DOKUZ EYLÜL UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

MOTION OPTIMIZATION OF A 6 AXES ROBOT MANIPULATOR USED FOR MATERIAL HANDLING PURPOSES VIA DYNAMIC PROGRAMMING

by
Umut ÇAKIR

June, 2014
İZMİR

MOTION OPTIMIZATION OF A 6 AXES ROBOT MANIPULATOR USED FOR MATERIAL HANDLING PURPOSES VIA DYNAMIC PROGRAMMING

A Thesis Submitted to the
Graduate School of Natural and Applied Sciences of Dokuz Eylül University
In Partial Fulfillment of the Requirements for the Degree of Master of Science in Mechanical Engineering, Machine Theory and Dynamics Program

by
Umut ÇAKIR

June, 2014
İZMİR

M.Sc THESIS EXAMINATION RESULT FORM

We have read the thesis entitled **"MOTION OPTIMIZATION OF A 6 AXES ROBOT MANIPULATOR USED FOR MATERIAL HANDLING PURPOSES VIA DYNAMIC PROGRAMMING"** completed by **UMUT ÇAKIR** under supervision of **ASSIST. PROF. DR. AYTAÇ GÖREN** andwe certify that in our opinion it is fully adequate, in scope and in quality, as a thesisfor the degree of Master of Science.

.....

Assist. Prof. Dr. Aytaç GÖREN

Supervisor

.....

Assoc.Prof.Dr. Zeki KIRAL

(Jury Member)

.....

Assist. Prof. Dr. Özgen BAŞER

(Jury Member)

Prof.Dr. Ayşe OKUR

Director

Graduate School of Natural and Applied Sciences

ACKNOWLEDGEMENT

I would like to thank to my thesis consultant Assist. Prof. Dr. Aytaç GÖREN helping me in completing my Master Thesis with his guidance during the development of this study. Furthermore I am also grateful to my parents who support me without depriving me of anything that I have needed so far.

Umut ÇAKIR

MOTION OPTIMIZATION OF A 6 AXES ROBOT MANIPULATOR USED FOR MATERIAL HANDLING PURPOSES VIA DYNAMIC PROGRAMMING

ABSTRACT

Due to the increasing cost of energy besides the enviromental concerns, energy consumption is one of the hot topic in today's world. In this context, companies are searching for the ways to reduce their energy consumptions and their Carbon footprint. For a manufacturing company, one method to accomplish these is to increase their process efficiency. In this thesis, the production cell which contains a robot manipulator used for material handling purposes is tried to be modelled in computer environment, using dynamic programming over the computer model, system parameters increasing production capacity in addition to reducing the energy consumption and improvements depending on these are presented.

Keywords: Robot manipulator, movement optimization, dynamic programming, optimal control

MALZEME TAŞIMA AMACIYLA KULLANILAN 6 EKSENLİ ROBOT MANİPULATORÜN DİNAMİK PROGRAMLAMA İLE HAREKET OPTİMİZASYONU

ÖZ

Çevresel kaygıların yanında artan enerji maliyetleri sebebiyle, enerji tüketimi günümüz dünyasındaki en önemli konu olmuştur. Bu bağlamda, şirketler enerji tüketimlerini ve Karbon ayak izlerini düşürmenin yollarını aramaktadırlar. Bir imalat firması için bunu başarmanın bir yolu proses verimliliğini artırmaktır. Bu tezde, malzeme taşıma amacıyla kullanılmakta olan robot manipulatörün bulunduğu hücredeki çalışma şartları bilgisayar ortamında modellenmeye çalışılmış, oluşturulan model üzerinden dinamik programlama kullanılarak enerji tüketimini azaltan ve kapasite artışı sağlayan optimize edilmiş sistem parametreleri ve buna bağlı iyileştirmeler sunulmuştur.

Anahtar kelimeler: Robot manipulatör, hareket optimizasyonu, dinamik programlama, optimum kontrol

CONTENTS

Page

THESIS EXAMINATION RESULT FORM ii
ACKNOWLEDGEMENTS iii
ABSTRACT iv
ÖZ v
LIST OF FIGURES viii
LIST OF TABLES ix

CHAPTER ONE – INTRODUCTION 1

1.1 Industrial Robots 2
1.2 Problem Statement 7

CHAPTER TWO – KINEMATIC ANALYSIS OF ROBOT MANIPULATOR 9

2.1 Kinematic Analysis of Robot Manipulator 9
2.1.1 Denavit Hartenberg Convention 10
2.2 Jacobian Matrices & Robot Dynamics 13
2.3 Robot Dynamics 14
2.4 Equation of Motion 15
2.5 Dynamic Programming 16
2.5.1 Optimal Control Law 17
2.6 Computed Torque Controller 18
2.7 PID Controller 21
2.7.1 Tuning PID Controller Parameters 22

CHAPTER THREE – METHODOLOGY 23

3.1 General Steps Followed 23
3.2 Inverse Kinematic Analysis of Robot Manipulator 25

3.3 Dynamic Analysis of Robot Manipulator 29

CHAPTER FOUR – CALCULATIONS 31

4.1 Kinematic Analysis of Staubli TX90 31
4.2 Dynamic Modelling of Robot Manipulator 33
4.3 Control Equations of Robot Manipulator 35
4.4 Selection of PID Parameters 37
4.5 Optimization with Dynamic Programming 39
4.6 Results of Calculations 43

CHAPTER FIVE – CONCLUSIONS 47

REFERENCES 48

APPENDICES 50

LIST OF FIGURES

Page

Figure 1.1 Symbolic representation of robot joints 2
Figure 1.2 Articulated robot manipulator structure, a) Articulated robot manipulator structure, b) Articulated robot manipulator structure 4
Figure 1.3 Spherical robot structure and its workspace 4
Figure 1.4 Cylindrical geometry robot manipulators, a) Cylindirical robot, b) Cylindrical robot manipulator workspace 5
Figure 1.5 SCARA geometry robot manipulator, a) Mitsubishi RH Series SCARA, b) SCARA robot manipulator workspace 5
Figure 1.6 Cartesian geometry robot manipulators, a) Schunk Uniplace 2300,b) Cartesian robot workspace 6
Figure 1.7 Wrist of robot manipulator 6
Figure 1.8 A photo of IRB 360 FlexPicker 7
Figure 1.9 Current process environment 8
Figure 2.1 Denavit Hartenberg frame assignment 10
Figure 2.2 Coordinate frames satisfying assumptions DH1 and DH2 11
Figure 2.3 PID computed-torque controller representation 20
Figure 2.4 PID tuning method ultimate gain and ultimate period definition 22
Figure 3.1 Current path followed by the end-effector 24
Figure 3.2 Optimized path followed by the end-effector 24
Figure 3.3 Path mate settings for inverse kinematic analysis. A is for the definition of the point on handling part. B is the path desired to be followed by end effector. C is the path constraint and D is the total length of the path 25
Figure 3.4 Path motor settings for inverse kinematic analysis. A is the tab used to select the path mate. Section B & C automatically appear on the screen. D is the tab on which the way of movement is adjusted 26
Figure 3.5 Velocity definition of the end effector in Solidworks motion analysis 26
Figure 3.6 End effector motion profile for current case, a) Velocity vs time, b) Acceleration vs time, c) Jerk vs time 27
Figure 3.7 Velocity definition of the end effector in Solidworks for optimized motion analysis 28

Figure 3.8 End effector motion profile for optimized case, a) Velocity vs time, b) Acceleration vs time, c) Jerk vs time 29
Figure 4.1 Staubli TX90 series industrial robots 31
Figure 4.2 Assigned coordinate frames on Staubli TX90 32
Figure 4.3 PID type computed torque controller scheme used in this work 37
Figure 4.4 System output versus time 38
Figure 4.5 Error in system output versus time 38
Figure 4.6 System outputs when $K_p = 500$ while $K_i = K_d = 0$. Output is oscillated with a period of 0.005sn 39
Figure 4.7 Joint angle vs time corresponding to initial path (blue) and optimized path (green) 43
Figure 4.8 Joint velocity vs time corresponding to initial path (blue) and optimized path (green) 44
Figure 4.9 x, y, z coordinate change of end effector tip vs time for geometrically optimized path (blue) and energetically optimized path (green)..... 45

LIST OF TABLES

Page

Table 2.1 PID tuning parameters with Ziegler-Nichols method 22
Table 3.1 Motion parameters for current case 27
Table 3.2 Motion parameters for optimized case 28
Table 4.1 DH parameters of Staubli TX90 32
Table 4.2 Error in final end effector point after optimization 45
Table 4.3 Results containing data regarding to the initial case and optimized case .. 46

CHAPTER ONE
INTRODUCTION

An official definition of robot by Robot Institute of America is “A robot is a reprogrammable, multifuntional manipulator designed to move material, parts, tools or specialized devices through variable programmed motions for the performance of a variety of tasks”.

According to ISO (1994), “An industrial robot is an automatically controlled, reprogrammable, multi-purpose, manipulator programmable in three or more axes, which may be either in place or mobile for use in industrial automation applications”.

The most critical point of the above definitons is the property of robots being reprogrammable. This property provides flexibility from one application to another and as a result of this, robots are prefered extensively in different industries such as automotive, aerospace, military, space exploration, hazardeous environments and working conditions constituting risks to worker health and safety.

In automotive industry, robots are frequently used in painting, welding and material handling applications.

Compared to humans, robots can work fast and accurate in the applications requiring fast and repetitive cycles and/or applications necessitating high precission and stability. Especially the work conditions posing risk to workers health and safety, robots can be the best solution to the application in reducing risks. Taking all the above facts into account, robots play an important role in both increasing productivity, quality and diminishing threat over workers health with a replacement in the place of human.

In this chapter, basic types of robots, their workspace, control method and actuator types are represented.

1.1 Industrial Robots

Robot manipulators are composed of links connected by joints to form kinematic chains. There are two types of joints used in robot manipulators. These are revolute joints and prismatic joints. Revolute joints are the joint type on which rotary motion in between two links exists. Prismatic joints are the joint type in which there is linear motion between two links.

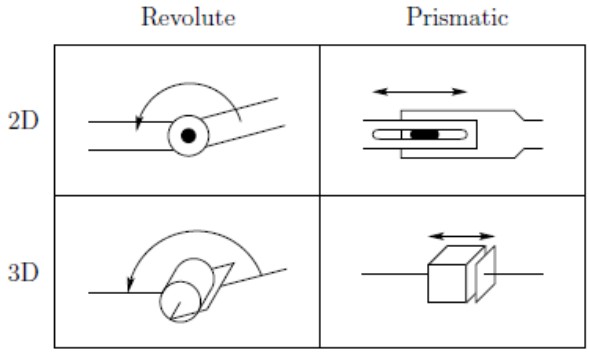


Figure 1.1 Symbolic representation of robot joints (Spong, Hutchinson and Vidyasagar, 2006)

Robot manipulators can be classified into several groups in terms of the power source used, kinematic structure, application type and control method. By considering the type of power source, hydraulic, pneumatic and electrically powered robots are used in industry. Hydraulic robots are suitable for the applications involving heavy load carrying. Due to the hydraulic equipments used, they are very noisy and occupy more space. Robots power with pneumatic source are not used extensively in industry due the limited capability in precise control. Electrical powered robots are the popular types of robot manipulators used in industry thanks to the servo motors being cheaper, cleaner and quiter in their design.

In terms of their method of control, there are two types of motors. One of them is servo controlled robots. The other is nonservo controlled motors. Old fashioned robots utilize from nonservo control which includes open loop control. Due to the nature of this open loop control, they are only prefered in material transfer applications. The servo controlled robot manipulators use closed loop computer

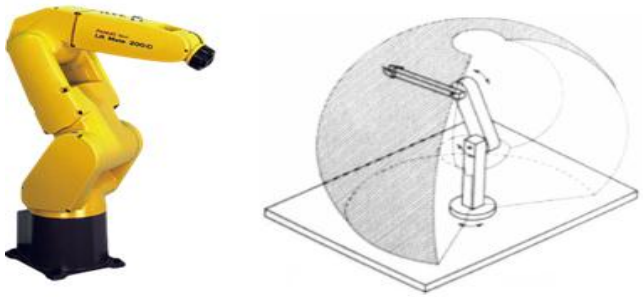
control to determine their motion and they tend to be reprogrammable and multifuctional.

Servo controlled robots are categorized into two groups in terms of the method that the controller uses for the guidance of the end effector. One of them is point to point robot via teach pendant in which a set of points are taught to the robot manipulator and there is no furher control of the path of the end effector follows. The other one is continuous path robots which includes the control of the end effector over the entire path. With this type of control, straight weldment pattern and contour machining operations can be handled utilizing from velocity/acceleration control.

Robot manipulators can also be classified into assembly and nonassembly robots with respect to their application area. Assembly robots are used in welding, painting and material handling applications whereas non-assembly robots are suitable for part insertion, screwing/unscrewing operations and machining operations.

The type of geometry is another criteria for catergorizing the robot manipulators. In terms of the kinematic configuration, robots are classified into serial link manipulator and parallel manipulator. Serial link manipulators are open kinematic chain robots and are further grouped into articulated, spherical, scara, cylindircal and cartesian robots with respect to their joint types.

The articulated robot manipulators (Figure 1.2(a)) have three revolute joint in its design. The first joint is perpendicular to the other two joints and it is mainly used in adjusting the orientation of the wrist joint group (Figure 1.7) on the horizontal plane. Its workspace (Figure 1.2(b)) resembles a sphere and this kind of coverage makes these robots suitable to the most of the applications.



(a) (b)

Figure 1.2 Articulated robot manipulator structure, a) Fanuc LR Mate 200iC (courtesy of Fanuc Robotics), b) Articulated robot workspace (Sciavicco & Siciliano, 2004)

The spherical geometry consists of two revolute and one primatic joints and is represented by RRP geometry. As the wrist group moves away from the base link, positional accuracy reduces due to deflection.

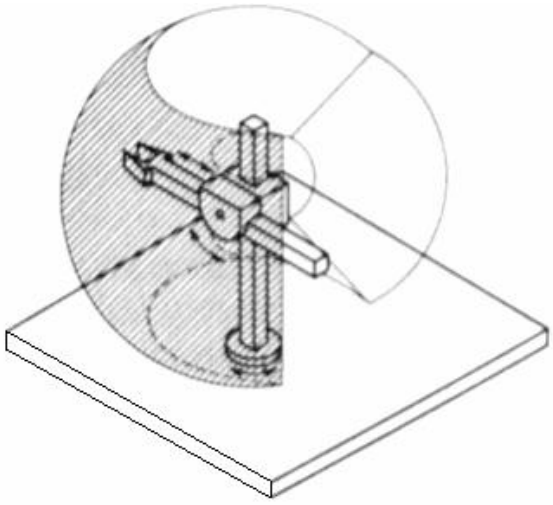
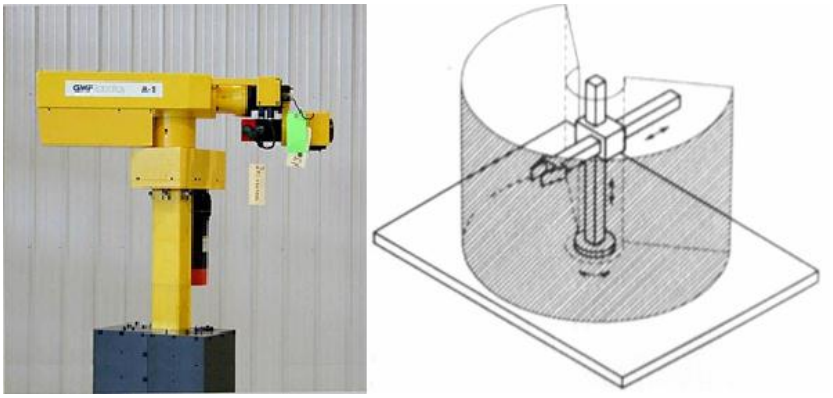


Figure 1.3 Spherical robot structure and its workspace (Sciavicco & Siciliano, 2004)

The cylindrical robot manipulator (Figure 1.4(a)) is represented with RPP kinematic structure and has one revolute joint and two prismatic joints. The workspace of cylindircal robots (Figure 1.4(b)) is an hollow cylinder. Positional accuracy of these robots also decreases as the wrist group moves away from the centerline. This type of robot manipulators are mainly used in load carrying applications.



(a) (b)

Figure 1.4 Cylindrical geometry robot manipulators, a) Cylindirical robot, b) Cylindrical robot manipulator workspace (Sciavicco & Siciliano, 2004)

SCARA geometry robots (Figure 1.5(a)) are especially employed in pick and place applications. They are represented by RRP joints. It has high vertical robustness and suitable for vertical assembly tasks. Workspace of SCARA resembles the cylindrical geometry robot manipulator but cylindrical robots have more interior side coverage.

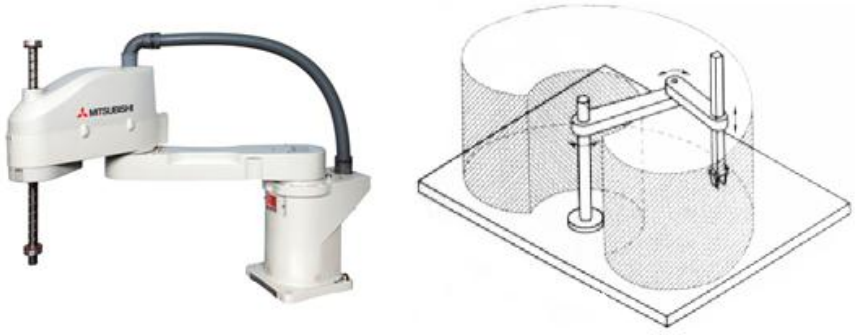


Figure 1.5 SCARA geometry robot manipulator, a) Mitsubishi RH Series SCARA (courtesy of Mitsubishi Robotics), b) SCARA robot manipulator workspace (Sciavicco & Siciliano, 2004)

Cartesian robot manipulators (Figure 1.6 (a)) are composed of three prismatic joints designated by PPP, all of which are orthogonal to each other. This type of geometry has high mechanical stiffness. High volume of workspace is one of the advantage of this type of robots. They are mainly used in material handling applications (Figure 1.6(b)).

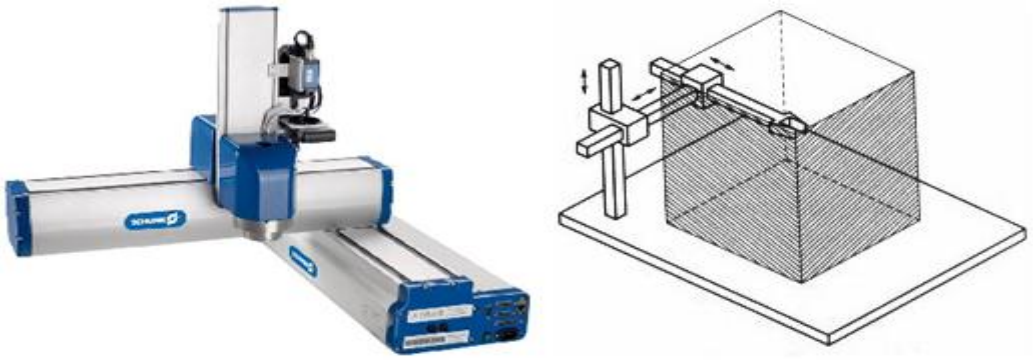


Figure 1.6 Cartesian geometry robot manipulators, a) Schunk Uniplace 2300,b) Cartesian robot workspace (Sciavicco & Siciliano, 2004)

The above manipulator structures adjust the position of the wrist of robot manipulators. The orientation of end effector (gripper, torch, etc.) is adjusted the last three joint groups called as wrist (Figure 1.7). In order to position the end effector in 3D, wrist should have at least three joints. If the three joint aces intersect at a common point, the robot manipulator has the highest dexterity.



Figure 1.7 Wrist of robot manipulator

Parallel robot manipulators (Figure 1.8) are the type of robots involving closed kinematic chain. They are mainly used in picking applications in industry.



Figure 1.8 A photo of IRB 360 FlexPicker (IRB 360 FlexPicker product specification)

1.2 Problem Statement

In this thesis, the material handling process is conceived in a computer environment in that Staubli TX90, a six DOF articulated robot manipulator, transfers a 3kg component from one station to another station (Figure 1.9).By taking the productivity concerns into account, the aim is both to reduce the power consumption of robot manipulator while keeping, at worst, the process time as in previous case. In this regard, for further increase in pruductivity, path length and path geomerty are also the focal point of this analysis in reducing cycle time which also affects the power consumption.

The current process condition involves material transfer along a path having square shape on a horizontal plane in 8s. By changing the geometry and time history of joint variables, both capacity increase thanks to a reduction of process time and energy consumption of robot manipulator are aimed to be improved. In this context, vertical triangular like path is proposed to improve the process in both capacity and energy consumption.

After completing inverse kinematic analysis of robot manipulator on both current and proposed path, the time history of all joint displacements are obtained. Using

differentiation formula, joint velocities and joint accelerations corresponding to each joint are calculated for the two path geometry.

The time history of joint configurations is used in the dynamic analysis of the robot manipulators which leads to the calculation of the required torque at each joint to transfer the part from one station to another station.

The proposed path is further analyzed so as to obtain the optimized joint configuration in terms of energy consumption by utilizing from dynamic programming.

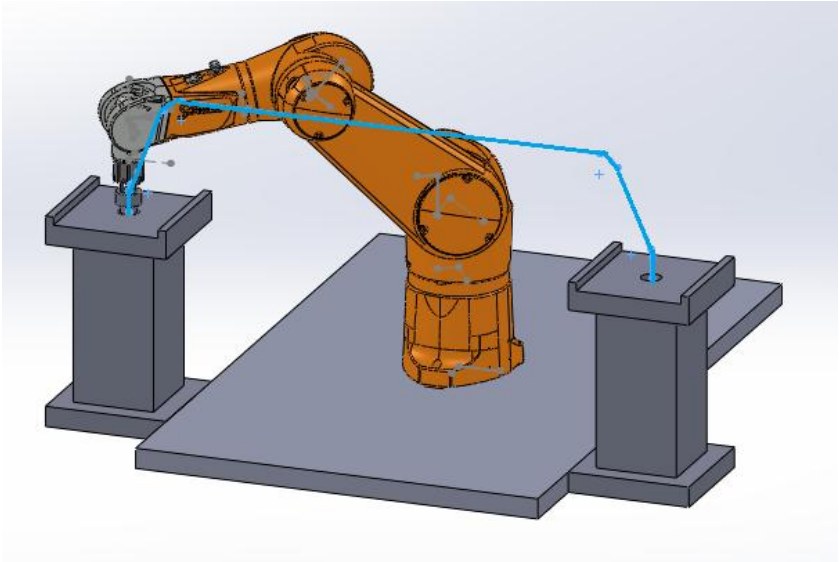


Figure 1.9 Current process environment - Staubli TX90 moving the 3kg transfered part from oe station to another station.

CHAPTER TWO
KINEMATICS OF ROBOT MANIPULATORS

Robot modelling, kinematic & dynamic analysis are the milestones of this research besides the dynamic programming and mathematical modelling of system. This chapter gives an overall way of expressing the kinematic and dynamic analysis of robot manipulator which construct base for the power consumption and optimal control of robot manipulator.

2.1 Kinematic Analysis of Robot Manipulator

Kinematic analysis involves the description of robot manipulator without taking forces/torques causing the motion into account. Forward kinematics is used to find out the position and orientation of the end effector with given or desired joint variables of robot.

The inverse kinematics is used to determine the joint variables with respect to the given end effector's position and orientation.

