**Title:** Simulation Exercises with the Robotics Toolbox for Matlab

**Name/Surname:** Burhan Mohayaddin

**Course:** Industrial Robotics

**Date:** 14/01/2021

## Table of Contents

# Introduction

This report presents broad explanation alongside solution to the laboratory tasks that have to be constructed and simulated through utilizing Matlab/Simulink.

# Task 1

## Straight line trajectory

The key task is to create Cartesian straight segment. So, ‘ctraj’ function has been utilized to create Cartesian space straight segment between two Cartesian points. Moreover, the tracking time of the trajectory has also been defined. However, in order to visualize the trajectory tracking of the robot manipulator, the Cartesian coordinates have to be converted to joint space coordinates. For that, ‘ikine6s’ inverse kinematics function has been utilized and the output is fed into the ‘p560.plot’ function to plot the motion of the robot.

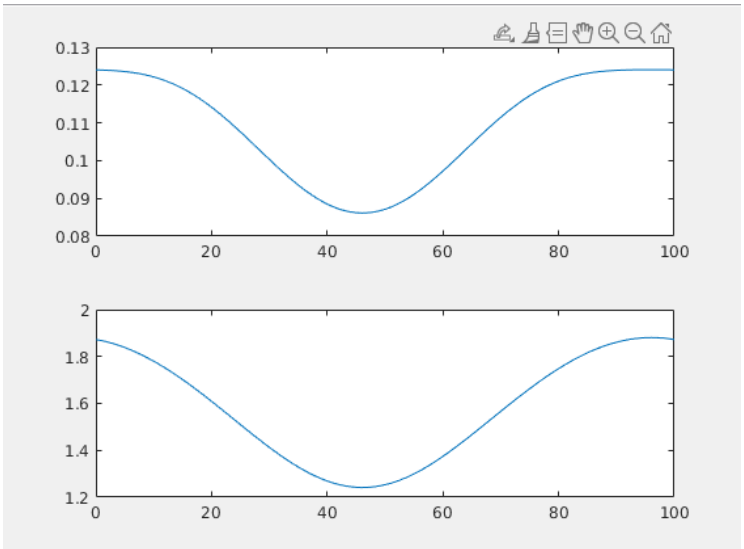
## Circular trajectory

Circular trajectory has been created using ‘circle’ block from Simulink library. The center of the circle and the radius have been specified beforehand. Moreover, inverse kinematics has been applied to get joint space coordinates in order to plot the motion of the robot manipulator.

## Manipulability

One of the most important things during control of the robot manipulator is to analyze its manipulability during its motion as it gives information about whether the robot is near singularity or in singularity. Individual rows of the manipulability matrix indicates the translational and rotational manipulability and as the value is higher the better the manipulability.  Let’s take 3 cases into consideration:

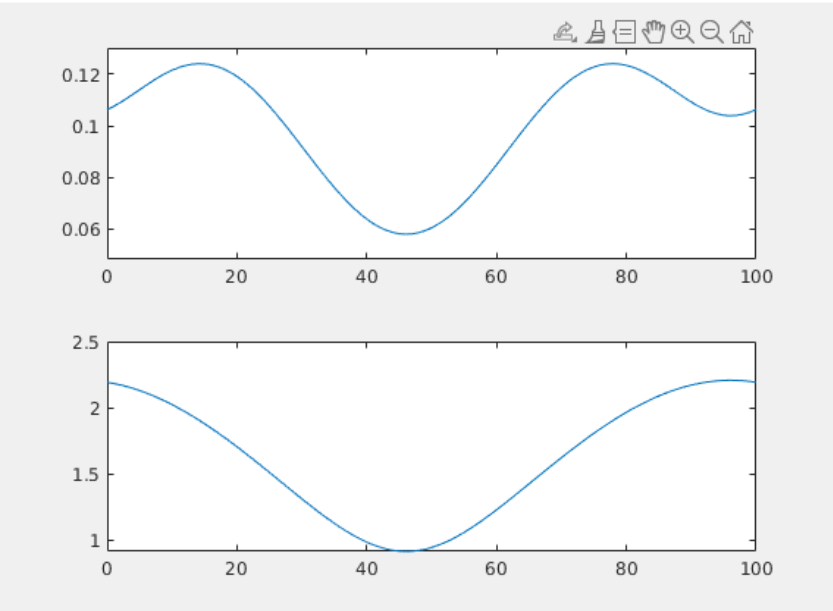
Case1: R = 0.1




As it is seen from above figure, the translational manipulability is low and it becomes even lover during middle of the trajectory. This is because robot manipulator becomes closer to its internal bound. However, rotational manipulability is better than the translational one, while it also starts to decrease during the middle of the trajectory. As the trajectory is circular the manipulability values reach their initial values.

