Sudan University of Science and Technology

College of Graduated Studies

M.sc. Program in Mechatronics Engineering

Modelling and Simulation of KUKA KR6 Robot Manipulator

محاكاة ونمذجة الذراع الألي (KUKA KR6)

A Thesis Submitted in Partial fulfillment for the Requirements of the Degree of M.Sc. in Mechatronics Engineering

Prepared by:

NASR ABDO ALI MOHAMMED

Supervised by:

Dr. MOHAMMED EL-NOUR ABDALLA

April 2017

الأية

قال الله تعالى :

﴿ ذَٰلِكَ ٱلْكِتَٰبُ لَا رَيْبَ ۛ فِيهِ ۛ هُدًى لِّلْمُتَّقِينَ ۝٢ ﴾

سورة البقرة

Dedication

All praise to Allah, today we fold the days' tiredness and the errand summing up between the cover of this humble work.

To the Spring that never stops giving, to my mother who weaves my happiness with strings from her merciful heart... to my mother.

To whom he strives to bless comfort and welfare and never stints what he owns to push me in the success way who taught me to promote life stairs wisely and patiently, to my dearest father

To whose love flows in my veins, and my heart always remembers them, to my brothers and sisters.

To those who taught us letters of gold and words of jewel of the utmost and sweetest sentences in the whole knowledge. Who reworded to us their knowledge simply and from their thoughts made a lighthouse guides us through the knowledge and success path, To our honored teachers and professors.

Acknowledgement

I would like to reflect on the people who have supported and helped me so much throughout period of search.

Special thanks goes to my search supervisor, Dr. MOHAMMED EL-NOUR. I have learned many things since I became his student. He spends very much time instructing me how to write a paper, how to search literature and how collect data.

ABSTRACT

This research about modeling and simulation of 6-DOF manipulator robots and the KUKA KR 6 manipulator robot have been selected as a target robot for study. The study performed using theoretical analysis including mathematical modeling of the robot based on transformation matrix and geometry approach for kinematic analysis and based on Euler-Lagrange equations and recursive Newton-Euler for dynamic analysis, then the modeling implemented by ARTE toolbox that work under MATLAB program. These analyses are important for manipulator robots where the execution of the specific task requires the manipulator to follow a preplanned path, which the practical geometric parameters, physical characteristics and restriction relations are adopted to establish robots' dynamics and kinematics model. The robot model has been developed as accurate as the real one by implementing dynamic model of robot in ARTE toolbox linking with MATLAB for motion studies. 3D Simulation experiments show that the modeling method is efficient and it provides an effective platform for researching on the assistant robot system.

المستخلص

يهدف البحث إلى دراسة نمذجة ومحاكاة روبوت متعدد الاستخدام ولديه القدرة على التحرك في ستة محاور (6-DOF) حيث تم اختيار الروبوت (KUKA KR6) كنموذج للبحث. تم نمذجة الروبوت بواسطة التحليل الرياضي الذي اعتمد على كل من نظرية المصفوفات الإنتقالية ومنهجية التحليل الهندسي لإيجاد معادلات الحركة الكيناماتيكية بينما استخدمت كل من نظرية (اولير-لاجرانج) و (ريكريسف نيوتن-اويلر) لإيجاد معادلة الحركة الديناميكية للروبوت. ومن ثم تم تنفيذ محاكاة الروبوت باستخدام ماتلاب تولبوكس، هذه النمذجة والمحاكاة للروبوت مهمة حيث أن تنفيذ الروبوت لأي مهمة يتطلب تحرك الروبوت في مسار معد ومخطط له مسبقا بحيث أن العوامل الهندسية لشكل الروبوت والخواص الفيزيائية للروبوت والعوامل الاخرى المقيدة لحركة الروبوت يجب أن تُكيف بحيث يتم التناسق في الحركة تبعا لمعادلات الحركة الديناميكية والكيناماتيكية. وللحصول على محاكاة دقيقة تم استخدام أداة جديدة (ARTE Toolbox) صممت بشكل خاص لدراسة وتحليل الأنظمة الروبوتية. توصلت الدراسة أن المحاكاة ثلاثية الأبعاد باستخدام (ARTE Toolbox) ذات كفاءة عالية وذات إمكانية تساعد الباحثين والمهتمين في هذا المجال بالإضافة أنها تفتح الباب أمام المزيد من الأبحاث المستقبلية في هذا المجال.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

ARTE	-	A Robotics Toolbox for Education
DGM	-	Direct Geometric Model
DH	-	Denavit-Hartenberg
DOF	-	Degree-Of-Freedom
FMS	-	Flexible Manipulator Systems
GUI	-	Graphical User Interface
KUKA	-	Keller und Knappich Augsburg
UMH	-	Miguel Hernández University of Elche
3D	-	Three Dimension

List of Symbols

D,d	-	Diameter
F	-	Force
G	-	Gravity = 9.81 m/s
I	-	Moment of inersia
l	-	Length
m	-	Mass
N	-	Rotational Velocity
Q,q	-	Volumetric Flow-Rate
r	-	Radius
T	-	Torque
V	-	Velocity
x	-	Displacement
z	-	Height
θ	-	Angle

Chapter One

Introduction

Chapter One

Introduction

1.1 Preface

The ever-increasing utilization of robotic manipulators in various applications in recent years has been motivated by the requirements and demands of industrial automation. Among the rigid and flexible manipulator types, attention is focused more towards flexible manipulators. This is owing to various advantages such manipulators offer as compared to their rigid counterparts. Exploitation of the potential benefits and capabilities of rigid and flexible manipulators introduces a further emerging line of research in which hybrid rigid–flexible manipulator structures are considered.

Flexural dynamics (vibration) in flexible manipulators has been the main research challenge in the modeling and control of such systems. Accordingly, research activities in flexible manipulators have looked into the development of methodologies to cope with the flexural motion dynamics of such systems.

A considerable amount of research on the development of dynamic models of flexible manipulators has been carried out. These have led to descriptions in the form of either partial differential equations, or finite-dimensional ordinary differential equations. From a control perspective, an input/output characterization of the system is desired, which can be obtained through suitable online estimation and adaptation mechanisms. Given the dynamic nature of flexible manipulator systems, the practical realization of such methodologies presents new challenges.

The motion of a mechanical system is related via a set of dynamic equations to the forces and torques the system is subject to. In this work, we will be primarily interested in robots consisting of a collection of rigid links connected through joints that constrain the relative motion between the links[1].

1.2 Problem Statement:

The purpose of this research is to present an analytical method and a geometrical approach for solving forward and inverse kinematics and dynamics problem of a particular six degree-of freedom serial manipulator, accordingly. Where kinematics analysis requires deriving the formulation that shows the relation between angles of robot joints and the position of the robot end-effector. Whereas the dynamics analysis requires deriving equations that explain the relations between the forces effecting on robot and the motions parameters such as velocity and acceleration. The target robot is a KR 6 KUKA which is an industrial manipulator production of KUKA corporation.

1.3 Proposed Solution:

This work will present the study and modelling of KR 6 KUKA Robot, of the Robotics and shows the MATLAB model (Computer Design), the direct kinematics, the inverse kinematics and the inverse dynamical model. The direct kinematic is based in the use of homogeneous matrix. The inverse kinematics uses the geometric approach model. The dynamical model is based on the use of Euler-Lagrange equations and recursive Newton-Euler.

1.4 Research Aim and Objectives:

This research will present the modeling of KUKA KR 6 Robot:

1. Robot modeling.
2. MATLAB simulation.

1.5 Research Methodology:

search supposed to contain two main parts:

first part: modeling and mathematical analysis:

There are two main formalisms for deriving the dynamic equations for such mechanical systems: Newton-Euler equations that are directly based on Newton’s laws and Euler-Lagrange equations that have their root in the classical work of Alembert and Lagrange on analytical mechanics and the work of Euler and Hamilton on vibrational calculus.

second part: MATLAB toolbox and Simulink analysis:

In this part of search where we can use MATLAB graphical user interface (GUI) aspects to build 3D model for robot and make motion control for that model.

1.6 Research layout:

Chapter two: This chapter presents basic concepts of manipulator robotic modeling.

Chapter three: This chapter studies the principles of kinematics and dynamics of KUKA KR6 robot.

Chapter four: This chapter shows the MATLAB simulation for KUKA KR6 robot and analysis its results.

Chapter five: This chapter shows the results that introduced from simulation and its discussions.

Chapter six: This chapter concludes the research and presents recommendations for future work.

Chapter Two

Fundamentals of Manipulator Robots Modeling

Chapter Two

Fundamentals of Manipulator Robots Modeling

2.1 Background:

Research on flexible manipulator systems (FMS) ranges from a single-link manipulator rotating about a fixed axis to three-dimensional multi-link arms. However, experimental work, in general, is almost exclusively limited to single-link manipulators. This is because of the complexity of multi-link manipulator systems, resulting from more degrees of freedom and the increased interactions between gross and deformed motions. It is important for control purposes to recognize the flexible nature of the manipulator system and to build a suitable mathematical framework for modelling of the system. FMSs offer several advantages in contrast to their traditional rigid counterparts. These include faster system response, lower energy consumption, the requirement of relatively smaller actuators, reduced non-linearity owing to elimination of gearing, lower overall mass and, in general, lower overall cost. However, owing to the distributed nature of the governing equations describing dynamics of such systems, the control of flexible manipulators has traditionally involved complex processes. Moreover, to compensate for flexural effects and thus yield robust control the design focuses primarily on non-collocated controllers[1].

2.2 Modeling and Identification of Serial Robots:

The design and control of robots require certain mathematical models, such as:

- transformation models between the operational space (in which the position of the end-effector is defined) and the joint space (in which the configuration of the robot is defined). The following is distinguished:

- direct and inverse geometric models giving the location of the end-effector
  (or the tool) in terms of the joint coordinates of the mechanism and vice versa.
- direct and inverse kinematic models giving the velocity of the end-effector in terms of the joint velocities and vice versa.
- dynamic models giving the relations between the torques or forces of the actuators, and the positions, velocities and accelerations of the joints[2].

2.3 Space Movement Representation:

For the representation of space movements there are several methods such as rotation matrix, vectors, quaternions, roll pitch and yaw, Euler angles, homogenous matrix. The selected method used for the developing of the direct kinematic model in this work is the homogeneous matrix.

2.4 Homogeneous Matrix:

Homogeneous matrices are 4×4 matrixes, which can represent rotations, translations, scales and perspectives. In general, the homogeneous matrices represent linear transformations. The general form is presented in equation:

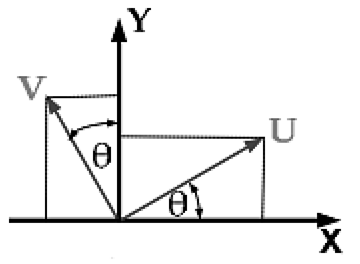
$$A = \begin{bmatrix} [R(3 \times 3)] & [T(3 \times 1)] \\ [P(3 \times 3)] & [E(1 \times 1)] \end{bmatrix} \tag{2.1}$$

$R(3 \times 3)$ Corresponds to a matrix of three rows by three columns representing rotations.

