Problem Statement

1. This project will be based on a 6 DOF Stanford Arm. The D-H parameters for the 6 DOF robot are given below and a schematic is shown in Figure 1:

i	a_i	r_i	θ_i	α_i
1	0	0	90°	variable
2	0	*l*	90°	variable
3	0	variable	0°	0
4	0	0	90°	variable
5	0	0	90°	variable
6	0	*m*	0°	variable

. All axes except axis 3 are revolute. Axis 3 is prismatic. Point O is the intersection of Axes 1 and 2, and this point will be taken to be the origin of the Frame '0'. E is the reference point on the end-effector. Develop a computer program that performs motion planning, the inverse dynamics, and the forward dynamics computations for this robot. The program should take the robot parameters (geometric, inertial, etc.) and the desired robot kinematics and generate the joint torques as the output. Assume that the design of the robot is such that there are three non-zero masses in the system:

(i) An 'L' shaped link OAB_1; the motion of this link is affected by the motions of joints 1 and 2.
(ii) Link B_2C; the motion of this link is affected by motions of joints 1, 2 and 3.
(iii) Link CE; the motion of this link is affected by the motions of all 6 joints.

Assume that each straight line part of the link is essentially a member that has a hollow circular cross section of outer radius = 0.025 m, and inner radius = 0.02m; and they are made of a homogeneous material that has a density of 7000 kg/m^3. Take OA = 0.1 m, AB_1 = 0.1 m, B_2C = 0.1 m, and CE = 0.1 m.

Plan and control the robot to move from the initial to the final position where

- E is initially at (0.5, 0, -0.1) and CE is vertical (parallel to Axis 1) with E below C.
- E should finally end up at (-0.4, 0.1, 0.1) with CE is again vertical with E below C.
- The time taken to go from the initial to the final configuration is 10 seconds.

Assume that the controller has to work with an unknown payload mass M. Assume M to be a point mass located at Point E. The program should be able to operate for M in the region [0, 0.4] kg. Use a combined feedforward and feedback control scheme.

Further, assume that the controller operates at 100 Hz, while the time step chosen for the simulator is 10X smaller (1 ms).

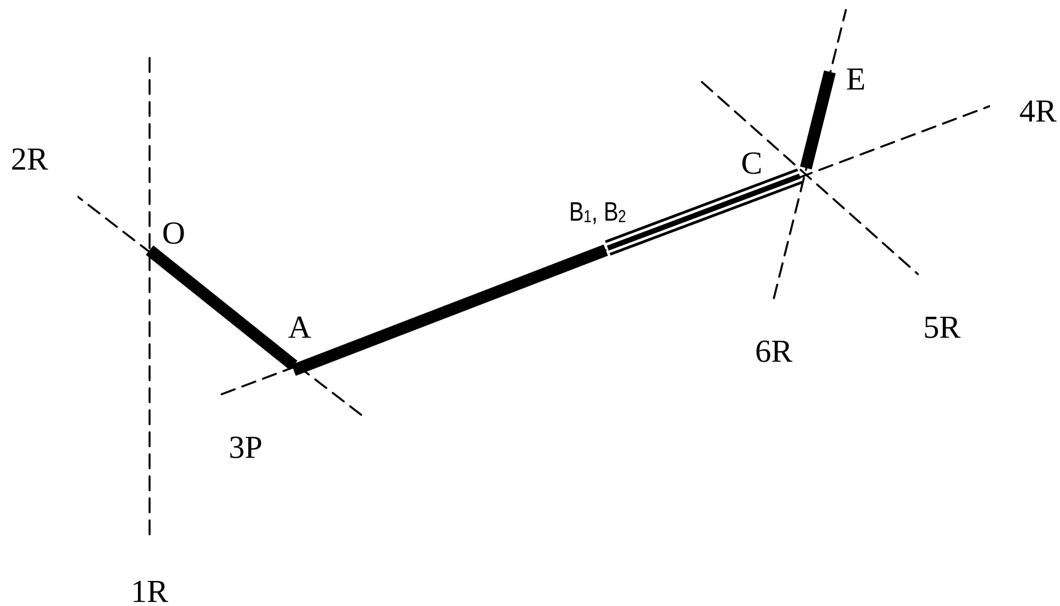


Figure 1: A 6 DOF Stanford Arm Robot

Solution

Simulation & Control of a Stanford Arm

ME 384R: Advanced Dynamics of Robotic Systems (Spring 2012)
Final Project (Question 1)
Gautam Salhotra

Contents:

Program structure:

The simulation program consists of two main components. The first is the offline calculation component, where the initial and final positions are taken as inputs, along with details of the Stanford arm (DH parameters, dimensions, link geometry and weights), to calculate the desired values of joint position, velocity and acceleration as well as the required feed forward torque.

The second component is the main simulation, where the simulation occurs at 10 times the frequency of the controller, updating the real Θ, Θdot and Θdoubledot every 1ms. At the end of every 10ms, these values are stored in an array (to be compared with the desired values calculated offline in the first component above). Comparison of the actual values with the desired values gives the error, which is used by the controller to calculate the feedback torque. Flowcharts for both the offline and online components of the simulation are given below.

Fig 1: Flowchart for offline calculation of desired values of position, velocity, acceleration and torque

