DESIGN AND MODELLING OF A 4DOF PAINTING ROBOT

MSc. Nilton Anchaygua A. Victor David Lavy B. Jose Luis Jara M.

Abstract

The following project has as goal the study of the kinematics, dynamics and control of a 4DOF robotics arm with an industrial application for painting. We will study different types of controls learned in an Advanced Robotics class. The tuning of the gains of our controller will be done manually; also in our design it will be considered the electric dynamics of DC motors with the purpose of using the parameters and constraints that could show up in the implementation of our robot.

INTRODUCTION

Robotics field has been one of the most interesting ones in the last years, mainly for the big applications that have been found and the ones that are in the process of discovering, from which we find a manifold going from industrial applications to domestic and entertained applications. Nevertheless, the modelling of these robots requires a detailed analysis and calculations, especially in the control part. For this first project, we have used a PID control, manually tuned, considering admissible values of torque and voltages of the actuators.

For this task, the robot has been modeled first calculating its Denavit-Hartenberg parameters; the inverse kinematics was done according to the geometric method with the desired position. The dynamics was calculated by designing our robot using the software Solidworks and was substantiated using SimMechanics from Simulink, Matlab and the Euler-Lagrange equations. After this, a compensation was performed to check our analysis and lastly the PID control is showed for a specific trajectory considering the electric dynamics from the motors and their restrains.

I. ROBOT'S DESIGN

Our robot is based in a 4DOF KUKA robot, in which every articulation in the robot are rotationals.

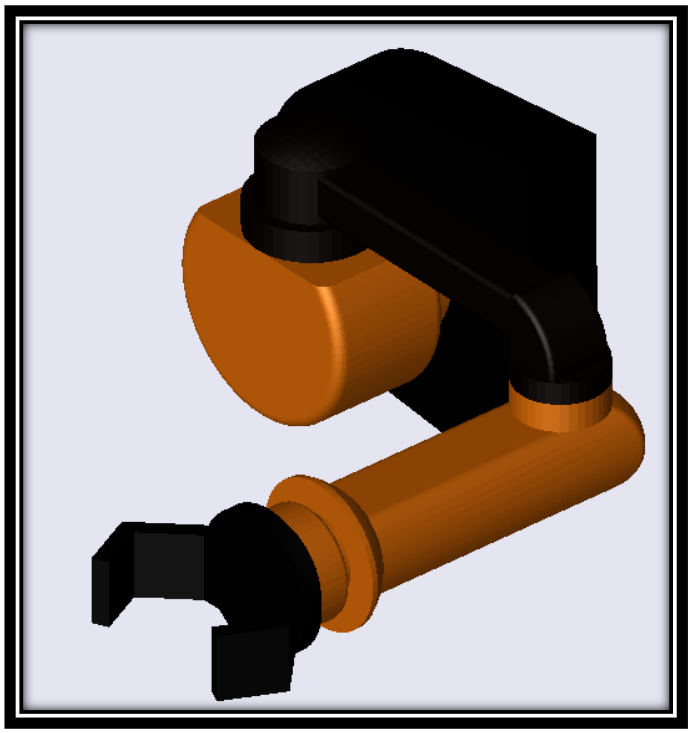


Figure 1 – 4DOF Robotics Arm

II. ROBOT'S KINEMATIC

2.1 Direct Kinematics

i	α	a	θ	d
1	$-\pi/2$	0	$q1$	$l1$
2	0	$l2$	$q2$	0
3	$\pi/2$	0	$q3$	0
4	0	0	$q4$	$l3$

Denavit-Hartenberg Parameters

2.2 Position's equations

$$ {}_{i-1}A^{i} = \begin{bmatrix} cosq_i & -cos\alpha_i * sinq_i & sinq_i * sin\alpha_i & a * cosq_i \\ sinq_i & cosq_i * cos\alpha_i & -sin\alpha_i * cosq_i & a * sinq_i \\ 0 & sin\alpha_i & cos\alpha_i & d \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ Homogeneous\ transformation\ Matrix $$