$T(3 \times 1)$ Corresponds to an array of three rows by a column that represents translation.

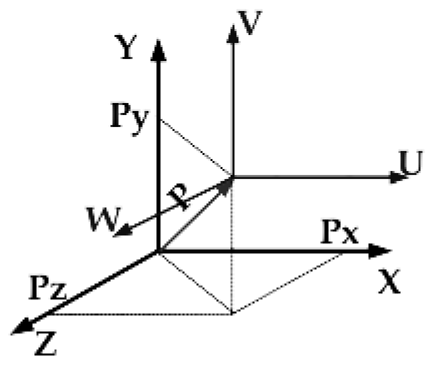
$P(3 \times 3)$ Represents a vector of a row of three columns representing the perspective.

$E(1 \times 1)$ Corresponds to a scalar that represents the scale of the transformation. For this case $\vec{P} = \vec{0}$ and E = 1 the principal homogeneous matrix Rotation around the Z axis as:



$$\begin{bmatrix} Cos(\theta) & -Sin(\theta) & 0 & 0 \\ Sin(\theta) & Cos(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Figure 2.1a: Rotation around to axis Z.



$$\begin{bmatrix} 1 & 0 & 0 & P_x \\ 0 & 1 & 0 & P_y \\ 0 & 0 & 1 & P_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Figure 2.1b: Translation Px, Py, Pz.

The movements in the space are represented by a series of rotations and translations, these rotations and translations figures (2.1a&b), can be represented as a homogeneous matrix multiplication.

2.5 Direct Kinematics:

Direct kinematics refers to the use of the kinematic equations of a robot to compute the position of the end-effector from specified values for the joint parameters[3]. In this model, the movements of the robot (coordinates of degrees of freedom) are given and the final positions are found. See Figure (2.2).

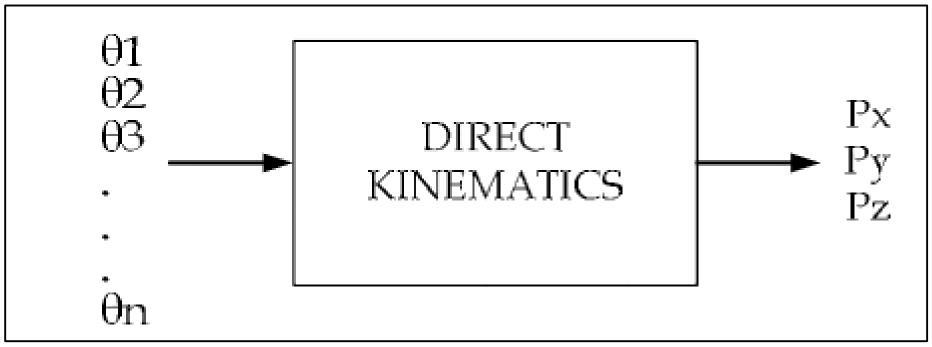


Figure 2.2: direct kinematics.

To find the direct kinematic model, using the homogeneous matrix method, is necessary to make the moves of coordinated system from the fixed base until the last link. For each movement, homogeneous matrices are obtained and the final result is the product of these matrices[4].

2.6 Geometric Modeling:

A systematic and automatic modeling of robots requires an appropriate method for the description of their morphology. Several methods and notations have been proposed. The most widely used one is that of Denavit-Hartenberg. However, this method, developed for simple open structures, presents ambiguities when it is applied to closed or tree-structured robots. Hence, the notation of Khalil and Kleinfinger enables the unified description of complex and serial structures of articulated mechanical systems[2].

2.7 Denavit-Hartenberg (DH) Convention:

The DH parameters were originally proposed by Denavit and Hartenberg (1955) and widely used to define links' configuration of a robotic manipulator consisting of one degree-of-freedom (DOF) joints, i.e., revolute or prismatic. Later, Khalil and Kleinfinger (1986) showed that the DH parameters are powerful tool for serial robots, but, lead to ambiguities in the case of closed and tree structured robots. They presented the modified DH parameter from its original definition. Craig (1991) also used modified DH notation for serial robots[5].

In order to compute the direct kinematics equation for an open-chain manipulator general method is to be derived to define the relative position and orientation of two consecutive links; the problem is that to determine two frames attached to the two links and compute the coordinate transformations between them. In general, the frames can be arbitrarily chosen as long as they are attached to the link they are referred to. Nevertheless, it is convenient to set some rules also for the definition of the link frames[6].

A robot manipulator consists of several links connected by, usually, single degree of freedom joints, say, a revolute or a prismatic joint. In order to control the end-effector with respect to the base, it is necessary to find the relation between the coordinate frames attached to the end-effector and the base[5].

2.8 Identification of Denavit-Hartenberg Parameters of an Industrial Robot:

The travel from the base frame to the end-effector frame is achieved by moving across two consecutive frames placed at the joints. The set of four parameters relates the transformation between Frame i to Frame i+1 by b_i, θ_i, a_i and α_i, as shown in figure (2.3) and figure (2.4) and the parameters defined as in table (2.1) [7].

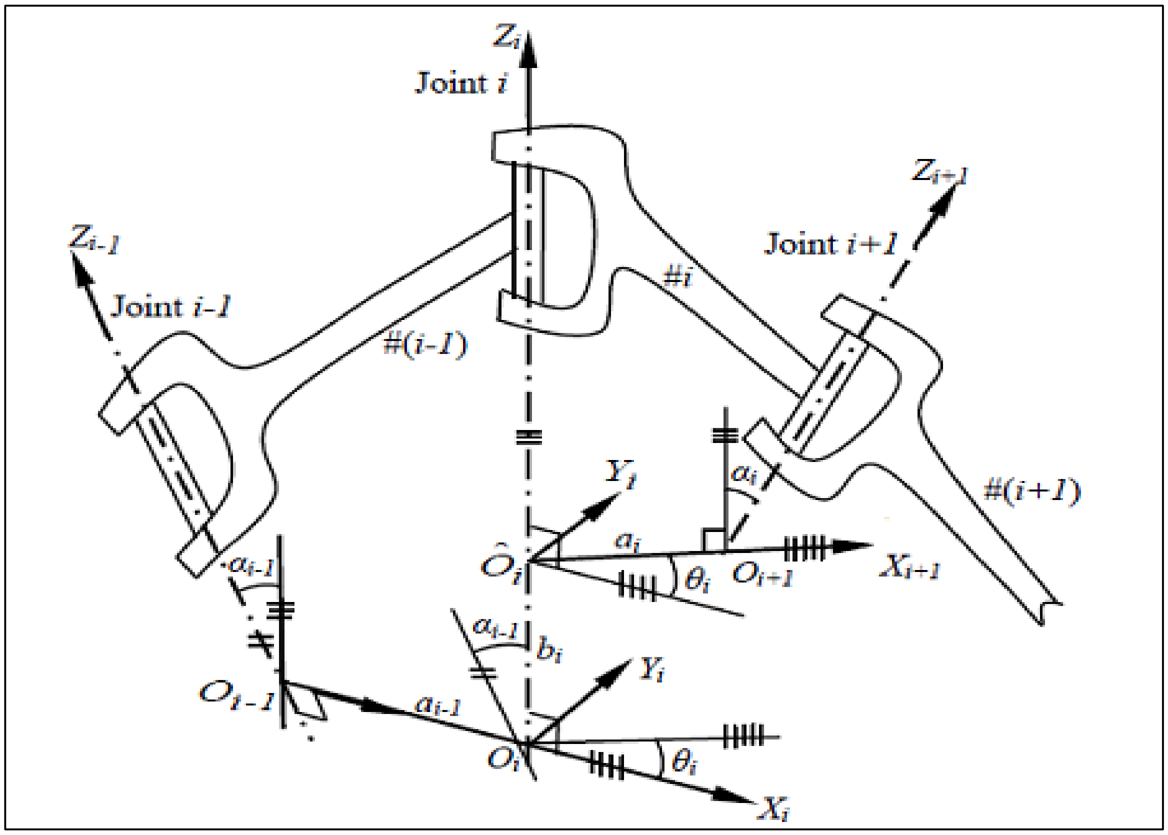


Figure 2.3: DH parameters and Frames attached.

Table 2.1: Symbolic notation used to describe the DH parameters with its definition[7].

Symbol	Name	Description
b_i	Joint offset	$X_i \xrightarrow[@Z_i]{\perp,\text{distance}} X_{i+1}$
θ_i	Joint Angle	$X_i \xrightarrow[@Z_i]{\text{rotation, ccw}} X_{i+1}$
a_i	Link Length	$Z_i \xrightarrow[@X_{i+1}]{\perp,\text{distance}} Z_{i+1}$
α_i	Twist Angle	$Z_i \xrightarrow[@X_{i+1}]{\text{rotation, ccw}} Z_{i+1}$

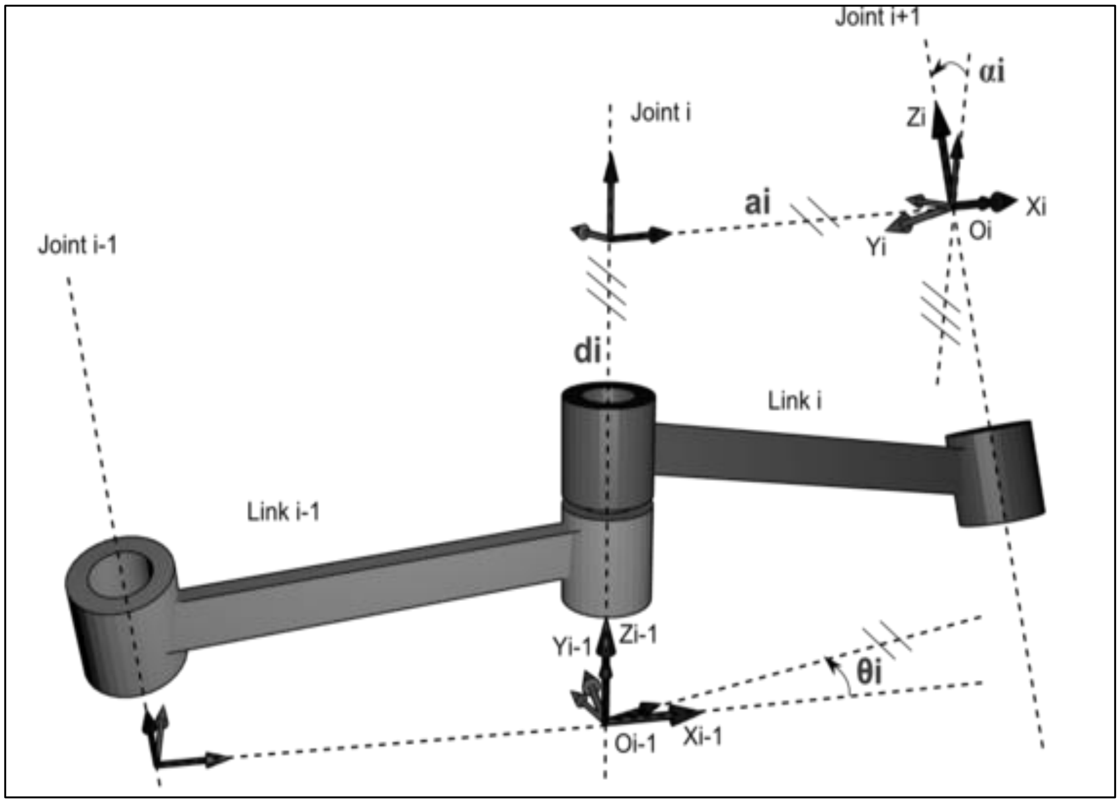


Figure 2.4: Frame convention for modified DH parameters.

