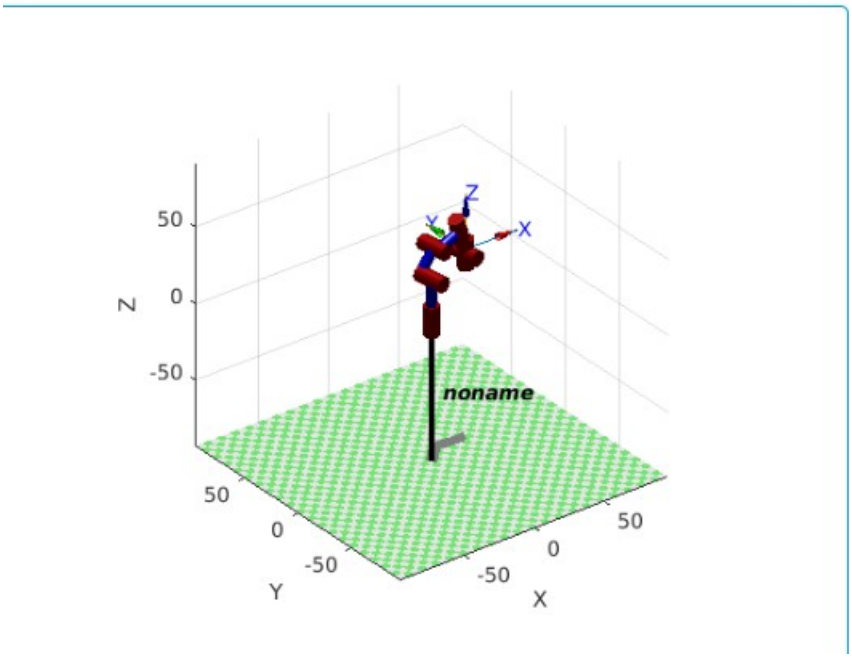


Fig: Matlab Output space

CODE:

```
clf

%Defining the DH parameters
 rev1 = Revolute('d', 26.45, 'a',    0, 'alpha',    pi/2);
 rev2 = Revolute('d',    -9.2, 'a',   17, 'alpha',       0);
 rev3 = Revolute('d',   3.7, 'a',    17, 'alpha',    pi/2);
 rev4 = Revolute('d', 17.05, 'a',    0, 'alpha',     pi/2);
 rev5 = Revolute('d',    0, 'a',    0, 'alpha',    -pi/2);
 rev6 = Revolute('d',    2.2, 'a',    0, 'alpha',       0);
%defining the joint matrices
joints = [rev1, rev2, rev3, rev4, rev5, rev6];
robo = SerialLink(joints)
% r.plot([0, 0, 0, 0, 0, 0, 0, 0])
```

```
% r.teach(joints)


%starting point
A = [10, 10, 10];
%end point
B = [ 32, 10, 30];
%gives translations happening at every point
T1 = transl(A);
T2 = transl(B);
%Put the obtained values into matrix
T = ctraj(T1,T2,20);
%Perform ikine function from robotics toolbox
q = robo.ikine(T);
robo.name('PUMA')
robo.plot(q);
```

Forward kinematics asks where the end effector of the **arm** will be following a sequence of rotations of the joints of the **arm**. **Inverse kinematics** asks what rotations of the joints will bring the end effector to a specified position. We have successfully demonstrated the motion of the robotic arm in MATLAB by using these algorithms.

2.2 INDEPENDENT JOINT CONTROLS

The analysis of the independent joint control of a motor can be done using the toolbox. The equivalent block diagram is given below and performance have been tested by changing the parameters