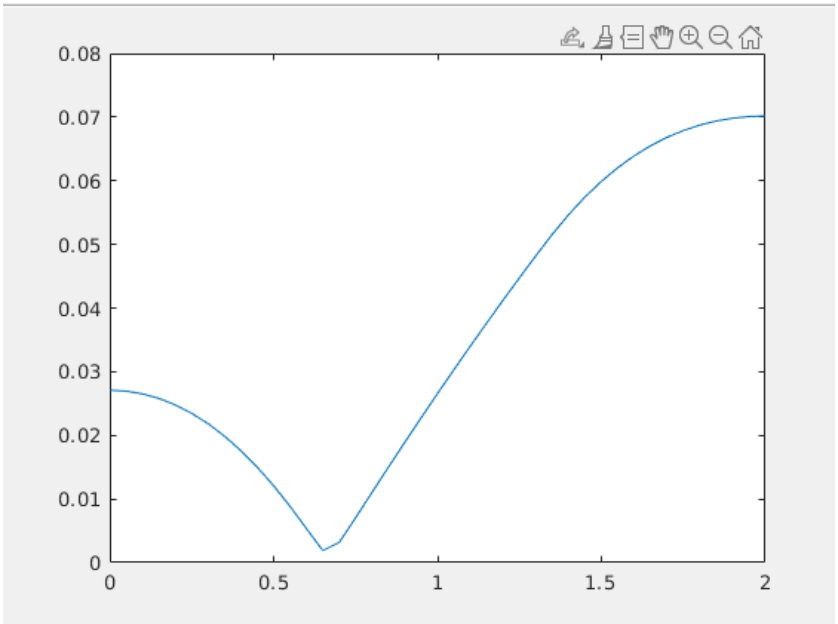
Case2: R = 0.2

As the radius is increased, the manipulator translational manipulability should decrease, as it becomes closer to its internal and outer (stretched) singulatiry.




Case3: Wrist singularity

Puma560 robot manipulator has wrist singularity when $z_3$ and $z_5$ axes are collinear. This can be observed from the below figure in which the manipulability matrix which has been obtained during straight motion of the robot manipulator, has been demonstrated.

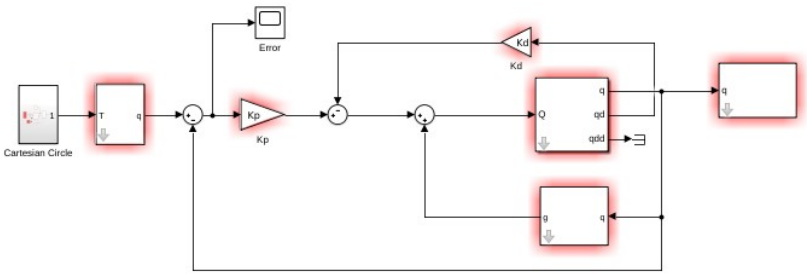



Finally, the manipulability values starts to decrease as the robot manipulator becomes closer to its internal or outer limits.

# Task 2

In the second task, PD and gravity compensation should have been applied in order to track the given trajectory. In this control method, derivative term improves system time response and acts like a damper. The global asymptotic stability is ensured with positive $K_p$ and $K_d$.

## Simulink graph

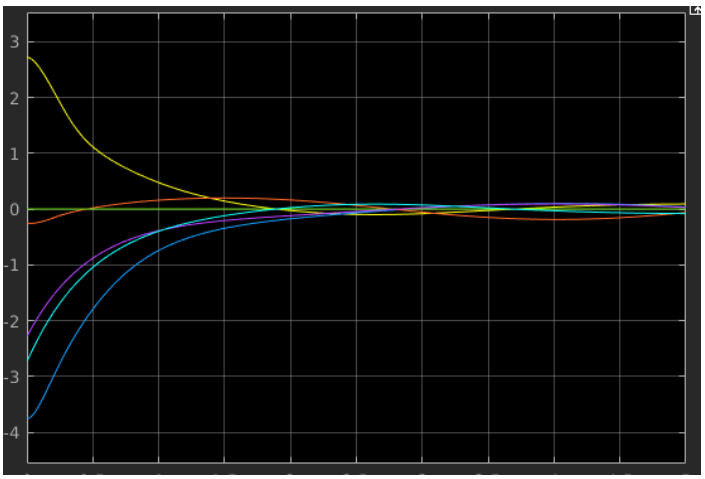



## Results

Choosing $K_p$ and $K_d$ is very important. As it is known, the steady-state error directly depends on the value of $K_p$. So, the steady-state error can be decreased by increasing the value of $K_p$. However, it

