MITACS Project Documentation

Saayuj Deshpande

Introduction:-

The task was to reduce or dampen the vibrations faced by a 6-axis robotic arm in its movement at certain orientations and certain velocities. For this purpose, Simscape was used to create a simulation with the various components of the robotic arm. Later, methods for input shaping were researched and app development was carried out in Python using a library called Streamlit.

Simulation-1:-

Here is the link to the simulation file.

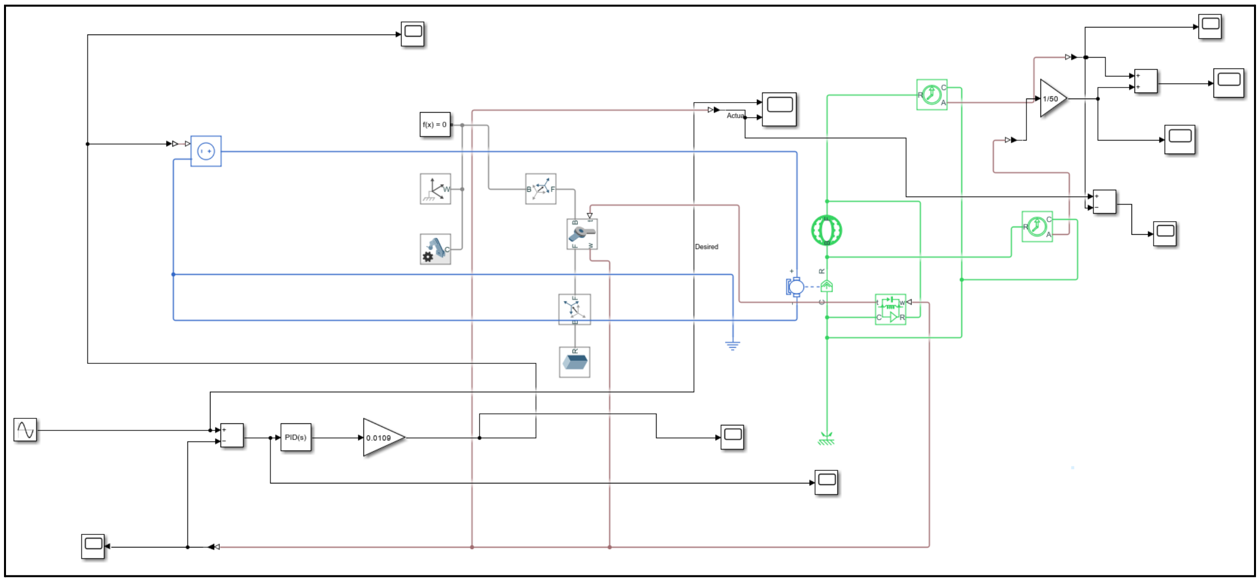


Fig. Initial Simulation

This is the initial simulation created, with a DC motor, a harmonic drive, and a link symbolizing the arm of the robot. The angular velocity from the revolute joint sensors is being controlled via the PID control loop. The PID parameters were tuned using the Transfer Function Based tuning app offered by MATLAB. The various parameter values (harmonic drive, motor, etc) were taken from the specification sheet given by AXIBO. The link is modelled as an Aluminium solid bar of dimensions 0.1*0.1*1 m^3.

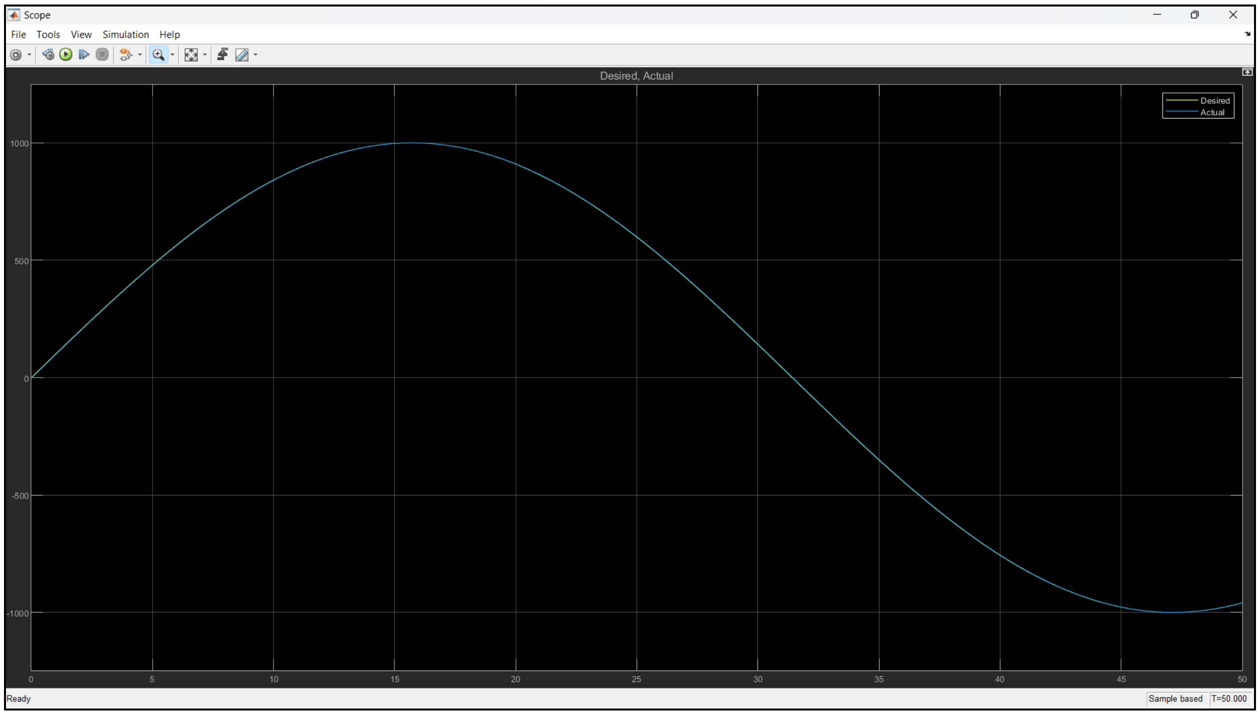


Fig. Desired and Actual Velocity Profiles

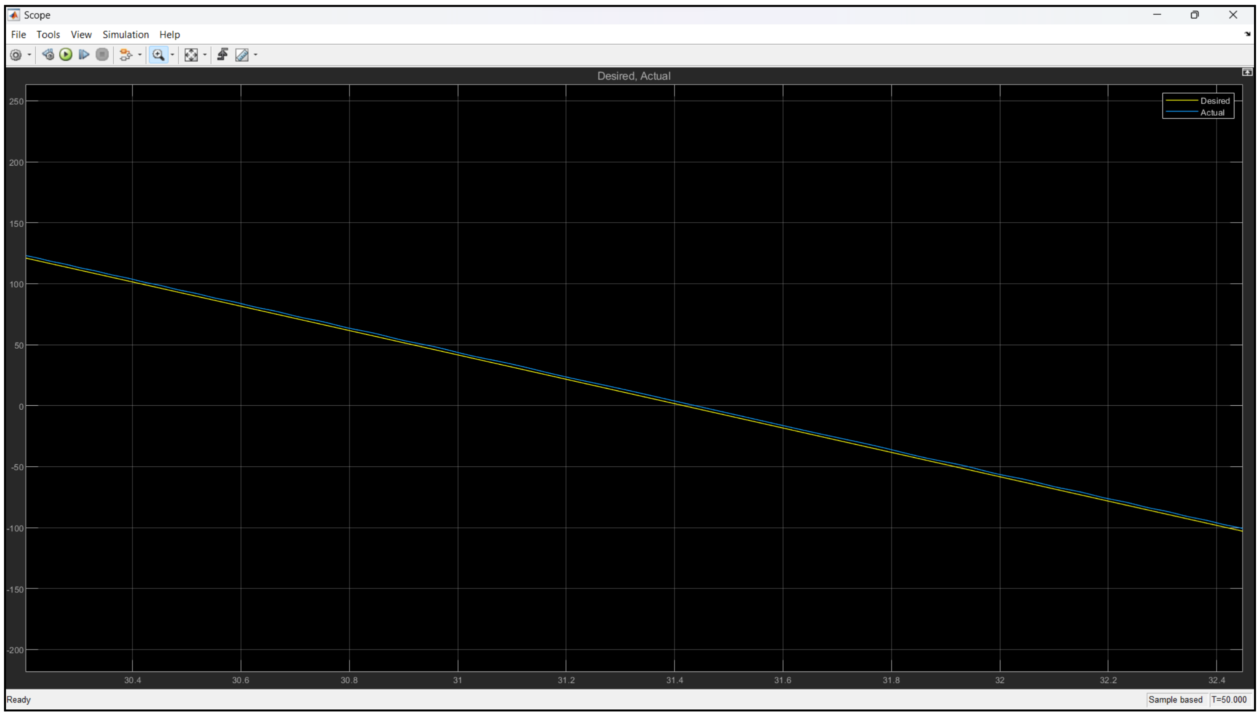


Fig. Desired and Actual Velocity Profiles (zoomed in)

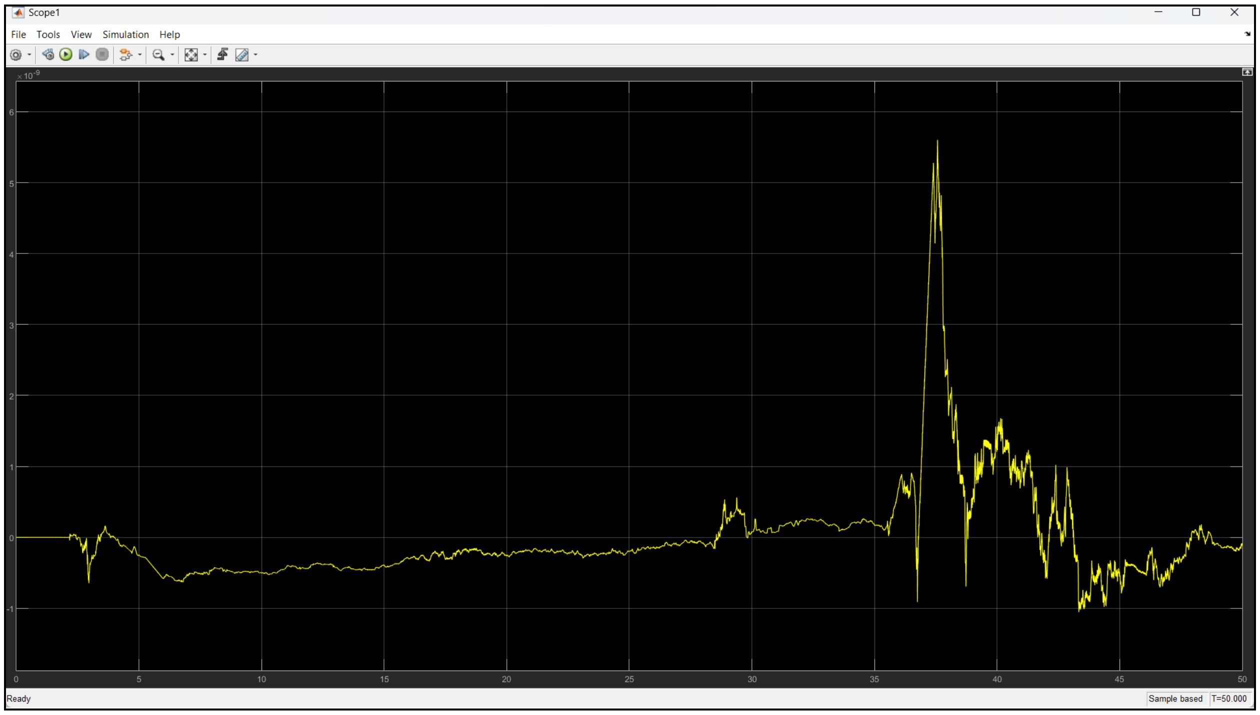


Fig. Transmission Error (of the order of 1e-9)

There were no significant vibrations observed here, hence we started studying harmonic drive as a separate component.