Calculation of the end-effector position

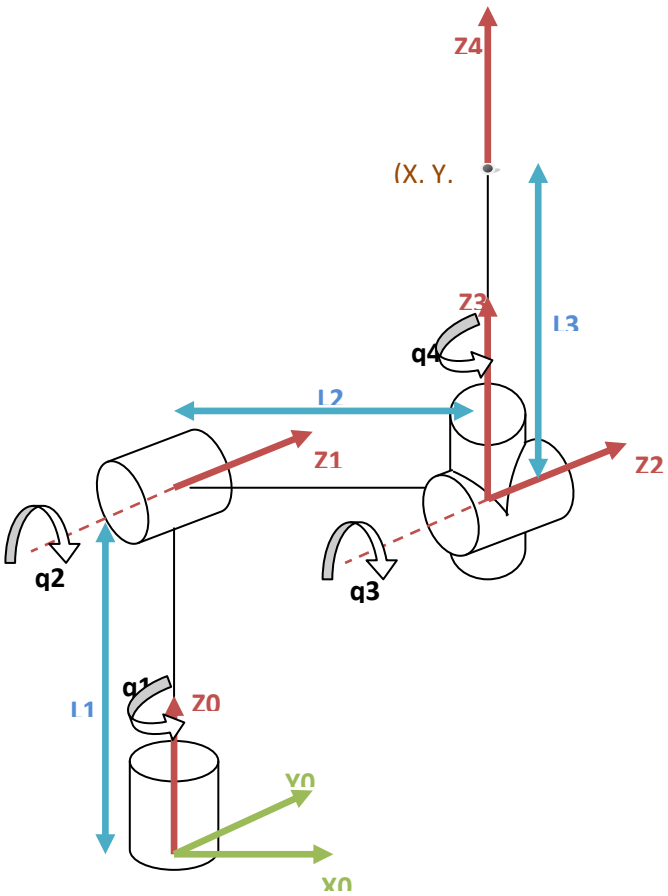
$$ {}_0T^4 = {}_0A^1 * {}_1A^2 * {}_2A^3 * {}_3A^4 $$

$$ Xp = \cos(q1) * [l3 * \sin(q2 + q3) + l2 * \cos(q2)] $$
$$ Yp = \sin(q1) * [l3 * \sin(q2 + q3) + l2 * \cos(q2)] $$
$$ Zp = l1 + l3 * \cos(q2 + q3) - l2 * \sin(q2) $$
$$ \varphi = q4 $$

It has been included a fourth parameter which corresponde to the orientation that defines our robot; this way we are improving our trajectory control.

2.3 Inverse Kinematics

To develop our inverse kinematics we have chosen the geometric method because it is much easier to analyze a specific point based on the functions of its articulations. According to this we have:



