ME604 Robotics

Course Project

Dynamics and Control of ABB IRB1600-8/1.45 Robot

Submitted by

Ayush Agrawal

170100044

Under the Supervision of

Prof. Abhishek Gupta

The report gives an account for the analytical, computational and simulation work that was done to complete the course project based on dynamics and control of 6DOF manipulator with spherical joint. A description of the choices of parameters chosen for the simulation is also provided in this report. The results and steady state errors from each simulation are presented using graphs, tables and figures.

Table of Contents

INDEPENDENT JOINT CONTROL

Problem Statement

Please follow these guidelines for all controllers:

1. Set all motor and link friction to zero.

2. Set your closed loop bandwidth to 5 Hz (10π rad/s).

3. Design your controllers to be critically damped (ς = 1), where possible.

Question

a. Design and implement an independent joint controller to move the robot from current position to a new one. This new position will be specified in the joint-space. For convenience, you may want to choose J_{eff} to be $J_{m_i} + d_{ii}$ for each joint (instead of using some constant average value).

b. Verify that the steady state errors are consistent with your choice of gains (note that accelerations and velocities are zero at steady state)

c. Implement a non-linear PD controller with gravity compensation and compare its performance with the independent joint controller.

Solution

The inertial and other dynamics related parameters are same as that given in the report of Assignment 3. Although assignment 3 was about observing the motion of manipulator under gravity, this assignment deals with tracking desired position using **Independent Joint control** and **PD controller with gravity compensation.**

The difference between the two controller lies in their expression and the choice of proportional and derivative gain for the controllers. In **independent joint control** we have local information related to the kinematics of the particular joint and that is why we cannot generate the additional torque out of controller to compensate the gravity load. Also, the $K_p \; and \; K_v$ for this controller get updated for each iteration as per the following equations:

$$\omega_n = 10\pi \; and \; \varsigma = 1$$

$$K_p = \frac{\omega_n^2 d_{ii}}{K}$$

$$K_n = \frac{2\omega_n d_{ii} - B_{eff}}{K}$$

Here, K is a motor constant for a particular joint driving a particular link. This value is different for different motors. We have considered its value as:

$$K = \frac{10\sqrt{J}}{R}$$

Here, R is the resistance of the motor and its value is 1Ω for all the motors.

For, **PD controller with gravity compensation** the K_p *and* K_v values are chosen by trial and error. This controller has access to global information about the kinematics of all joints and hence we can compute the torque necessary to balance the gravity load.

Now as we see that that independent joint controller lags the gravity balancing torque, this will render errors when the controller achieves steady state. The steady state error can be calculated using the following equation:

$$e_i = -\frac{d_i}{K_p}$$

Remember K_p is different for all the joints.

With these things in mind lets have a look at the results after applying the controllers.

Results

Initial configuration = $[0, \frac{\pi}{6}, 0, \frac{\pi}{2}, \frac{\pi}{2}, \frac{\pi}{2}]$

Final configuration = $[\pi, \pi, \pi, \frac{\pi}{4}, \frac{\pi}{12}, 2\pi]$

Initial joint velocities = [3, 0, 5, 6, 1, -5] rad/sec

Control Strategies:

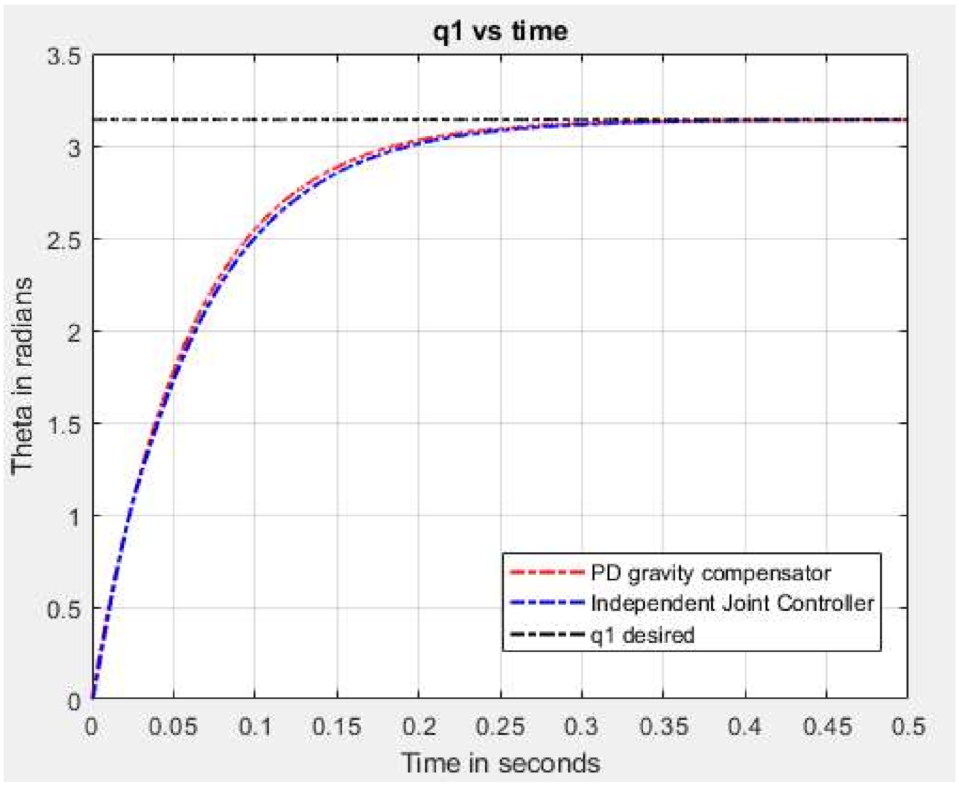
Independent Joint Control: $\boldsymbol{\tau = K_p(q_{des} - q) - K_v\dot{q}}$

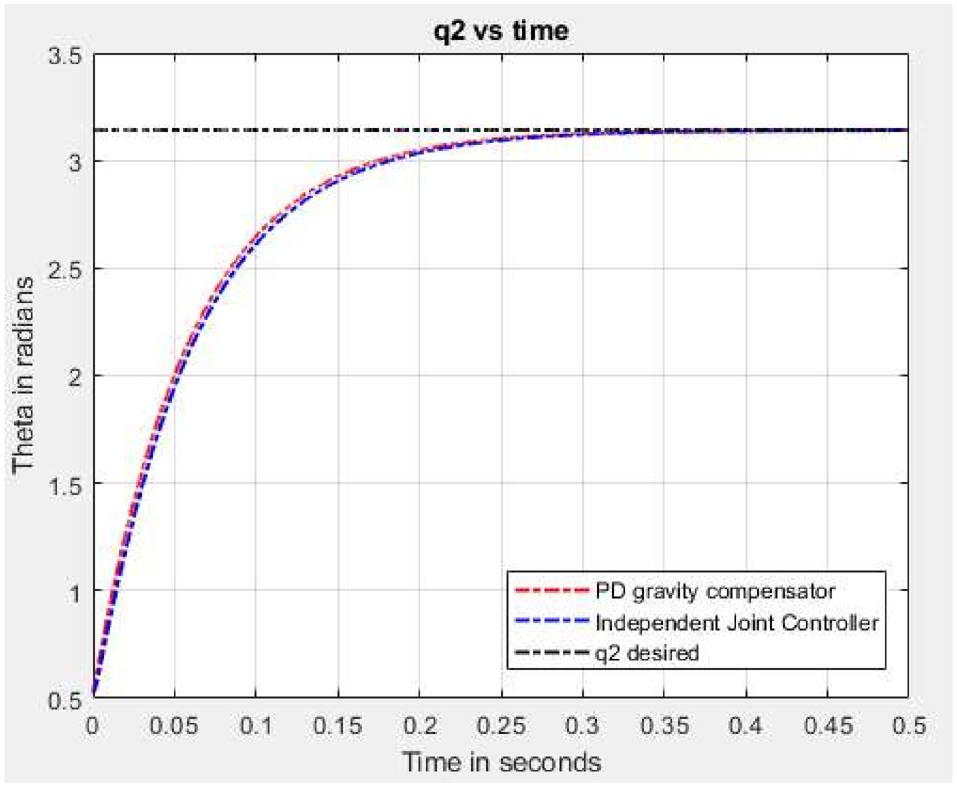
PD control (gravity compensator): $\boldsymbol{\tau = K_p(q_{des} - q) - K_v\dot{q} + \tau_g}$