The transformation matrix defining the frame i the frame i+1 is obtained from figure (2.4):

$$^{n-1}T_n = Rot_{x_{n-1}}(\alpha_{n-1}).Trans_{Z_{n-1}}(\alpha_{n-1}).Rot_{Z_n}(\theta_n).Trans_{Z_n}(d_n) \quad (2.2)$$

$$^{n-1}T_n = \begin{bmatrix} C\theta_{n+1} & -S\theta_{n+1}C\alpha_{n+1} & S\theta_{n+1}S\alpha_{n+1} & \alpha_{n+1}C\theta_{n+1} \\ S\theta_{n+1} & C\theta_{n+1}C\alpha_{n+1} & -C\theta_{n+1}S\alpha_{n+1} & \alpha_{n+1}S\theta_{n+1} \\ 0 & S\alpha_{n+1} & C\alpha_{n+1} & d_{n+1} \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.3)$$

2.9 Inverse kinematics:

Inverse kinematics refers to the use of the kinematics equations of a robot to determine the joint parameters that provide a desired position of the end-effector figure (2.5). Specification of the movement of a robot so that its end-effector achieves a desired task is known as motion planning.

Inverse kinematics transforms the motion plan into joint actuator trajectories for the robot. The inverse kinematics seeks the coordinates of each degree of freedom based on the final position of the robot[3].

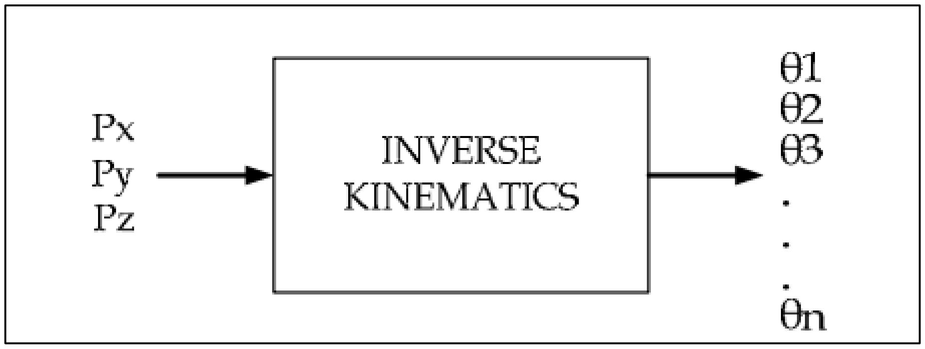


Figure 2.5: Inverse kinematics.

There are two approaches to solve the inverse kinematics problem of a robot manipulator; mathematical or algebraic and geometrical. The higher degrees of freedom requires the more complicated algebraic solution[8]. Therefore, this section has been devoted to present a geometrical solution for the inverse kinematics problem of a KUKA KR6.

2.10 Dynamic Modeling:

Dynamic modeling means deriving equations that explicitly describes the relationship between force and motion. These equations are important to consider in simulation of robot motion, and in design of control algorithms[9].

During the work cycle a manipulator must accelerate, move at constant speed and decelerate. This time-varying position and orientation of manipulator is termed as its dynamic behavior. Time-varying torques are applied at the joints to balance out the internal and external forces[10].

The internal forces are cause by motion of link. Inertial, Coriolis, and frictional forces are some of the internal forces. The external forces are the forces exerted by the environment. These include load and gravitational forces[10].

The real problem in robot dynamics is a practical one, namely, that of finding formulations for the equations of motion that lead to efficient computational algorithms. To derive these equations, we can use well established procedures from classical mechanics such as those based on the equations of Newton-Euler, Euler-Lagrange[11].

2.11 Newton-Euler and Euler-Lagrange formulations:

In the Newton-Euler approach, the derivation of the equations of motion is based on direct application of Newton's and Euler's laws, while in the Lagrangian approach, the equations of motion are derived from two scalar quantities, namely, the kinetic and potential energy[11]. The resulting dynamic model is the same for both methods and can be written in matrix form as[9].

$$T(t) = D\big(\theta(t)\big)\ddot{\theta}(t) + h\left(\theta(t), \dot{\theta}(t)\right) + c\big(\theta(t)\big) \tag{2.4}$$

Were:

$T(t)$ =Vector torque.

$\theta(t)$ = Vector of joint positions.

$\dot{\theta}(t)$ = Vector of angular Velocities.

$\ddot{\theta}(t)$ = Vector of angular acceleration.

D = Inertia matrix.

$h\big(\theta(t), \dot{\theta}(t)\big)$ = Vector of Coriolis and centrifugal force.

$c(\theta)$ = Gravity forces vector.

Chapter Three

Kinematics and Dynamics of KUKA KR6 Robot

Chapter Three

Kinematics and Dynamics of KUKA KR6 Robot

3.1 KUKA KR6 robot specifications:

KUKA KR6 is an industrial robot designed by links which are connected to each other by six revolute joints. All the joints of this robot are the same and there is no prismatic, cylindrical, planar or any other type of joint in the structure of the robot, the table (3.1) shows the robot specifications.

Table 3.1: The KUKA KR6- robot specifications.

KUKA KR6-2 robot specifications	
Payload	6 kg
Total load	16 kg
Maximum reach	1611 mm
Number of controlled axes	6
Position repeatability	±0,05 mm
Weight	235 kg
Mounting positions	Floor
Ambient temperature	0 °C bis + 0 °C
Controller	KR C4
Protection class	IP 65
Protection class inline wrist	IP 65

Pictures and dimension of the robot is appended in appendix A

3.2 Kinematics Model of KUKA KR6:

The manipulator kinematics model is based on the use of homogeneous matrix for this purpose; coordinated systems are located in a convention proposed by the authors. Supported by recommendations of the Denavit-Hartenberg algorithm[4].

3.3 The Direct Geometric Model:

The direct geometric model (DGM) is the set of relations which express the position of the end-effector, i.e. operational coordinates of the robot, according to its joint coordinates. In the case of a simple open-chain, it can be represented by the transformation matrix ^0T_k.

$$^0T_k = {}^0T_1(q_1)\,{}^0T_2(q_2)\ldots{}^{n-1}T_k(q_n) \tag{3.1}$$

q being the vector of joint coordinates

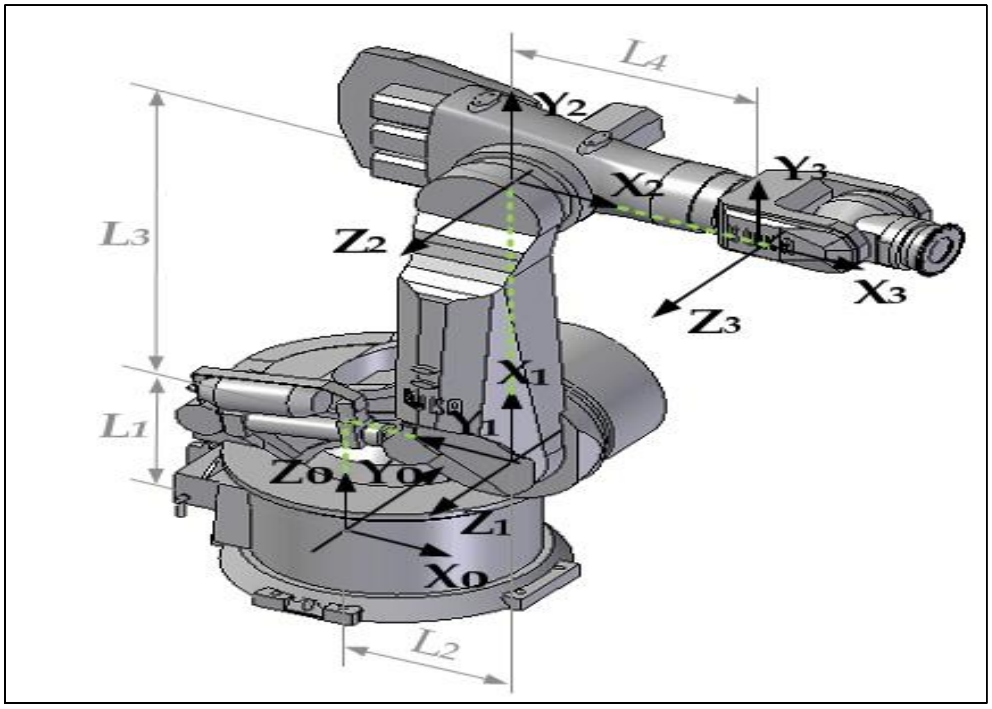


Figure 3.1: Reference coordinate systems, for the KR6 KUKA Robot.

The generated movements for going from one frame to another are mathematically represented by homogeneous matrix transformations and follow the particular geometry of the robot link to link as in figure (3.1)[4]:

1. $\mathrm{R}(\mathrm{Zo}, \theta\mathrm{o}) * \mathrm{T}(\mathrm{Zo}, \mathrm{L1})$ (3.2)

2. $\mathrm{T}(\mathrm{Xo}', \mathrm{L2}) * \mathrm{R}\left(\mathrm{Xo}', \frac{\pi}{2}\right) * \mathrm{R}\left(\mathrm{Zo}', \frac{\pi}{2}\right)$ (3.3)

3. $\mathrm{R}(\mathrm{Z1}, \theta1) * \mathrm{T}(\mathrm{X1}, \mathrm{L3}) * \mathrm{R}\left(\mathrm{Z1}', -\frac{\pi}{2}\right)$ (3.4)

4. $\mathrm{R}(\mathrm{Z2}, \theta2) * \mathrm{T}(\mathrm{X2}, \mathrm{L4})$ (3.5)

The full kinematic model is presented in equation:

$$T = \mathrm{R}(\mathrm{Zo}, \theta\mathrm{o}) \times \mathrm{T}(\mathrm{Zo}, \mathrm{L1}) \times \mathrm{T}(\mathrm{Xo}', \mathrm{L2}) \times \mathrm{R}\left(\mathrm{Xo}', \frac{\pi}{2}\right) \times \mathrm{R}\left(\mathrm{Zo}', \frac{\pi}{2}\right) \times \mathrm{R}(\mathrm{Z1}, \theta1) \times \mathrm{T}(\mathrm{X1}, \mathrm{L3}) \times \mathrm{R}\left(\mathrm{Z1}', -\frac{\pi}{2}\right) \times \mathrm{R}(\mathrm{Z2}, \theta2) \times \mathrm{T}(\mathrm{X2}, \mathrm{L4}) \quad (3.6)$$

3.4 The D-H Parameters of KUKA KR 6 Robot:

Denavit and Hartenberg put forwards to a matrix method to build the attached coordinate system on each link in the joint chains of the robot to describe the relationship of translation or rotation between the contiguous links way back in 1955. This robot kinematic model is based on the D-H Coordination system figure (3.2) and table (3.2) shows D-H Parameters for KUKA KR 6 robot.