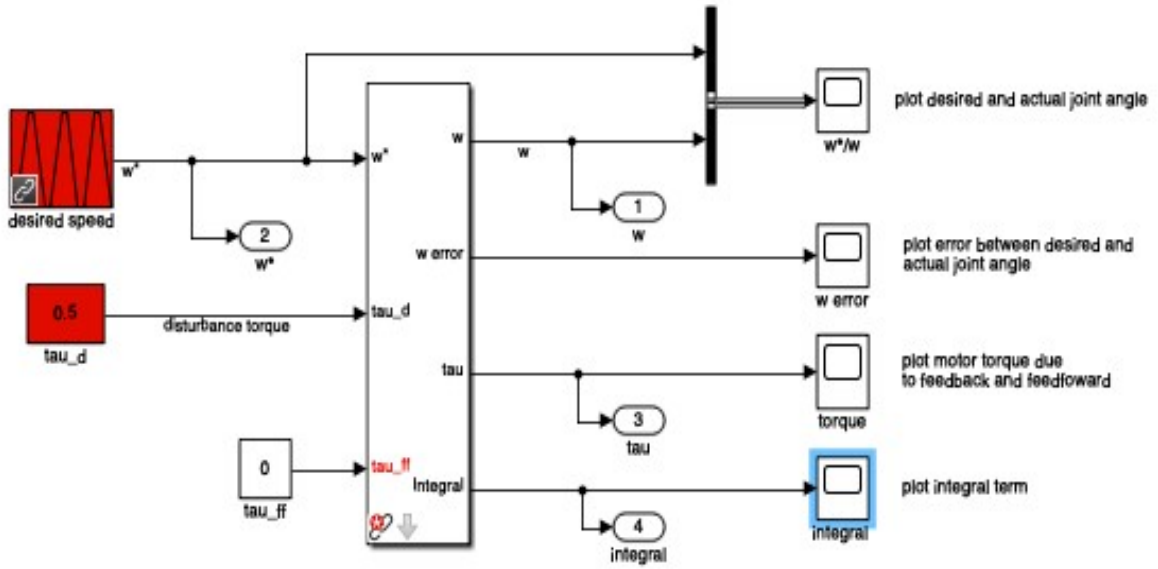
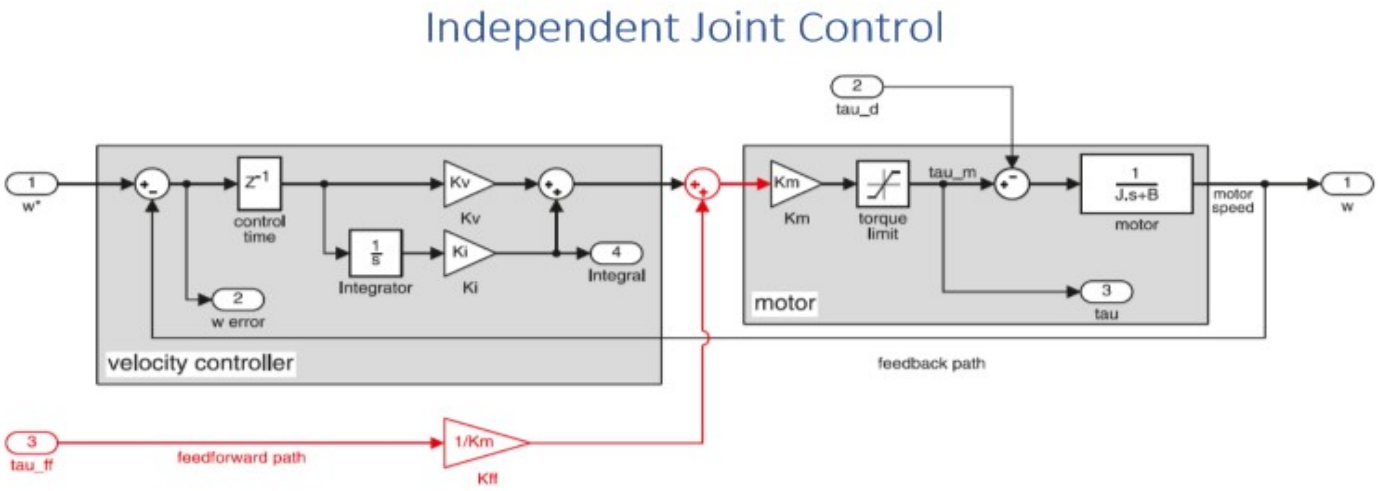