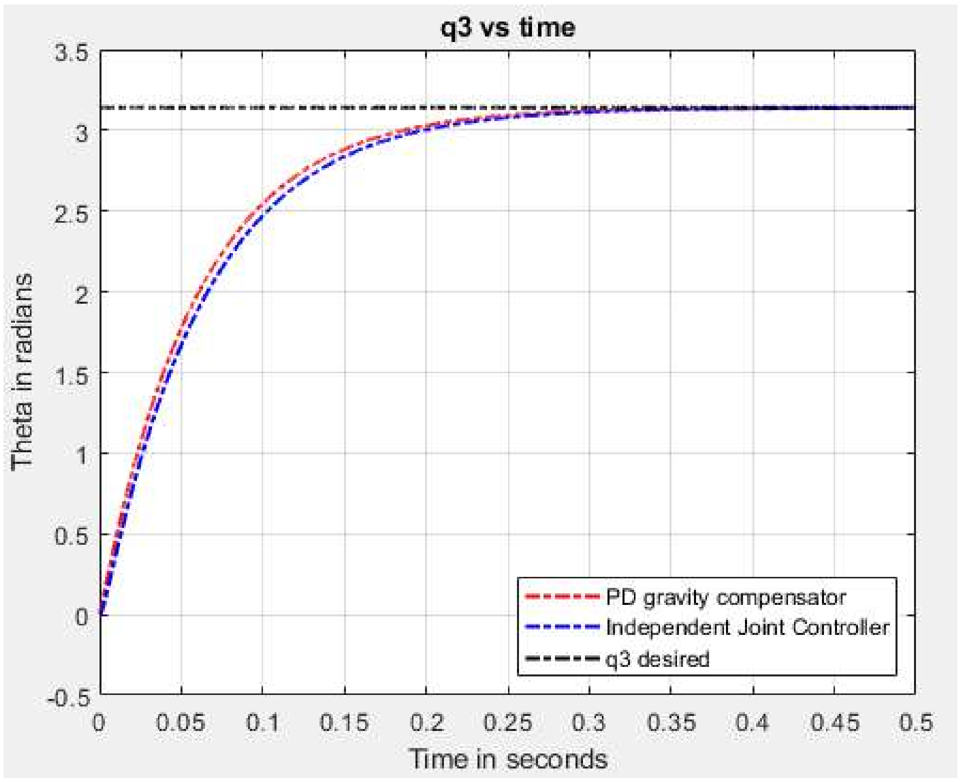
Note: The values are K_p and K_v are different for different joints and controllers.

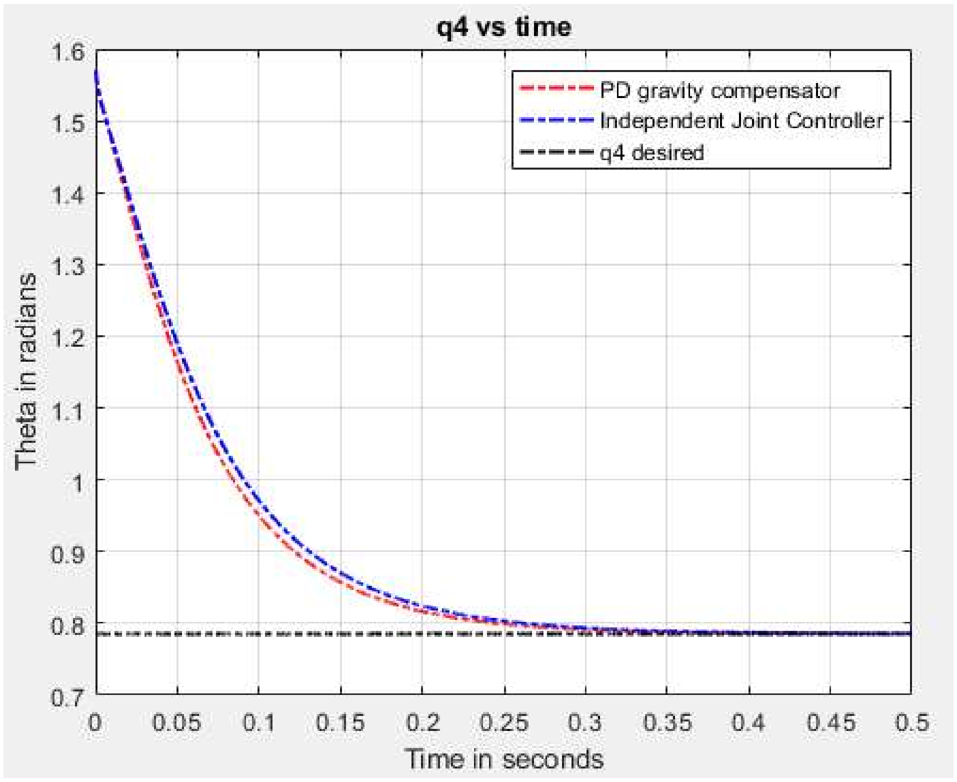
Now, let's compare their performance using some graphs:

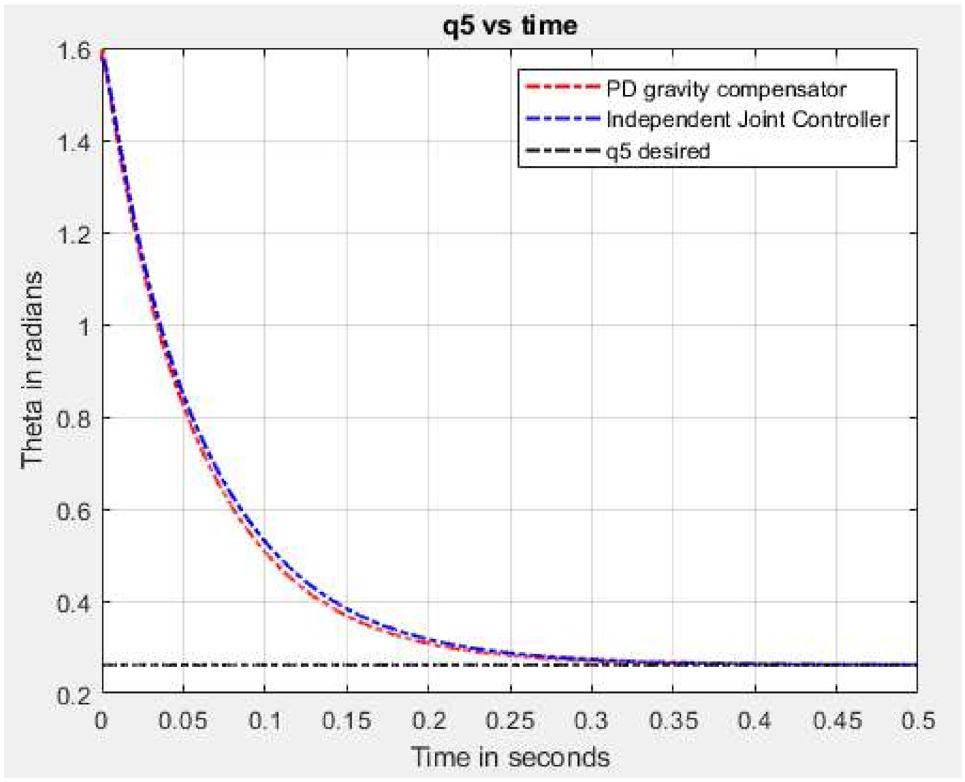
From the graph it can be seen that the independent joint controller is a bit slower as compared to the PD controller.

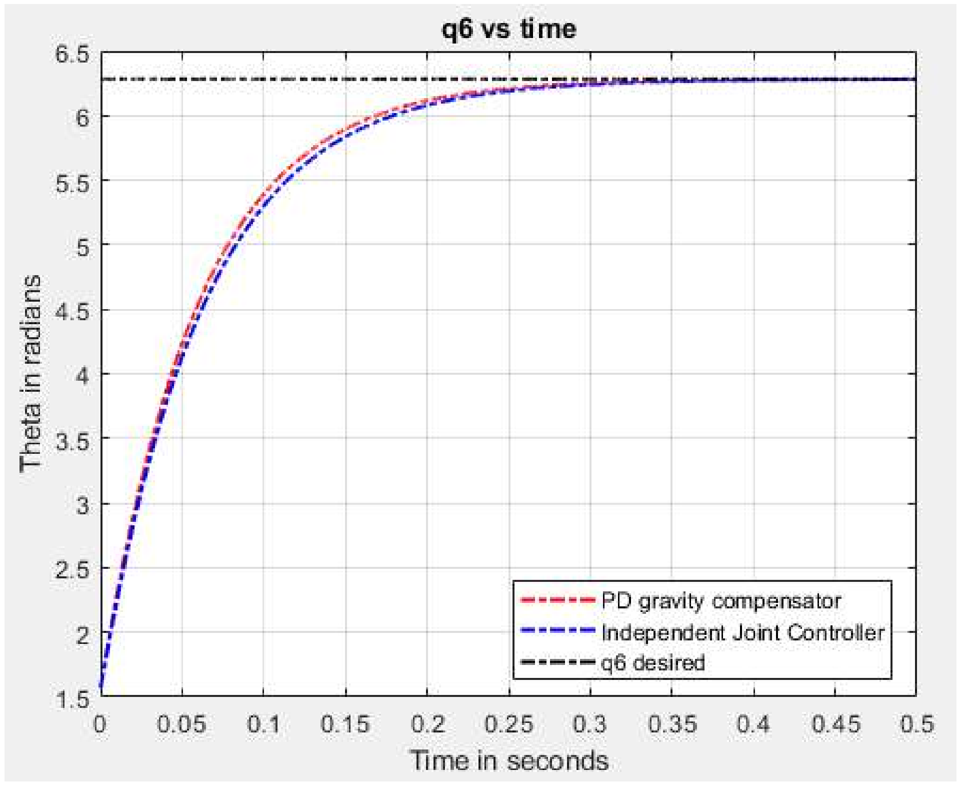

q1 vs time
Theta in radians
Time in seconds
0
0.5
1
1.5
2
2.5
3
3.5
0.05
0.1
0.15
0.2
0.25
0.3
0.35
0.4
0.45
PD gravity compensator
Independent Joint Controller
q1 desired