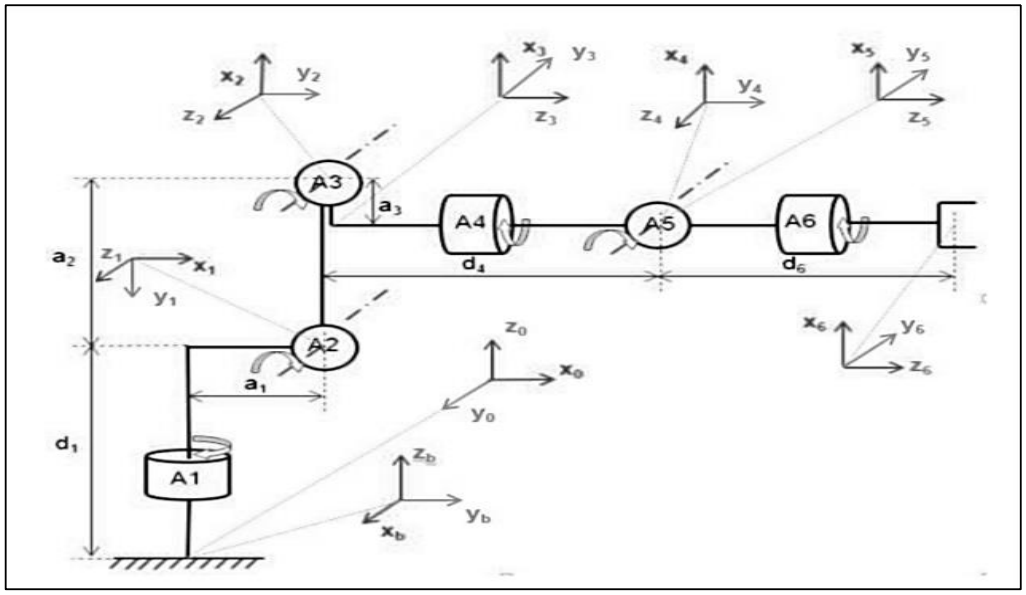


Figure 3.2: The joints of the robot with coordinate system following the DH-convention.

Table 3.2: D-H Parameters for KUKA KR 6 robot.

Joint Number	Joint angle (θ_i)deg	Joint offset (d_i)m	Link length (a_i)m	Twist angle (α_i)deg
1	θ1	d1	a1	-90
2	θ2	0	a2	0
3	θ3	0	a3	90
4	θ4	d4	0	-90
5	θ5	0	0	90
6	θ6	d6	0	180

The transformations between each two successive joints can be written by simply substituting the parameters from the parameters table into the T matrix. The transform matrix is given by the following order of operations[10]:

$$^{n-1}T_n = Rot_{x_{n-1}}(\alpha_{n-1}).Trans_{Z_{n-1}}(\alpha_{n-1}).Rot_{Z_n}(\theta_n).Trans_{Z_n}(d_n) \quad (3.7)$$

Thus, the matrix of the modified DH parameters becomes**:**

$$^{n-1}T_n = \begin{bmatrix} C\theta_{n+1} & -S\theta_{n+1}C\alpha_{n+1} & S\theta_{n+1}S\alpha_{n+1} & \alpha_{n+1}C\theta_{n+1} \\ S\theta_{n+1} & C\theta_{n+1}C\alpha_{n+1} & -C\theta_{n+1}S\alpha_{n+1} & \alpha_{n+1}S\theta_{n+1} \\ 0 & S\alpha_{n+1} & C\alpha_{n+1} & d_{n+1} \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.8)$$

Where (C = cosine, S = sine)

The transformation matrices are computed in the following:

$$^0T_1 = \begin{bmatrix} C\theta1 & 0 & -S\theta1 & a1C\theta1 \\ S\theta1 & 0 & -C\theta1 & a1S\theta1 \\ 0 & -1 & 0 & d1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.9)$$

$$^1T_2 = \begin{bmatrix} C\theta2 & -S\theta2 & 0 & a2C\theta2 \\ S\theta2 & C\theta2 & 0 & a2S\theta2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.10)$$

$$^2T_3 = \begin{bmatrix} C\theta3 & 0 & S\theta3 & a3C\theta3 \\ S\theta3 & 0 & -C\theta3 & a3S\theta3 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.11)$$

$$^3T_4 = \begin{bmatrix} C\theta4 & 0 & -S\theta4 & 0 \\ S\theta4 & 0 & C\theta4 & 0 \\ 0 & -1 & 0 & d4 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.12)$$

$$^{4}T_{5}=\begin{bmatrix} C\theta5 & 0 & S\theta5 & 0\\ S\theta5 & 0 & -C\theta5 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\end{bmatrix} \tag{3.13}$$

$$^{5}T_{6}=\begin{bmatrix} C\theta6 & -S\theta6 & 0 & 0\\ S\theta6 & C\theta6 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix} \tag{3.14}$$

The total transformation between the base of the robot and the hand is:

$$^{R}T_{H}=\mathrm{T1.T2.T3.T4.T5.T6} \tag{3.15}$$

$$^{R}T_{H}=\mathrm{T1.T2.T3.T4.T5.T6}=\begin{bmatrix} S_x & n_x & a_x & P_x\\ S_y & n_y & a_y & P_y\\ S_z & n_z & a_z & P_z\\ 0 & 0 & 0 & 1\end{bmatrix} \tag{3.16}$$

After calculation and identification of the terms of two matrices then we can find the values of P_x, P_y & P_z let,

$$^{0}T_{6}=\begin{bmatrix} r_{11} & r_{12} & r_{13} & r_{14}\\ r_{21} & r_{22} & r_{23} & r_{24}\\ r_{31} & r_{32} & r_{33} & r_{34}\\ 0 & 0 & 0 & 1\end{bmatrix} \tag{3.17}$$

The position and the orientation of the end effector in roll-pitch-yaw representation are as follows[8]:

$$^{0}P_{6}=\begin{bmatrix} r_{14}\\ r_{24}\\ r_{34}\end{bmatrix} \tag{3.17}$$

$$pitch = Atan2\left(r_{13}, \sqrt{r_{23}^{2}+r_{33}^{2}}\right) \tag{3.18}$$

$$roll = \begin{cases} 0 & pith = \frac{\pi}{2}, -\frac{\pi}{2} \\ Atan2\left(-\frac{r_{12}}{\cos(pitch)}, \frac{r_{11}}{\cos(pitch)}\right) & o.w \end{cases} \tag{3.19}$$

$$yaw = \begin{cases} Atan2(r_{32}, r_{22}) & pitch = \frac{\pi}{2} \\ -Atan2(r_{32}, r_{22}) & pitch = -\frac{\pi}{2} \\ Atan2\left(-\frac{r_{12}}{\cos(pitch)}, \frac{r_{11}}{\cos(pitch)}\right) & o.w \end{cases} \tag{3.20}$$

Where Atan = $\tan^{-1}$

3.5 Inverse Kinematic Model of KUKA KR 6 Robot:

Transformation matrix Equation will be used to calculate inverse kinematics equations. Its solution, however, is much more complex than direct kinematics since there is no unique analytical solution. Each manipulator needs a particular method considering the system structure and restrictions[12].

3.6 Geometric Approach:

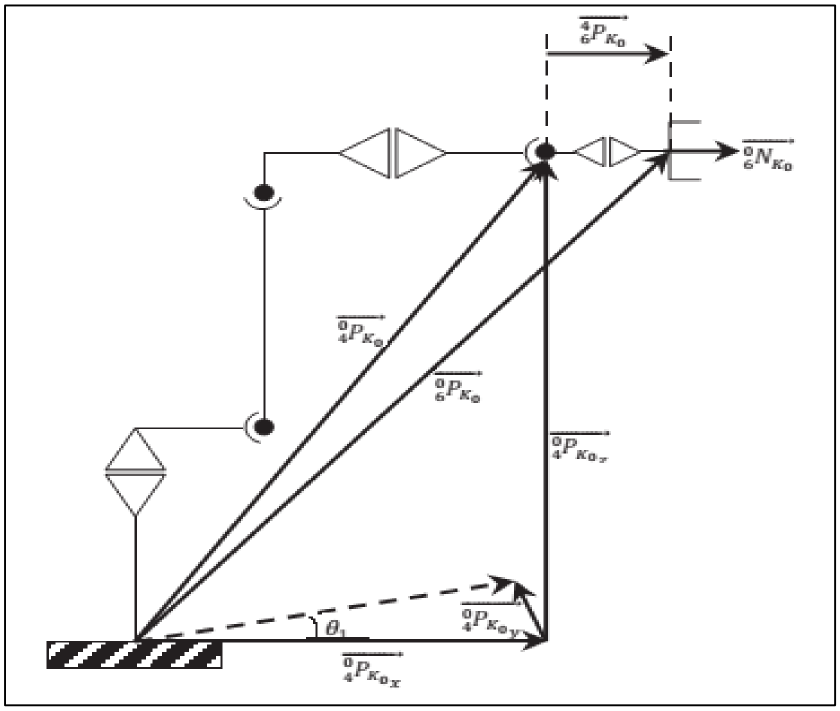
The user specifies the desired target position of the end-effector in Cartesian space as (x, y, z) where z is the height, and the angle of the end-effector relative to ground. In a geometrical method, vectors describe the robot's state to solve the problem which is the calculation of the joint angles of the robot. This section is divided into five subsections to illustrate the joint angles' computing method[8].