Harmonic Drive Simulation:-

[Here](#) is the link to the first simulation file.

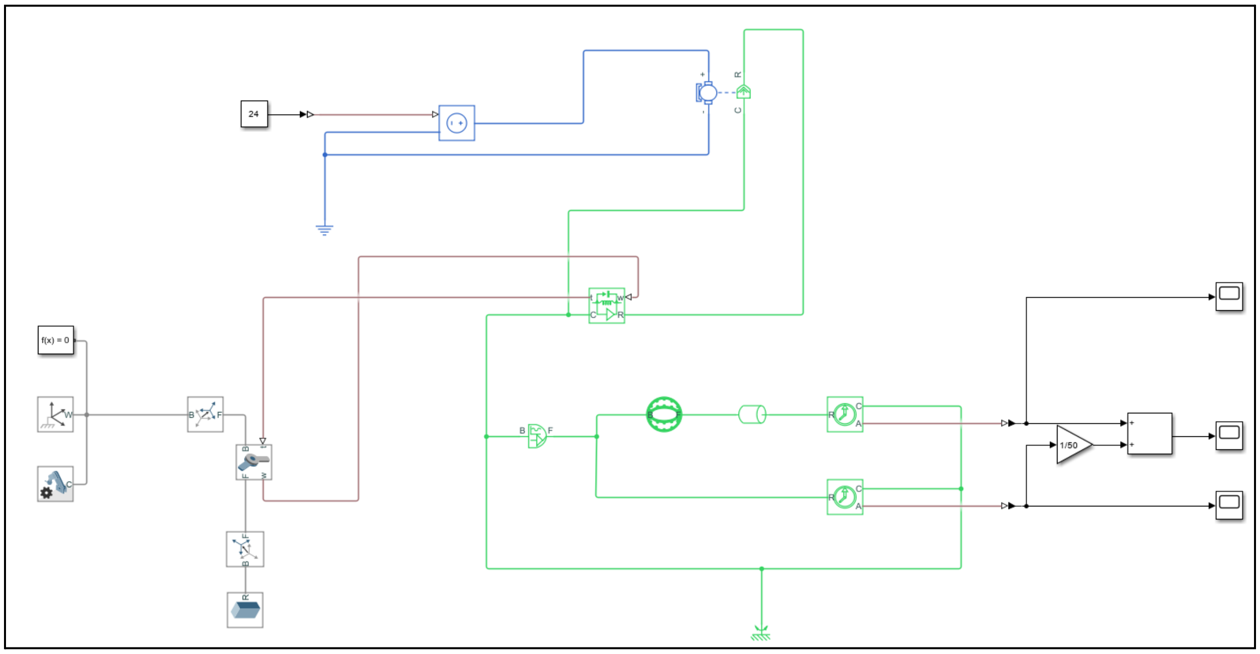


Fig. Simulation to Study Transmission Error in Harmonic Drive

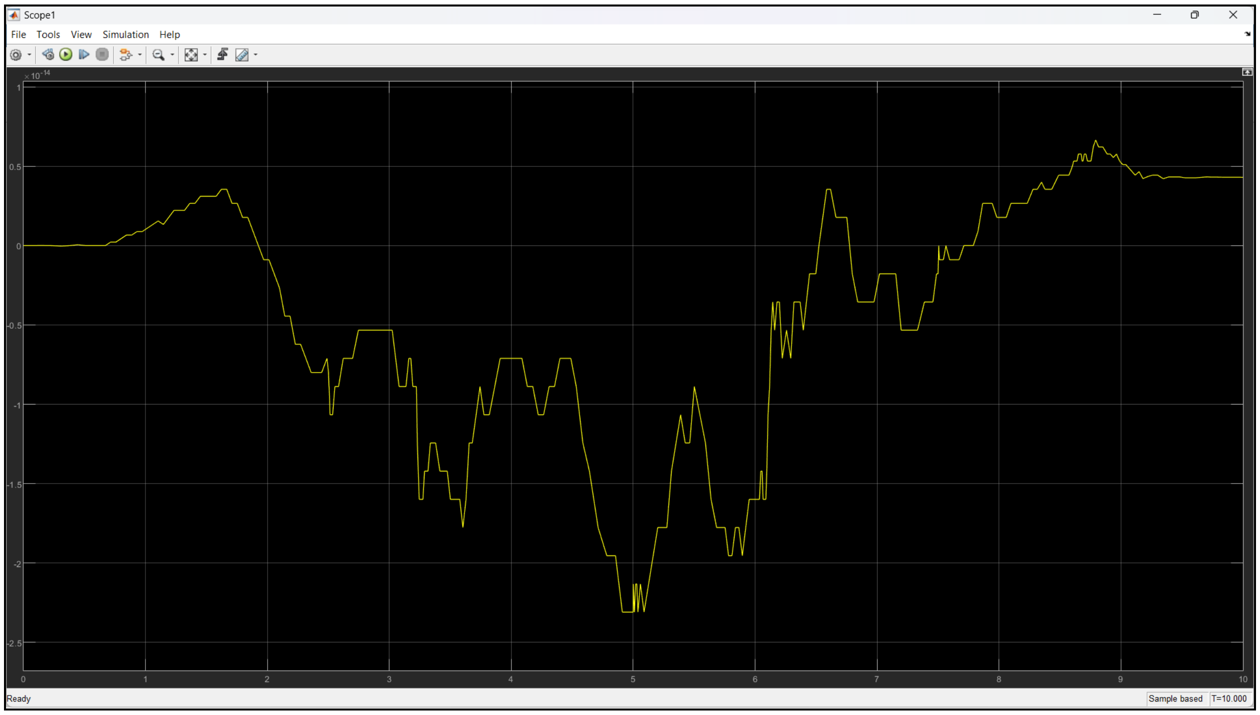


Fig. Transmission Error (of the order of 1e-14)

To study the characteristics of the harmonic drive element in Simscape, a sinusoidal rotational velocity source was fed into the harmonic drive, and the transmission error (error in the angular positions of the input and output shafts of the gear) was observed. Again, not very promising results were obtained, even with increasing the load/inertia on the drive. The precision of MATLAB is up to 16 digits, hence 1e-14 can also be the roundoff error.

After researching what elements could resemble the real-world scenario, another simulation was set up with a rotational spring in place before the harmonic drive. Here is the link to the second simulation file.

