Islamic University of Gaza
Deanery of Graduate Studies
Faculty of Engineering
Electrical Engineering Department

Master Thesis

Simulation and Interfacing of 5 DOF Educational Robot Arm

Mohammed Reyad AbuQassem

Advisors
Dr. Hatem Elaydi
Dr. Iyad Abuhadrous

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Science in Electrical Engineering

June 2010
جمادي الثاني 1431هـ

صفحة نتيجة الحكم على البحث (نتيجة الحكم من قبل لجنة المناقشة)

ABSTRACT

Many universities and institutes experience difficulty in training people to work with expensive equipments. A common problem faced by educational institutions concerns the limited availability of expensive robotics equipments, with which students in the academic programs can work, in order to acquire valuable "hands on" experience. Therefore, the Robot Simulation Software (RSS) nowadays is paramount important. Moreover, hands on experience with programmable robots gives student great understanding.

This work reports the development of a visual software package where an AL5B Robot arm has been taken as a case study. It adopts the virtual reality interface design methodology and utilizes MATLAB/Simulink and AutoCAD as tools for testing motional characteristics of the AL5B Robot arm. Moreover, the developed model is implemented and tested in order to analyze and improve the algorithms of Kinematics, Inverse Kinematics, Velocity Kinematics "Jacobian" and Trajectory Planning. The package life cycle is documented. Then, a comparison between the simulated package and the physical arm is accomplished in terms of motion, trajectories, and kinematics.

The developed package is used as an educational tool in to enhance the applied and experimental research opportunities and it improve the robotics curricula at the graduate and undergraduate levels.

Keywords: Virtual Reality, Modeling, Simulation, Interface, MATLAB/Simulink, AL5B Robot arm, Forward Kinematics, Inverse Kinematics, Trajectory Planning, Jacobian

ملخص

ربط ومحاكاة لذراع روبوت تعليمي ذو خمس مفاصل

تواجه الكثير من الجامعات والمعاهد صعوبة في تدريب الطلاب على العمل مع معدات باهظة الثمن. وهناك مشكلة مشتركة تواجهها المؤسسات التعليمية هي قلة توافر الروبوتات وذلك لإرتفاع ثمنها. والتي يستخدمها الطلاب في البرامج التعلمية من أجل الحصول على خبرة قيمة في مجال التحكم .لذلك فإن تصميم برمجيات لعمل محاكاه مع الربوت له اهمية قصوي في الوقت الحاضر. علاوةً علي ذلك الخبرة العملية في برمجة أجهزة الروبوت تعطي الطلبة فهم معمق.

هذا العمل سوف يقدم تفصيلاً عن وضع برامج المحاكاه حيث استخدم ذراع الربووت AL5B كدراسة حالة. حيث انها تعتمد على ربط الواقع الافتراضي بتصميم منهجي باستخدام برنامج MATLAB/Simulink وأيضاً برنامج أوتوكاد لعمل نموذج لهذا الربوت وايضاً استخدامها كأدوات لاختبارالتصميم و الخصائص الحركية لذراع الروبوت من نوع AL5B. سيتم تطوير نموذج محاكاة لهذا الروبوت عن طريق تحليل الحركة الامامية ، الحركة العكسية ، السرعة الحركية وايضاً مسار الحركة. وسيتم تنفيد جميع ما ذكر واختباره. ثم سنقوم في النهاية بعمل مقارنة بين نموذج المحاكاه المطور مع الذراع الحقيقي المستخدم في هذه الدراسة بالنسبة للحركة والمسارات.

وسوف يُستخدم هذا البرنامج المُتطور كأداة تعليمية. ونتوقع ان يساهم هذا العمل في زيادة فرص التعليم والتدريب والبحث وايضاً امكانية تطوير مناهج الروبوتات للدراسات العليا والبكالوريوس.

To my beloved parents...

...brother and sisters..

..My wife and my beloved son Mohammed

ACKNOWLEDGEMENT

At the very outset, all my prayers and thankfulness are to Allah the almighty for facilitating this work and for granting me the opportunity to be surrounded by great and helpful people at IUGAZA and PTC.

I would like to express my everlasting gratitude to my supervisors, Dr. Hatem Elaydi and Dr. Iyad Abuhadrous for their valuable encouragement, guidance and monitoring, without which this work would not have reached the point of fruition, so I ask Allah to reward them on my behalf.

I would never have attempted to obtain a master degree if it weren't for the continuous encouragement that I received from my father, the man to whom I will be grateful, for the rest of my life.

The warm heart, my mother, deserves all the credit here; she has been a source of inspiration to me for years. I would never forget her continuous prayer for the sake of my success.

No acknowledgement would be complete without expressing my appreciation and thankfulness for my wife; I can't refute her long lasting patience and support which she showed during this work and which was essential to accomplish it.

My siblings, to whom I belong, have shadowed me with their concern all the time, so they deserve my acknowledgement too.

Finally, I am grateful to my dear friend, Eng. Sa'id Ibrahim Abu Al-Roos to help me in completing this work.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

NOMENCLATURE

3D	*Three-Dimensional Graphics*
DOF	*Degrees of Freedom*
GUI	*Graphical User Interface*
RSS	*Robot Simulation Software*
VSP	*Visual Software Program*
PC	*Personal Computer*
DH	*Denavit-Hartenberg*
FK	*Forward Kinematic*
IK	*Inverse Kinematic*
RRP	*Revolute – Revolute - Prismatic*
RRR	*Revolute – Revolute - Revolute*
WARTG	*Wrist Angle Relative to Ground*
TP	*Trajectory Planning*
LSPB	*Linear Segments with Parabolic Blends*
ADC	*Analogue to Digital Converter*
PWM	*Pulse-Width Modulation*
PTC	*Palestine Technical Collage*

ABBREVIATIONS

T_i	*Homogenous Transformation Matrix*
a_i	*Link Length*
α_i	*Link Twist of Two Joints*
d_i	*Link Offset*
θ_i	*Joint Angle*
$c_{\theta i}$	*Cosine* θi
$s_{\theta i}$	*Sin* θi
q_i	*Joint angle*
R	*Rotation Matrix*
θ_{234}	$\theta_2 + \theta_3 + \theta_4$
$\dot{\theta}$	*Joint Velocities*
$\dot{p}$	*Linear Velocities*
ω	*Angular Velocities*
J	*Jacobian Matrix*
A^+	*Pseudo Inverse*
ζ	*End-Effector Velocity*
F	*Forces*
τ	*Torques*
δ	*Displacement Caused by the Force*
$q(t)$	*Angle Move*
$\dot{q}(t)$	*Angle Velocity*
$\ddot{q}(t)$	*Angle Acceleration*
q_0	*Initial Angle Value*
q_f	*Final Angle Value*
v_0	*Initial Angle Velocity*
v_f	*Final Angle Velocity*
t_0	*Initial Time Value*
t_f	*Final Time Value*
rad	*Radian*
deg	*Degrees*

CHAPTER 1 INTRODUCTION

1.1 Motivation

Over the last two decades, robotics education has been based on mobile robotics and manipulator-based robotics. The accessibility of small inexpensive mobile robots has promoted their use in the classroom across abroad spectrum of educational levels all over the world [KOL 01]; however, robotics is still an emerging topic in Gaza strip and access to commercial robots is next to impossible.

Researchers around the world developed educational models and exposed kindergarten students [MIL 00] and middle to high school students [WED 02] to hands-on learning employing mobile robotics. Robotics education on undergraduate and graduate levels is still the main focus of educators [FER 00].

Manipulator- based robotics education requires a large startup investment; thus, did not enjoy sharp exposure. Murphy [MUR 00] promoted the use of robotics to teach artificial intelligence and offered hands-on learning and robot contests [MUR 01]. Sutherland [SUT 00] described a successful approach to expose undergraduate student to robotics with limited resources. Palestine Technical College at Deir el Balah (PTCDB) holds an annual contest and is open to all students [PTCDB].

The outcomes of this study serve these universities by developing software package to be used as an educational tool for robotics classes by enhancing the course with simulation and practical lab. The devolved package enriches the blended theoretical robotics presentations introduced in these universities.

1.2 Problem Statement and Goal

There are several universities and colleges in Gaza strip that teach robotics course; namely, the Islamic university, Alazhar University, and Palestine Technical College. Robotics courses at local universities are mainly theoretical; there are no practical labs to apply the theoretical concepts of these courses. The goal of this research is to develop a visual software package, which simulates a 5DOF robot arm; this package will cover most of the important topics given in the introductory course in robotics manipulators. This will increase the education, training and research in graduate and undergraduate studies in the robotics field, taking into consideration the limited availability of educational tools for robotics courses and the high cost of robot equipments and tools.

The AL5B robot arm [LYN 06] presents a simple inexpensive solution and a good example for robotic manipulators, this arm is chosen as a case study in this research. MATLAB/Simulink and AutoCAD will be used for testing motional characteristics of the arm. A complete study and mathematical analysis for the forward kinematics, inverse kinematics, velocity kinematics (Jacobian), and trajectory planning problems is presented, implemented and tested. An interfacing card is designed and developed. The developed algorithms were implemented and applied to the AL5B physical arm. A comparison between the kinematic solutions of the developed software package with the robot arm's physical motional behaviors is discussed.

1.3 System Overview

The complete system block diagram shown in Figure (1.1) consists of many parts like, personal computer with serial communication adapter, CUBLOC microcontroller [COM 05] and AL5B Robot arm. The Graphical User Interface (GUI), designed by MATLAB software, consists of four parts; forward, inverse kinematic, path and trajectory planning, Jacobian and controller. The forward kinematics consists of finding the position of the end-effector in the space knowing the movements of its joints. The inverse kinematics consists of the determination of the joint variables corresponding to a given end-effector position and orientation. Path is defined as sequence of robot configurations in particular order without regard for timing of these configurations, trajectory is concerned about when each part of the path must be obtained thus specifying timing. Each joint velocity at the specified joint positions needs to be found; this is accomplished using Jacobian. The last part is the simple controller block used to control the robot arm by GUI program.

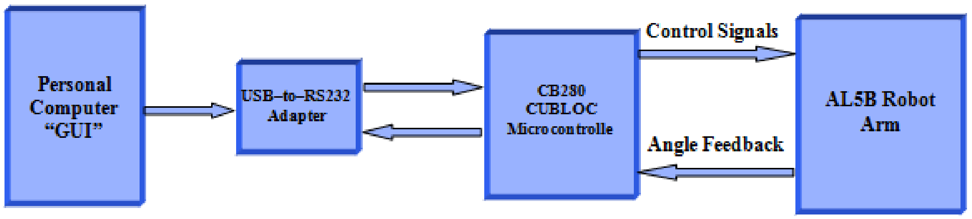


Figure (1.1): System Block Diagram

Serial communication is the simplest way to communicate between two devices. A serial interface is established through a serial port object, which can be created using the SERIAL function by MATLAB. The main function of the CUBLOC microcontroller is making interface between PC and AL5B robot arm by receiving data from serial port and sending this data to the arm servo motors. then feeding the data from servo motors encoders back to the PC through serial port.

1.4 Literature Review

Many industrial robot arms are built with simple geometries such as intersecting or parallel joint axes to simplify the associated kinematics computations [MAN 96]. However, their costs are high for students and research workers. AL5B is a good alternative for such robot manipulators, because it is inexpensive, flexible and similar to industrial robot arms.

Papers that developed software for modeling 2D and 3D robots arm such as [MAN 96, KOY 07 and GUR 97], forward and inverse Kinematic are analyzed and then according to the model a computer simulation is generated, a simulation and testing characteristics of this robot arm is prepared by a programming languages. 2D and 3D visualization are used to build GUI friendly for users as educational tool. The software is incomplete, because it did not investigate anything about the path and trajectory planning.

[PAS 07] used V-Realm Builder 2.0 and Simulink for virtual reality prototyping and testing the viability of designs before the implementation phase for the industrial SCARA robot, located in the Control Robot Lab of the University of Oradea. In addition, they illustrated the use of the 3D Joystick for manipulating objects in a virtual world.

Martin and Arya in [ROH 00, WIR 04], developed Robot Simulation Software for forward and inverse kinematic using VRML and MATLAB Simulink. The output of the system had good graphic capability and flexibility in terms of 3D representation. However, the system was not able to run as stand-alone application and was not user friendly.

[JAM 08] reported the development of the Robot Simulation Software (RSS) where a Mitsubishi RV-2AJ robot was taken as a case study. The project adopted the virtual reality interface design methodology and utilized MATLAB/Simulink and V-Realm Builder as tools. A robot model was developed and a RSS software life cycle was implemented.

[MAR 06] presented a Visual C++ and OpenGL application for 3D simulation of the serial industrial robots. It started from the forward kinematics of the robot taken into consideration. The functions implemented in the source code are able to calculate the position and orientation of each robot joint, including the position and orientation of the robot gripper. With the help of the OpenGL functions, the application was able to draw and simulate the 3D kinematic scheme of the robot.

An approach proposed to develop real-time simulators of complex electromechanical systems by exploiting the most powerful non real-time modeling and control design tools in [FER 08]. This approach relied on standard and commercial tools and on open source packages, and required the development of few interface blocks to be included within the Simulink and Dymola models, respectively. The modeling and validation work carried out on a joint prototype in the early phase of the arm development process could be fully included in the real-time simulation model, achieving quite accurate and reliable results almost effortlessly. The Simulink arm controller description can also be easily tested in an incremental way. A significant effort was devoted to create a human machine interface able to support the input of motion commands and force disturbances, together with the 3D visualization of the arm motion, relying on a powerful open source package.

There are large amount of literature which discuses the kinematics analysis of industrial robots [CRA 05]. The majority of them shy away from discussing the low cost educational robot arms. After going over the last group of papers, we can notice that none of them gives a complete educational tool to control AL5B robot arm for student at college level. Thus, this research will study mathematical model and kinematical analysis of the AL5B educational robot. A Visual Software Program (VSP) will be also developed to show the robot arm motion with respect to its mathematical analysis and interfacing with physical robot.

1.5 Objectives

In order to achieve the main goal objectives of this study, the work is going to be divided into two phases.

Phase 1

1) Developing a visual software package, for testing motional characteristics of the AL5B Robot arm

   a. Drawing a 3D Model for the Robot Arm using MATLAB/Simulink and AutoCAD.
   b. Designing a Graphical User Interface "GUI" using MATLAB.
2) Derivation of a complete kinematic model for the robot.
   a. Studying the theory of kinematics in order to analyze of the 5 DOF AL5B Robot Arm.
   b. Applying the Denavit-Hartenberg (D-H) model to the physical arm links and joints to derive the forward kinematic equations.
   c. Finding the Inverse kinematics solutions for this educational manipulator and suggesting a method for decreasing multiple solutions in IK.
3) Derivation of the Velocity Kinematics (Jacobian) of the Manipulator considering singularity.
4) Applying a Path and Trajectory planning algorithm.

Phase 2:

1) Development of an electronic interfacing circuit between the AL5B robot arm and the developed GUI program.
2) Holding a comparison between the physical arm and the simulated one.

1.6 Thesis Contribution

The contribution of this thesis concentrates on developing two components related to the AL5B. The first component is concerned with a simulation toolbox, while the second component focuses on interfacing the physical AL5B with the PC.

In the first component, a 3D model is developed to emulate the AL5B motion, which is manly based on a developed analytical kinematics.

The second component develops an interface between the AL5B and the PC using serial communication. A new type of microcontrollers called CUBLOC is used to interface the AL5B with PC by designing an educational interfacing card for this purpose.

The thesis also compares the results of the real-time system with the simulation model.

1.7 Thesis Outline

This thesis structured in the following way: chapter 2 provides theoretical background, which describes the different types of robot arm and shows the workspace for each of them. Some definition such as kinematic modeling, simulation and programming techniques are presented through this chapter. Chapter 3 discusses the Kinematics analysis: the DH parameters, forward, inverse kinematic and it shows the modeling of the robot arm under study. Chapter 4 illustrates the Jacobian equation of the robot arm in section 1. Section 2 presents the singularity problem and the last two sections present the inverse velocity equation and the relation between the torque and the force. Chapter 5 discusses the Trajectory Planning problem and it illustrates the different types of trajectories used in this thesis like Cubic and Quantic polynomials trajectories. Chapter 6 shows the hardware and software implementation of the AL5B robot arm. Explain the main function of the software and flowchart. Also in this chapter, we explain how we can make the interface between robot and computer. Chapter 7 shows the results of testing the developed system are presented and discussed. Finally, a general conclusion is provided as well as recommendations and perspectives for future work are presented in chapter 8.

CHAPTER 2 THEORETICAL BACKGROUND

2.1 Common Kinematic Arrangements of Manipulators

Robotics is a relatively young field of modern technology that crosses traditional engineering boundaries. Understanding the complexity of robots and their applications requires knowledge of electrical engineering, mechanical engineering, systems and industrial engineering, computer science, economics, and mathematics. New disciplines of engineering, such as manufacturing engineering, applications engineering, and knowledge engineering have emerged to deal with the complexity of the field of robotics and factory automation [SPO 05].

This thesis is concerned with the fundamentals of robotics, including kinematics, motion planning, velocity kinematic, computer interfacing, and control. This chapter introduces the most important concepts in these subjects as applied to industrial robot manipulators. The majority of robot applications deal with industrial robot arms operating in structured factory environments so that a first introduction to the subject of robotics must include a rigorous treatment of the topics in this thesis.

The word robot was introduced in 1920 be a Czech playwright which mean work. Basically, a robot is an autonomous device that use computer such as teleoperators, underwater vehicles, autonomous land rovers, etc [SPO 05].

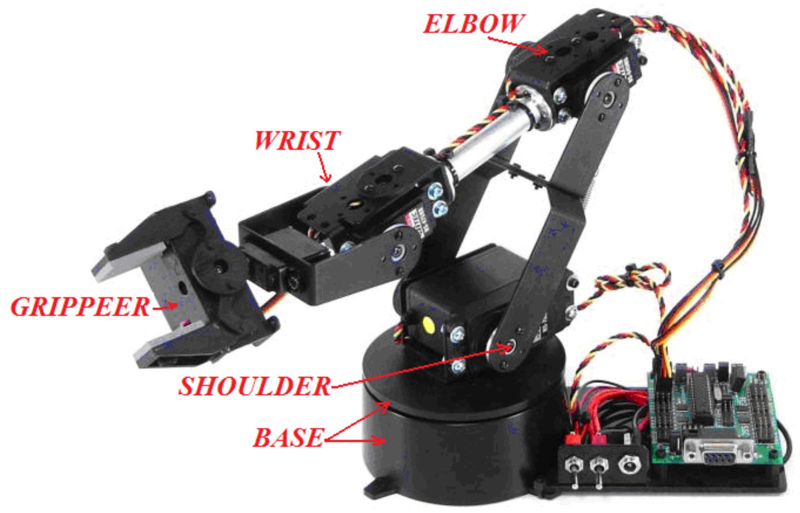


Figure (2.1): AL5B Robotic Arm

Figure (2.1) shows a typical robot that is essentially a mechanical arm operating under computer control. Such devices, though far from the robots of science fiction, are nevertheless extremely complex electro-mechanical systems whose analytical description requires advanced methods, presenting many challenging and interesting research problems.

A robot manipulator is seen as more than just a series of mechanical linkages. Arm mechanism is only one element in a comprehensive automated system, is shown in Figure (2.2) which consists of an arm, external power source, end-of-arm tooling, external and internal sensors, computer interface, and control computer. Even the programmed software is considered as an integral part of the overall system, since the manner in which the robot is programmed and controlled can have a major impact on its performance and subsequent range of applications.

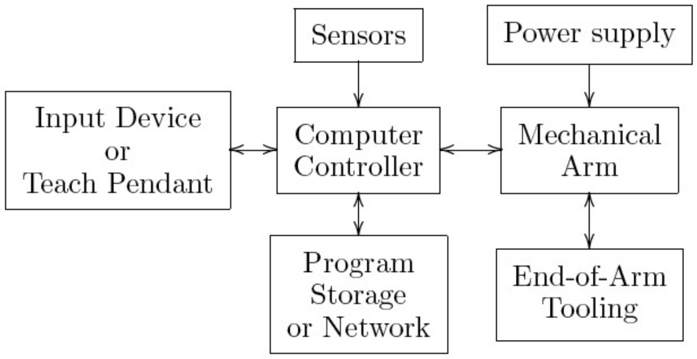


Figure (2.2): Components of Robotic System.

Although there are many possible ways that use prismatic and revolute joints to construct kinematic chains, in practice only a few of these are commonly used. Here we briefly describe several arrangements that are most typical.

2.1.1. Articulated Manipulator (RRR)

Figure (2.3) shows the (ABB IRB1400) articulated manipulator which called a revolute manipulator [SPO 05]. The (RRR) means the type of joint is (Revolute – Revolute – Revolute) and (P) means the type of joint is (Prismatic)

Figure (2.3): The ABB IRB1400 Robot.

A common revolute joint design is the parallelogram linkage such as the motorman SK16, shown in Figure (2.4) in both of these arrangements joint axis z_2 is parallel to z_1 and both z_1 and z_2 are perpendicular to z_0. This kind of manipulator is known as an elbow manipulator. The structure and terminology associated with the elbow manipulator are shown in Figure (2.5) and its workspace is shown in Figure (2.6)

The revolute manipulator provides relatively large freedom of movement in a compact space; the elbow manipulator has several advantages that make it an attractive and popular design. The parallelogram linkage manipulator is that the actuator for joint 3 is located on link 1. Since the weight of the motor is born by link 1, links 2 and 3 can be made more lightweight and the motors themselves can be less powerful.

Figure (2.4): The Motoman SK16 Manipulator.

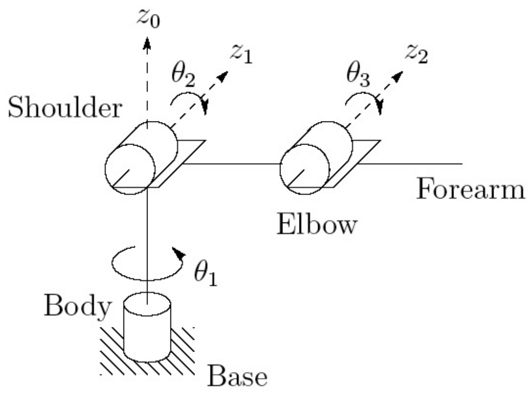


Figure (2.5): Structure of The Elbow Manipulator.

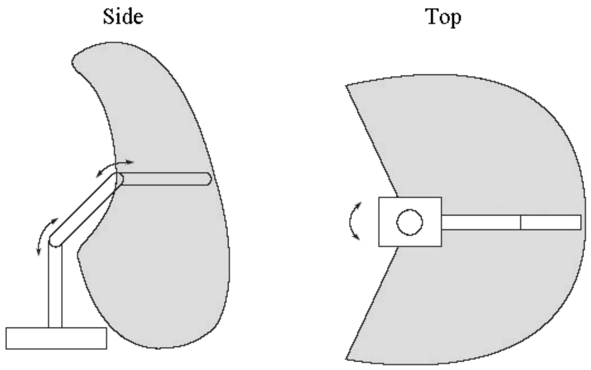


Figure (2.6): Workspace of the Elbow Manipulator.

2.1.2. Spherical Manipulator (RRP)

The spherical manipulator can be obtained by replacing the third or elbow joint in the revolute manipulator by a prismatic joint, as shown in Figure (2.7). The term spherical manipulator derives from the fact that the spherical coordinates defining the position of the end-effector with respect to a frame whose origin lies at the intersection of the three z-axes are the same as the first three joint variables. Figure (2.8) shows the Stanford arm, [SPO 05], one of the most well known spherical robots. The workspace of a spherical manipulator is shown in Figure (2.9).

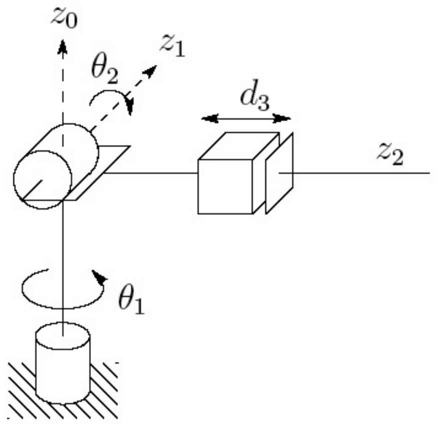


Figure (2.7): The Spherical Manipulator.

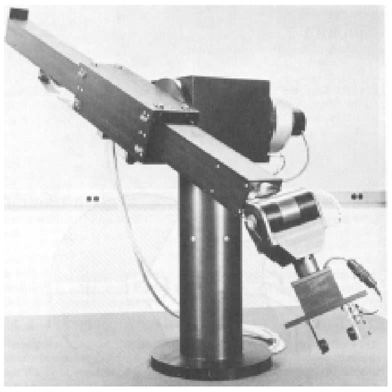


Figure (2.8): The Stanford Arm.

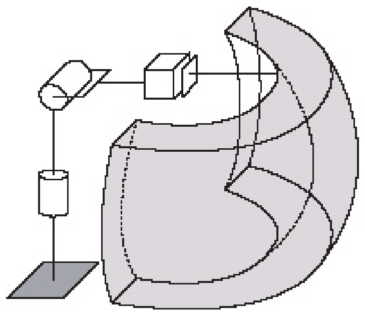


Figure (2.9): Workspace of the Spherical Manipulator.

2.1.3. SCARA Manipulator (RRP)

The SCARA arm (for Selective Compliant Articulated Robot for Assembly) shown in Figure (2.10) is a popular manipulator [SPO 05]. The SCARA has an RRP structure; it is quite different from the spherical manipulator in both appearance and in its range of applications. The SCARA has z_0, z_1, and z_2 mutually parallel. Figure (2.11) shows the Epson E2L653S manipulator [SPO 05]. The SCARA manipulator workspace is shown in Figure (2.12)

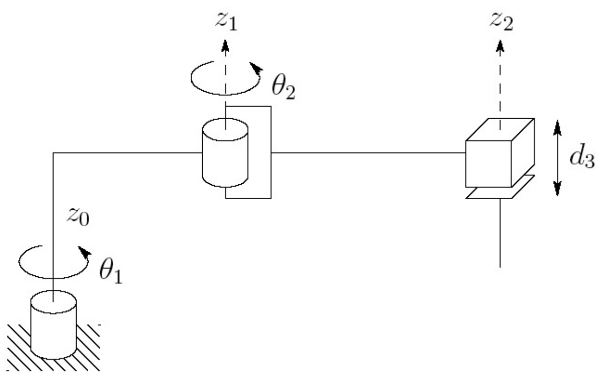


Figure (2.10): The SCARA (Selective Compliant Articulated Robot for Assembly).