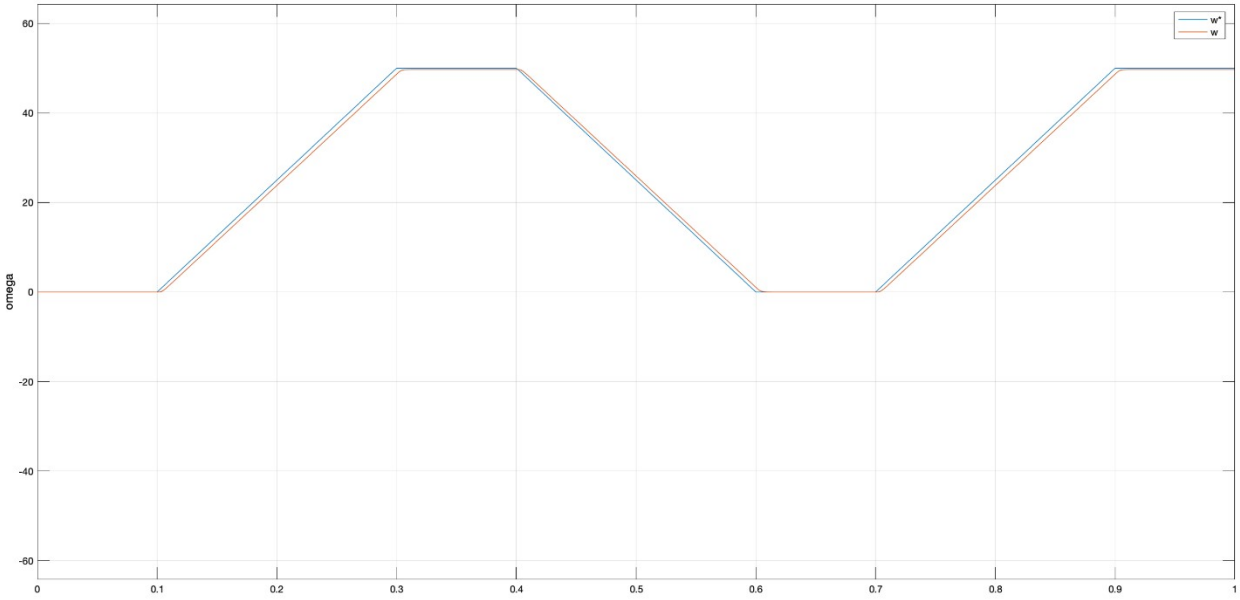