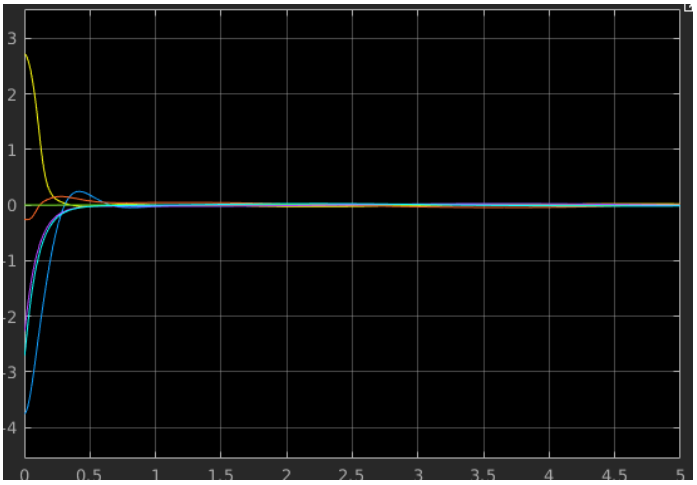
cannot be increased arbitrarily high, because the closed loop system can become oscillatory and unstable. Let's analyze below three cases:

Case1: $K_p = 100$, $K_d = 50$



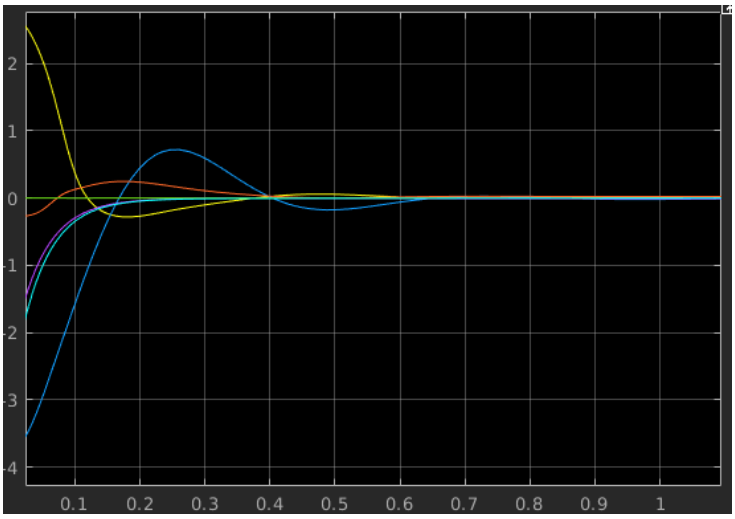

Above figure illustrates the joint errors during the tracking of the trajectory and the steady-state error can be observed clearly.

Case2: $K_p = 500$, $K_d = 50$



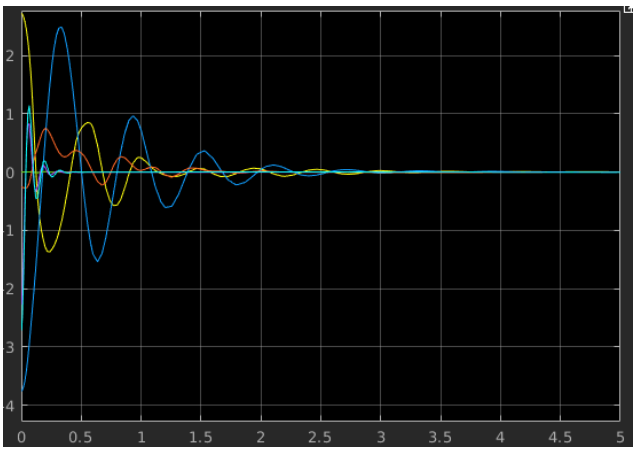

As it can be seen from the above figure that the steady-state error decreased significantly, while some overshoot can be observed.