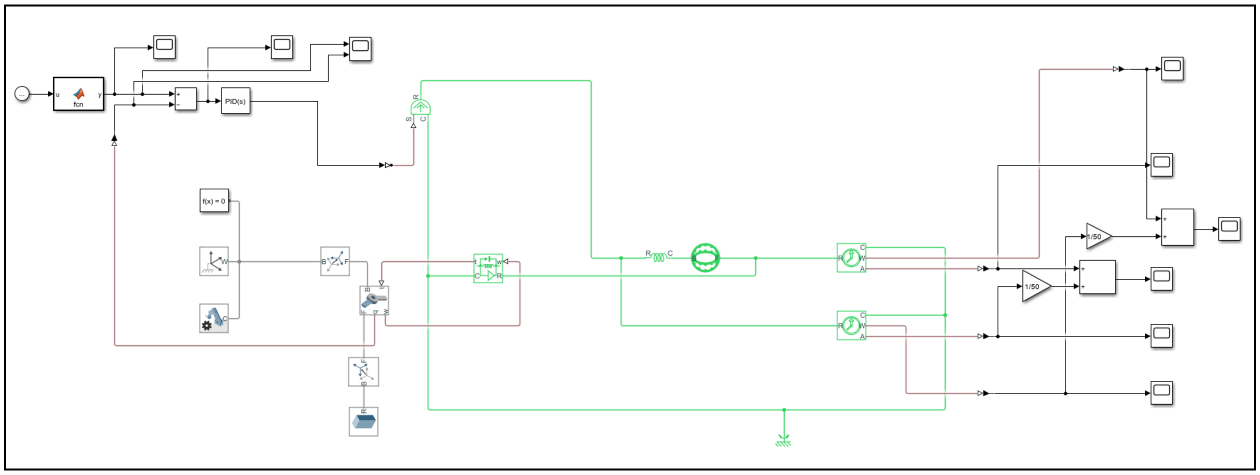


Fig. Simulation to Study Transmission Error in Harmonic Drive

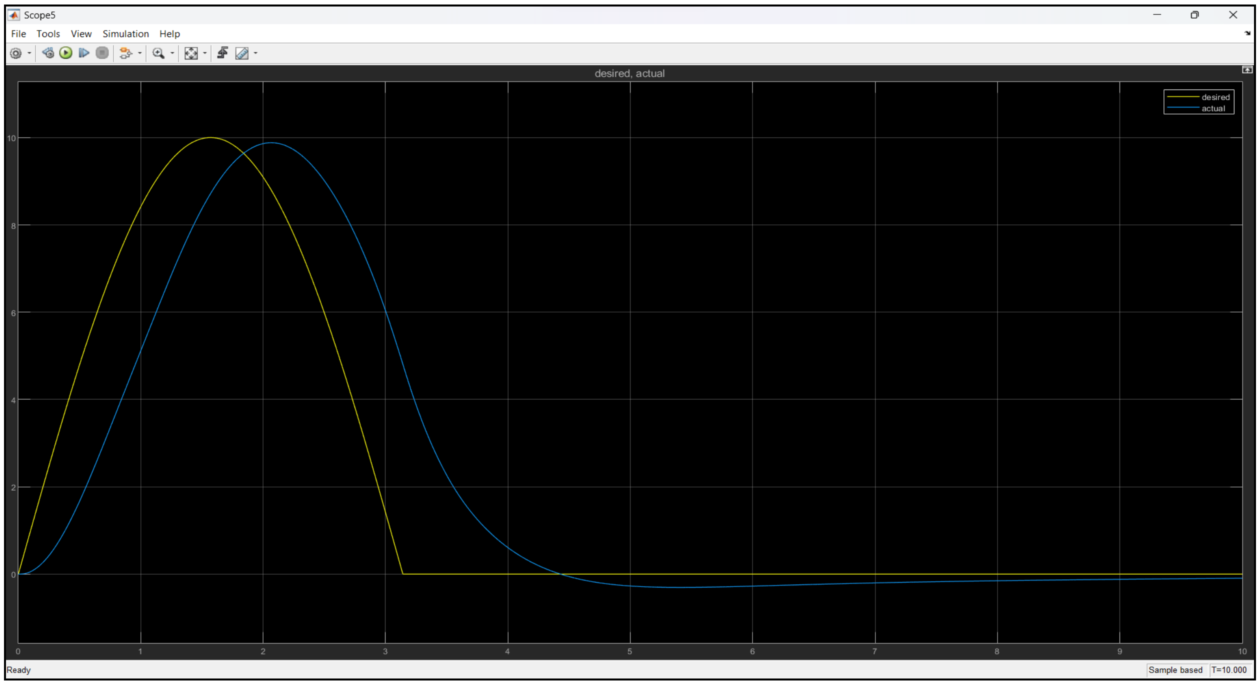


Fig. Desired and Actual Velocity Profiles

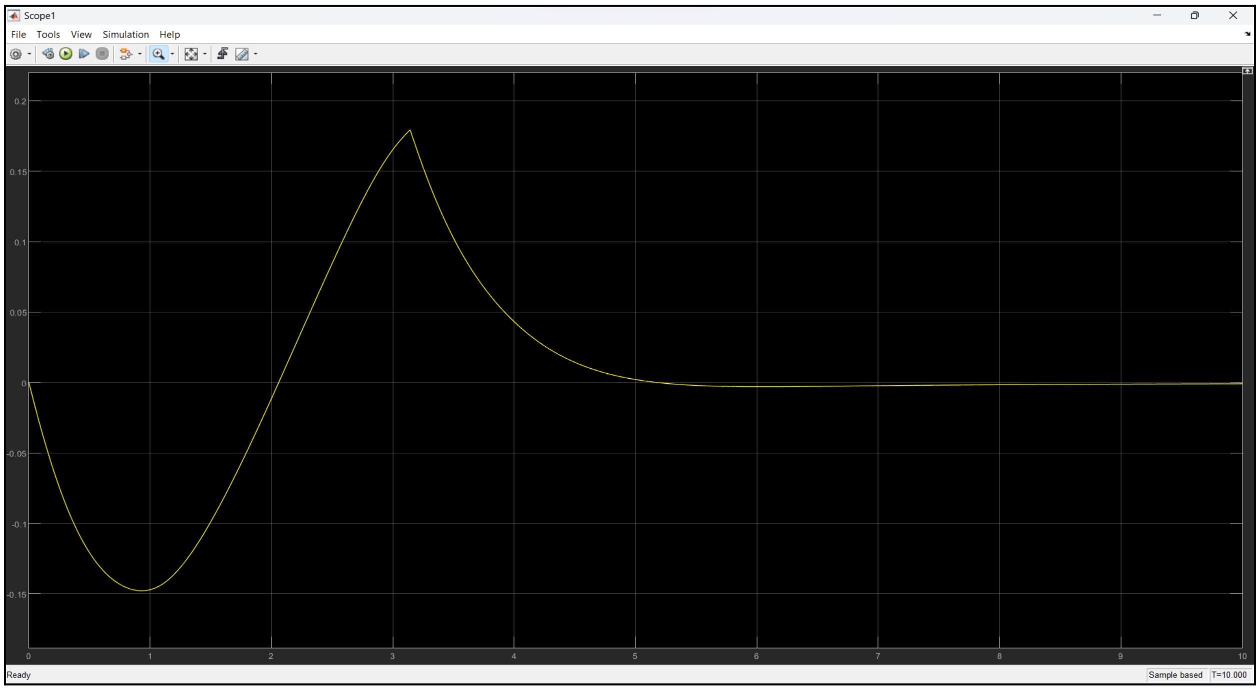


Fig. Transmission Error

A few salient features of this iteration were that the rotational spring gave promising results, with some vibration observed. But it still was not even comparable to the real-world scenario. The PID control loop was now executing angular position tracking, with a MATLAB function for the desired position. The error in the above plot is high, but that is due to the sudden change in the nature of the desired velocity profile.

Harmonic Drive Research:-

Due to the harmonic drive element in Simscape not giving satisfactory results, alternate ways to model the harmonic drive or/and its characteristics were researched.

This paper describes a model for the harmonic drive. The friction, stiffness, and transmission error, all have been considered while creating the model. The only issue is that the proposed differential equations are not giving a feasible solution, mostly due to a typo in the research paper itself. Here is the attempt made to recreate the model given in the paper.

This paper describes a model for the two-inertia axis system and a vibration suppression controller. Here is the Simulink file for the axis system and here is the controller.

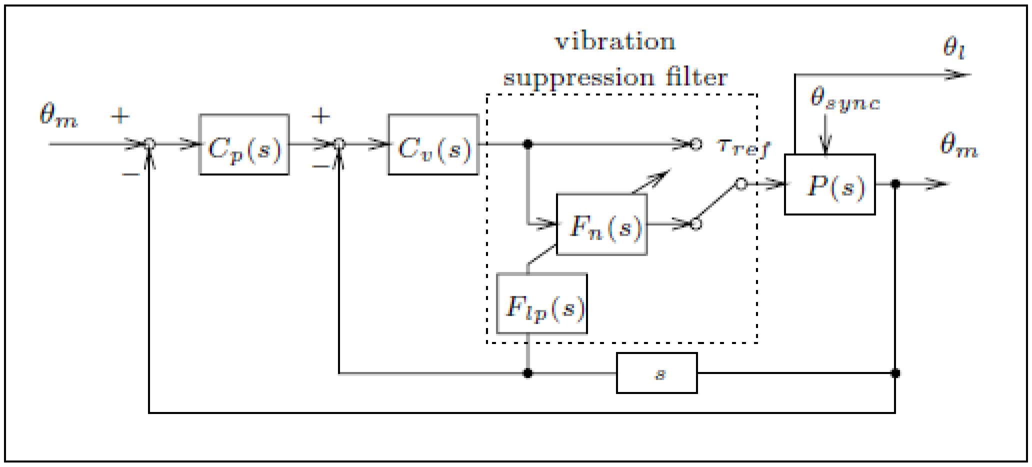


Fig. Vibration Suppression Controller as suggested in the paper

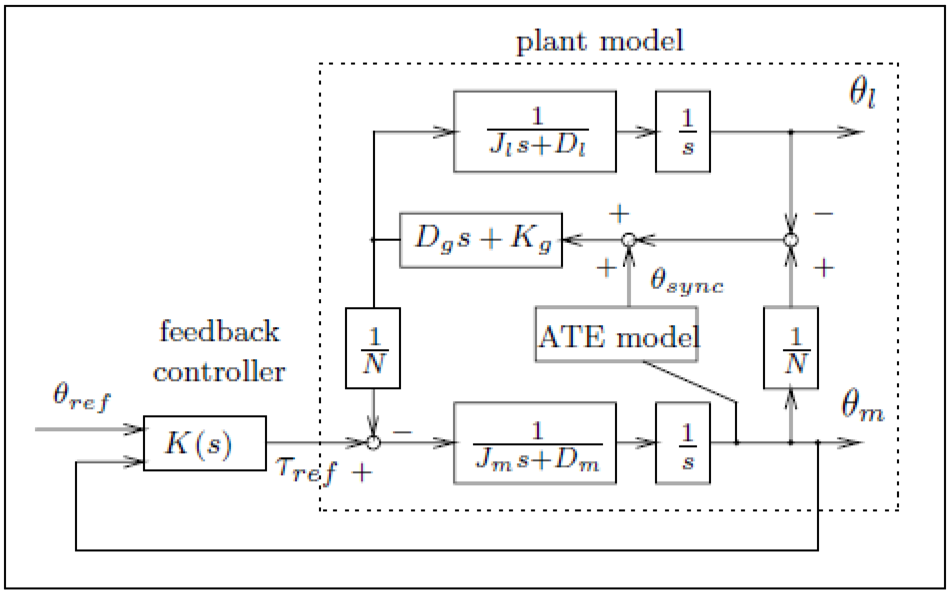


Fig. Plant model (two-inertia axis system) as suggested in the paper

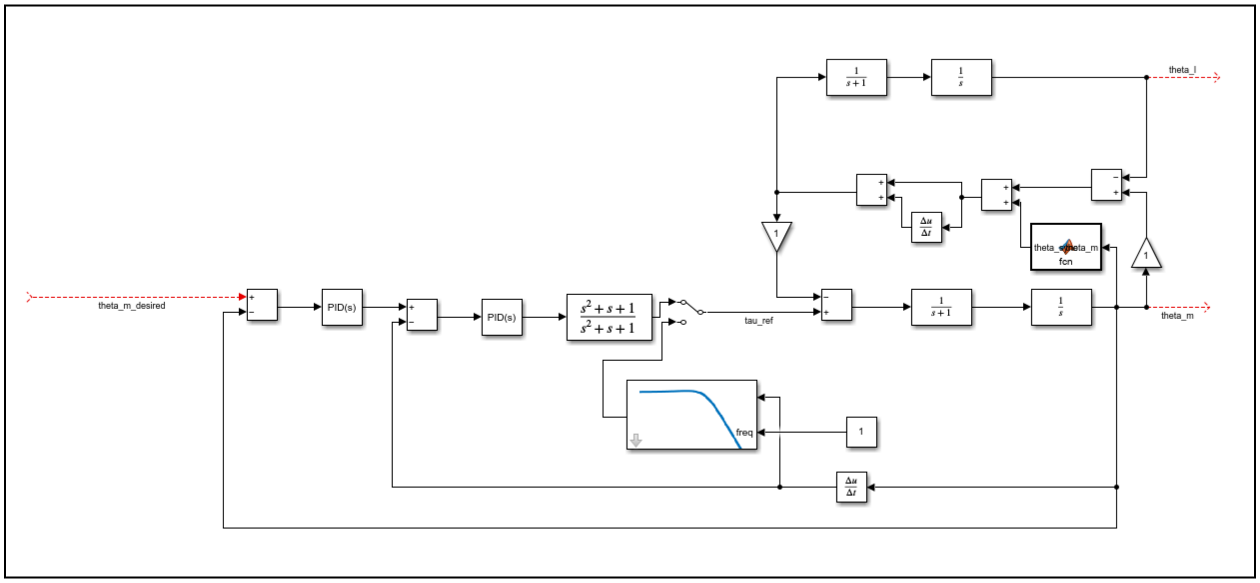


Fig. Simulink model of the controller and plant system described above

The above controller consists of a variable notch filter to reduce the vibrations, and a low-pass filter to remove sensor noise. There is a switching mechanism which controls when the notch filter is active.

Simulation-2:-

Here is the link to the simulation file.

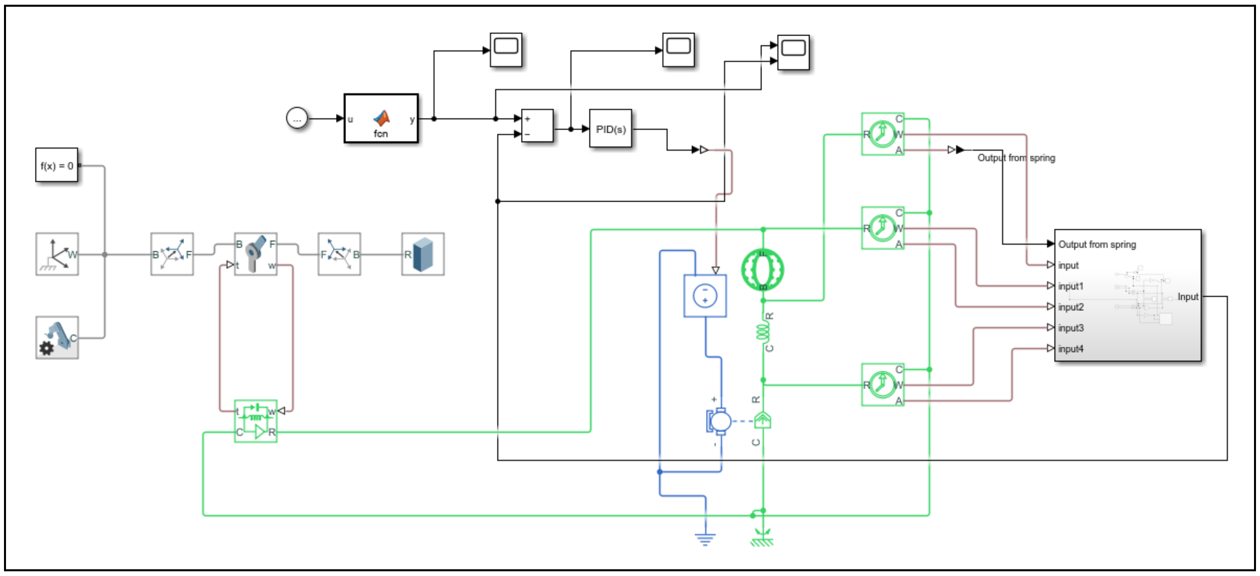


Fig. Updated simulation of DC motor and harmonic drive

A few stark differences from simulation 1 are that this employs angular position tracking with the desired position as a MATLAB function, hence can be modified easily, the rotational spring has been introduced into the system, and the position feedback is not taken from the revolute joint sensor but directly from the motor, which is closer to the real world scenario. This gave better results than the previous iteration.