Joint 1

The first joint angle's calculation, as shown in figure (3-3), is accomplished by the projection of a vector which originates from the origin of frame K_0 and ends to the origin of frame $K_4(\overrightarrow{{}^{0}_{4}P_{k0}})$ on the X-Y plane of frame $K_{0.}$

let ${}_{G}^{0}T$ be the target transformation matrix relative to the base which defines the target position and orientation.

$$
{}_{G}^{0}T = \begin{bmatrix} {}_{G}^{0}T_{11} & {}_{G}^{0}T_{12} & {}_{G}^{0}T_{13} & {}_{G}^{0}T_{14} \\ {}_{G}^{0}T_{21} & {}_{G}^{0}T_{22} & {}_{G}^{0}T_{23} & {}_{G}^{0}T_{24} \\ {}_{G}^{0}T_{31} & {}_{G}^{0}T_{32} & {}_{G}^{0}T_{33} & {}_{G}^{0}T_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} \Rightarrow \overrightarrow{{}_{6}^{0}N_{k0}} = \begin{bmatrix} {}_{G}^{0}T_{13} \\ {}_{G}^{0}T_{23} \\ {}_{G}^{0}T_{33} \end{bmatrix} \tag{3.21}
$$



$\overrightarrow{{}_{6}^{4}P_{K_0}}$
$\overrightarrow{{}_{6}^{0}N_{K_0}}$
$\overrightarrow{{}_{4}^{0}P_{K_0}}$
$\overrightarrow{{}_{6}^{0}P_{K_0}}$
$\overrightarrow{{}_{4}^{0}P_{K_{0_z}}}$
$\overrightarrow{{}_{4}^{0}P_{K_{0_y}}}$
θ_1
$\overrightarrow{{}_{4}^{0}P_{K_{0_x}}}$

Figure 3.3: geometrical representation of first joint angle calculation.

then,

$$
\begin{cases} \overrightarrow{{}_{6}^{4}P_{k0}} = d_6 \times \overrightarrow{{}_{6}^{4}PN_{k0}} \\ \overrightarrow{{}_{6}^{0}P_{k0}} = \begin{bmatrix} {}_{G}^{0}T_{14} \\ {}_{G}^{0}T_{24} \\ {}_{G}^{0}T_{34} \end{bmatrix} \end{cases} \tag{3.22}
$$

$$\Rightarrow \overrightarrow{{}_{6}^{4}P_{k0}} = \overrightarrow{{}_{6}^{4}P_{k0}} - \overrightarrow{{}_{6}^{4}P_{k0}} = \begin{bmatrix} {}_{G}^{0}T_{14} - d_6\,{}_{G}^{0}T_{13} \\ {}_{G}^{0}T_{24} - d_6\,{}_{G}^{0}T_{23} \\ {}_{G}^{0}T_{34} - d_6\,{}_{G}^{0}T_{33} \end{bmatrix} \qquad (3.23)$$

so,

$$\theta_1 = \begin{cases} Atan2({}_{G}^{0}T_{24} - d_6\,{}_{G}^{0}T_{23}, {}_{G}^{0}T_{14} - d_6\,{}_{G}^{0}T_{13}) \\ Atan2({}_{G}^{0}T_{24} - d_6\,{}_{G}^{0}T_{23}, {}_{G}^{0}T_{14} - d_6\,{}_{G}^{0}T_{13}) + \pi \end{cases} \qquad (3.24)$$

Joint 3

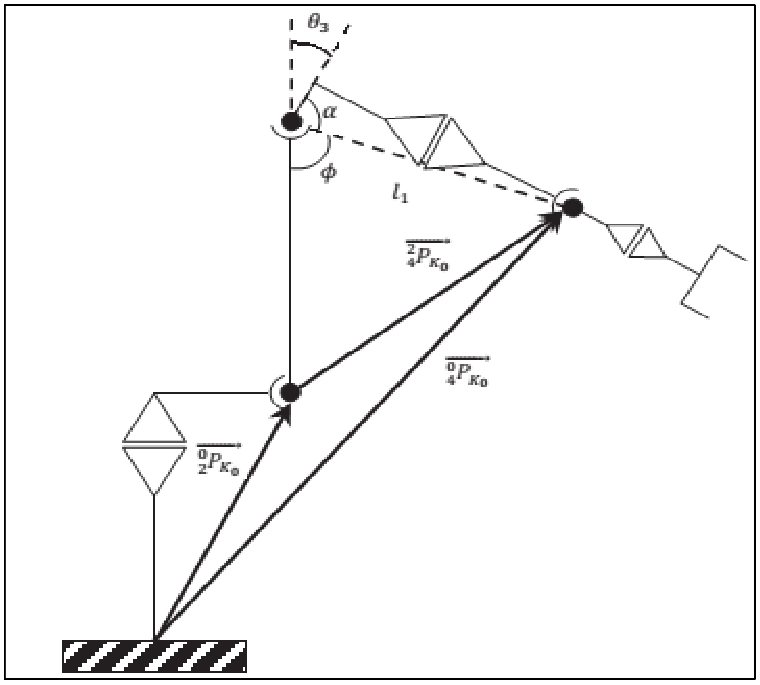

θ_3
α
ϕ
l_1
$\overrightarrow{{}_{4}^{2}P_{K_0}}$
$\overrightarrow{{}_{4}^{0}P_{K_0}}$
$\overrightarrow{{}_{2}^{0}P_{K_0}}$

Figure 3.4: Visual representation of joint 3 angle calculation.

Based on figure (3-4) illustration, to calculate θ_3, first $\overrightarrow{{}_{4}^{2}P_{k0}}$ needs to be calculated. In order to compute $\overrightarrow{{}_{4}^{2}P_{k0}}$, $\overrightarrow{{}_{4}^{0}P_{k0}}$ should be available, beforehand. By having $\overrightarrow{{}_{4}^{2}P_{k0}}$ and l$_1$, ϕ can be calculated and then by using a simple geometric rule, which helps to compute the angle of between two edges of a triangle, α will be quantified.

let $\theta_2 = 0$

and

$$ {}_{2}^{0}T = \begin{bmatrix} {}_{2}^{0}T_{11} & {}_{2}^{0}T_{12} & {}_{2}^{0}T_{13} & {}_{2}^{0}T_{14} \\ {}_{2}^{0}T_{21} & {}_{2}^{0}T_{22} & {}_{2}^{0}T_{23} & {}_{2}^{0}T_{24} \\ {}_{2}^{0}T_{31} & {}_{2}^{0}T_{32} & {}_{2}^{0}T_{33} & {}_{2}^{0}T_{34} \\ & & & \end{bmatrix} \qquad (3.25) $$

Thus,

$$ \Longrightarrow \overrightarrow{{}_{2}^{0}N_{k0}} = \begin{bmatrix} {}_{2}^{0}T_{13} \\ {}_{2}^{0}T_{23} \\ {}_{2}^{0}T_{33} \end{bmatrix} \qquad (3.26) $$

$$ \overrightarrow{{}_{4}^{2}P_{k0}} = \overrightarrow{{}_{4}^{0}P_{k0}} - \overrightarrow{{}_{2}^{0}P_{k0}} = \begin{bmatrix} \overrightarrow{{}_{4}^{2}P_{k0x}} \\ \overrightarrow{{}_{4}^{2}P_{k0y}} \\ \overrightarrow{{}_{4}^{2}P_{k0z}} \end{bmatrix} \qquad (3.27) $$

$$ \emptyset = Asin\left(\frac{\left(l_1^2 - a_2^2 + \left| \overrightarrow{{}_{4}^{2}P_{k0}} \right|^2 \right)}{2\left| \overrightarrow{{}_{4}^{2}P_{k0}} \right| l_1} \right) + Asin\left(\frac{\left| \overrightarrow{{}_{4}^{2}P_{k0}} \right| - \frac{l_1^2 - a_2^2 + \left| \overrightarrow{{}_{4}^{2}P_{k0}} \right|^2}{2\left| \overrightarrow{{}_{4}^{2}P_{k0}} \right|}}{a_2} \right) \qquad (3.28) $$

$$ \alpha = Atan2(-d_4, a_3) \qquad (3.29) $$

So,

$$ \theta_3 = \begin{cases} \pi - \emptyset - \alpha \\ \pi + \emptyset - \alpha \end{cases} \qquad (3.30) $$

Joint 2

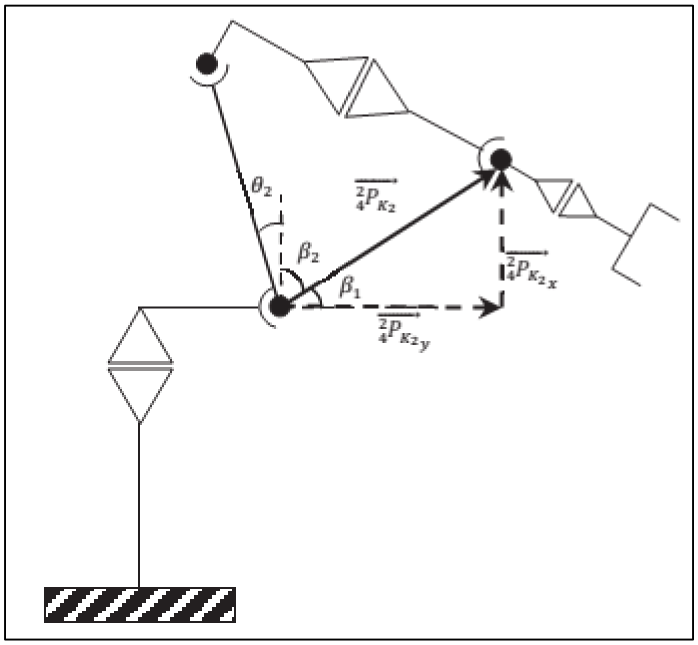

θ_2
$\overrightarrow{{}^2_4P_{K_2}}$
β_2
β_1
$\overrightarrow{{}^2_4P_{K_{2x}}}$
$\overrightarrow{{}^2_4P_{K_{2y}}}$

Figure 3.5: Joint 2 angle calculation vector representation.

θ2 is computed by $\overrightarrow{{}^2_4P_{k0}}$, β1 and β2 as Figure (3.5) displays

$$\overrightarrow{{}^2_4P_{k0}} = {}^2_0R\overrightarrow{{}^2_4P_{k0}} = {}^2_0R\overrightarrow{{}^{-1}{}^2_4P_{k0}} \tag{3.31}$$

$$ {}^0_2T = \begin{bmatrix} {}^0_2R & {}^0_2R_{ORG} \\ 0 & 1 \end{bmatrix} \tag{3.32}$$

$$ {}^0_2R = \begin{bmatrix} {}^0_2T_{11} & {}^0_2T_{12} & {}^0_2T_{13} \\ {}^0_2T_{21} & {}^0_2T_{22} & {}^0_2T_{23} \\ {}^0_2T_{31} & {}^0_2T_{32} & {}^0_2T_{33} \end{bmatrix} = {}^2_0R^{-1} \tag{3.33}$$

$$\overrightarrow{{}^2_4P_{k2}} = \begin{bmatrix} {}^0_2T_{11} & {}^0_2T_{12} & {}^0_2T_{13} \\ {}^0_2T_{21} & {}^0_2T_{22} & {}^0_2T_{23} \\ {}^0_2T_{31} & {}^0_2T_{32} & {}^0_2T_{33} \end{bmatrix} \begin{bmatrix} \overrightarrow{{}^2_4P_{k0x}} \\ \overrightarrow{{}^2_4P_{k0y}} \\ \overrightarrow{{}^2_4P_{k0z}} \end{bmatrix} \tag{3.34}$$

Thus

$$\beta_1 = Atan2\left(\overrightarrow{{}^2_4P_{k2x}}, \overrightarrow{{}^2_4P_{k2y}}\right) \tag{3.35}$$

$$\beta_2 = Asin\left(\frac{a_2^2 - \left|\overrightarrow{{}^2_4P_{k0}}\right|^2 + l_1^2}{2l_1a_2}\right) + Asin\left(\frac{l_1 - \frac{a_2^2 - \left|\overrightarrow{{}^2_4P_{k0}}\right|^2 + l_1^2}{2l_2}}{\left|\overrightarrow{{}^2_4P_{k0}}\right|}\right) \quad (3.36)$$