For an N DOF manipulator, there are (n+1) links due to the fact that each joint connects two linkages. Kinematic analysis starts with numbering each joint begininng from 1 to n. Assigning the base of manipulator as the $\{0^{th}\}$linkage, we proceed to the last linkage and number it with n.

As can be seen from Figure – joint i is fixed with respect to the link i and the joint variable denoted by qi is associated with joint i. In the case of joint variable being revolute

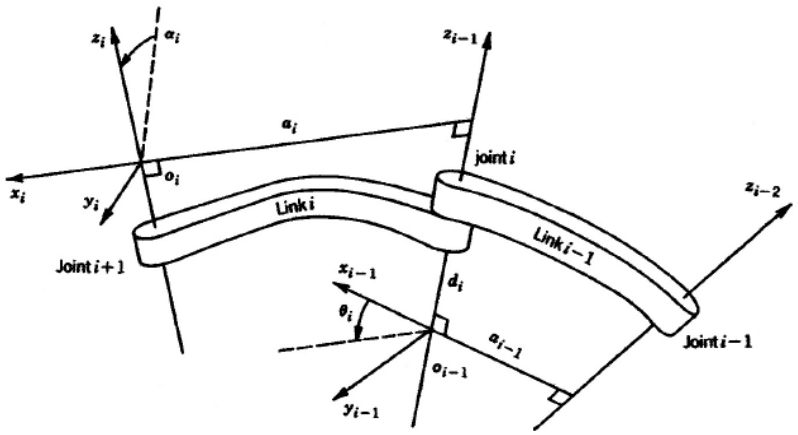


Figure 2.1 Denavit Hartenberg frame assignment (Spong, Hutchinson and Vidyasagar, 2006)

To perform kinematic analysis, we attach a coordinate frame rigidly to each link. We attach oi-xi-yi-zi to link i. This enables constant coordinates of each point on link i expressed with respect to frame {i} whatever motion the robot manipulator executes.

Suppose that A_i is the homogeneous transformation matrix that gives the position and orientation of the origin of frame {i} with respect to the origin of frame {i-1}. The homogeneous transformation matrix that expresses the position and orientation of oi-xi-yi-zi is called as transformation matrix and denoted byT_{ij} (Spong, Hutchinson and Vidyasagar, 2006),

$$T_{ij} = A_{i+1}.A_{i+2}.\ \ldots\ .A_j \tag{2.1}$$

Thus the position and orientation of the end effector in the base frame are found by

$$H = T_{0n} = A_1(q_1).A_2(q_2).\ \ldots\ A_n(q_n) \tag{2.2}$$

$$H = \begin{bmatrix} R_{0n} & o_{0n} \\ 0\ 0\ 0 & 1 \end{bmatrix} \tag{2.3}$$

2.1.1 Denavit Hartenberg Convention

In this section, we develop the forward kinematics for rigid robot manipulators. The forward kinematics deals with the relationship between the joint variables and the position and orientation of the tool and end effector.

Transformation matrix is written as follows

$$A_i = \begin{bmatrix} c\theta_i & -s\theta_i . c\alpha_i & s\theta_i . s\alpha_i & a_i . c\theta_i \\ s\theta_i & c\theta_i . c\alpha_i & -c\theta_i . s\alpha_i & a_i . s\theta_i \\ 0 & s\alpha_i & c\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.4}$$

where a_i, α_i,d_i and θ_i are link length, link twist, link offset and joint angle respectively. In the case of prismatic joint , joint angle is replaced with joint displacement (Spong & the others, 2006).

In assigning coordinate frames, the following features should exist in between two frames (Spong & the others, 2006).

(DH1) The axis x1 is perpendicular to z0 axis.
(DH2) The axis x1 intersects the axis z0 axis.

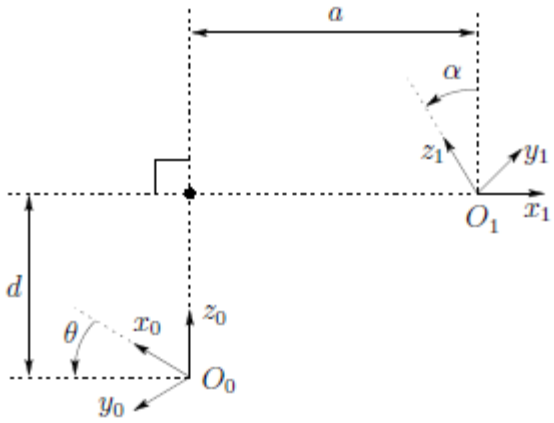


Figure 2.2 Coordinate frames satisfying assumptions DH1 and DH2 (Image source: Spong & the others, 2006)

For a given robot manipulator, coordinate frames starting from 0 to n are adjusted in such a way that the DH1 and DH2 conditions are satisfied. The following steps are used to assign the coordinate frames on the robot manipulators (Spong & the others, 2006).

Step 1. Locate and label the joint axes $z_0, \ldots, z_{n-1}$.

Step 2. Establish the base frame. Set the origin anywhere on the z_0-axis. The x_0 and y_0 axes are chosen conveniently to form a right-handed frame.
For i = 1, . . . , n − 1, perform Steps 3 to 5.

Step 3. Locate the origin o_i where the common normal to z_i and z_{i-1}intersects z_i. If z_i intersects z_{i-1}locate oi at this intersection. If z_i and z_{i-1} are parallel, locate o_i in any convenient position along z_i.

Step 4. Establish x_i along the common normal between z_{i-1} and z_i through o_i, or in the direction normal to the $z_{i-1} - z_i$ plane if $z_{i-1}$and z_i intersect.

Step 5. Establish y_i to complete a right-handed frame.

Step 6. Establish the end-effector frame on-xn-yn-zn. Assuming the n-th joint is revolute, set $z_n = a$ along the direction z_{n-1}. Establish the origin o_n conveniently along z_n, preferably at the center of the gripper or at the tip of any tool that the manipulator may be carrying. Set $y_n = s$ in the direction of the gripper closure and set $x_n = n$ as s × a. If the tool is not a simple gripper set x_n and y_n conveniently to form a right-handed frame.

Step 7. Create a table of link parameters a_i, α_i, d_i and θ_i
a_i = distance along x_i from oi to the intersection of the x_i and z_{i-1}axes.
d_i = distance along z_{i-1} from $o_{i-1}$to the intersection of the x_i and z_{i-1}axes. d_i is variable if joint i is prismatic.
α_i = the angle between z_{i-1} and z_i measured about x_i.
θ_i = the angle between x_{i-1} and x_i measured about z_{i-1}. θ_i is variableif joint i is revolute.

Step 8. Form the homogeneous transformation matrices A_i by substituting the above parameters into (2.4)

Step 9. Form $T_{0n} = A_1.A_2. \ \ldots \ A_n$. This then gives the position and orientation of the tool frame expressed in base coordinates.

2.2 Jacobian Matrices & Robot Dynamics

In previous section, it is shown how to determine the position and orientation of the end effector with the given joint angles. This section includes how to relate the linear and angular velocities of the end effector with respect to the joint velocities.

The forward kinematics equations are defined mathematically with a function between that relates the space of Cartesian positions and orientations with the space of joint positions. The velocity relationships are determined by the Jocobian of this function.

In the robotics field, the Jocobian is a matrix of partial derivatives that relate joint velocities to Cartesian velocities of the tip of the arm.

For an n link manipulator, we first derive the Jacobian representing the instantaneous transformation between the n vector of joint velocities and the 6-vector consisting of the linear and angular velocities of the end effector. This Jacobian matrix is of 6xn dimension.

Consider an n-link manipulator with joint variables q_1, q_2, ... , q_n, the Jacobian matrix is defined as

$$J = \begin{bmatrix} J_v \\ J_\omega \end{bmatrix} \tag{2.5}$$

where

$$J_v = [J_{v1} \quad J_{v2} \quad J_{v3} \quad J_{v4} \quad J_{v5} \quad J_{v6}] \tag{2.6}$$

$$J_\omega = [J_{\omega 1} \quad J_{\omega 2} \quad J_{\omega 3} \quad J_{\omega 4} \quad J_{\omega 5} \quad J_{\omega 6}] \tag{2.7}$$

The upper half of Jacobian "J_v" is given by

$$J_{vi} = \begin{cases} z_{i-1} \times (o_n - o_{i-1}) \\ z_{i-1} \end{cases} \tag{2.8}$$

The lower half of Jacobian "J_ω" is given by

$$J_{\omega i} = \begin{cases} z_{i-1} \\ 0 \end{cases} \quad (2.9)$$

As the above formulations state, Jacobian is obtained after the forward kinematics are solved out. Once z_i and the origins $o_1, o_2, \ldots, o_n$ are calculated. Jacobian is calculated as in Equation (2.5) using Equation (2.8) and Equation (2.9).

2.3 Robot Dynamics

A manipulator consisting of n links is considered for the dynamic analysis. Linear and angular velocities of any point on any link can be expressed in terms of the Jacobian matrix and the derivative of the joint variables.

$$v_i = J_{v_i}(q).\dot{q} \quad (2.10)$$

$$\omega_i = J_{\omega_i}(q).\dot{q} \quad (2.11)$$

Now suppose the mass of link i is m_i and that the inertia matrix of link i,evaluated around a coordinate frame parallel to frame i but whose origin is atthe center of mass, equals I_i. The overall kineticenergy of the manipulator equals

$$K = \frac{1}{2}.\dot{q}^T.\sum_{i=1}^{n}[m_i.J_{v_i}(q)^T.J_{v_i}(q) + J_{\omega_i}(q)^T.R_i(q).I_i.R_i(q)^T.J_{\omega_i}(q)].\dot{q} \quad (2.12)$$

In other words, the kinetic energy of the manipulator is of the form

$$K = \frac{1}{2}.\dot{q}^T.D(q).\dot{q} \quad (2.13)$$

where $D(q)$ is a symmetric positive definite matrix that is in general configuration dependent. The matrix $D(q)$ is called the inertia matrix and it can be computed for several configuration of the robot manipulators.

The potential energy of the i-th link can be computed by assuming that the mass of the entire object is concentrated at its center of mass and is given by

$$P_i = g^T.r_{ci}.m_i \quad (2.14)$$

$$P = \sum_{i=1}^{n} P_i = \sum_{i=1}^{n} g^T.r_{ci}.m_i \quad (2.15)$$

2.4 Equation of Motion

Euler-Lagrange equations are derived to obtain the equation of motion of robot manipulator. The Euler-Lagrange equations for such a system can be derived as follows based on (Spong & the others, 2006).

$$L = K - P = \frac{1}{2}.\sum_{i,j}^{n} d_{i,j}(q)\,.\,\dot{q}_i\dot{q}_j - P(q) \tag{2.16}$$

$$\frac{\partial L}{\partial \dot{q}_k} = \sum_j d_{kj}\,.\,\dot{q}_j \tag{2.17}$$

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}_k} = \sum_i d_{kj}\,.\,\ddot{q}_j + \sum_i \frac{d}{dt} d_{kj}\,.\,\dot{q}_j \tag{2.18}$$

$$= \sum_i d_{kj}\,.\,\ddot{q}_j + \sum_{i,j} \frac{\partial d_{kj}}{\partial q_i}.\,\dot{q}_i.\,\dot{q}_j \tag{2.19}$$

$$\frac{\partial L}{\partial q_k} = \frac{1}{2}.\sum_{i,j} \frac{\partial d_{ij}}{\partial q_k}.\,\dot{q}_i.\,\dot{q}_j - \frac{\partial P}{\partial q_k} \tag{2.20}$$

Thus the Euler-Lagrange equations can be written

$$\sum_i d_{kj}\,.\,\ddot{q}_j + \sum_{i,j}\{\frac{\partial d_{kj}}{\partial q_i} - \frac{1}{2}.\frac{\partial d_{ij}}{\partial q_k}\}.\,\dot{q}_i.\,\dot{q}_j \; - \; \frac{\partial P}{\partial q_k} = \tau_k \tag{2.21}$$

By interchanging the order of summation and taking advantage of symmetry, we can show that

$$\sum_{i,j}\{\frac{\partial d_{kj}}{\partial q_i}\}.\,\dot{q}_i.\,\dot{q}_j = \frac{1}{2}.\sum_{i,j}\{\frac{\partial d_{kj}}{\partial q_i} - \frac{1}{2}.\frac{\partial d_{ki}}{\partial q_j}\}.\,\dot{q}_i.\,\dot{q}_j \tag{2.22}$$

$$\sum_{i,j}\{\frac{\partial d_{kj}}{\partial q_i} - \frac{1}{2}.\frac{\partial d_{ij}}{\partial q_k}\}.\,\dot{q}_i.\,\dot{q}_j = \frac{1}{2}.\sum_{i,j}\{\frac{\partial d_{kj}}{\partial q_i} + \frac{\partial d_{ki}}{\partial q_j} - \frac{\partial d_{ij}}{\partial q_k}\}.\,\dot{q}_i.\,\dot{q}_j \tag{2.23}$$

Christoffel Symbols of the first kind is expressed as

$$c_{ijk} = \frac{1}{2}.\{\frac{\partial d_{kj}}{\partial q_i} + \frac{\partial d_{ki}}{\partial q_j} - \frac{\partial d_{ij}}{\partial q_k}\} \tag{2.24}$$

where terms c_{ijk} in Equation (2.24) are known as Christoffel symbols. For a fixed k, we have c_{ijk}= c_{jik}, which reduces the effort involved in computing these symbols by a factor of about one half.

The Euler-Lagrange equation is written as

$$\sum_i d_{kj}(q) \cdot \ddot{q}_j + \sum_{i,j} c_{ijk}(q) \cdot \dot{q}_i \cdot \dot{q}_j + \phi_k(q) = \tau_k, \; k = 1, 2, .., n \quad (2.25)$$

where

$$\phi_k = \frac{\partial P}{\partial q_k} \quad (2.26)$$

Matrix Form of Euler-Lagrange Equations is expressed as

$$D(q).\ddot{q} + C(q, \dot{q}).\dot{q} + g(q) = \tau \quad (2.27)$$

where the k, j-th element of the matrix $C(q, \dot{q})$ is defined as

$$c_{jk} = \sum_{i=1}^{n} c_{ijk}(q) \cdot \dot{q}_i \quad (2.28)$$

$$c_{jk} = \sum_{i=1}^{n} \{\frac{\partial d_{kj}}{\partial q_i} + \frac{\partial d_{ki}}{\partial q_j} - \frac{\partial d_{ij}}{\partial q_k}\} \cdot \dot{q}_i \quad (2.29)$$

2.5 Dynamic Programming

There are three criterion to be investigeted for obtaining optimal trajectory in the analysis of robot manipulator. These are the minimum time, minimum energy and minimum power consumption performance criterion.

Once the performance measure for a system has been chosen, the next task is to determine a control function that minimizes this criterion. Two methods of accomplishing the minimization are the minimum principle of Pontryagin and the method of dynamic programming develpoed by R.E.Bellman (Kirk, 2004)

2.5.1 Optimal Control Law

An optimal control with $u^*(t) = f(x(t), t)$ form is defined as being feedback optimal control. In this notation, the function f is called as optimal control law.

Consider a system represented by a first order differential equation

$$\frac{d}{dt}[x(t)] = ax(t) + bu(t) \quad (2.30)$$

where x(t) and u(t) are the state and control variables with a and b are constants.

Suppose that it is desired to minimize the power consumption during the utilization of the system. Using the performance measure for minimum power consumption of robot manipulator is formed while keeping the state variables as close to the desired state values as possible.

Before getting into detailed numerical dynamic programming calculations, the differential equations representing the robotic system is approximated by the difference equations, whereas the integral terms in the performance measure is approximated by a summation. In order to have good accuracy, the time increment is set as small as possible without yielding the calculation time to be excessive. In this regard, dividing the time interval into N equal intervals denoted by Δt.

$$\frac{x(t+\Delta t)-x(t)}{\Delta t} \approx ax(t) + bu(t) \tag{2.31}$$

$$x(t + \Delta t) = [1 + a\Delta t].x(t) + b.\Delta t.u(t) \tag{2.32}$$

Provided that the time increment is small enough, the control signals can be approcimated by a piecewise constant function.
For t = k.Δt, the system difference equation can be written as

$$x(k + 1) = [1 + a\Delta t].x(k) + b.\Delta t.u(k) \tag{2.33}$$

Similarly, the performance measure is written as

$$J = (X^{(m)}(tf) - R^{(m)}(tf))^2 + \sum_{i=1}^{N}(X^{(m)}(t) - R^{(m)}(t))^2 + \sum_{i=1}^{N}(u^{(j)}(t))^2 \tag{2.34}$$

where $X(tf)$ is the state value at final time tf, $R(tf)$ is the desired state value at final time tf, $u(t), X(t)\ and\ R(t)$ are the control input value, state value and the desired state value at an arbitraty time t, m is the index of admissible value for state value and j is the index of admissible value for control input.

The first step in computation is to find the optimal control law for the last stage of the system equation. In this context, state values and control values are quantized into admissible values by taking the physical constraints of system into account. Performance index corresponding to each admissible state and control value is

calculated. The optimal control for each state value is the one that minimizes the performance index.

2.6 Computed Torque Control

Computed torque control is a special application of feedback linearization of nonlinear systems in which innder forward loop is derived by considering the robot dynamic equations which are based on (Lewis, Dawson, Abdallah, 2004) as follows,

$$M(q).\ddot{q} + V(q,\dot{q}).\dot{q} + G(q) + \tau_d = \tau \quad (2.35)$$

$$M(q).\ddot{q} + N(q,\dot{q}) + \tau_d = \tau \quad (2.36)$$

Supposing that a desired trajectory $q_d(t)$ has been selected for the end effector motion during pick and place process. To ensure trajectory tracking by the joint variable, tracking error is tried to be minimized or eliminated.

$$e(t) = q_d(t) - q(t) \quad (2.37)$$

To demonstrate the influence of the input voltage, the first and second derivative of the error equation is determined.

$$\dot{e}(t) = \dot{q_d}(t) - \dot{q}(t) \quad (2.38)$$

$$\ddot{e}(t) = \ddot{q_d}(t) - \ddot{q}(t) \quad (2.39)$$

Putting the above error equations into robot dynamics equation gives us

$$M(q).(\ddot{q_d}(t) - \ddot{e}(t)) + N(q,\dot{q}) + \tau_d = \tau \quad (2.40)$$

$$\ddot{e}(t) = \ddot{q_d}(t) + M^{-1}(q).(N(q,\dot{q}) + \tau_d - \tau) \quad (2.41)$$

Defining the control input function and the disturbance function as

$$u(t) = \ddot{q_d}(t) + M^{-1}(N - \tau) \quad (2.42)$$

$$w(t) = M^{-1}.\tau_d \quad (2.43)$$

respectively, the state equations representing system can be written as follows.

$$\frac{d}{dt}\begin{bmatrix} e \\ \dot{e} \end{bmatrix} = \begin{bmatrix} 0 & I \\ 0 & 0 \end{bmatrix} . \begin{bmatrix} e \\ \dot{e} \end{bmatrix} + \begin{bmatrix} 0 \\ I \end{bmatrix} u + \begin{bmatrix} 0 \\ I \end{bmatrix} w \tag{2.44}$$

Inverting the feedback linearization equation yields to computed-torque control law which is represented as follows,

$$\tau = M.(\ddot{q_d} - u) + N \tag{2.45}$$

There are several ways to determine the computed-torque equation. The outer-loop signal $u(t)$ can be chosen in accordance with the selected controller type. Using PD Outer Loop design gives constant steady state error when there is a constant disturbance acting over system even if we know all the arm parameters in advance. Thus selecting the PID controller (Figure 2.3) eliminates this steady state error caused by the unpredictable disturbances and inaccuracies resulting from the unknown link and motor parameters.

Adding an integrator to the controller eliminates the nonzero steady state error encountered in PD type controller. Thus the system becomes PID computed torque controller which gives zero steady state error.

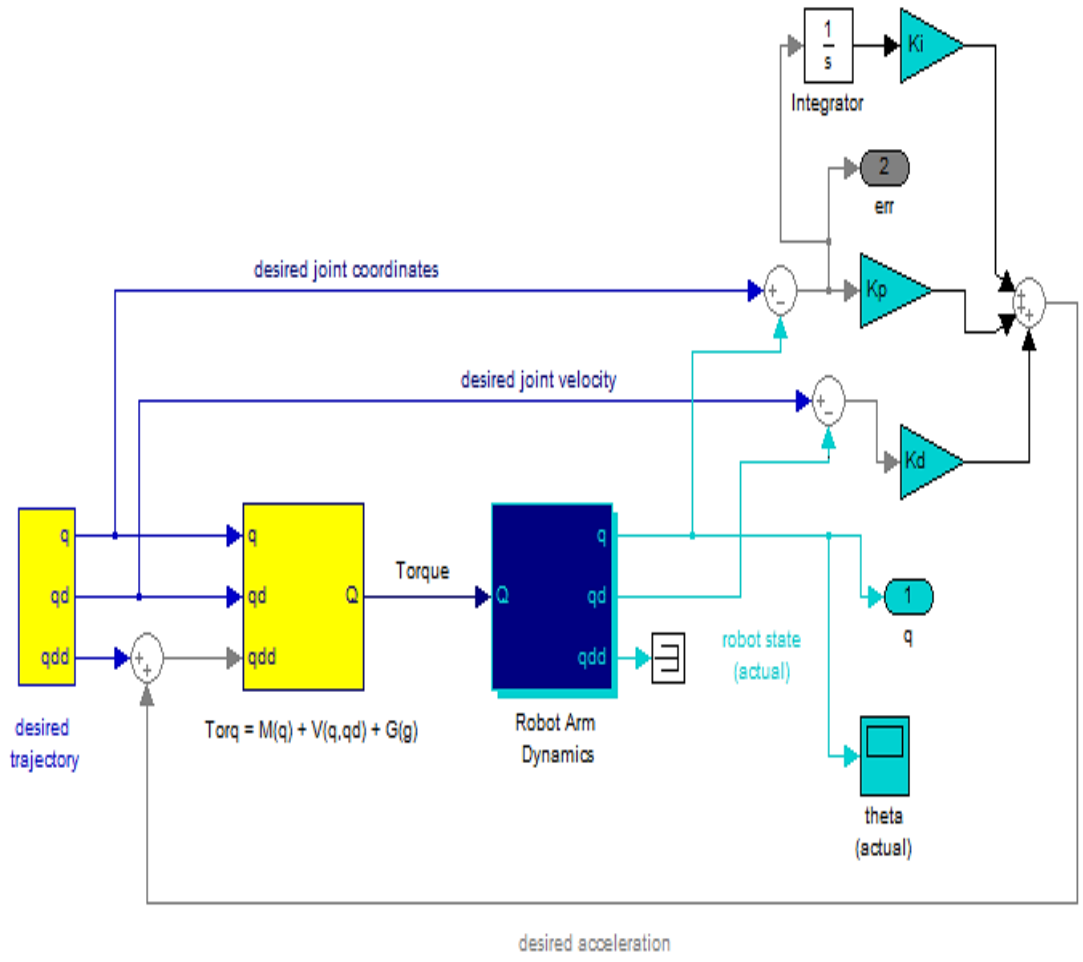


Figure 2.3 PID computed-torque controller representation

$$\dot{\varepsilon} = e \tag{2.46}$$

$$u = -K_v.\dot{e} - K_p.e - K_i.\varepsilon \tag{2.47}$$

The control input becomes

$$\tau = M(q).(\ddot{q_d} + K_v.\dot{e} + K_p.e + K_i.\varepsilon) + N(q,\dot{q}) \tag{2.48}$$

with $\varepsilon(t)$ the integral of the tracking error $e(t)$.