Figure (2.11): The Epson E2L653S SCARA Robot.

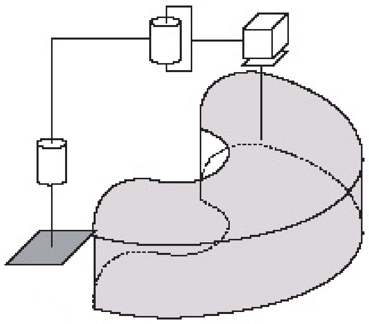


Figure (2.12): Workspace of the SCARA Manipulator.

2.1.4. Cylindrical Manipulator (RPP)

The cylindrical manipulator is shown in Figure (2.13).The first joint is revolute and produces a rotation about the base, while the second and third joints are prismatic. As the name suggests, the joint variables are the cylindrical coordinates of the end-effector with respect to the base. A cylindrical robot, the Seiko RT3300 [SPO 05], is shown in Figure (2.14), with its workspace shown in Figure (2.15).

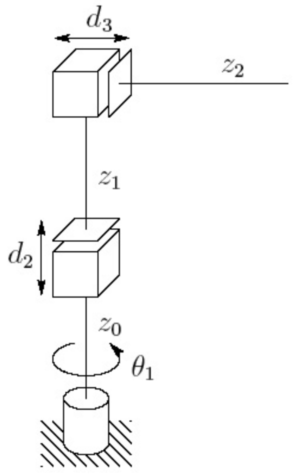


Figure (2.13): The Cylindrical Manipulator.

Figure (2.14): The Seiko RT3300 Robot.

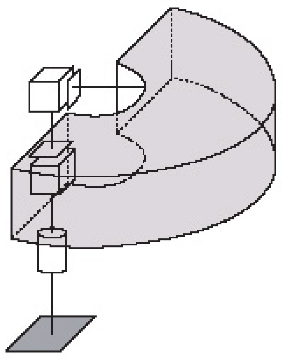


Figure (2.15): Workspace of the Cylindrical Manipulator.

2.1.5. Cartesian Manipulator (PPP)

A manipulator whose first three joints are prismatic is known as a Cartesian manipulator, shown in Figure (2.16). For the Cartesian manipulator, the joint variables are the Cartesian coordinates of the end-effector with respect to the base. An example of a Cartesian robot, from Epson-Seiko, [SPO 05] is shown in Figure (2.17). The workspace of a Cartesian manipulator is shown in Figure (2.18).

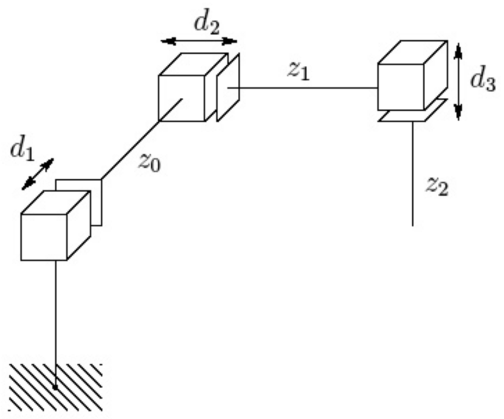


Figure (2.16): The Cartesian Manipulator.

Figure (2.17): The Epson Cartesian Robot.

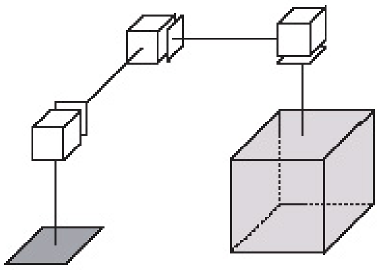


Figure (2.18): Workspace of the Cartesian Manipulator.

2.1.6. Parallel Manipulator

A parallel manipulator has two or more independent kinematic chains connecting the base to the end-effector. Figure (2.19) shows the ABB IRB 940 Tricept robot [SPO 05]. The kinematic description of parallel robots is fundamentally different from that of serial link robots; therefore, requires different methods of analysis.

Figure (2.19): The ABB IRB940 Tricept Parallel Robot.

2.2 Kinematic Modeling

In robot simulation, system analysis needs to be done, such as the kinematics analysis, its purpose is to carry through the study of the movements of each part of the robot mechanism and its relations between itself. The kinematics analysis is divided into forward and inverse analysis. The forward kinematics consists of finding the position of the end-effector in the space knowing the movements of its joints as $F(\theta_1, \theta_2, \ldots, \theta_n) = [x, y, z, R]$, and the inverse kinematics consists of the determination of the joint variables corresponding to a given end-effector position and orientation as $F(x, y, z, R) = \theta_1, \theta_2, \ldots, \theta_n$. Figure (2.20) below shows a simplified block diagram of kinematic modeling.

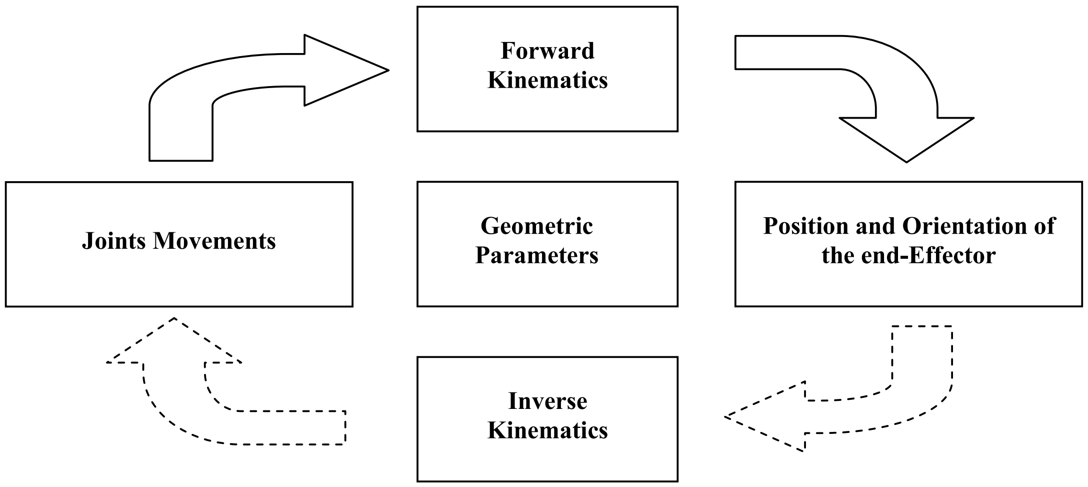


Figure (2.20): Kinematics Block Diagram

A commonly used convention for selecting frames of reference in robotic applications is the Denavit-Hartenberg or D-H convention as shown in Figure (2.21). In this convention each homogenous transformation T_i is represented as a product of "four" basic transformations

$$T_i = Rot(z,\theta_i)Trans(z,d_i)Trans(x,a_i)Rot(x,\alpha_i) \tag{2.1}$$

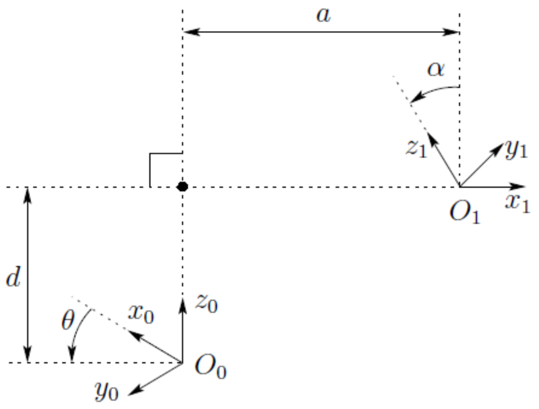


Figure (2.21): DH Frame Assignment

Where the notation $Rot(x,\alpha_i)$ stands for rotation about x_i axis by α_i, $Trans(x,a_i)$ is translation along x_i axis by a distance a_i, $Rot(z,\theta_i)$ stands for rotation about z_i axis by θ_i, and $Trans(z,d_i)$ is the translation along z_i axis by a distance d_i.

$$T_i = \begin{bmatrix} c_{\theta i} & -s_{\theta i} & 0 & 0 \\ s_{\theta i} & c_{\theta i} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c_{\alpha i} & -s_{\alpha i} & 0 \\ 0 & s_{\alpha i} & c_{\alpha i} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$
$$= \begin{bmatrix} c_{\theta i} & -s_{\theta i}c_{\alpha i} & s_{\theta i}s_{\alpha i} & a_i c_{\theta i} \\ s_{\theta i} & c_{\theta i}c_{\alpha i} & -c_{\theta i}s_{\alpha i} & a_i s_{\alpha i} \\ 0 & s_{\alpha i} & c_{\alpha i} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.2}$$

Where the four quantities θ_i, a_i, d_i, α_i are the parameters of link i and joint i. The Figure below illustrates the link frames attached so that frame {i} attached rigidly to link i.

The various parameters in previous equation are given the following names:

a_i (Length) is the distance from z_i *to* z_{i+1}, measured along z_i;

α_i (Twist), is the angle between z_i *and* z_{i+1}, measured about x_i ;

d_i (Offset), is the distance from x_i to x_{i+1} measured along z_i; and

θ_i (Angle), is the angle between x_i and x_{i+1} measured about z_i;

In the usual case of a revolute joint, is called the joint variable, the other three quantities are the fixed link parameters.

Another expression can be used where a homogeneous transformation matrix H represents a rotation by angle α about the current x-axis followed by a translation of a units along the current x-axis, followed by a translation of d units along the current z-axis, followed by a rotation by angle θ about the current z-axis, is given by H where H is given by:

$$H = Rot_{x,\alpha} Trans_{x,a} Trans_{z,d} Rot_{z,\theta} \tag{2.3}$$

$$= \begin{bmatrix} c_\theta & -s_\theta & 0 & a \\ c_\alpha s_\theta & c_\theta c_\alpha & -s_\alpha & -ds_\alpha \\ s_\alpha s_\theta & s_\alpha c_\theta & c_\alpha & dc_\alpha \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.4}$$

The homogeneous representation given in previous equation is a special case of homogeneous coordinates, which have been extensively used in the field of computer graphics. There, one is interested in scaling and/or perspective transformations in addition to translation and rotation. The most general homogeneous transformation takes the form

$$H = \begin{bmatrix} R_{3x3} & d_{3x1} \\ f_{1x3} & s_{1x1} \end{bmatrix} = \begin{bmatrix} Rotation & Translation \\ Perspective & Scale\ factor \end{bmatrix}$$

$$= \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.5}$$

Where the 3 by 3-augmented matrix, R_{3x3}, represents the rotation, the 3 by 1 augmented matrixes, d_{3x1}, represents the translation; the f_{1x3} represents the perspective transformation and S_{1x1} is the factor of universal scale.

The direct kinematics made from the composition of homogeneous transformation matrices, where each translation (prismatic joint) or rotations (rotation joint) correspond to one 4 by 4-augmented matrix:

$$T_i^j = T_{j+1}^j .. T_i^{i-1} \tag{2.6}$$

2.3 Simulation and Modeling Tools

Robot Simulation Software (RSS) and on-off line programming seem likely to be an important issue in robotics research because it is essential for evaluating and predicting the behavior of a robot and have increasingly important role in the evolution of manufacturing automation. Much attention has been devoted to investigate and to develop the on-off line programming of industrial robots. The programming trends and challenges in the development of the RSS can be divided into two components, the graphical user interface (GUI) and the control software. Started with the use of structure programming language, followed with the use of third party package, object programming language, web-programming tools, and artificial intelligence programming language, challenge has been a concern among software developers in order to produce better RSS that cover these two components. There are many ways for designing a graphical user interface, for drawing 3D models and developing real time software simulators for robotics manipulators. In addition, many excellent tools can be used for programming these simulators such as:

1. MATLAB Virtual Realty toolbox with Simulink and V-Realm Builder.

2. The AutoCAD 3D program is used to design the robot graphically; CAD2MATLAB function can be used to convert the resulting graph to an acceptable format by MATLAB, function takes a CAD file in (.stl or .slp) format and converts it to MATLAB.
3. Using OpenGL graphics library under visual C++ or MATLAB.
4. Other well-known tools on the web like UltraArc, CimStation, RobotScrips, ROPSIM, RobotStudio and Dymola.

The increasing interest in 3D graphics has gone hand in hand with the development of a new generation of 3D graphics file formats. Although 3D package could be expected to support tens, if not hundreds of file formats, supporting every format is impossible. Data exchanging between software packages is difficult or impossible. The best format to use for interchanging data often depends on the type of 3D application being used, for example, in order to move data between 3D CAD programs such as AutoCAD, ProE or I-DEAS there are several graphics file formats available, for example, the Autodesk DXF file format, IGES file format and ACIS *SAT* file format. 3D Modelers' and animators must also consider file formats, for example, a common 'in-between' file format from 3D Studio Max to Maya is the DXF for moving geometry between the two packages.

In this research, the file type DWG or DXF will be exported from 3D AutoCAD. Then by using the PolyTrans program, it can be converted to SLP file, finally we can deal easily with this format using MATLAB function "CAD2MAT".

Visual programming is a rather wide concept. In this case however, state of the art visual programming systems are only interesting if they are applicable to robot programming. This approach turned out to present two types of topics, general-purpose visual programming software and visual programming tools concentrated towards the robot process industry. The application used in this thesis presented by visual programming tools. They are intended for various industrially related tasks, such as the robot industry, but need not be used specifically for programming robots. Some visual programming tools will be introduced in the following subsections.

2.3.1. UltraArc

UltraArc is a simulation and offline programming solution, with calibration tools that let users adjust the simulation model to accurately reflect real world device relationships. The interface lets programmers easily modify robot devices to achieve very accurate robot motion results [ULT].

UltraArc holds a library of arc welding robots and weld guns, including the latest robots from ABB, Fanuc and Motoman [ULT]. It also includes a built-in CAD package to create custom work cell components and supports direct import of CAD files via IGES, DXF and direct translations. Robot programs can then be automatically generated from information contained in weld details. There is also support for robot controller-specific weld process information (seam tracking, seam searching, speeds, currents, voltages, etc).

2.3.2. RobotStudio

RobotStudio is a software tool for simulation and offline programming for robots. It is built on the ABB Virtual Controller, an exact copy of the real software that runs the robots in production; hence provides very realistic simulations, using real robot programs and configuration files [ROB].

2.3.3. CimStation Robotics

CimStation Robotics program is similar to RobotStudio. The advantage of CimStation is that supports many different robot suppliers and their products [CIM].

2.3.4. ROPSIM

ROPSIM is a PC based model driven robot simulation system with 3D visualization. The simulation is performed virtually and allows production simulation on screen. It is a robot programming system for use in design, layout, production and maintenance of work cells in integrated production systems [ROP].

2.3.5. RobotScript

RobotScrips it produces code textually, but because it operates in a Windows environment, the end-user has the advantage of using any third-party software to enhance the operation of the robot cell. It also provides an intuitive, graphical user interface to reduce operator training and minimize errors. It can easily be customized using the Software Development Kit to provide a standard, enterprise-wide operator interface [ROB].

2.3.6. Dymola

Dymola is a general purpose modeling program and language, appropriate for building all sorts of mechanical and electrical systems. It has an object-oriented approach, enabling several of the powerful characteristics of such languages, e.g. hierarchical structures, model classes and even inheritance [DYM].

Dymola is built on using equations for describing modeling details. Then the equations are automatically solved and interpreted to symbolical representations. These models and symbols can then be generated on different formats. Its supports C and FORTRAN and is available for UNIX and Windows platforms [DYM].

2.3.7. V-Realm Builder

Virtual Reality (VR) is a system that allows one or more users to move and react in a computer generated environment. The basic VR systems allow the user to gather visual or sound information using computer screens, stereoscopic displays or headphones. V-Realm Builder, which came with MATLAB Virtual Reality Toolbox, was used to make the modifications. A VR sink was then used in the model to interface with the workstation [MAT].

V-Realm Builder made the design of the workstation much simpler. It has several shapes in the program that can be resized and rotated in order fit the requirements of the desired object. Various patterns and colors are also available. The most helpful element of the software is the definition of parent and child classes. In this way pieces can be combined into a more complex component. All the individual parts then use the larger module as a frame of reference. Consequently, the component can be moved or rotated.

2.3.8. OpenGL

OpenGL is API (Application Program Interface) that does not depend on hardware's and Operation System (OS). It functions with high performance for the display of three dimension figures though it is possible to use it to display two dimension Figures. It is used for real-time generation of 3D-CG images of the game and so on [OPE].

2.3.9. Other File Format Converters

There are few graphics file format converters available on the market now. The most common is PolyTrans from Okino [POL], which, according to the most recent plugging have been seen does not yet, translate animation and does not support the ASE file format. PolyTrans is also very expensive; however, it is widely known and used throughout the industry. There are several websites devoted to plugging for the program and it may be constantly updated for new file types.

Another, more recent file format converter, 3D Exploration is available as shareware yet covers numerous file types including MAX, ASE and OBJ file format, however, it only supports information on objects, materials, cameras and light sources, all other information is skipped. Similarly, most other file converters support neither the file types used, often do not support any Maya format, other than OBJ, format and do not deal with animation at all.

PolyTrans, for example, contains the NuGraf rendering system and can be used to perform various actions on polygonal objects, etc. and render them entirely within the package. 3D Exploration contains a simple OpenGL interface to view the workspace; however, it contains virtually no tools to modify the scene.

2.4 Basic Categories of Programming Languages

Virtually all robots are programmed with some kind of robot programming language. These programming languages are used to command the robot to move to certain locations, to output signal, and to read inputs. The programming language is what gives robots flexibility. When learning any programming language, like a robot language or a computer language, one of the most difficult tasks is learning what the commands are and how to use them.

To get an overview of different types of robot programming languages [MIK 02], it is appropriate to put them in three basic categories:

1. **Specialized robot languages.** These languages have been developed specifically for robots. The commands found in these languages are mostly motion commands with minimal logic statements available. Most of the early robot languages were of this type, although many still exist today. VAL1 is an example of such a robot language [MIK 02].
2. **Robot library for a new general-purpose language.** This is based on creating new general- purpose language, then adding specific robot commands. They are generally more capable than a specialized language, since they tend to have better logic testing capabilities. KAREL is a good example of robot programming language from Fanuc Robotics [MIK 02].
3. **Robot library for an existing computer language.** These languages are developed by creating extensions to already existing popular computer programming languages. Consequently, the robot languages resemble traditional computer programming languages, providing the same power as these widely used languages. RobotScript is an example of this type of language [MIK 02].

Today, industrial robots are programmed in one of two possible ways. In reality, these techniques are often combined, resulting in what is known as hybrid programming. The two main techniques are described shortly below.

2.4.1. Online Programming

Online programming means creating the control program directly on the robot's onboard computer; hence, by manually steering the robot to different positions using a jog or similar control mechanism. Each desired position contributes to the code as a number of coordinates. An advantage with online programming is exactness and few later corrections due to programming the actual robot in its actual real-world environment. The main drawbacks of this method are that it is time consuming and it has long production stops.

2.4.2. Offline Programming

In contrast to online programming, offline programming means creating the control program on a detached unit, such as a PC. This involves either manual editing of code in a text editor, or automatically generated code using a modeling environment. Once the program is ready for deployment, it is moved to the robot's computer for manual correction and tuning. An advantage with this method is that robots can be programmed before installation and stay in production while being reprogrammed; meaning production breaks usually are significantly shortened. On the other hand, manual correction sometimes gets very extensive, and a programmer is also required to write the code offline.

The differences between the on-line and off-line programming and the practical characteristics of off-line programming are shown in Table 2.1.

Table (2.1): Differences between the on-line and off-line programming

ON-LINE	OFF-LINE	ON-LINE PROGRAMMING ADVANTAGES	OFF-LINE PROGRAMMING DISADVANTAGES
Sequential operation mode	Parallel working mode	Increases robot's efficiency	High initial costs
Operational robots requested	No physical robot and workcell's components	Provides a safe environment for simulation	Fast information exchanges between engineering departments
Attention with errors	Early examinations and optimizations.	Integrated CAD-CAM systems	Reorganization
Requires staff for supervising	Quality information regarding the process	Simplification of complex tasks	Necessity of robot's calibration in real working environment
Extra time for workcell's physic arrangement	Compound vision of the simulation.	Verification of programs before loading it into robot controller Fast and easy optimization	Low precision
	Saving costs	Analysis provided by simulation software	Software errors and programming bugs

CHAPTER 3 KINEMATICS

3.1 Introduction

Kinematics is the description of motion without regard to the forces that cause it. It deals with the study of position, velocity, acceleration, and higher derivatives of the position variables.

The kinematics solutions of any robot manipulator are divided into tow solution, the first one is the solution of Forward kinematics, and the second one is the inverse kinematics solution. Forward kinematics will determine where the robot's manipulator hand will be if all joints are known. Where the inverse kinematics will calculate what each joint variable must be if the desired position and orientation of end-effector is determined. Hence, Forward kinematics is defined as transformation from joint space to Cartesian space where as Inverse kinematics is defined as transformation from Cartesian space to joint space.

3.2 Direct/Forward Kinematics

The forward kinematics problem can be stated as follows: Given the joint variables of the robot, determine the position and orientation of the end-effector. Since each joint has a single degree of freedom, the action of each joint can be described by a single number, i.e. $\theta_1,\theta_2\dots,\theta_n$, the angle of rotation in the case of a revolute joint. The objective of forward kinematic analysis is to determine the cumulative effect of the joint variables.

Suppose a robot has n_{+l} links numbered from zero to n starting from the base of the robot, which is taken as link 0. The joints are numbered from one to n, and z_i is a unit vector along the axis in space about which the links i-1 and i are connected. The i-th joint variable is denoted by q, In the case of a revolute joint, q, is the angle of rotation, while in the case of a prismatic joint q, is the joint translation. Next, a coordinate frame is attached rigidly to each link. To be specific, we choose frames 1 through n such that the frame i is rigidly attached to link i. Figure (3.1) illustrates the idea of attaching frames rigidly to links in the case of an AL5B robot.

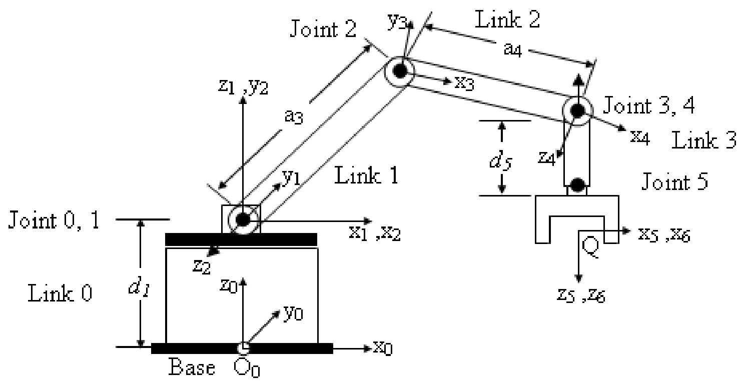


Figure (3.1): AL5B Robot Arm Frame Assignment

T_i^{i-1} is a homogenous matrix which is defined to transform the coordinates of a point from frame i to frame i_{-1}. The matrix T_i^{i-1} is not constant, but varies as the configuration of the robot is changed. However, the assumption that all joints are either revolute or prismatic means that T_i^{i-1} is a function of only a single joint variable, namely q_i. In other words,

$$T_i^{i-1} = T_i^{i-1}(q_i) \tag{3.1}$$

The homogenous matrix that transforms the coordinates of a point from frame i to frame j is denoted by T_i^j (i > j). Denoting the position and orientation of the end-effector with respect to the inertial or the base frame by a three dimensional vector d_n^0 and a 3x3 rotation matrix R_n^0, respectively, we define the homogenous matrix

$$T_n^0 = \begin{bmatrix} R_n^0 & d_n^0 \\ 0 & 1 \end{bmatrix} \tag{3.2}$$

Then the position and orientation of the end-effector in the inertial frame are given by

$$T_n^0(q_1, q_2, ..., q_n) = T_1^0(q_1)T_2^1(q_1)...T_n^{n-1}(q_n) \tag{3.3}$$

Each homogenous transformation T_i^{i-1} is of the form

$$T_i^{i-1} = \begin{bmatrix} R_i^{i-1} & d_i^{i-1} \\ 0 & 1 \end{bmatrix} \tag{3.4}$$

Hence

$$T_i^j = T_{j+1}^j..T_i^{i-1} = \begin{bmatrix} R_i^j & d_i^j \\ 0 & 1 \end{bmatrix} \tag{3.5}$$

The matrix R_i^j expresses the orientation of frame i relative to frame j (i > j) and is given by the rotational parts of the T_i^j -matrices (i > j) as

$$R_i^j = R_{j+1}^j ... R_i^{i-1} \tag{3.6}$$

The vectors d_i^j (i > j) are given recursively by the formula

$$d_i^j = d_{j+1}^j + R_{i-1}^j d_i^{i-1} \tag{3.7}$$

3.2.1. Assigning the Coordinate Frames

AL5B has five rotational joints and a moving grip as shown in Figure (3.1). Joint 1 represents the shoulder and its axis of motion is z_1. This joint provides a rotational θ_1 angular motion around z_1 axis in x_1y_1 plane. Joint 2 is identified as the Upper Arm and its axis is perpendicular to Joint 1 axis. It provides a rotational θ_2 angular motion around z_2 axis in x_2y_2 plane. z_3 axes of Joint 3 (Forearm) and Joint 4 (Wrist) are parallel to Joint 2 z-axis; they provide θ_3 and θ_4 angular motions in x_3y_3 and x_4y_4 planes respectively. Joint five are identified as the grip rotation. Its z_5 axis is vertical to z_4 axis and it provides θ_5 angular motions in x_5y_5 plane [MOH 09]. A graphical view of all the joints was displayed in Figure. (3.2).