$$ \beta = arctg(\frac{z - L1}{\pm\sqrt{x^2 + y^2}}) $$

$$ \alpha = arctg(\frac{L5 * \sin(q3)}{L3 + L5 * \cos(q3)}) $$

$$ \cos(q3) = \frac{(z - L1)^2 + x^2 + y^2 - L5^2 - L3^2}{2 * L3 * L5} $$

$$ \boldsymbol{q1 = arctg(\frac{y}{x})} $$

$$ \boldsymbol{q2 = \beta - \alpha} $$

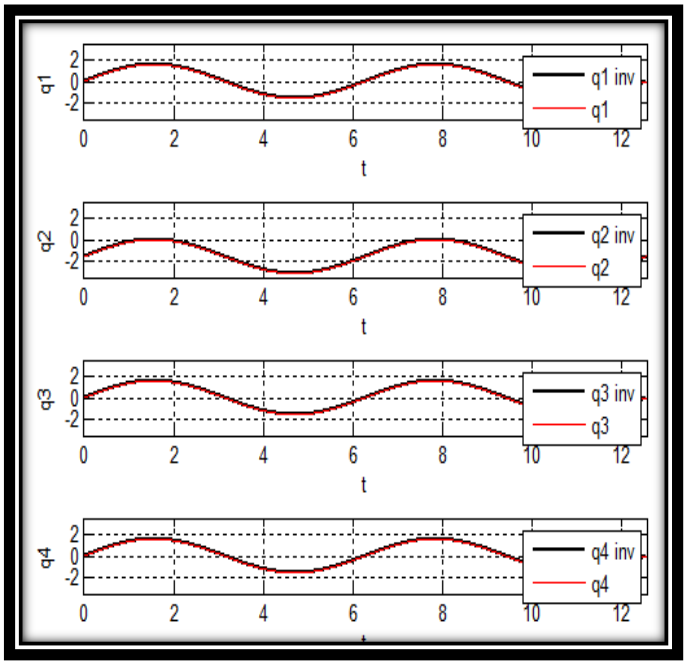
$$ \boldsymbol{q3 = arctg(\frac{\pm\sqrt{1 - \cos^2(q3)}}{\cos(q3)})} $$

$$ \boldsymbol{q4 = \varphi} $$

2.4 Inverse Kinematics Analysis

- Choose random positions:

$$ q_1 = \frac{pi}{2} * sen(t) $$
$$ q_2 = \frac{pi}{2} * sen(t) - \frac{pi}{2} $$
$$ q_3 = \frac{pi}{2} * sen(t) $$
$$ q_4 = \frac{pi}{2} * sen(t) $$



III. DIFFERENTIAL KINEMATICS

3.1 Jacobian

To calculate the Jacobian matrix, we apply the following equation:

$$J = \begin{bmatrix} \frac{\partial Px}{dq_1} & \frac{\partial Px}{dq_2} & \frac{\partial Px}{dq_3} & \frac{\partial Px}{dq_4} \\ \frac{\partial Py}{dq_1} & \frac{\partial Py}{dq_2} & \frac{\partial Py}{dq_3} & \frac{\partial Py}{dq_4} \\ \frac{\partial Pz}{dq_1} & \frac{\partial Pz}{dq_3} & \frac{\partial Pz}{dq_3} & \frac{\partial Pz}{dq_4} \\ \frac{\partial \varphi}{dq_1} & \frac{\partial \varphi}{dq_4} & \frac{\partial \varphi}{dq_3} & \frac{\partial \varphi}{dq_4} \end{bmatrix}$$

IV. ROBOT'S DYNAMIC

4.1 Dynamics using Lagrange Algorithm

In this section we will calculate the robot's dynamic using the following equation:

$$\tau = H(q)\ddot{q} + C(q,\dot{q}) + G(q)$$

$\boldsymbol{\tau}$: Torques and forces in the actuatiors
$\boldsymbol{H}$: Inertial square matrix
$\boldsymbol{C}$: Coriolis forces column matrix
$\boldsymbol{G}$: Gravity column matrix

With this equation we get the matrices by differentiating it, in which we get the following equations:

- Matrix H

$$H = \sum_{i=1}^{n} ({J_i^L}^T m_i J_i^L + {J_i^A}^T I_i J_i^A)$$

$J_i^L = Individuals\ Jacobians\ respect\ to\ the\ center\ of\ mass$
$m_i = Mass\ of\ each\ link$
$I_i = Inertial\ moment\ of\ each\ link$

- Matrix C

Using Christoffel symbols starting with the inertial tensor.

$$h_{ijk} = \frac{\partial H_{ij}}{\partial q_k} - \frac{1}{2}\frac{\partial H_{jk}}{\partial q_i}$$

- Matrix G

$$G = -\sum_{i=1}^{n} m_j g^T J_{L_i}^j$$

Where:

g = Gravity vector with respect to the base system

4.2 Dynamic using SimMechanics

One of the simulation software we have used is the tool SimMechanics from Matlab, it simulates our robot in 3D providing physical features for an adequate control, the block is as follows:

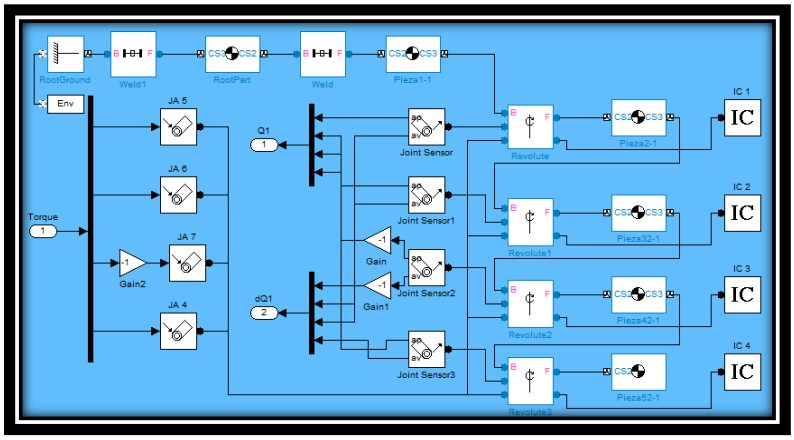


Figure 4.1 – Block in SimMechanics

4.3 Dynamics through Simulink

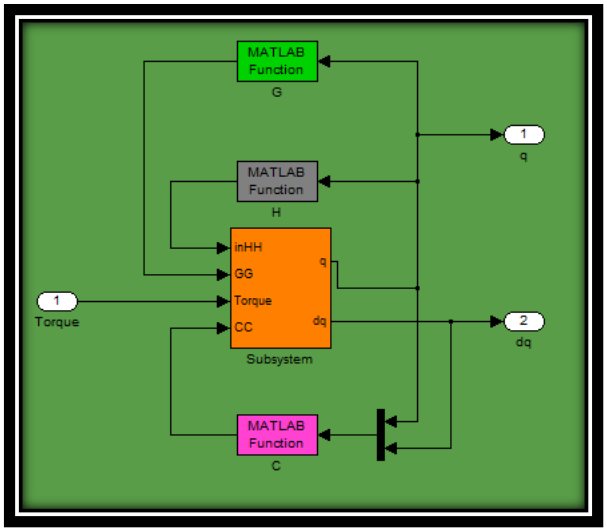


Figure 4.2 – Block in Simulink

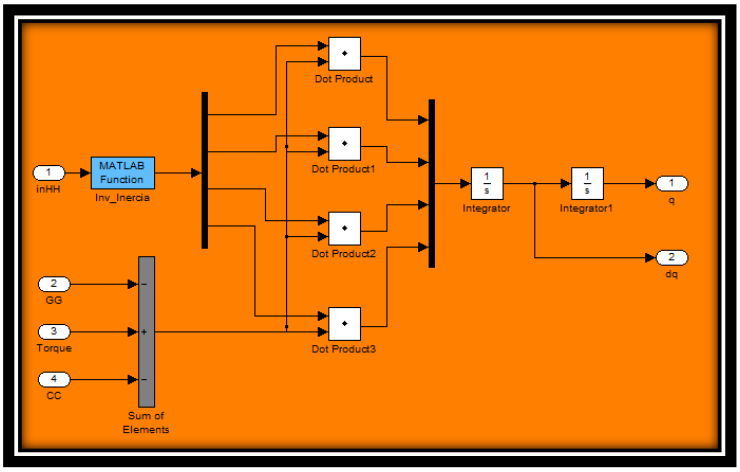


Figure 4.3 – Sub-Block in Simulink

V. ROBOT'S CONTROL

5.1 Gravity compensation

As a first step for controlling the robot, we performed a gravity compensation, so in this way we can verify that our algorithms were correct.

For this, we used the SimMechanics block, showed as follows.