The state equations of the system is expressed as follows,

$$\frac{d}{dt}\begin{bmatrix}\varepsilon\\ e\\ \dot{e}\end{bmatrix} = \begin{bmatrix}0 & I & 0\\ 0 & 0 & I\\ -K_i & -K_p & -K_v\end{bmatrix}.\begin{bmatrix}\varepsilon\\ e\\ \dot{e}\end{bmatrix} + \begin{bmatrix}0\\ 0\\ I\end{bmatrix}.w \tag{2.49}$$

The closed loop characteristic equation is

$$\Delta c(s) = \left| s^3.I + K_v.s^2 + K_p.s + K_i \right| \quad (2.50)$$

Control gains are expressed as

$$K_v = diag\{k_{vi}\} \quad (2.51)$$

$$K_p = diag\{k_{pi}\} \quad (2.52)$$

$$K_i = diag\{k_{ii}\} \quad (2.53)$$

$$\Delta c(s) = \prod_{i=1}^{n} (s^3.I + k_{vi}.s^2 + k_{pi}.s + k_{ii}) \quad (2.54)$$

According to (Lewis & the others, 2004) by using the Routh test it can be found that for closed-loop stability we require the inequality$k_{ii} < k_{vi}.k_{pi}$ .Furthermore to prevent the undesirable overshoot which may cause impact during operation of robot, gains are usually selected for critical damping ξ=1and the following equations hold

$$k_{vi} = {\omega_n}^2 \quad (2.55)$$

$$k_{pi} = 2.\omega_n \quad (2.56)$$

$$\omega_r = \sqrt{kr/J} \quad (2.57)$$

with J the link inertia and kr the link stiffness and ω_n natural frequency. To avoid exciting the resonant mode, we should select ω_n=ω_r/2. In this thesis, controlller parameters are defined with Zeigler-Nichols rule without considering these natural and resonant frequency formulas.

2.7 PID Controller

PID controller is one of the most commonly used controller in industry. It stands for proportional, integral and derivative controller helping to reach steady state value irrespective of any disturbances that may result from robot models involving many unknowns from inertia of the linkages to the COM of the linkages in addition to the some model errors due to approximations.

2.7.1 Tuning PID Controller Parameters

Among the several methods, Ziegler-Nichols model is selected for tuning purposes of the PID controller. In this method, K_v and K_i are set to zero and K_p increases until the system response is sustained periodic oscillations (Figure 2.4). The gain value at which system response is oscillatory is defined as ultimate gain and represented with K_u . Based on this ultimate gain and ultimate period, K_p , K_i and K_v are determined using formulas tabulated in Table 2.1 according to the type of controller used in the system.

Table 2.1 PID tuning parameters with Ziegler-Nichols method

	K_p	K_i	K_d
P	$K_u/2$	-	-
PI	$K_u/2.2$	$P_u/1.2$	-
PID	$K_u/1.7$	$P_u/2$	$P_u/8$

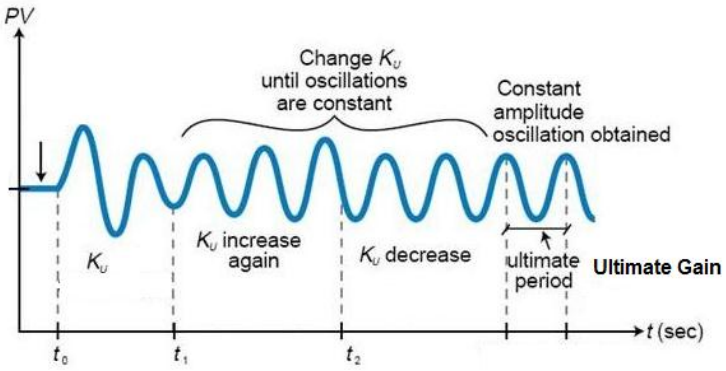


Figure 2.4 PID tuning method ultimate gain and ultimate period definition

CHAPTER THREE
METHODOLOGY

3.1 General Steps Followed

In this chapter, the methodology steps are written in detail in order to give insight about the analysis and the calculations performed so far. This work starts with the inverse kinematic analysis of robot manipulator for the given representative path for the current process condition.Inverse kinematic analysis is performed with Solidworks 2012 using path mate in motion analysis part. After obtaining the time history of joint angles for the current path (Figure 3.1),which is square-like path on the horizontal plane, joint velocities and joint accelerations are determined by using difference equation. According to these calculated joint variables, dynamic analysis of robot manipulator for the current process situation is completed using Robotic Toolbox in MATLAB. This toolbox provides ease in torque calculation. After having torque values corresponding to each joint configuration, power consumption for the current path is determined.

The above calculation steps are valid for the optimized path calculation. For the optimization purposes, triangular horizontal path is selected due to the fact that it is the most energy efficient of all other possible path geometry in addition to the minimum path length which reduces the ellapsed time for process (Carter, 2009). Furthermore the optimized path geometry is adjusted in such a way that no sharp corners exist on it. Leaving the corner points for the end effector path results in the stopping of the end effector at the corner point which in turn yields to non-smooth movement of the robot manipulator. In order to prevent this, utilizing from the rounded end effector profile creates smoother and quicker motion (Meike, Ribickis, 2011).

After defining the candidate optimized path (Figure 3.2),which is traingular horizontal path in this thesis, the minimum energy consumption path is further analyzed with dynamic programming. Dynamic programming yields to optimal

values for joint configuration by taking the minimum energy performance index into account. These optimal values are the joint configuration values which leads to minimum energy consumption among the admissible joint configuration values.

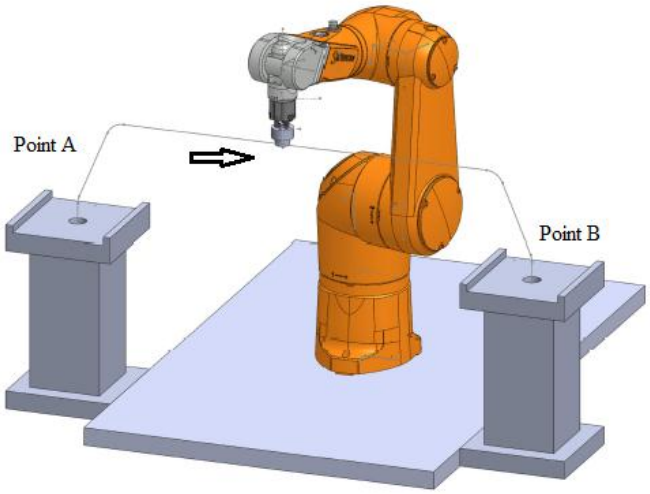


Figure 3.1 Current path followed by the end-effector

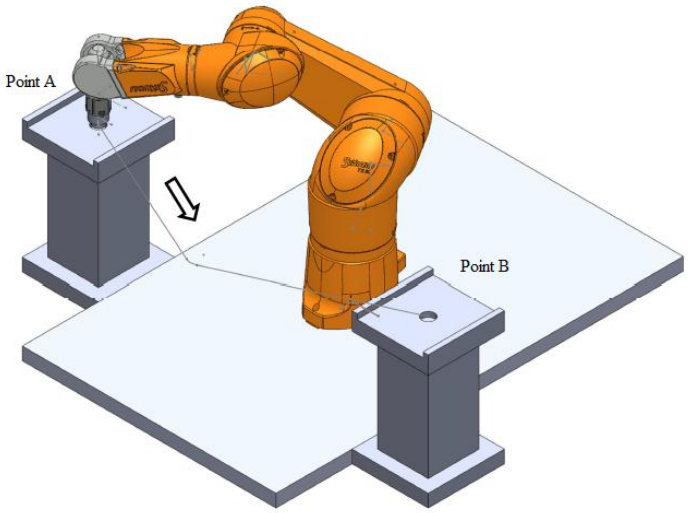


Figure 3.2 Optimized path followed by the end-effector

The optimized path provides both a cycle time reduction of 2sn in addition to the energy consumed during process.

3.2 Inverse Kinematic Analysis of Robot Manipulator

The current path is modelled in Solidworks and using “Path Mate” option (Figure 3.3) in the assembly section, the end effector of the robot is made to follow the path.

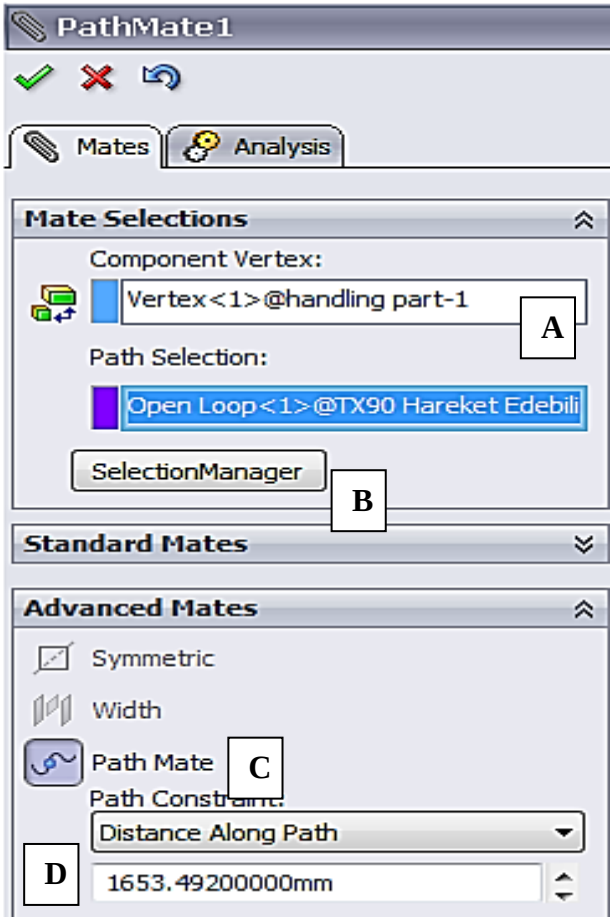


Figure 3.3 Path mate settings for inverse kinematic analysis. A is for the definition of the point on handling part. B is the path desired to be followed by end effector. C is the path constraint and D is the total length of the path.

Motion Analysis tool gives us the rotation of each joint during this movement. To use this analysis tool, first of all “Path Mate Motor” is opened and required settings (Figure 3.4) are adjusted.

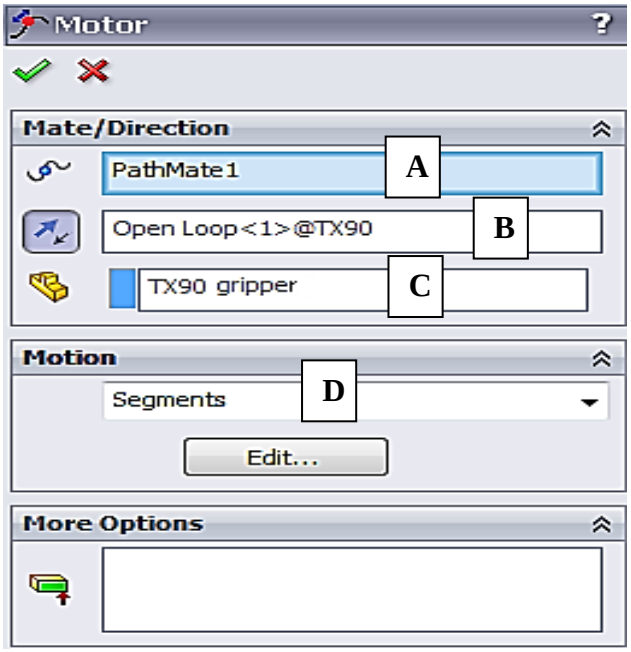


Figure 3.4 Path motor settings for inverse kinematic analysis. A is the tab used to selectthe path mate. Section B & C automatically appear on the screen. D is the tab on which the way of movement is adjusted.

Clicking on "Edit" button is used to further adjust the movement profile of the end effector. The initial condition to start the motion is zero since gripper takes the material from the station and then proceeds its movement. In order to model this initial condition, the following settings (Figure 3.5) are made. Here, 3-4-5 Polynomial is selected for the "Segment Type" so as to smoothen the jerk encountered in the analysis.

Start X	End X	Value	Segment Type
-	-	0.00000000	Initial
0s	1s	244.000000	3-4-5-Polyn
1s	7s	244.000000	3-4-5-Polyn
7s	8s	0.00000000	3-4-5-Polyn
Click to add row			

Figure 3.5 Velocity definition of the end effector in Solidworks motion analysis

The velocity values given in Figure 3.5 is entered by taking the process time into account and assuming that the end effector reaches the nominal speed in 1s. Motion parameters for current situation is given in Table 3.1. The time history of the end effector velocity, acceleration and jerk are represented in Figure 3.6

Table 3.1 Motion parameters for current case

Path Length	1708 mm
Process Time	8 sn
Avg. End Effector Speed	244.04 mm/sn

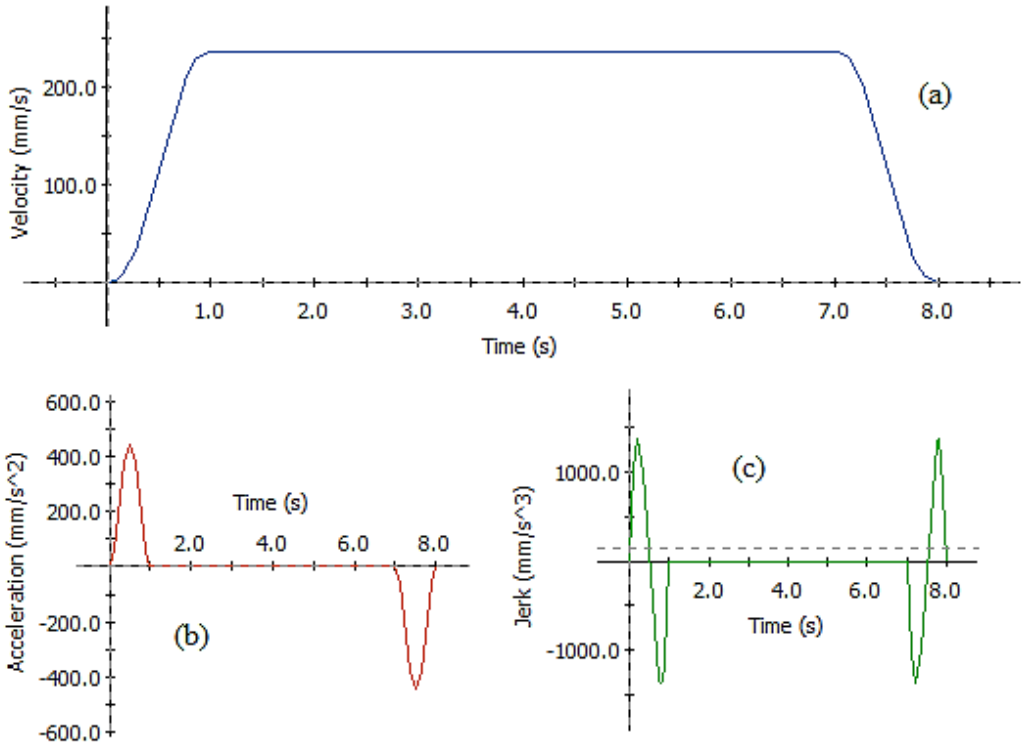


Figure 3.6 End effector motion profile for current case, a) Velocity vs time, b) Acceleration vs time, c) Jerk vs time

After completed the end effector velocity definiton, motion analysis is performed and the change in each joint is determined entered into MATLAB which in the later stage are used in the dynamic analysis of robot manipulator.

The joint velocity and joint acceleration for each point are computed using difference equation. To have good accuracy, the number of elements used in the analysis should be larger than 500. In this context, “Frames per seconds” in Motion Study Properties is adjusted to a value which equals 500/(process time). For current situation “Frames per seconds” is changed as 63.

The focal points in the path optimization in this thesis are to reduce the process time and energy consumption. The main contributor to these improvement are the path length and path geometry (Carter, 2009). According to the Carter (2009) “a triangular path moving horizontally showed to consume the least amount of energy while also requiring the least amount of travel time and travel distance” (pg 53). First off all, the path geometry is defined. In defining the path geometry, the sharp points are smoothen to decrease the effects of sudden changes in motor parameters which damages the electric motor used in robot manipulators.

The target process time is 6s. To satisfy this constraint, the length of the horizontal triangular path is determined and process parameters are tabulated in Table 3.1.

When compared to the initial path programmed with teach pendant in a indiscriminated manner by only considering the physical constraints to make a quick setup to initiate the process, focusing on the path geometry improves the process capacity in addition to the energy consumption.

Table 3.2 Motion parameters for optimized case

Path Length	1422 mm
Process Time	6 sn
Avg. End Effector Speed	284,4 mm/sn

After completing the velocity definition of the end effector (Figure 3.7) in accordance with the tabulated data in Table 4.3, we have velocity profile as shown in (Figure 3.8).

Start X	End X	Value	Segment Type
-	-	0.00000000	Initial
0s	1s	284.000000	3-4-5-Polyn
1s	5s	284.000000	3-4-5-Polyn
5s	6s	0.00000000	3-4-5-Polyn
Click to add row			

Figure 3.7 Velocity definition of the end effector in Solidworks for optimized motion analysis

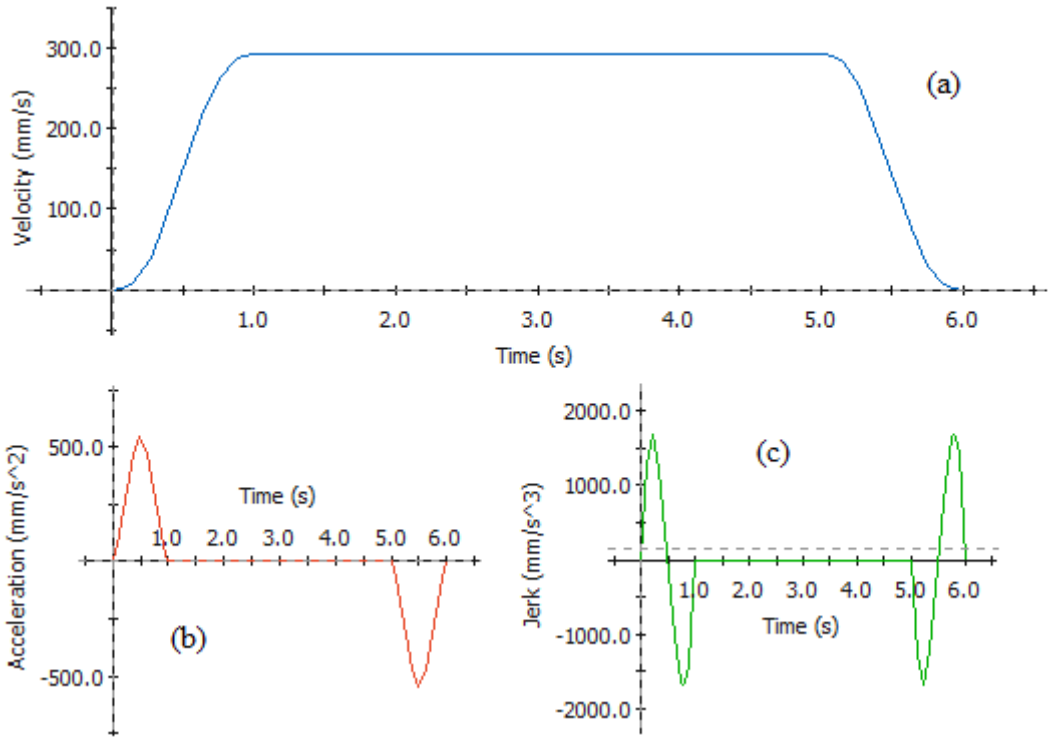


Figure 3.8 End effector motion profile for optimized case, a) Velocity vs time, b) Acceleration vs time, c) Jerk vs time

After completed the end effector velocity definiton, motion analysis is performed and the change in each joint is determined entered into MATLAB as stated before.
To have good accuracy in difference formula, the number of elements used in the analysis should be larger than 500. In this context, “Frames per seconds” in Motion Study Properties is adjusted to a value which equals 500/(process time). For optimized situation “Frames per seconds” is changed as 84.

The following section contains the calculations based on the obtained inverse kinematic results with Solidworks.

3.3 Dynamic Analysis of Robot Manipulator

Dynamic analysis of robot manipulator is carried out using MATLAB Robotic Toolbox. With this toobox, the required torque value for the corresponding joint coordinates, velocity and acceleration is calculated and constitutes basis to the

following optimization calculations. Detailed information regarding to the Robotic Toolbox is given in Appendix-B

CHAPTER FOUR
CALCULATIONS

This chapter contains the necessary calculations to be performed in the movement optimization of robot manipulator. First, DH parameters are defined into MATLAB. Secondly, inverse kinematic analysis is completed using Solidworks Motion Analysis and thirdly, dynamic analysis of the robot manipulator is performed to calculate the torque required to move the handling part from one station to another using MATLAB Robotic Toolbox. Finally, the obtained values are used in the dynamic programming to find out the optimized joint configuration value corresponding to the minimum energy consumption.

4.1 Kinematic Analysis of Staubli TX90

The robot manipulator used in material transport is Staubli TX90 (Figure 4.1).Detailed information about TX90 is given in Appendix-E. The Denavit-Hartenberg parameters in accordance with the assumed coordinate frames (Figure 4.2) are tabulated in Table 4.1 The coordinate frames are assigned in accordance with the steps stated in Chapter 2.

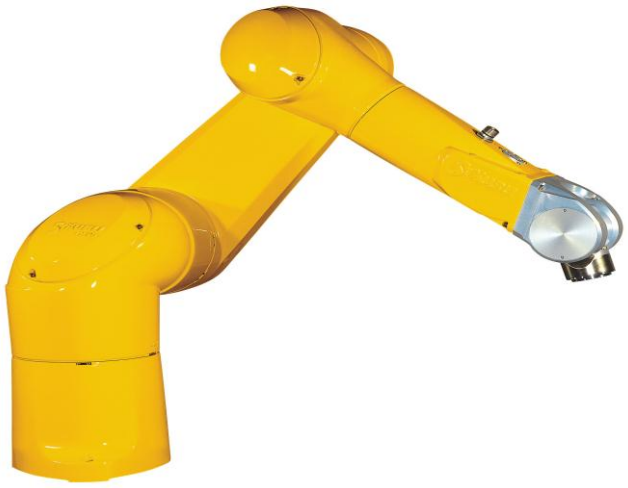


Figure 4.1 Staubli TX90 series industrial robots (Staubli TX90 product specification)

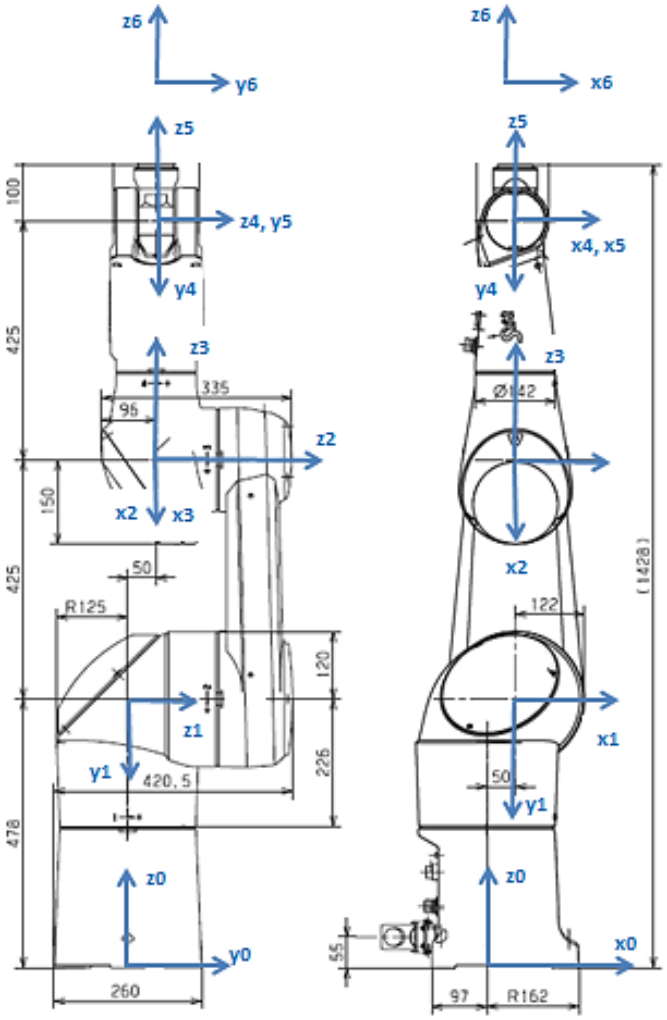


Figure 4.2 Assigned coordinate frames on Staubli TX90

Table 4.1 DH parameters of Staubli TX90

Link	θ_i	d_i (m)	a_i (m)	α_i
1	θ_1	0.478	0.05	-90°
2	θ_2	0.05	0.425	0°
3	θ_3	0	0	90°
4	θ_4	0.425	0	-90°
5	θ_5	0	0	90°
6	θ_6	0.1775	0	0°

The DH parameters are entered into MATLAB by using the Robotic Toolbox functions defined for this purpose. Detailed information regarding to the Robotic Toolbox functions are given in Appendix-B.