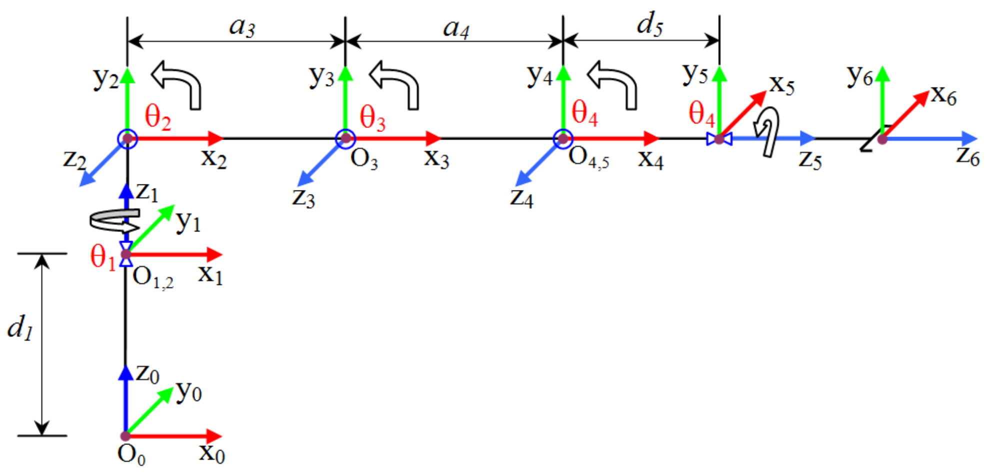


Figure (3.2): Coordinate Frames of AL5B Robotic Arm

A rigid body is completely described in space by its position to a reference frame (translation) and its orientation.

3.2.2. AL5B DH Parameters

As explained in chapter 2 many methods can be used in the direct kinematics calculation. The Denavit-Hartenberg analysis is one of the most used, in this method the direct kinematics is determined from some parameters that have to be defined, depending on each mechanism. However, it was chosen to use the homogeneous transformation matrix. In this, analysis, once it is easily defined one coordinate transformation between two frames, where the position and orientation are fixed one with respect to the other it is possible to work with elementary homogeneous transformation operations. D-H parameters for AL5B defined for the assigned frames in Table 3.1.

Table (3.1): DH Parameter for AL5B Robot Arm

i	α_{i-1}	a_{i-1}	d_i	θ_i
1	0	0	d_1	θ_1^*
2	90	0	0	θ_2^*
3	0	a_3	0	θ_3^*
4	0	a_4	0	$(\theta_4\text{-}90)^*$
5	-90	0	d_5	θ_5^*
6	0	0	0	Gripper

By substituting the parameters from Table (3.1) into equation (2.4), the transformation matrices T_1 to T_6 can be obtained as shown below. For example, T_1 shows the transformation between frames 0 and 1 (designating C_i as $\cos\theta_i$ and S_i as $\sin\theta_i$ etc).

$$T_1^0 = \begin{bmatrix} c_{\theta_1} & -s_{\theta_1} & 0 & 0 \\ s_{\theta_1} & c_{\theta_1} & 0 & 0 \\ 0 & 1 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.8}$$

$$T_2^1 = \begin{bmatrix} c_{\theta_2} & -s_{\theta_2} & 0 & 0 \\ 0 & 0 & -1 & 0 \\ s_{\theta_2} & c_{\theta_2} & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.9}$$

$$T_3^2 = \begin{bmatrix} c_{\theta_3} & -s_{\theta_3} & 0 & a_3 \\ s_{\theta_3} & c_{\theta_3} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.10}$$

$$T_4^3 = \begin{bmatrix} c_{\theta_4} & -s_{\theta_4} & 0 & a_4 \\ s_{\theta_4} & c_{\theta_4} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.11}$$

$$T_5^4 = \begin{bmatrix} c_{\theta_5} & -s_{\theta_5} & 0 & 0 \\ 0 & 0 & 1 & d_5 \\ s_{\theta_5} & c_{\theta_5} & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.12}$$

$$T_{Gripper} = \begin{bmatrix} c_{\theta_5} & -s_{\theta_5} & 0 & 0 \\ s_{\theta_5} & c_{\theta_5} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.13}$$

Using the above values of the transformation matrices; the link transformations can be concatenated (multiplied together) to find the single transformation that relates frame (5) to frame (0):

$$T_5^0 = T_1^0 T_2^1 T_3^2 T_4^3 T_5^4 = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.14}$$

The transformation given by equation (3.14) is a function of all 5 joint variables. From the robots joint position, the Cartesian position and orientation of the last link may be computed using above equation (3.14).

The first three columns in the matrices represent the orientation of the end effectors, whereas the last column represents the position of the end effectors [MOH 09]. The orientation and position of the end effectors can be calculated in terms of joint angles using:

$$\begin{aligned} n_x &= ((c_1c_2c_3 - c_1s_2s_3)c_4 + (-c_1c_2s_3 - c_1s_2c_3)s_4)c_5 + s_1s_5 \\ n_y &= ((s_1c_2c_3 - s_1s_2s_3)c_4 + (-s_1c_2s_3 - s_1s_2c_3)s_4)c_5 - c_1s_5 \\ n_z &= ((s_2c_3 + c_2s_3)c_4 + (-s_2s_3 + c_2c_3)s_4)c_5 \end{aligned} \tag{3.15}$$

$$\begin{aligned} o_x &= -((c_1c_2c_3 - c_1s_2s_3)c_4 + (-c_1c_2s_3 - c_1s_2c_3)s_4)s_5 + s_1c_5 \\ o_y &= -((s_1c_2c_3 - s_1s_2s_3)c_4 + (-s_1c_2s_3 - s_1s_2c_3)s_4)s_5 - c_1c_5 \\ o_z &= (c_2c_3 - s_2s_3)s_4)s_5 - ((s_2c_3 + c_2s_3)c_4 \end{aligned} \tag{3.16}$$

$$\begin{aligned} a_x &= -(c_1c_2c_3 - c_1s_2s_3)s_4 + (-c_1c_2s_3 - c_1s_2c_3)c_4 \\ a_y &= -(s_1c_2c_3 - s_1s_2s_3)s_4 + (-s_1c_2s_3 - s_1s_2c_3)c_4 \\ a_z &= (c_2c_3 - s_2s_3)c_4 - (s_2c_3 + c_2s_3)s_4 \end{aligned} \tag{3.17}$$

$$\begin{aligned} d_x &= (-(c_1c_2c_3 - c_1s_2s_3)s_4 + (-c_1c_2s_3 - c_1s_2c_3)c_4)d_5 + (c_1c_2c_3 - c_1s_2s_3)a_4 + c_1c_2a_3 \\ d_y &= (-(s_1c_2c_3 - s_1s_2s_3)s_4 + (-s_1c_2s_3 - s_1s_2c_3)c_4)d_5 + (s_1c_2c_3 - s_1s_2s_3)a_4 + s_1c_2a_3 \\ d_z &= (-(s_2c_3 + c_2s_3)s_4 + (-s_2s_3 + c_2c_3)c_4)d_5 + (s_2c_3 + c_2s_3)a_4 + s_2a_3 + d_1 \end{aligned} \tag{3.18}$$

3.3 Inverse kinematics

Inverse Kinematics (IK) analysis determines the joint angles for desired position and orientation in Cartesian space. Total transformation matrix in equation. (3.14) will be used to calculate inverse kinematics equations. IK is more difficult problem than forward kinematics.

The solution of inverse kinematic is more complex than direct kinematics and there is not any global analytical solution method. Each manipulator needs a particular method considering the system structure and restrictions. There are two solutions approaches namely, geometric and algebraic used for deriving the inverse kinematics solution. Let's start with geometric approach.

3.3.1. Geometric Approach

Using IK-Cartesian mode, the user specifies the desired target position of the gripper in Cartesian space as (x, y, z) where z is the height, and the angle of the gripper relative to ground, ψ (see Figure 3.4), is held constant. This constant ψ allows users to move objects without changing the object's orientation (the holding a cup of liquid scenario). In addition, by either keeping ψ fixed in position mode or keeping the wrist fixed relative to the rest of the arm, the inverse kinematic equations can be solved in closed form as we now show for the case of a fixed ψ [MOH 09].

The lengths d_1, a_3, a_4 and d_5 correspond to the base height, upper arm length, forearm length and gripper length, respectively are constant. The angles θ_1, θ_2, θ_3, θ_4 and θ_5 correspond to shoulder rotation, upper arm, forearm, wrist, and end effector, respectively. These angles are updated as the specified position in space changes. We solve for the joint angles of the arm, $\theta_{1:4}$ given desired position (x, y, and z) and ψ which are inserted by the user.

From Figure (3.3), we clearly see that $\theta_1 = \text{atan2}(y, x)$ and the specified radial distance from the base d are related to x and y by:

$$d = \sqrt{x_d^{\,2} + y_d^{\,2}}$$
$$x_d = d\cos(\theta_1) \qquad (3.19)$$
$$y_d = d\sin(\theta_1)$$

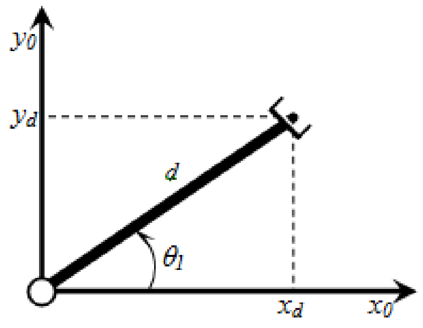


Figure (3.3): Top View of Robot.

Moving now to the planar view in Figure (3.4), we find a relationship between joint angles θ_2, θ_3 and θ_4 and ψ as follows:

$$\psi = \theta_2 + \theta_3 + \theta_4 \qquad (3.20)$$

Since ψ is given, we can calculate the radial distance and height of the wrist joint:

$$r_4 = r_d - a_5\cos(\psi)$$
$$z_4 = z_d - a_5\sin(\psi)$$
$$or \qquad (3.21)$$
$$r_4 = a_3\cos(\theta_2) + a_4\cos(\theta_2 + \theta_3)$$
$$z_4 = a_3\sin(\theta_2) + a_4\sin(\theta_2 + \theta_3) + d_1$$

Now we want to determine θ_2 and θ_3. We first solve for α, β and s (from Figure 3.4) uses the law of cosines as:

$$\beta = a\tan 2(s^2 + a_3^{\,2} - a_4^{\,2}, 2a_3 s)$$
$$\alpha = a\tan 2(z_4 - d_1, r_4) \qquad (3.22)$$
$$s = \sqrt{(z_4 - d_1)^2 + r_4^{\,2}}$$

With these intermediate values, we can now find the remaining angle values as:

$$\theta_2 = \alpha \pm \beta$$
$$\theta_3 = a\tan 2(s^2 - a_3^2 - a_4^2, 2a_3 a_4) \qquad (3.23)$$
$$\theta_4 = \psi - \theta_2 - \theta_3$$

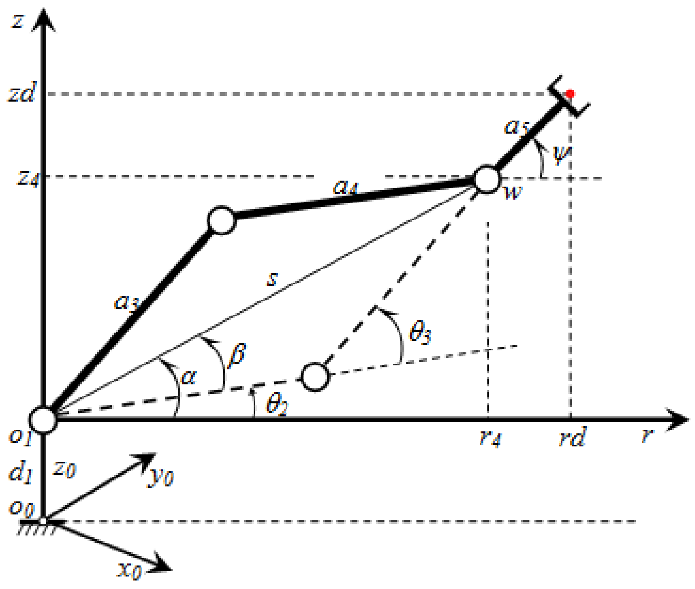


Figure (3.4): Planar View of AL5B Robot Arm.

3.3.2. Analytical (algebraic) Approach

Using the X, Y and Z resultants gotten in the direct kinematics:

$$x = c_1[a_3c_2 + a_4c_{23} + d_5c_{234}] \tag{3.24}$$

$$y = s_1[a_3c_2 + a_4c_{23} + d_5c_{234}] \tag{3.25}$$

$$z = [d_5c_{234} + a_4s_{23} + a_3s_2] + d_1 \tag{3.26}$$

The simplified equation is gotten:

$$(3.24)^2 + (3.25)^2 \Rightarrow a_3c_2 + a_4c_{23} = \pm\sqrt{x^2 + y^2} - d_5c_{234} \tag{3.27}$$

The first joint movement, defined by θ1, can be calculated using geometric parameters only:

$$\therefore \theta_1 = a\tan 2(y, x)$$

Now we can calculate θ3, by using equation (3.26):

$$(3.26) \Rightarrow a_3s_2 + a_4s_{23} = z - d_5s_{234} - d_1 \tag{3.28}$$

$$(3.27)^2 + (3.28)^2 \Rightarrow c3 = \frac{(z - d_5s_{234} - d_1)^2 + (\pm\sqrt{x^2 + y^2} - d_5c_{234})^2 - a_3^2 - a_4^2}{2a_3a_4} \tag{3.29}$$

$$c_\psi = c_{234}, s_\psi = s_{234}$$

$$where$$

$$c_{234} = \cos(\theta_2 + \theta_3 + \theta_4), s_{234} = \sin(\theta_2 + \theta_3 + \theta_4)$$

$$\therefore c_3 = \frac{(z - d_5s_\psi - d_1)^2 + (\pm\sqrt{x^2 + y^2} - d_5c_\psi)^2 - a_3^2 - a_4^2}{2a_3a_4} \tag{3.30}$$

$$s_3 = \pm\sqrt{1 - c_3^2}$$

$$\therefore \theta_3 = a\tan 2(s_3, c_3) \tag{3.31}$$

After calculate θ₃ we can find θ₂ by:

$$\theta_2 = \alpha - \beta$$
$$\alpha = a\tan 2(z - d_5 s_\psi - d_1, \pm\sqrt{x^2+y^2} - d_5 c_\psi) \tag{3.32}$$
$$\beta = a\tan 2(a_4 s_3, a_3 + a_4 c_3)$$

$$\therefore \theta_2 = a\tan 2(z - d_5 s_\psi - d_1, \pm\sqrt{x^2+y^2} - d_5 c_\psi) - a\tan 2(a_4 s_3, a_3 + a_4 c_3) \tag{3.33}$$

$$\because \theta_\psi = \theta_2 + \theta_3 + \theta_4 \tag{3.34}$$

$$\therefore \theta_4 = \theta_\psi - \theta_2 - \theta_3 \tag{3.35}$$

We can find θ₅ by using total transformation matrix in equation (3.14):

$$\begin{aligned} s_5 &= s_1 r_{11} - c_1 r_{21} \\ c_5 &= s_1 r_{12} - c_1 r_{22} \\ r_{11} &= ((c_1c_2c_3 - c_1s_2s_3)c_4 + (-c_1c_2s_3 - c_1s_2c_3)s_4)c_5 + s_1s_5 \\ r_{12} &= -((c_1c_2c_3 - c_1s_2s_3)c_4 + (-c_1c_2s_3 - c_1s_2c_3)s_4)s_5 + s_1c_5 \\ r_{21} &= ((s_1c_2c_3 - s_1s_2s_3)c_4 + (-s_1c_2s_3 - s_1s_2c_3)s_4)c_5 - c_1s_5 \\ r_{22} &= -((s_1c_2c_3 - s_1s_2s_3)c_4 + (-s_1c_2s_3 - s_1s_2c_3)s4)s_5 - c_1c_5 \end{aligned} \tag{3.36}$$

$$\therefore \theta_5 = a\tan 2(s_5, c_5) \tag{3.37}$$

Another algebraic solution by using total transformation matrix in equation (3.14), we can find the inverse kinematics solution for an AL5B manipulator.

$$T = T_0^1 * T_1^2 * T_2^3 * T_3^4 * T_4^5 = G = \begin{bmatrix} r_{11} & r_{12} & r_{13} & x \\ r_{21} & r_{22} & r_{23} & y \\ r_{31} & r_{32} & r_{33} & z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.38}$$

To find the inverse kinematics solution for the first joint θ₁ as a function of the known elements of $T_{base}^{end-effector}$, the link transformation inverses are remultiplied as shown in equation (3.39):

$$\left[T_0^1\right]^{-1} * T_0^5 = \left[T_0^1\right]^{-1} * \left[T_0^1\right] * T_1^2 * T_2^3 * T_3^4 * T_4^5 \tag{3.39}$$

Where $\left[T_0^1\right]^{-1} * \left[T_0^1\right] = I$, I is identity matrix. In this case, the above equation is given by $\left[T_0^1\right]^{-1} * T_0^5 = T_1^2 * T_2^3 * T_3^4 * T_4^5$; the solution of this equation is explained in equation (3.40).

$$\begin{bmatrix} c_1r_{11} + s_1r_{21} & c_1r_{12} + s_1r_{22} & c_1r_{13} + s_1r_{23} & c_1x + s_1y \\ -s_1r_{11} + c_1r_{21} & -s_1r_{12} + c_1r_{22} & -s_1r_{13} + c_1r_{23} & -s_1x + c_1y \\ r_{31} & r_{32} & r_{33} & z - d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \alpha_{11} & \alpha_{12} & \alpha_{13} & \alpha_{14} \\ \alpha_{21} & \alpha_{22} & \alpha_{23} & \alpha_{24} \\ \alpha_{31} & \alpha_{32} & \alpha_{33} & \alpha_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.40}$$

$$\alpha_{11} = ((c_2c_3 - s_2s_3)(c_4 - 1) + (-c_2s_3 - s_2c_3)s_4)c_5$$
$$\alpha_{12} = -((c_2c_3 - s_2s_3)(c_4 - 1) + (-c_2s_3 - s_2c_3)s_4)s_5$$
$$\alpha_{13} = -(c_2c_3 - s_2s_3)s_4 + (-c_2s_3 - s_2c_3)c_4$$
$$\alpha_{14} = (-(c_2c_3 - s_2s_3)s_4 + (-c_2s_3 - s_2c_3)c_4)d_5 + (c_2c_3 - s_2s_3)a_4 + c_2a_3$$
$$\alpha_{21} = -s_5$$
$$\alpha_{22} = -c_5$$
$$\alpha_{23} = 0$$
$$\alpha_{24} = 0$$
$$\alpha_{31} = ((s_2c_3 + c_2s_3)(c_4 - 1) + (c_2c_3 - s_2s_3)s_4)c_5$$
$$\alpha_{32} = -((s_2c_3 + c_2s_3)(c_4 - 1) + (c_2c_3 - s_2s_3)s_4)s_5$$
$$\alpha_{33} = -(s_2c_3 + c_2s_3)s_4 + (c_2c_3 - s_2s_3)c_4$$
$$\alpha_{34} = (-(s_2c_3 + c_2s_3)s_4 + (c_2c_3 - s_2s_3)c_4)d_5 + (s_2c_3 + c_2s_3)a_4 + s_2a_3$$

$$= \begin{bmatrix} ((c_{23}(c_4 - 1) - s_{234})c_5 & -((c_{23})(c_4 - 1) - s_{234})s_5 & -s_{234} & -s_{234}d_5 + c_{23}a_4 + c_2a_3 \\ -s_5 & -c_5 & 0 & 0 \\ (s_{23}(c_4 - 1) + c_{23}s_4)c_5 & -(s_{23}(c_4 - 1) + c_{23}s_4)s_5 & c_{234} & c_{234}d_5 + s_{23}a_4 + s_2a_3 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad (3.41)$$

From equation (3.40), (3.41) we find:

$$c_1y - s_1x = 0$$
$$\therefore \theta_1 = a\tan 2(y, x) \quad (3.42)$$

To find the other variables, the following equations are obtained as a similar manner.

$$\left[T_0^1 * T_1^2\right]^{-1} * T_0^5 = T_2^3 * T_3^4 * T_4^5$$
$$\left[T_0^1 * T_1^2 * T_2^3\right]^{-1} * T_0^5 = T_3^4 * T_4^5$$
$$\left[T_0^1 * T_1^2 * T_2^3 * T_3^4\right]^{-1} * T_0^5 = T_4^5$$

Now, we can compute θ_2 from the above equation:

$$c_1x + s_1y = -s_{234}d_5 + c_{23}a_4 + c_2a_3$$
$$c_1x + s_1y = -s_{234}d_5 + c_2(c_3a_4 + a_3)$$
$$c_2 = \frac{c_1x + s_1y + s_{234}d_5}{(c_3a_4 + a_3)}$$
$$z - d_1 = c_{234}d_5 + s_{23}a_4 + s_2a_3$$
$$z - d_1 = c_{234}d_5 + s_2(s_3a_4 + a_3)$$
$$z - d_1 - c_{234}d_5 = s_2(s_3a_4 + a_3)$$
$$s_2 = \frac{z - d_1 - c_{234}d_5}{(s_3a_4 + a_3)}$$
$$\therefore \theta_2 = a\tan 2(s_2, c_2) \quad (3.43)$$

Similar we can find θ_3:

$$c_1x + s_1y = -s_{234}d_5 + c_{23}a_4 + c_2a_3$$
$$z - d_1 = c_{234}d_5 + s_{23}a_4 + s_2a_3$$
$$[c_1x + s_1y - s_{234}d_5]^2 = [c_{23}a_4 + c_2a_3]^2$$
$$[z - d_1 + c_{234}d_5]^2 = [s_{23}a_4 + s_2a_3]^2$$
$$c_3 = \frac{[c_1x + s_1y - s_{234}d_5]^2 + [z - d_1 + c_{234}d_5]^2 - a_3a_4}{2a_3a_4}$$
$$s_3 = \pm\sqrt{1-c_3^2}$$
$$\therefore \theta_3 = a\tan 2(s_3, c_3) \tag{3.44}$$

$$s_{234} = -(c_1r_{13} + s_1r_{23})$$
$$\mathrm{c}_{234} = r_{33}$$
$$\theta_{234} = a\tan 2(s_{234}, c_{234}) \tag{3.45}$$

In this context, θ234 (WARTG=Wrist Angle Relative to Ground) is user supplied by entering both xyz position coordinates of the selected target position on the workspace and a constant WARTG value. Equation [3.40] and [3.41] elements of the resultant matrix equation give:

$$s_5 = s_1r_{11} - c_1r_{21}$$
$$c_5 = s_1r_{12} - c_1r_{22}$$
$$\therefore \theta_5 = a\tan 2(s_5, c_5) \tag{3.46}$$

Similar multiplication procedures for joints 4 also, yields' θ4:

$$(T_{01} * T_{12} * T_{23})^{-1}G = T_{34} * T_{45} \tag{3.47}$$

$$\begin{bmatrix} * & * & c_{123}r_{13} + c_{23}s_1r_{23} + s_{23}r_{33} & * \\ * & * & -s_{23}c_1r_{13} - s_{123}r_{23} + c_{23}r_{33} & * \\ * & * & * & * \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} (c_4-1)c_5 & -(c_4-1)s_5 & -s_4 & -s_4d_5 + a_4 \\ s_4c_5 & -s_4s_5 & c_4 & c_4d_5 \\ s_5 & c_5 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$s_4 = -(c_{123}r_{13} + c_{23}s_1r_{23} + s_{23}r_{33})$$
$$c_4 = -(s_{23}c_1r_{13} + s_{123}r_{23} - c_{23}r_{33})$$

Which results,

$$\therefore \theta_4 = a\tan 2(s_4, c_4) \tag{3.48}$$

In summary, θ1, θ2, θ3, θ4, θ5 are given by (3.42), (3.43) , (3.44), (3.48), (3.46), the initial values of these angles are given by the user defining the initial configuration of the robot arm.