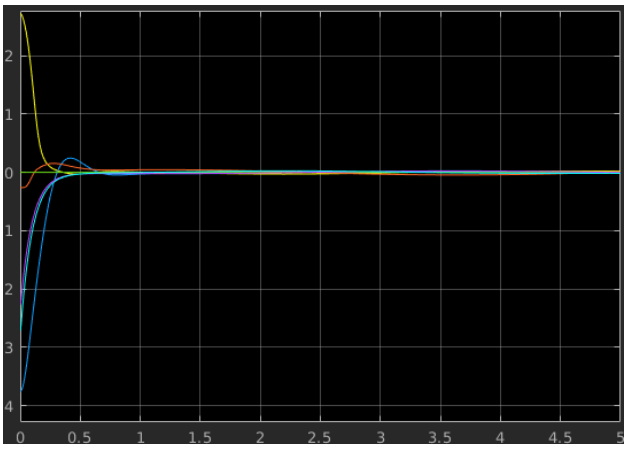
Case3: $K_p = 1000$, $K_d = 50$




As it can be seen from above figure, very high proportional gain can cause oscillations in the system.

In terms of $K_d$, it not only dampens the oscillations in the signal, but helps to reach the steady-state value faster. Let's analyze below cases:

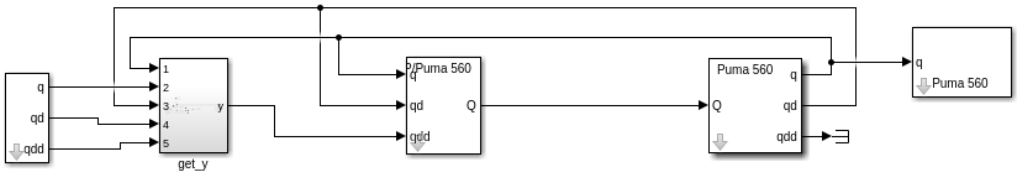



On the left figure, the derivative gain, $K_d$, has been chosen small, namely $K_d = 5$. Due to this reason, severe oscillations can be observed in the error signals. Moreover, it takes more time to reach steady-state than the figure on the right in which $K_d = 50$. As the derivative gain is high, the oscillations have been dampened out pretty faster.

# Task3

## Simulink graph

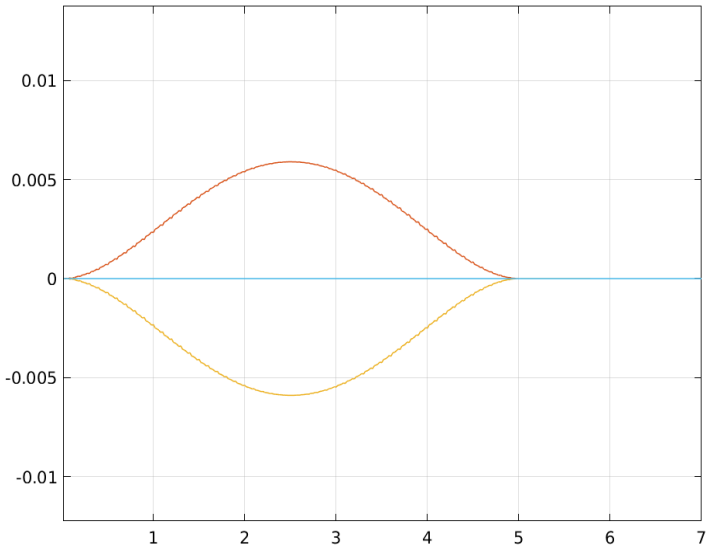



## Results

Inverse dynamics control is one of the important centralized control method in which firstly, the nonlinear robot dynamics is linearized and decoupled through nonlinear state feedback linearization and then, stabilizing linear control (PD) is applied to the obtained system. However, it requires the perfect knowledge of dynamics model of the manipulator which is not realizable for many cases. So, the nonlinear state feedback linearization doesn't work as intended and nonlinear and, coupled system has been obtained. This cause increase in steady-state error and instabilities during tracking of the trajectory.