4.2 Dynamic Modelling of Robot Manipulator

Dynamic equations of the robot manipulator can be represented with Euler-Lagrange equations

$$M(q).\ddot{q} + V(q,\dot{q}).\dot{q} + G(q) = B.\tau + {J_V}^T.F_{ext} \quad (4.1)$$

Furthermore, AC servo motor is approximated as a brushless DC servo motor in the following analysis. The brushless DC motor is a 3-phase synchroneous AC motor which includes a position sensor to transmit the motor shaft position to the drive amplifier in controlling the current passing through the motor windings.Thus, AC servo motors used in Staubli TX90 robot manipulator is approximated as using brushless DC servo motor for the sake of simplicty in the mathematical modelling.The following mathematical model is based on a study conducted by (Aghanouri, Habibollahi, Esmaeilli, Faghihiani, Koloushani, 2011).

$$V_a = i_a.R_a + L_a.\frac{di}{dt} + K_E.\dot{q}_m \quad (4.2)$$

L_a: inductance of motor windings (H)

R_a: armature resistance (Ω)

V_a: motor voltage (V)

$\dot{q}_m$: motor velocity (rad/s)

i_a: armature current (A)

K_E: motor velocity constant (V.s/rad)

r: gear ratio

q: joint displacement

K_T: motor torque constant (N.m/A)

B_m: friction constant of motor

The constants "K_E, K_T and r" are approximated from the equivalent motor catalogue given in Appendix-D.

According to the technical data given in Appendix-DL_a<<R_a, it is disregarded in Equation (4.2)

Therefore, motor voltage is expressed as

$$V_a = i_a.R_a + K_E.\dot{q}_m \tag{4.3}$$

Armature current is obtained from Equation (4.3) as follows,

$$i_a = \frac{V_a - K_E.\dot{q}_m}{R_a} \tag{4.4}$$

with

$$\dot{q}_m = \dot{q}.r \tag{4.5}$$

$$\ddot{q}_m = \ddot{q}.r \tag{4.6}$$

and

$$J_m.\ddot{q}_m = K_T.i_a - B_m.\dot{q}_m - \tau \tag{4.7}$$

Putting Equation (4.4) into Equation (4.7) yields to

$$J_m.\ddot{q}_m = K_T.\frac{(V_a - K_E.\dot{q}_m)}{R_a} - B_m.\dot{q}_m - \tau \tag{4.8}$$

$$J_m.r.\ddot{q} = \frac{K_T.V_a}{R_a} - (\frac{K_T.K_E + R_a.B_m}{R_a}).r.\dot{q} - \tau \tag{4.9}$$

Equation (4.9) can be written as

$$\tau = \frac{K_T.V_a}{R_a} - (\frac{K_T.K_E + R_a.B_m}{R_a}).r.\dot{q} - J_m.r.\ddot{q} \tag{4.10}$$

By rewritting Equation (4.1) which involves τ on the left hand side of equation alone gives us

$$\tau = B^{-1}.(M(q).\ddot{q} + V(q,\dot{q}).\dot{q} + G(q) - {J_V}^T.F_{ext}) \tag{4.11}$$

Combining Equation (4.10) and Equation (4.11) yields to

$$\frac{K_T.V_a}{R_a} - (\frac{K_T.K_E + R_a.B_m}{R_a}).r.\dot{q} - J_m.r.\ddot{q} = B^{-1}.(M(q).\ddot{q} + V(q,\dot{q}).\dot{q} + G(q) - {J_V}^T.F_{ext}) \tag{4.12}$$

Letting V_a term alone in the equation with a little mathematical manipulation gives us

$$\frac{B.K_T.V_a}{R_a} = (M(q) + J_m.B.r).\ddot{q} + (V(q,\dot{q}) + (\frac{B.K_T.K_E + B.R_a.B_m}{R_a}).r).\dot{q} + G(q) - J_V{}^T.F_{ext} \quad (4.13)$$

$$V_a = \frac{R_a}{B.K_T}.\{(M(q) + J_m.B.r).\ddot{q} + (V(q,\dot{q}) + (\frac{B.K_T.K_E + B.R_a.B_m}{R_a}).r).\dot{q} + G(q) - J_V{}^T.F_{ext}\} \quad (4.14)$$

4.3 Control Equations of Robot Manipulator

Computed torque control is a special application of feedback linearization of nonlinear systems. PID type compted torque control is selected for the controlling of robot manipulator since it includes zero steady state error and no stability problemsFurthermore, as a result of the unknown parameter existance such as link inertia PID type controller eliminates the error resulting from this.In this way, computed torque control law is expressed by (Murray, Li&Sastry, 1994) as follows.

$$\tau = M(q).(\ddot{q}_d + K_d.\dot{e} + K_p.e + K_i.\varepsilon) + V(q,\dot{q}).\dot{q} + G(q) - J_V{}^T.F_{ext} \quad (4.15)$$

In this notation, the computed torque control is composed of two terms, one of them is feedforward component and the other is feedback component.

The feedforward component is

$$\tilde{\tau} = B^{-1}.(M(q).\ddot{q}_d + V(q,\dot{q}).\dot{q} + G(q) - J_V{}^T.F_{ext}) \quad (4.16)$$

The feedback component is

$$\hat{\tau} = B^{-1}.M(q).(K_d.\dot{e} + K_p.e + K_i.\varepsilon) \quad (4.17)$$

where q_d , $\dot{q}_d$, $\ddot{q}_d$ are the desired robot parameters, $e = q_d - q$ is the K_d, K_p and K_i are the derivative, proportional and integral gains respectively. The computed torque controller used in this work is represented in Figure 4.3 and K_p, K_i and K_d are defined as 300, 100 and 40 respectively using Ziegler-Nichols method. The corresponding system outputs and error are shown in Figure 4.4 and Figure4.5.

Work actuators can be modelled as voltage source instead of being torque source (Aghanouri, Habibollahi, Esmaeilli, Faghihiani, Koloushani, 2011). In this context, the feedforward and feedback equations can be used in obtaining feedforward and feedback voltage equations.

$$\tilde{V}_a = \frac{R_a}{B.K_T}.\{(M(q) + J_m.B.r).\ddot{q}_d + (V(q,\dot{q}) + (\frac{B.K_T.K_E + B.R_a.B_m}{R_a}).r).\dot{q} + G(q) - J_V{}^T.F_{ext}\} \quad (4.18)$$

$$\hat{V}_a = \frac{R_a}{B.K_T}.(M(q) + J_m.B.r).(K_d.\dot{e} + K_p.e + K_i.\varepsilon) \quad (4.19)$$

Equation (4.18) and Equation (4.19) represent the feedforward (open loop) voltage and feedback (compansating) voltage respectively. The actual voltage is expressed as given in Equation (4.20)

$$V_a = \tilde{V}_a + \hat{V}_a \quad (4.20)$$

Actuator voltage with computed torque control is expressed as

$$V_a = \frac{R_a}{B.K_T}.\{(M + J_m.B.r).(\ddot{q}_d + K_d.\dot{e} + K_p.e + K_i.\varepsilon) + (V + C).\dot{q} + G - J_V{}^T.F_{ext}\} \quad (4.21)$$

where

$$C = \frac{B.K_T.K_E.r + B.R_a.B_m.r}{R_a} \quad (4.22)$$

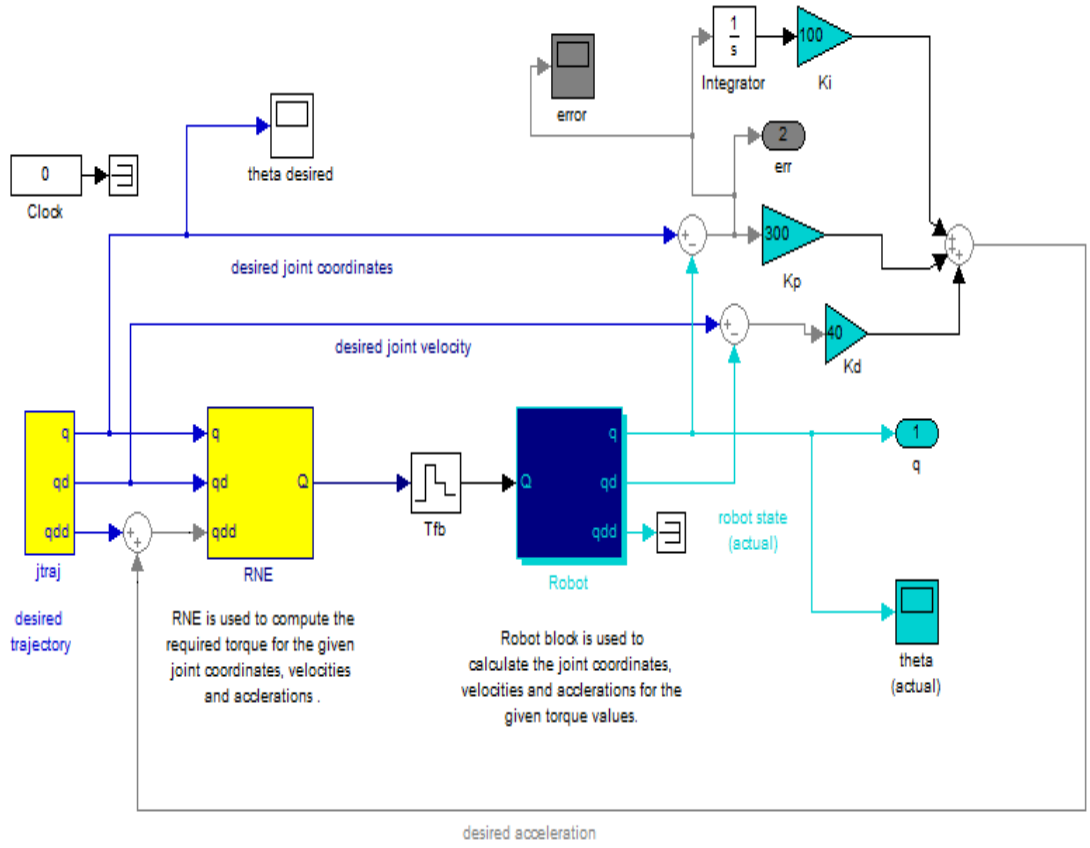


Figure 4.3PID type computed torque controller scheme used in this work

4.4 Selection of PID Parameters

Selection of PID parameters and the corresponding system output are based on Ziegler-Nichols method. In this method, system output is first tried to be oscillated by increasing K_u value while keeping K_i and K_d zero the result of which can be seen in Figure 4.6. Then based on the tabulated data used for PID controller gain approximation given in Section (2.7) and K_p , K_i , K_d are determined as 300, 100 and 40 respectively. These PID controller parameters yield good results in eliminating steady state error with less overshoot.

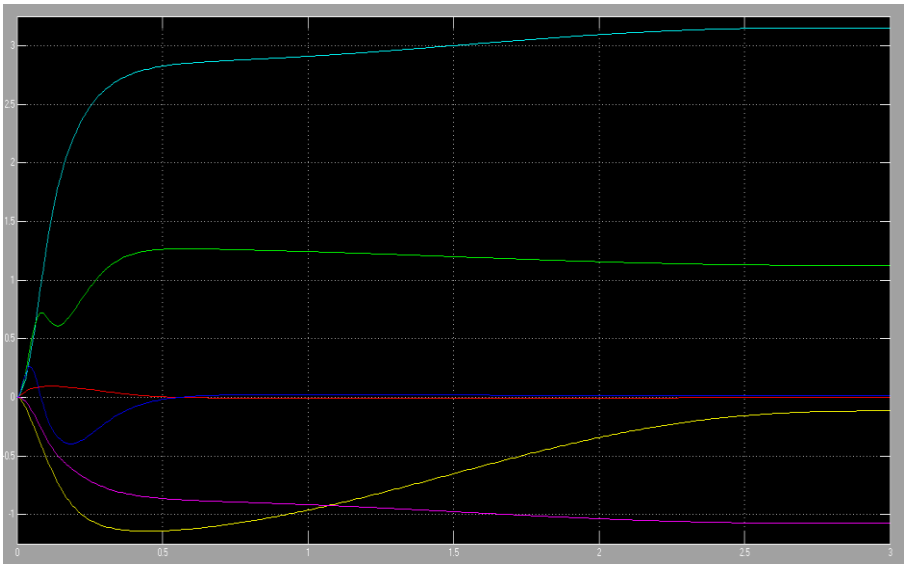


Figure 4.4 System output versus time (joint-1, joint-2, joint-3, joint-4, joint-5 and joint-6 represented with yellow, purple, cyan, blue, green and red respectively.)

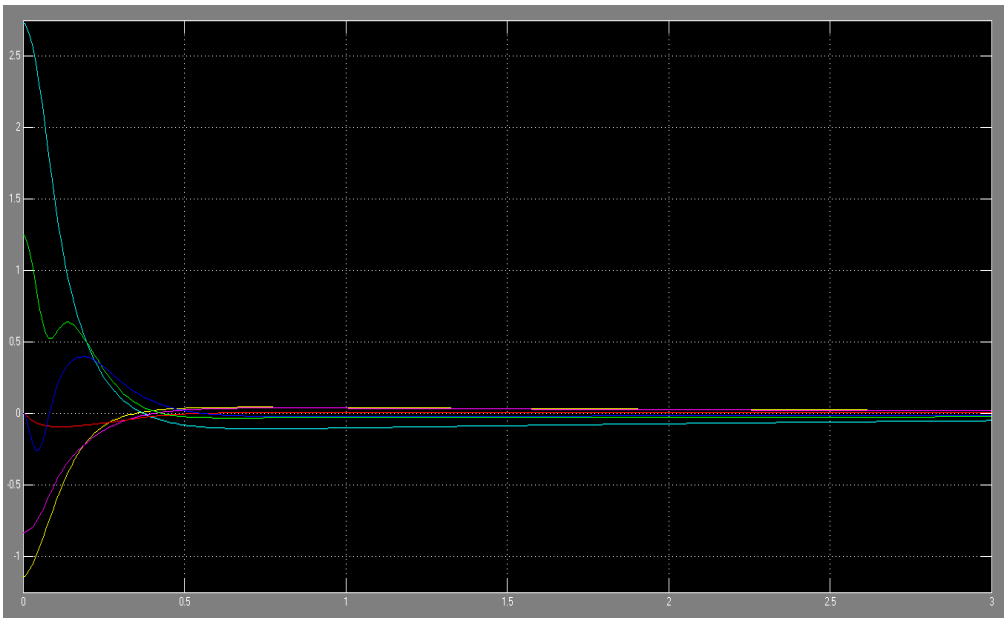


Figure 4.5 Error in system output versus time (joint-1, joint-2, joint-3, joint-4, joint-5 and joint-6 represented with yellow, purple, cyan, blue, green and red respectively.)

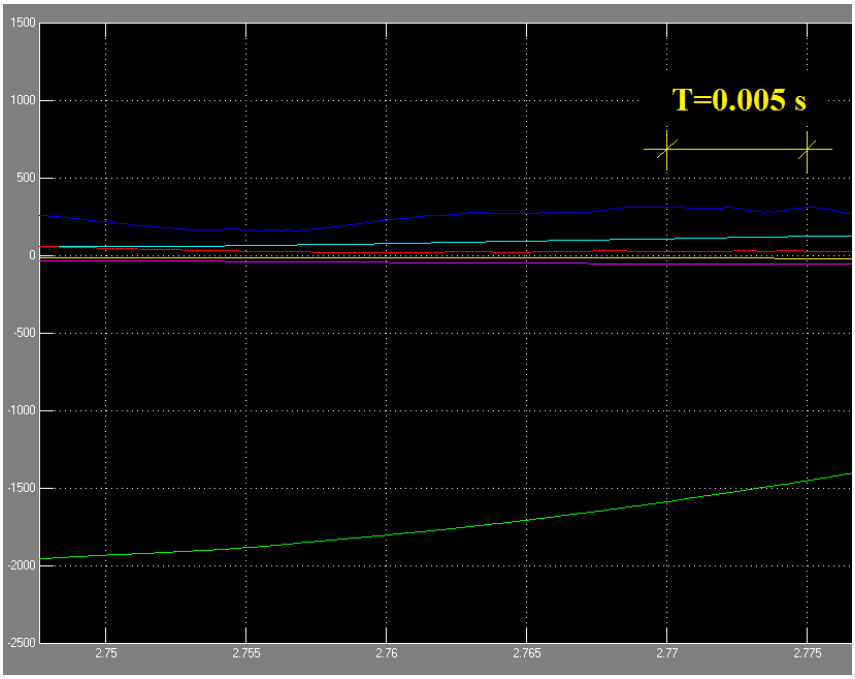


Figure 4.6 System outputs when $K_p = 500$ while K_i =K_d= 0. Output is oscillated with a period of 0.005sn.

4.5 Optimization with Dynamic Programming

Dynamic programming is applied for optimizing the robot parameters in our pick and place applications for the time optimized triangular horizontal path. This method is based on discretization of the equation and using the quantized state and input values. DP procedure begins with going from the last stage and iterating the equations using the quantized state and input values.

In this regard, the first thing to be used is the linearization of the nonlinear system equations into the following form.

$$\dot{X} = f(X(t), u(t), t) \tag{4.23}$$

Using the difference equations for the first derivative term, we obtain

$$\frac{X(t+\Delta t)-X(t)}{\Delta t} = f(X(t), u(t), t) \tag{4.24}$$

Substituting $t = k.\Delta t$ into above equation yields to

$$X(k+1) = f(X(k), u(k), k).\Delta t + X(k) \tag{4.25}$$

For a 6 DOF robot manipulator

$$x_i(t) = \begin{cases} q_i(t), & i = 1,2,..,6 \\ \dot{q}_i(t), & i = 7,8,..,12 \end{cases} \tag{4.26}$$

The control input is actuator voltage in this research which is denoted by u column vector as

$$u(k) = \begin{bmatrix} V_{a1}(k) \\ V_{a2}(k) \\ V_{a3}(k) \\ V_{a4}(k) \\ V_{a5}(k) \\ V_{a6}(k) \end{bmatrix} \tag{4.27}$$

and the state variables are denoted as X column vector

$$X(k) = \begin{bmatrix} x_1(k) \\ x_2(k) \\ . \\ . \\ x_{11}(k) \\ x_{12}(k) \end{bmatrix} \tag{4.28}$$

Using the above column vectors, eq (4.25) can be written as

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \\ . \\ . \\ x_{11}(k+1) \\ x_{12}(k+1) \end{bmatrix} = \begin{bmatrix} f_{11} \\ f_{21} \end{bmatrix}.\Delta t + \begin{bmatrix} x_1(k) \\ x_2(k) \\ . \\ . \\ x_{11}(k) \\ x_{12}(k) \end{bmatrix} \tag{4.29}$$

The terms denoted by f_{11} is represented as follows whereas f_{21} is obtained from Equation (4.21) with a little modification over the formula in that $\ddot{q}$ is let alone on the left hand side of equation.

$$f_{11} = \begin{bmatrix} x_7(k) \\ x_8(k) \\ . \\ . \\ x_{11}(k) \\ x_{12}(k) \end{bmatrix} \tag{4.30}$$

$$f_{21} = (M(\begin{bmatrix} x_1(k) \\ x_2(k) \\ x_3(k) \\ x_4(k) \\ x_5(k) \\ x_6(k) \end{bmatrix}) + J_m.B.r)^{-1}.(\frac{B.K_T}{R_a}.\begin{bmatrix} V_{a1}(k) \\ V_{a2}(k) \\ V_{a3}(k) \\ V_{a4}(k) \\ V_{a5}(k) \\ V_{a6}(k) \end{bmatrix} - (V(\begin{bmatrix} x_1(k) \\ x_2(k) \\ x_3(k) \\ x_4(k) \\ x_5(k) \\ x_6(k) \end{bmatrix}, \begin{bmatrix} x_7(k) \\ x_8(k) \\ x_9(k) \\ x_{10}(k) \\ x_{11}(k) \\ x_{12}(k) \end{bmatrix}) +$$

$$C).\begin{bmatrix} x_7(k) \\ x_8(k) \\ x_9(k) \\ x_{10}(k) \\ x_{11}(k) \\ x_{12}(k) \end{bmatrix} - G(\begin{bmatrix} x_1(k) \\ x_2(k) \\ x_3(k) \\ x_4(k) \\ x_5(k) \\ x_6(k) \end{bmatrix}) + J_V{}^T.F_{ext}) \quad (4.31)$$

By using the actuator voltage equation Equation (4.21), the control input is expressed as Equation (4.32).

$$u(k) = \frac{R_a}{B.K_T}.\{(M(\begin{bmatrix} x_1(k) \\ x_2(k) \\ x_3(k) \\ x_4(k) \\ x_5(k) \\ x_6(k) \end{bmatrix}) + J_m.B.r).(\ddot{q}_d(k) + K_D.(\dot{q}_d(k) - \begin{bmatrix} x_7(k) \\ x_8(k) \\ x_9(k) \\ x_{10}(k) \\ x_{11}(k) \\ x_{12}(k) \end{bmatrix}) +$$

$$K_p.(q_d(k) - \begin{bmatrix} x_1(k) \\ x_2(k) \\ x_3(k) \\ x_4(k) \\ x_5(k) \\ x_6(k) \end{bmatrix}) + K_i.(q_d(k) - \begin{bmatrix} x_1(k) \\ x_2(k) \\ x_3(k) \\ x_4(k) \\ x_5(k) \\ x_6(k) \end{bmatrix}).\Delta t) + (V(\begin{bmatrix} x_1(k) \\ x_2(k) \\ x_3(k) \\ x_4(k) \\ x_5(k) \\ x_6(k) \end{bmatrix}, \begin{bmatrix} x_7(k) \\ x_8(k) \\ x_9(k) \\ x_{10}(k) \\ x_{11}(k) \\ x_{12}(k) \end{bmatrix}) +$$

$$C).\begin{bmatrix} x_7(k) \\ x_8(k) \\ x_9(k) \\ x_{10}(k) \\ x_{11}(k) \\ x_{12}(k) \end{bmatrix} + G(\begin{bmatrix} x_1(k) \\ x_2(k) \\ x_3(k) \\ x_4(k) \\ x_5(k) \\ x_6(k) \end{bmatrix}) - J_V{}^T.F_{ext}\} \quad (4.32)$$

Then the state values which are $[x_1(k) \quad x_2(k) \dots \quad x_6(k)]^T$ and $[x_7(k) \quad x_8(k) \dots \quad x_{12}(k)]^T$ are quantized into S admissible values. In addition to this, control input values, $[u_1(k) \quad u_2(k) \dots \quad u_6(k)]^T$ are also quantized into C admissible values.

$$X^{(m)}(k),\ m = 1, 2, \dots, S \quad (4.33)$$

$u^{(j)}(k),\ j = 1, 2, \dots, C$ (4.34)

To reduce the time ellapsed during calculation, the time history of each joint angle values are divided into NN1=51 values. Each of these values is called as quantized state value. After defining the each quantized state (joint angle) values, admissible range is determined. Admissible range for joint angles is selected as ±2°so as not to deviate from the given path for the end effector movement. Larger admissible joint angle values for each joint lead to larger error in between the orignal and the optimized path. By considering the working environment and the possibility of interference of robot manipulator with the other objects (fixtures, tools, etc.) it is better to limit the range of admissible values in order not to obtain a path that is different from the given path. Furthermore the number of admissible joint angle and admissible joint velocity is chosen as N1=3 to decrease the calculation time.