CHAPTER 4 DIFFERENTIAL KINEMATICS AND STATICS

4.1 Velocity Kinematics/Arm Jacobian

The Jacobian is one of the most important quantities in the analysis and control of robot motion. It is used for smooth trajectory planning and execution in the derivation of the dynamic equation. To investigate target with specified velocity, each joint velocity at the specified joint positions needs to be found. This is accomplished using Jacobian, which is used to relate joint velocities to the linear and angular velocities of the end-effector [SPO 05]. The relationships between joint velocities $\dot{\theta}$, and the linear and angular velocities, $\dot{p}$ and ω respectively, of the end effector:

$$\dot{p} = J_P(\theta)\dot{\theta} \tag{4.1}$$

$$w = J_w(\theta)\dot{\theta} \tag{4.2}$$

$$\text{where} \quad \theta = \begin{bmatrix} \theta_1 \\ \vdots \\ \theta_n \end{bmatrix} \tag{4.3}$$

The above equations are combined to form J, which relates both linear, and angular velocity:

$$v = J(\theta)\dot{\theta} \tag{4.4}$$

Where J(θ) is in the form

$$J = \begin{bmatrix} J_{p1} & & J_{pn} \\ & 6\times n & \\ J_{w1} & & J_{wn} \end{bmatrix} \tag{4.5}$$

The number of columns of the Jacobian represents the number of degrees of freedom or links of the manipulator. There are always three rows for linear velocity in the x, y and z directions, and three for angular velocity. Hence, for a six degree of freedom manipulator, the Jacobian is a 6 by 6 square matrix. The Jacobian can be calculated from the following equation.

$$\begin{aligned} J_{vi} &= \begin{cases} z_{i-1}\times(o_n - o_{i-1}) & \textit{for revolute } \text{joint } i \\ z_{i-1} & \textit{for prismatic } \text{joint } i \end{cases} \\ J_{wi} &= \begin{cases} z_{i-1} & \textit{for revolute } \text{joint } i \\ 0 & \textit{for prismatic } \text{joint } i \end{cases} \end{aligned} \tag{4.6}$$

Where: J_{vi} angular velocity and J_{wi} linear velocity. For the AL5B robot arm the Jacobian matrix is equal 6x5 [MOH 09].

$$J(q) = \begin{bmatrix} z_0\times(o_5-o_0) & z_1\times(o_5-o_1) & z_2\times(o_5-o_2) & z_3\times(o_5-o_3) & z_4\times(o_5-o_4) \\ z_0 & z_1 & z_2 & z_3 & z_4 \end{bmatrix} \tag{4.7}$$

From the forward kinematic we can find:

$$o_0=\begin{bmatrix}0\\0\\0\end{bmatrix},\ o_1=\begin{bmatrix}0\\0\\d1\end{bmatrix},\ o_2=\begin{bmatrix}0\\0\\d1\end{bmatrix},\ o_3=\begin{bmatrix}c_1c_2a_3\\c_1s_1a_3\\s_2a_3+d_1\end{bmatrix}$$
$$o_4=\begin{bmatrix}c_1c_{23}a_4+c_1c_2a_3\\s_1c_{23}a_4+s_1c_2a_3\\s_{23}a_4+s_2a_3+d_1\end{bmatrix},\ o_5=\begin{bmatrix}c_1c_{234}d_5+c_1c_{23}a_4+c_1c_2a_3\\s_1c_{234}d_5+s_1c_{23}a_4+s_1c_2a_3\\s_{234}d_5+s_{23}a_4+s_2a_3+d1\end{bmatrix} \tag{4.8}$$

Moreover, we can find:

$$z_0=\begin{bmatrix}0\\0\\1\end{bmatrix},\ z_1=\begin{bmatrix}0\\0\\1\end{bmatrix},\ z_2=\begin{bmatrix}s_1\\-c_1\\0\end{bmatrix}$$
$$z_3=\begin{bmatrix}s_1\\-c_1\\0\end{bmatrix},\ z_4=\begin{bmatrix}s_1\\-c_1\\0\end{bmatrix},\ z_5=\begin{bmatrix}c_1c_{234}\\s_1c_{234}\\-s_{234}\end{bmatrix} \tag{4.9}$$

Now we can write the Jacobian matrix as shown in equation (4.10):

$$J(q)=[J_1\quad J_2\quad J_3\quad J_4\quad J_5] \tag{4.10}$$

$$J_1=z_0\times(o_5-o_0)=\begin{bmatrix}-s_1c_{234}d_5-s_1c_{23}a_4-s_1c_2a_3\\c_1c_{234}d_5+c_1c_{23}a_4+c_1c_2a_3\\0\\0\\0\\1\end{bmatrix} \tag{4.11}$$

$$J_2=z_1\times(o_5-o_1)=\begin{bmatrix}-s_1c_{234}d_5-s_1c_{23}a_4-s_1c_2a_3\\c_1c_{234}d_5+c_1c_{23}a_4+c_1c_2a_3\\0\\0\\0\\1\end{bmatrix} \tag{4.12}$$

$$J_3=z_2\times(o_5-o_2)=\begin{bmatrix}-c_1(s_{234}d_5+s_{23}a_4+s_2a_3)\\-s_1(s_{234}d_5+s_{23}a_4+s_2a_3)\\s_1(s_1c_{234}d_5+s_1c_{23}a_4+s_1c_2a_3)+c_1(c_1c_{234}d_5+c_1c_{23}a_4+c_1c_2a_3)\\s_1\\-c_1\\0\end{bmatrix} \tag{4.13}$$

$$J_4=z_3\times(o_5-o_3)=\begin{bmatrix}-c_1(s_{234}d_5+s_{23}a_4)\\-s_1(s_{234}d_5+s_{23}a_4)\\s_1(s_1c_{234}d_5+s_1c_{23}a_4)+c_1(c_1c_{234}d_5+c_1c_{23}a_4)\\s_1\\-c_1\\0\end{bmatrix} \tag{4.14}$$

$$J_5 = z_4 \times (o_5 - o_4) = \begin{bmatrix} -c_1 s_{234} d_5 \\ -s_1 s_{234} d_5 \\ s_1^2 c_{234} d_5 + c_1^2 c_{234} d_5 \\ s_1 \\ -c_1 \\ 0 \end{bmatrix} \tag{4.15}$$

4.2 Kinematic Singularities

The Jacobian can also be used to indicate possible configurations at which singularities are present. Singularities are manipulator configurations in which one or more degrees of freedom are made redundant. This reduces the ability of the robot to move in 3D space close to a singularity, even though the area could be well within its workspace. In order to calculate the joint velocities necessary to produce a given Cartesian velocity the Jacobian matrix is inversed. If the inverse Jacobian is applied close to a singularity, the joint velocities approach infinity. For this reason, it is essential that the robot be designed so that it operates away from singularities, both boundary and internal. This is particularly important if the inverse Jacobian is to be calculated in a real time system.

The rank of a matrix is not necessarily constant. Indeed, the rank of the manipulator Jacobian matrix will depend on the configuration q. Configurations for which the rank J(q) is less than its maximum value are called singularities or singular configurations. Identifying manipulator singularities is important for several reasons:

- Singularities represent configurations from which certain directions of motion may be unattainable.
- At singularities, bounded end-effector velocities may correspond to unbounded joint velocities.
- At singularities, bounded end-effector forces and torques may correspond to unbounded joint torques.
- Singularities usually (but not always) correspond to points on the boundary of the manipulator workspace, that is, to points of maximum reach of the manipulator.
- Singularities correspond to points in the manipulator workspace that may be unreachable under small perturbations of the link parameters, such as length, offset, etc.
- Near singularities there will not exist a unique solution to the inverse kinematics problem. In such cases, there may be no solution or there may be infinitely many solutions.

At a singular configuration, the manipulator loses one or more degrees of freedom. The singular configurations are classified into two categories based on the location of end effector in the workspace.

(i) Boundary singularities

The boundary singularities occur when the end effector is on the boundary of the workspace, that is, the manipulator is either fully stretched out or fully retracted. For example, consider the case of two links, 2-DOF planar arm fully stretched out, as shown in Figure (4.1). In this configuration, two links are in a straight line and the end effector can be moved only in a direction perpendicular to the two links because it cannot move out of the workspace. Thus, the manipulator loses one degree of freedom. A similar situation will occur with θ_2 equal 180°. Boundary singularities can be avoided by ensuring that the manipulator is not driven to boundaries of the reachable workspaces during its work cycle [SPO 05].

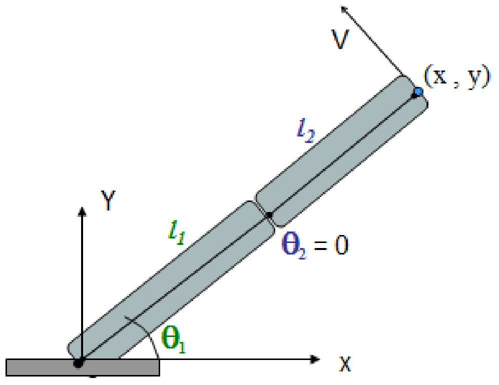


Figure (4.1): 2-DOF planar manipulator fully stretched out

(ii) Internal singularities

Internal singularities as shown in Figure (4.2) occur when the end effector is located inside the reachable workspace of the manipulator. These are caused when two or more joint axes become collinear or at specific end effector configurations [SPO 05].

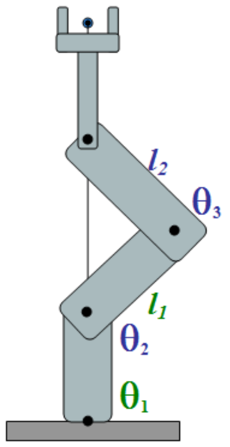


Figure (4.2): Internal Singularities Type

In all the situations, it is essential that singularities are avoided. Therefore, one important criterion for a good design of manipulator configuration is to minimize the singularities.

4.2.1. Computation of Singularities

The computation of internal singularities can be carried out by analyzing the rank of the Jacobian matrix. The Jacobian matrix loses its rank becomes ill conditioned at values of joint variables q at which its determinant vanishes, that means if the Jacobian is a $6 \times n$ matrix and a configuration q is singular if and only if:

$$\det J(q) = 0 \tag{4.16}$$

If we now partition the Jacobian J into 3x3 blocks as

$$J = [Jp \mid Jo] = \left[\begin{array}{c|c} \frac{J_{11}}{J_{21}} & \frac{J_{12}}{J_{22}} \end{array}\right] \tag{4.17}$$

As the singularities are typical of configuration and are not dependent on frames chosen for kinematic analysis, the origin of the end effector frame can be chosen at the end of arm point this will make $J_{12} = 0$. In such a situation computation of determinant is greatly simplified, as

$$|J| = |J_{11}||J_{22}|$$

Hence, for a manipulator with a spherical wrist, the arm singularities are found from $|J_{11}| = 0$, and wrist singularities are found from $|J_{22}| = 0$. However, in our case the Jacobian matrix is non-square then we can find the det J (q) by using the pseudo inverse [SPO 05]. Let A be an m × n matrix, and let A^+ be the pseudo inverse of A. If A is of full rank, then A^+ can be computed as:

$$A^+ = \begin{cases} A^T[AA^T]^{-1} & m \le n \\ A^{-1} & m = n \\ [A^T A]^{-1} A^T & m \ge n \end{cases}$$

Then by applying the MATLAB function (pinv), the non square matrix can be solved.

4.3 Inverse Velocity and Acceleration

The Jacobian relationship

$$\zeta = J\dot{q} \tag{4.18}$$

specifies the end-effector velocity ζ that will result when the joints move with velocity $\dot{q}$. Equation (4.18) represents the forward differential motion model or differential model presented schematically in Figure (4.3), which is similar to the forward kinematic model. Note that the Jacobian J(q) is a function of the joint variables [SPO 05].

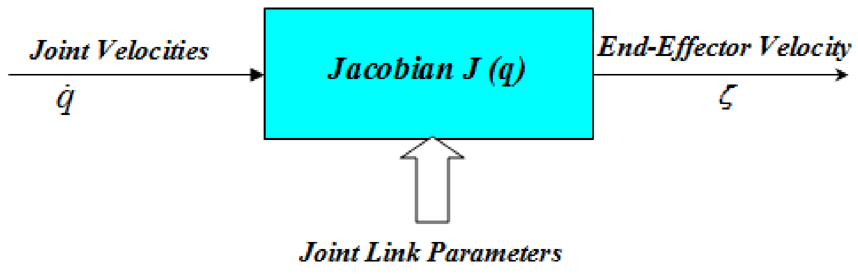


Figure (4.3): The Forward Differential Motion Model

The inverse velocity problem is the problem of finding the joint velocities $\dot{q}$ that produce the desired end-effector velocity. It is perhaps a bit surprising that the inverse velocity relationship is conceptually simpler than inverse position. When the Jacobian is square (i.e., $J \in R^{n\times n}$) and nonsingular, this problem can be solved by simply inverting the Jacobian matrix to give

$$\dot{q} = J^{-1}\zeta \tag{4.19}$$

For manipulators that do not have exactly six links, the Jacobian cannot be inverted. In this case, there will be a solution to equation (4.18) if and only if ζ lies in the range space of the Jacobian. This can be determined by the following simple rank test. A vector ζ belongs to the range of J if and only if

$$rank\ J(q) = rank\ [J(q)\,|\,\zeta] \tag{4.20}$$

In other words, equation (4.18) may be solved for $\dot{q} \in R^n$ provided that the rank of the augmented matrix $[J(q)\,|\,\zeta]$ is the same as the rank of the Jacobian J (q). This is a standard result from linear algebra, and several algorithms exist, such as Gaussian elimination, for solving such systems of linear equations. For the case when n > 6 we can solve for $\dot{q}$ using the right pseudo inverse of J.

4.4 Force/Torque Relationship

Interaction of the manipulator with the environment will produce forces and moments at the end-effector or tool. Let F = $(F_x, F_y, F_z, n_x, n_y, n_z)^T$ represents the vector of forces and torques at the end-effector, expressed in the tool frame. Thus F_x, F_y, F_z are the components of the force at the end-effector, and n_x, n_y, n_z are the components of the torque at the end-effector [SPO 05].

Let τ denote the vector of joint torques, and let δ_X represents a virtual end-effector displacement caused by the force F. Finally, let δq represents the corresponding virtual joint displacement. These virtual displacements are related through the manipulator Jacobian J (q) according to

$$\delta x = J(q)\delta q. \tag{4.21}$$

The virtual work δw of the system is

$$\delta w = F^T \delta X - \tau^T \delta q. \tag{4.22}$$

Substituting (4.21) into (4.22) yields

$$\delta w = (F^T J - \tau^T)\delta q \tag{4.23}$$

This is equal to zero if the manipulator is in equilibrium. Since the generalized coordinate q is independent, we have the equality

$$\tau = J(q)^T F. \tag{4.24}$$

In other words, the end-effector forces are related to the joint torques by the transpose of the manipulator Jacobian according to (4.24). Figure (4.4) shows the AL5B manipulator with a force.

$$F = \left(F_X,\ F_Y,\ F_z, n_x,\ n_y,\ n_z\right)^T \tag{4.25}$$

applied at the end effector. The Jacobian of this manipulator is given by equation (4.10). The resulting joint torques $\tau = [\tau_1\ \tau_2\ \tau_3\ \tau_4\ \tau_5]^T$ is then given by equation (4.24). The transpose of J is:

$$J(q)^T = \begin{bmatrix} J_1 \\ J_2 \\ J_3 \\ J_4 \\ J_5 \end{bmatrix} \tag{4.26}$$

Substituting Equations (4.25) and (4.26) in Equation (4.24) gives

$$\begin{bmatrix} \tau_1 \\ \tau_2 \\ \tau_3 \\ \tau_4 \\ \tau_5 \end{bmatrix}_{(5x1)} = \begin{bmatrix} J_1 \\ J_2 \\ J_3 \\ J_4 \\ J_5 \end{bmatrix}_{(5x6)} * \begin{bmatrix} F_X \\ F_Y \\ F_z \\ n_x \\ n_y \\ n_z \end{bmatrix}_{(6x1)} \tag{4.27}$$

Using MATLAB, we can find the torque for each joint of the robot arm.

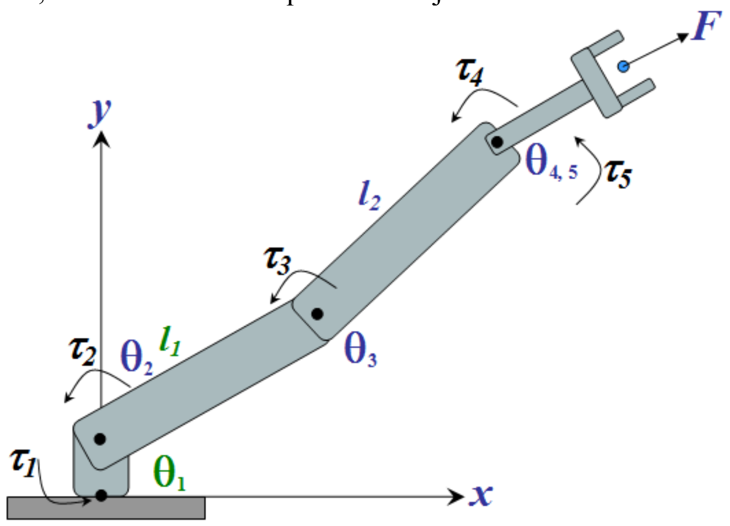


Figure (4.4): AL5B Robot Arm Torque Label

In conclusion this chapter presented the importance of the Jacobian Matrix in the robotics manipulators, this matrix is s defined to represent the mapping of velocities from joint space to Cartesian space. The Jacobian of a manipulator is also used for mapping of forces and torques. When the force and moment at the end-effector are given and the set of joint torques is required. In addition, the Jacobian is very helpful in understanding the singular configuration of the manipulator.

CHAPTER 5 TRAJECTORY PLANNING

The main problem of this chapter is to find a trajectory that connects an initial to a final configuration while satisfying other specified constraints at the endpoints (e.g., velocity and/or acceleration constraints). Without loss of generality, we will consider planning the trajectory for a single joint, since the trajectories for the remaining joints will be created independently and in exactly the same way. Thus, we will concern ourselves with the problem of determining q (t), where q (t) is a scalar joint variable [SPO 05].

We suppose that at time t_0 the joint variable satisfies

$$q(t_0) = q_0 \tag{5.1}$$

$$\dot{q}(t_0) = v_0 \tag{5.2}$$

and we wish to attain the values at t_f

$$q(t_f) = q_f \tag{5.3}$$

$$\dot{q}(t_f) = v_f \tag{5.4}$$

Figure (5.1) shows a suitable trajectory for this motion. In addition, we may wish to specify the constraints on initial and final accelerations. In this case, we have two additional equations

$$\ddot{q}(t_0) = \alpha_0 \tag{5.5}$$

$$\ddot{q}(t_f) = \alpha_f \tag{5.6}$$

The desired path is approximated by a class of polynomial functions. It generates a sequence of time-based "control set points" for the control of manipulator from the initial configuration to its destination. Figure (5.2) shows the trajectory planning block diagram.

5.1 Cubic Polynomial Trajectories

Suppose that we wish to generate a trajectory between two configurations, and that we wish to specify the start and end velocities for the trajectory. One way to generate a smooth curve such as that shown in Figure (5.1) is by a polynomial function of t. If we have four constraints to satisfy, such as (5.1)-(5.3), we require a polynomial with four independent coefficients that can be chosen to satisfy these constraints. Thus, we consider a cubic trajectory of the form

$$q(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 \qquad \textit{For Distance} \tag{5.7}$$

Then, the desired velocity is given as

$$\dot{q}(t) = a_1 + 2a_2 t + 3a_3 t^2 \qquad \textit{For Velocity} \tag{5.8}$$

$$\ddot{q}(t) = 2a_2 + 6a_3 t \qquad \textit{For Acceleration} \tag{5.9}$$

Combining equations (5.7) and (5.8) with the four constraints yields four equations in four unknowns

$$q_0 = a_0 + a_1 t_0 + a_2 t_0^2 + a_3 t_0^3 \tag{5.10}$$

$$v_0 = a_1 + 2a_2 t_0 + 3a^3 t_0^2 \tag{5.11}$$

$$q_f = a_0 + a_1 t_f + a_2 t_f^2 + a_3 t_f^3 \tag{5.12}$$

$$v_f = a_1 + 2a_2 t_f + 3a_3 t_f^2 \tag{5.13}$$

These four equations can be combined into a single matrix equation

$$\begin{bmatrix} 1 & t_0 & t_0^2 & t_0^3 \\ 0 & 1 & 2t_0 & 3t_0^3 \\ 1 & t_f & t_f^2 & t_f^3 \\ 0 & 1 & 2t_f & 3t_f^2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} q_0 \\ v_0 \\ q_f \\ v_f \end{bmatrix} \tag{5.14}$$

For example Figure (5.3) shows cubic trajectory with $q_0 = 10$, $q_f = -20$, $t_0=0$, $t_f = 1$ and $V_0 = 0$, $V_f=0$. The corresponding angle, velocity and acceleration curves are given in Figures (5.3).

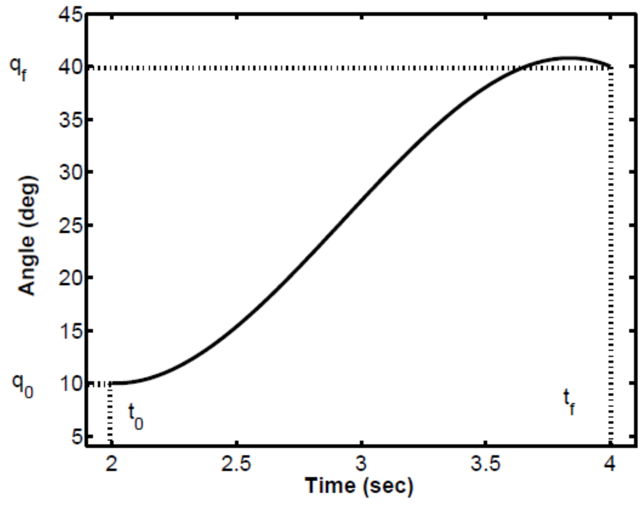


Figure (5.1): Typical Joint Space Trajectory

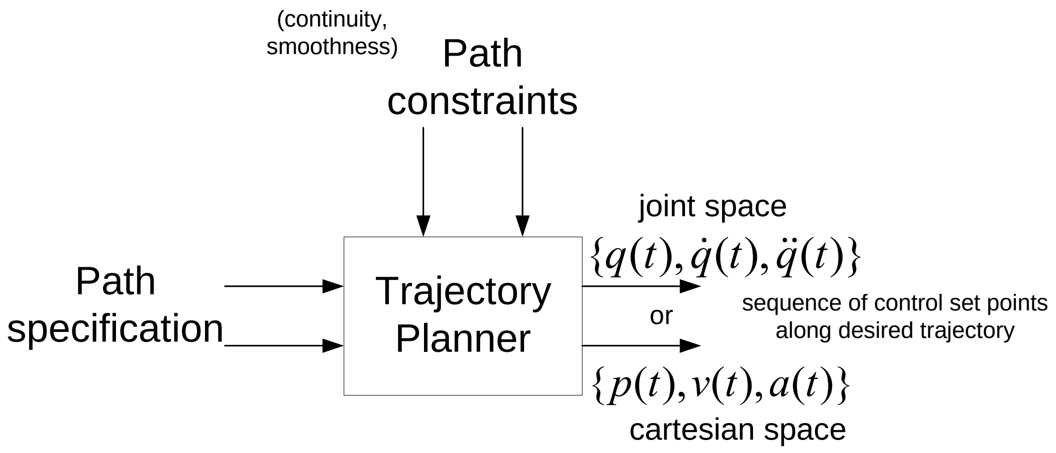


Figure (5.2): Trajectory Planning Block Diagram

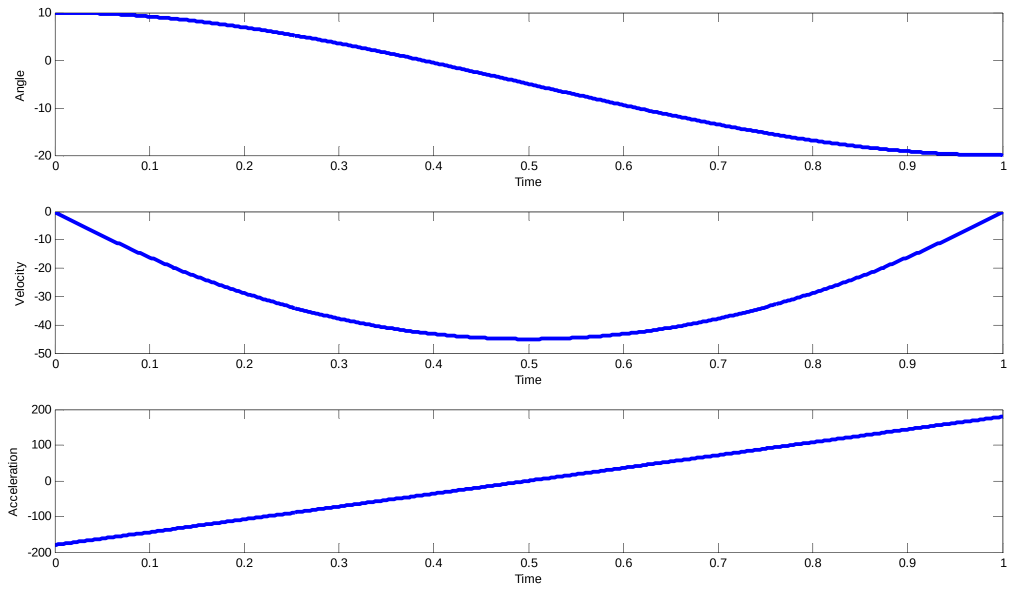


Figure (5.3): Cubic polynomial trajectory

5.2 Quantic Polynomial Trajectories

As can be seen in Figure (5.3), a cubic trajectory gives continuous positions and velocities at the start and finish points times but discontinuities in the acceleration. The derivative of acceleration is called the jerk. A discontinuity in acceleration leads to an impulsive jerk, which may excite vibration modes in the manipulator and reduce tracking accuracy.

For this reason, one may wish to specify constraints on the acceleration as well as on the position and velocity. In this case, we have six constraints (one each for initial and final configurations, initial and final velocities, and initial and final accelerations). Therefore, we require a fifth order polynomial

$$\begin{aligned} q(t) &= a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5 && \textit{For Distance} \\ \dot{q}(t) &= a_1 + 2a_2 t + 3a_3 t^2 + 4a_4 t^3 + 5a_5 t^4 && \textit{For Velocity} \\ \ddot{q}(t) &= 2a_2 + 6a_3 t + 12a_4 t^2 + 20a_5 t^3 && \textit{For Acceleration} \end{aligned} \tag{5.15}$$

Using equations (5.1) - (5.6) and taking the appropriate number of derivatives, we obtain the following equations,

$$q_0 = a_0 + a_1 t_0 + a_2 t_0^2 + a_3 t_0^3 + a_4 t_0^4 + a_5 t_0^5 \tag{5.16}$$
$$v_0 = a_1 + 2a_2 t_0 + 3a_3 t_0^2 + 4a_4 t_0^3 + 5a_5 t_0^4 \tag{5.17}$$
$$\alpha_0 = 2a_2 + 6a_3 t_0 + 12a_4 t_0^2 + 20a_5 t_0^3 \tag{5.18}$$
$$q_f = a_0 + a_1 t_f + a_2 t_f^2 + a_3 t_f^3 + a_4 t_f^4 + a_5 t_f^5 \tag{5.19}$$
$$v_f = a_1 + 2a_2 t_f + 3a_3 t_f^2 + 4a_4 t_f^3 + 5a_5 t_f^4 \tag{5.20}$$
$$\alpha_f = 2a_2 + 6a_3 t_f + 12a_4 t_f^2 + 20a_5 t_f^3 \tag{5.21}$$

which can be written as

$$\begin{bmatrix} 1 & t_0 & t_0^2 & t_0^3 & t_0^4 & t_0^5 \\ 0 & 1 & 2t_0 & 3t_0^3 & 4t_0^3 & 5t_0^4 \\ 0 & 0 & 2 & 6t_0 & 12t_0^2 & 20t_0^3 \\ 1 & t_f & t_f^2 & t_f^3 & t_f^4 & t_f^5 \\ 0 & 1 & 2t_f & 3t_f^2 & 4t_f^3 & 5t_f^4 \\ 0 & 0 & 0 & 6t_f & 12t_f^2 & 20t_f^3 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{bmatrix} = \begin{bmatrix} q_0 \\ v_0 \\ \alpha_0 \\ q_f \\ v_f \\ \alpha_f \end{bmatrix} \tag{5.22}$$

Figure (5.4) shows a quintic polynomial trajectory with q(0) = 0, q(2) = 40 with zero initial and final velocities and accelerations.