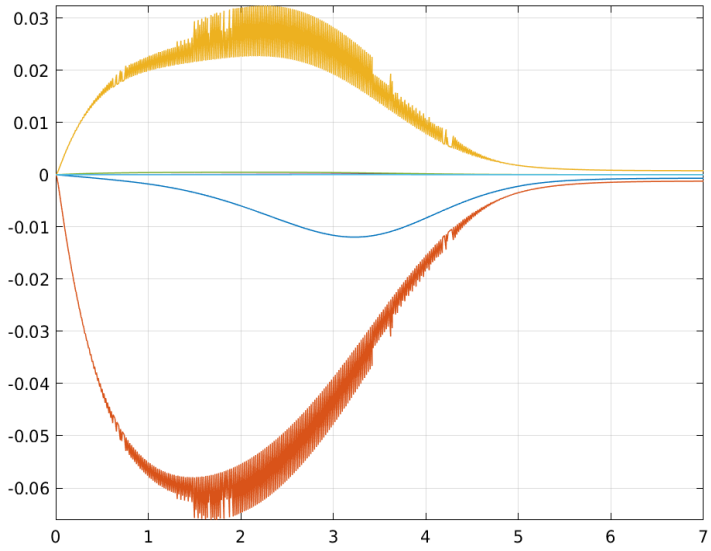
Let's analyze below two cases:

Case1: Perfect knowledge of parameters




As the dynamical parameters are known exactly, the control is smooth and the steady-state error is almost zero ($1.3e^{-10}$).

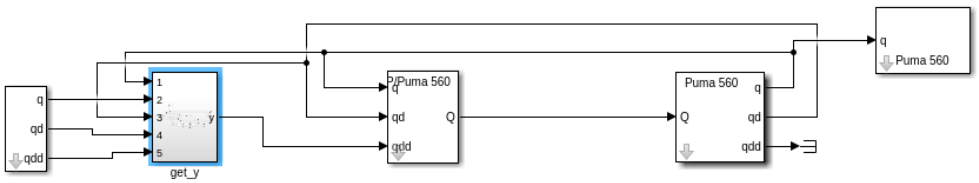
Case2: Perturbed parameters



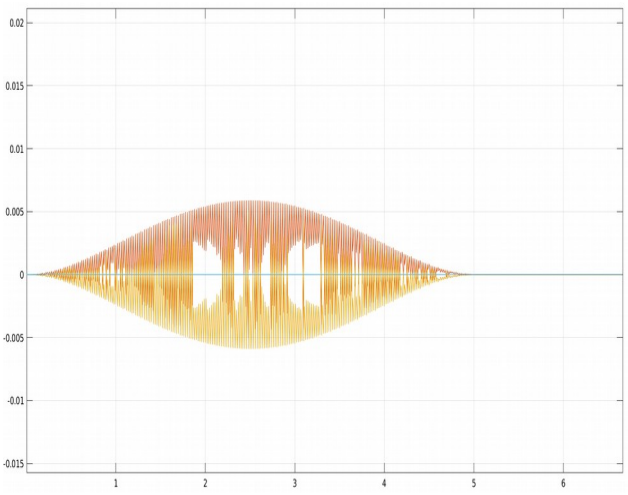

It is obvious from the above graph which illustrates the trajectory tracking error, the steady-state error is higher than the case1, due to the perturbed model parameters.