q2 vs time
Theta in radians
Time in seconds
0.5
1
1.5
2
2.5
3
3.5
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
0.4
0.45
PD gravity compensator
Independent Joint Controller
q2 desired


q3 vs time
Theta in radians
Time in seconds
PD gravity compensator
Independent Joint Controller
q3 desired
3.5
3
2.5
2
1.5
1
0.5
0
-0.5
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
0.4
0.45
0.5


q4 vs time
Theta in radians
Time in seconds
PD gravity compensator
Independent Joint Controller
q4 desired
1.6
1.5
1.4
1.3
1.2
1.1
1
0.9
0.8
0.7
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
0.4
0.45
0.5


q5 vs time
Theta in radians
Time in seconds
PD gravity compensator
Independent Joint Controller
q5 desired
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
0.4
0.45
0.5


q6 vs time
Theta in radians
Time in seconds
PD gravity compensator
Independent Joint Controller
q6 desired
6.5
6
5.5
5
4.5
4
3.5
3
2.5
2
1.5
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
0.4
0.45
0.5

Steady State Error in Independent Joint Controller:

Joint	$q_{desired}$	q_{final}	Numerical Error	Theoretical Error
1	3.1416	3.1070	-0.011	0
2	3.1416	3.1259	-0.005	-0.007
3	3.1416	3.0850	-0.017	-0.0005
4	0.7854	0.7826	0.004	0.000001
5	0.2618	0.2607	0.004	-0.0001
6	6.2832	6.1701	-0.018	0

Here, theoretical error is:

$$e_{theoretica} = \frac{R.gravload(q_{final})}{K_p}$$

Steady State Error in PD controller with gravity compensation:

Joint	$q_{desired}$	q_{final}	Numerical Error
1	3.1416	3.1408	-0.0008
2	3.1416	3.1410	-0.0005
3	3.1416	3.1408	-0.0007
4	0.7854	0.7856	0.0002
5	0.2618	0.2621	0.0003
6	6.2832	6.2821	-0.0011

From the numerical results it can be seen that the steady state errors for PD controller is smaller as compared to independent joint controller. Although the values are really close. The incorporation of gravity load torque in PD controller improved the performance but the improvement is not much significant if we take independent controller as reference.

Joint Space Inverse Dynamics Controller

Problem Statement:

Please follow these guidelines for all controllers:

1. Set all motor and link friction to zero.

2. Set your closed loop bandwidth to 5 Hz ($(10\pi\ rad/\text{sec}$)

3. Design your controllers to be critically damped, where possible.

Question:

a. Given the goal position (location and orientation) of the end-effector, use inverse kinematics to determine joint space goal configuration of the robot.

b. Implement a non-linear PD controller with gravity compensation to move the robot from the start to goal configuration

c. Plan a smooth joint-space trajectory to move the robot from the current joint configuration to a specified goal configuration. The robot should be at rest at both locations.

d. Implement an inverse dynamics controller to move the robot along the trajectory planned in the previous step.

e. How is the performance affected if you underestimate the gravity vector by 5%, i.e., your estimate of the gravity vector is .95 g(q)? Explain your findings.

Regarding Assignment 5 Folder:

Along with the main code file and functions, the folder contains README file for the code. It is recommended to read it before running the code. The folder also contains two video files that show the animation of dynamics of robot under PD controller and joint space inverse dynamics controller.

Solution:

The inertial and other dynamics related parameters are same as that given in the report of Assignment 3. This assignment deals with tracking desired trajectory using **PD controller with gravity compensation** and **Joint space inverse dynamics controller.**

As an input to these controllers, user need to give initial and final pose (position and orientation) and hence the code transforms these points into configuration space for the purpose of calculating dynamics of robot in **joint space.**

Control Strategies:

$Natural\ frequency\ \omega_n = 10\pi\ and\ damping\ coefficient\ \varsigma = 1.$ These are the values that are used in both the controllers. The input control torque for the two controllers are given below.

PD control with gravity compensator

$$u = K_p(q_{desired} - q) + K_v(\dot{q}_{desired} - \dot{q}) + robot.gravload(q)$$

Joint space inverse dynamics controller

$$u = M(q)a_q + C(q,\dot{q})\dot{q} + g(q)$$

$$a_{\boldsymbol{q}} = \ddot{q}_{desired} + K_p(q_{desired} - q) + K_v(\dot{q}_{desired} - \dot{q})$$

$$C(q,\dot{q}) = robot.coriolis(q,\dot{q})$$

$$g(q) = robot.gravload(q)$$

The proportional and derivative gain are chosen appropriately.

Planning Trajectory:

The trajectory is planned by considering a fifth order polynomial that defines the variation of joint angles with time.

$$q = a_0 + a_1t + a_2t^2 + a_3t^3 + a_4t^4 + a_5t^5$$

$$\dot{q} = a_1 + 2a_2t + 3a_3t^2 + 4a_4t^3 + 5a_5t^4$$

$$\ddot{q} = 2a_2 + 6a_3t + 12a_4t^2 + 20a_5t^3$$

Boundary conditions:

Initial kinematics of joints are defined at t = 0 and final joint kinematics at t = 1.

Joint	q_i	q_f	$\dot{q}_i$	$\dot{q}_f$	$\ddot{q}_i$	$\ddot{q}_f$
1	2.1845	-0.7922	0	0	0	0
2	-0.5092	-1.1405	0	0	0	0
3	1.0618	2.1561	0	0	0	0
4	0.1590	-2.5294	0	0	0	0
5	0.3621	0.1293	0	0	0	0
6	-1.0926	2.5357	0	0	0	0

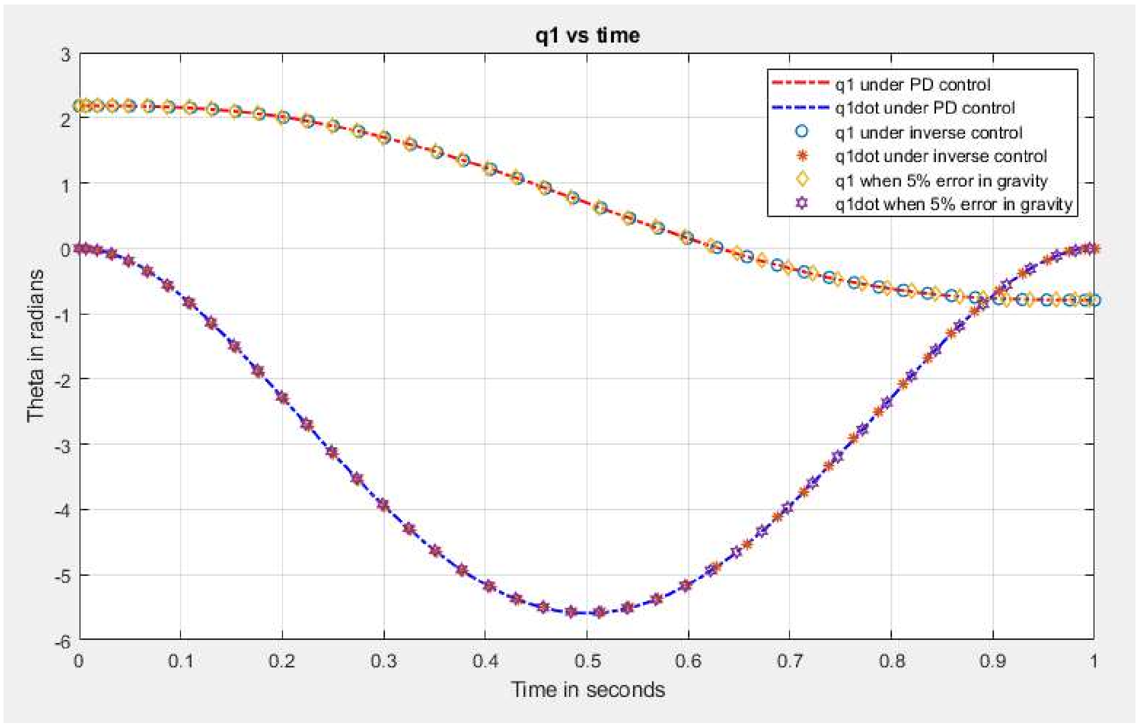
These initial and final configurations are calculated using inverse kinematics. Since final joint kinematics is defined at t = 1, $T_{max} = 1$.