And then,

$$\theta_2 = \begin{cases} \frac{\pi}{2} - (|\beta_1| + \beta_2) \\ \frac{\pi}{2} + \left(|\beta_1|_{\beta_2}\right) \end{cases} \quad (3.37)$$

Joint 5

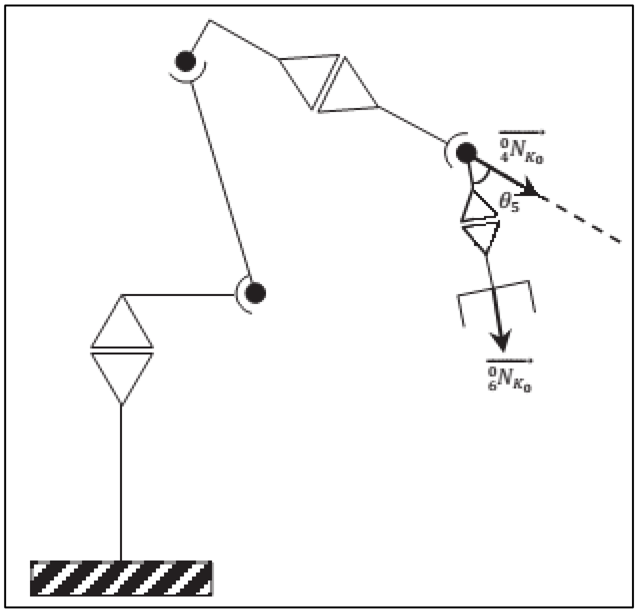

$\overrightarrow{{}^0_4N_{K_0}}$
θ_5
$\overrightarrow{{}^0_6N_{K_0}}$

Figure 3.6: Joint 5 angle calculation geometrical visualization.

In order to calculate θ_5, ${}^0_4T$ is computed by assuming θ_4 is equal to 0. Then by using the definition of dot product of two normal vectors which are shown in Figure (3.6), θ_5 is obtained.

$$ {}_4^0T = \begin{bmatrix} {}_4^0T_{11} & {}_4^0T_{12} & {}_4^0T_{13} & {}_4^0T_{14} \\ {}_4^0T_{21} & {}_4^0T_{22} & {}_4^0T_{23} & {}_4^0T_{24} \\ {}_4^0T_{31} & {}_4^0T_{32} & {}_4^0T_{33} & {}_4^0T_{34} \end{bmatrix} \Longrightarrow \overrightarrow{{}_4^0N_{k0}} = \begin{bmatrix} {}_4^0T_{13} \\ {}_4^0T_{23} \\ {}_4^0T_{33} \end{bmatrix} \tag{3.38} $$

So we have,

$$ \theta_5 = \pi - Acos\left(\overrightarrow{{}_4^0N_{k0}}, \overrightarrow{{}_6^0N_{k0}}\right) \tag{3.39} $$

Joint 4 and 6

To obtain θ$_4$ and θ$_6$, rotation matrix ${}_6^4R$ is used. On the one hand ${}_6^4R$ is:

$$ {}_6^4R = {}_4^0R^{-1}\,{}_6^0R = {}_0^4R\,{}_6^0R \tag{3.40} $$

And on the other hand,

$$ {}_6^4R = Rot_z(\theta_4)Rot_y(\theta_5 + \pi)Rot_z(\theta_6) \tag{3.41} $$

In which,

$$ Rot_y(\theta_5 + \pi) = \begin{bmatrix} cos(\theta_5 + \pi) & 0 & sin(\theta_5 + \pi) \\ 0 & 1 & 0 \\ -sin(\theta_5 + \pi) & 0 & cos(\theta_5 + \pi) \end{bmatrix} \tag{3.42} $$

$$ Rot_z(\theta_6) = \begin{bmatrix} cos(\theta_6) & -sin(\theta_6) & 0 \\ 0sin(\theta_6) & cos(\theta_6) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.43} $$

Thus,

$$ {}_6^4R = \begin{bmatrix} -c_4c_5c_6 - s_4s_6 & c_4c_5s_6 - s_4c_6 & -c_4s_5 \\ -s_4c_5c_6 + c_4s_6 & s_4c_5s_6 + c_4c_6 & -s_4s_5 \\ s_5c_6 & -s_5s_6 & -c_5 \end{bmatrix} \tag{3.44} $$

in which C$_4$ is corresponding to cos (θ$_4$) and S$_4$ is sin (θ$_4$) and so forth. For the sake of simplicity, let:

$$ {}_{6}^{4}R = \begin{bmatrix} {}_{6}^{4}R_{11} & {}_{6}^{4}R_{12} & {}_{6}^{4}R_{13} \\ {}_{6}^{4}R_{21} & {}_{6}^{4}R_{22} & {}_{6}^{4}R_{23} \\ {}_{6}^{4}R_{31} & {}_{6}^{4}R_{32} & {}_{6}^{4}R_{33} \end{bmatrix} \tag{3.45} $$

So, we have,

$$ \theta_4 = Atan2(-{}_{6}^{4}R_{23}, {}_{6}^{4}R_{13}) \tag{3.46} $$

$$ \theta_6 = Atan2(-{}_{6}^{4}R_{32}, {}_{6}^{4}R_{31}) \tag{3.47} $$

3.7 Dynamic Modeling of KUKA KR 6 Robot Manipulator:

In a dynamic model of a system there are two main aspects with which one is concerned: motion and forces. The motion of a system is called its trajectory and consists of a sequence of desired positions, velocities, and accelerations of some point or points in the system. Forces are usually characterized as internal (or constraint) forces and external (or applied) forces. The external forces are the ones which cause motion[11].

3.8 Forward Dynamics:

The Forward or direct dynamics is one where the forces which act on a robot are given and we wish to solve for the resulting motion. In its simplest form, the forward dynamics problem can be expressed symbolically as a vector differential equation of the form:

$$ \ddot{q} = h(q, \dot{q}, \tau, \text{manipulator parameters}) \tag{3.48} $$

where, q is the vector of generalized coordinates joint variables, $\ddot{q}$ and $\dot{q}$ are its derivatives with respect to time, 't is the (input) generalized force vector, i.e., the vector of joint torques and/or joint forces and the "manipulator parameters" are all those parameters which characterize the particular geometry and dynamics of a robot manipulator[11].

3.9 Inverse Dynamics:

The inverse dynamics is one in which we need to determine the generalized forces that will produce a specified motion trajectory. The inverse dynamics problem can be described mathematically by an equation of the form:

$$\tau = f(q, \dot{q}, \ddot{q}, \text{manipulator parameters}) \qquad (3.49)$$

3.10 KUKA KR 6 Manipulator Robot Parameters:

The dynamic simulation of a robotic is based upon a set of equations and assumes that a great number of parameters are known. These parameters include the inertia matrices that model the robot links, the center of mass of each link with respect to its D-H reference system and more... such as viscous friction factors.

To determine the inertia tensor of each, link the CAD model was used, obtaining the inertia moments around of the reference system used in the assembly module of the CAD software. As an example is presented the case of link 2[4].

Table 3.3: data of inertia and centroids using the software solid edge[4].

m2	38,767	Kg
I_{xx2}	112,0126	Kg-m^2
I_{yy2}	107,2872	Kg-m^2
I_{zz2}	16	Kg-m^2
I_{xy2}	7,1702	Kg-m^2
I_{xz2}	-22,7151	Kg-m^2
I_{yz2}	-30,4911	Kg-m^2

X_c	-0,3661	M
Y_c	-0,505	M
Z_c	1,55	M

3.11 Dynamic Formulations:

Based on Euler-Lagrange formulations the model represented in a matrix form is shown in equation as following:

$$T(t) = D\big(\theta(t)\big)\ddot{\theta}(t) + h\left(\theta(t), \dot{\theta}(t)\right) + c\big(\theta(t)\big) \qquad (3.50)$$

Were:

$T(t) = [T_1(t)T_2 \dots \dots T_n(t)]^T$Vector torque, size nx1

$\theta(t) = [\theta_1(t)\theta_2 \dots \dots \theta_n(t)]^T$Vector of joint positions, size nx1.

$\dot{\theta}(t) = \left[\dot{\theta}_1(t)\dot{\theta}_2 \dots \dots \dot{\theta}_n(t)\right]^T$Vector of angular Velocities, size nx1.

$\ddot{\theta}(t) = \left[\ddot{\theta}_1(t)\ddot{\theta}_2 \dots \dots \ddot{\theta}_n(t)\right]^T$Vector of angular acceleration, size nx1.

$$D_{ik} = \sum_{j=\max(i,k)}^{n} Tr\big(U_{jk}J_jU_{ji}^T\big) \text{ i, k} = 1,2.., \text{n Inertia matrix, size nxn}(3.51)$$

$$J_i = \begin{bmatrix} \frac{-I_{xx}+I_{yy}+I_{zz}}{2} & I_{xy} & I_{xz} & m_i\overline{x_i} \\ I_{xy} & \frac{I_{xx}-I_{yy}+I_{zz}}{2} & I_{yz} & m_i\overline{y_i} \\ I_{xz} & I_{yz} & \frac{I_{xx}+I_{yy}-I_{zz}}{2} & m_i\overline{z_i} \\ m_i\overline{x_i} & m_i\overline{y_i} & m_i\overline{z_i} & m_i \end{bmatrix} \text{Inertia Tensor, size 4x4 (3.52)}$$

By using data of inertia given in equation (3.52) the Inertia tensor can be determined as following matrix [4].

$$\begin{bmatrix} 2,295 & 4,062 & 7,170 & 0 \\ 4,062 & 53,840 & 22,715 & 7,753 \\ 7,170 & 22,715 & 1,111 & 14,195 \\ 0 & 7,753 & 14,195 & 38,767 \end{bmatrix}$$

$h(\theta, \dot{\theta})$
$= [h_1\ h_2 \ldots \ldots h_n]^T$Vector of Coriolis and centrifugal force, Size nx1(3.53)

$$h_i = \sum_{k=1}^{n} \sum\nolimits_{m=1}^{n} h_{ikm} \dot{\theta}_k \dot{\theta}_m \qquad i = 1,2, \ldots \ldots n \tag{3.54}$$

$$h_{ikm} = \sum_{j=\max(i,k,m)}^{n} Tr\left(U_{jkm} J_j U_{ji}^T\right) \text{ i, k, m} = 1,2, \ldots, \text{n} \tag{3.55}$$

$c(\theta) = [C_1\ C_2 \ldots \ldots \ldots C_n]^T$Gravity forces vector size nx1

$$C_i = \sum\nolimits_{j=i}^{n} \left(-m_j g U_{ji}^j r_j\right) \qquad i = 1,2, \ldots., n \tag{3.56}$$

From the direct kinematic model, presented it is necessary determine U_{jk} matrices, the inertia tensor Ji for each link, the inertia effects D, the matrix hi and h_{ijk} of Coriolis and centrifugal acceleration, the position vector R and the gravitational vectors force C.