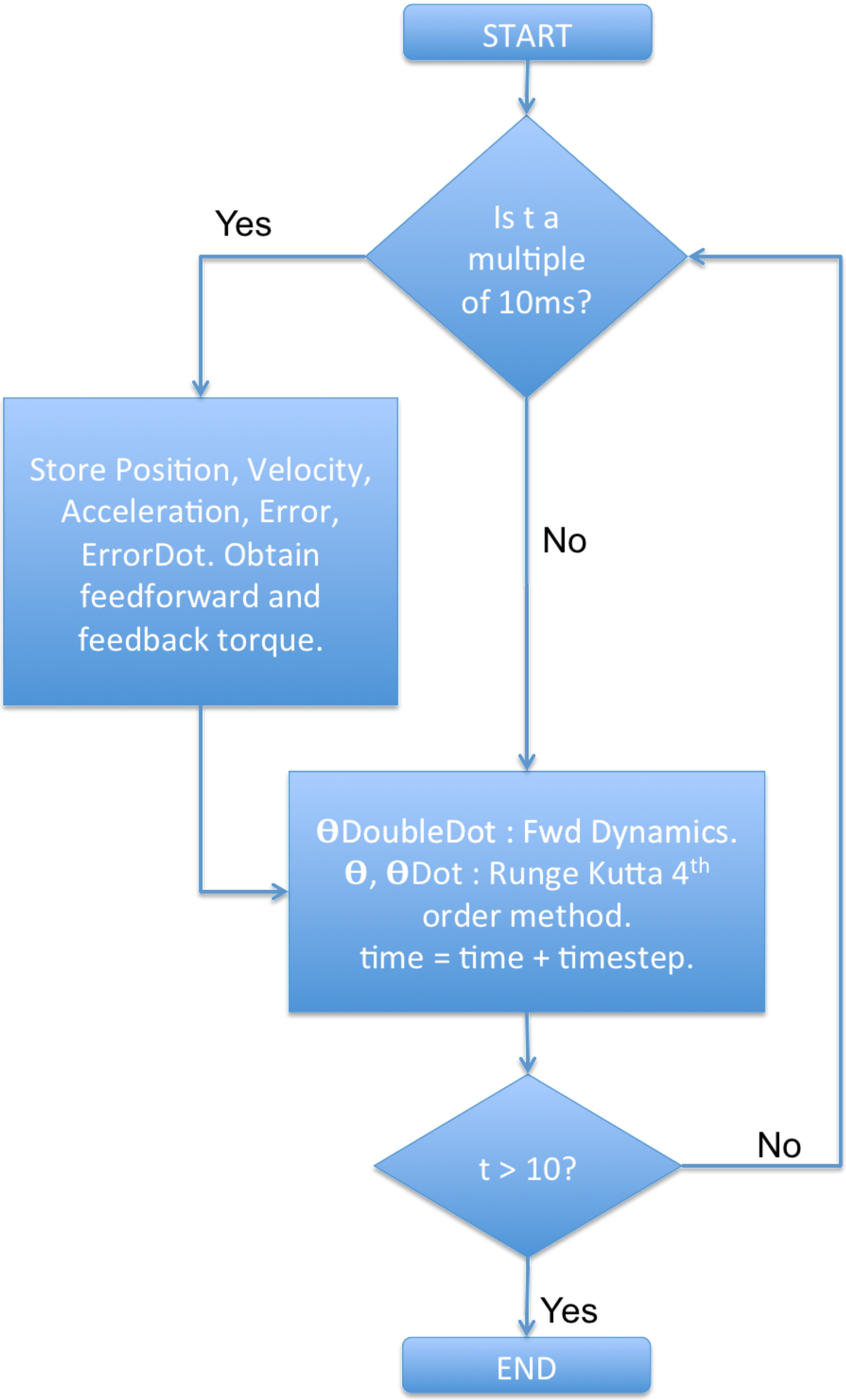


Fig 2: Flowchart for online components of simulation

Modules:

Inverse Kinematics: This function takes the end effector position and orientation (in terms of a rotation matrix) as inputs and outputs the possible branches of joint positions.

Forward Kinematics: This function takes joint position values for each joint and outputs the end effector position and orientation (rotation matrix).

Task Planning: This module is used to calculate the desired position, velocity and acceleration values, as well as the feed forward torques, offline. It uses the initial and final values of position, velocity and acceleration of each joint to create a 5th order polynomial describing the desired joint trajectory.

Inverse Dynamics: Inverse Dynamics calculates the torque required for a given state of the Stanford arm. It takes the current dynamic state of the arm (position, velocity, acceleration) and the payload mass to calculate the actual torque required to achieve this state.

Forward Dynamics: Calculation of forward dynamics involves calling the inverse dynamics function several times. This function requires the current position, velocity and torque along with the payload mass & geometry to give you the resulting acceleration.

Controller: The controller is a simple PID control that takes the error and its derivative to calculate the feedback torque. In the context of this simulation, simple PD control was effective to give small errors.

ODE Solver: This is a Runge Kutta rk45 fixed time solver I wrote.

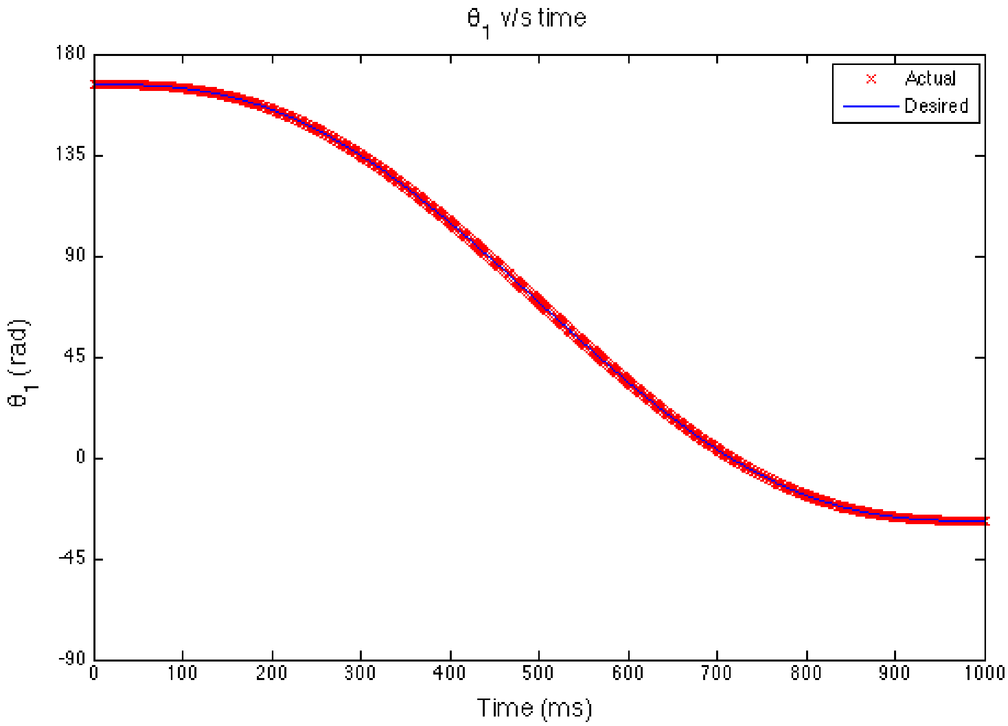
Main simulation file: This file contains the main online simulation, and also some modules such as task planning. It integrates all the modules and runs the simulation over the entire 10-sec period of simulation.