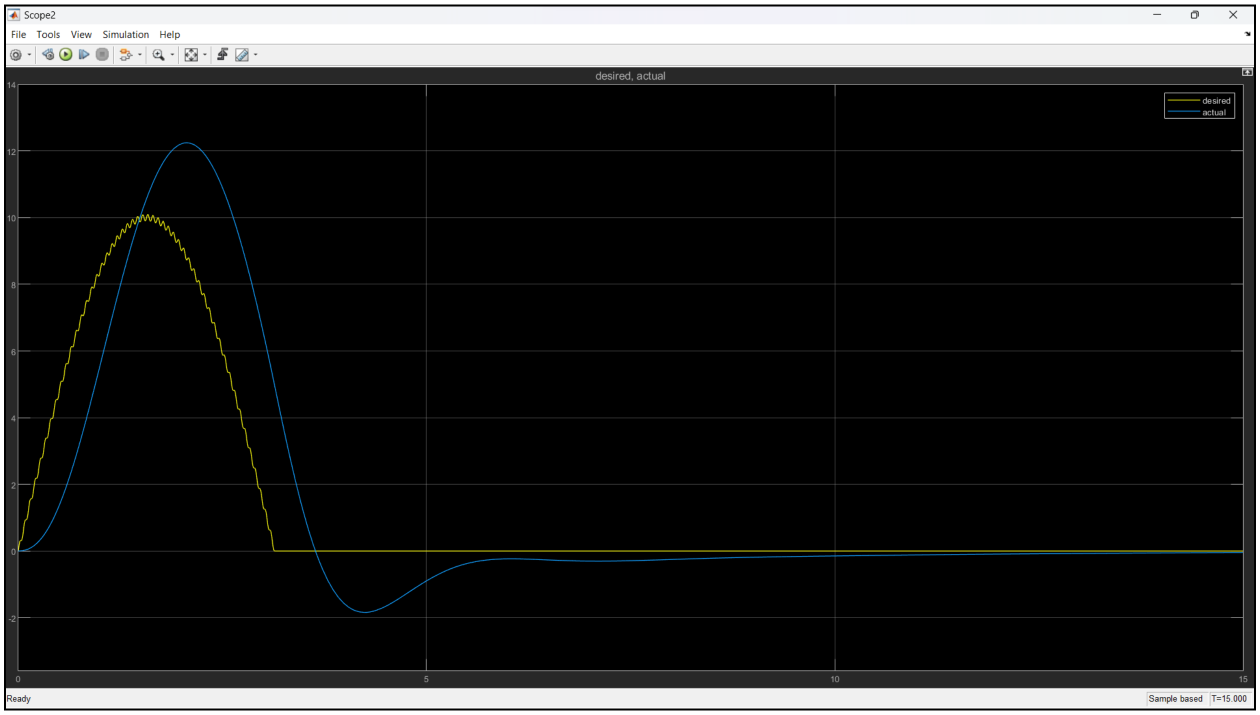


Fig. Desired and Actual Velocity Profiles

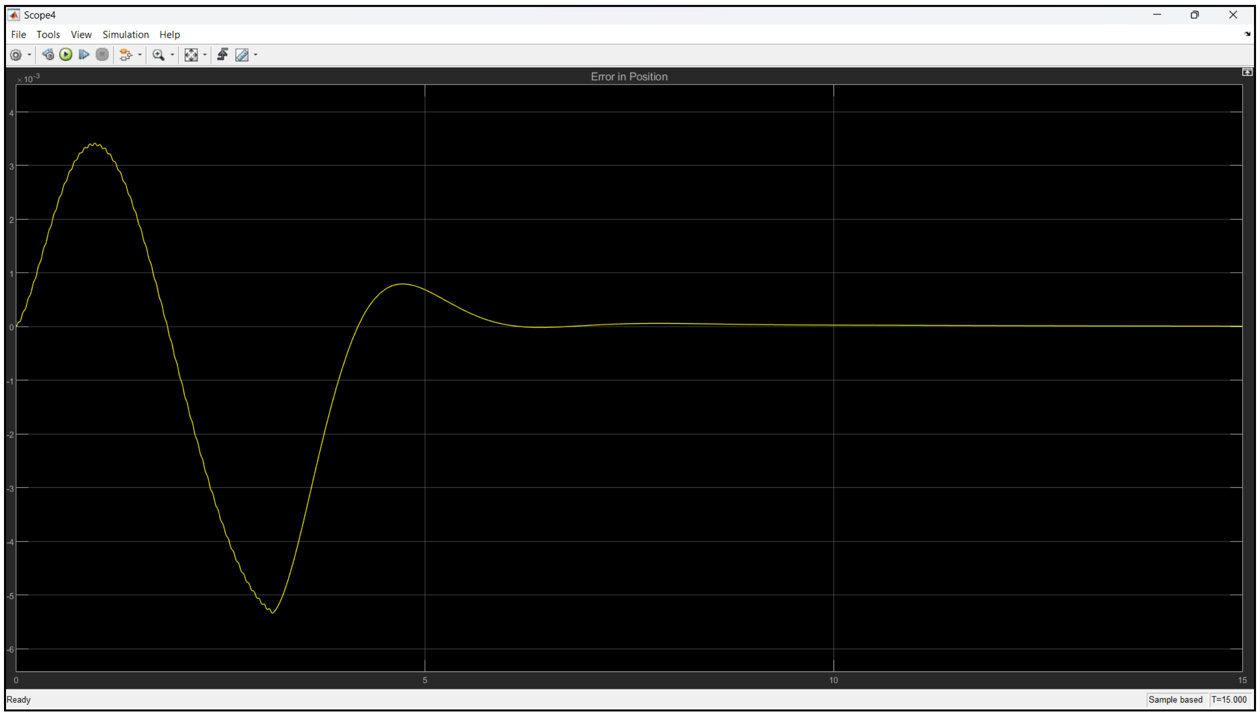


Fig. Transmission Error

We can see that the error has increased by a great margin compared to the previous simulation.

BLDC Motor Simulation:-

With the other simulation components in place, we moved to the implementation of the BLDC motor in place of the DC motor used above.

To understand the working of the BLDC motor, this model was used. This model does not use the BLDC block offered by Simscape and hence is not the most efficient one.