To calculate the matrix U_{jk} is used the canonical equation as:

$$U_{jk} = \frac{\partial\ {}^0A_j}{\partial \theta_k} = {}^0A_{j-1} Q_i^{j-1} A_k \tag{3.57}$$

For the determination of the matrix D (matrix of inertial effects). It is necessary to use the following Equation**:**

$$D(\theta) = \begin{bmatrix} D_{11} & D_{12} & D_{13} \\ D_{21} & D_{22} & D_{23} \\ D_{31} & D_{32} & D_{33} \end{bmatrix} \tag{3.58}$$

were:

$$D_{11} = Tr\left(U_{11}J_1U_{11}^T\right) + Tr\left(U_{21}J_2U_{21}^T\right) + Tr\left(U_{31}J_3U_{31}^T\right)$$

$$D_{12} = D_{21} = Tr\left(U_{22}J_2U_{21}^T\right) + Tr\left(U_{32}J_3U_{31}^T\right)$$

$$D_{13} = D_{31} = Tr\left(U_{33}J_3U_{31}^T\right)$$

$$D_{22} = Tr\left(U_{22}J_2U_{22}^T\right) + Tr\left(U_{32}J_3U_{32}^T\right)$$

$$D_{23} = D_{32} = Tr\left(U_{33}J_3U_{32}^T\right)$$

$$D_{33} = Tr\left(U_{33}J_3U_{33}^T\right)$$

For the determination of vector h, vector of Coriolis and centrifugal forces, the following equation is proposed. This equation presents the angular velocities independently through the matrix $H_{i,v}$

$$h_i = \dot{\theta}^T H_{i,v}\dot{\theta} \tag{3.59}$$

Where

$$H_{i,v} = \begin{bmatrix} h_{i_{11}} & h_{i_{12}} & h_{i_{13}} \\ h_{i_{21}} & h_{i_{22}} & h_{i_{23}} \\ h_{i_{31}} & h_{i_{32}} & h_{i_{33}} \end{bmatrix} \tag{3.60}$$

$$h_{ikm} = Tr\left(U_{jkm}J_jU_{ji}^T\right) \quad j = \max(i,k,m) \tag{3.61}$$

$$U_{jkm} = \frac{\partial U_{jk}}{\partial \theta_m} = {}^0A_{k-1}Q_k{}^{k-1}A_{m-1}Q_m{}^{m-1}A_i \quad j \geq k \geq m \tag{3.62}$$

$$U_{jkm} = \frac{\partial U_{jk}}{\partial \theta_m} = {}^0A_{m-1}Q_m{}^{m-1}A_{k-1}Q_k{}^{k-1}A_i \quad j \geq m \geq k \tag{3.63}$$

$$U_{jkm} = \frac{\partial U_{jk}}{\partial \theta_m} = 0 \qquad \text{se} \quad j < k \quad \text{ou} \quad j < m \tag{3.64}$$

$$U_{jkm} = \frac{\partial U_{jk}}{\partial \theta_m} = \frac{\partial}{\partial \theta_m}\frac{\partial\, {}^0A_j}{\partial \theta_k} \tag{3.65}$$

Thus, the vector *h* of centrifugal and Coriolis forces is:

$$\begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} \dot{\theta}_1 & \dot{\theta}_2 & \dot{\theta}_2 \end{bmatrix} \begin{bmatrix} h_{111} & h_{112} & h_{113} \\ h_{121} & h_{122} & h_{123} \\ h_{131} & h_{132} & h_{122} \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_2 \end{bmatrix} \\ \begin{bmatrix} \dot{\theta}_1 & \dot{\theta}_2 & \dot{\theta}_2 \end{bmatrix} \begin{bmatrix} h_{211} & h_{212} & h_{213} \\ h_{221} & h_{222} & h_{223} \\ h_{231} & h_{232} & h_{222} \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_2 \end{bmatrix} \\ \begin{bmatrix} \dot{\theta}_1 & \dot{\theta}_2 & \dot{\theta}_2 \end{bmatrix} \begin{bmatrix} h_{311} & h_{312} & h_{313} \\ h_{321} & h_{322} & h_{323} \\ h_{331} & h_{332} & h_{322} \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_2 \end{bmatrix} \end{bmatrix} \tag{3.66}$$

For the Determination of the gravity force vector C:

$$c(\theta) = [C_1 \quad C_2 \quad C_3]^T \tag{3.67}$$

$$C_i = \sum_{j=i}^{3} \left(-m_j g U_{ji} {}^{j}\overline{r_j}\right) \ i = 1,2,3 \tag{3.68}$$

$$C_1 = -m_1 g U_{11} {}^{1}\overline{r_1} - m_2 g U_{21} {}^{2}\overline{r_2} - m_3 g U_{31} {}^{3}\overline{r_3}$$

$$C_2 = -m_2 g U_{22} {}^{2}\overline{r_2} - m_3 g U_{32} {}^{3}\overline{r_3}$$

$$C_3 = -m_3 g U_{33} {}^{3}\overline{r}$$

$$\begin{pmatrix} C_1 \\ C_2 \\ C_3 \end{pmatrix} = \begin{bmatrix} -m_1 g U_{11} {}^{1}\overline{r_1} - m_2 g U_{21} {}^{2}\overline{r_2} - m_3 g U_{31} {}^{3}\overline{r_3} \\ -m_2 g U_{22} {}^{2}\overline{r_2} - m_3 g U_{32} {}^{3}\overline{r_3} \\ -m_3 g U_{33} {}^{3}\overline{r} \end{bmatrix} \tag{3.69}$$

The vector r in the reference system of rotation axes is:

$${}^{1}r_1 = \begin{bmatrix} -0.0052 \\ 0.0026 \\ 0.369 \\ 1 \end{bmatrix} {}^{2}r_2 = \begin{bmatrix} 0 \\ 0.2 \\ 0.366 \\ 1 \end{bmatrix} {}^{3}r_3 = \begin{bmatrix} 0 \\ 0.454 \\ -.00015 \\ 1 \end{bmatrix} \tag{3.70}$$

$$g = [0 \ \ 0 \ -g \ \ 0\,]$$

The total system is then as follows:

$$T(t) = D\big(\theta(t)\big)\ddot{\theta}(t) + h\left(\theta(t), \dot{\theta}(t)\right) + c\big(\theta(t)\big)$$

Chapter Four

MATLAB Simulation for KUKA KR 6 Robot

Chapter Four

MATLAB Simulation for KUKA KR 6 Robot

4.1 Introduction:

The fundamental of robotics technology such as kinematics, dynamics, coordinate transformation and trajectory planning is neither easy nor fun like watching robot movies or playing with robot toys for the students.

Apart from difficulty in grasping the mathematical concept and relate it immediately to robot practical behavior, visualizing the concept through conventional teaching approach is challenging. The safety concern also hinders the extent to which researchers could be allowed to independently handle and explore robotics equipment's, and hence limits the attainment of the learning outcomes. It is therefore important to review strategy with incorporation of relevant information technology tools which is constantly evolving for effective and productive outcomes.

Several tools are now available in this respect, among them is a MATLAB based computational toolboxes dedicated to robotics applications. This toolbox provides collection of functions (tools) that support representation and presentation of fundamental concepts in robotics such as robot configuration based on standard notations, robot kinematics, dynamics and trajectory generation, etc.

4.2 Robotics Technology and MATLAB Toolboxes:

Robotics technology is the art, knowledge base, and the know-how of designing, applying and using robots in human endeavors. It is an integrated field of study incorporating several areas of science and engineering discipline. The major learning activities and skills required been: physical design of structures and mechanisms; computational design including software/program development; and lastly mission planning. The details of each of these activities depends on the type of robot and its domain of application.

In this research, focus is beamed on serial robots, also known as robot manipulator due to their overwhelming popularity in industrial application and automation. The fundamental concepts needed in the design and development of this type of robots are: pose and coordinate transformation, geometry description; forward and inverse kinematics; and trajectory planning for a given application. These concepts are not only mathematically intensive; they require intuitive understanding in relation to their practical applications. With these challenges in mind, MATLAB robotics toolboxes was developed[13].

4.3 MATLAB Toolboxes:

The toolboxes are organized functionally to address each of the fundamental concepts. It provides computation and visualization capabilities for effective learning and students engagements. The next subsections outline these basic concepts alongside the functional tools provided by the toolboxes.

In this research toolbox used is a new toolbox ARTE (A Robotics Toolbox for Education) that focused on the teaching of robotic manipulators which it is developed by SPAINS searchers from Miguel Hernández University of Elche (UMH).

4.4 A Robotics Toolbox for Education (ARTE):

ARTE is a new library of toolbox focused on the teaching of robotic manipulators. The library works under MATLAB and has been designed to strengthen the theoretical concepts of manipulator robots. The educational approach is focused on the main concepts through developing math modeling and simulation. The library possesses features that typically needed the usage of proprietary software, such as the visualization of a realistic 3D representation of commercial robotic arms and the programming of those arms in an industrial language. That includes the concepts of direct and inverse kinematics, inverse and direct dynamics, path planning and robot programming. As a transversal practice, during the sessions, the student is asked to choose and integrate a new robotic arm in the library, proposing a particular solution to the direct and inverse kinematic problem, as well as the inclusion of other important parameters[14].

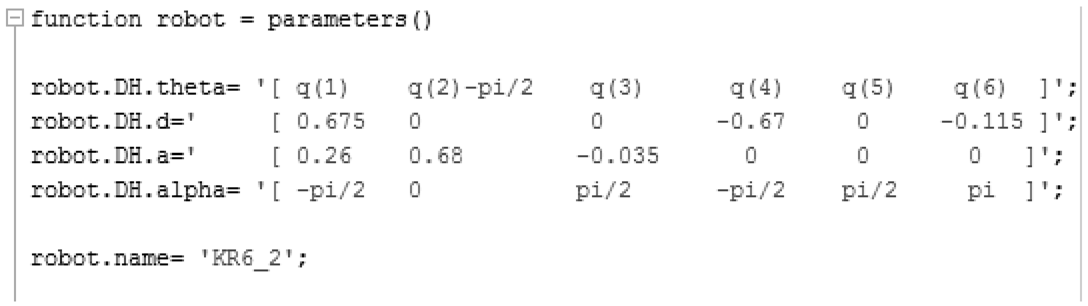
4.5 Main Features of The Toolbox:

The toolbox presents the following main features:

(a) Denavit–Hartenberg's representation of the robotic manipulator.
(b) Capacity to visualize the position, velocity and acceleration of the joint variables of the robot when it performs a movement. In addition, the capacity to represent the velocity of the end effector during the simulation.
(c) Capacity to visualize joint forces and torques during the performance of any movement.
(d) Realistic 3D representation of the robot link as solid objects.
(e) Path planning of joint trajectories.
(f) Programming the robot in an industrial language. Step-bystep simulation of the program.
(g) Programming of the robot using a virtual teach pendant. Creation of target points and way points using a graphical interface.
(h) Easy inclusion of new robots.
(i) Realistic representation of the robot in a robotic cell with auxiliary equipment. The toolbox is available to download in its web site[15].