Fig: General Block diagram for independent joint control for motor

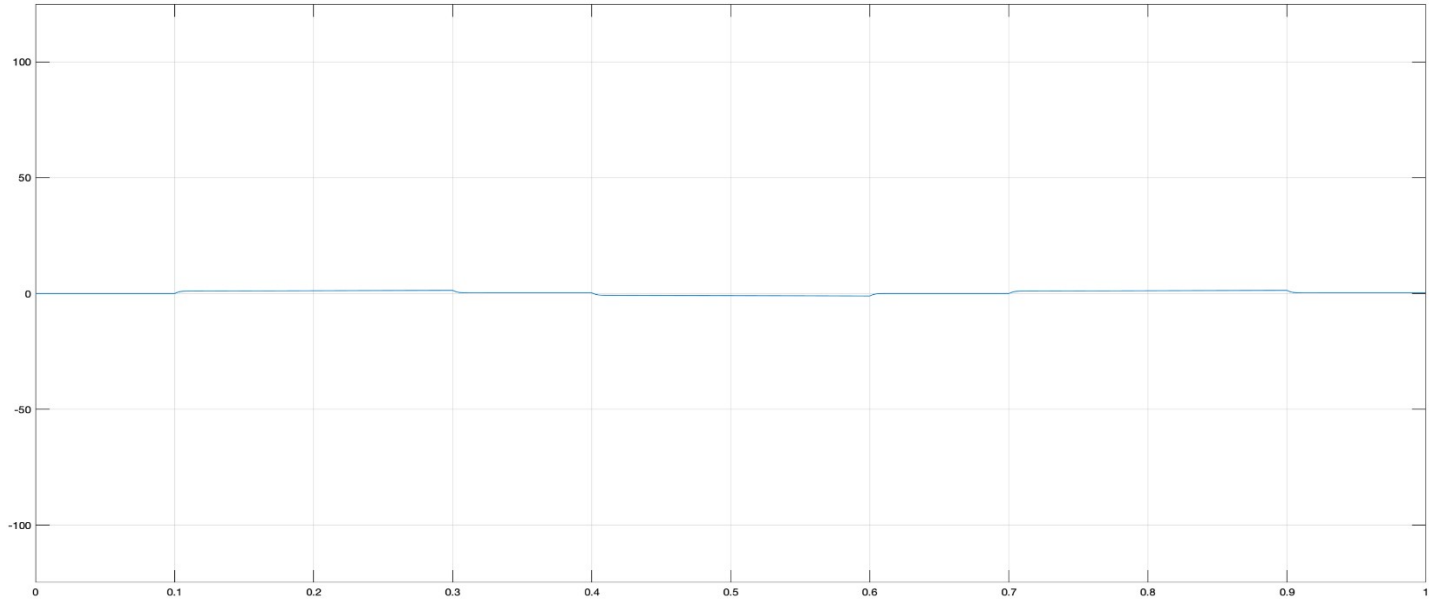


This design uses PI controller to compensate the velocity of the motor to the given inputs and output load. The feedforward controller function tracks the time varying trajectories and compensates the time varying disturbances.

A) The curve of actual and desired joint speed(w^*/w)

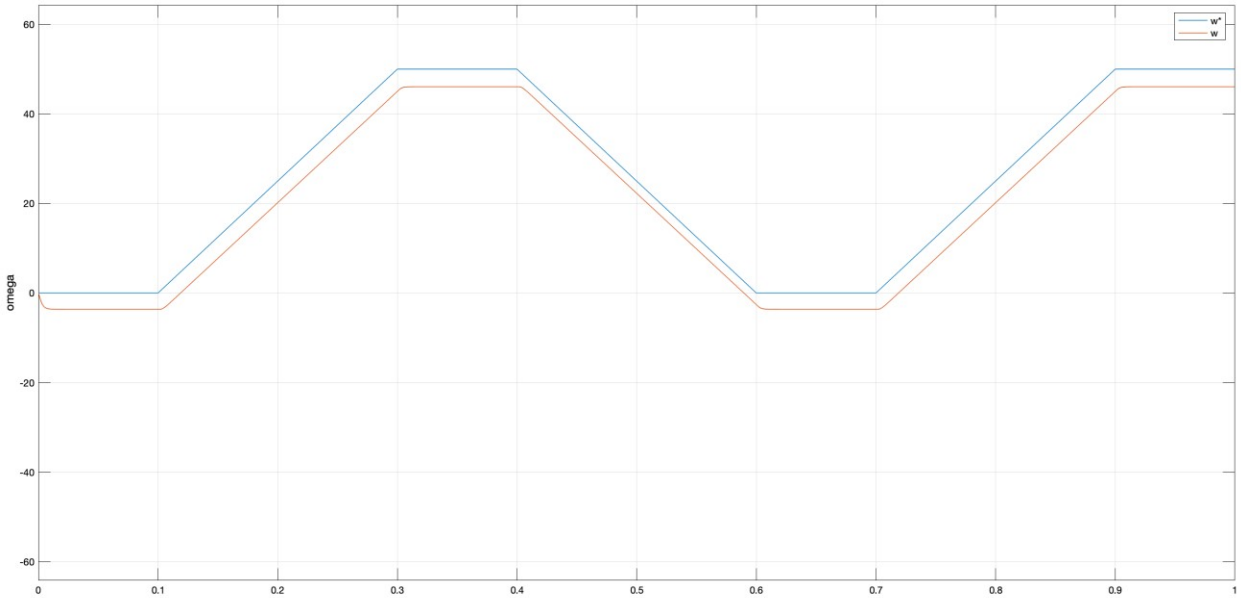


The speed error (w error)