At first, $X^{(m)}(N-1)$ is supposed and putting this assumed state values into u(k) equation, we find $u^{(j)}(N-1)$. Again using both $u^{(j)}(N-1)$ and $X^{(m)}(N-1)$, $X^{(m)}(N)$ is calculated. In a similar iteration procedure, the assumed $X^{(m)}(N-2)$ is used to calculate $u^{(j)}(N-2)$, $u^{(j)}(N-2)$ and $X^{(m)}(N-2)$ are used to obtain $X^{(m)}(N-1)$. This iteration continues until $X^{(m)}(1)$ is gathered.

These obtained values are acceptable if one of them be equal to one of the admissible values determined from the performance index selected in accordance with the type of optimization desired to have. In this analysis, for the minimum power consumption with little deviation from the starting and terminal point of the motion, the following performance index is used in which k is the stage number, N is the last stage, X(k) is the state values (joint angle/velocity) at kth stage and R(k) is the desired state values at kth stage.

$J = (X^{(m)}(N) - R^{(m)}(N))^2 + \sum_{k=1}^{N}(X^{(m)}(k) - R^{(m)}(k))^2 + \sum_{k=1}^{N}(u^{(j)}(k))^2$ (4.35)

In each stage for m and j, different performance index values are calculated. The minimum of these calculated performance index values is selected. The state values corresponding to this minimum performance index denoted by J^* are defined as the

optimized robot parameters which yield to minimum power consumption of robot manipulator in pick and place application.

4.6 Results of Calculations

Time history of joint angle & joint velocity obtained in the analysis of the initial path and the optimized path using the MATLAB codes given in Appendix-A are represented in Figure 4.7 and 4.8 in this section. Thanks to the selection of admissible range as small as possible, the error encountered in final location of robot manipulator (Figure 4.9) is kept at small value given in Table 4.4.

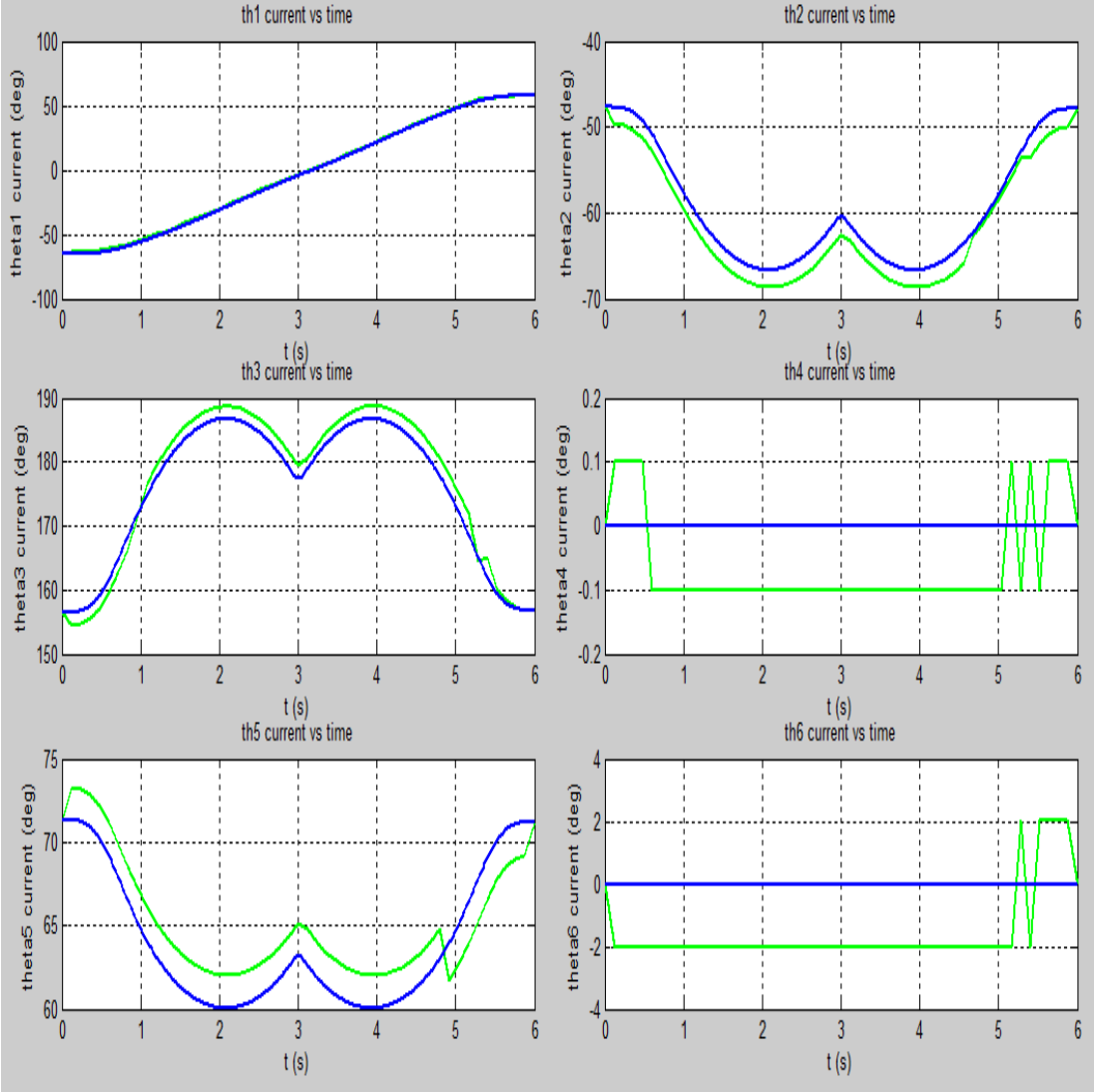


Figure 4.7 Joint angle vs time corresponding to initial path (blue) andoptimized path (green)

The optimized joint angle values are within the joint angle limits given in Appendix-E.

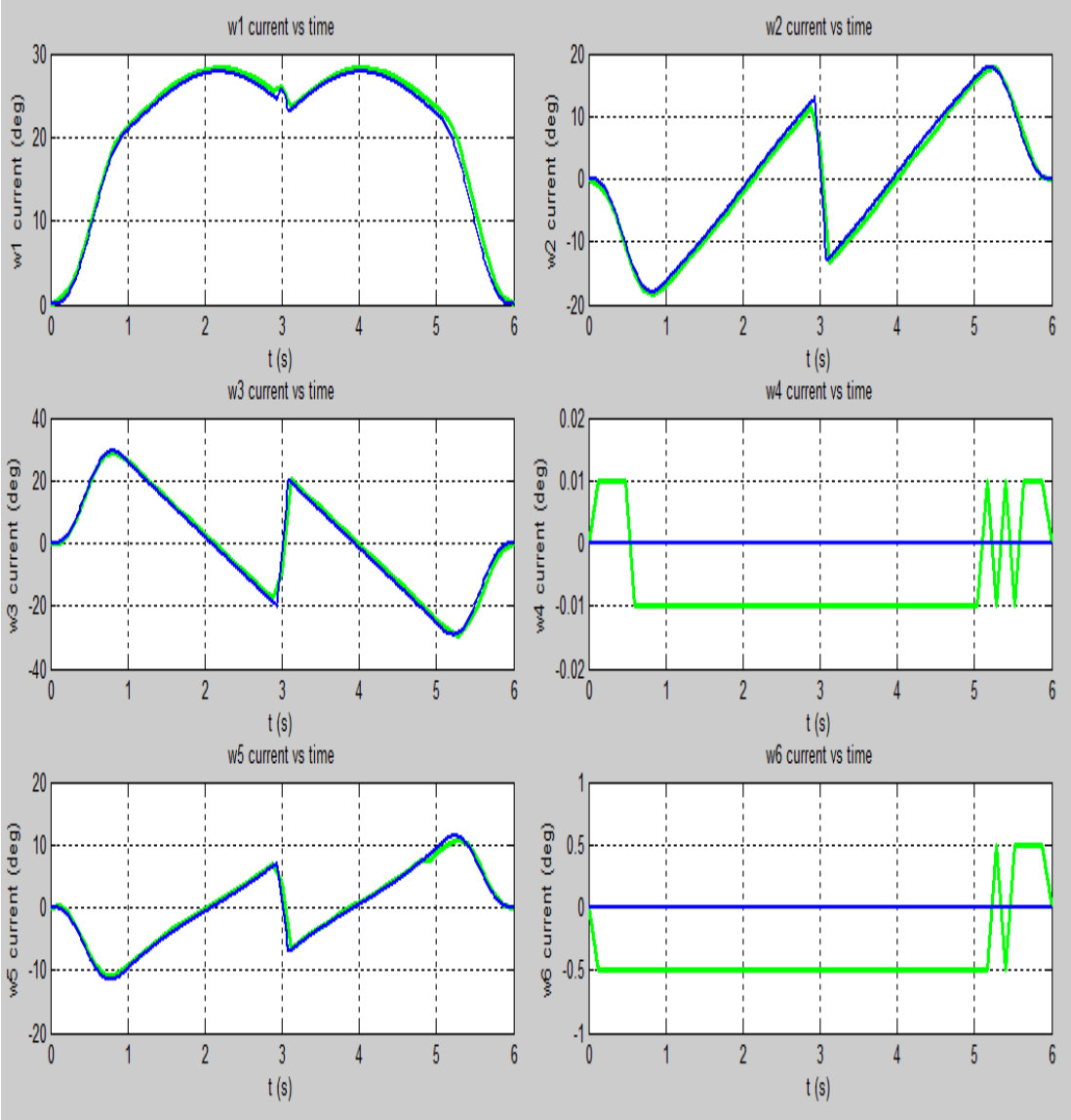


Figure 4.8 Joint velocity vs time corresponding to initial path (blue) and optimized path (green)

The optimized joint velocity values are within the joint velocity limits stated in Appendix-E.

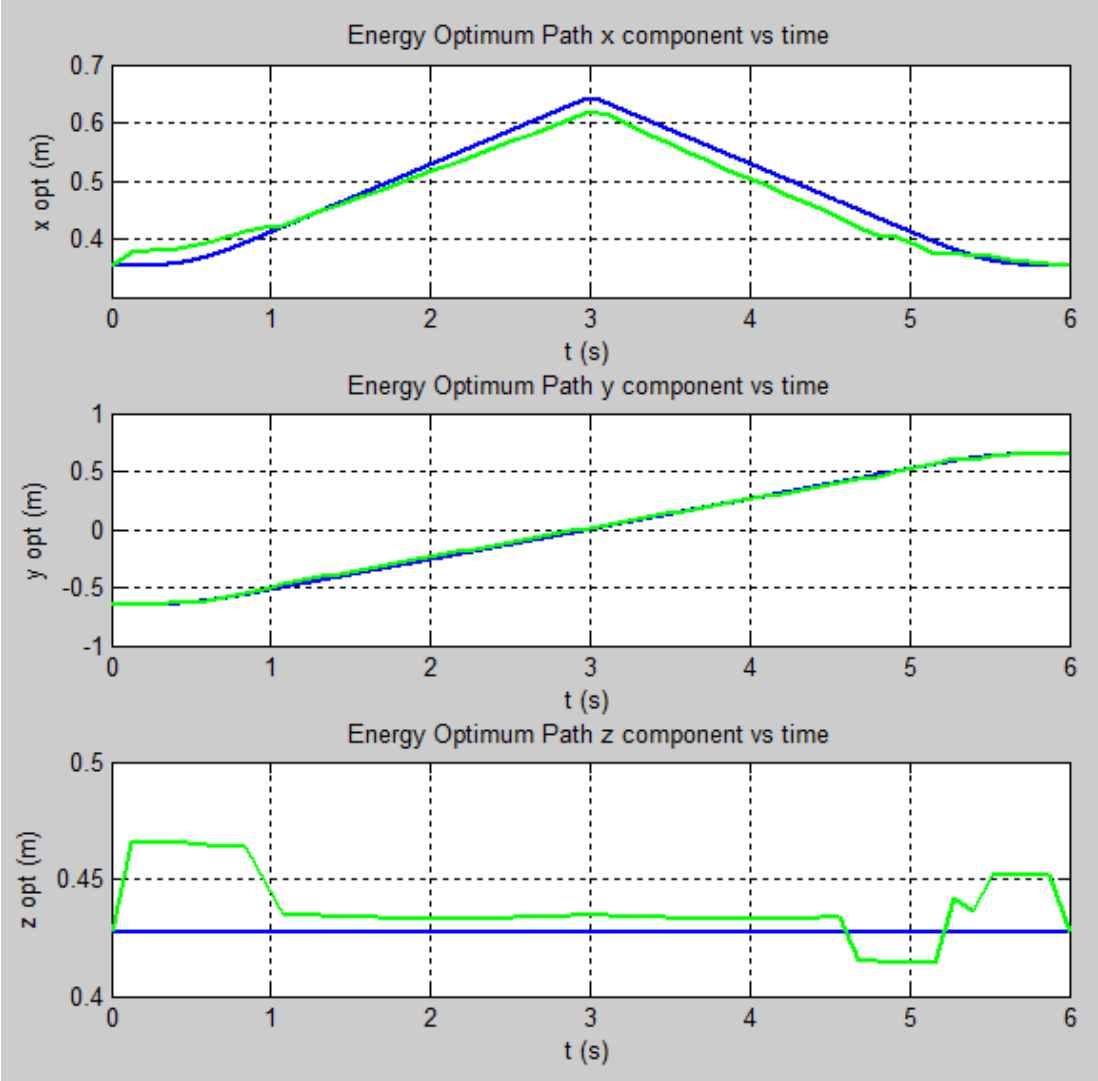


Figure 4.9 x, y, z coordinate change of end effector tip vs time for geometrically optimized path (blue) and energetically optimized path (green)

Table 4.2 Error in final end effector point after optimization

Error_x	0.000 m
Error_y	0.000 m
Error_z	0.000 m

Although the system involves indexing operation, the positional accuracy is one of the important parameter in this work. To eliminate the positional error encountered in this motion analysis, we set the initial and final joint coordinates of the robot manipulator the same as the ones obtained after geometrical optimization.

The optimization procedure which takes 36min to be completed yields to the optimal parameters which reduces the energy consumption of robot manipulator among several admissible joint angle and joint velocity values. Table 4.5 represents process parameters to make a comparision in between initial condition, simple path geometry optimization and parameter optimization obtained with dynamic programming applied on the geometrically optimized path taking minimum energy performance index into account. The results show that production capacity increases by an amount of 25% in addition to the energy saving of approximately 420 joule for one cycle of robot manipulator.

Table 4.3 Results containing data regarding to the initial case and optimized case

	Initial Case Results	Geometry Optimization Results	Parameter Optimization Results
Path Length	1708 mm	1422 mm	-
Process Time	8 sn	6 sn	-
Avg. End Effector Speed	244.0 mm/sn	284.4 mm/sn	-
Energy Consumption	700.7 Joule	386.3 Joule	280.5 Joule

CHAPTER FIVE
CONCLUSION

In this thesis, it is aimed to reduce both process time and energy consumption to satisfy the desire in decreasing manufacturing cost. To start with, initial manufacturing conditions are modelled in computer environment by taking process time and end effector motion into account and power consumption of this movement is calculated. Then, geometry of path is optimized for which a substantial amount of process time reduction is possible. After the geometry of path for optimized case is defined without going beyond robot manipulator limits, optimal process parameters in implementing material transfer from one station to another in an energy optimal way are determined with dynamic programming.

To sum up, thanks to this analysis, it is possible to increase production capacity by 25% besides the savings in energy consumption which is approximately 420 joule for one cycle. These savings provide the company a substantial amount of improvement in this material handling process and will be a reference work for further process development in many other robot cells.

In future work, these optimal robot parameters are implemeted to the robot manipulator and the energy consumption of robot manipulator is measured to verify the results. Furthermore, due to the approximated link parameters and motor specifications, the results presented here should be taken into account as approximations. These unknown parameters may lead to motor saturation if the PID gains are not adjusted in detail. Therefore, in the future work, motor saturation problem will also be analyzed by clarifying all robot and motor parameters precisely.

REFERENCE

Aghanouri, M., Habibollahi, A., Esmaeili, A., Faghihian, H., & Koloushani, M. (2011). Optimization of robotic manipulators parameters modeled with integrated equations of actuators and links. *Electrodynamics and Mechatronics (SCE III), 2011 3rd International Students Conference on, 1*, 32.

Aghanouri, M., Habibollahi, A., Esmaeili, A., Faghihian, H., & Koloushani, M. (2011). Optimization of robotic manipulators parameters modeled with integrated equations of actuators and links. *Electrodynamics and Mechatronics (SCE III), 2011 3rd International Students Conference on, 1*, 31-36

Carter, T. (2009). *The modeling of a six degree of freedom industrial robot for the purpose of efficient path planning.* Retrieved May 2009, from https://etda.libraries.psu.edu/paper/9682/

Carter, T. (2009). *The modeling of a six degree of freedom industrial robot for the purpose of efficient path planning.* Retrieved May 2009, from https://etda.libraries.psu.edu/paper/9682/

Fanuc America Corporation, (2014). *LR Mate Series.* Retrieved April 15, 2014, from http://www.fanucamerica.com/products/robots

ISO Standard 8373 (1994). *Manipulating industrial robots - vocabulary,* retrieved April 2014, from http://en.wikipedia.org/wiki/Industrial_robot.

Kirk, E. D. (1998*). Optimal control theory: An introduction* (13rd ed.). N.Y: Dovel Publication, Inc.

Lewis, F.L., Dawson, D. M., & Abdallah, C. T. (2004). *Robot manipulator control theory and practice (*2nd ed.). N.Y.: Marcel Dekker, Inc.

Lewis, F.L., Dawson, D. M., & Abdallah, C. T. (2004). *Robot manipulator control theory and practice (*2nd ed.). N.Y.: Marcel Dekker, Inc.

Meike, D., & Ribickis, L. (2011). Analysis of the energy efficient usage methods of medium and high payload industrial robots in the automobile industry. In: *10th International Symposium "Topical Problems in the Field of Electrical and Power Engineering. Doctoral School of Energy and Geotechnology II"*, 62-66.

Mitsubishi Electric America, (2014). RH-F Scara. Retrieved April 27, 2014, http://www.meau.com/eprise/main/sites/public/Products/Robots

Murray, R. M., Li, Z., & Sastry, S. S. (1994). *A mathematical introduction to robotic manipulation* (2nd ed.). USA: CRC Press.

Sciavicco, L., & Siciliano, B. (2004). *Modeling and control of robot manipulators* (4th ed.). London: Springer.

Spong, M. W., Hutchinson, S., & Vidyasagar, M. (2006). R*obot modelling and control* (2nd ed.). USA: John Wiley & Sons, Inc.

Spong, M. W. Hutchinson, S., & Vidyasagar, M. (2006). *Robot modelling and control* (2nd ed.). USA: John Wiley & Sons, Inc.

Staubli Turkey, (2014). Medium payload robots TX90, Retrieved April 27, 2014, from http://www.staubli.com/en/robotics/6-axis-scara-industrial-robot/