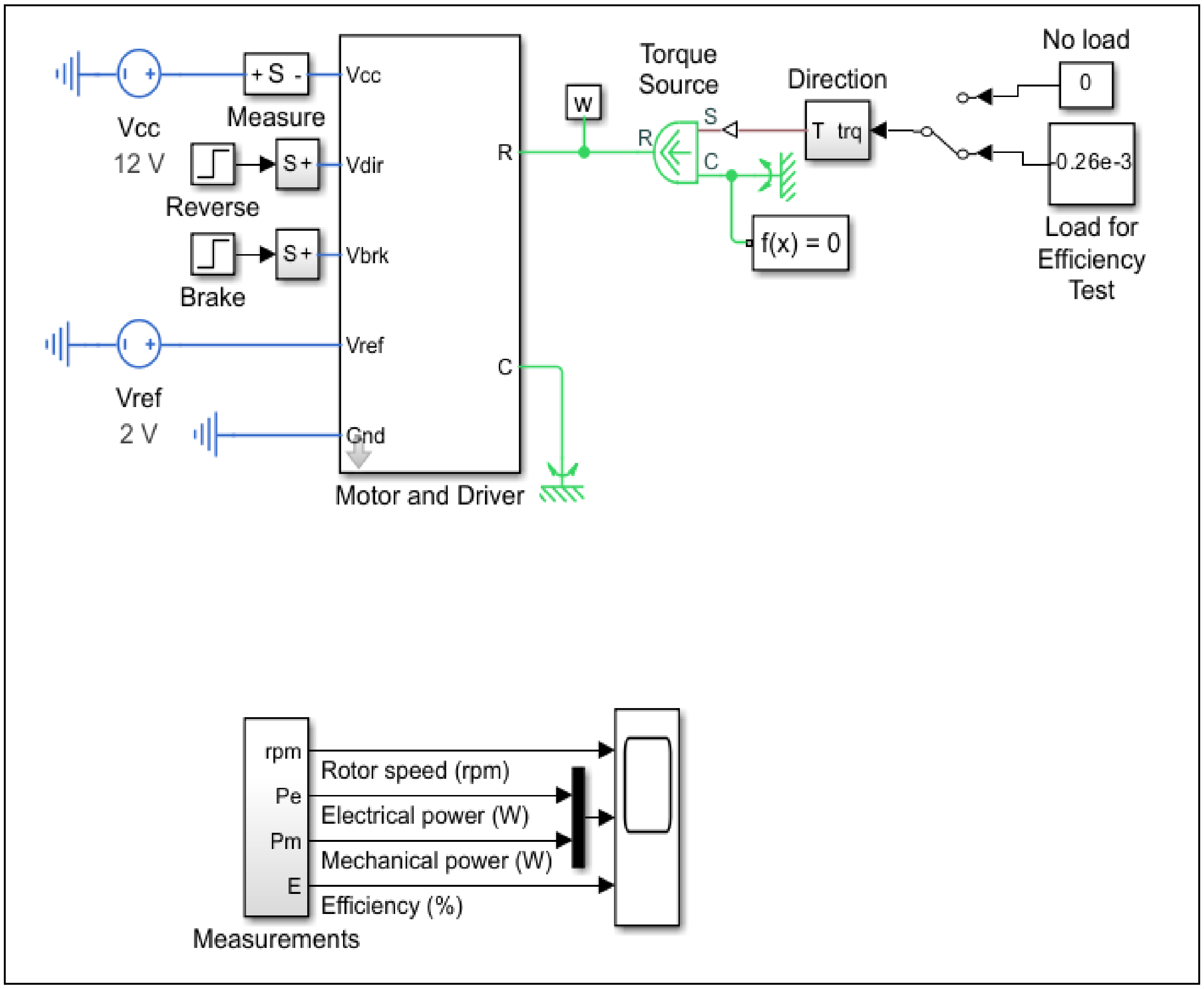


Fig. BLDC model without using BLDC block offered by Simulink

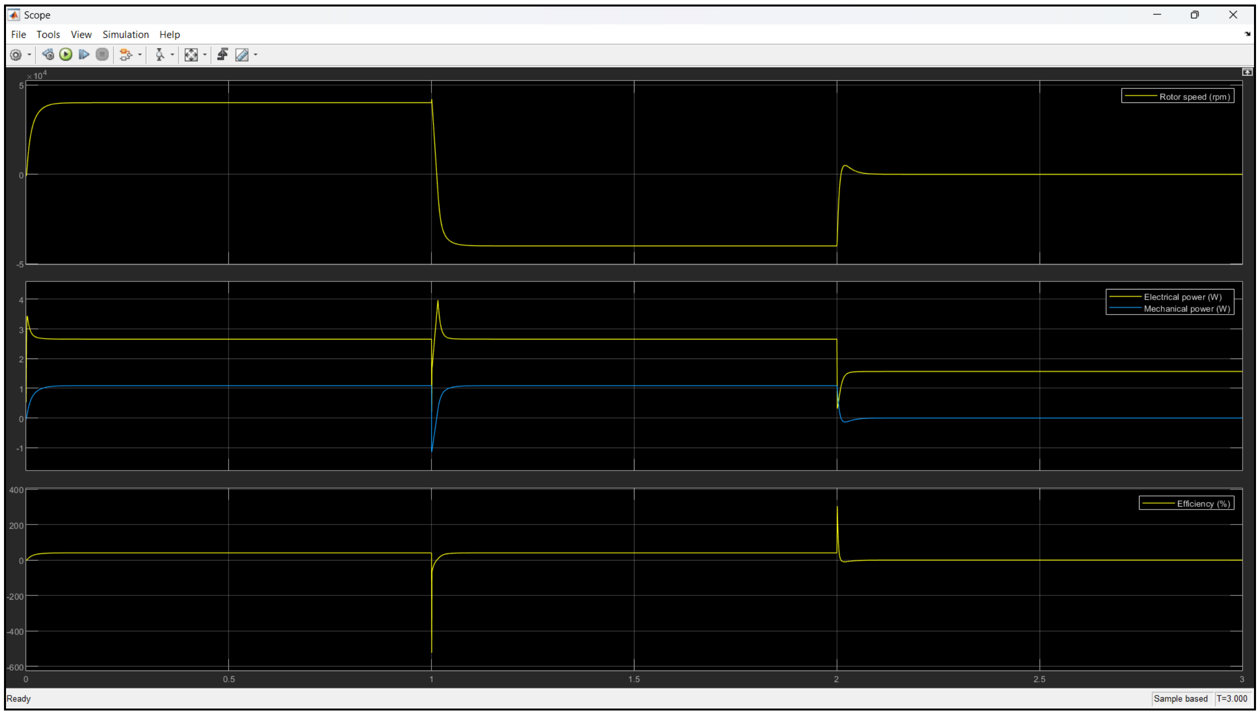


Fig. Motor characteristics of the above simulation

Here is an intermediate model of the entire setup constructed using a BLDC motor. This had some errors in it, and they were sorted out in the upcoming iterations.

The first iteration of the simulation with the BLDC motor can be found here.

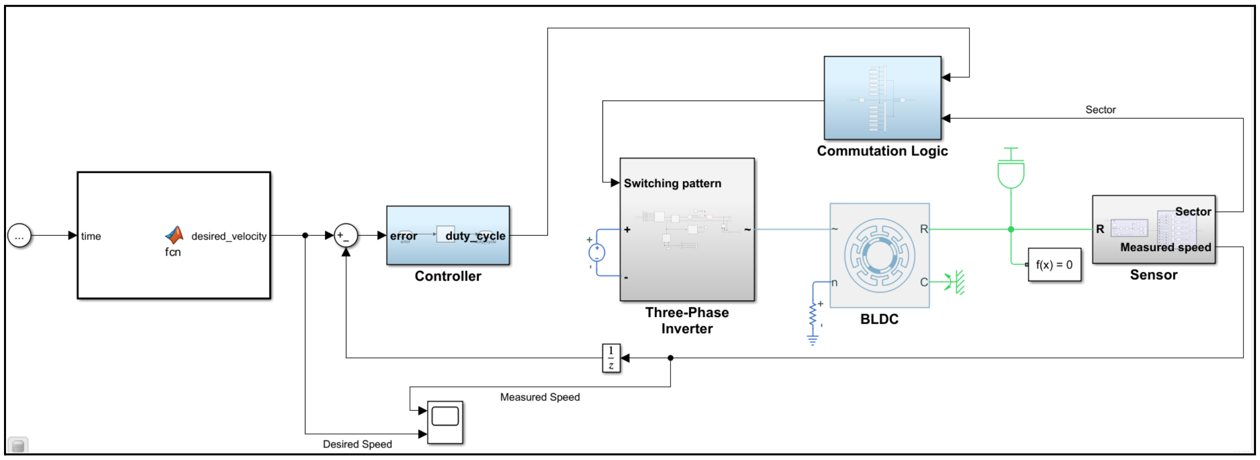


Fig. Velocity tracking with BLDC motor

A three-phase inverter is used to run the BLDC motor. A PID controller ensures velocity tracking and the desired velocity can be input as a function of time.

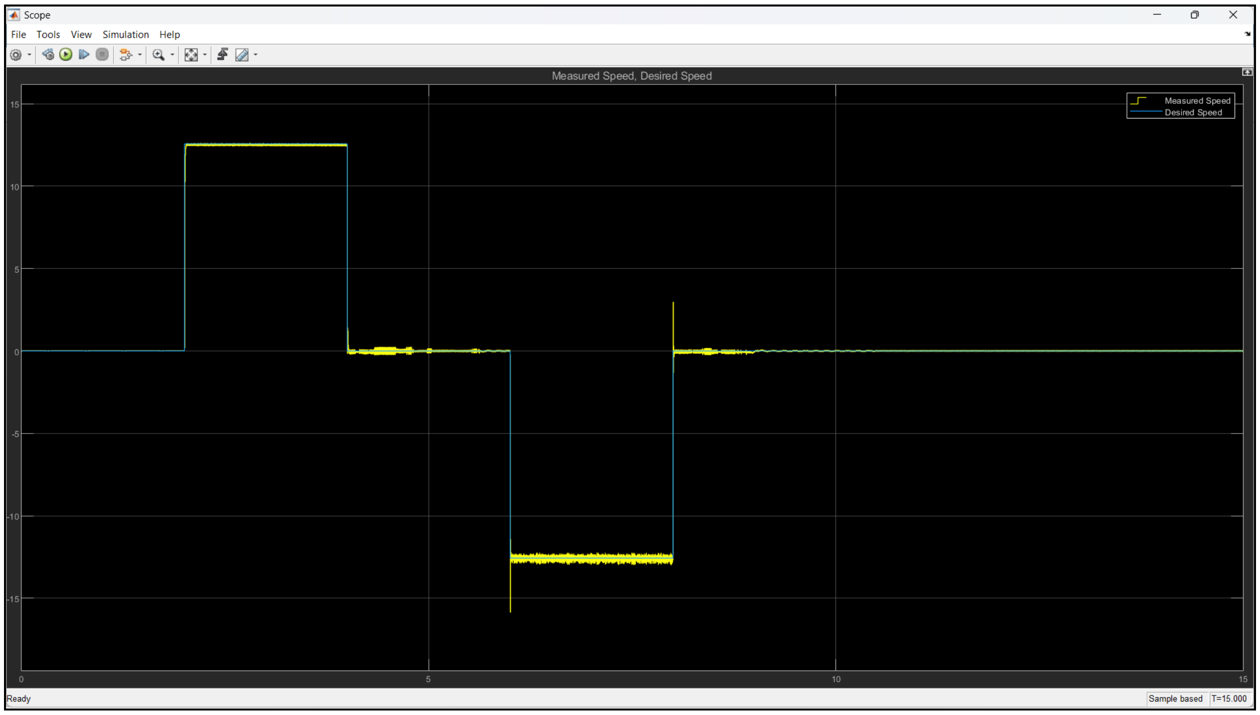


Fig. Desired and Actual Velocity Profiles

The BLDC motor gave promising results, and the next step was to add the harmonic drive and load along with the BLDC motor.

BLDC Motor & Harmonic Drive Simulation:-

Here is the Simulink file.