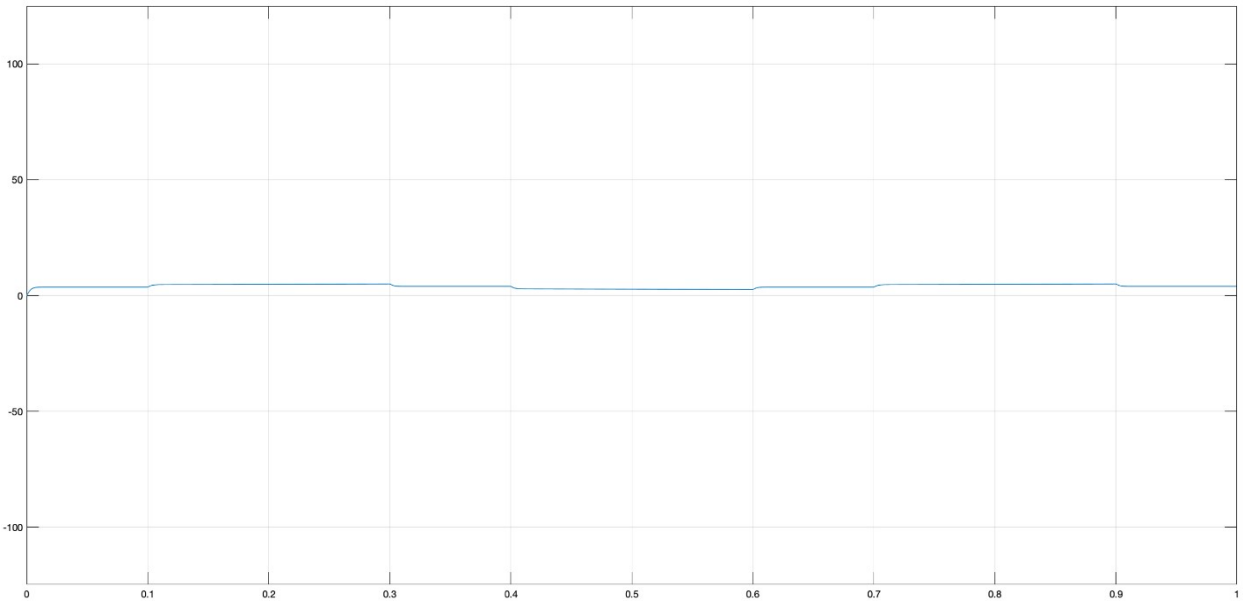


B) When Tau_d is 0.5

The curve of actual and desired joint speed(w^*/w)



The speed error (w error)