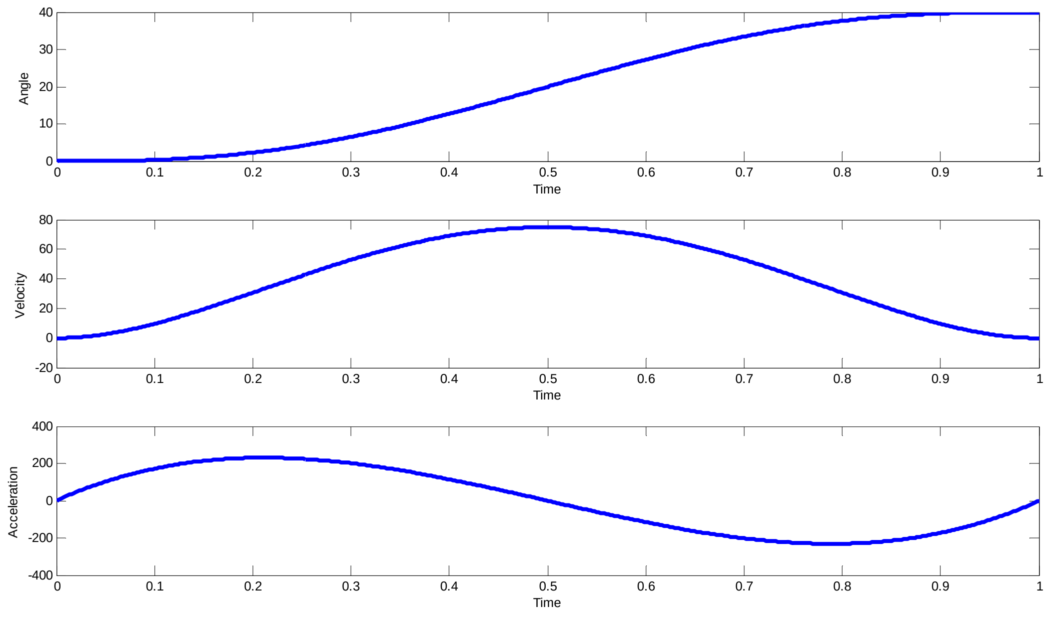


Figure (5.4): Quintic Polynomial Trajectory

5.3 Linear Segments with Parabolic Blends (LSPB)

Another way to generate suitable joint space trajectories is by so-called Linear Segments with Parabolic Blends or (LSPB) for short. This type of trajectory is appropriate when a constant velocity is desired along a portion of the path. The LSPB trajectory is such that the velocity is initially "ramped up" to its desired value and then "ramped down" when it approaches the goal position. To achieve this, we specify the desired trajectory in three parts. The first part from time t_0 to time t_b is a quadratic polynomial. This results in a linear "ramp" velocity. At time t_b, called the blend time, the trajectory switches to a linear function. This corresponds to a constant velocity. Finally, at time $t_f - t_b$ the trajectory switches once again, this time to a quadratic polynomial so that the velocity is linear.

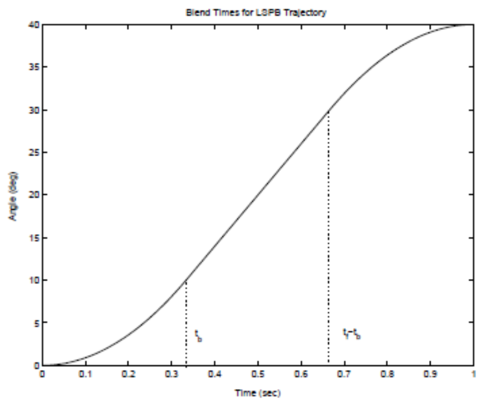


Figure (5.5): Blend times for LSPB trajectory

We choose the blend time t_b so that the position curve is symmetric as shown in Figure (5.5) for convenience suppose that t_0 = zero and $\dot{q}(tf) = 0 = \dot{q}(0)$. Then between times 0 and t_b we have

$$q(t) = a_0 + a_1 t + a_2 t^2 \tag{5.23}$$

so that the velocity is

$$\dot{q}(t) = a_1 + 2a_2 t \tag{5.24}$$

The constraints $q(0) = q_0$ and $\dot{q}(0) = 0$ imply that

$$a_0 = q_0 \tag{5.25}$$

$$a_1 = 0 \tag{5.26}$$

At time t_b we want the velocity to equal a given constant, say V. Thus, we have

$$\dot{q}(t_b) = 2a_2 t_b = V \tag{5.27}$$

This implies that

$$a_2 = \frac{V}{2t_b} \tag{5.28}$$

Therefore, the required trajectory between 0 and t_b with $\alpha = \frac{V}{t_b}$ is given as

$$q(t) = q_0 + \frac{V}{2t_b}t^2 = q_0 + \frac{\alpha}{2}t^2 \tag{5.29}$$

$$\dot{q}(t) = \frac{V}{t_b}t = \alpha t \tag{5.30}$$

$$\ddot{q} = \frac{V}{t_b} = \alpha \tag{5.31}$$

Where α denotes the acceleration.
Now, between time t_f and t_f - t_b, the trajectory is a linear segment (corresponding to a constant velocity V)

$$q(t) = a_0 + a_1 t = a_0 + Vt \tag{5.32}$$

Since, by symmetry,

$$q(\frac{t_f}{2}) = \frac{q_0 + q_f}{2} \tag{5.33}$$

We have

$$\frac{q_0 + q_f}{2} = a_0 + V\frac{t_f}{2} \tag{5.34}$$

which yields

$$a_0 = \frac{q_0 + q_f - V\,t_f}{2} \tag{5.35}$$

Since the two segments must "blend" at time t_b we require

$$q_0 + \frac{V}{2}t_b = \frac{q_0 + q_f - V\,t_f}{2} + Vt_b \tag{5.36}$$

which gives upon solving for the blend time t_b

$$t_b = \frac{q_0 - q_f + V\,t_f}{V} \tag{5.37}$$

Note that we have the constraint $0 < t_b \le \frac{t_f}{2}$. This leads to the inequality

$$\frac{q_f - q_0}{V} < t_f \le \frac{2(q_f - q_0)}{V} \tag{5.38}$$

the inequality can be written in another way

$$\frac{q_f - q_0}{t_f} < V \le \frac{2(q_f - q_0)}{t_f} \tag{5.39}$$

Thus, the specified velocity must be between these limits or the motion is not possible. The portion of the trajectory between t_f-t_b and t_f is now found by symmetry considerations. The complete LSPB trajectory is given by

$$q(t) = \begin{cases} q_0 + \frac{a}{2}t^2 & 0 \le t \le t_b \\ \frac{q_f + q_0 - Vt_f}{2} + Vt & t_b < t \le t_f - t_b \\ q_f - \frac{at^2{}_f}{2} + at_f t - \frac{a}{2}t^2 & t_f - t_b < t \le t_f \end{cases} \tag{5.40}$$

Figure (5.6) shows such an LSPB trajectory q_0= zero, q_f = 40, and t_0=0, t_f=1, where the maximum velocity V = 60. In this case $t_b = \frac{1}{3}$. The velocity and acceleration curves are given in the same Figure.

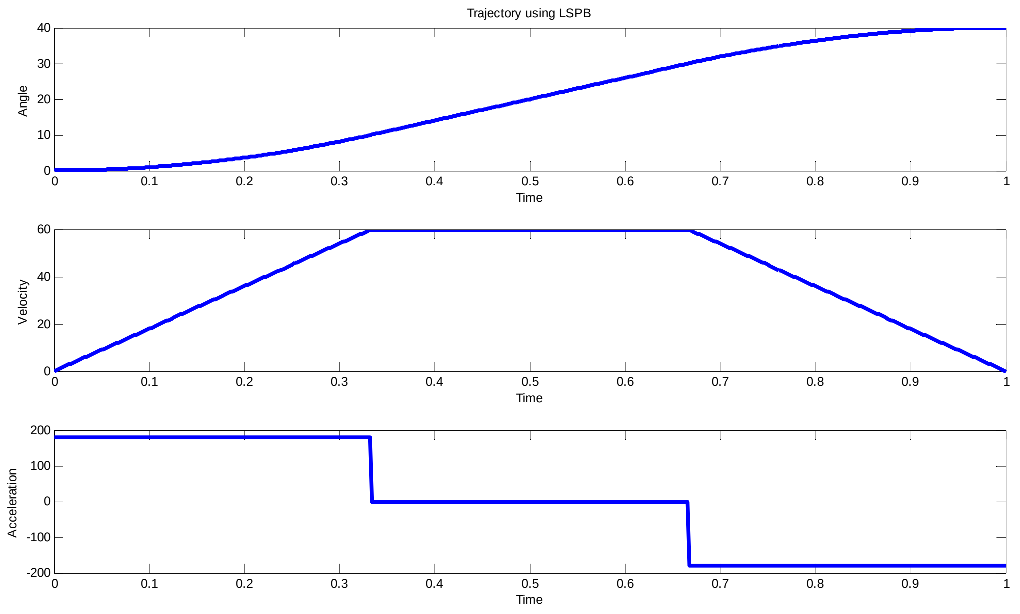


Figure (5.6): Trajectory using LSPB

CHAPTER 6 ROBOT HARDWARE AND SOFTWARE

To achieve a control of a robot arm by using personal computer, we must make the connection between the robot and PC. This connection is called interface connection and it is done by using a microcontroller. Microcontrollers are inexpensive devices commonly used in embedded computing applications to impart computing and smart decision-making capabilities to machines, products, and processes. Microcontrollers are designed to interface to and interact with electrical/electronic devices, sensors and actuators, and high-tech gadgets to automate systems. Microcontrollers are generally embedded directly into the product or process for automated decision-making. They are not meant to interface with human beings; however, microcontrollers do not have graphical user interface (GUI) capabilities that are common in many personal computer (PC) applications. The complete control process can be divided into two categories, hardware and software. Now we will discuss individually each one of these categories.

6.1 Hardware Environment

The hardware environment for this thesis consists of a CUBLOC microcontroller, a PC, and a data link between the PC and AL5B robot arm. The microcontroller is a device that interfaces to sensors and robot actuators and performs embedded computing. The PC is used to control the robot by GUI; it is used to write the user defined embedded program which is to be run on the microcontroller. Also serves as a debugging environment when prototyping microcontroller based products and processes. It allows the user to receive sensory information and other selected data. A data link is needed for the microcontroller and PC to communicate. In this thesis, we use a serial communication link(wire) between the microcontroller and PC. Figure (6.1) shows the hardware environment used in this thesis. A robot arm with CUBLOC Kit is shown connected to a PC through a DB-9 serial cable.

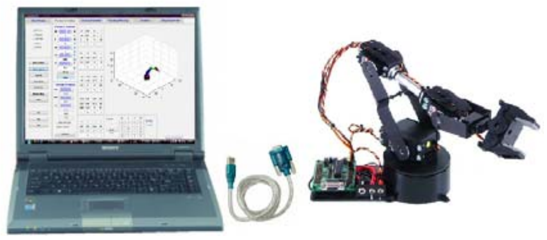


Figure (6.1): Hardware environment

6.1.2. CUBLOC Microcontroller

The CUBLOC "CB280" with 64-pin is manufactured by Comfile Technology where 49 of its pins can be used for I/O operations. It can be powered with 6-12DC volt, where regulators on its Kit supply it with a steady 5VDC. The CB280 comes with 80KB Program Memory (Flash), 4KB Electronically Erasable Programmable ROM (EEPROM), and a small amount of 2K RAM. The microprocessor used in CB280 is Atmega128 that runs at 18.432 MHz with Program Speed of 36,000/sec. The CUBLOC is programmed with CublocStudio in BASIC and/or Ladder Logic Language; the instruction set is permanently stored on the CUBLOC ROM. The user-defined program is downloaded into the EEPROM from a PC through a DB-9 serial cable connection between the PC and CB280 Kit. CB280 have eight Channels for Analog Inputs 10-bit

ADCs, 6 Channel Analog Outputs 16-bit PWMs (DACs), 2 Channel 16-bit High Speed Counters, 2 High-speed hardware-independent serial ports and 4 Channels External Interrupts. Each pin can source (supply) a maximum current of 40mA. See [COM 05] for more details on CUBLOC hardware features.

CB280
ENG.MOHAMMED REYAD ABU QASSEM

Figure (6.2): CB280 Chip and CUBOC Kit

6.1.3. Personal Computer

As previously mentioned, the PC is used to write Basic programs that the CUBLOC executes and to display sensory data processed by the CUBLOC and control the robot using MATLAB GUI. Any PC that supports MATLAB can be used. In this thesis, a Pentium class PC running MATLAB 7.5 under Windows vista is used.

6.1.4. Interface Kit

The AL5B consists of a group of 6 RC servo motors. An electronics interfacing circuit (Figure (6.3)) is designed to connect the servo motor group to the CUBLOC microcontroller. This interfacing circuit consists of input and output ports; those can be easily connected to the CUBLOC and the servo motor group. Moreover, designed circuit has 6 PWM channels, 8 ADC channels and other I/O pins.

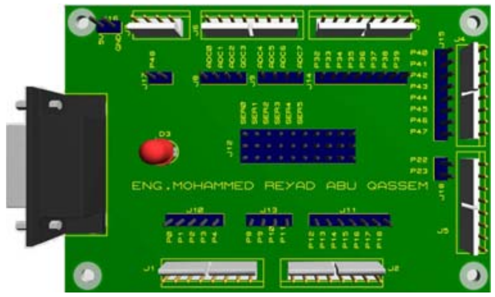


Figure (6.3): Interface kit

Referring to the RC servo motor shown in Figure (6.4), there are three wires to the RC servo motor. The black wire is ground and red wire is for power. The third wire “yellow” for inputting PWM signal. We can use the PWM to easily implement an RC Servo motor into AL5B robot.

The RC servo motor will move to a location set by pulse and duty value and will hold its position. By being able to control the exact angles at which the RC servo stops, we can control the RC servo as freely as we want. In this case, of control the RC servo motor has internal feedback. Because we want to measure the angle of all joint in the arm we modified the servo motor by cutting the internal signal feedback and connect it to the ADC input from interface kit to achieve the closed loop control by send these angles to the computer by CB280 microcontroller.

Figure (6.4): RC Servo Motor

6.1.5. DB-9 Serial Cable

The CUBLOC and PC communicate through a serial communication link. A variety of serial communication links are currently in use. The CUBLOC uses the RS-232 serial communication. The serial cable, which is used in this thesis, is called the DB-9 serial cable. The cable links a serial, or COM, port on the PC to the CB280 Kit. This allows the user to download a program into the CUBLOC. In addition, this serial connection enables data communication between the CUBLOC and the PC. The pinout schematic for a DB-9 serial cable is shown in Figure (6.5).

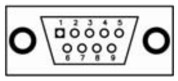


Figure (6.5): Schematic for a DB-9 Serial Cable

Each of the pins performs a specific task to which it is assigned. The assignment for each pin is shown in Table (6.1).

Table (6.1): Pin assignments for a DB-9 serial cable

Pin #	Label	Signal Name	Signal Type
1	CD	Carrier detect	Control
2	RD	Received data	Data
3	TD	Transmitted data	Data
4	DTR	Data terminal ready	Control
5	GND	Signal ground	Ground
6	DSR	Data set ready	Control
7	RTS	Request to send	Control
8	CTS	Clear to send	Control
9	RI	Ring indicator	Control

6.2 Software Environment

Software environment can be divided into two parts: the CUBLOC microcontroller program and MATLAB Program. In CUBLOC program, we write a code to make the interfacing between PC and AL5B arm. The MATLAB program consists of the Serial Communication code and the graphical user interface (GUI). In this section, we explain the complete system functions, CUBLOC program and MATLAB program.

6.2.1. Overall System

The complete system functions is shown in Figure (6.6) it consists of four parts, forward kinematics, inverse kinematic, trajectory planning and a controller.

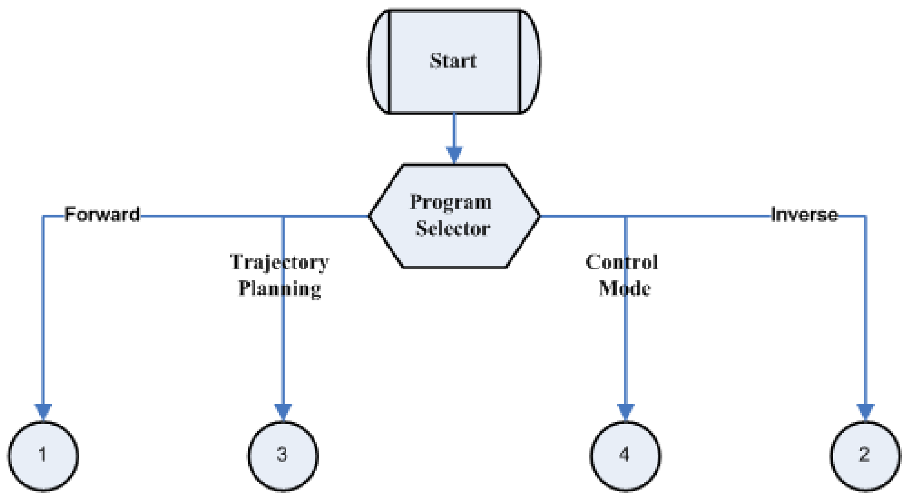


Figure (6.6): Complete System Functions

The forward kinematics (FK) consists of finding the position and orientation of the end-effector in the space knowing the movements of its joints and after calculating the FK we can compute the Jacobian and arm singularity. The FK flowchart is shown in Figure (6.7). The inverse kinematics (IK) consists of the determination of the joint variables corresponding to a given end-effector position and orientation. The IK flowchart is shown in Figure (6.7).

The path is defined as a sequence of robot configurations in particular order with no regard to the timing of these configurations. ***Trajectory*** is concerned with the specific time for each part of the path. Each joint velocity at the specified joint positions needs to be found. This is accomplished using the ***Jacobian***. Figure (6.8) shows Trajectory Planning Flowchart. The last part is the ***controller*** of the robot arm by GUI program. Next, we describe various elements of the software environment used in this thesis.

6.2.2. CUBLOC Program

The CB280 is programmed using the CublocStudio in basic programming language. It is a BASIC-like language developed by Parallax, Inc. In addition to simple arithmetic, the CB280 executes certain task specific commands. See [COM 05] for more details on the CUBLOC CB280 programming language.

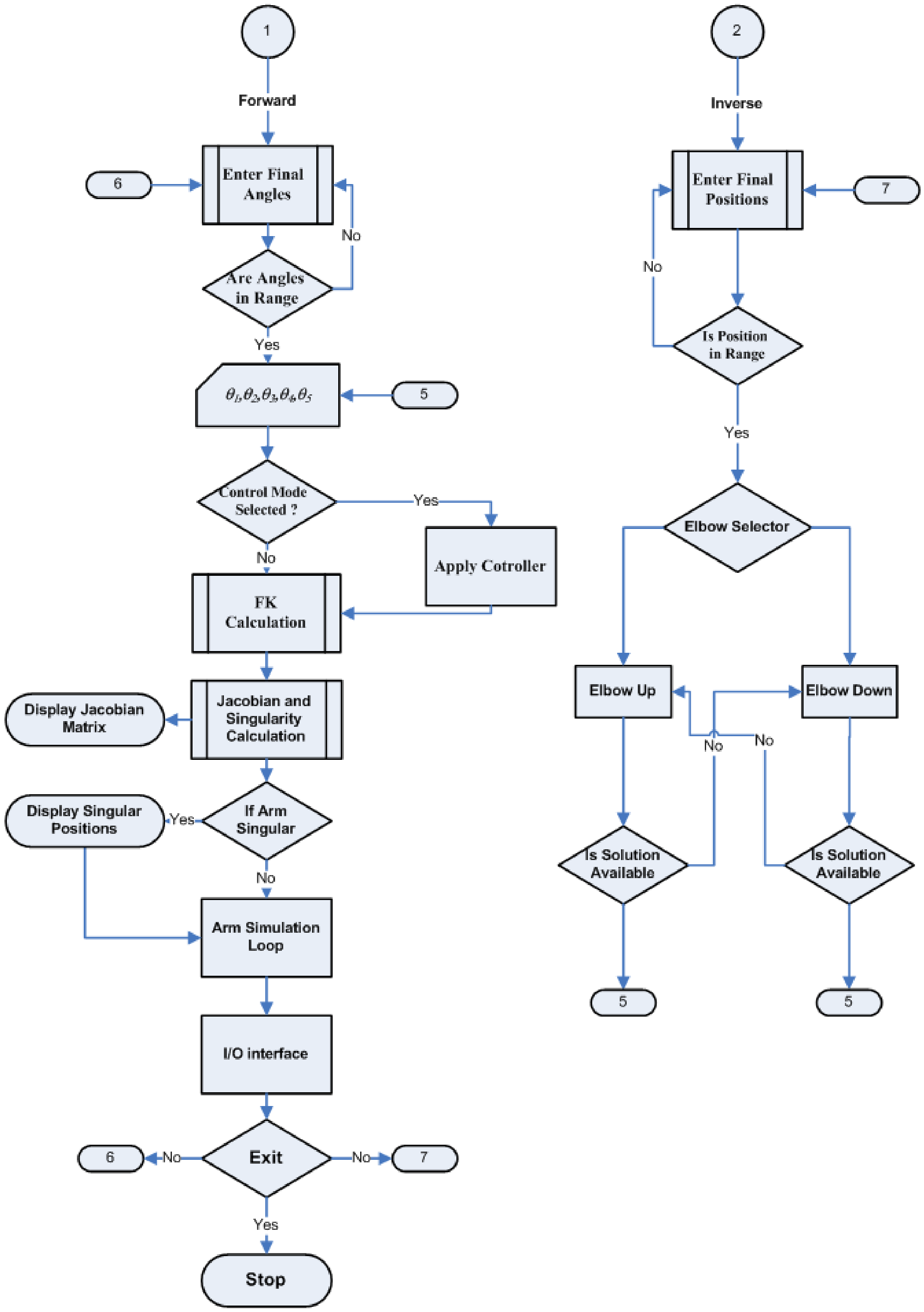


Figure (6.7): Forward and Inverse Kinematic Flowchart

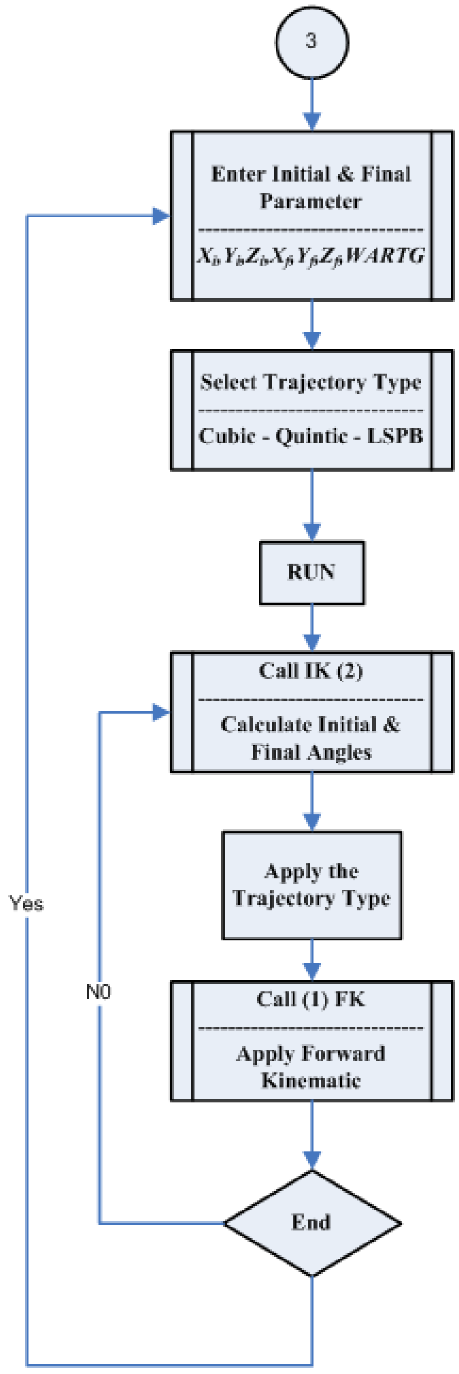

3
Enter Initial & Final Parameter
$X_b Y_b Z_b X_f Y_f Z_f$ WARTG
Select Trajectory Type
Cubic - Quintic - LSPB
RUN
Call IK (2)
Calculate Initial & Final Angles
Apply the Trajectory Type
Call (1) FK
Apply Forward Kinematic
End
Yes
NO

Figure (6.8): Trajectory Planning Flowchart

6.2.3. Serial Communication

The serial communication is a low-level protocol used for data communication between two or more devices. Serial communication uses a data port to send/receive data in a serial manner, i.e., one bit at a time. Programming two or more devices to communicate serially requires that the devices operate at the same communication rates. Configure serial port communications

Here is an example serial session connecting MATLAB to the serial port (COM1) with a baud rate of 4800:

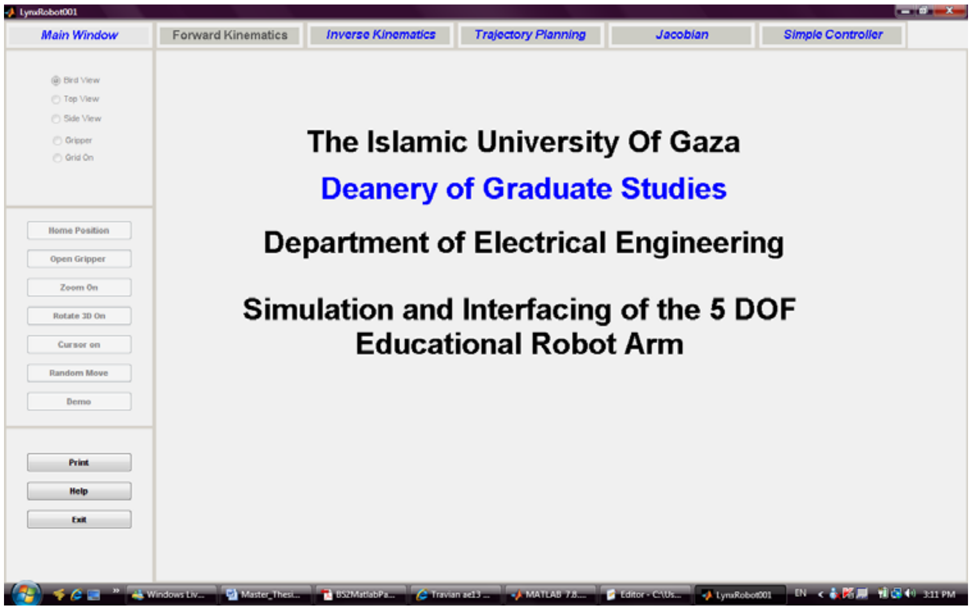
```
s = serial ('COM1');
set(s,'BaudRate', 4800);
fopen(s);
fprintf(s,'*IDN?')
out = fscanf(s);
fclose(s)
delete(s)
clear s
```

The “*IDN?” Command above is a typical instrument command and can be replaced by any command that is valid for your specific device. *IDN queries the device for identification information, which is returned to out. If your device does not support this command, or if it is connected to a different serial port, you should modify the above example accordingly.