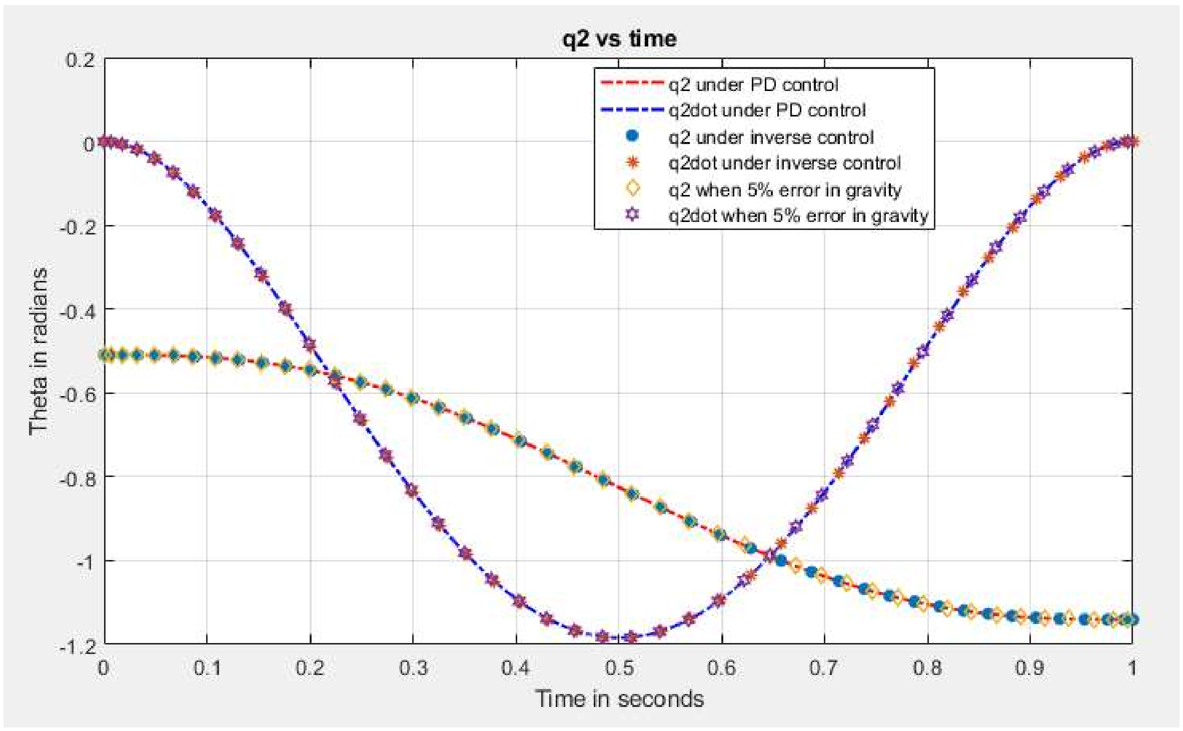
Calculated coefficients are:

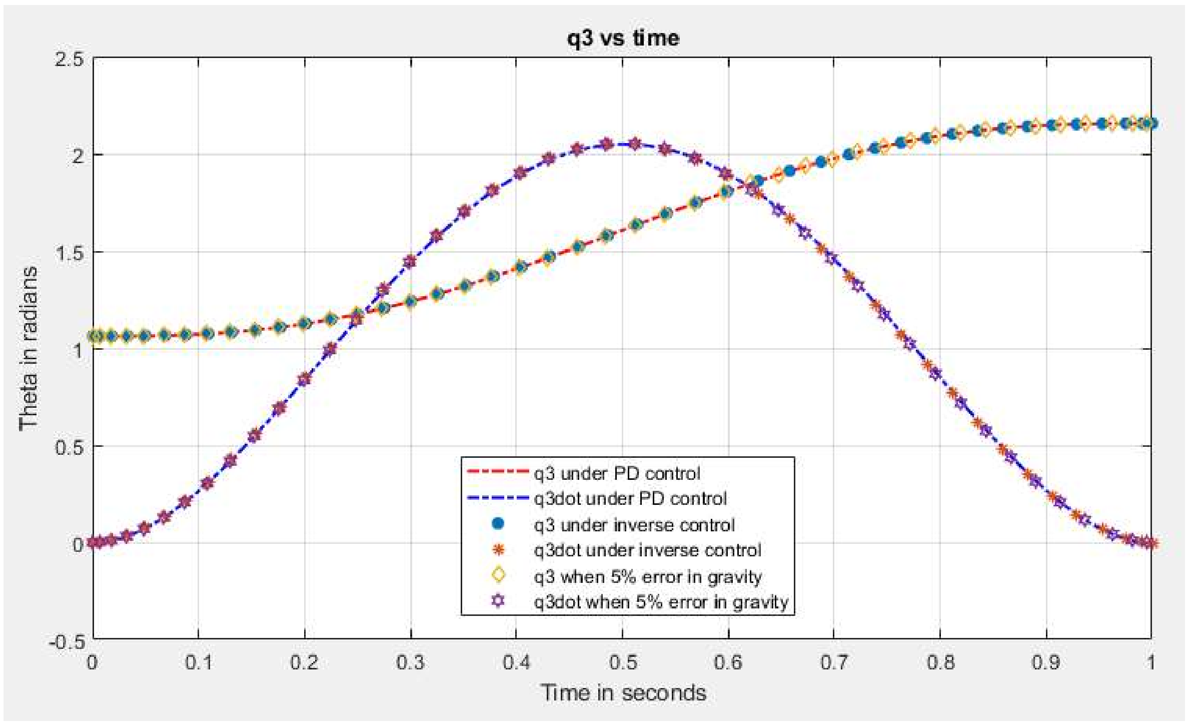
Joint	a_0	a_1	a_2	a_3	a_4	a_5
1	2.1845	0	0	-29.7672	44.6508	-17.8603
2	-0.5092	0	0	-6.3131	9.4697	-3.7879
3	1.0618	0	0	10.9437	-16.4156	6.5662
4	0.1590	0	0	-26.8842	40.3262	-16.1305
5	0.3621	0	0	-2.3283	3.4924	-1.3970
6	-1.0926	0	0	36.2828	-54.4242	21.7697

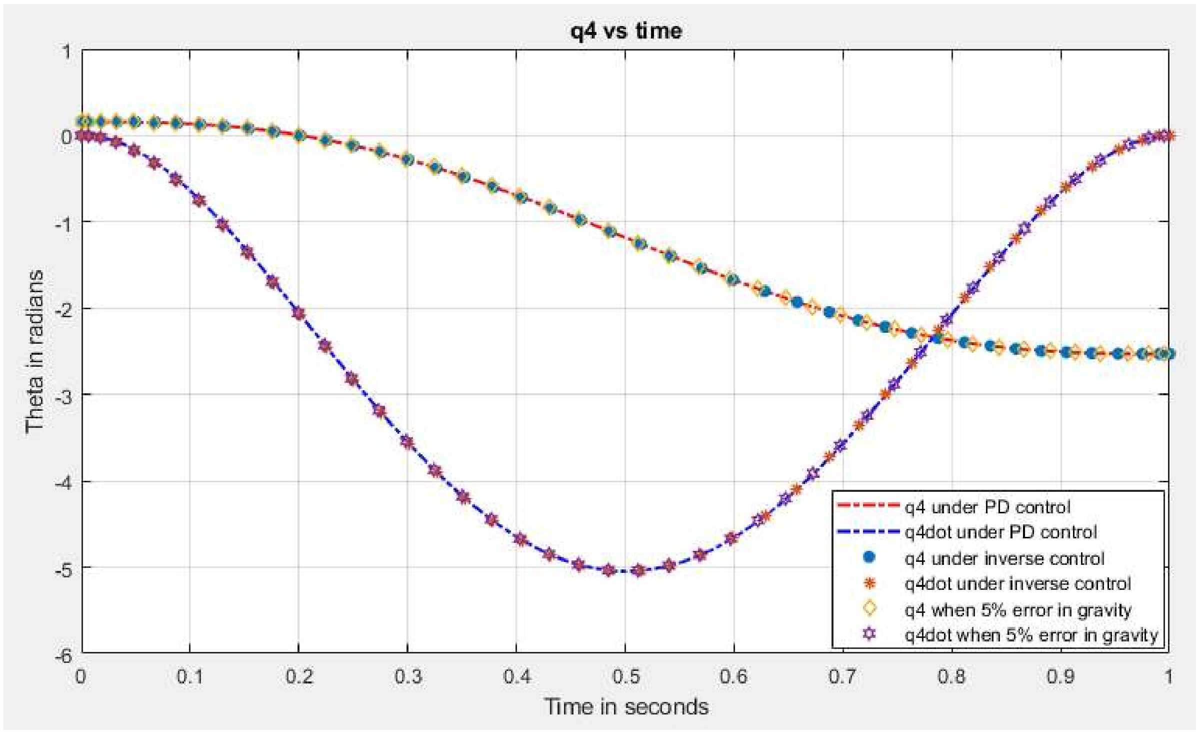
Comparing the results of the two controllers:

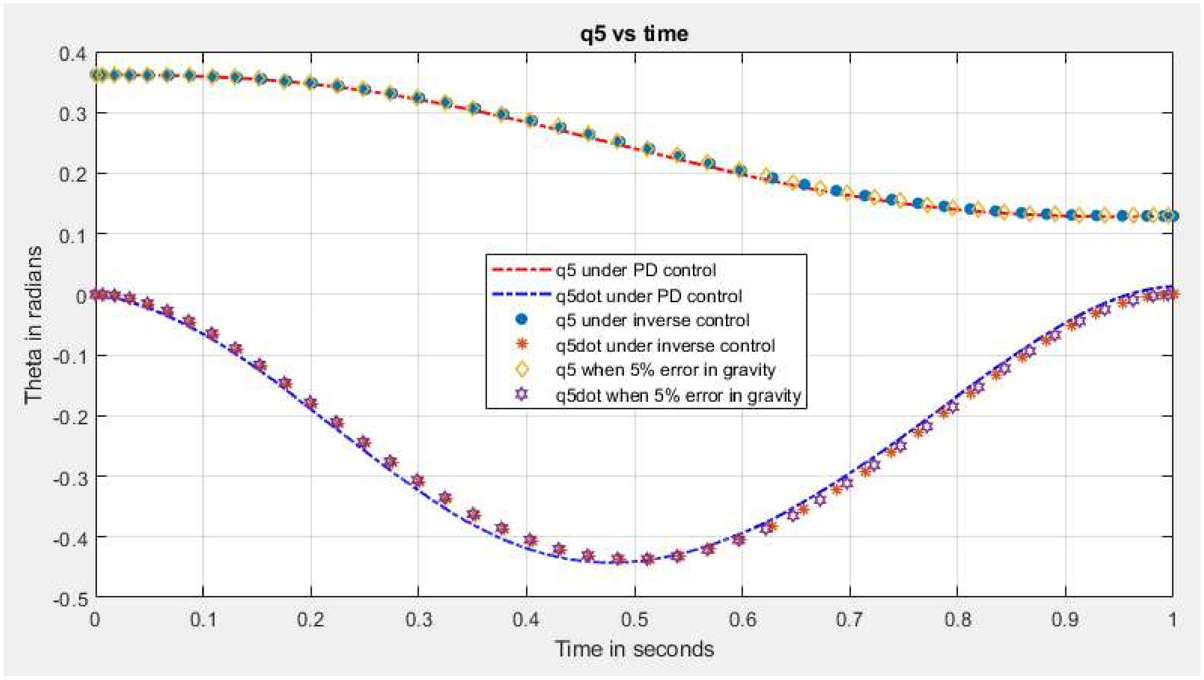
The graphs below show the variation of q *and* $\dot{q}$ with time. These are plotted for different controllers as well. Also, the effect of 5% error in estimation of g(q) can be seen in these graphs.

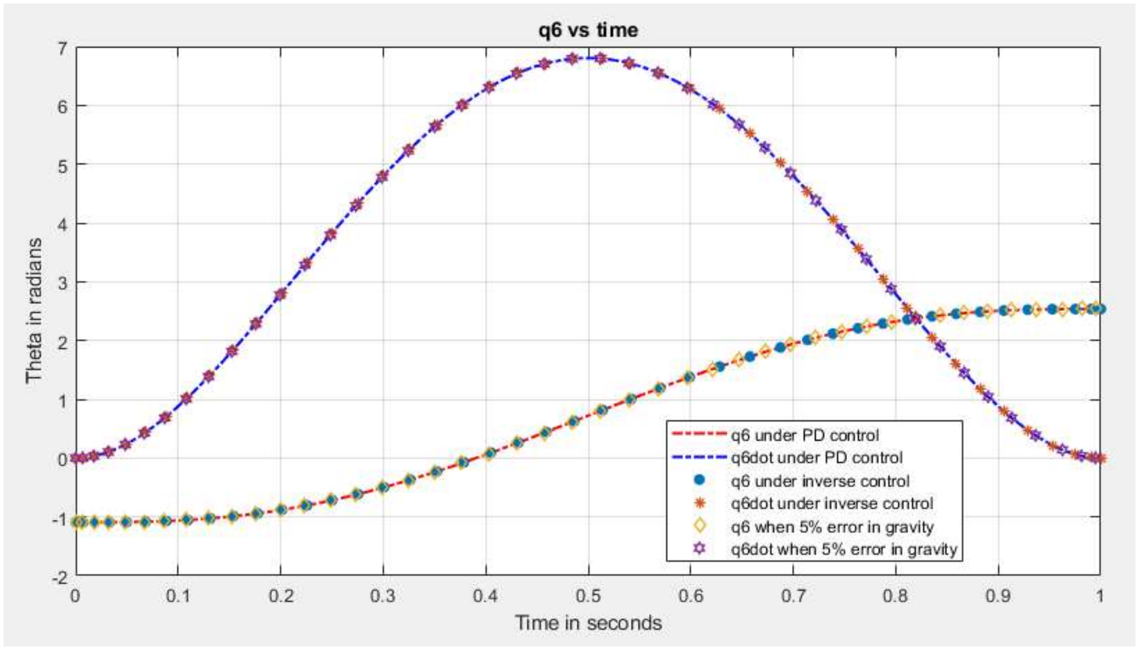



q2 vs time
q2 under PD control
q2dot under PD control
q2 under inverse control
q2dot under inverse control
q2 when 5% error in gravity
q2dot when 5% error in gravity
Theta in radians
Time in seconds
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1.2
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


q3 vs time
q3 under PD control
q3dot under PD control
q3 under inverse control
q3dot under inverse control
q3 when 5% error in gravity
q3dot when 5% error in gravity
Theta in radians
Time in seconds
2.5
2
1.5
1
0.5
0
-0.5
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


q4 vs time
Theta in radians
Time in seconds
q4 under PD control
q4dot under PD control
q4 under inverse control
q4dot under inverse control
q4 when 5% error in gravity
q4dot when 5% error in gravity


q5 vs time
Theta in radians
Time in seconds
q5 under PD control
q5dot under PD control
q5 under inverse control
q5dot under inverse control
q5 when 5% error in gravity
q5dot when 5% error in gravity



Errors in the two controllers

Along with the steady state errors in the two controllers, the errors in the case of 5% error in estimation of g(q) is also considered.

Joint	$q_{desired}$	PD controller	Inverse Dynamics Controller	If g(q) has 5% error
1	-0.7922	-0.7924	-0.7922	-0.7922
2	-1.1405	-1.1405	-1.1405	-1.1405
3	2.1561	2.1562	2.1561	2.1561
4	-2.5294	-2.5292	-2.5294	-2.5294
5	0.1293	0.1285	0.1293	0.1292
6	2.5357	2.5357	2.5357	2.5357

Error

PD controller	Inverse Dynamics Controller	If g(q) has 5% error
-0.0002	**0**	0
0	**0**	0
0.0001	**0**	0
0.0002	**0**	0
-0.0008	**0**	-0.0001
0	**0**	0

From the above values it can be seen that the **Joint Space Inverse Dynamics Controller** has zero error in the final state while **PD controller** still has some errors in it. This error is occurring because we are not incorporating the second derivative of position in the controller. Also, the coriolis term is ignored in this control strategy.
In the joint space inverse dynamics controller, even if there is a *5% error in estimation of gravity load*, the controller is able to follow the desired trajectory with zero error. This means that the controller is able to reject such kind of disturbances.