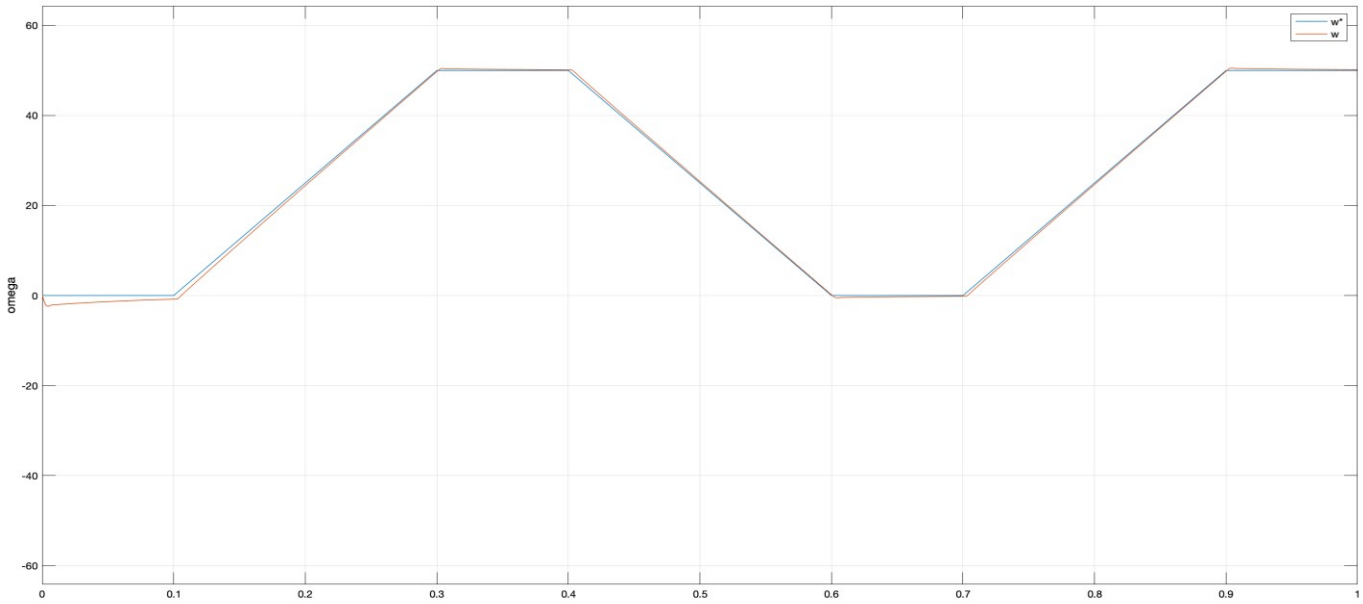


C)

The speed of the motor is rised linearly after required input fed into the controller. Increasing the proportional gain K_p will reduce the steady-state error. However, also recall that increasing K_p often results in increased overshoot, therefore, it appears that not all of the design requirements can be met with a simple proportional controller. We put Ki much larger in order to build up the integral action and eliminate the steady state error as fast as possible. So, I on the accumulation of past errors, and D is a prediction of future errors, based on current rate of change.

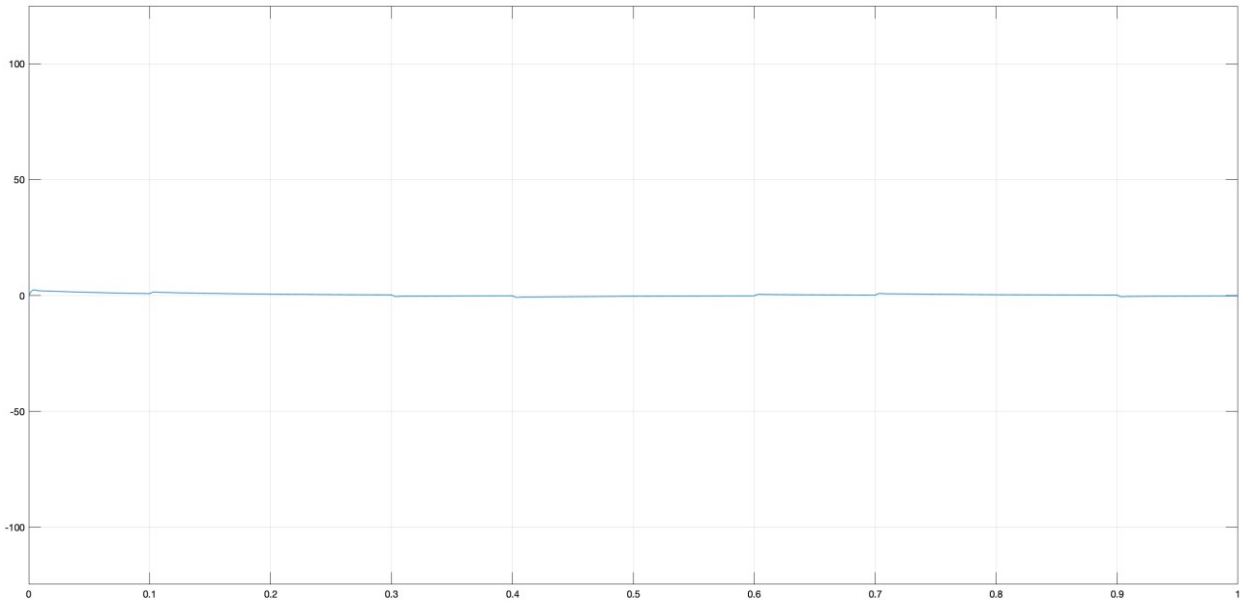
$$\boldsymbol{K_v = 1 \quad K_I = 10}$$

The curve of actual and desired joint speed(w^*/w)