APPENDIX A

MATLAB COMMANDS

```
%PART-1
%------

L(1) = Link([0 0.478 0.05 -pi/2]);
L(2) = Link([0 0.05 0.425 0]);
L(3) = Link([0 0 0. pi/2]);
L(4) = Link([0 0.425 0 -pi/2]);
L(5) = Link([0 0 0 pi/2]);
L(6) = Link([0 0 0 0]);
tx90 = SerialLink(L, 'name', 'TX90')

% Tool frame is away 0.1775m from the z axis of coordinate frame {6}
tx90.tool = transl(0,0,0.2285);

% Transfered part has 3.0kg mass and 0.05m offset from coordinate
frame {6}
tx90.payload(3.0,[0,0,0.05]);

% TOTAL MASS OF TX90 IS APPROXIMATELY 110KG

% dynamic: link mass
tx90.links(1).m = 33.935;
tx90.links(2).m = 25.937;
tx90.links(3).m = 15.502;
tx90.links(4).m = 9.540;
tx90.links(5).m = 2.569;
tx90.links(6).m = 1.19148;

% dynamic: link COG wrt link coordinate frame 3x1
tx90.links(1).r = [0.0191;0.0248;0.4106];
tx90.links(2).r = [0;-0.1731;0.2202];
tx90.links(3).r = [-0.0036;0;0.0151];
tx90.links(4).r = [-0.0025;0;0.2952];
tx90.links(5).r = [0;-0.0228;-0.0001];
tx90.links(6).r = [-0.0005;0.0005;0.1263];
```

```
% actuator: motor inertia (motor referred)
tx90.links(1).Jm = 1.35*10^-4;
tx90.links(2).Jm = 1.35*10^-4;
tx90.links(3).Jm = 1.35*10^-4;
tx90.links(4).Jm = 0.451*10^-4;
tx90.links(5).Jm = 0.451*10^-4;
tx90.links(6).Jm = 0.451*10^-4;

% dynamic: link inertia matrix, symmetric about link COG.
tx90.links(1).I = [0.4446 -0.0476 0.033;
    -0.0476 0.3142 -0.0634;
    0.033 -0.0634 0.418];

tx90.links(2).I = [0.8506 0 0;
    0 0.109 0.0058;
    0 0.0058 0.9046];

tx90.links(3).I = [0.1178 0 0;
    0 0.0686 0.0074;
    0 0.0074 0.1138];

tx90.links(4).I = [0.0864 0 -0.0014;
    0 0.0768 0;
    -0.0014 0 0.029];

tx90.links(5).I = [0.0022 0 0;
    0 0.0008 0;
    0 0 0.0022];

tx90.links(6).I = [0.0002 0 0;
    0 0.0004 0;
    0 0 0.0002];

% dynamic: link viscous friction (motor referred)
tx90.links(1).B = 0;
tx90.links(2).B = 0;
tx90.links(3).B = 0;
tx90.links(4).B = 0;
tx90.links(5).B = 0;
```

```
tx90.links(6).B = 0;

% dynamic: link Coulomb friction
tx90.links(1).Tc = 0;
tx90.links(2).Tc = 0;
tx90.links(3).Tc = 0;
tx90.links(4).Tc = 0;
tx90.links(5).Tc = 0;
tx90.links(6).Tc = 0;

% actuator: gear ratio (SGMAS catalogue is taken as reference r =
qm/q = 1/5)
tx90.links(1).G = 1/5;
tx90.links(2).G = 1/5;
tx90.links(3).G = 1/5;
tx90.links(4).G = 1/5;
tx90.links(5).G = 1/5;
tx90.links(6).G = 1/5;

%Inverse kinematic analysis results are entered here for each joint.
%th1 to th6 are row vector
th1 = [];
th2 = [];
th3 = [];
th4 = [];
th5 = [];
th6 = [];

N=length(th1);
tp=6;   %Process Time
t=linspace(0,tp,N);
dt=tp/length(t);
r=pi/180;

% Results are aligned with Solidworks data
% To adjust Solidworks value in accordance with assumed sense of
direction of Coordinate Frames

qr=[0 -pi/2 pi 0 0 0]; %Zero position of TX90
```

```
th1 = th1*r*(-1);
th2 = (-th2*r) + qr(1,2)*ones(1,length(th2));
th3 = qr(1,3)*ones(1,length(th3)) -(pi/2*ones(1,length(th3))+th3*r);
th4 = th4*r;
th5 = th5*r + pi/2*ones(1,length(th5));
th6 = th6*r*(-1);

% Armature Circuit Resistance
Ra=[9.4 0 0 0 0 0;0 9.4 0 0 0 0;0 0 9.4 0 0 0;0 0 0 0.832 0 0;0 0 0
0 0.832 0;0 0 0 0 0 0.832];

% Armature Circuit Inductance
La=[12.77*10^-3 0 0 0 0 0;0 12.77*10^-3 0 0 0 0;0 0 12.77*10^-3 0 0
0;0 0 0 2.73*10^-3 0 0;0 0 0 0 2.73*10^-3 0;0 0 0 0 0 2.73*10^-3];

B=eye(6,6);

%Torque Constant
KT=[0.487 0 0 0 0 0;0 0.487 0 0 0 0;0 0 0.487 0 0 0;0 0 0 0.375 0
0;0 0 0 0 0.375 0;0 0 0 0 0 0.375];

%Rotor Inertia
Jm=[1.35*10^-4 0 0 0 0 0;0 1.35*10^-4 0 0 0 0;0 0 1.35*10^-4 0 0 0;0
0 0 0.451*10^-4 0 0;0 0 0 0 0.451*10^-4 0;0 0 0 0 0 0.451*10^-4];

%Gear Ratio
r_mot=[0.2 0 0 0 0 0;0 0.2 0 0 0 0;0 0 0.2 0 0 0;0 0 0 0.2 0 0;0 0 0
0 0.2 0;0 0 0 0 0 0.2];

% Proportional Constant
Kp=300*eye(6,6);

% Derivative Constant
Kd=40*eye(6,6);

% Integral Constant
Ki=100*eye(6,6);
```

```
%Speed Constant from BLS71 - BLS72 catalogue
KE=[0.28 0 0 0 0 0;0 0.28 0 0 0 0;0 0 0.28 0 0 0;0 0 0 0.22 0 0;0 0
0 0 0.22 0;0 0 0 0 0 0.22];

% Friction Constant  (N.m.s)
Bm=0.1*eye(6,6);
Fext=zeros(6,1);
Jv=zeros(6,6);

for i=1:N
    th(i,1:6) = [th1(1,i) th2(1,i) th3(1,i) th4(1,i) th5(1,i)
th6(1,i)];
end


dth1=zeros(1,N);
dth2=zeros(1,N);
dth3=zeros(1,N);
dth4=zeros(1,N);
dth5=zeros(1,N);
dth6=zeros(1,N);

ddth1=zeros(1,N);
ddth2=zeros(1,N);
ddth3=zeros(1,N);
ddth4=zeros(1,N);
ddth5=zeros(1,N);
ddth6=zeros(1,N);

for i=1:N
if i==1
        dth1(1,i)=0;        % Initial conditions are set to 0. w0=0,
alpha0=0
        ddth1(1,i)=0;
        dth2(1,i)=0;
        ddth2(1,i)=0;
        dth3(1,i)=0;
        ddth3(1,i)=0;
        dth4(1,i)=0;
```

```
        ddth4(1,i)=0;
        dth5(1,i)=0;
        ddth5(1,i)=0;
        dth6(1,i)=0;
        ddth6(1,i)=0;
else
        dth1(1,i) = (th1(1,i) - th1(1,i-1))/dt;
        ddth1(1,i) = (dth1(1,i) - dth1(1,i-1))/dt;
        dth2(1,i) = (th2(1,i) - th2(1,i-1))/dt;
        ddth2(1,i) = (dth2(1,i) - dth2(1,i-1))/dt;
        dth3(1,i) = (th3(1,i) - th3(1,i-1))/dt;
        ddth3(1,i) = (dth3(1,i) - dth3(1,i-1))/dt;
        dth4(1,i) = (th4(1,i) - th4(1,i-1))/dt;
        ddth4(1,i) = (dth4(1,i) - dth4(1,i-1))/dt;
        dth5(1,i) = (th5(1,i) - th5(1,i-1))/dt;
        ddth5(1,i) = (dth5(1,i) - dth5(1,i-1))/dt;
        dth6(1,i) = (th6(1,i) - th6(1,i-1))/dt;
        ddth6(1,i) = (dth6(1,i) - dth6(1,i-1))/dt;
end

    th(i,1:6) = [th1(1,i) th2(1,i) th3(1,i) th4(1,i) th5(1,i)
th6(1,i)];
    dth(i,1:6) = [dth1(1,i) dth2(1,i) dth3(1,i) dth4(1,i) dth5(1,i)
dth6(1,i)];
    ddth(i,1:6) = [ddth1(1,i) ddth2(1,i) ddth3(1,i) ddth4(1,i)
ddth5(1,i) ddth6(1,i)];
    M(6*i-5:6*i,1:6) = tx90.inertia(th(i,1:6));
    V(6*i-5:6*i,1:6) = tx90.coriolis(th(i,1:6),dth(i,1:6));
    G(i,1:6) = tx90.gravload(th(i,1:6));
    Torque(1:6,i) = M(6*i-5:6*i,1:6)*transpose(ddth(i,1:6)) + V(6*i-
5:6*i,1:6)*transpose(dth(i,1:6)) + transpose(G(i,1:6));
    Torque(1:6,i) = B*Torque(1:6,i);
end



%PART-2
%_____________________________________________________________________
```

```
N1=3; % Number of admissible value for each quantization

xth1=zeros(N1,N1);
xth2=zeros(N1,N1);
xth3=zeros(N1,N1);
xth4=zeros(N1,N1);
xth5=zeros(N1,N1);
xth6=zeros(N1,N1);

xdth1=zeros(N1,N1);
xdth2=zeros(N1,N1);
xdth3=zeros(N1,N1);
xdth4=zeros(N1,N1);
xdth5=zeros(N1,N1);
xdth6=zeros(N1,N1);

NN1=51;     % Number of quantization

adm_range=2;       % Degree
adm_vel=0.5;       % Degree/sn

adm_range4=0.1;    % Degree
adm_vel4=0.01;     % Degree/sn


for i=1:NN1
    xth1min(1,i) = th1(1,(N-5)/(NN1-1)*i-9)-adm_range*r;
    xth1max(1,i) = th1(1,(N-5)/(NN1-1)*i-9)+adm_range*r;
    xth1(:,i) = linspace(xth1min(1,i),xth1max(1,i),N1);
end

xth1(:,1) = th1(1,1)*ones(N1,1);%For positional accuracy at initial
xth1(:,NN1) = th1(1,N)*ones(N1,1);% and final position.

for i=1:NN1
    th2min(1,i) = th2(1,(N-5)/(NN1-1)*i-9)-adm_range*r;
    xth2max(1,i) = th2(1,(N-5)/(NN1-1)*i-9)+adm_range*r;
    xth2(:,i) = linspace(xth2min(1,i),xth2max(1,i),N1);
end
```

```
xth2(:,1) = th2(1,1)*ones(N1,1);
xth2(:,NN1) = th2(1,N)*ones(N1,1);

for i=1:NN1
    xth3min(1,i) = th3(1,(N-5)/(NN1-1)*i-9)-adm_range*r;
    xth3max(1,i) = th3(1,(N-5)/(NN1-1)*i-9)+adm_range*r;
    xth3(:,i) = linspace(xth3min(1,i),xth3max(1,i),N1);
end

xth3(:,1) = th3(1,1)*ones(N1,1);
xth3(:,NN1) = th3(1,N)*ones(N1,1);

for i=1:NN1
    xth4min(1,i) = th4(1,(N-5)/(NN1-1)*i-9)-adm_range4*r;
    xth4max(1,i) = th4(1,(N-5)/(NN1-1)*i-9)+adm_range4*r;
    xth4(:,i) = linspace(xth4min(1,i),xth4max(1,i),N1);
end

xth4(:,1) = th4(1,1)*ones(N1,1);
xth4(:,NN1) = th4(1,N)*ones(N1,1);

for i=1:NN1
    xth5min(1,i) = th5(1,(N-5)/(NN1-1)*i-9)-adm_range*r;
    xth5max(1,i) = th5(1,(N-5)/(NN1-1)*i-9)+adm_range*r;
    xth5(:,i) = linspace(xth5min(1,i),xth5max(1,i),N1);
end

xth5(:,1) = th5(1,1)*ones(N1,1);
xth5(:,NN1) = th5(1,N)*ones(N1,1);

for i=1:NN1
    xth6min(1,i) = th6(1,(N-5)/(NN1-1)*i-9)-adm_range*r;
    xth6max(1,i) = th6(1,(N-5)/(NN1-1)*i-9)+adm_range*r;
    xth6(:,i) = linspace(xth6min(1,i),xth6max(1,i),N1);
end

xth6(:,1) = th6(1,1)*ones(N1,1);
xth6(:,NN1) = th6(1,N)*ones(N1,1);
```

```
for i=1:NN1
    xdth1min(1,i) = dth1(1,(N-5)/(NN1-1)*i-9)-adm_vel*r;
    xdth1max(1,i) = dth1(1,(N-5)/(NN1-1)*i-9)+adm_vel*r;
    xdth1(:,i) = linspace(xdth1min(1,i),xdth1max(1,i),N1);
end

xdth1(:,1) = dth1(1,1)*ones(N1,1);
xdth1(:,NN1) = dth1(1,N)*ones(N1,1);

for i=1:NN1
    xdth2min(1,i) = dth2(1,(N-5)/(NN1-1)*i-9)-adm_vel*r;
    xdth2max(1,i) = dth2(1,(N-5)/(NN1-1)*i-9)+adm_vel*r;
    xdth2(:,i) = linspace(xdth2min(1,i),xdth2max(1,i),N1);
end

xdth2(:,1) = dth2(1,1)*ones(N1,1);
xdth2(:,NN1) = dth2(1,N)*ones(N1,1);

for i=1:NN1
    xdth3min(1,i) = dth3(1,(N-5)/(NN1-1)*i-9)-adm_vel*r;
    xdth3max(1,i) = dth3(1,(N-5)/(NN1-1)*i-9)+adm_vel*r;
    xdth3(:,i) = linspace(xdth3min(1,i),xdth3max(1,i),N1);
end

xdth3(:,1) = dth3(1,1)*ones(N1,1);
xdth3(:,NN1) = dth3(1,N)*ones(N1,1);

for i=1:NN1
    xdth4min(1,i) = dth4(1,(N-5)/(NN1-1)*i-9)-adm_vel4*r;
    xdth4max(1,i) = dth4(1,(N-5)/(NN1-1)*i-9)+adm_vel4*r;
    xdth4(:,i) = linspace(xdth4min(1,i),xdth4max(1,i),N1);
end

xdth4(:,1) = dth4(1,1)*ones(N1,1);
xdth4(:,NN1) = dth4(1,N)*ones(N1,1);

for i=1:NN1
    xdth5min(1,i) = dth5(1,(N-5)/(NN1-1)*i-9)-adm_vel*r;
```

```
    xdth5max(1,i) = dth5(1,(N-5)/(NN1-1)*i-9)+adm_vel*r;
    xdth5(:,i) = linspace(xdth5min(1,i),xdth5max(1,i),N1);
end

xdth5(:,1) = dth5(1,1)*ones(N1,1);
xdth5(:,NN1) = dth5(1,N)*ones(N1,1);

for i=1:NN1
    xdth6min(1,i) = dth6(1,(N-5)/(NN1-1)*i-9)-adm_vel*r;
    xdth6max(1,i) = dth6(1,(N-5)/(NN1-1)*i-9)+adm_vel*r;
    xdth6(:,i) = linspace(xdth6min(1,i),xdth6max(1,i),N1);
end

xdth6(:,1) = dth6(1,1)*ones(N1,1);
xdth6(:,NN1) = dth6(1,N)*ones(N1,1);


%PART-3


p=1;
X1=[];
while p<=NN1
for i=1:N1
for j=1:N1
for k=1:N1
for l=1:N1
for m=1:N1
for n=1:N1
                              X1=[X1;xth1(n,p) xth2(m,p) xth3(l,p)
xth4(k,p) xth5(j,p) xth6(i,p)];
end
end
end
end
end
end
    p=p+1;
end
```

```
p=1;
X2=[];
while p<=NN1
for i=1:N1
for j=1:N1
for k=1:N1
for l=1:N1
for m=1:N1
for n=1:N1
                              X2=[X2;xdth1(n,p) xdth2(m,p) xdth3(l,p)
xdth4(k,p) xdth5(j,p) xdth6(i,p)];
end
end
end
end
end
end
    p=p+1;
end


X1=transpose(X1);
X2=transpose(X2);

X=[X1;X2];


%PART-4
%_____________________________

for kkk=1:NN1
    o=(length(X)/NN1)*(kkk-1);
for i=1:N
for j=1:length(X)/NN1
            u(1:6,j+o) = Ra/B/KT*((M(6*i-
5:6*i,1:6)+B*Jm*r_mot)*(transpose(ddth(i,1:6))+Kp*(transpose(th(i,1:
6))-X1(1:6,j+o))+Kd*(transpose(dth(i,1:6))-
```

```
X2(1:6,j+o))+Ki*(transpose(th(i,1:6))-X1(1:6,j+o))*dt)+( V(6*i-
5:6*i,1:6)+(B*r_mot*KT*KE + B*r_mot*Ra*Bm)/Ra
)*X2(1:6,j+o)+transpose(G(i,1:6))+transpose(Jv)*Fext);
end
end
end

for i=1:length(u)
    u(1:6,i) = abs(u(1:6,i));
end

%
N11=length(X1);
N22=length(X2);
Nth=length(th);
Ndth=length(dth);

A=zeros(length(X1),1);
BB=zeros(length(X1),1);
C=zeros(length(X1),1);
J=zeros(length(X1),1);


%Performance Measure Calculation "J". J is formed in such a way that
states
%are close to their final value at final time.

for kkk=1:NN1
    o=729*(kkk-1);
for i=1:N
for j=1:length(X1)/NN1
            A(j+o,1) = transpose([X1(1:6,N11);X2(1:6,N22)]-
[transpose(th(Nth,1:6));transpose(dth(Ndth,1:6))])*([X1(1:6,N11);X2(
1:6,N22)]-[transpose(th(Nth,1:6));transpose(dth(Ndth,1:6))]);
            BB(j+o,1) = transpose( [X1(1:6,j+o);X2(1:6,j+o)]-
transpose([th(i,1:6),dth(i,1:6)]) )*([X1(1:6,j+o);X2(1:6,j+o)]-
transpose([th(i,1:6),dth(i,1:6)]));
            C(j+o,1) = transpose(u(1:6,j+o))*u(1:6,j+o);
            J(j+o,1) = A(j+o,1) + BB(j+o,1) + C(j+o,1);
end
```

```
end
end

min_index=[];
for k=1:NN1
    [min_J(k,1),min_index(k,1)] = min(J((length(X1)/NN1*k-
(length(X1)/NN1-1)):length(X1)/NN1*k,1));
end

% min_index gives us the index of performance measure corresponding
to theminimum performance measure value.
for k=1:NN1
if k==1
        min_index(k,1)=min_index(k,1);
else
        min_index(k,1)=min_index(k,1)+length(X1)/NN1*(k-1);
end
end


for i=1:N
    Torque(1:6,i) = M(6*i-5:6*i,1:6)*transpose(ddth(i,1:6)) + V(6*i-
5:6*i,1:6)*transpose(dth(i,1:6)) + transpose(G(i,1:6));
    Torque(1:6,i) = B*Torque(1:6,i);
end


for i=1:N
    Power(i) = abs(dth(i,1:6))*abs(Torque(1:6,i));
if i>1
        Total_Power_Consumption(i) = Power(i-1) + Power(i);
        Total_Energy_Consumption(i) = Power(i-1)*dt + Power(i)*dt;
else
        Total_Power_Consumption(i) = Power(i);
        Total_Energy_Consumption(i) = Power(i)*dt;
end
end
```

```
Total_Energy_Consumption = sum(Total_Energy_Consumption)

for i=1:NN1
    th_opt(1:6,i) = X1(1:6,min_index(i,1));
    dth_opt(1:6,i) = X2(1:6,min_index(i,1));
end

% % TX90 ANGLE CONSTRAINTS
% th1min=-180*r;
% th1max=180*r;
% th2min=-130*r;
% th2max=147.5;
% th3min=-145*r;
% th3max=145*r;
% th4min=-270*r;
% th4max=270*r;
% th5min=-115*r;
% th5max=140*r;
% th6min=-270*r;
% th6max=270*r;

th_opt=transpose(th_opt);
dth_opt=transpose(dth_opt);

for i=1:NN1
if i==1
        dth_opt(i,1)=0;
        dth_opt(i,2)=0;
        dth_opt(i,3)=0;
        dth_opt(i,4)=0;
        dth_opt(i,5)=0;
        dth_opt(i,6)=0;

else

        ddth_opt(i,1) = (dth_opt(i,1) - dth_opt(i-1,1))/dt;
        ddth_opt(i,2) = (dth_opt(i,2) - dth_opt(i-1,2))/dt;
        ddth_opt(i,3) = (dth_opt(i,3) - dth_opt(i-1,3))/dt;
        ddth_opt(i,4) = (dth_opt(i,4) - dth_opt(i-1,4))/dt;
```

```
        ddth_opt(i,5) = (dth_opt(i,5) - dth_opt(i-1,5))/dt;
        ddth_opt(i,6) = (dth_opt(i,6) - dth_opt(i-1,6))/dt;
end
end

% Optimized power consumption

for i=1:NN1
    M_opt(6*i-5:6*i,1:6) = tx90.inertia(th_opt(i,1:6));
    V_opt(6*i-5:6*i,1:6) =
tx90.coriolis(th_opt(i,1:6),dth_opt(i,1:6));
    G_opt(i,1:6) = tx90.gravload(th_opt(i,1:6));
    Torque_opt(1:6,i) = M(6*i-5:6*i,1:6)*transpose(ddth_opt(i,1:6))
+ V(6*i-5:6*i,1:6)*transpose(dth_opt(i,1:6)) +
transpose(G_opt(i,1:6));
    Torque_opt(1:6,i) = B*Torque_opt(1:6,i);
end

dt_opt=tp/length(th_opt);

for i=1:NN1
    Power_opt(i) = abs(dth_opt(i,1:6))*abs(Torque_opt(1:6,i));
if i>1
        Total_Power_Consumption_opt(i) = Power_opt(i-1) +
Power_opt(i);
        Total_Energy_Consumption_opt(i) = Power_opt(i-1)*dt +
Power_opt(i)*dt_opt;
else
        Total_Power_Consumption_opt(i) = Power_opt(i);
        Total_Energy_Consumption_opt(i) = Power_opt(i)*dt_opt;
end
end


Total_Energy_Consumption_opt = sum(Total_Energy_Consumption_opt)

Energy_Savings = Total_Energy_Consumption -
Total_Energy_Consumption_opt
```

```
t_opt=linspace(0,tp,NN1);

subplot(3,2,1)
plot(t_opt,th_opt(:,1)*180/3.14,'g')
title('th1 opt vs time');
xlabel(' t (s) ');
ylabel('theta1 opt (deg)');
grid on
hold on
plot(t,th1*180/3.14)
title('th1 current vs time');
xlabel(' t (s) ');
ylabel('theta1 current (deg)');

grid on

subplot(3,2,2)
plot(t_opt,th_opt(:,2)*180/3.14,'g')
title('th2 opt vs time');
xlabel(' t (s) ');
ylabel('theta2 opt (deg)');
grid on
hold on
plot(t,th2*180/3.14)
title('th2 current vs time');
xlabel(' t (s) ');
ylabel('theta2 current (deg)');
grid on

subplot(3,2,3)
plot(t_opt,th_opt(:,3)*180/3.14,'g')
title('th3 opt vs time');
xlabel(' t (s) ');
ylabel('theta3 opt (deg)');
grid on
hold on
plot(t,th3*180/3.14)
title('th3 current vs time');
xlabel(' t (s) ');
```

```
ylabel('theta3 current (deg)');
grid on

subplot(3,2,4)
plot(t_opt,th_opt(:,4)*180/3.14,'g')
title('th4 opt vs time');
xlabel(' t (s) ');
ylabel('theta4 opt (deg)');
grid on
hold on
plot(t,th4*180/3.14)
title('th4 current vs time');
xlabel(' t (s) ');
ylabel('theta4 current (deg)');
grid on

subplot(3,2,5)
plot(t_opt,th_opt(:,5)*180/3.14,'g')
title('th5 opt vs time');
xlabel(' t (s) ');
ylabel('theta5 opt (deg)');
grid on
hold on
plot(t,th5*180/3.14)
title('th5 current vs time');
xlabel(' t (s) ');
ylabel('theta5 current (deg)');
grid on

subplot(3,2,6)
plot(t_opt,th_opt(:,6)*180/3.14,'g')
title('th6 opt vs time');
xlabel(' t (s) ');
ylabel('theta6 opt (deg)');
grid on
hold on
plot(t,th6*180/3.14)
title('th6 current vs time');
xlabel(' t (s) ');
```

```
ylabel('theta6 current (deg)');
grid on

figure
subplot(3,2,1)
plot(t_opt,dth_opt(:,1)*180/3.14,'g')
title('w1 opt vs time');
xlabel(' t (s) ');
ylabel('w1 opt (deg/s)');
grid on
hold on
plot(t,dth1*180/3.14)
title('w1 current vs time');
xlabel(' t (s) ');
ylabel('w1 current (deg)');
grid on

subplot(3,2,2)
plot(t_opt,dth_opt(:,2)*180/3.14,'g')
title('w2 opt vs time');
xlabel(' t (s) ');
ylabel('w2 opt (deg/s)');
grid on
hold on
plot(t,dth2*180/3.14)
title('w2 current vs time');
xlabel(' t (s) ');
ylabel('w2 current (deg)');
grid on

subplot(3,2,3)
plot(t_opt,dth_opt(:,3)*180/3.14,'g')
title('w3 opt vs time');
xlabel(' t (s) ');
ylabel('w3 opt (deg/s)');
grid on
hold on
plot(t,dth3*180/3.14)
title('w3 current vs time');
```

```
xlabel(' t (s) ');
ylabel('w3 current (deg)');
grid on

subplot(3,2,4)
plot(t_opt,dth_opt(:,4)*180/3.14,'g')
title('w4 opt vs time');
xlabel(' t (s) ');
ylabel('w4 opt (deg/s)');
grid on
hold on
plot(t,dth4*180/3.14)
title('w4 current vs time');
xlabel(' t (s) ');
ylabel('w4 current (deg)');
grid on

subplot(3,2,5)
plot(t_opt,dth_opt(:,5)*180/3.14,'g')
title('w5 opt vs time');
xlabel(' t (s) ');
ylabel('w5 opt (deg/s)');
grid on
hold on
plot(t,dth5*180/3.14)
title('w5 current vs time');
xlabel(' t (s) ');
ylabel('w5 current (deg)');
grid on

subplot(3,2,6)
plot(t_opt,dth_opt(:,6)*180/3.14,'g')
title('w6 opt vs time');
xlabel(' t (s) ');
ylabel('w6 opt (deg/s)');
grid on
hold on
plot(t,dth6*180/3.14)
title('w6 current vs time');
```

```
xlabel(' t (s) ');
ylabel('w6 current (deg)');
grid on

for i=1:N
    T(4*i-3:4*i,1:4)=tx90.fkine(th(i,1:6));
    x_path(1,i)=T(4*i-3,4);
    y_path(1,i)=T(4*i-2,4);
    z_path(1,i)=T(4*i-1,4);
end

for i=1:NN1
    TTT(4*i-3:4*i,1:4)=tx90.fkine(th_opt(i,1:6));
    x_opt(1,i)=TTT(4*i-3,4);
    y_opt(1,i)=TTT(4*i-2,4);
    z_opt(1,i)=TTT(4*i-1,4);
end

figure
subplot(3,1,1)
plot(t,x_path)
grid on
title('Current x path vs time');
xlabel(' t (s) ');
ylabel('x current (m)');
hold on
plot(t_opt,x_opt,'g')
hold off
grid on
title('Energy Optimum Path x component vs time');
xlabel(' t (s) ');
ylabel('x opt (m)');
subplot(3,1,2)
plot(t,y_path)
grid on
title('Current y path vs time');
xlabel(' t (s) ');
ylabel('y current (m)');
hold on
```

```
plot(t_opt,y_opt,'g')
grid on
title('Energy Optimum Path y component vs time');
xlabel(' t (s) ');
ylabel('y opt (m)');
hold off
subplot(3,1,3)
plot(t,z_path)
grid on
title('Current z path vs time');
xlabel(' t (s) ');
ylabel('z current (m)');
hold on
plot(t_opt,z_opt,'g')
grid on
title('Energy Optimum Path z component vs time');
xlabel(' t (s) ');
ylabel('z opt (m)');
hold off

%Positional Error Calculation
Error_x=x_path(1,N)-x_opt(1,NN1)
Error_y=y_path(1,N)-y_opt(1,NN1)
Error_z=z_path(1,N)-z_opt(1,NN1)
```

APPENDIX B

ROBOTIC TOOLBOX

Robotic toolbox is a useful toobox to study forward and inverse kinematics of robot manipulator besides the dynamic analysis, trajectory generation and simulation of robot manipulators. In this toolbox, there are many functions specifically defined to study the robot manipulators using MATLAB objects in which parameters representing kinematic and dynamic properties are defined.