___

Task Space Inverse Dynamics Controller

Problem Statement:

Please follow these guidelines for all controllers:

1. Set all motor and link friction to zero.

2. Set your closed loop bandwidth to 5 Hz ($\omega = 10\pi$).

3. Design your controllers to be critically damped, where possible.

Question

a. Verify that the Task space inverse dynamics controller (slide 20, manipulator control) and task space non-linear decoupling controller (slide 27, manipulator control) are essentially same. (you may want to compute τ and compare)

b. Implement a task space PD controller (with gravity compensation) to move the robot end-effector from its current location to a specified goal location. You need not control the orientation.

c. Incorporate damping in the motors in the form $b = \alpha J$ to see how friction affects performance, where α is some constant (same for all joints) and J is the motor inertia. Start with a small value of alpha, say .1, and increase it to see what happens. Comment on your findings.

About the Assignment 6 folder:

Along with the main code file and functions, the folder contains README file for the code. It is recommended to read it before running the code. The folder also contains one video file that show the animation of dynamics of robot under Task Space inverse dynamics controller.

Solution:

The inertial and other dynamics related parameters are same as that given in the report of Assignment 3. This assignment deals with set point tracking using **Task space inverse dynamics controller** and **Task space nonlinear decoupling controller.**

Verifying that both Task Space *Inverse dynamics controller and Nonlinear decoupling controller* give the same expression for Torque:

The controller for task space inverse dynamics controller is defined as:

$$u = M(q)J^{-1}\left(a_x - \dot{J}\dot{q}\right) + C(q,\dot{q})\dot{q} + g(q) \quad \textit{(1)}$$

$$a_x = K_p(x_{desired} - x) - K_v\dot{x} \quad (2)$$

Here, u is the control torque.

Now, let’s try to simplify this controller to *Joint space inverse dynamics controller.* We know the following relations:

$$\dot{x} = J\dot{q} \qquad (3)$$

Differentiating this with respect to time we get

$$\ddot{x} = J\ddot{q} + \dot{J}\dot{q} \qquad (4)$$

From control perspective we can write

$$a_x = Ja_q + \dot{J}\dot{q} \qquad (5)$$

Putting (5) in equation (1) we get

$$u = M(q)a_q + C(q,\dot{q})\dot{q} + g(q) \qquad (6)$$

Similar to equation (2), PD controller for a_q can be constructed

$$a_q = K_p(q_{desired} - q) - K_v\dot{q} \qquad (7)$$

Hence the task space inverse dynamics controller is same as the Joint space inverse dynamics controller.

Now let us look at *Task Space Nonlinear Decoupling Controller:*

$$F = \widehat{M(x)}F' + \hat{V}_x(x,\dot{x}) + \hat{G}(x) \quad , \quad u = J^T F \qquad \textit{(8)}$$

$$F' = K_p(x_{desired} - x) - K_v\dot{x} \qquad (9)$$

Here, F’ is the control force for unit mass, simply F’ = a_x

Now, let’s try to simplify this controller to *Joint space inverse dynamics controller.* We know the following Task space relations:

$$\hat{G}(x) = J^{-T}g(q) \qquad \textit{(10)}$$

$$\hat{V}_x(x,\dot{x}) = J^{-T}C(q,\dot{q})\dot{q} - \widehat{M(x)}\,\dot{J}\dot{q} \qquad (11)$$

$$\widehat{M(x)} = J^{-T}M(q)J^{-1} \qquad (12)$$

Substituting equation (10), (11) and (12) into (8)

$$J^{-T}u = J^{-T}M(q)J^{-1}a_x + J^{-T}C(q,\dot{q})\dot{q} - \dot{\widehat{M(x)}}\,\dot{J}\dot{q} + J^{-T}g(q) \qquad (13)$$

Now substitute (5) in (13)

$$J^{-T}u = J^{-T}M(q)J^{-1}(Ja_q + \dot{J}\dot{q}) + J^{-T}C(q,\dot{q})\dot{q} - J^{-T}\dot{M(q)}J^{-1}\dot{J}\dot{q} + J^{-T}g(q) \qquad (14)$$

Simplifying (14), we get

$$u = M(q)a_q + C(q,\dot{q})\dot{q} + g(q) \qquad (15)$$

$$a_q = K_p(q_{desired} - q) - K_v\dot{q}$$

Therefore, we can conclude that both the techniques are exactly same. Both controllers will give same results for set point tracking as well as trajectory following.

Applying the controller in MATLAB

As an input to these controllers, user need to give initial and final pose (position and orientation) and hence the code transforms these points into configuration space for the purpose of calculating dynamics of robot in joint space.

Control Strategies:

$Natural\ frequency\ \omega_n$= 4π $and\ damping\ coefficient$ ς = 5. These are the values that are used for the controllers. For higher natural frequency (with choice of any damping ratio) oscillatory random behaviour was observed. Equation (1) and (2) are used in the numerical model of the controller. For the equation (2), K_p is a matrix with all diagonal entries equal to $16\pi^2$. Similarly, K_v is a matrix with all diagonal entries equal to 40π. The values are kept same because the inertia will get multiplied to this matrix at later stages in code.

Since the structure of the controller and the related parameters are now clear, let's move on to the results.

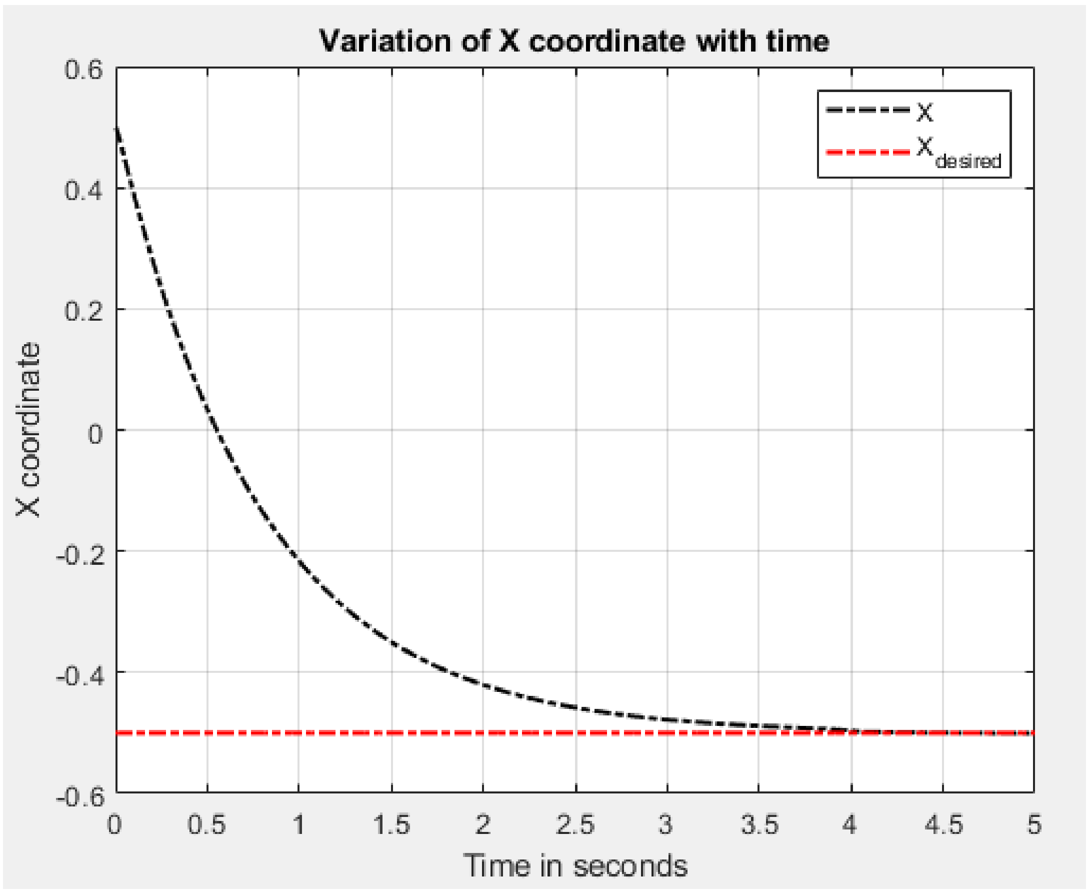
Results from PD controller (with gravity compensation)