The Kp and Ki has been tuned to get minimum possible error in actual and desired joint speed graphs

The speed error (w error)



D) A root-locus analysis of vloop to determine the maximum permissible gain for the proportional case

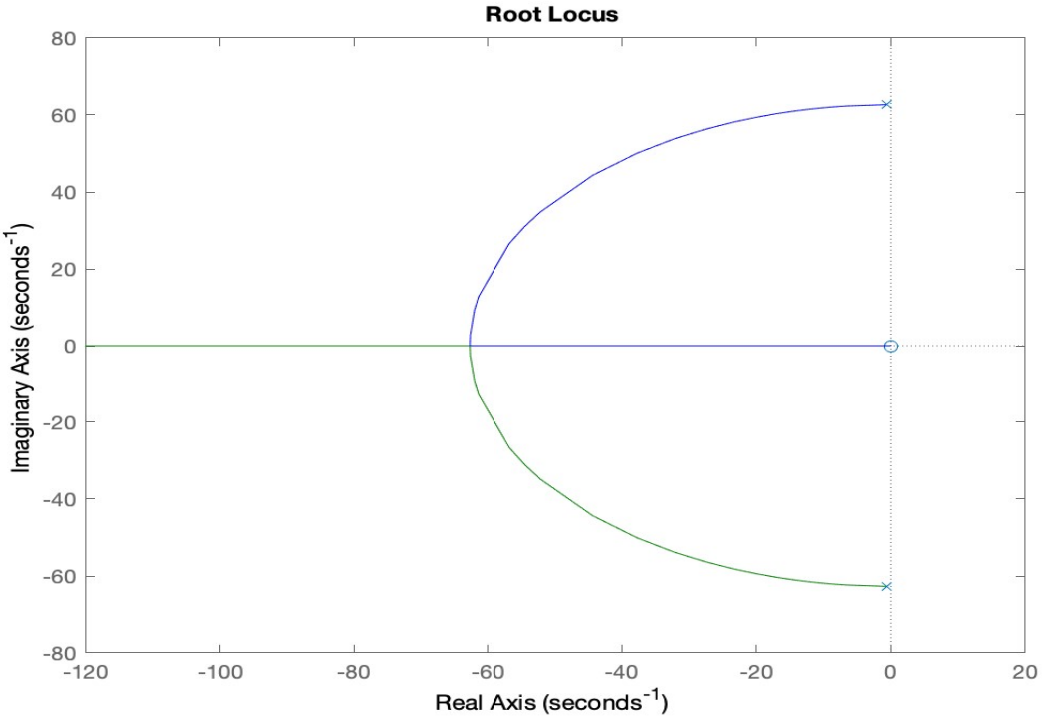
Code:

```
J = 580e-6;          % Inertia
B = 817e-6;          % Viscous friction coefficient
Km = 0.228;          % Motor torque constant
Ki = 10 ;      % Integral gain

num = [Km 0];
den = [J B Ki*Km];

Hs = tf(num,den) ;
```

```
rlocus(Hs);
```



This root locus shows that our values taken for Kv and Ki are appropriate and stable

E. Analyses for Integral windup:

Integral windup is the process of accumulating the integral component beyond the saturation limits of final control element. The formula for integral component in PID is

Integral component = (Ts/Ti) * error

Where Ts = time cycle

Ti = Integral time constant or reset rate

In our case when the speed is increased suddenly, the integrator control watches it as an error and compensates the increased speed which may over compensate and enter into a point beyond the physical limit (called saturation point). When this happens, we lose our control and the torque of the motor remains constant.

Now if the operator has raised the set-point to 90% suddenly then according to the error the PI controller increases the output to raise the speed of the motor.