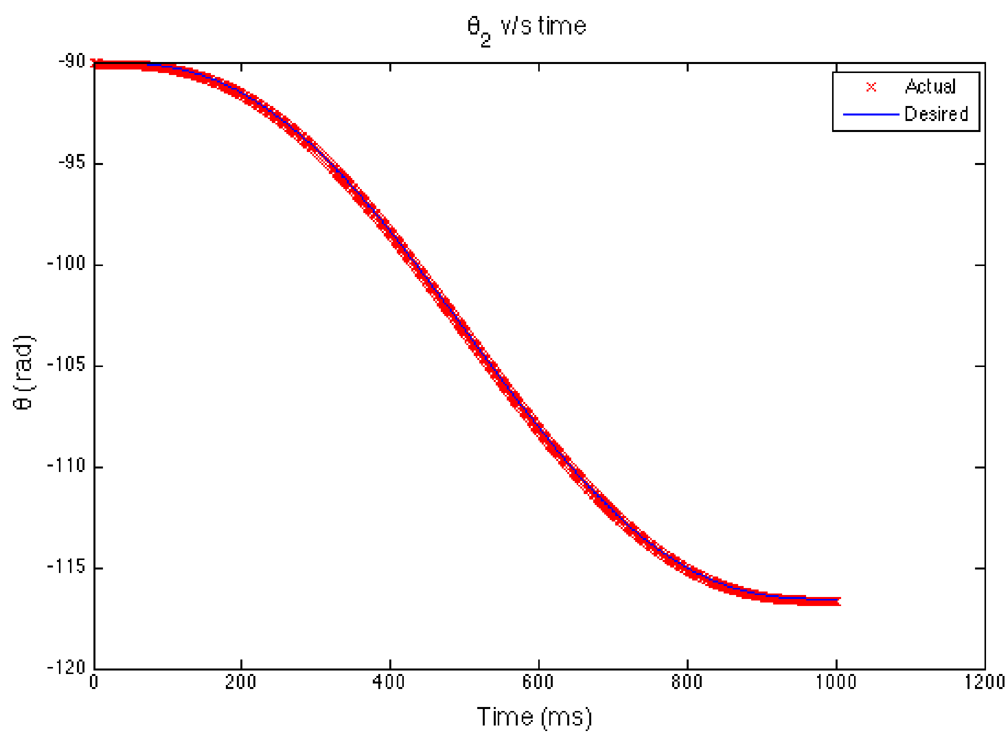
Miscellaneous: Apart from these, other smaller functions were created and called at various points during the simulation, such as the SpatialTranspose and SpatialCrossProduct functions. Also, a graphs.m file was created to produce graphs of data after the simulation was over.

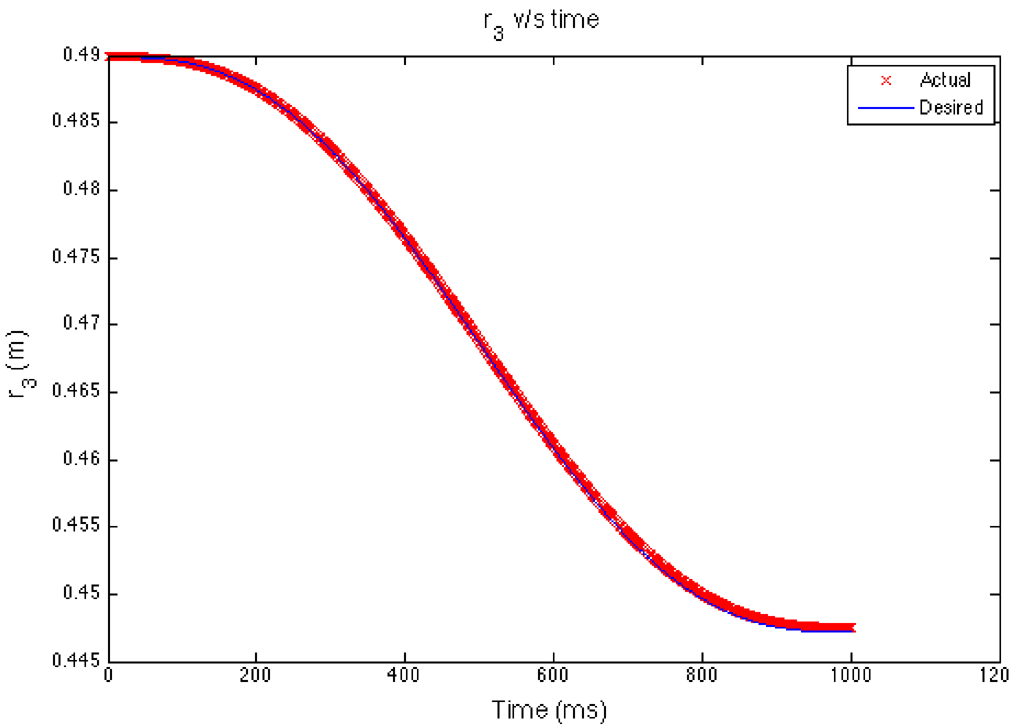
Results: Handling a point mass of unknown weight

The following is the data for the maximum unknown weight (0.4kg) to represent the maximum possible error.