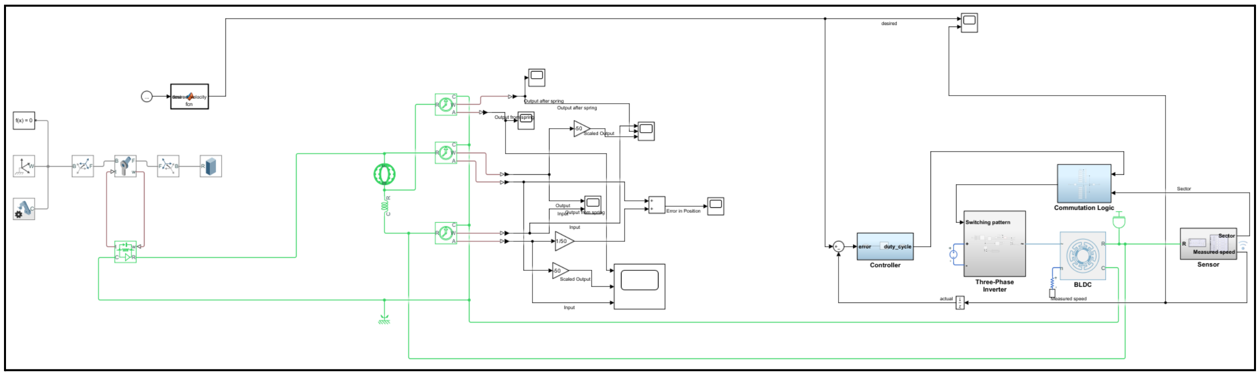


Fig. Simulation of the BLDC motor and harmonic drive system with load

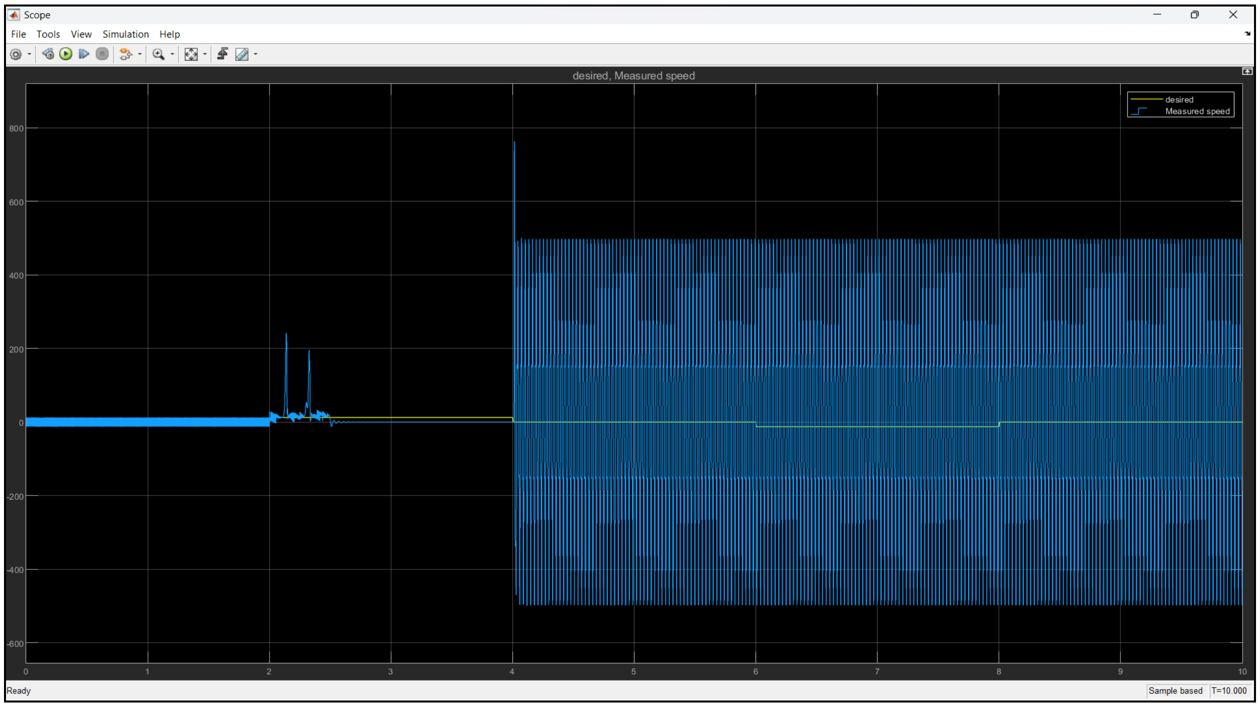


Fig. Desired and actual velocity profiles

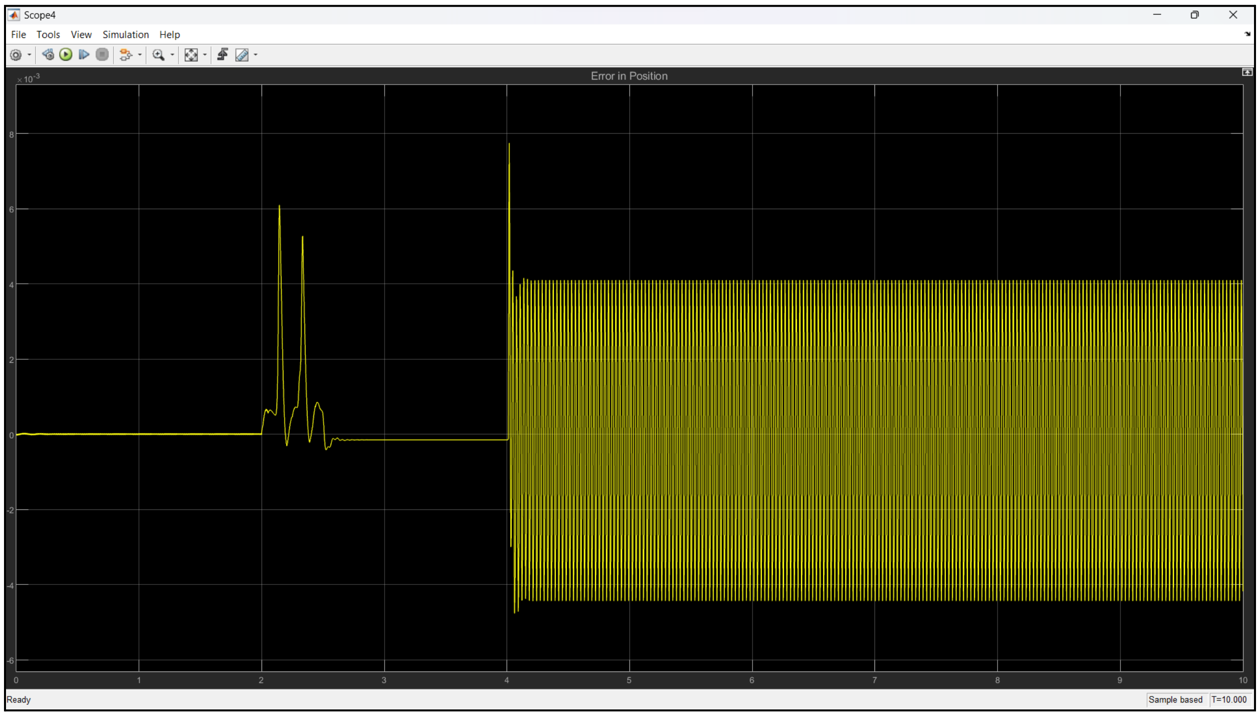


Fig. Transmission error

This combines everything used before, the BLDC motor, the harmonic drive, the rotational spring, the PID control loop, and the link configuration. But there were wild oscillations observed in the velocity profile, which were not solved. As a result, the transmission error also comes out to be very large (of the order of 1e-3).

Input Shaping for Vibration Reduction:-

To understand input shaping and the various methods used to implement it, this paper was studied. Input shaping is a method to reduce vibrations in robotic arms, especially at the end effector. Through thorough research, multi-mode Time-Varying Input Shaping Technology was established as the best method. Though the BLDC motor simulations above were also designed for input shaping, a dedicated simulation was also set up.

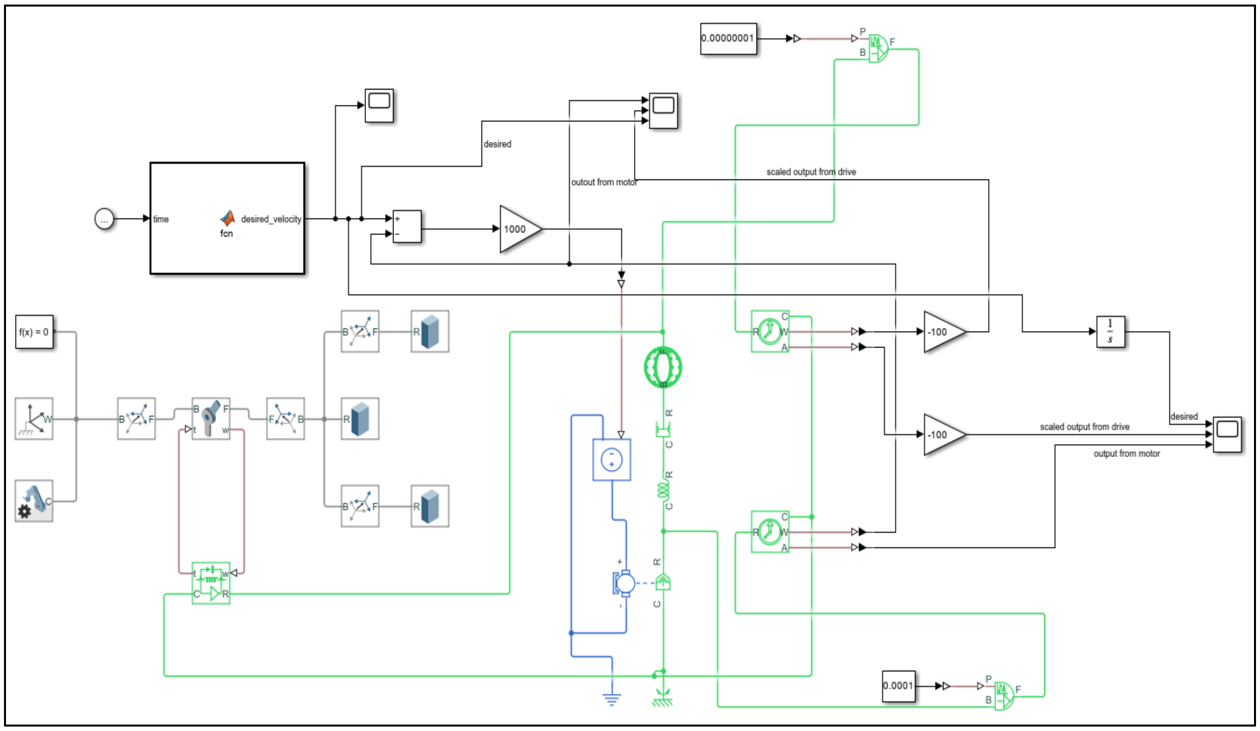


Fig. Input shaping simulation (with DC motor)

To mimic real-world scenarios, some noise was added to the velocity sensors. Along with that, a rotational damper was added. Two loads were also added at the end of the link.