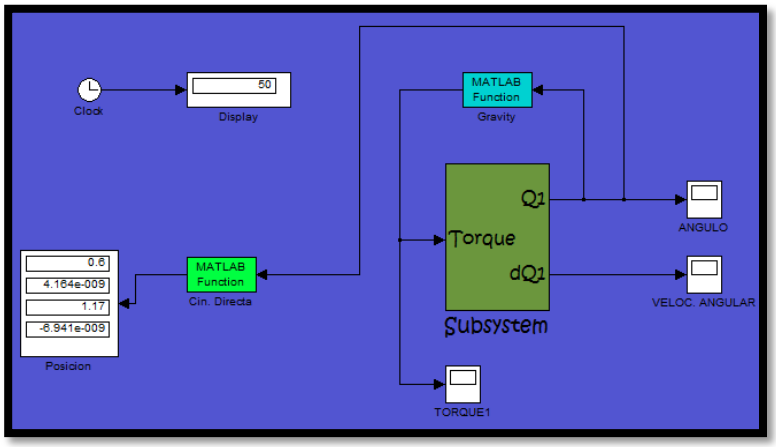


Figure 5.1 – Gravity compensation in Simechanics

5.2 PD control for regulacion

By succeeding in checking our dynamics equations are correct, we start by implementing our different control techniques. As our first control technique we will use PD control with gravity compensation. For this type of control, gravity compensation acts as a correction bias, manually tuning and finding the correct values for K which belong to the following equation:

$$\tau = K_p(q_d - q) - K_{v(\dot{q})} + G(q)$$

This type of control is often utilized as a set-point regulator, for example q_d = constant.
We have the following control block:

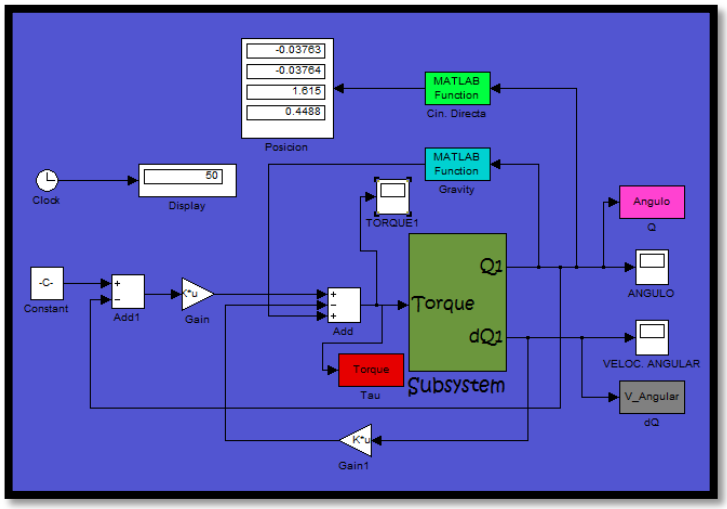


Figure 5.2 – PD Control for regulation

5.3 PID Control

An integral action has to be added to the PD control to compensate gravity forces. This PID regulator has the following formula:

$$\tau = K_p(q_d - q) + K_I \int f(q_d - q)dt - K_v\dot{q}$$

Where:

If $f(q_d - q) = q_d - q, we\ have\ PID\ control$

If $K_I \int(-\dot{q})\ \ es\ added, we\ have\ PI^2D\ control$

If $f(.) = \tanh(.)$, $we\ have\ PD + non\ linear\ integral\ control$

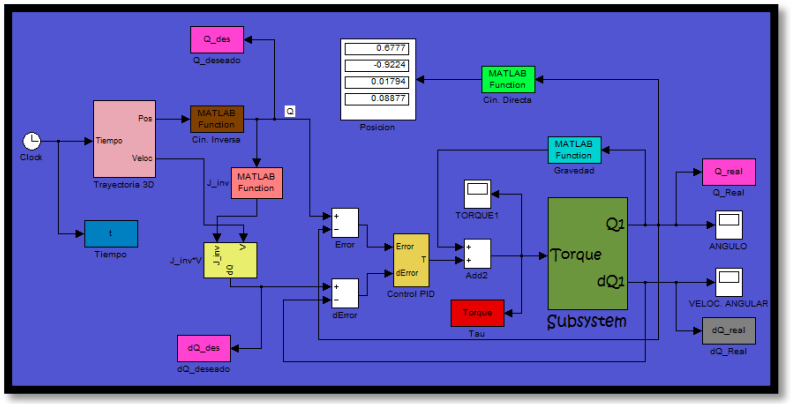


Figure 5.3 –PID Control with manual tuning

5.4 Control using neural networks

Many times calculating the inverse kinematics is hard to achieve; because of that the use of genetic algorithms to obtain those parameteres is benefitial in some cases.