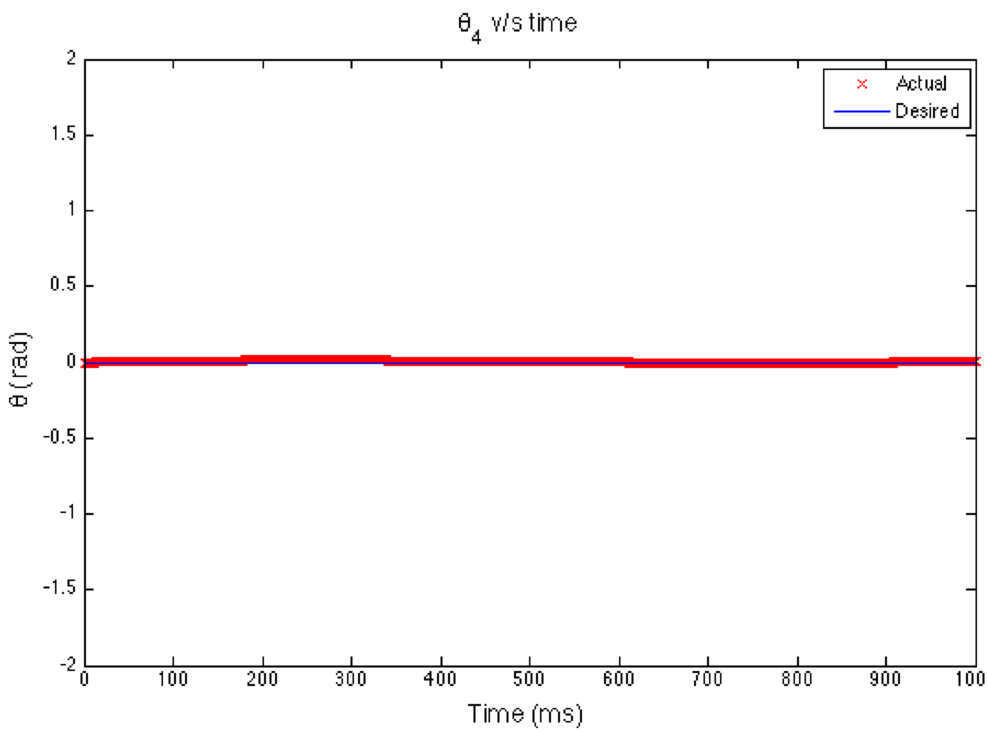
Desired & Actual Joint Kinematics:



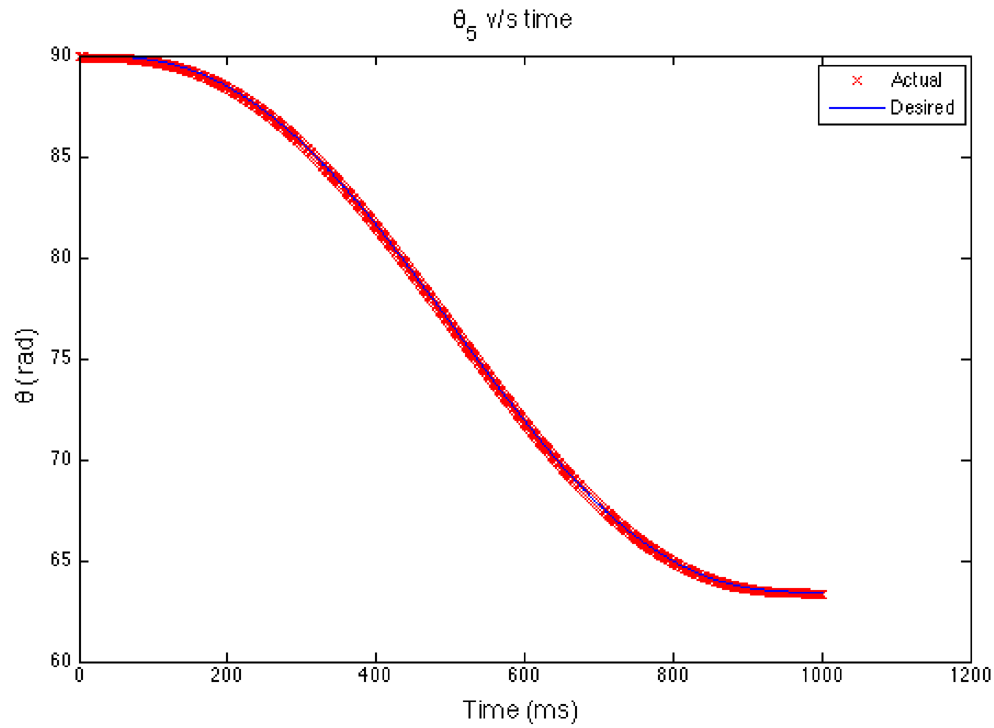


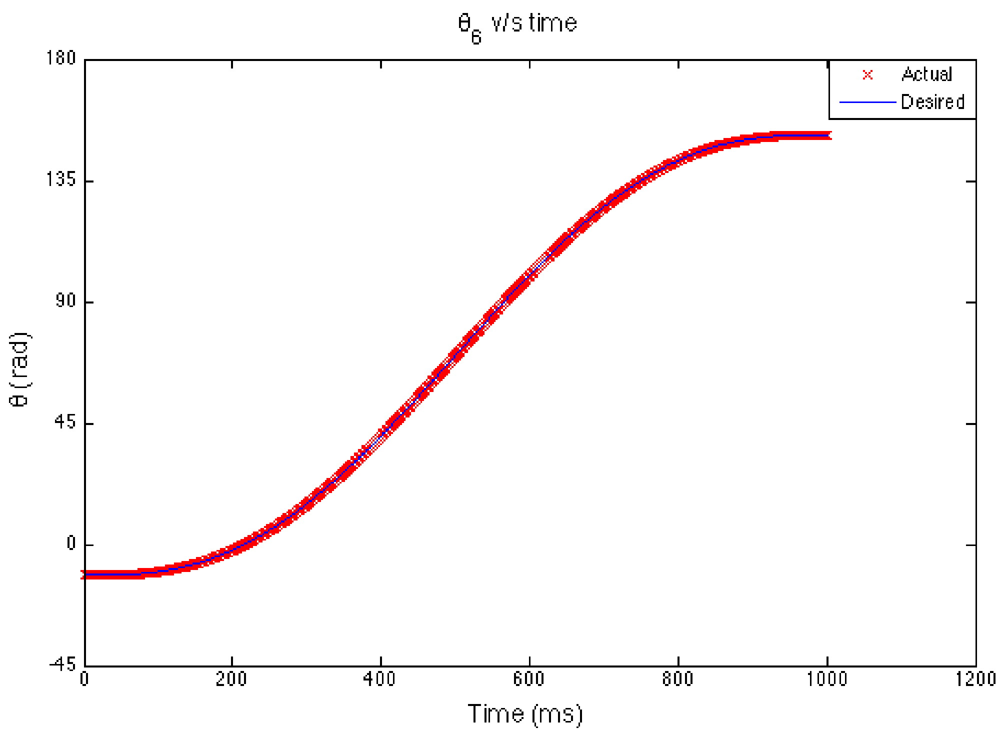


r_3 v/s time
Actual
Desired
r_3 (m)
Time (ms)