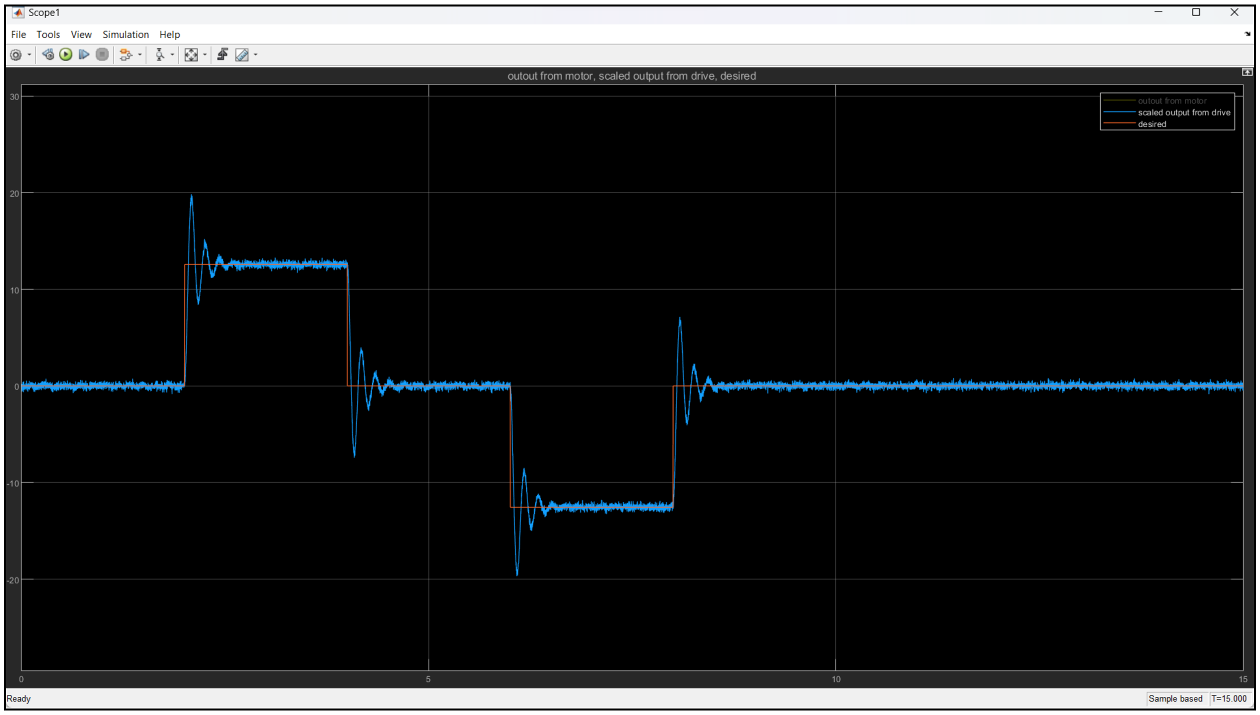


Fig. Desired and actual velocity profiles

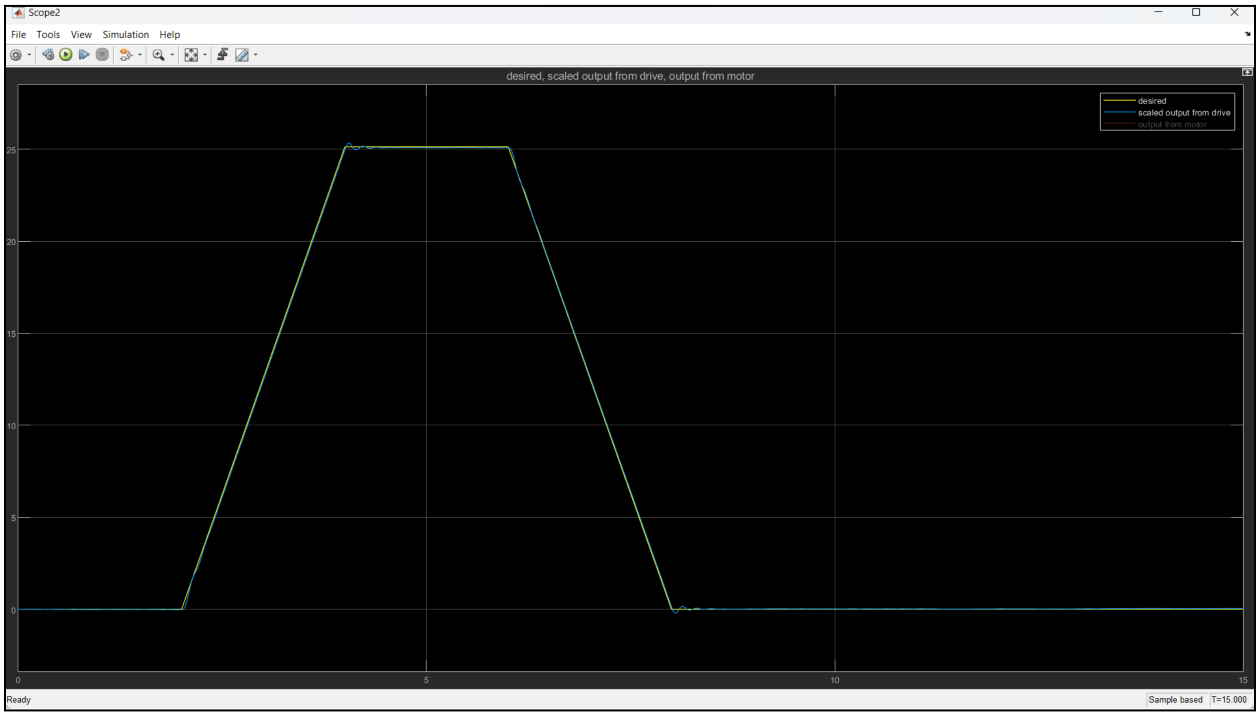


Fig. Desired and actual position profiles

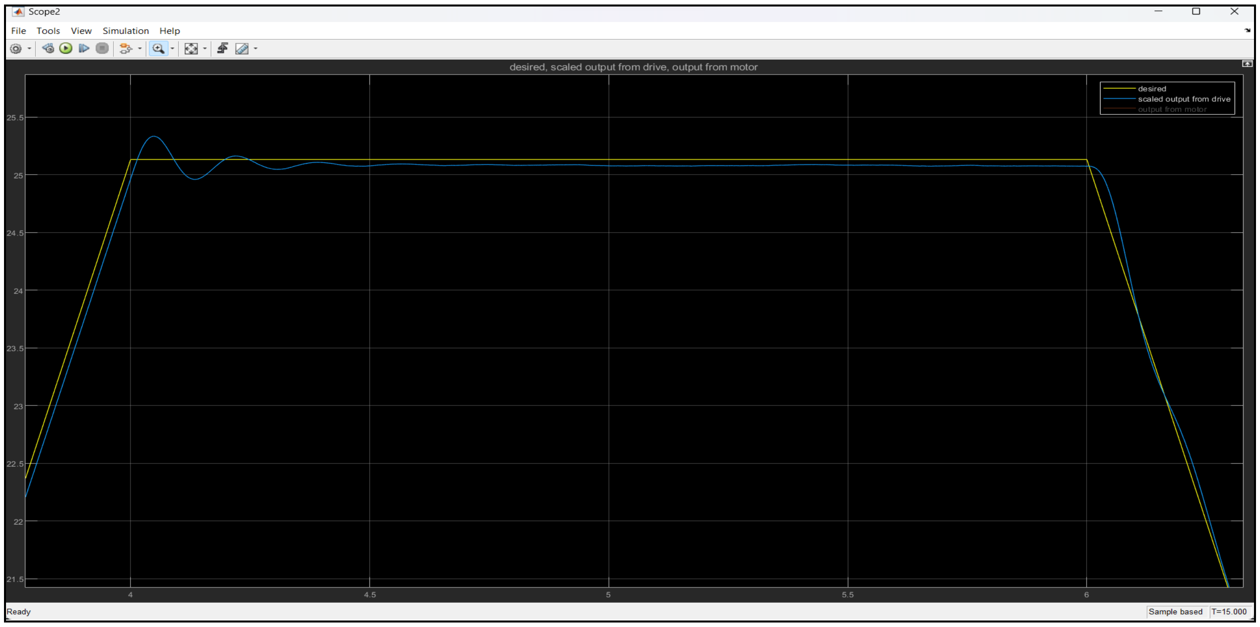


Fig. Desired and actual position profiles (zoomed in)

We can see that these results are quite similar to the real-world scenario. The overshoot and the gradual settling of the velocity profile are what are observed in real-time too.

Work on the Python App using the Streamlit library:-

Streamlit is a library in Python which can be used to create apps/websites on your local machine. One just needs to run the code in their local machine and it will redirect to their browser to open up a website.

The initial task was to create a constantly updating plot (with time). There would be two buttons, one to start the plotting and one to stop it. The code file can be found here.

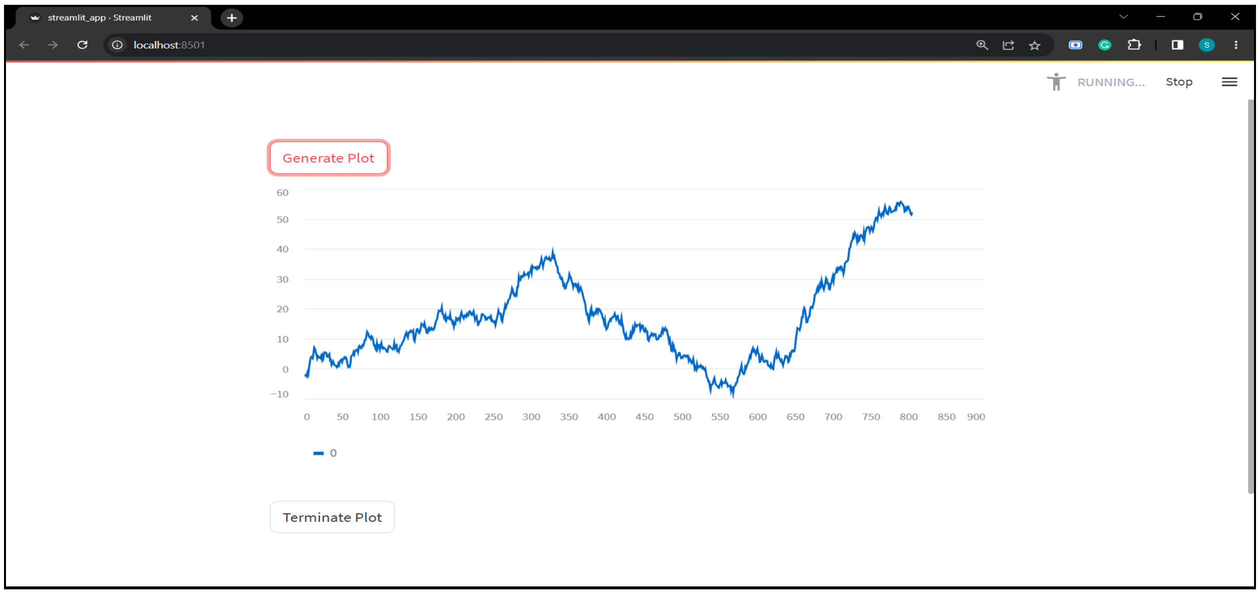
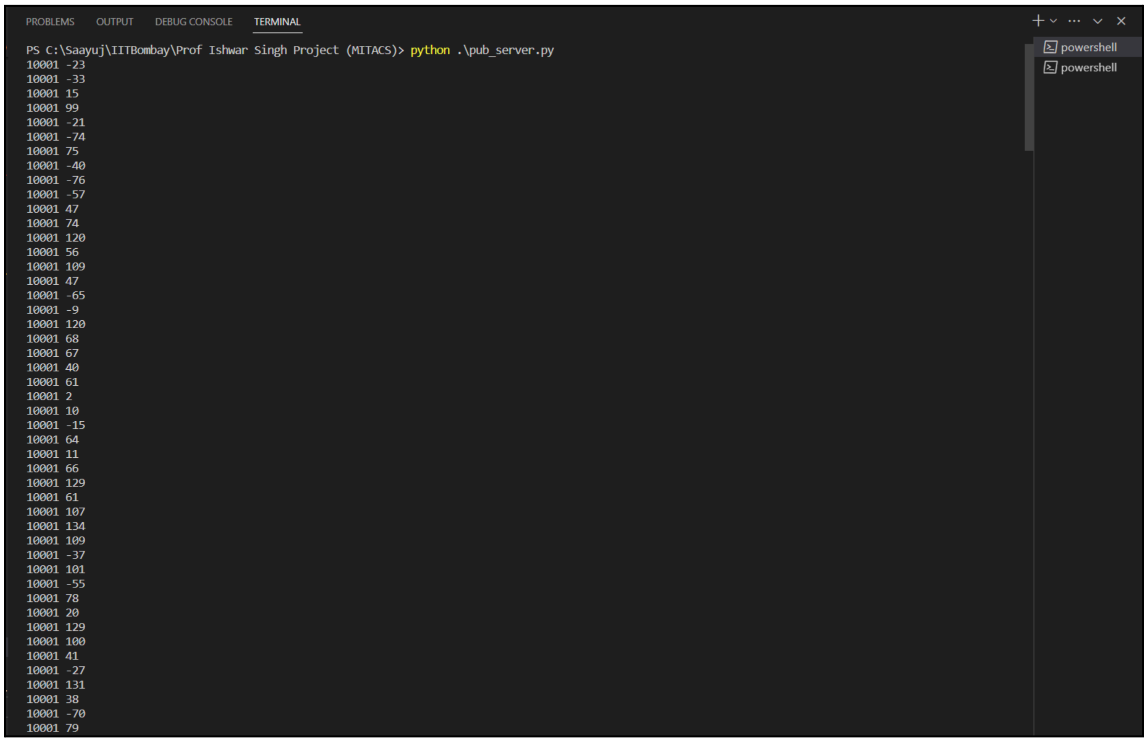


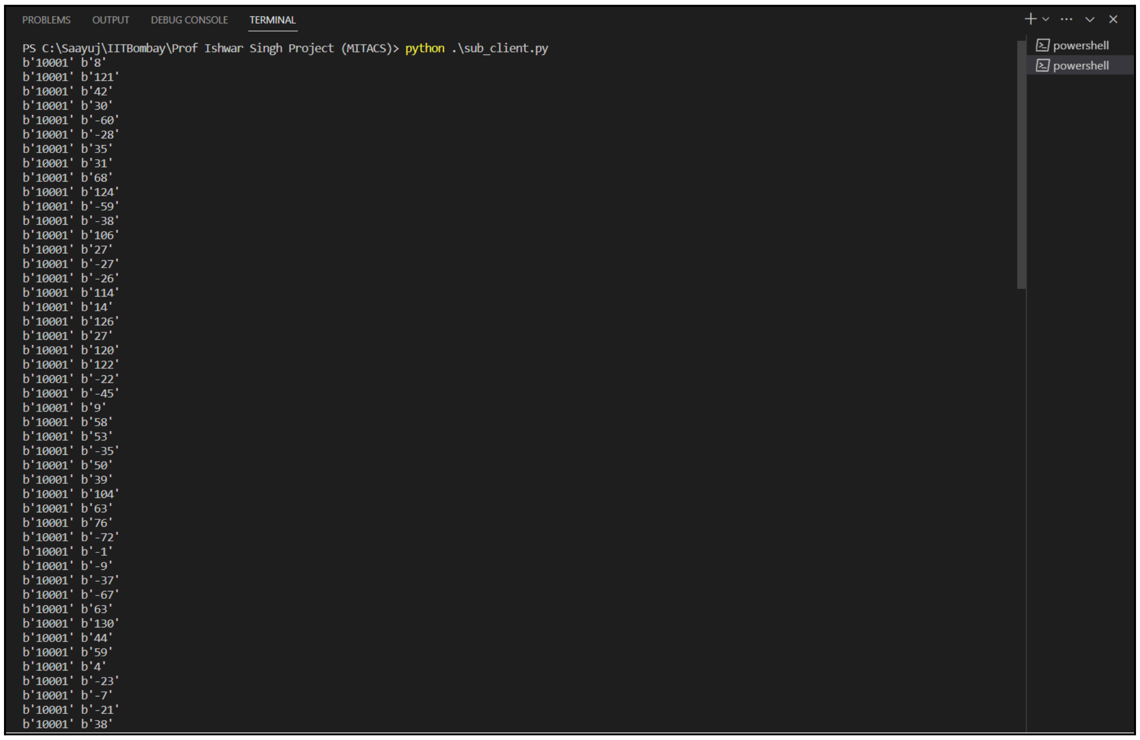
Fig. Constantly updating plot with two buttons