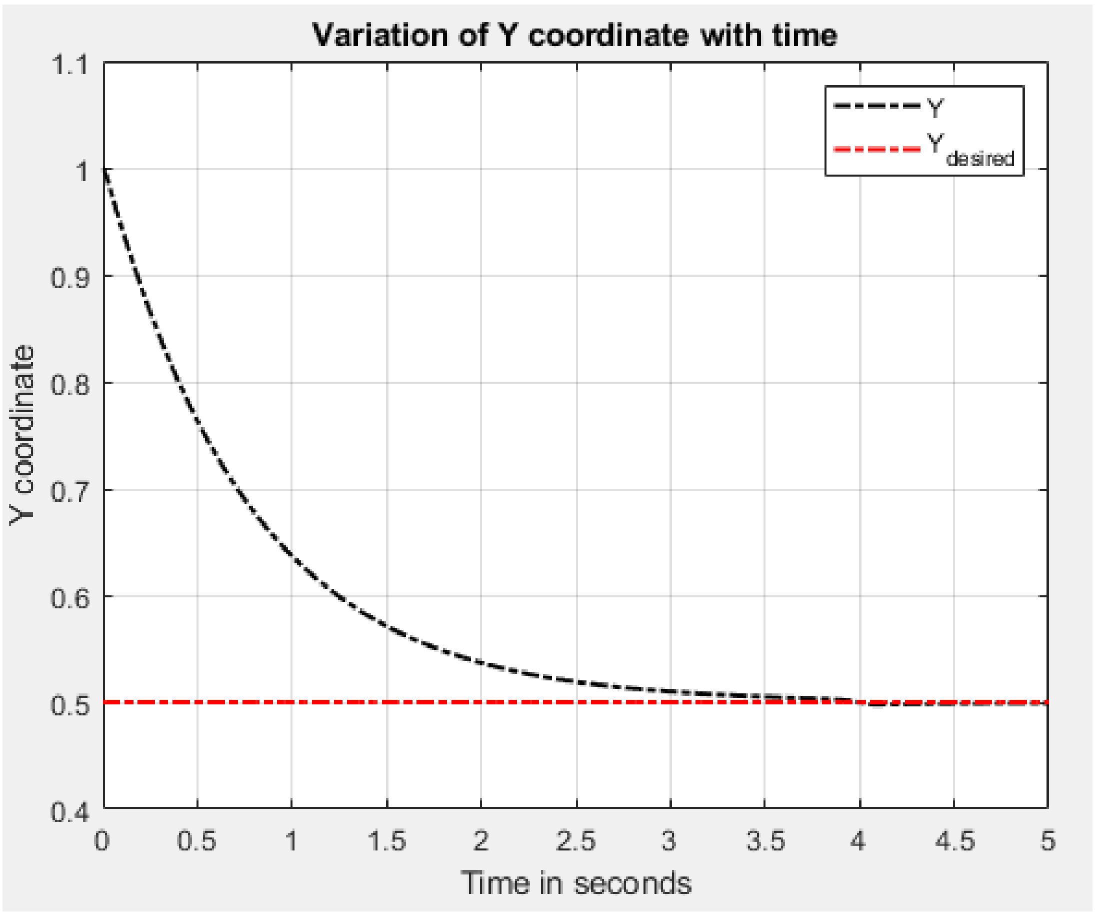
The graphs shown below display the evolution of x, y and z coordinates of the end effector in world frame. In this simulation value of **α is 0.1.**

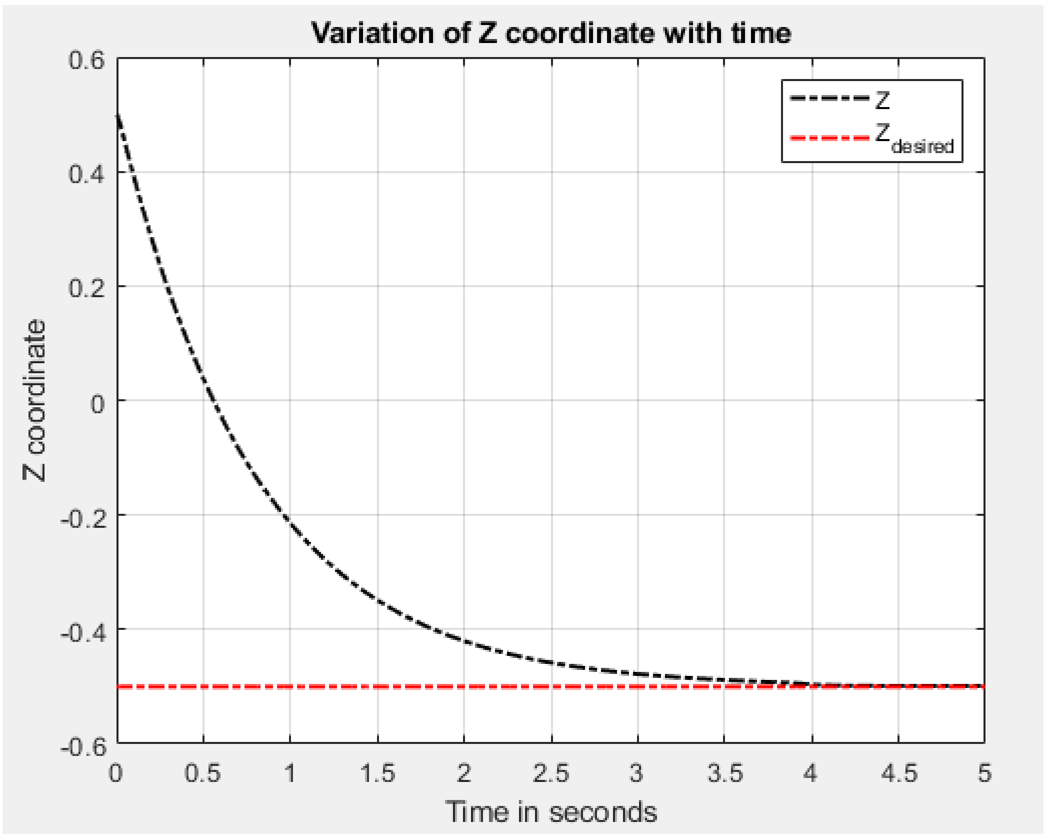
Initial Position of End Effector: [0.5, 1, 0.5, π/3, π/6, π/4]

Final Position of End Effector: [-0.5, 0.5, 0.5, π/3, π/6, π/4]

Initial joint velocities: [0,0,0,0,0,0] rad/sec


Variation of X coordinate with time
X coordinate
Time in seconds
X
$X_{desired}$
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5


Variation of Y coordinate with time
Y coordinate
Time in seconds
Y
$Y_{desired}$
1.1
1
0.9
0.8
0.7
0.6
0.5
0.4
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5



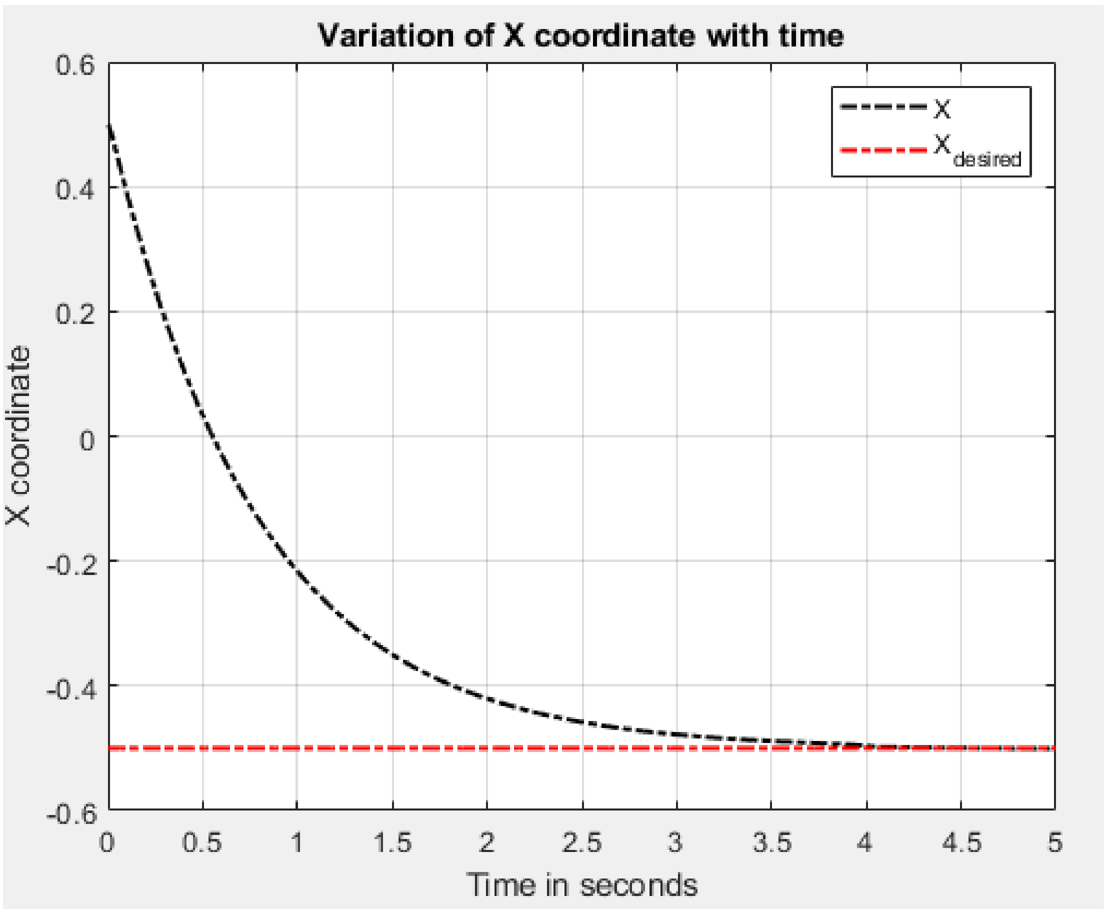
Steady state error in coordinates with α is 0.1.