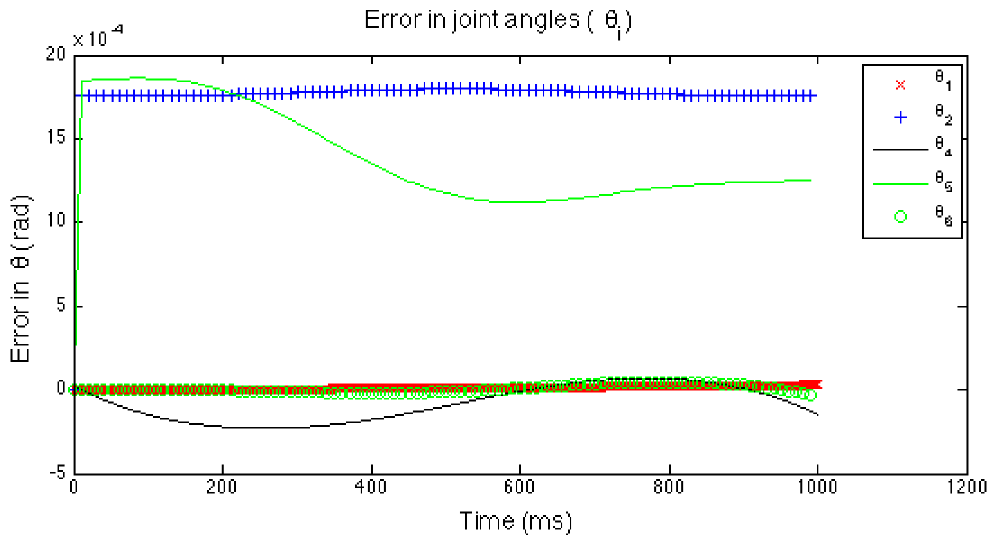


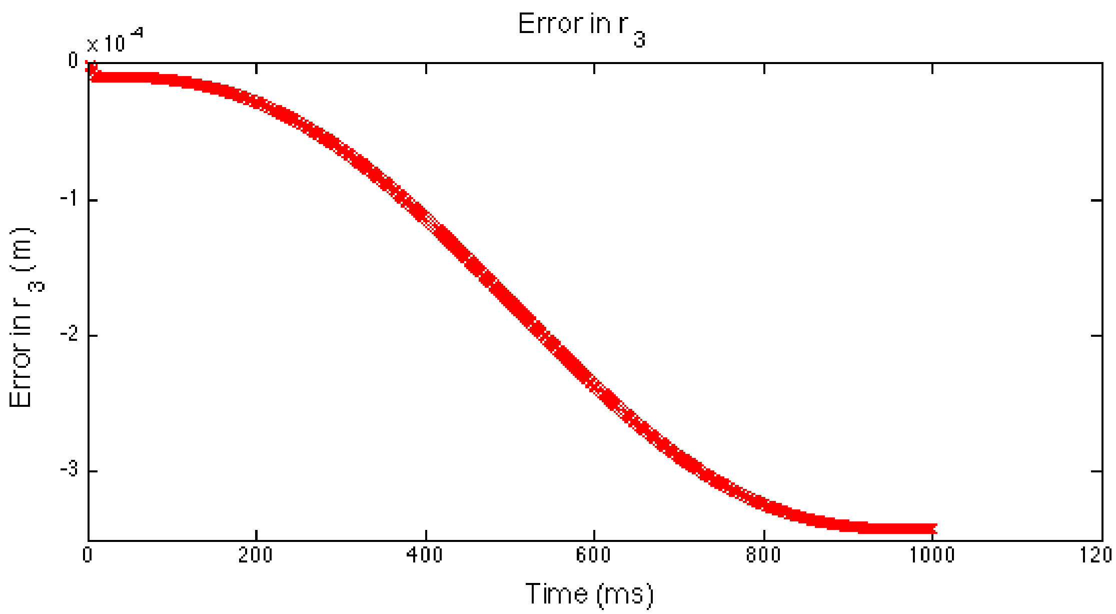
θ_4 v/s time
Actual
Desired
θ (rad)
Time (ms)


θ5 v/s time
Actual
Desired
θ (rad)
90
85
80
75
70
65
60
Time (ms)
0
200
400
600
800
1000
1200

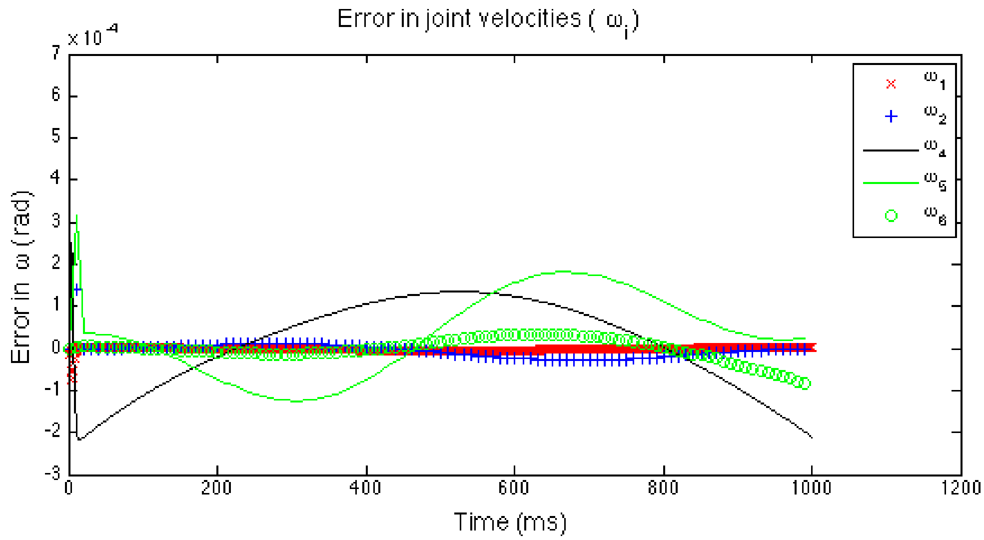

θ6 v/s time
Actual
Desired
θ (rad)
180
135
90
45
0
-45
Time (ms)
0
200
400
600
800
1000
1200