Next, a server for publishing and a client for subscribing was set up using the pyzmq library. The rate of publishing and the message filter can be defined in the code.



```
PS C:\Saayuj\IITBombay\Prof Ishwar Singh Project (MITACS)> python .\pub_server.py
10001 -23
10001 -33
10001 15
10001 99
10001 -21
10001 -74
10001 75
10001 -40
10001 -76
10001 -57
10001 47
10001 74
10001 120
10001 56
10001 109
10001 47
10001 -65
10001 -9
10001 120
10001 68
10001 67
10001 40
10001 61
10001 2
10001 10
10001 -15
10001 64
10001 11
10001 66
10001 129
10001 61
10001 107
10001 134
10001 109
10001 -37
10001 101
10001 -55
10001 78
10001 20
10001 129
10001 100
10001 41
10001 -27
10001 131
10001 38
10001 -70
10001 79
```

Fig. The server publishes data in this format



```
PS C:\Saayuj\IITBombay\Prof Ishwar Singh Project (MITACS)> python .\sub_client.py
b'10001' b'8'
b'10001' b'121'
b'10001' b'42'
b'10001' b'30'
b'10001' b'-60'
b'10001' b'-28'
b'10001' b'35'
b'10001' b'31'
b'10001' b'68'
b'10001' b'124'
b'10001' b'-59'
b'10001' b'-38'
b'10001' b'106'
b'10001' b'27'
b'10001' b'-27'
b'10001' b'-26'
b'10001' b'114'
b'10001' b'14'
b'10001' b'126'
b'10001' b'27'
b'10001' b'120'
b'10001' b'122'
b'10001' b'-22'
b'10001' b'-45'
b'10001' b'9'
b'10001' b'58'
b'10001' b'53'
b'10001' b'-35'
b'10001' b'50'
b'10001' b'39'
b'10001' b'104'
b'10001' b'63'
b'10001' b'76'
b'10001' b'-72'
b'10001' b'-1'
b'10001' b'-9'
b'10001' b'-37'
b'10001' b'-67'
b'10001' b'63'
b'10001' b'130'
b'10001' b'44'
b'10001' b'59'
b'10001' b'4'
b'10001' b'-23'
b'10001' b'-7'
b'10001' b'-21'
b'10001' b'38'
```

Fig. The subscriber receives data in this format

The client was modified to not only receive the messages from the publisher but also plot the received data using Streamlit. The code can be found here. Note that the pub_server.py file has to be run before this file is executed.

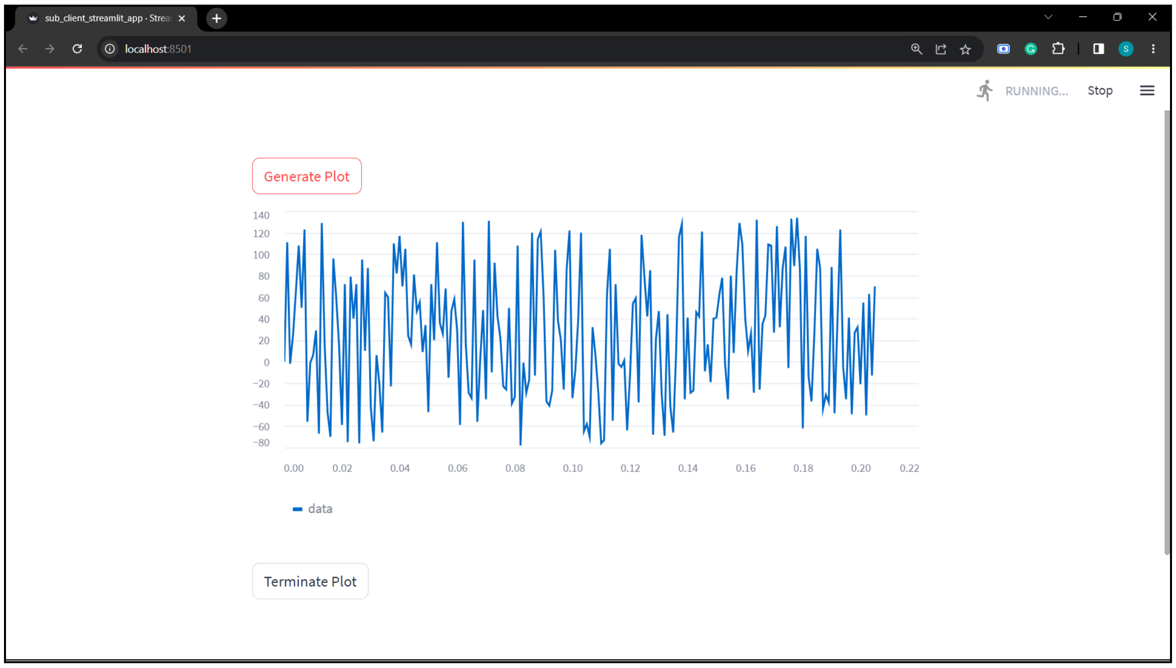
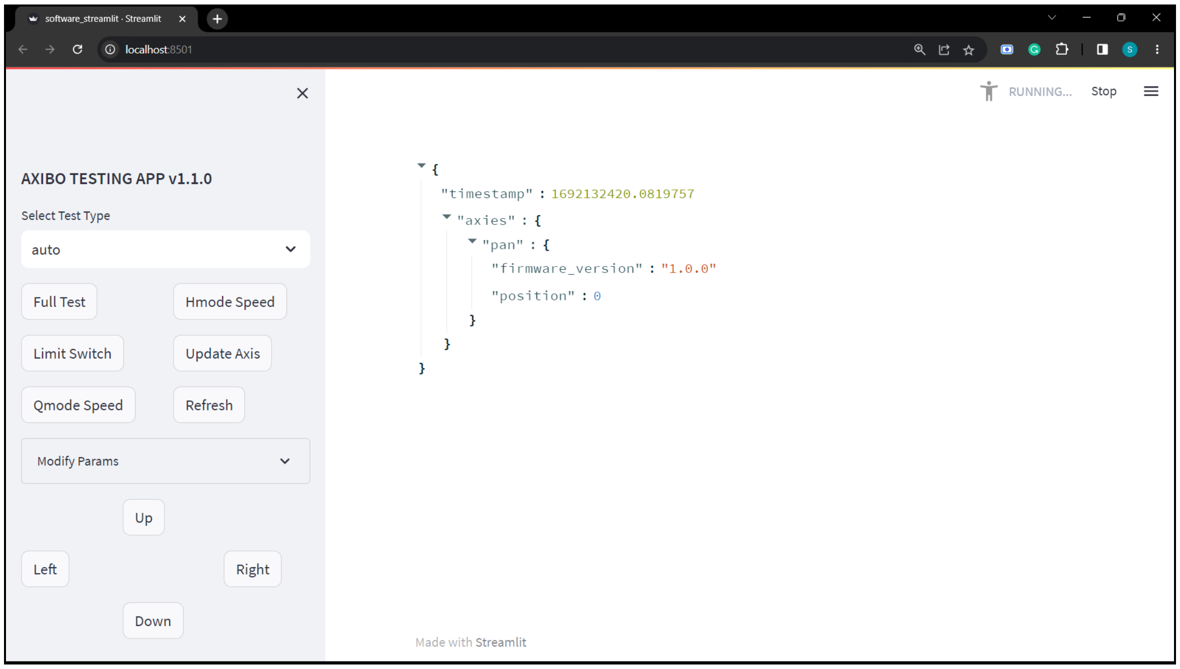


Fig. Constantly updating plot with data obtained from pub_server.py with two buttons to start and stop plotting

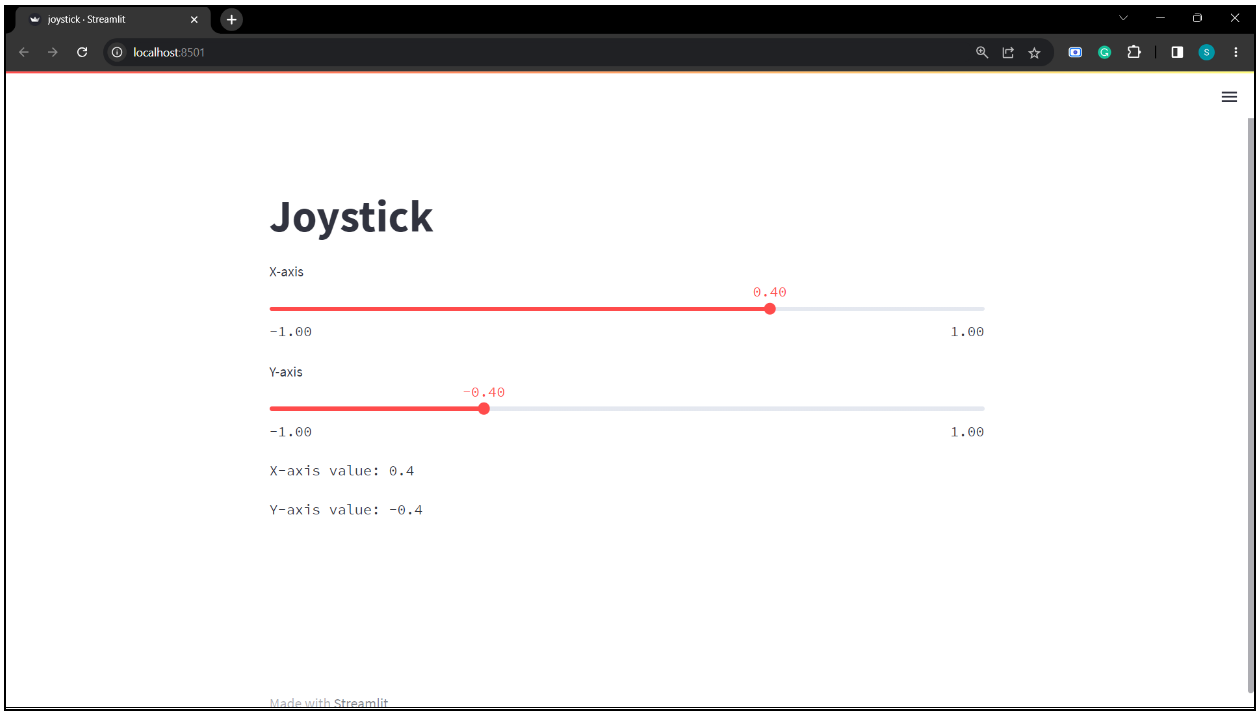
Parts of this code were shared by AXIBO. It is used to create an interface for testing the robot software. The updated code with an ever-updating axis status every 1 second can be found here.



```
{
  "timestamp" : 1692132420.0819757
  "axies" : {
    "pan" : {
      "firmware_version" : "1.0.0"
      "position" : 0
    }
  }
}
```

Fig. The testing software

This code is to create a two-dimensional joystick. Apparently, it is not possible to create a normal joystick using Streamlit. Hence, had to create one with two axes whose functioning is exactly the same as that of the normal joystick.



```
X-axis value: 0.4

Y-axis value: -0.4
```

Fig. The joystick created using Streamlit