However, the motor speed cant be raised all of a sudden and usually takes some time to attain the desired speed, but according to the integrator, the motor speed is lagging and keeps increasing the gain to a point that the motor speed cant reach the value. Now when we try to reduce the speed, the gain at the integrator is not well synchronized and hence it remains in the saturation point where the motor controller cannot respond properly to the given input and speed remains high.

We can prevent this error by setting suitable set points for the integral control and slowly changing the motor speed. And also, by designing a anti-windup controller along with our PI, which could lead to more stable but complicated circuit.

The graph below shows the sudden increase in the integral control gain when the speed is increased to its maximum

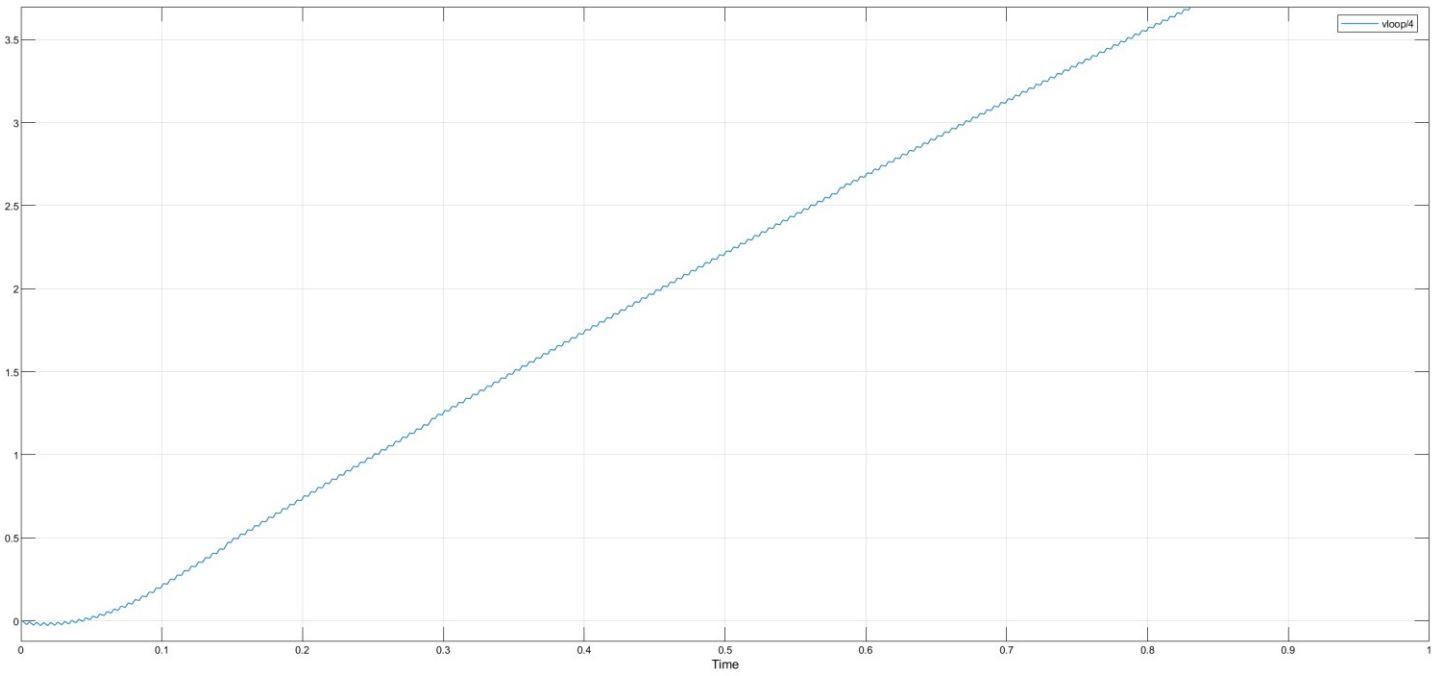


Fig: Integral control response for sudden step input

3. CONCLUSION:

Thus, we have successfully demonstrated the animation of PUMA560 and have performed analyses of independent joint control for different values for gain, and have discussed the windup phenomenon which could possibly happen in case of inappropriate step input. Our current model is ready to be implemented with motor joints of the PUMA56.