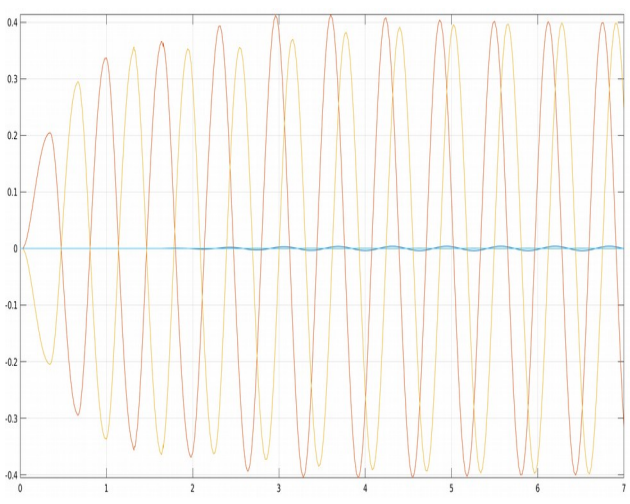
# Task4

## Simulink graph




## Results

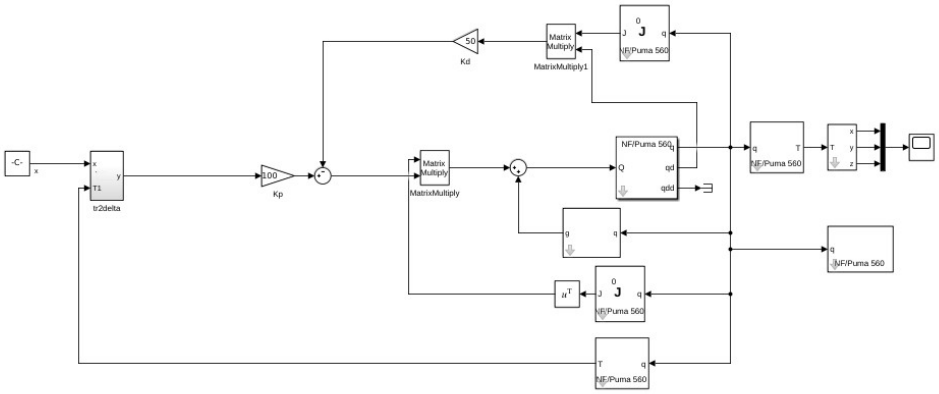


The control method is basically inverse dynamics, while robustness term has been added as an addition to introduce robustness against the parameter uncertainties and to remove other unknown parameters. The key concept is that there has to be bounds of uncertainties as in this case, namely trajectory control of the robot manipulator. The variable “ρ” is proportional to the degree of uncertainties. The variable “Q” which determines the sliding surface dynamics has to be positive definite. Here, “ρ” has been chosen 500. As it can be seen from above figures, the positive definite and symmetric “Q” cause the error to converge to zero which is on the left. However, the negative “Q” which is on the right, cause the error to diverge. Moreover, if the trajectory tracking error of robust controller and simple inverse controller are compared, it will be noticed that the tracking error of the robust controller is significantly lower.

# Task5

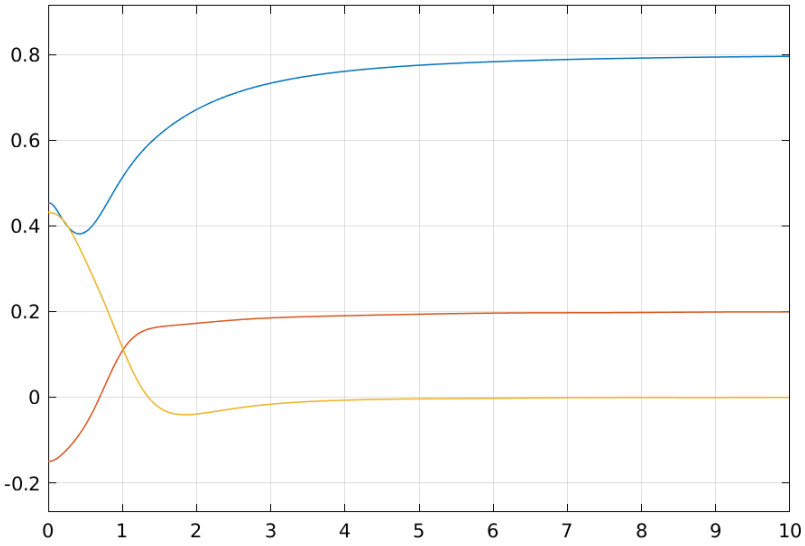
## Simulink graph




## Results

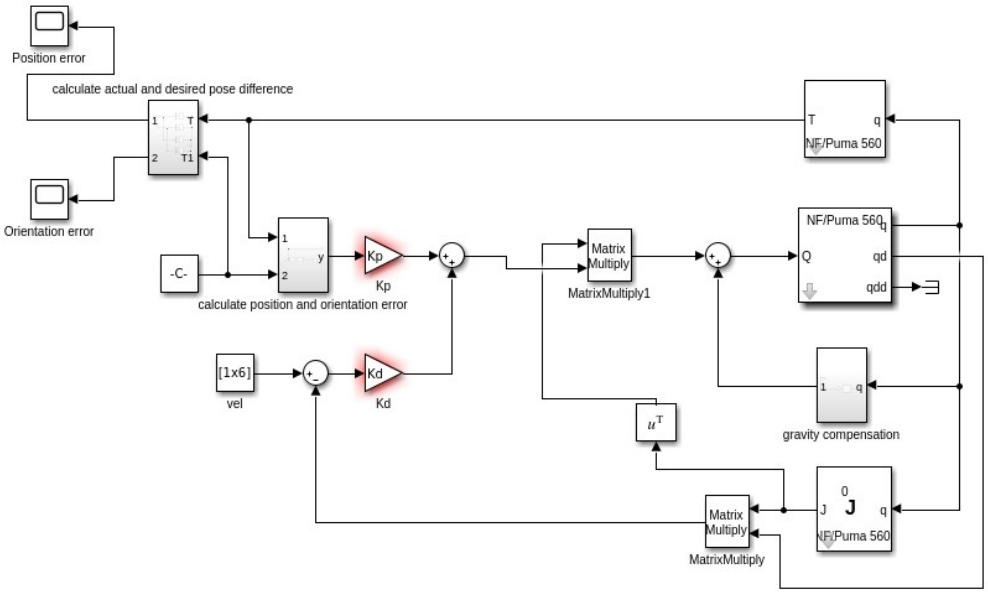
In the second task, PD + gravity compensation has been applied in joint-space, while in this case in work-space. The key difference is that, the work-space control requires calculation of Jacobian of the manipulator online. Additionally, while it is easy to control the orientation in joint-space, this task becomes harder in work-space which will be handled in the next task. Moreover, as the desired points are assigned in work-space coordinates the work-space control is preferred. However, it is difficult to see singularities during the motion of the manipulator which is not an issue in joint-space control.