There are a number of Robotic Toolbox functions used in this analysis. These are represented in the below.

Link:

Link object is used to define all information regarding to robot link from kinematics parameters, rigid-body inertial parameters to motor and transmission parameters.

L(n) = Link([θ d a α]) is used to define the DH parameters of the n-th link of robot manipulator.

SerialLink

SerialLink is the constructor function to build robot manipulator using the pre-defined “Link” objects.

“ tx90 = SerialLink(L, 'name', 'TX90') “ entry in m.file results in a SerialLink object in which robot manipulator with the written DH parameters are encapsulated. This SerialLink is within the variable name we use in the left hand side of the equation. In our case it is named as “tx90”.

“ tx90.tool = transl(x, y, z) “ function is used to offset the tool from the last coordinate frame by an amount of x, y and z along the x, y and z axes of the last coordinate frame.

“tx90.payload(m, [x, y, z])” is used to add an m kg mass at the tip of the end effector with respect to the coordinate frame of the last link at a point away from x, y and z unit along x, y and z axes of the coordinate frame respectively.

tx90.links(n).m = m_n creates an object defining the mass of the n-th link of robot manipulator.

tx90.links(n).r = $[r_{cx}, r_{cy}, r_{cz}]$ is used to define the location of the COM of n-th linakge with respect to the n-th coordinate frame.

“ tx90.links(n).I = Jma “ function is used to enter the motor inertia taken from catalogue for the n-th link.

tx90.links(n).I = [Ixx Ixy Ixz; Iyx Iyy Iyz; Izx Izy Izz] is the function used to enter the inertia of the n-th link with respect to the coordinate frame alinged with

“tx90.links(n).B = c” is used to enter the link viscous friction.

“tx90.links(n).Tc = c” is used to enter the Coloumb friction.

“ tx90.links(n).G = r_n” is the function used to set the gear ratio of the n-th joint servo motor.

“M = tx90.inertia(th)” is the function used to calculate the inertia matrix of the system for an instant when the robot manipulator is passing through the joint displacements stored in “th” vector.

“V = tx90.coriolis(th, dth)” is the function used to calculate the Coriolis matrix of the system for an instant when the robot manipulator is passing through the joint displacements stored in “th” vector at a joint velocity stored in “dth” vector.

“G = tx90.gravload(th)” is the function used to calculate the gravity matrix of the system for an instant when the robot manipulator is passing through the joint displacements stored in “th”.

“RNE Block” is used to computed the required torque for the given joint coordinates, velocities and acclerations.

“Robot Block” is used to compute the instaneous joint coordinates, velocities and acclerations for the given applied joint torque.

APPENDIX C

CALCULATION OF INERTIA MATRIX AND LINK MASSES

Link-1

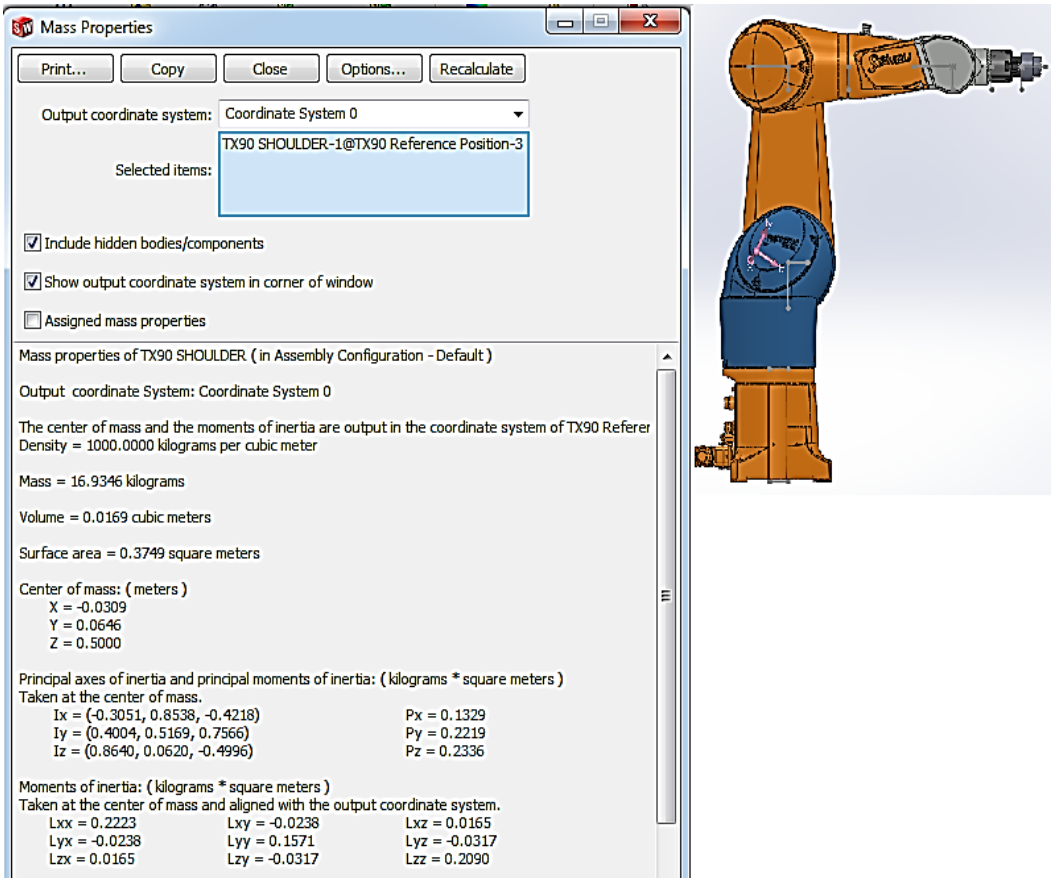


Figure 1. Mass properties of link-1

The actual mass of the first link of Staubli TX90 is assumed to be 33.935kg, which is two times larger than the Solidworks value.

Therefore the moment of inertia taken at the center of mass aligned with the output coordinate system which, in this case, is the coordinate system 0 written on the "Mass Properties" window is multiplied with two to get consistency in results.

Moment of inertia of link 1 taken at the COM aligned with the output coordinate system ($kg.m^2$)

Lxx =	0.4446	Lxy =	-0.0476	Lxz =	0.033
Lyx =	-0.0476	Lyy =	0.3142	Lyz =	-0.0634
Lzx =	0.033	Lzy =	-0.0634	Lzz =	0.418

Link-2

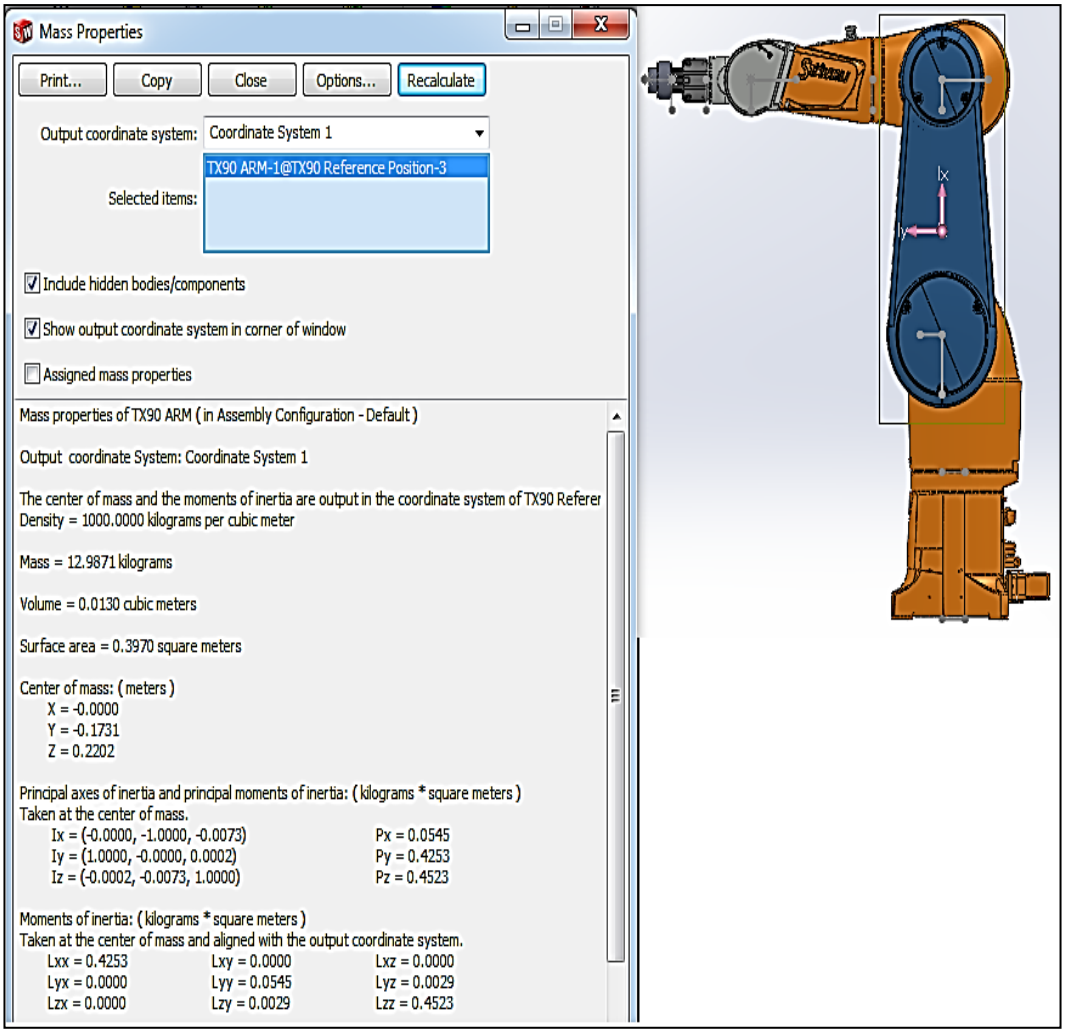


Figure 2. Mass properties of link-2

The actual mass of the second link of Staubli TX90 is assumed to be 25.937kg, which is two times larger than the Solidworks value.

Therefore the moment of inertia taken at the center of mass aligned with the output coordinate system which, in this case, is the coordinate system 1written on the "Mass Properties" window is multiplied with two to get consistency in results.

Moment of inertia of link 2 taken at the COM aligned with the output coordinate system ($kg.m^2$)

Lxx =	0.8506	Lxy =	0	Lxz =	0
Lyx =	0	Lyy =	0.109	Lyz =	0.0058
Lzx =	0	Lzy =	0.0058	Lzz =	0.9046

Link-3

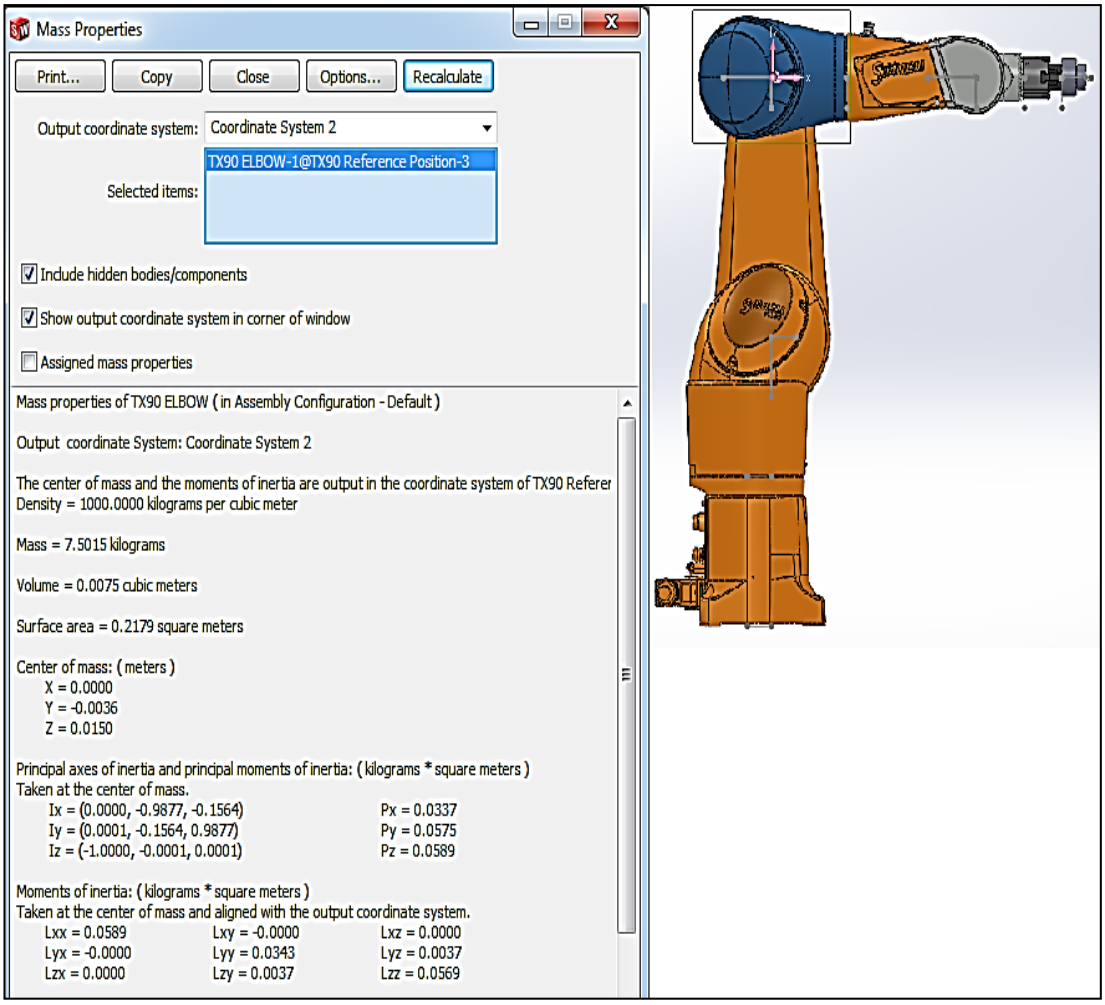


Figure 3. Mass properties of link-3

The actual mass of the third link of Staubli TX90 is assumed to be 15.502kg, which is two times larger than the Solidworks value.

Therefore the moment of inertia taken at the center of mass aligned with the output coordinate system which, in this case, is the coordinate system 2written on the "Mass Properties" window is multiplied with two to get consistency in results.

Moment of inertia of link 3 taken at the COM aligned with the output coordinate system ($kg.m^2$)

Lxx =	0.1178	Lxy =	0	Lxz =	0
Lyx =	0	Lyy =	0.0686	Lyz =	0.0074
Lzx =	0	Lzy =	0.0074	Lzz =	0.1138

Link-4

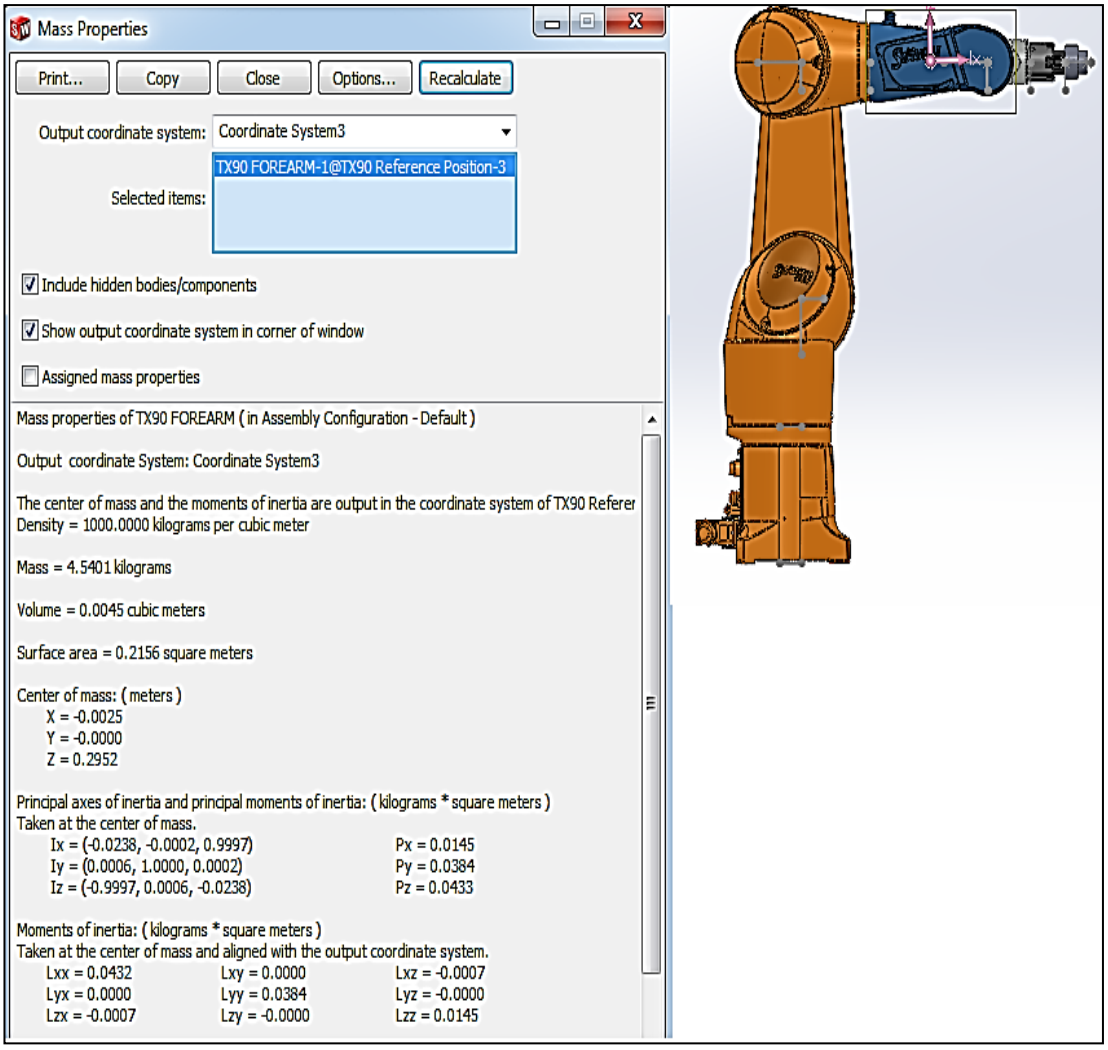


Figure 4. Mass properties of link-4

The actual mass of the fourth link of Staubli TX90 is assumed to be 9.540 kg, which is two times larger than the Solidworks value.

Therefore the moment of inertia taken at the center of mass aligned with the output coordinate system which, in this case, is the coordinate system 3 written on the "Mass Properties" window is multiplied with two to get consistency in results.

Moment of inertia of link 4 taken at the COM aligned with the output coordinate system ($kg.m^2$)

Lxx =	0.0864	Lxy =	0	Lxz =	-0.0014
Lyx =	0	Lyy =	0.0768	Lyz =	0
Lzx =	-0.0014	Lzy =	0	Lzz =	0.029

Link-5

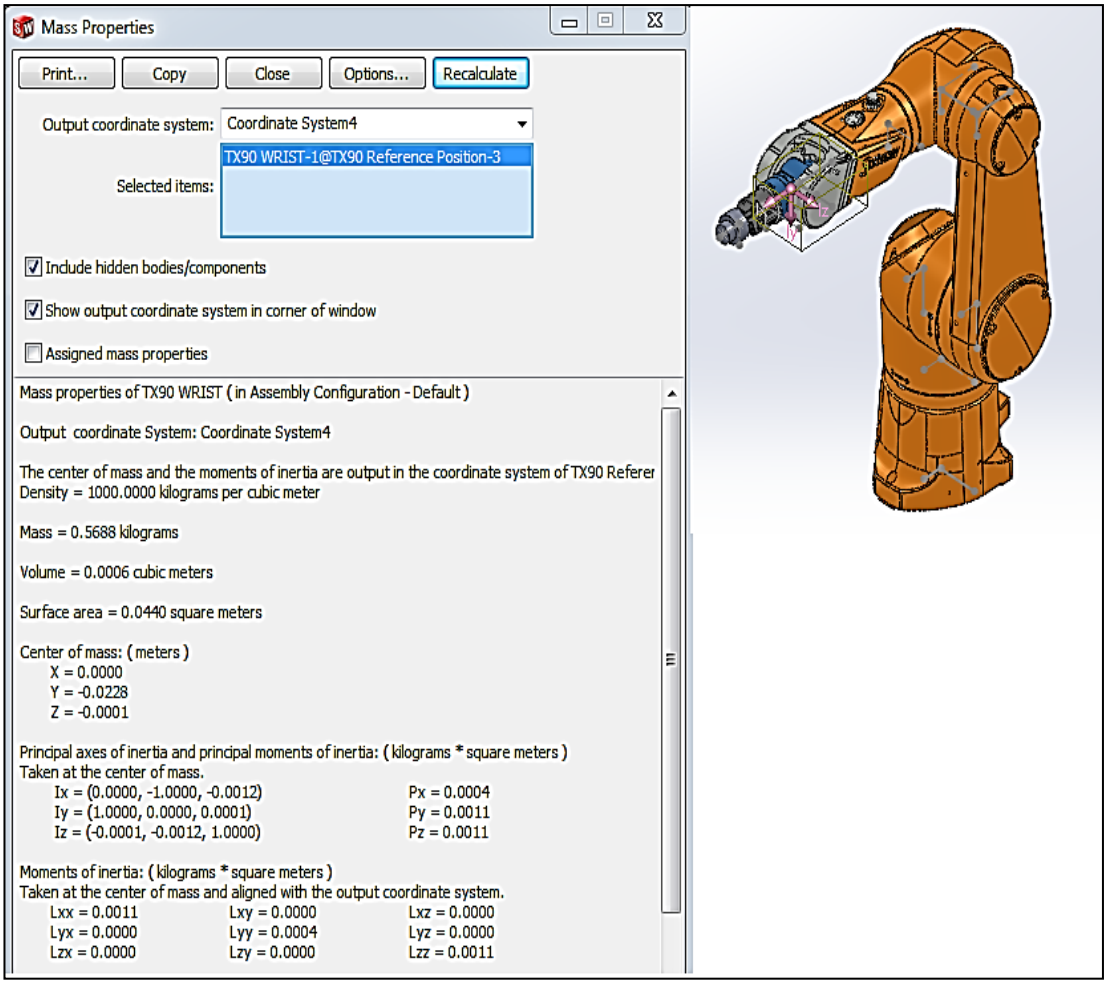


Figure 5. Mass properties of link-5

The actual mass of the fifth link of Staubli TX90 is assumed to be 2.569 kg, which is two times larger than the Solidworks value.