➢ **Graphical User Interface**

The GUI for the AL5B robotic arm control was written in MATLAB program; MATLAB is a powerful software package that allows for plotting data in multiple dimensions and it easy to work with three dimensions. The program consists of six different windows of the arm control.

The main screen of the GUI program, shown in Figure (6.9), consists of five-tab window, Forward Kinematic, Inverse Kinematic, Trajectory Planning, Jacobian and controller. Now we will explain the entire screen in detail in the following explanation:



The Islamic University Of Gaza

Deanery of Graduate Studies

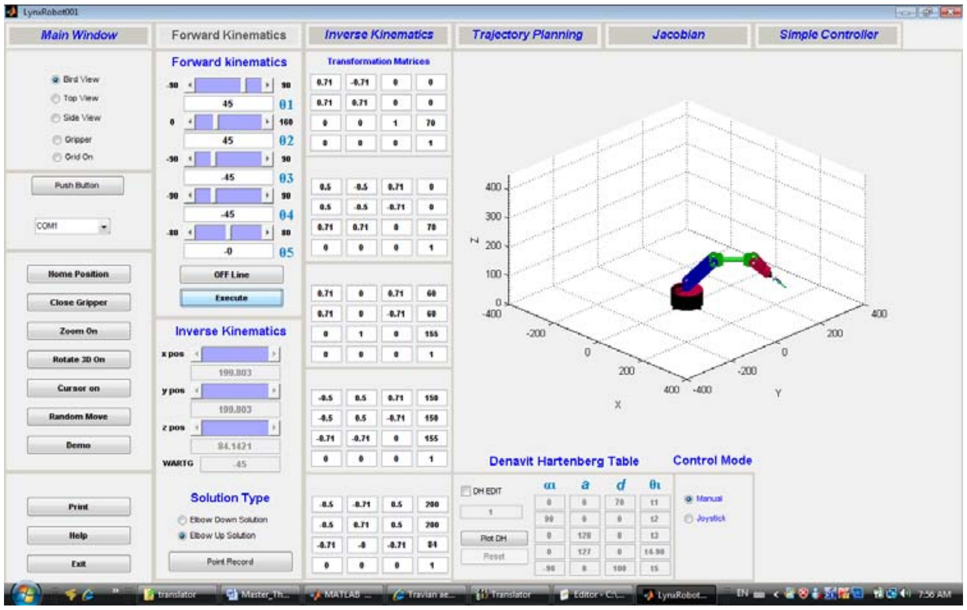
Department of Electrical Engineering

Simulation and Interfacing of the 5 DOF Educational Robot Arm

Figure (6.9): GUI Main Window

1- Forward and inverse kinematic window

There are two parts of this window; the first is the forward kinematics and the second is the inverse kinematics. This window is shown in Figure (6.10).The main function of the first part is to allow the user to compute the position of the end effector by entering the angles in the edit boxes or by moving the angle slider. There are two methods to move the robot arm by using off line button or by execute button. After entering all angles, the transformation matrices are displayed in the FK window. The robot arm simulation is shown in the right of the window; this allows the user to see the robot 3D motion. Another function of this window is displaying the DH table, this table allows the user to edit the DH parameters and plot it as shown in Figure (6.11).



Forward kinematics

-90	45	90	θ1
0	45	160	θ2
-90	-45	90	θ3
-90	-45	90	θ4
-80	-0	80	θ5

Inverse Kinematics

x pos	199.803
y pos	199.803
z pos	84.1421
WARTG	45

Solution Type: ( ) Elbow Down Solution [✓ Elbow Up Solution]

Transformation Matrices

0.71	-0.71	0	0
0.71	0.71	0	0
0	0	1	70
0	0	0	1

0.5	-0.5	0.71	0
0.5	-0.5	-0.71	0
0.71	0.71	0	70
0	0	0	1

0.71	0	0.71	60
0.71	0	-0.71	60
0	1	0	155
0	0	0	1

-0.5	0.5	0.71	150
-0.5	0.5	-0.71	150
-0.71	-0.71	0	155
0	0	0	1

-0.5	-0.71	0.5	200
-0.5	0.71	0.5	200
-0.71	-0	-0.71	84
0	0	0	1

Denavit Hartenberg Table

DH EDIT: 1

α_i	a	d	θ_i
0	0	70	t1
90	0	0	t2
0	120	0	t3
0	127	0	t4-90
-90	0	100	t5

Control Mode: [✓ Manual] ( ) Joystick

Figure (6.10): Forward and Inverse Kinematic Window

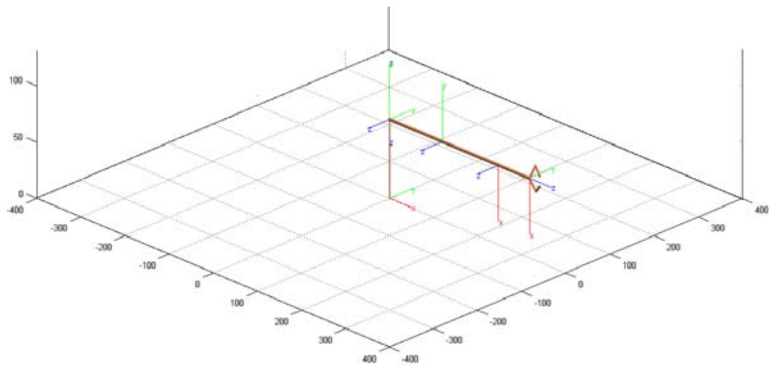
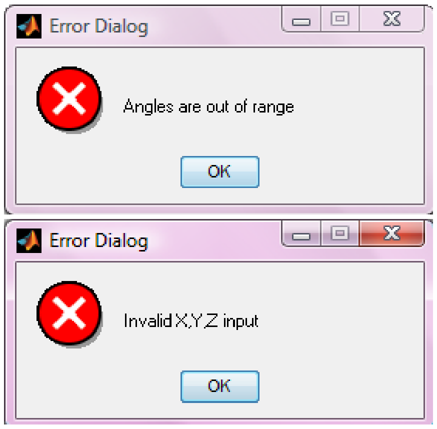


Figure (6.11): Robot Arm Frame Coordinate

In the same window, there are two main buttons to control the robot by joystick mode or manual mode. If we select manual mode, we can use the mouse or keyboard to edit the angles, and we can select the joystick mode to move the robot.

The Second part of this window called inverse kinematic. The main function of the inverse kinematic is finding the joint variables in terms of the end-effector position and orientation. This function allows the user to compute the joint variables of the arm by entering the position (x, y, z, and WARTG) of the end effector in the edit boxes or by moving the angle slider.

After entering the position and the WARTG angle, the user must select the type of solution by selecting Elbow UP or Elbow down and by pressing the same button in forward kinematic. We can show the arm simulation and all transformation matrix of the arm. If the user enters wrong angle or position, a dialog message appears as shown in Figure (6.12).


Error Dialog

Angles are out of range

OK

Error Dialog

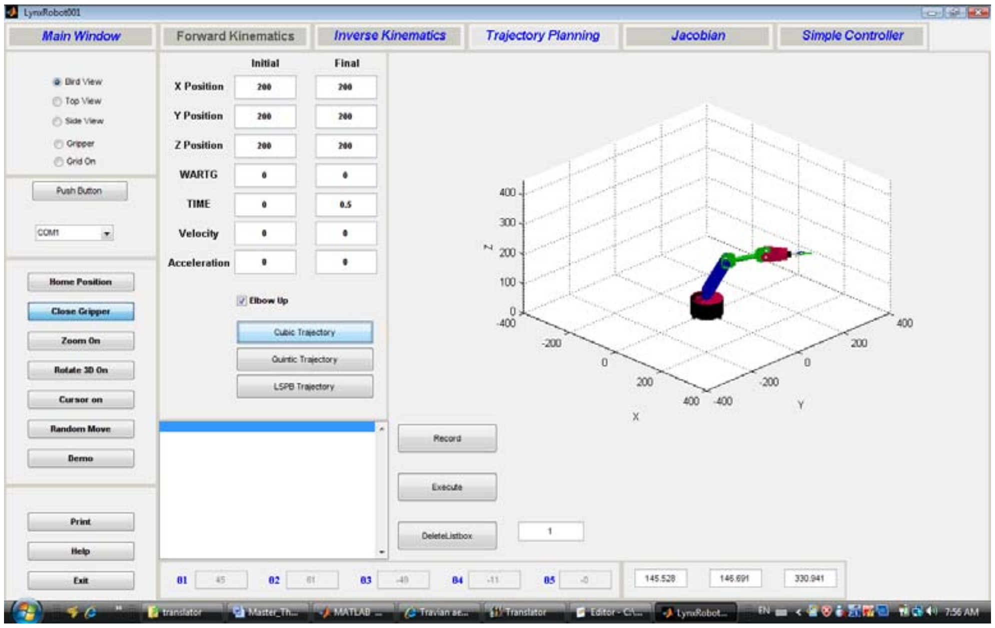
Invalid X,Y,Z input

OK

Figure (6.12): Error Dialog message

2- Trajectory Planning Window

The manipulator shall move in the workspace along the pre-specified desired paths to accomplish the desired action. A trajectory is a path with all necessary timing specifications to calculate the required position, and velocity of the robot configuration. In Trajectory Planning Window as shown in Figure (6.13), the user must enter the initial and final position, WARTG and time, and then the user can choose the type of polynomial as cubic, quintic or LSPB trajectory, then select the type of solution Elbow down or up then the robot arm can move with this trajectory. Another function of this window is to record the path of robot motion or multi-path by pressing record button and by pressing execute button we can repeat the robot arm motion.



	Initial	Final
X Position	200	200
Y Position	200	200
Z Position	200	200
WARTG	0	0
TIME	0	0.5
Velocity	0	0
Acceleration	0	0

[✓ Elbow Up]

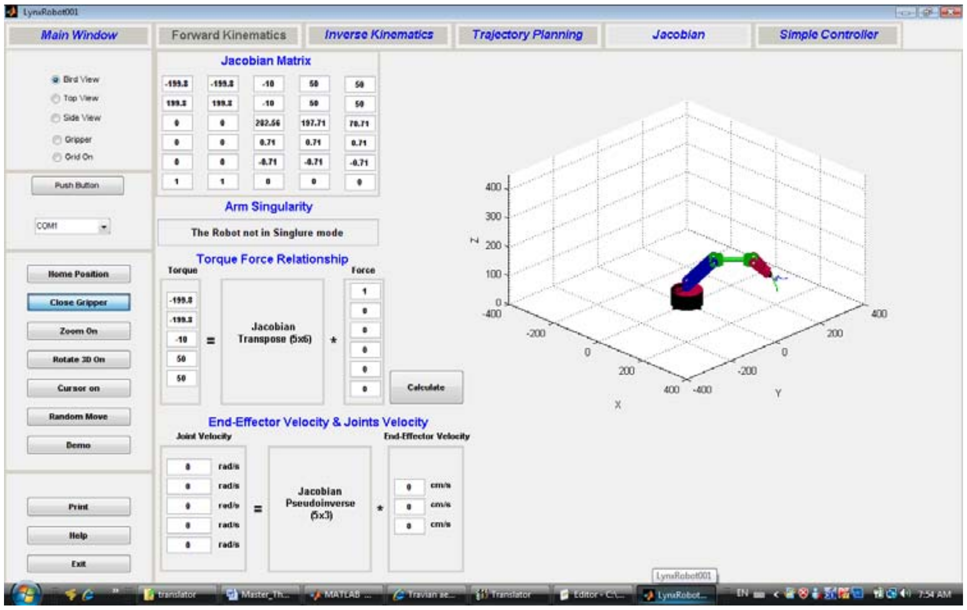
θ1 45 θ2 61 θ3 -40 θ4 -11 θ5 -0

145.528 146.691 330.941

Figure (6.13): Trajectory Planning Window

3- Jacobian Window

The Jacobian is one of the most important quantities in the analysis and control of robot motion. The Jacobian can also be used to indicate possible configurations at which singularities are present. Figure (6.14), shows the Jacobian matrix computed by the program while moving the arm. Moreover, another function of this window is to display a message appears stating the existence of the singularity of the arm if it occurs during the work. This window can also be used to compute the relationship between static force and torque by entering the force at all joint. The last function of this window is to compute the relationship between the end-effector velocity and the joints velocity by using the pseudo inverse Jacobian. The user enters the end-effector velocity in cm/s and the program can get the result in rad/s.



Jacobian Matrix

-199.8	-199.8	-10	50	50
199.8	199.8	-10	50	50
0	0	282.56	197.71	70.71
0	0	0.71	0.71	0.71
0	0	-0.71	-0.71	-0.71
1	1	0	0	0

Arm Singularity

The Robot not in Singlure mode

Torque Force Relationship

Torque: -199.8, -199.8, -10, 50, 50 = Jacobian Transpose (5x6) * Force: 1, 0, 0, 0, 0, 0

End-Effector Velocity & Joints Velocity

Joint Velocity: 0 rad/s, 0 rad/s, 0 rad/s, 0 rad/s, 0 rad/s = Jacobian Pseudoinverse (5x3) * End-Effector Velocity: 0 cm/s, 0 cm/s, 0 cm/s

Figure (6.14): Jacobian Window

4- Controller Window

Controller is very important for control of the robot. In this window, the user can choose the type of controller he wants to apply for the robot. In future work, we can use Simulink for classical controller or advance controller, like P, PD, PID, Fuzzy, Optimal, Neural Network, Sliding Mode, H-infinity, Backstepping control, etc.

After the selection of the controller type, the robot will move under this controller and the step response can be shown.

6.3 System Limitations

Figure (6.15) illustrates the joints and their directions of rotation. The arrows show where the joint angle is zero and the plus and minus signs indicate whether the angle is positive or negative in that direction of rotation. Joint zero rotates the robot relative to the base and Joint 5 is the roll of the wrist.

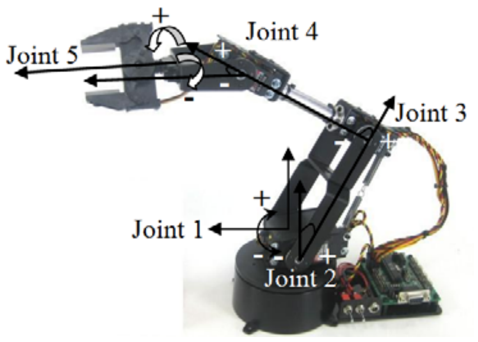


Figure (6.15): Robot Arm Joints

Table 3 lists the Joint limits of the AL5B robot arm in raw values (the servo control input values are integers [0,…., 254]) and the correspondent values in degrees.

Table (6.2): Robot Arm Joint Limits

Joint	**Joint limits** [Duty-time]	**Joint limits [Deg]**
1	$0.650742 < \theta_1 < 2.465258$	$-90 \leq \theta_1 \leq +90$
2	$0.765 < \theta_2 < 2.453825$	$0 \leq \theta_2 \leq +180$
3	$2.31583 < \theta_3 < 0.68461$	$-90 \leq \theta_3 \leq +90$
4	$0.634138 < \theta_4 < 2.437746$	$-90 \leq \theta_4 \leq +90$
5	$2.34222 < \theta_5 < 0.67778$	$-80 \leq \theta_5 \leq +80$
gripper	$1.726323 \leq \theta_6 \leq 0.159678$	$0 \leq \theta_6 \leq +60$

CHAPTER 7 RESULTS AND DISCUSSIONS

This chapter presents simulations and results. Mathematical modeling and kinematic analysis of a low cost AL5B Robot arm, was carried out in this study. Robot arm was mathematically modeled with Denavit Hartenberg (D-H) method. Forward Kinematics, Inverse Kinematics, Velocity Kinematic “Jacobian”, and Trajectory Planning solutions are generated and implemented by the developed software. An analysis technique was introduced to reduce the multiple solutions in inverse kinematics part. The developed software included a simulator part to test the motional kinematics and to show the relevant motion in 3D. A typical example, calculated with the generated software, was included here for the user.

7.1 Experimental Results

In this section, the experimental results of simulation and interfacing are introduced with brief comparison between the simulation results and the physical arm results.

7.1.1. Forward Kinematics

An initial position angle is given in Figure (7.1) with zero θ $\theta i(0) = 0 \ \ , i = 0,1,....,5$. The total transformation matrix of this position is shown in equation (7.1). This matrix gives the initial position and orientation of the robot arm.

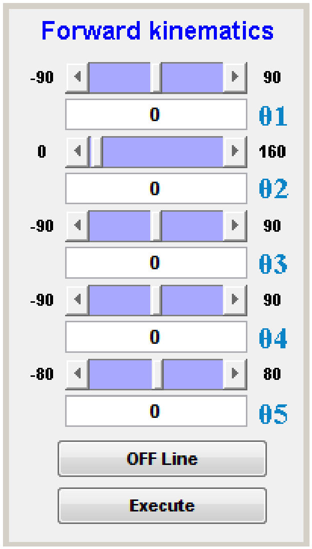


Figure (7.1): Initial Position Angle

$$T_5^0(final) = \begin{bmatrix} 0 & 0 & 1 & 347 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 70 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (7.1)$$

From equation (7.1), we find that the (x, y, and z) position of the end-effector is equal to (347, 0, and 70). Figure (7.2) shows the 3D graphics of AL5B in this position.

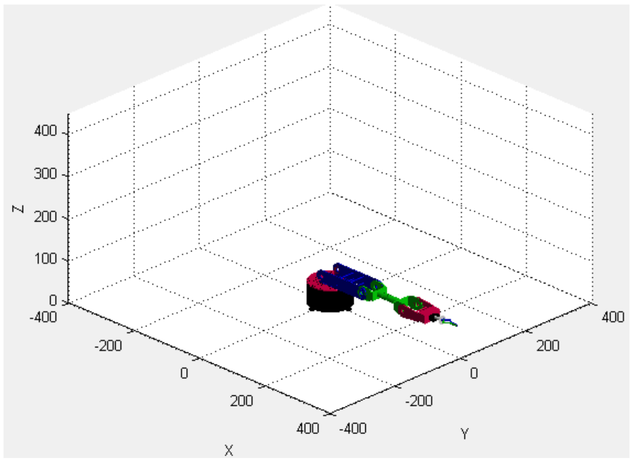
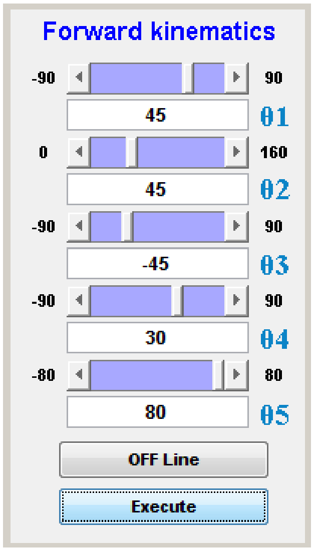


Figure (7.2): AL5B 3D Graphics Initial Position

When θ values are changed from zero to given values as shown in Figure (7.3).The total transformation matrix, T_5^0 between the base of the robot arm and the end effectors is shown in equation (7.2).



Forward kinematics

-90 90
45 θ1
0 160
45 θ2
-90 90
-45 θ3
-90 90
30 θ4
-80 80
80 θ5

Figure (7.3): Final Position Angle

$$T_5^0(final) = \begin{bmatrix} -0.63 & -0.47 & 0.61 & 211 \\ 0.76 & -0.23 & 0.61 & 211 \\ -0.15 & 0.85 & 0.5 & 205 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (7.2)$$

From equation (7.2), we can find the position of the end-effector equal [x, y, and z] = [211, 211, and 205]. Figure (7.4) shows the 3D graphics of AL5B in this position.

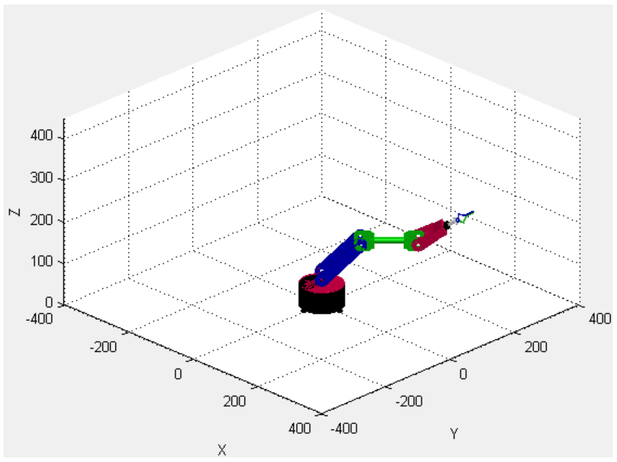


Figure (7.4): AL5B 3D Graphics Final Position

T_5^0 is determined by the developed software and it is the final forward kinematics solution of the robot arm. T_5^0 Matrix values are checked against the physical positions of the robot arm in Table 7.1.

Table (7.1): Differences Between Calculated And Physical Values Of AL5B Robot Arm

Position Values	T_5^0 Values (mm)	Measured Values (mm)	Percentage Error
X	211	208.85	≈ 1%
Y	211	209.15	≈ 1%
z	205	194	≈ 5%

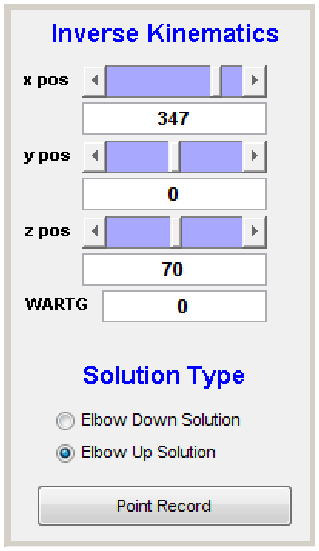
When calculating xyz coordinates of the target, position compared with the measured coordinates as in Table 7.1, it is observed that the values were very close to each other. However, there are some errors in (x, y and z) direction, but the error in z direction is larger than the error in x or y direction. This error is a result of the weight of the arm and the contents of the servo motor, so dynamic modeling should be applied to eliminate this error.

7.1.2. Inverse Kinematics

On the other hand, inverse kinematics equations will be used to determine the target position and its orientation for the robot arm. The developed software will calculate the required angles for target orientation and target positioning.

An initial position is given in Figure (7.5) with x = 347, y = zero, z =70 and WARTG = zero values. The total transformation matrix of this position is shown in equation (7.1). This matrix gives the initial position and orientation of the robot arm. Figure (7.2), shows the 3D graphics of AL5B in this position.

When x, y, z and WARTG values are changed from initial position to given values in Figure (7.6) and select the solution type as ***Elbow Up*** .The total transformation matrix, T_5^0 between the base of the robot arm and the end effectors is shown in equation (7.3).


Inverse Kinematics

x pos: 347

y pos: 0

z pos: 70

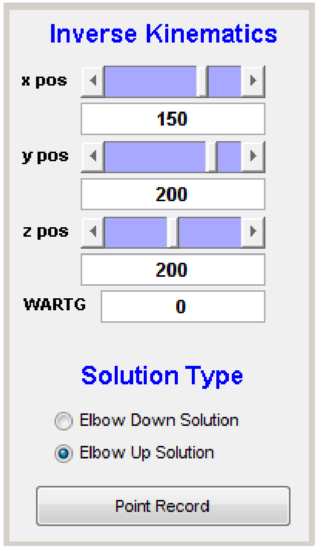
WARTG: 0

Solution Type

( ) Elbow Down Solution

[✓ Elbow Up Solution]

Figure (7.5): X, Y, Z and WARTG Initial position


Inverse Kinematics

x pos: 150

y pos: 200

z pos: 200

WARTG: 0

Solution Type

( ) Elbow Down Solution

[✓ Elbow Up Solution]

Figure (7.6): X, Y, Z and WARTG Final position

$$T_5^0(final) = \begin{bmatrix} 0 & -0.8 & 0.6 & 150 \\ 0 & 0.6 & 0.8 & 200 \\ -1 & 0 & 0 & 200 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{7.3}$$

From forward kinematic editor we can find the angles of the end-effector [θ_1, θ_2, θ_3, θ_4 and θ_5] equal [53, 79, -73, -6 and 0]. Figure (7.7), shows the 3D graphics of AL5B in this position with ***Elbow Up*** solution.

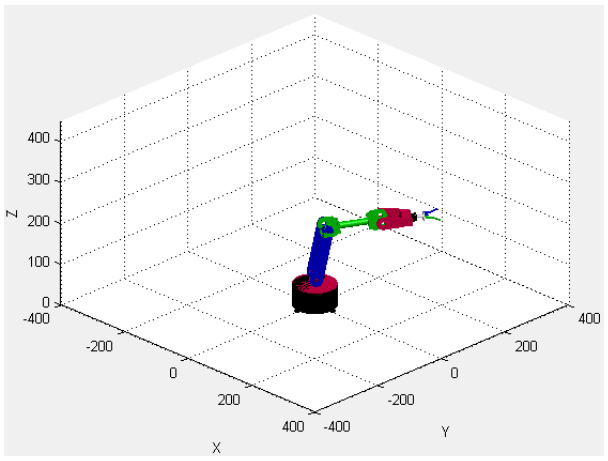


Figure (7.7): AL5B 3D Graphics Final Position with Elbow Up Solution

If we select the solution type as ***Elbow Down***. The total transformation matrix, T_5^0 between the base of the robot arm and the end effectors is shown in equation (7.3). From forward kinematic editor we can find the angle of the end-effector equal [θ_1, θ_2, θ_3, θ_4 and θ_5] = [53, 3, 73, -76 and 0]. Figure (7.8) shows the 3D graphics of AL5B in this position with Elbow Down solution.

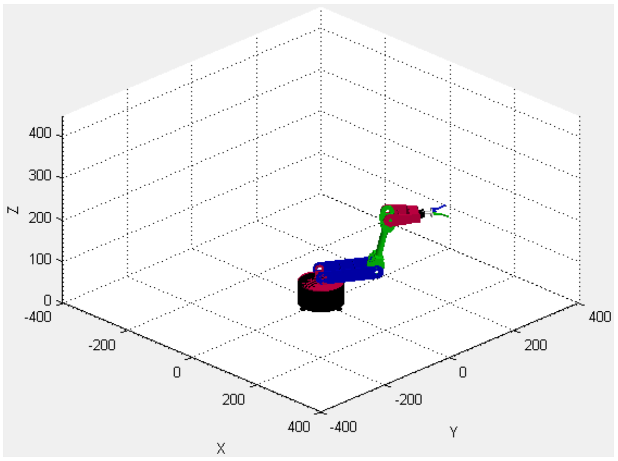


Figure (7.8): AL5B 3D Graphics Final Position with Elbow Down Solution

The desired values are checked against the physical positions of the robot arm in Table 7.2. When desired values xyz coordinates of the target position are compared with the measured coordinates as in Table 7.2, it is observed that the values were very close to each other.