Having the direct kinematics, we créate a Matlab program that will generate the knowledge charts and will perform the training process of the neural network.

The features of this net are the following:

- Multilayer using backpropagation training
- 3 input values X, Y, Z
- 2 hidden layers of 15 neurons each and 1 output layer with 4 neurons each; each output represents the articulations
- Using the gradient descent algorithm with a variable learning rate between 0.001 and 0.01

Evaluation chart:

Input			Direct Kinematics				Neural Network			
X	Y	Z	Q_1	Q_2	Q_3	Q_4	Q_1	Q_2	Q_3	Q_4
1.45	0	0.32	0	0	pi/2	0.17	0	1e-4	pi/2	0.17
1.43	0	0.57	0	0.17	pi/2	0.34	0	0.17	pi/2	034
1.36	0	0.81	0	0.34	pi/2	0.52	0	0.35	pi/2	0.52
1.25	0	1.04	0	0.52	pi/2	0.70	0	0.52	pi/2	0.70
1.11	0	1.25	0	0.70	pi/2	0.87	0	0.70	pi/2	0.87
0.93	0	1.43	0	0.87	pi/2	1.04	0	0.87	pi/2	1.05
0.72	0	1.57	0	1.04	pi/2	1.22	0	1.04	pi/2	1.22
0.50	0	1.68	0	1.22	pi/2	1.39	0	1.22	pi/2	1.40
0.25	0	1.75	0	1.39	pi/2	1.57	0	1.40	pi/2	1.57
0	0	1.77	0	1.57	pi/2	1.74	0	1.57	pi/2	1.74

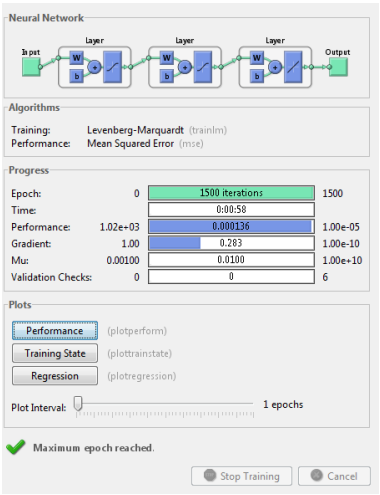


Figure 5.4 – Net training

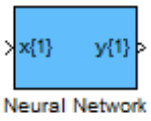


Figure 5.5 – Net simulation block

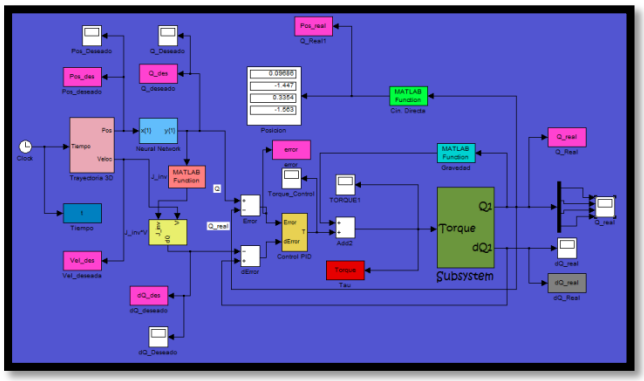


Figure 5.6 – Control using genetic algorithms

5.5 Fuzzy Logic Control

VI. RESULTS OBTAINED FOR POSITION CONTROL

- PD Control with regulation

Final set point will have the following angle articulations:

$$q_1 = \frac{pi}{2}$$

$$q_2 = -\frac{pi}{3}$$

$$q_3 = \frac{pi}{4}$$

$$q_4 = \frac{pi}{5}$$

With this values, the measured error has the following graphic

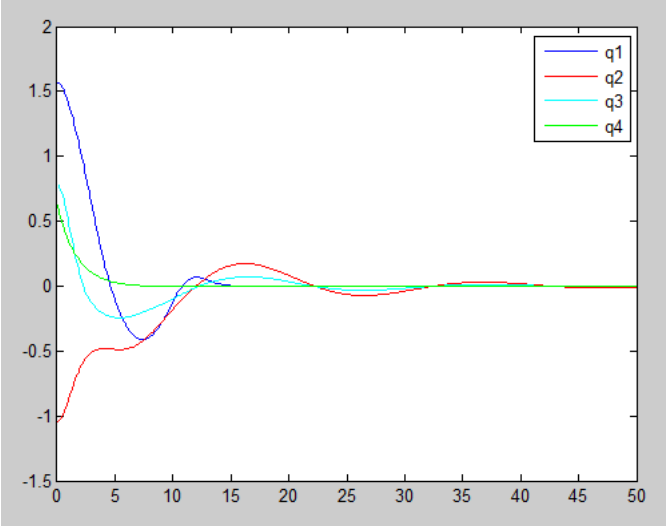


Figure 6.1 – Error measurement with PD Control

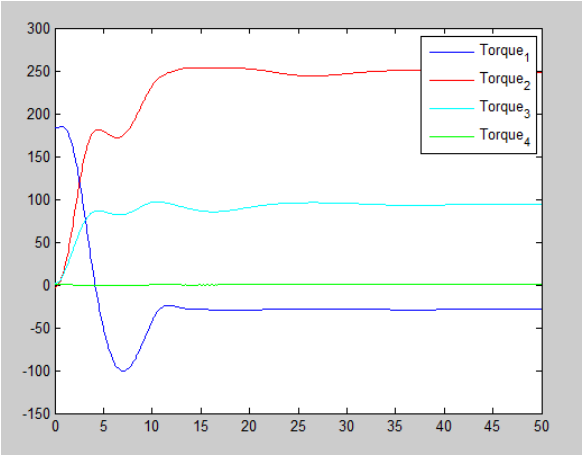


Figure 6.2 – Torque calculated for control

Now we will simulate in the Simulink block

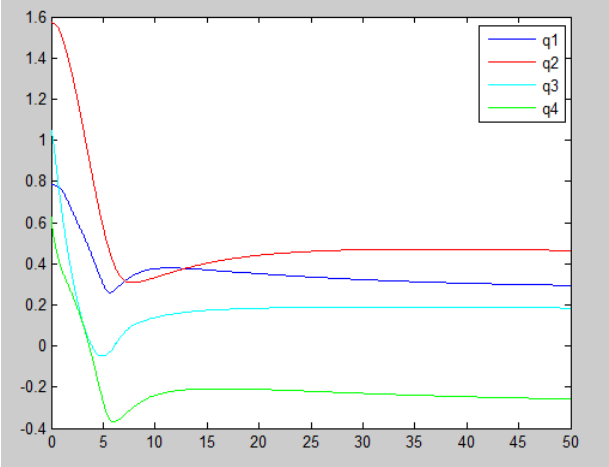


Figure 6.3 – Simulink Articulations

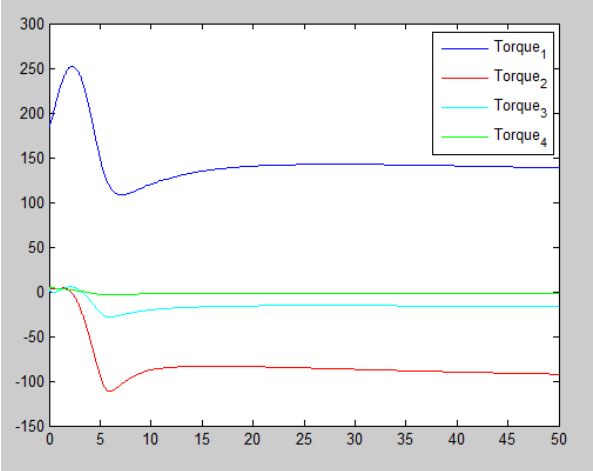


Figure 6.4 – Torque obtained for control