Therefore the moment of inertia taken at the center of mass aligned with the output coordinate system which, in this case, is the coordinate system 3 written on the "Mass Properties" window is multiplied with two to get consistency in results.

Moment of inertia of link 5 taken at the COM aligned with the output coordinate system ($kg.m^2$)

Lxx =	0.0022	Lxy =	0	Lxz =	0
Lyx =	0	Lyy =	0.0008	Lyz =	0
Lzx =	0	Lzy =	0	Lzz =	0.0022

Link-6

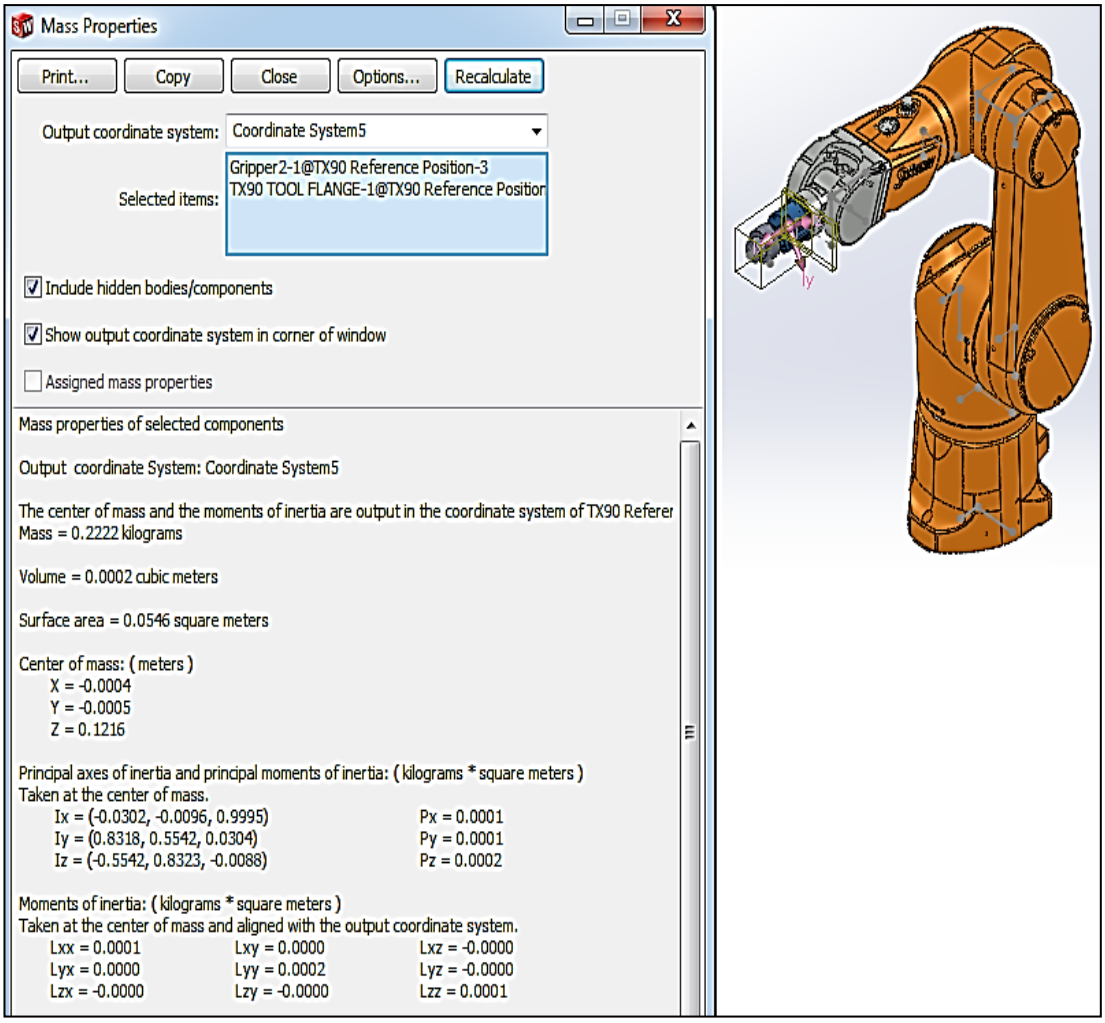


Figure 6. Mass properties of link-6

The actual mass of the fifth link of Staubli TX90 is assumed to be 1.191 kg, which is two times larger than the Solidworks value.

Therefore the moment of inertia taken at the center of mass aligned with the output coordinate system which, in this case, is the coordinate system 3 written on the "Mass Properties" window is multiplied with two to get consistency in results.

Moment of inertia of link 6 taken at the COM aligned with the output coordinate system ($kg.m^2$)

Lxx =	0.0002	Lxy =	0	Lxz =	0
Lyx =	0	Lyy =	0.0004	Lyz =	0
Lzx =	0	Lzy =	0	Lzz =	0.0002

APPENDIX D

DETERMINATION OF THE MOTOR CONSTANTS

Staubli TX90 is assumed to use 750 W AC type servo motors for controlling the first three axes whereas for the other remaining three axes 200W AC type servo motors are supposed to be used and Yaskawa SGMAS 08A□A is assumed to be used in 1st, 2nd and 3rd axes and Yaskawa SGMAS 02A□A is supposed to be used in 4th, 5th and 6th joints of Staubli TX90.

The following table summarizes the technical data of the AC servos used in the robot manipulator. Torque constants are taken from SGMAS catalogue and speed constants in addition to the armature resistance and armature inductance are taken from BLS technical data sheet. BLS-71 & BLS-72 resembles SGMAS 02A & SGMAS 08A AC servo motors.

Table 1. Product specifications of AC servo motors used in TX90 for the analysis

	SGMAS 02A□A	SGMAS 08A□A
Supply Voltage	200 VAC	200 VAC
Rated Output	200 W	750 W
Rated Speed	3000 rpm (314 rad/s, 18000deg/s)	3000 rpm (314 rad/s, 18000deg/s)
Max. Speed	6000 rpm (628 rad/s, 36000deg/s)	6000 rpm (628 rad/s, 36000deg/s)
Rated Angular Acc.	54900 rad/s^2	31000 rad/s^2
Rated Torque	0.637 N.m	2.39 N.m
Gear Type	HDS planatery	HDS planatery
Gear Ratio	1/5	1/5
Moment of Inertia Motor + Gear	0.451*10^{-4} kg.m^2	1.35*10^{-4} kg.m^2
Torque Constant (KT)	0.375 N.m/A_{rms}	0.487 N.m/Arms
Speed Constant (KE)	0.22	0.28
Armature Resistance (Ra)	0.832 Ω	9.4 Ω
Armature Inductance (La)	2.73*10^{-3} H	12.77*10^{-3} H

SGMAS TECHNICAL SPECIFICATIONS-1

3.2 Ratings and Specifications of SGMAS Servomotors

3.2.1 SGMAS Servomotors without Gears

(1) Ratings and Specifications

- Time Rating: Continuous
- Vibration Class: 15 μm or below
- Insulation Resistance: 500 VDC, 10 MΩ min.
- Ambient Temperature: 0 to 40°C
- Excitation: Permanent magnet
- Mounting: Flange method
- Thermal Class: B
- Withstand Voltage: 1500 VAC for one minute
- Enclosure: Totally enclosed, self-cooled, IP55 (except for shaft opening)
- Ambient Humidity: 20% to 80% (no condensation)
- Drive Method: Direct drive

Ratings and Specifications for SGMAS Servomotors without Gears

Voltage		200 V							
Servomotor Model SGMAS-		A5A	01A	C2A	02A	04A	06A	08A	12A
Rated Output*1	W	50	100	150	200	400	600	750	1150
Rated Torque*1, *2	N·m	0.159	0.318	0.477	0.637	1.27	1.91	2.39	3.66
Instantaneous Peak Torque*1	N·m	0.477	0.955	1.43	1.91	3.82	5.73	7.16	11.0
Rated Current*1	A_{rms}	0.66	0.91	1.8	1.9	2.6	4.3	5.4	8.5
Instantaneous Max. Current*1	A_{rms}	2.1	2.8	5.7	6.5	8.5	13.6	16.9	26.0
Rated Speed*1	min^{-1}	3000							
Max. Speed*1	min^{-1}	6000							
Torque Constant	$N·m/A_{rms}$	0.265	0.375	0.284	0.375	0.527	0.496	0.487	0.467
Rotor Moment of Inertia	$kg·m^2×10^{-4}$	0.0242	0.0380	0.0531	0.116	0.190	0.326	0.769	1.20
Rated Power Rate*1	kW/s	10.4	26.6	42.8	35.0	84.9	112	74.1	112
Rated Angular Acceleration*1	rad/s^2	65800	83800	89900	54900	67000	58600	31000	30500
Applicable SERVOPACK	SGDS-	A5	01	02	02	04	08	08	15

* 1. These items and torque-motor speed characteristics quoted in combination with an SGDS SERVOPACK are at an armature winding temperature of 100°C. Other values quoted at 20°C.

* 2. Rated torques are continuous allowable torque values at 40°C with an aluminum heat sink of the following dimensions attached.
SGMAS-A5, -01, -C2, -02, -04 and -08: 250 mm × 250 mm × 6 mm
SGMAS-06: 300 mm × 300 mm × 12 mm
SGMAS-12: 350 mm × 350 mm × 12 mm

Note: If the heating conditions are more severe than the *2 conditions above, continuous allowable torque decreases. Contact your Yaskawa representative.

Figure 7. Ratings and specifications of SGMAS AC servo motors

SGMAS TECHNICAL SPECIFICATIONS-2

Ratings and Specifications for SGMAS Servomotors with Low-backlash Gears

Servomotor Model SGMAS-	Servomotor			Gear					
	Out-put (W)	Rated Speed (min^{-1})	Rated Torque (N·m)	Gear Ratio	Lost Motion (arc-min)	Rated Torque /Effi-ciency*2 (N·m)/(%)	Instanta-neous Peak Torque (N·m)	Rated Speed (min^{-1})	Max. Speed*1 (min^{-1})
A5A□AH1□	50	3000	0.159	1/5	3	0.557/70	1.67	600	800
A5A□AH2□				1/9		1.00/70	3.01	333	444
A5A□AHC□				1/21		2.67/80	8.02	143	190
A5A□AH7□				1/33		4.20/80	12.6	91	121
01A□AH1□	100	3000	0.318	1/5	3	1.27/80	3.82	600	800
01A□AHB□				1/11		2.80/80	8.41	273	363
01A□AHC□				1/21		5.34/80	18.2	143	190
01A□AH7□				1/33		8.40/80	28.7	91	121
C2A□AH1□	150	3000	0.477	1/5	3	1.91/80	5.73	600	800
C2A□AHB□				1/11		4.20/80	12.6	273	363
C2A□AHC□				1/21		8.01/80	24.0	143	190
C2A□AH7□				1/33		12.6/80	37.8	91	121
02A□AH1□	200	3000	0.637	1/5	3	2.55/80	8.4	600	800
02A□AHB□				1/11		5.96/85	19.3	273	363
02A□AHC□				1/21		11.4/85	37.3	143	190
02A□AH7□				1/33		17.9/85	58.6	91	121
04A□AH1□	400	3000	1.27	1/5	3	5.40/85	17.6	600	800
04A□AHB□				1/11		11.9/85	39.1	273	363
04A□AHC□				1/21		22.7/85	72.2	143	190
04A□AH7□				1/33		33.5/80	115	91	121
06A□AH1□	600	3000	1.91	1/5	3	8.12/85	24.3	600	800
06A□AHB□				1/11		17.9/85	53.7	273	363
06A□AHC□				1/21		34.1/85	102	143	190
06A□AH7□				1/33		50.4/80	151	91	121
08A□AH1□	750	3000	2.39	1/5	3	10.2/85	33.3	600	800
08A□AHB□				1/11		22.3/85	71.0	273	363
08A□AHC□				1/21		42.7/85	140	143	190
08A□AH7□				1/33		67.0/85	206	91	121
12A□AH1□	1150	3000	3.66	1/5	3	15.6/85	46.7	600	800
12A□AHB□				1/11		34.2/85	103	273	363
12A□AHC□				1/21		65.3/85	196	143	190
12A□AH7□				1/33		96.6/80	290	91	121

Figure 8. Maximum torque and power values of SGMAS AC servo motors with different gear ratios

SGMAS TECHNICAL SPECIFICATIONS-3

Ratings and Specifications for SGMAS Servomotors with Low-backlash Gears (cont'd)

Servomotor Model SGMAS-	Moment of Inertia	
	Motor + Gear ($kg \cdot m^2 \times 10^{-4}$)	Gear ($kg \cdot m^2 \times 10^{-4}$
A5A□AH1□	0.054	0.030
A5A□AH2□	0.052	0.028
A5A□AHC□	0.042	0.019
A5A□AH7□	0.037	0.013
01A□AH1□	0.116	0.078
01A□AHB□	0.086	0.048
01A□AHC□	0.081	0.043
01A□AH7□	0.071	0.033
C2A□AH1□	0.131	0.078
C2A□AHB□	0.124	0.071
C2A□AHC□	0.096	0.043
C2A□AH7□	0.085	0.032
02A□AH1□	0.451	0.335
02A□AHB□	0.201	0.085
02A□AHC□	0.226	0.110
02A□AH7□	0.181	0.065
04A□AH1□	0.525	0.335
04A□AHB□	0.385	0.195
04A□AHC□	0.385	0.195
04A□AH7□	0.363	0.173
06A□AH1□	0.661	0.335
06A□AHB□	0.518	0.192
06A□AHC□	0.904	0.578
06A□AH7□	0.501	0.173
08A□AH1□	1.35	0.583
08A□AHB□	1.30	0.528
08A□AHC□	1.36	0.593
08A□AH7□	1.03	0.263
12A□AH1□	1.78	0.583
12A□AHB□	2.30	1.10
12A□AHC□	1.79	0.593
12A□AH7□	6.14	4.94

Figure 9. Moment of inertia values of SGMAS AC servo motor gear with/without motor shaft

MAVILOR BLS AC SERVO MOTOR TECHNICAL SPECIFICATION

Table 2. BLS Technical Specifications

BLS ~ Technical Specifications

ALL CHARACTERISTICS MEASURED AT 25°C AMBIENT TEMPERATURE	SYMBOLS	UNITS	BLS-71		BLS-72	
			220 VAC	400 VAC	220 VAC	400 VAC
MAX MECHANICAL SPEED	n	rpm	11,000		11,000	
STALL TORQUE (1) ±10%	M_S	Nm	0.8	0.8	1.9	1.9
STALL CURRENT	I_S	A	2.11	1.13	3.96	2.37
PEAK TORQUE ±10%	M_J	Nm	3.2	3.2	7.6	7.6
TORQUE-WEIGHT RATIO	T_W	Nm/kg	0.53	0.53	1	1
EMF CONSTANT ±5%	K_E	Vs/rad	0.22	0.41	0.28	0.46
TORQUE CONSTANT ±5%	K_T	Nm/A	0.38	0.71	0.48	0.8
RELUCTANCE TORQUE (*)	T_R	Nm	<3.5%		<3.5%	
WINDING RESISTANCE ±5%	R	Ω	10.7	33.8	5.3	15.5
WINDING INDUCTANCE ±5%	L	mH	7.4	24	5.4	13.2
ROTOR INERTIA	J	kg $m^2 10^{-3}$	0.027	0.027	0.051	0.051
MECHANICAL TIME CONSTANT	τ_M	ms	3.46	3.14	2.01	2.15
ELECTRICAL TIME CONSTANT	τ_E	ms	0.69	0.71	1.02	0.85
THERMAL TIME CONSTANT	τ_{TH}	s	1,100	1,100	1,280	1,280
THERMAL RESISTANCE	R_{TH}	°C/W	1.21	1.34	0.69	0.69
MASS	M	kg	1.5	1.5	1.9	1.9
RADIAL LOAD (at mid-length of shaft)	F_R	N	216		245	
AXIAL LOAD	F_A	N	98		98	
INSULATION			CLASS-F		CLASS-F	
PROTECTION			IP-65		IP-65	
(1) With an aluminium heat sink plate			300x300x10		300x300x10	
(*) Respect to the Stall Torque						

APPENDIX E

STAUBLI TX90 SPECIFICATIONS

MODEL		TX90	TX90L	TX90XL
Characteristics				
Maximum load (1)		20 kg	15 kg	9 kg
Nominal load		7 kg	6 kg	5 kg
Reach (between axis 1 and 6)		1000 mm	1200 mm	1450 mm
Number of degrees of freedom		6	6	6
Repeatability - ISO 9283		± 0,03 mm	± 0,035 mm	± 0,04 mm
Motion range	Axis 1 (A)	± 180°	± 180°	± 180°
	Axis 2 (B)	+147,5°/-130°	+147,5°/-130°	+147,5°/-130°
	Axis 3 (C)	± 145°	± 145°	± 145°
	Axis 4 (D)	± 270°	± 270°	± 270°
	Axis 5 (E)	+140°/-115°	+140°/-115°	+140°/-115°
	Axis 6 (F)	± 270° (2)	± 270° (2)	± 270° (2)
Work envelope	Maximum reach between axis 1 and 5 (R. M)	900 mm	1100 mm	1350 mm
	Minimum reach between axis 1 and 5 (R. m1)	850 mm	1050 mm	1300 mm
	Minimum reach between axis 2 and 5 (R. m2)	200 mm	272 mm	327 mm
	Minimum reach between axis 3 and 5 (R. b)	256 mm	320 mm	391 mm
Maximum speed	Axis 1	425 mm	550 mm	650 mm
	Axis 2	400°/s	400°/s	400°/s
	Axis 3	400°/s	390°/s	350°/s
	Axis 4	430°/s	400°/s	410°/s
	Axis 5	540°/s	540°/s	540°/s
	Axis 6	475°/s	475°/s	475°/s
	Axe 6	760°/s	760°/s	760°/s
Maximum speed at load gravity center		10,42 m/s	10,54 m/s	11,09 m/s
Maximum inertias	Axis 5	1,5 kg.m²	1,25 kg.m²	1 kg.m²
	Axis 6	0,25 kg.m²	0,20 kg.m²	0,15 kg.m²
Weight		111 kg	114 kg	116 kg

Motion range

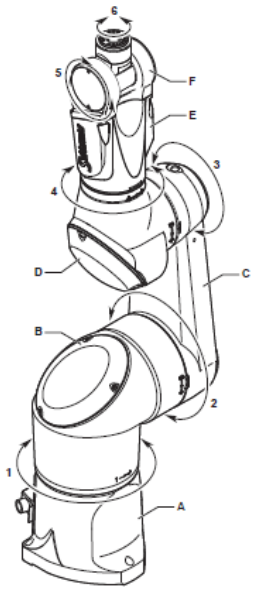


Figure 10. Staubli TX90 specifications

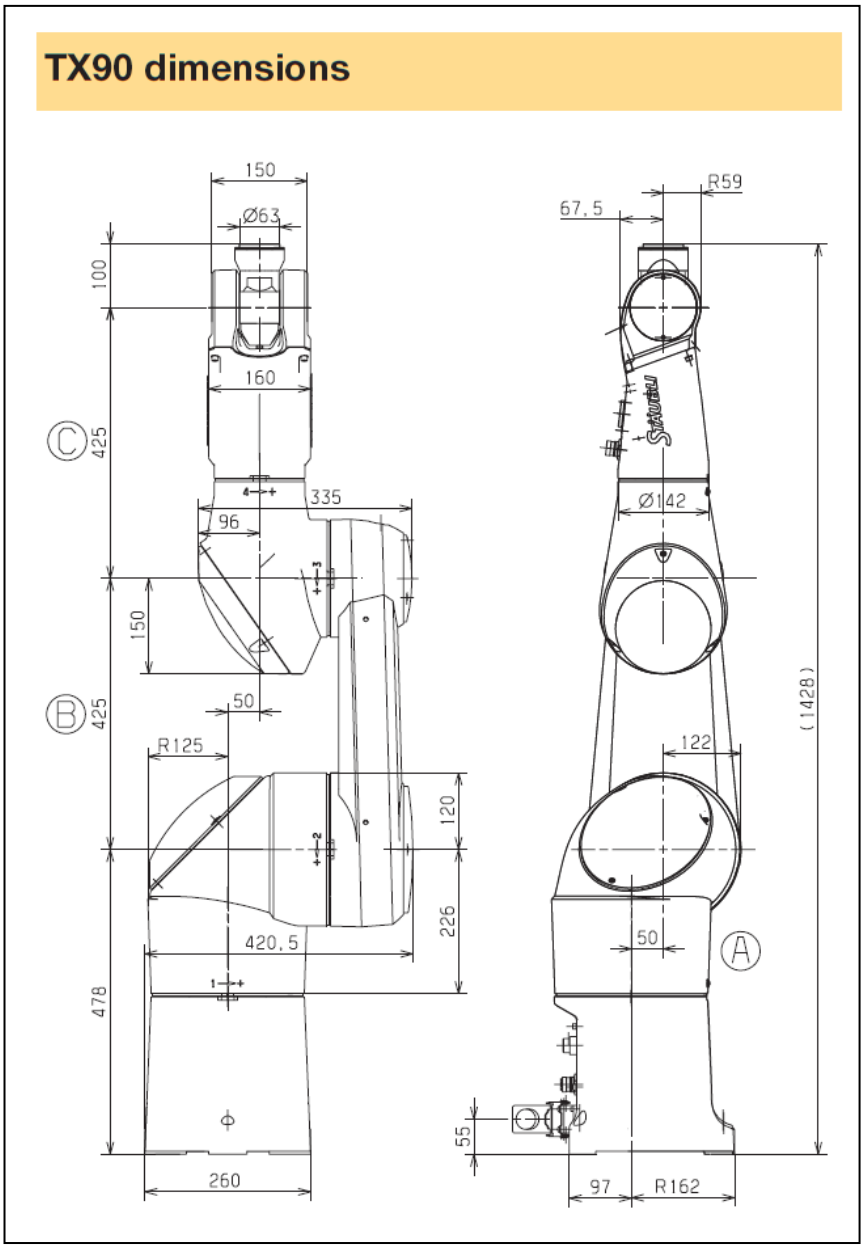


Figure 11. Staubli TX90 dimensions