Example set-point: (0.8, 0.2, 0)




# Task6

## Simulink graph




## Results

The purpose of the task is to control pose of the end-effector in work-space. In order to accomplish this task both, linear position and orientation error have to be calculated in work-space coordinates. It is easy to calculate the linear position error, while it becomes difficult to calculate orientation

error in workspace coordinates. So, transformation matrices have been used in this task in order to get both, linear position and orientation error.

Namely, there are two transformation matrices,

$$T_{current} = \begin{bmatrix} n(t) & o(t) & a(t) & p(t) \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_{desired} = \begin{bmatrix} n_d(t) & o_d(t) & a_d(t) & p_d(t) \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Errors can be calculated,

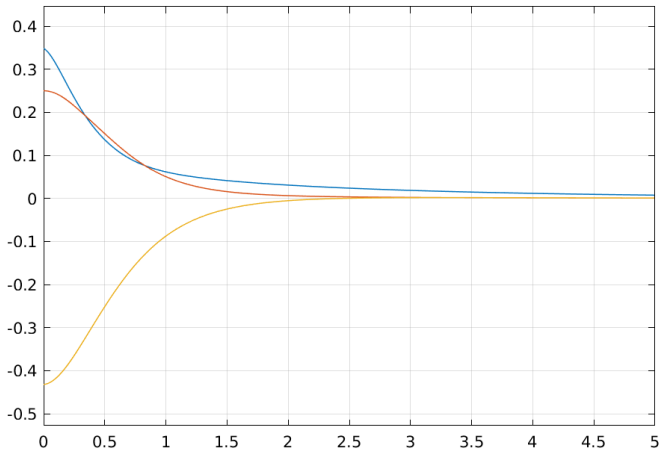
$e_p = p_d - p$

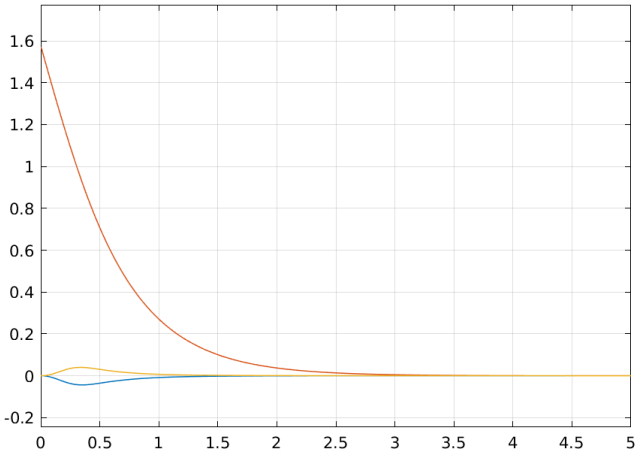
$e_o = 0.5 * (n \otimes n_d + o \otimes o_d + a \otimes a_d)$

During calculation of velocity error, Jacobian has been utilized.

Taking into account above formulations, PD + gravity compensations can be utilized very easily.

Results of the simulation: position (left) and orientation (right) errors




# Conclusion

In conclusion, on these six laboratory tasks, centralized control techniques, like PD control + gravity compensation, inverse dynamics control have been applied both in point-point control and trajectory tracking which have been created both in joint-space and workspace.