Table (7.2): Differences Between Desired and Real Values Of AL5B Robot Arm Positions

Position Values	Desired Values (mm)	Measured Values (mm)	Percentage Error
X	150	148	≈ 1%
Y	200	198	≈ 1%
Z	200	192	≈ 4%

However, there are some errors in (x, y and z) direction. From table 7.1 and 7.2, we can see the error in x and y direction is less than the error in z direction. This error is normal and we can eliminate this error by applying any controller low like PID, but in z direction the error is large and we can eliminate this error by analyzing the dynamic modeling for AL5B robot arm.

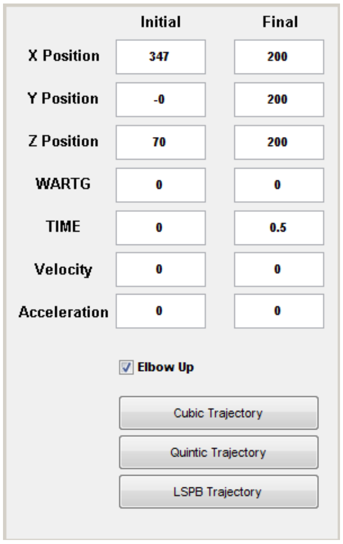
7.1.3. Trajectory Planning

Straight-line motions are most common in the industrial applications; however, movement on a line is mostly obtained by specifying the discrete time joint-displacements at a constant time rate. The velocity and acceleration of the points can be calculated from the numerical approximation of the time derivatives. Several methods were used to compress the describing data of the trajectories as cubic, quintic and LSPB trajectory.

Cubic Trajectory

Cubic polynomial or third order polynomial approximation describes the path parametrically as a function of time with the position and velocity constraints at initial time t = zero and final time t_f. Third order polynomials provide continuity of displacements and velocities, but may result in discontinuity of accelerations, thus abrupt changes in joint torques. Higher order polynomials are required to guarantee the smoothness of the joint accelerations.

An initial and final position is given in trajectory editor as shown in Figure (7.9) with x = 347, y = zero, z =70 and WARTG = zero values and final position with x = 200, y = 200, z =200 and WARTG = zero values. As shown in the same figure we choose the solution type as Elbow Up.



	Initial	Final
X Position	347	200
Y Position	-0	200
Z Position	70	200
WARTG	0	0
TIME	0	0.5
Velocity	0	0
Acceleration	0	0

[✓] Elbow Up

Figure (7.9): Trajectory Editor

After selecting cubic trajectory button, the robot arm will be moved to the final position and when that happens we see the cubic trajectory curve as shown in Figure (7.10). This figure is divided into three parts shows the relation between the angle, velocity and acceleration with time.

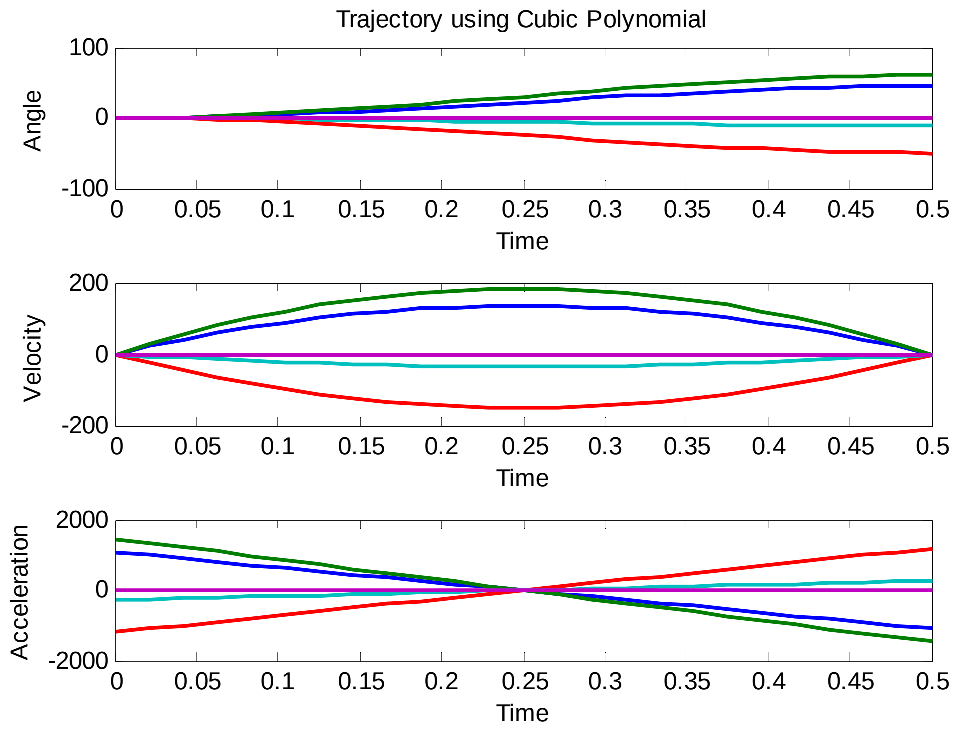


Figure (7.10): Cubic Polynomial Trajectory

Quintic Trajectory

An initial and final position is given in trajectory editor as shown in Figure (7.9) with x = 347, y = zero, z =70 and WARTG = zero values and final position with x = 200, y = 200, z =200 and WARTG = zero values. As shown in the same figure we choose the solution type as Elbow Up.

After selecting quintic trajectory button, the robot arm will be moved to the final position and when that happens we see the quintic trajectory curve as shown in Figure (7.11). This figure is divided into three parts are shows the relation between the angle, velocity and acceleration with time.

Parabolic blend with linear segments

This type of trajectory is appropriate when a constant velocity is desired along a portion of the path. The LSPB trajectory is such that the velocity is initially “ramped up” to its desired value and then “ramped down” when it approaches the goal position.

An initial and final position is given in trajectory editor as shown in Figure (7.12) with x = 347, y = zero, z =70 and WARTG = zero values and final position with x = 200, y =200, z =200 and WARTG = zero values. The LSPB needs to enter the initial and final velocity for all angles, but in MATLAB program the LSPB function compute the velocity for all angles by using equation (5.39). As shown in the same figure, we choose the solution type as Elbow Up.

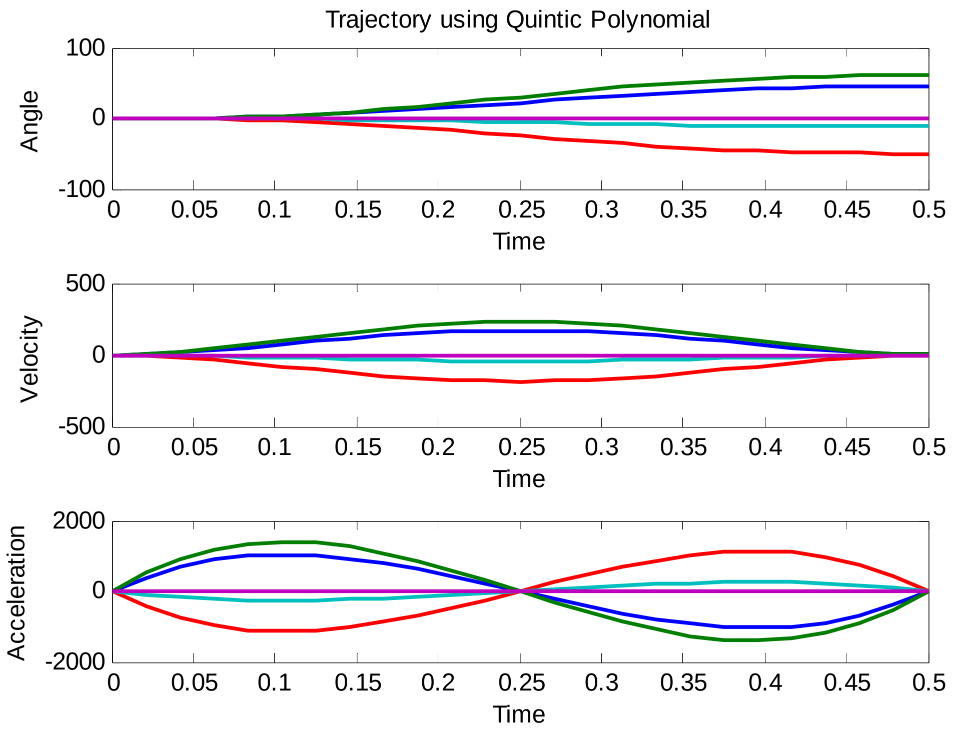
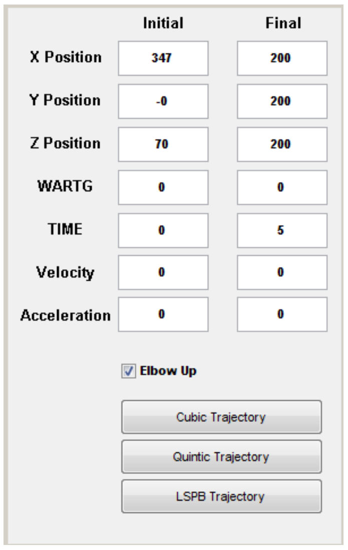


Figure (7.11): Quintic Polynomial Trajectory



	Initial	Final
X Position	347	200
Y Position	-0	200
Z Position	70	200
WARTG	0	0
TIME	0	5
Velocity	0	0
Acceleration	0	0

☑ Elbow Up

Figure (7.12): LSPB Trajectory Editor

After selecting LSPB trajectory button, the robot arm will be moved to the final position and when that happens we see the LSPB trajectory curve as shown in Figure (7.13). This figure is divided into three parts and shows the relation between the angle, velocity and acceleration with time.

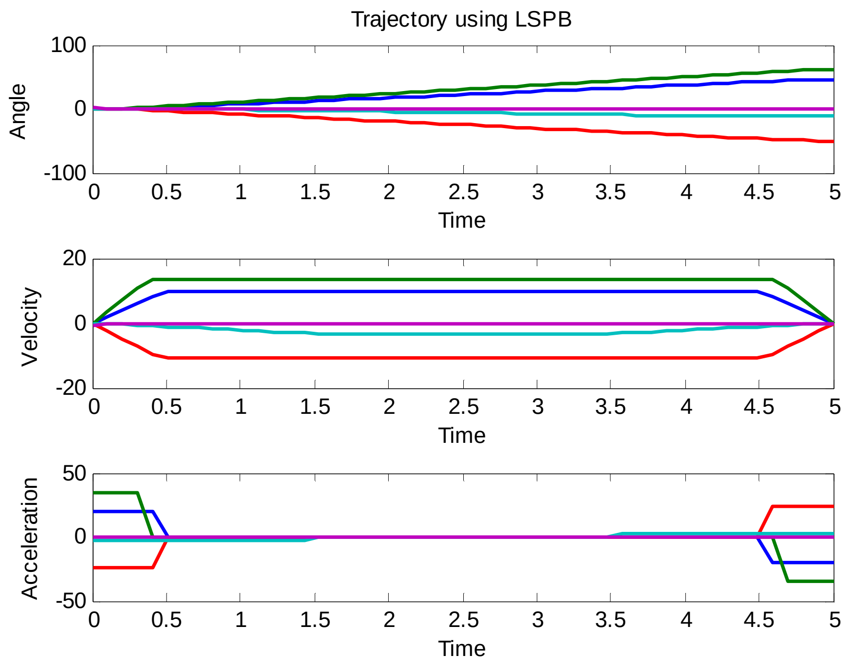


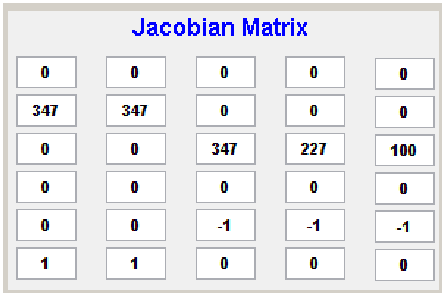
Figure (7.13): LSPB Polynomial Trajectory

7.1.4. Velocity Kinematic

Jacobian Matrix

Example 1:

When the robot arm is moving the developed software will calculate the Jacobian matrix for target orientation and target positioning. An initial position is given in Figure (7.5) with x = 347, y = zero, z =70 and WARTG = zero values. The Jacobian matrix of this position is shown in Figure (7.14).

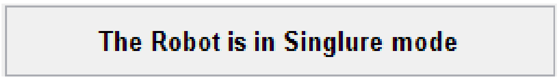


Jacobian Matrix

0	0	0	0	0
347	347	0	0	0
0	0	347	227	100
0	0	0	0	0
0	0	-1	-1	-1
1	1	0	0	0

Figure (7.14): Example 1 Jacobian Matrix

From the Jacobian matrix in Figure (7.14), the developed software compute the arm singularity. In this position the robot well be in singular mode, and the message shown in Figure (7.15) will show that.

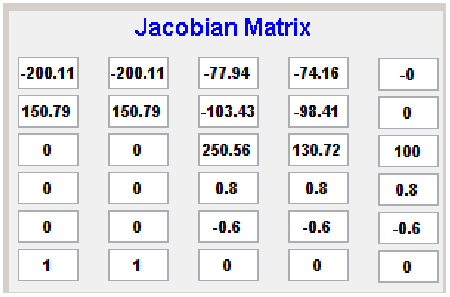


The Robot is in Singlure mode

Figure (7.15): Example 1 Singular Mode

Example 2

When position values are changed from home position to another position as shown in Figure (7.6). The developed software will calculate the Jacobian matrix for the target orientation and target position. The Jacobian matrix of this position is shown in Figure (7.16).

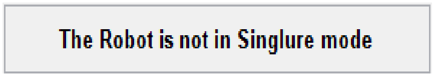


Jacobian Matrix

-200.11	-200.11	-77.94	-74.16	-0
150.79	150.79	-103.43	-98.41	0
0	0	250.56	130.72	100
0	0	0.8	0.8	0.8
0	0	-0.6	-0.6	-0.6
1	1	0	0	0

Figure (7.16): Example 2 Jacobian Matrix

From the Jacobian matrix in Figure (7.16), the developed software compute the arm singularity. In this position the robot is in non-singular mode, and the message shown in Figure (7.17) will show that.



The Robot is not in Singlure mode

Figure (7.17): Example 2 Singular Mode

Torque and Force

The developed software computes the relationship between static force and torque by entering the force at end-effector and compute the torque.

Example 3

An initial force (N) is given in Figure (7.18) with zero values. The torque results is shown in the same figure equal zero (N.m).

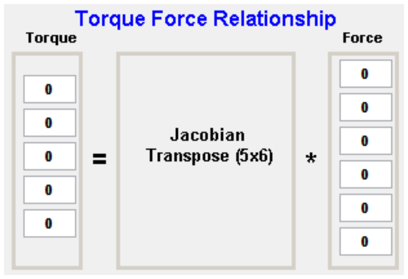
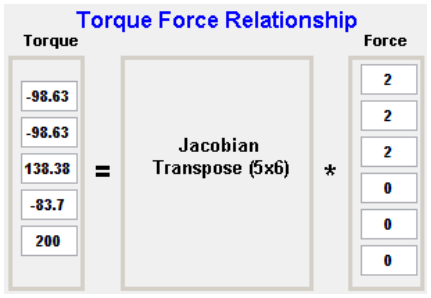


Figure (7.18): Example 3 Torque - Force Relationship

Example 4

When force values are changed from zero value to another value as shown in Figure (7.19). The developed software will calculate the torque as shown in Figure (7.19).


Torque Force Relationship

Torque				Force
-98.63	=	Jacobian Transpose (5x6)	*	2
-98.63				2
138.38				2
-83.7				0
200				0
				0

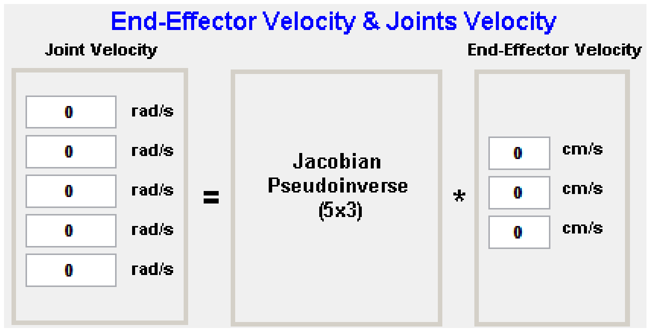
Figure (7.19): Example 4 Torque - Force Relationship

End-effector and Joints Velocity

We can compute the relationship between the end-effector velocity (cm/s) and the joints velocity (rad/s) by using developed software.

Example 5

An initial end-effector velocity (cm/s) is given in Figure (7.20) with zero values. The joints velocity results have shown in the same figure equal zero (rad/s).

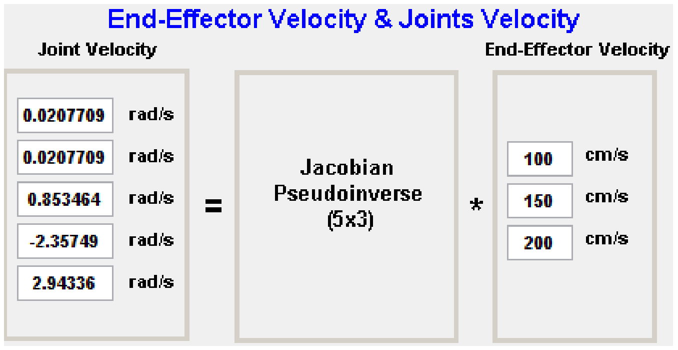

End-Effector Velocity & Joints Velocity

Joint Velocity				End-Effector Velocity
0 rad/s	=	Jacobian Pseudoinverse (5x3)	*	0 cm/s
0 rad/s				0 cm/s
0 rad/s				0 cm/s
0 rad/s				
0 rad/s				

Figure (7.20): Example 5 End-effector and Joints Velocity

Example 6

When the end-effector velocity values are changed from zero value to another value, the developed software will calculate joints velocity as shown in Figure (7.21).


End-Effector Velocity & Joints Velocity

Joint Velocity				End-Effector Velocity
0.0207709 rad/s	=	Jacobian Pseudoinverse (5x3)	*	100 cm/s
0.0207709 rad/s				150 cm/s
0.853464 rad/s				200 cm/s
-2.35749 rad/s				
2.94336 rad/s				

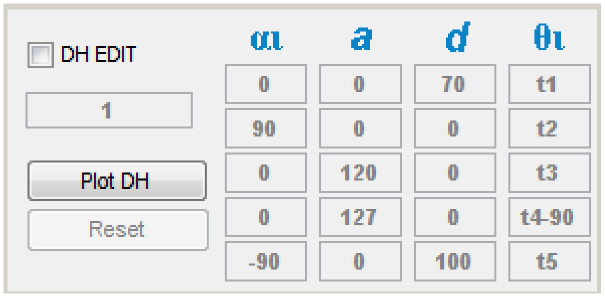
Figure (7.21): Example 6 End-effector and Joints Velocity

Denavit-Hartenberg

The developed software allows user to change the DH parameters shown in Figure (7.22) and plot the robot link.

Example 7

An initial parameter is given in Figure (7.22) with α = [0,90,0,0,-90], a = [0,0,120,127,0], d =[70,0,0,0,100] values. Figure (7.23) shows the Robot Arm Frame Coordinate.



DH EDIT

1

αι	a	d	θι
0	0	70	t1
90	0	0	t2
0	120	0	t3
0	127	0	t4-90
-90	0	100	t5

Figure (7.22): DH Initial Parameter

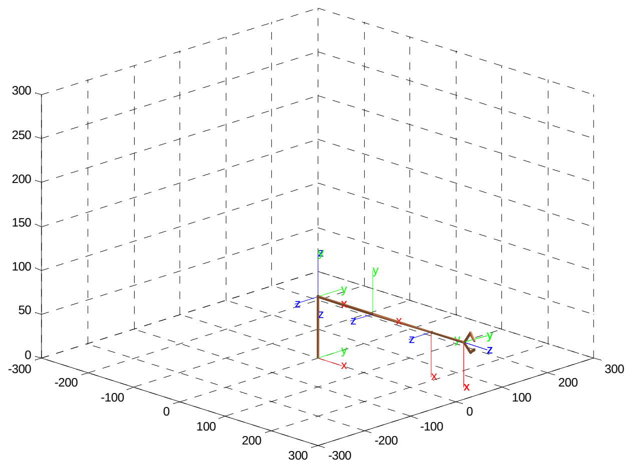
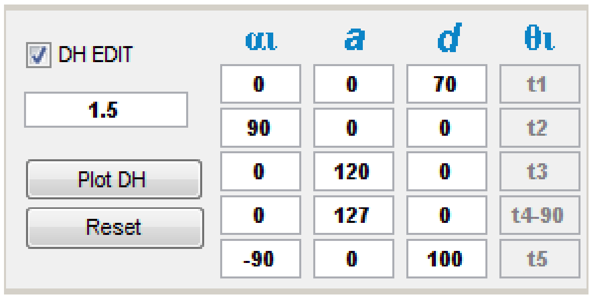


Figure (7.23): Example 7 Robot Arm Frame Coordinate

Example 8

When the DH parameter values are changed from initial value to with α = [0,0,0,0,0], a = [0,0,100,127,0], d =[70,0,0,0,100] and link radius = 1.5 as shown in Figure (7.24). The developed software plots the Robot Arm Frame Coordinate as shown in Figure (7.25). If the user enters wrong DH parameters then the 3D model of the robot arm will be as shown in Figure (7.26). We note that false DH parameters break down the arm.



☑ DH EDIT

1.5

αι	a	d	θι
0	0	70	t1
90	0	0	t2
0	120	0	t3
0	127	0	t4-90
-90	0	100	t5

Figure (7.24): Example 8 DH Parameter

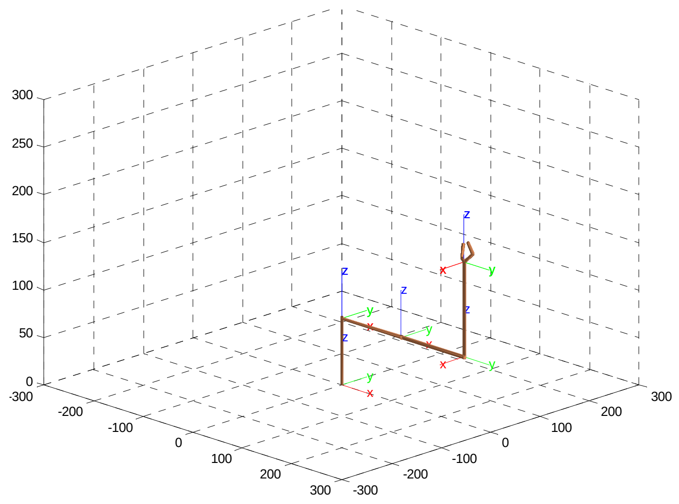


Figure (7.25): Example 8 Robot Arm Frame Coordinate

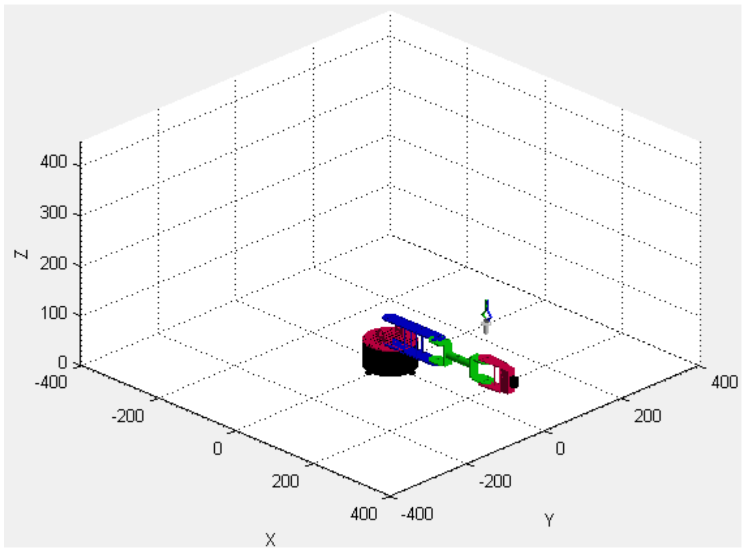


Figure (7.26): Example 8 Robot Arm 3D Graphical

Pick and Place

One of the main applications used in industrial robot is to move things from one place to another.

Example 9

We want to move four blocks from initial position to another position, the initial positions for the blocks are = [280 0 45; 280 0 30; 280 0 15; 280 0 0; 347 0 70] and the Final Positions equal [0 280 0; 0 280 15; 0 280 30; 0 280 45]. The developed software make this job as shown in Figure (7.27).

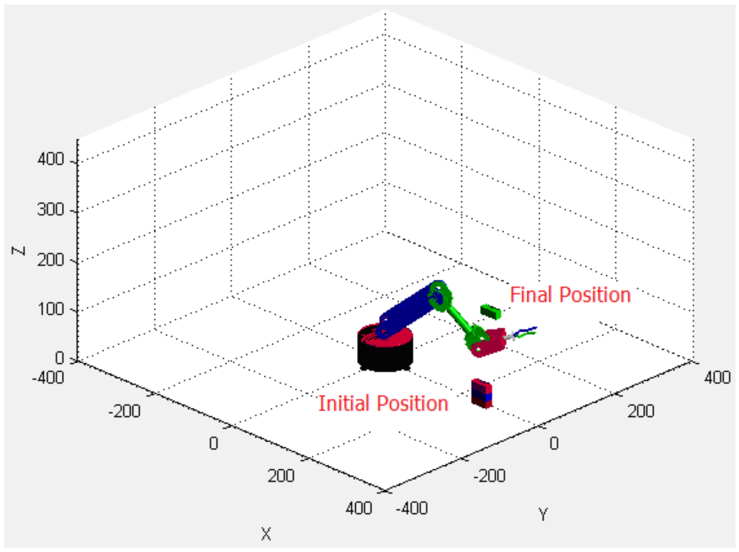


Figure (7.27): Move Blocks from Initial Position to Final Position

CHAPTER 8 CONCLUSION AND RECOMMENDATIONS

This report presented the development of educational software package using MATLAB/Simulink and 3D model. AL5B robot arm was modeled in this research. The complete software life cycle was implemented and validated. A complete mathematical model of AL5B robot is developed including complete Kinematics analyses of the AL5B robot arm. Forward and inverse kinematics equations were derived using Denavit-Hartenberg notation. Velocity Kinematic "Jacobian", and Trajectory Planning solutions were generated and implemented by the developed software. Simulation studies were performed by using MATLAB software's. By using 3D graphics program, structure for the AL5B robot was built which enable the researchers to investigate robot parameters using both forward and inverse kinematics and in turn, this was facilitated the process of designing, constructing and inspecting on the robots in the real world.