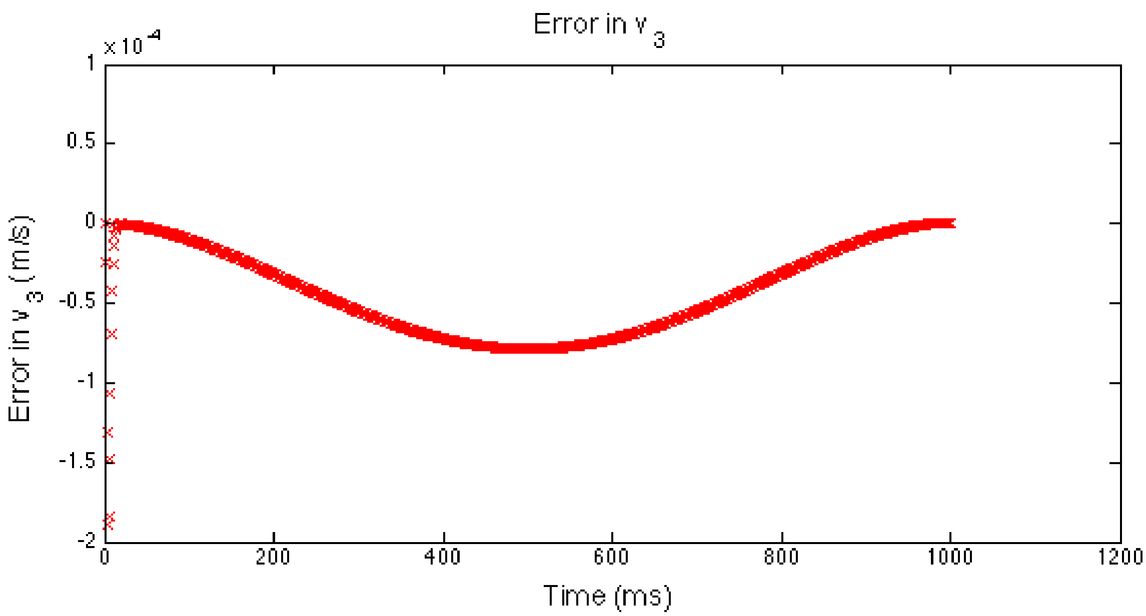
Joint Position errors (Part 1 – point mass)



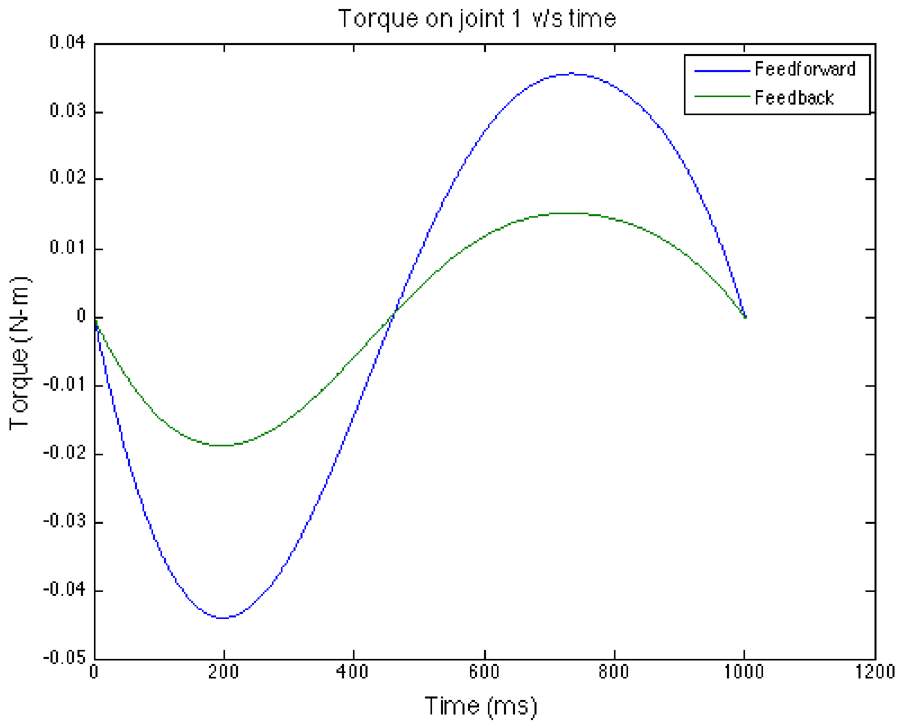


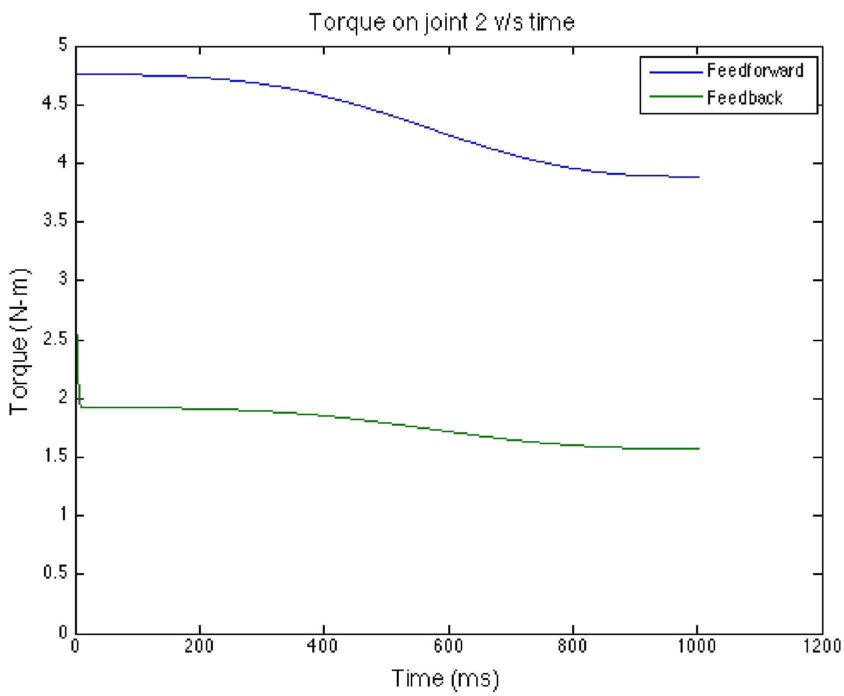
Joint Velocity errors (Part 1- point mass)