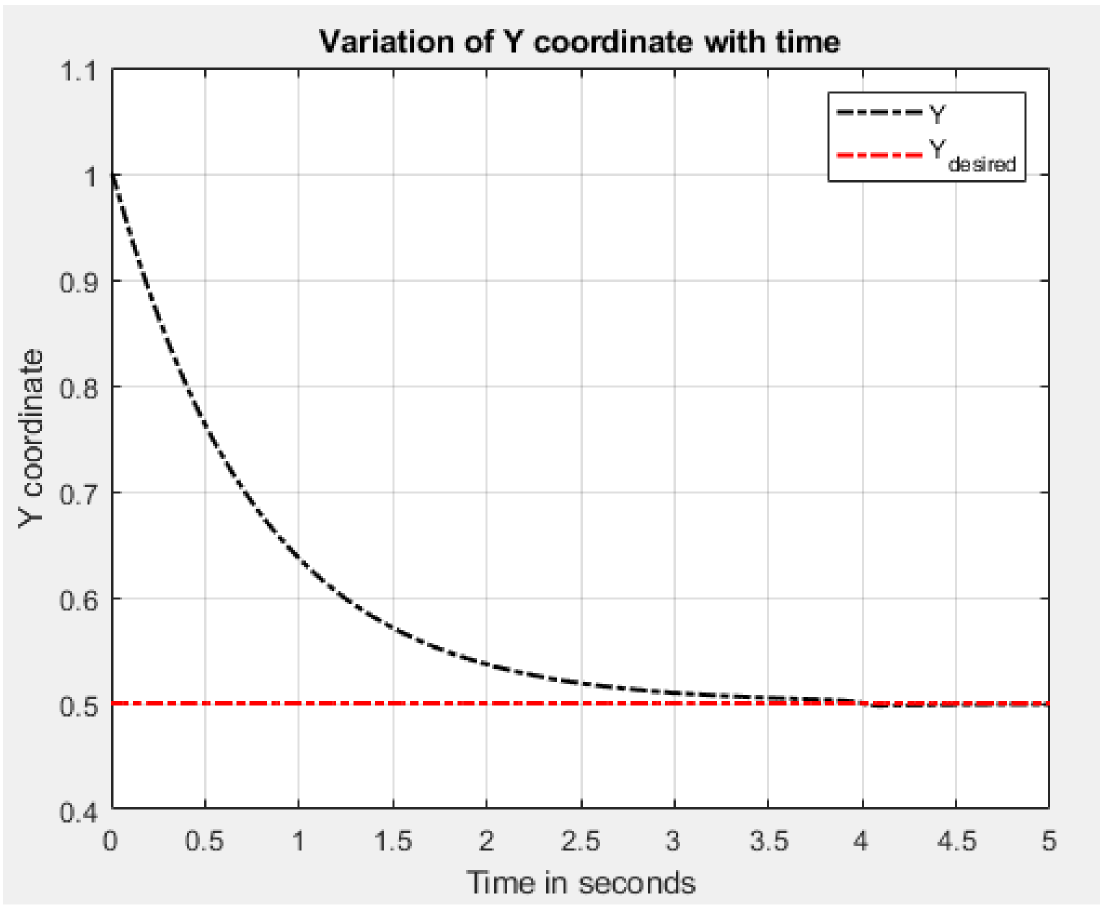
$X_{desired}$	X	Error
-0.5	-0.5004	0.0004
0.5	0.4989	0.0011
-0.5	-0.4986	-0.0014

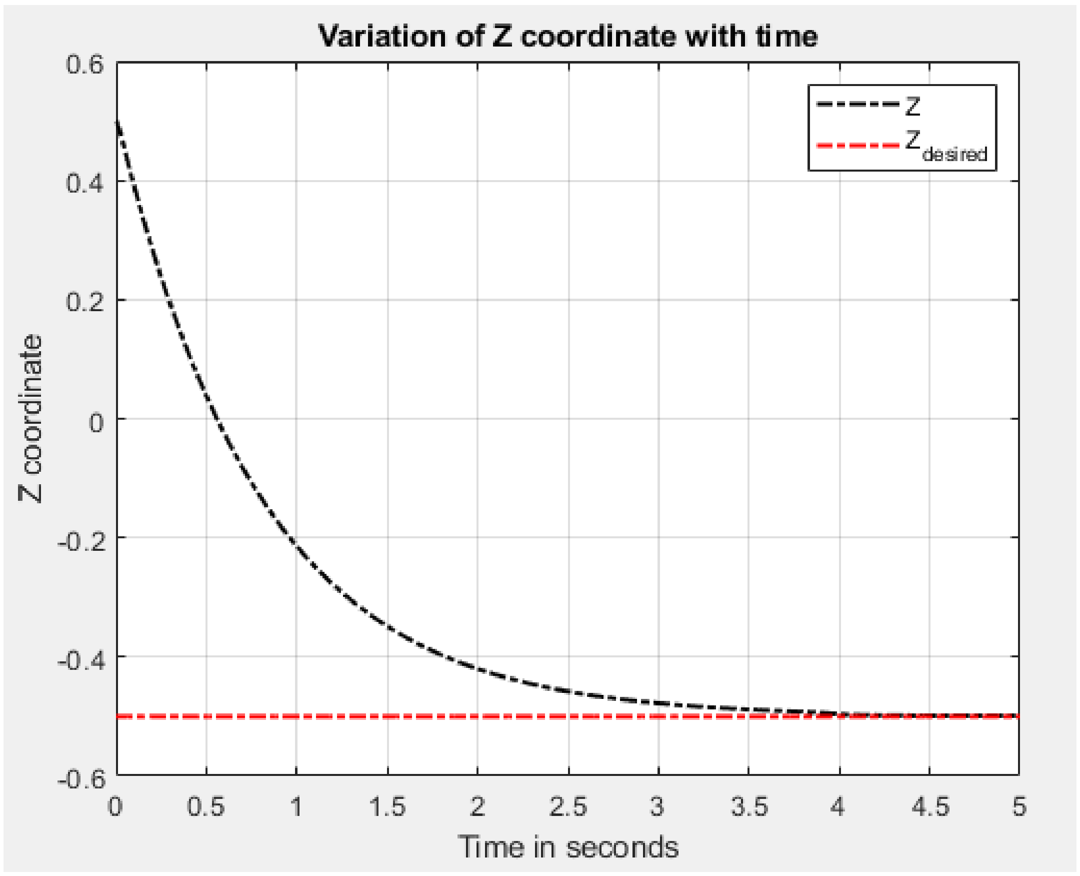
NOTE: If the user tries to look into the values of joint angles, he will find higher error. But in this control strategy our goal is to reach the coordinates in cartesian space. Hence priority is given to reducing the error between robot current position and desired position which can be achieved at a slightly different configuration.

Effect of variation in Coulomb Friction:

What happens to the variation of coordinates with time if we increase the value of **α from 0.1 to 1**? Does it increase the steady state error? Let's look into this through the following graphs.


Variation of X coordinate with time
X coordinate
Time in seconds
X
$X_{desired}$
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5


Variation of Y coordinate with time
Y coordinate
Time in seconds
Y
$Y_{desired}$
1.1
1
0.9
0.8
0.7
0.6
0.5
0.4
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5



Steady state error in coordinates with α is 1.

$X_{desired}$	X	Error
-0.5	-0.5004	0.0004
0.5	0.4989	0.0011
-0.5	-0.4986	-0.0014

So, it can be seen that changing the value of α, either increasing or decreasing, will not have any effect on the final result as it can be seen that the steady state errors are same as that of the previous case. This means that the controller is able to overcome the variation in friction.