The Graphical User Interface (GUI) of the software package was developed for testing motional characteristics of the Robot arm. A physical interface between the AL5B robot arm and the GUI was designed and built. A comparison between kinematics solutions of the virtual arm and the robot's arm physical motional behaviors were been accomplished. The results are displayed in a graphical format and the motion of all joints and end effector can be observed.

The developed system was identified as an educational experimental tool; it can be used in graduate and undergraduate robotic courses to realize the relationships between theoretical and practical aspects of robot manipulator motions in real time. Some of these applications are:

- Forward Kinematics.
- Inverse Kinematics.
- Velocity Kinematic
- Trajectory Planning and Path Planning
- Reedy to Apply Controller Low

Since MATLAB is slow in the execution time, we recommend using a high-level computer programming language to perform the software part.

A future work can be focused on different topics, like the development of different types of controllers be applied on the developed platform then selecting the best control strategy for this type of manipulators. This is to improve the obtained results and to minimize the error between the real arm and the simulated one.

Many future developments can be carried on this robot arm like other types for robots, these developments include path-planning, dynamics modeling, force control and computer vision.

REFERENCES

[KOL 01] E. Kolberg, and N Orlev, “Robotics Learning as a Tool for Integrating Science-Technology curriculum in K-12 Schools,” 31st Annual Frontiers in Education Conference. Impact on Engineering & Science Education. Conference Proceedings, Reno, NV, USA, 2001.

[MIL 00]D.P. Miller and C. Stein,”So That's What Pi is For" and Other Educational Epiphanies from Hands-on Robotics, in Robots for kids: Exploring new technologies for learning experiences, A. Druin, A. & J. Hendler (Eds.) San Francisco, CA: Morgan Kaufmann 2000.

[WED 02] K. Wedeward, and S. Bruder, "Incorporating Robotics into Secondary Education," Robotics Manufacturing Automation and Control. Vol.14.Proceeding of the 5th Biannual World Automation Congress (WAC 2002) ISORA 2002, ISIAC 2002 and ISOMA, Orlando, FL, USA.2002.

[FER 00] N. M. F. Ferreira and J. A. T. Machado, “RobLib: an educational program for robotics,” Symposium on Robot Control (SYROCO 2000), Vienna, Austria, Volume:2, PP 563-568, 2000

[MUR 00] R.R. Murphy, “Robots and Education”, Intelligent. Systems IEEE, Vol. 15, No. 6, pp. 14 -15, 2000.

[MUR 01] R. R. Murphy, “Competing for a Robotics Education”, IEEE Robotics & Automation Magazine, Volume 8, Issue 2, pp. 44-55,2001.

[SUT 00] K. T. Sutherland, “Undergraduate robotics on a shoestring,” IEEE Intelligent Systems, Volume: 15,. Issue: 6, pp. 28-31, 2000

[MAN 96] R. Manseur, "A Software Package For Computer-Aided Robotics Education", pp.1409-1412, 26th Annual Frontiers in Education - Vol 3, 1996

[ROH 00] Martin Rohrmeier, "Web Based Robot Simulation Using VRML", Proceedings of the 2000 Winter Simulation Conference

[SPO 05] Mark W. Spong, Seth Hutchinson, and M. Vidyasagar, *Robot Modeling and Control*, 1st Edition, John Wiley & Sons.2005

[CRA 05] Johan J.Craig, *Introduction to Robotics Mechanics and Control*, 3rd Edition, pp 109-114, Prentice Hall, 2005

[BRA 07] Leniel Braz de Oliveira Macaferi, "Construction and Simulation of a Robot Arm with Opengl", May 16, 2007

[KOY 07] Baki koyuncu, and Mehmet Güzel, "Software Development For the Kinematic Analysis Of A AL5B Robot Arm", pwaset volume 24 october 2007 issn 1307-6884.

[JAM 08] Muhammad Ikhwan Jambak, Habibollah Haron, Dewi Nasien, "Development of Robot Simulation Software for Five Joints Mitsubishi RV-2AJ Robot using MATLAB/Simulink and V-Realm Builder", Fifth International Conference on Computer Graphics, Imaging and Visualization, 2008

[AUN 08] Chun Htoo Aung, Khin Thandar Lwin, and Yin Mon Myint, "Modeling Motion Control System for Motorized Robot Arm using MATLAB", Proceedings Of

World Academy Of Science, Engineering And Technology Vol. 32 August 2008 Issn 2070-3740

[WIR 04] Arya Wirabhuana1, Habibollah bin Haron "Industrial Robot Simulation Software Development Using Virtual Reality Modeling Approach (VRML) and MATLAB- Simulink Toolbox", University Teknologi Malaysia, 2004

[GUR 97] Osman Gürdal, Mehmet Albayrak And Tuncay Aydogan, "Computer Aided Control And Simulation Of Robot Arm Moving In Three Dimension", Electrical & Computer Education Department, Isparta / Turkey, 1997

[PAS 07] Ildiko Paşc, Radu Ţarcă, Florin Popenţiu-Vlădicescu, The VRML Model And Vr Simluation For A Scara Robot, Annals Of The Oradea University, Fascicle Of Management And Technological Engineering, Volume Vi (Xvi), 2007

[BRO 05] Tim Brooks," Teleoperated Robotic Arm with Force Feedback" Institute of Information Science and Technology, Massey University, Private Bag 11222, Palmerston North, New Zealand

[MAR 06] An OpenGL Application for Industrial Robots Simulation, C. Marcu, Gh. Lazea, R. Robotin Technical University of Cluj-Napoca, Romania, 2006 IEEE.

[FER 08] Ferretti Gianni, Magnani Gianantonio, Porrati Paolo, Rizzi Gianpaolo, Rocco Paolo, Rusconi Andrea: "Real-Time Simulation of a Space Robotic Arm", IEEE 2008.

[MIK 02] Mikael Johnsson and Andreas Örmo," Visual Programming Simplified online programming of arc welding robots", Department of Computer Science at Mälardalen University, Västerås, Sweden January 2002.

[MOH 09] Mohammed Abu Qassem, Iyad Abuhadrous, Hatem Elaydi, "Modeling and Simulation of 5 DOF Educational Robot Arm", The 2nd IEEE International Conference on Advanced Computer Control, Shenyang, 20. Dec. 2009.

[PTCDB] Palestine Technical College at Deir El-Balah, Annual Robot Contest, online: http://www.ptcdb.edu.ps/

[LYN 06] AL5B, Lynxmotion, 2006, access date: 1-5-2009, online: http://www.lynxmotion.com/

[COM 05] CB280, COMFILE Technology, 2005, access date: 1-9-2010, online: http://cubloc.com/product/01_01cb280.php

[ULT] UltraArc,http://www.3ds.com/products/delmia, last access: 1-2010.

[ROB] RobotStudio, http://www.microsoft.com/robotics/, last access: 1-2010.

[CIM] CimStation Robotics http://www.acel.us/manufacturing-simulation/e-hub/cimstation-robotics/, last access: 1-2010.

[ROP] ROPSIM, http://www.camelot.dk/, last access: 1-2010.

[ROB] RobotScript, http://www.plantautomation.com,last access: 1-2010.

[DYM] Dymola, *www.dymola.com/*, last access: 1-2010.

[MAT], V-Realm Builder, http://www.mathworks.de/,last access: 1-2010.

[OPE] OpenGL, http://www.opengl.org/, last access: 1-2010.

[POL] Poly Trans, http://www.okino.com/conv/conv.htm, last access: 1-2010.

APPENDIX A: ROBOT DESCRIPTION AND SPECIFICATION

A.1. AL5B Arm Configuration

The AL5B Arm service represents the physical robotic arm as a list of joints, implementing the contract defined in Articulated Arm State. The five joints of the arm are defined in a serial order from the base to the wrist rotate. Each joint has only one degree of "twist" angular freedom and a range from -90 degrees to 90 degrees about the joint axis. The joint axis and joint normal are designated using coordinate system. The dimension and details about AL5B robot arm are shown in Table (8.1). Figure (8.1) shown the AL5B robot arm dimensions

Table (A.1) AL5B Robot Arm Dimension

AL5B Robot Arm	Dimension in (mm)
Base	Diameter 100,hight 50
Shoulder	Diameter 100,hight 30
Upper Arm	Length 127, Thickness 53
Forearm	Length 120, Thickness 43
Wrist	Length 70, Thickness 53
Flange	Length 7, wide 15,height 30
Gripper	Length 51, Gripper Opening 50

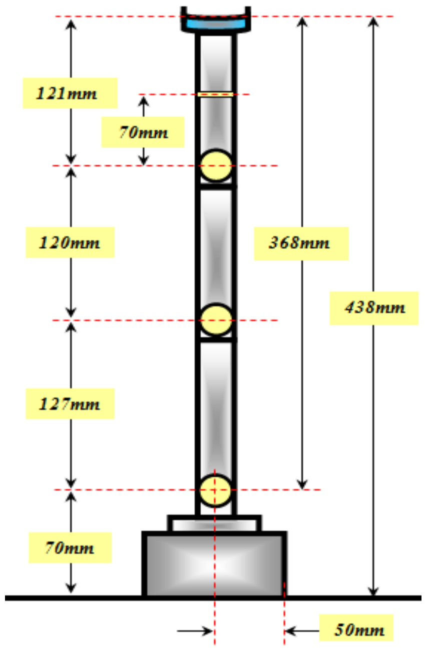


Figure (A.1) AL5B Robot Arm Dimension

A.2. Mechanical System

The mechanical system consists of all non-electrical components of the robotic arm. The system used to reproduce or simulate the mechanics of a human arm. The basic "bone" structure designed and constructed of some type of metal or plastic, and like the human arm, critical joints will connect these "bones". These joints designed to provide both stability and the proper range of motion. A variety of motors used to actuate the assembly at these joints. The mechanical system shown in Figure (8.2) divided into the following components.

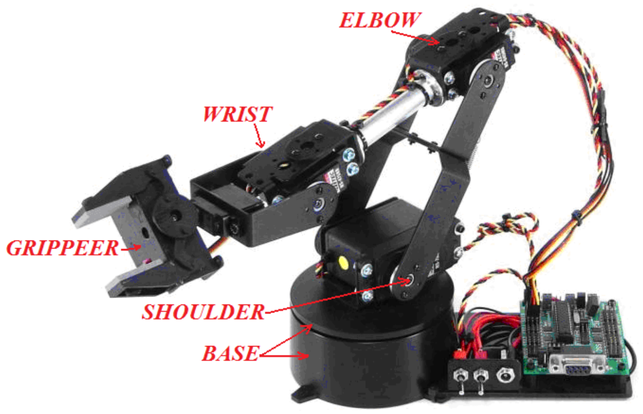


Figure (A.2) Mechanical System

- **Arm Mount**

The first step is to design a base for the arm to be mounted to; the mount will be attached to a solid base that will sit on the ground or a table. The mount must be long enough to provide the desired range of arm motion without making the system unstable. The arm mount and base will house the Robotic Actuator Controller (RAC).

Figure (A.3) Arm Mount

- **Shoulder**

After the base is designed, a shoulder joint will be designed. The shoulder must provide three functions. It must anchor the rest of the arm to the arm mount, provide the desired range of arm motion, and provide a connector for the upper arm. The shoulder will also house the electromechanical assembly used for the upper arm actuation.

Figure (A.4) Shoulder

- **Upper Arm and Forearm**

The upper arm must connect to the shoulder connector and provide a "Forearm" joint for which the lower arm will be connected. The upper arm needs to be stable and long enough to support the range of arm motion. The Forearm must provide a connector for the lower arm as well as holding the electromechanical assembly used for the actuation of the lower arm.

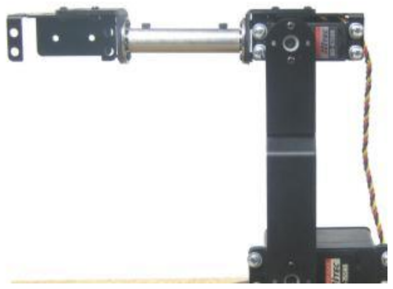


Figure (A.5) Upper Arm and Forearm

- **Wrist and gripper**

The Wrist arm must connect to the Forearm connector and provide a "wrist" joint for the hand to attach to. The Wrist arm must be stable and needs to be long enough to support the arm's desired range of motion. The wrist must provide a connector for the hand and house the electromechanical assembly for the actuation of the gripper.

Figure (A.6) Wrist and gripper

A.3. Drawing Description

The first step in this process is to design the arm in AutoCAD 3D program. The program chosen for this was Autodesk Inventor. Inventor allows the arm to be designed and visualized at the same time. It also allows the arm to be checked for possible collisions and link interference. Because each link depends upon the previous link, the design of the arm needs to begin at the base and finish at the end effector or gripper. Trunk or base is therefore the first to be designed, followed by shoulder, and so on. This means that the design process is fairly involved, as each link has to be redesigned several times. Now we will show links and joint of our AL5B as follow:

- **Trunk**

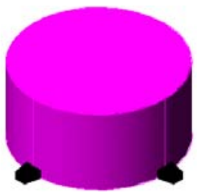


Figure (A.7) Trunk

- **Shoulder**

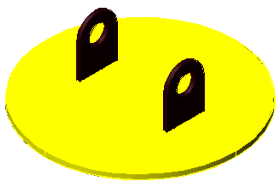


Figure (A.8) Shoulder

- **Upper Arm**

Figure (A.9) Upper Arm

- **Forearm**

Figure (A.10) Forearm

- **Wrist**

Figure (A.11) Wrist

- **Flange**

Figure (A.12) Flange

- **Gripper**

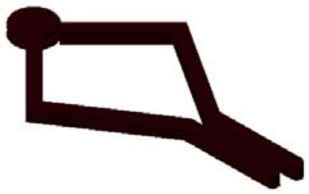


Figure (A.13) Gripper

A.4. Electrical System

The electrical system will consist of all the electrical and computer components of the project. The system will be used to respond to the input stimuli and control the actuation of the robotic arm. The electrical system can be divided into the components discussed below.

- **RC Servo Motor**

The arm uses HS-475HB servo motor in the base, HS-755HB in the shoulder, HS-645MG in the elbow, HS-475HB in the wrist, and HS-422 in the gripper. The wrist rotate uses HS-85BB.

- **CUBLOC Microcontroller**

As shows in Figure (1), the main board has one positive power inputs, in the middle there is a C8280.The main voltage is between 6 V to 12 V. Then we used a voltage regulator to stable at 5 V. Rest switch used to restart the board, com port used to connect the kit with the PC, the I/O port used to connect the main board with the interface kit shown in Figure(2).

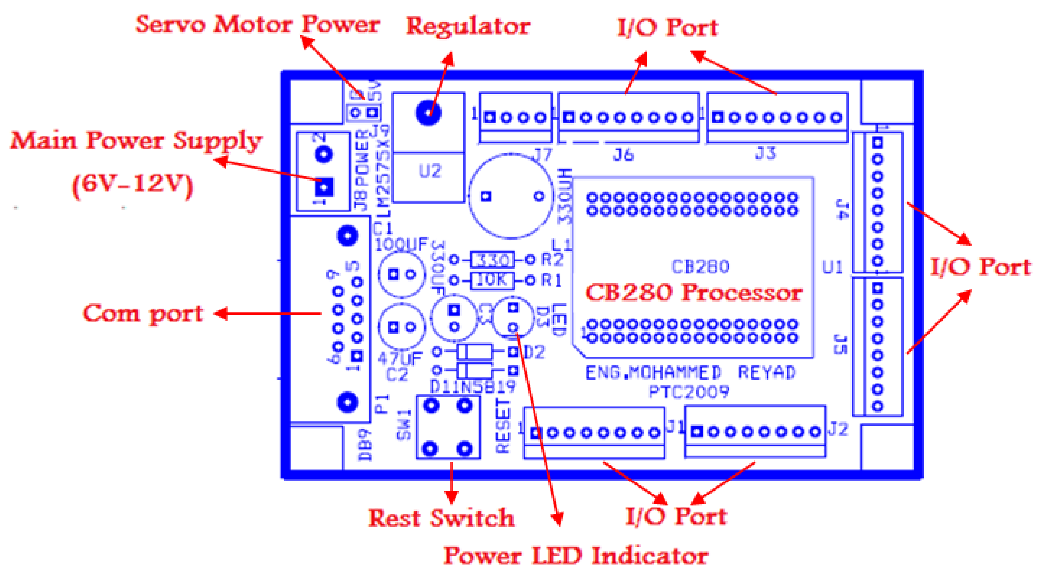


Figure (A.14) MAIN BOARD CUBLOC

Figure (2) shown the interface kit, it consists may port like, ADC port used to read the analogue value of the angle, PWM port used to send the control signal to servo motor,

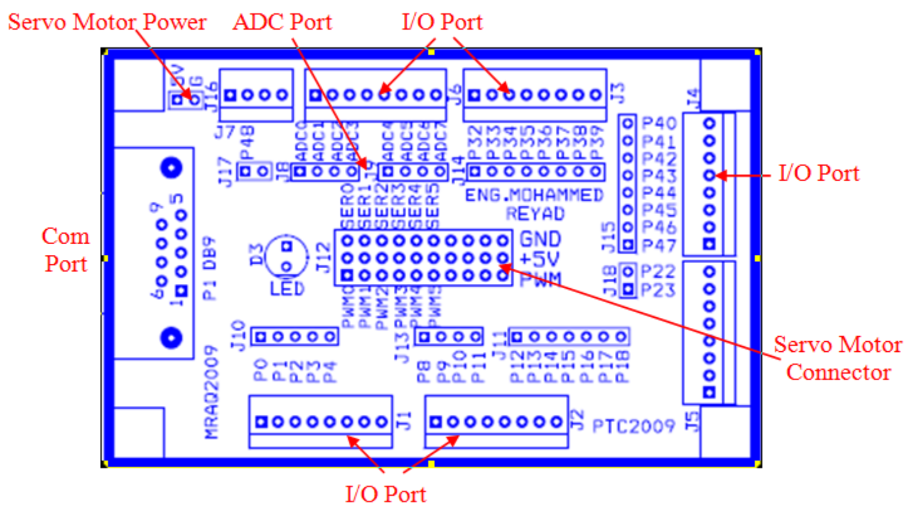


Figure (A.15) Interface Kit

The contact pin of Servo motor has 6 lines, each line owns 3 pins. See the above Figure in blue words; the pins in the most lateral connect to GND. Power shows in the middle. The signal pin lays in the most inner connect to the yellow wires of servo motor. The I/O port connecting with the main board this used to received and send data between the main board and interface board.

- **Wiring Harness**

The wiring harness will connect all of the servo motor to the interface kit. It will be designed so that it can plug in directly to the connectors on the interface kit.

- **Serial Cable**

The CUBLOC and PC communicate through a serial communication link. A variety of serial communication links are currently in use. The CUBLOC uses the RS-232 serial communication. The serial cable, which is used in this thesis, is called the DB-9 serial cable. The cable links a serial, or COM, port on the PC to the CB280 Kit. This allows the user to download a program into the CUBLOC. In addition, this serial connection enables data communication between the CUBLOC and the PC.

A.5. Schematic Diagram

- **CB280 Kit**

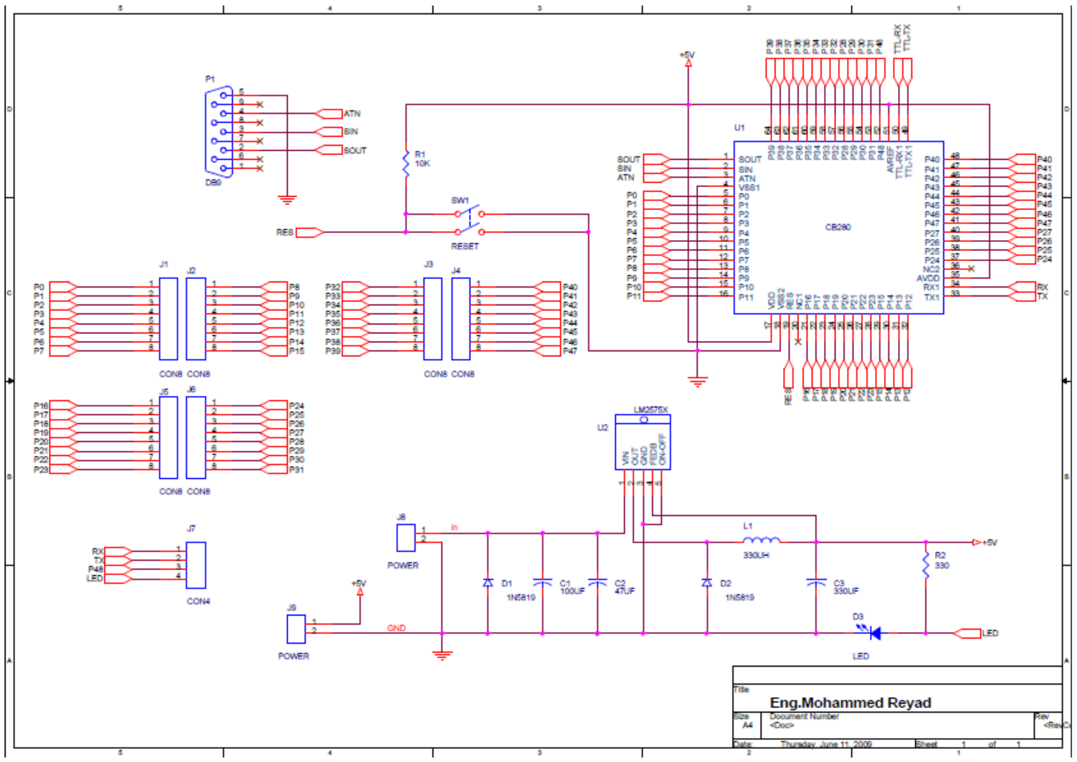


Figure (A.16) CB280 Kit Schematic Diagram

- **Interface kit**

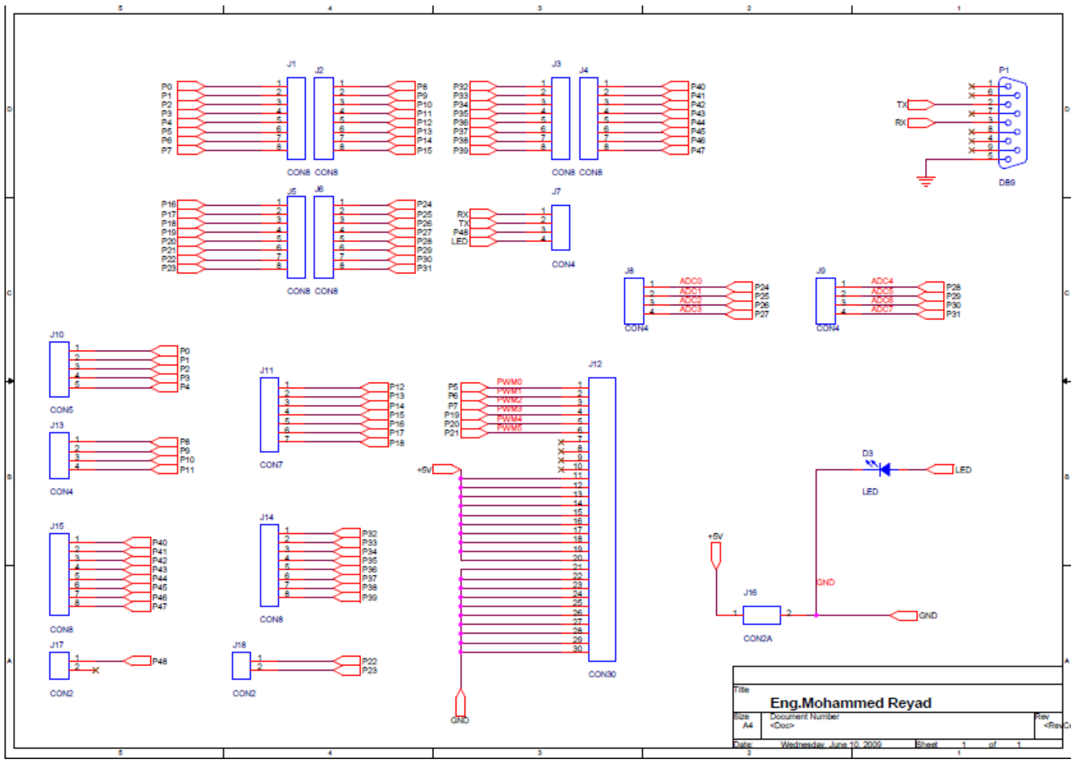


Figure (A.17) CB280 Interface Kit

APPENDIX B: ECONOMIC COST AND MATLAB FUNCTION

Economic Cost

Economic cost of the project can be separated in two groups. First of them is software cost. Another one is hardware cost. Table (B.1) shows software cost and hardware cost.

Table (B.1) Software Cost And Hardware Cost.

Hardware	**Price ($)**	**Software**	**Price ($)**
AL5B Robot Arm	1000$	Windows Operating System	100
CB280 Microcontroller	150	MATLAB 7.1 or higher	150
Microcontroller interface Kit	100	CublocStudio	free
Computer	500	AL5B Robot Arm Software	free
Total	**1750**		**250**

MATLAB function description

MATLAB File	**Description**
LynxRobot001.m	Main MATLAB file
LynxRobot001.fig	Main GUI file
AllLinks.mat	3D Drawing Link
forkin.m	Forward Kinematic Function
invkin.m	Inverse Kinematic Function
cubic.m	Cubic Trajectory Polynomial Function
quintic.m	Quintic Trajectory Polynomial Function
LSPB.m	(Linear Segment Parabolic Blend) trajectory
LynxJacman.m	AL5B Jacobian Function
build_block.m	Pick and Place Function
staticForce.m	Relation Between Torque and Force Function
DH_axis.m	Homogeneous Transform that Describes the End-Effector Frame
plotframe.m	Plot Coordinate Frame
tmat.m	Homogeneous Transformation Function
rotx.m	Rotation about x Function
roty.m	Rotation about y Function
rotz.m	Rotation about z Function