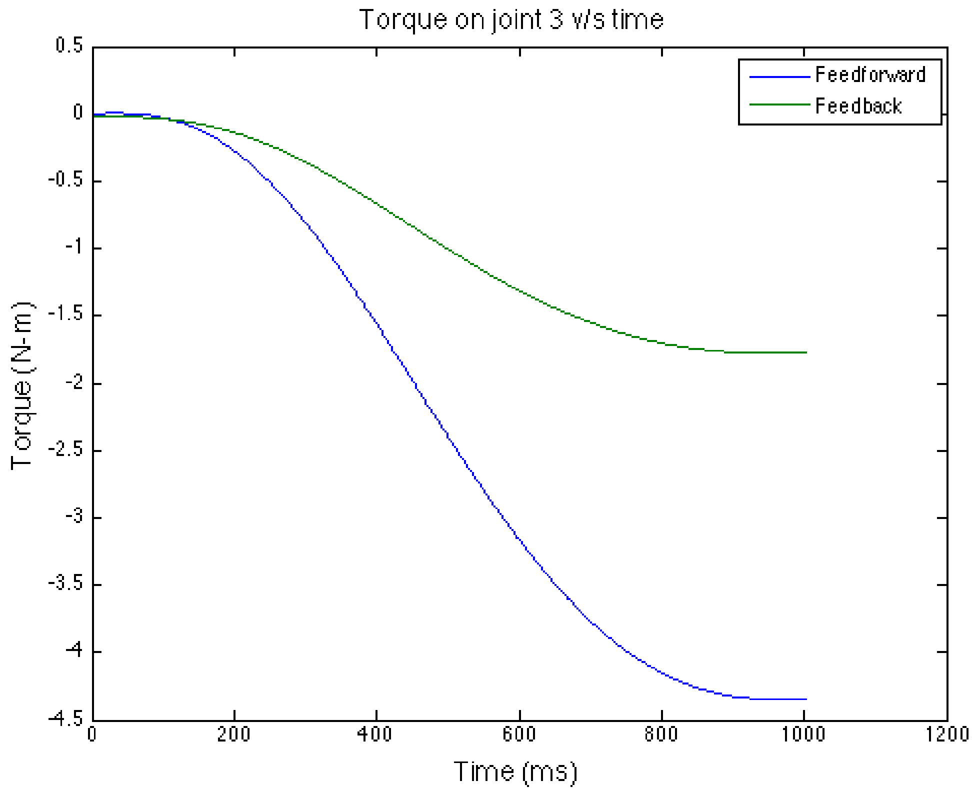




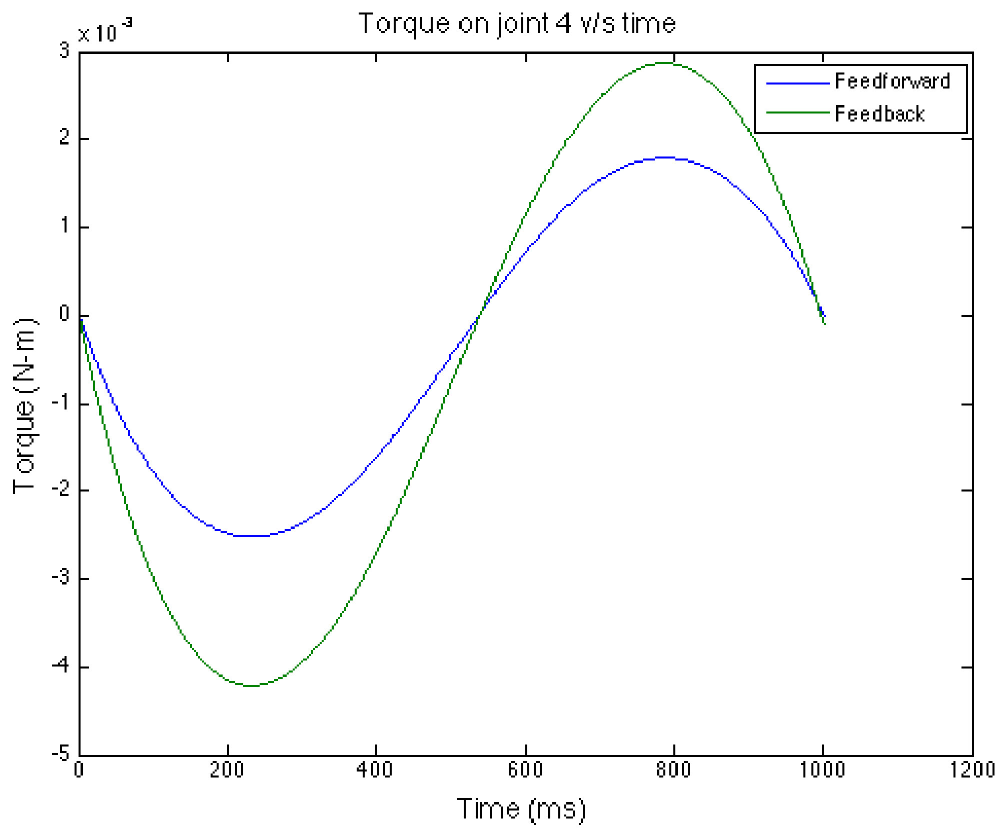
Feedforward and Feedback Joint Torques (part 1 – point mass):



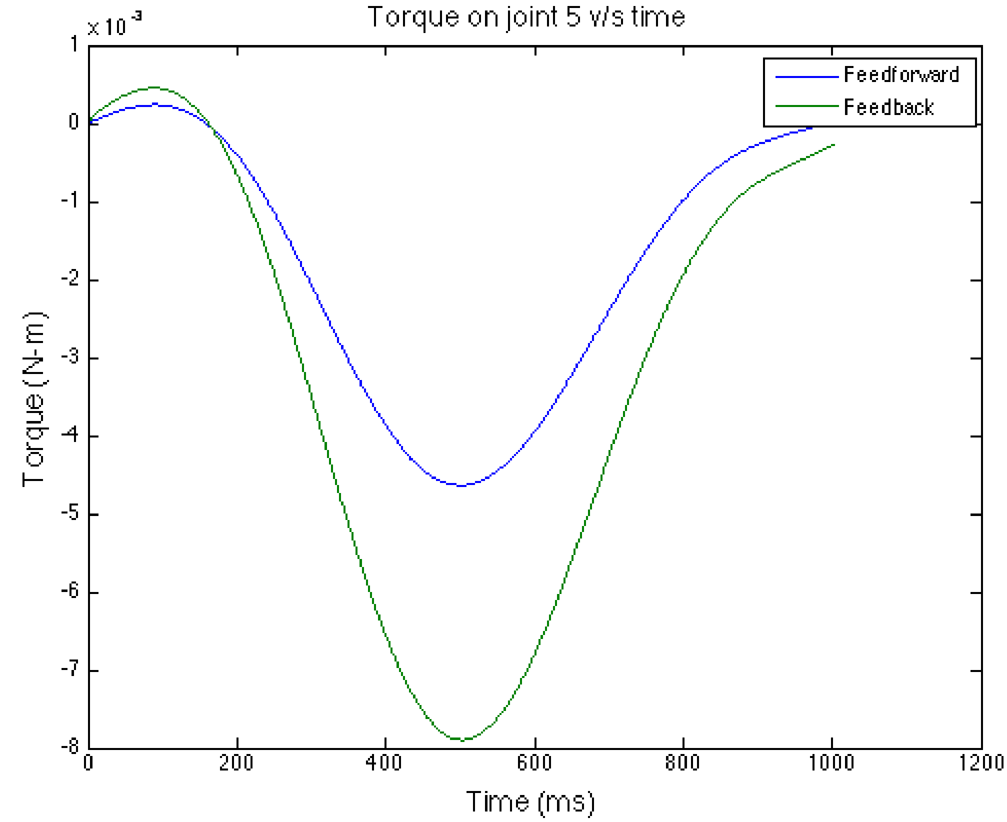


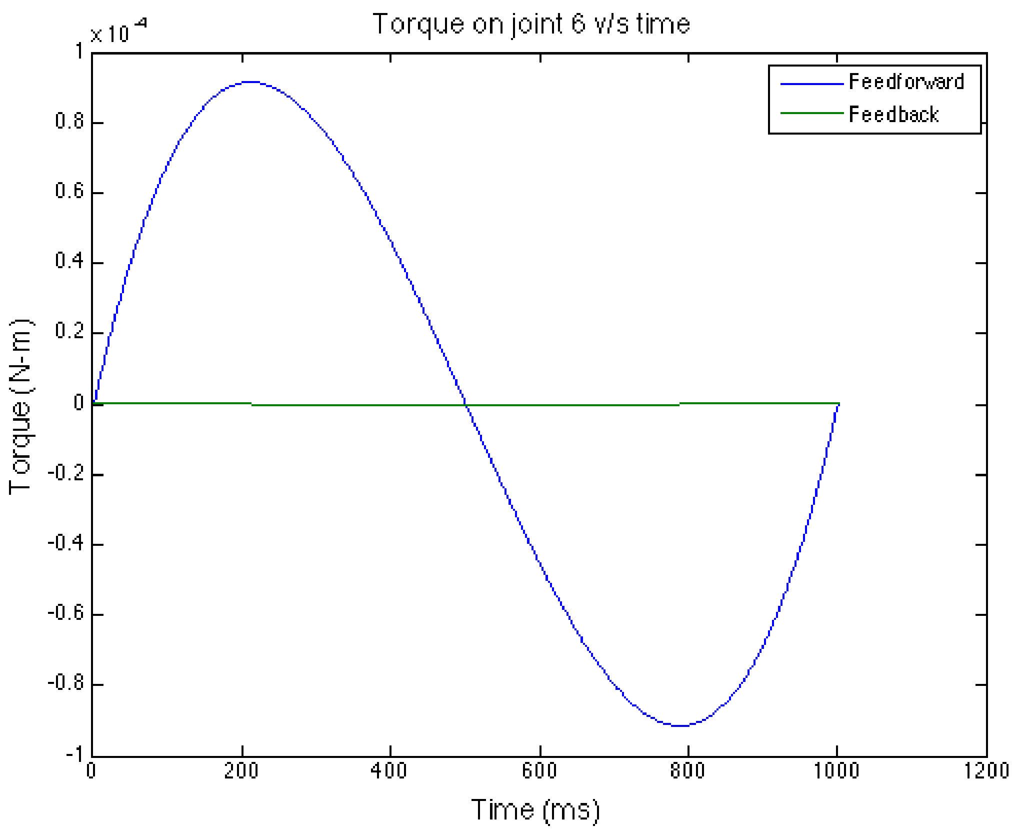


Torque on joint 3 v/s time
Feedforward
Feedback
Torque (N-m)
Time (ms)
0.5
0
-0.5
-1
-1.5
-2
-2.5
-3
-3.5
-4
-4.5
0
200
400
600
800
1000
1200



Torque on joint 4 v/s time
x 10^-3
Feedforward
Feedback
Torque (N-m)
Time (ms)
3
2
1
0
-1
-2
-3
-4
-5
0
200
400
600
800
1000
1200


Torque on joint 5 v/s time
x 10^-3
Feedforward
Feedback
Torque (N-m)
Time (ms)
1
0
-1
-2
-3
-4
-5
-6
-7
-8
0
200
400
600
800
1000
1200


Torque on joint 6 v/s time
x 10^-4
Feedforward
Feedback
Torque (N-m)
Time (ms)
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
200
400
600
800
1000
1200

Summary:

The data from these simulations gives very small errors, as is substantiated in the graphs above. Thus, it is believed that this simulation and control system is functional. The PD controller itself was enough to achieve such small errors in the simulation, and thus we can say that the integral (I) component of the PID controller was not integral to computation of feedback torque!