4.6 Geometry Description:

This involves systematic description of robot arm geometry, and determination of its configuration parameters based on number of joints, links and relative pose of each joint to its preceding one. A common approach is to use method proposed by Denavit and Hartenberg , known as D-H notation.

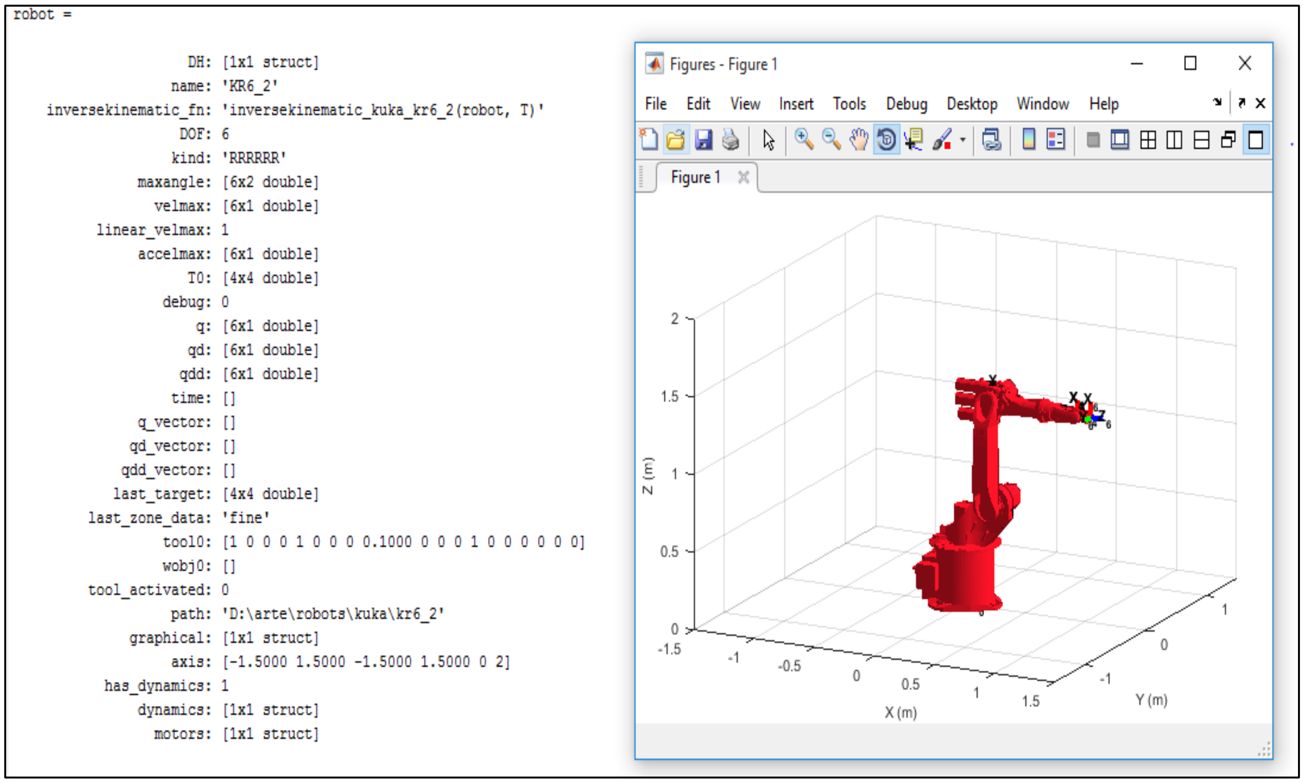


```
function robot = parameters()

robot.DH.theta= '[ q(1)    q(2)-pi/2    q(3)      q(4)    q(5)    q(6)  ]';
robot.DH.d='      [ 0.675   0            0        -0.67     0     -0.115 ]';
robot.DH.a='      [ 0.26    0.68        -0.035      0       0        0   ]';
robot.DH.alpha= '[ -pi/2   0           pi/2      -pi/2    pi/2     pi  ]';

robot.name= 'KR6_2';
```

4.7 Visualization Tool:

A visualization tool is highly required in this task for effective studies' engagement. This is one of the great potential of the toolboxes. Functions such as "load_robot" is provided to represent a link and create a given serial robot, respectively.

Visualize and animate the created robot using following functions robot = load_robot(manufacturer, version), which returns a robot data structure of the specified robot. Each robot parameters are stored in the directory named robots/manufacturer/version. For our study robot we can load a data structure corresponding to the robot by using following function robot=load_robot('kuka', 'kr6_2'), as shown in figure (4.1), MATLAB code for Visualization robot in [appendix B1]



```
robot =

                   DH: [1x1 struct]
                 name: 'KR6_2'
 inversekinematic_fn: 'inversekinematic_kuka_kr6_2(robot, T)'
                  DOF: 6
                 kind: 'RRRRRR'
             maxangle: [6x2 double]
               velmax: [6x1 double]
        linear_velmax: 1
             accelmax: [6x1 double]
                   T0: [4x4 double]
                debug: 0
                    q: [6x1 double]
                   qd: [6x1 double]
                  qdd: [6x1 double]
                 time: []
             q_vector: []
            qd_vector: []
           qdd_vector: []
          last_target: [4x4 double]
       last_zone_data: 'fine'
                tool0: [1 0 0 0 1 0 0 0 0.1000 0 0 0 1 0 0 0 0 0 0]
                wobj0: []
       tool_activated: 0
                 path: 'D:\arte\robots\kuka\kr6_2'
            graphical: [1x1 struct]
                 axis: [-1.5000 1.5000 -1.5000 1.5000 0 2]
         has_dynamics: 1
             dynamics: [1x1 struct]
               motors: [1x1 struct]
```

Figure 4.1: KUKA KR6 view and its corresponding data structure.

The robot view can be adjusted by using (adjust_view) function, and (draw_axes) to draw axes (T, X_text, Y_text, Z_text, scale). There are another functions for visualization requirements such as:

(animate (robot, q) and drawrobot3d(robot, q)).

4.8 Kinematics concept:

kinematics is the study of robot movement (motion) without recourse to the force that causes it. Two types of kinematics are involved in serial robot manipulator. forward kinematics and inverse kinematics.

4.9 Direct Kinematics:

Forward kinematics seek to answer the following operational query, “given the robot joint variables (angles), q, determine the robot end-effector pose, relative to the reference frame. Mathematically, transformation process expressed as product of individual link transformation matrices. The transform matrix is given by the equations (3.3 – 3.4) in previous chapter [10].

The total transformation between the base of the robot and the hand is:

$$^{R}T_{H} = \mathrm{T1.T2.T3.T4.T5.T6} \tag{4.1}$$

The forward kinematics are computed using the toolbox function, "directkinematic"

if we assume initial values for each joint (6 joints)

q = [0.5 -0.5 pi/6 0.1 0.1 0.1]

to compute direct kinematics for this position q

T = directkinematic(robot, q) figure(4.2).

MATLAB code for forward kinematic of the KUKA KR 6 robot appended in [appendix B2].

4.10 Inverse Kinematics:

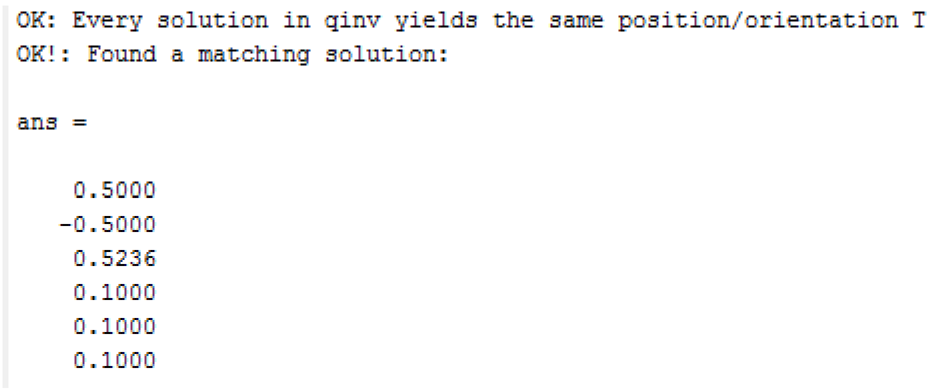
The second type of kinematics, known as inverse kinematics is used to determine the required joint angles for a given robot end-effector. There are eight possible solutions for the inverse kinematic problem for most of these robots. Unlike forward kinematics, it is quite computational intensive, it is not unique, and a close-form solution may not exist for classes of robots. The inverse kinematics are computed using the toolbox function, "inversekinematic"

n_solutions = 8; not all the eight possible solutions will be feasible for an anthropomorphic 6 axis robot.

A call the "inversekinematic" for this robot. All the possible solutions are stored at qinv. At least, one of the possible solutions should match q

qinv = inversekinematic(robot, T); as shown in figure(4.3) for n = 8 solutions all of them matched with input q values.

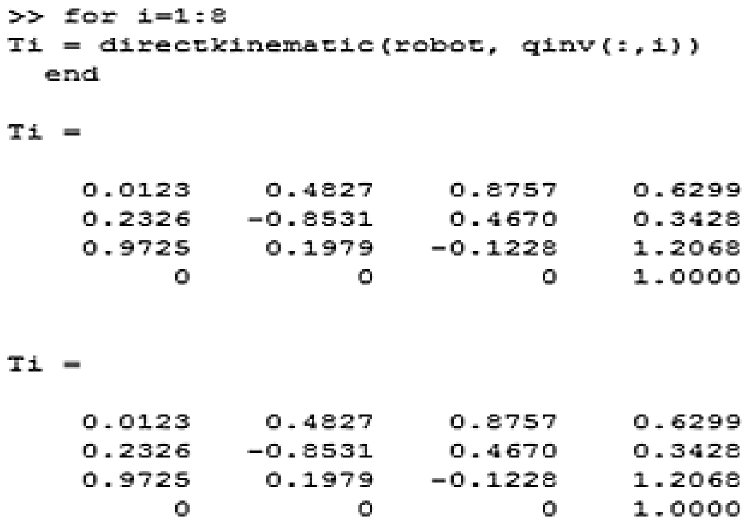
MATLAB code for inverse kinematic of the KUKA KR 6 robot appended in [appendix B3].



```
OK: Every solution in qinv yields the same position/orientation T
OK!: Found a matching solution:

ans =

    0.5000
   -0.5000
    0.5236
    0.1000
    0.1000
    0.1000
```

Figure 4.2: snapshot for MATLAB kinematic function results for KUKA KR6 robot.

To check that all of them are feasible solutions and every Ti equals for i=1:8, Ti = directkinematic(robot, qinv(:,i))



```
>> for i=1:8
Ti = directkinematic(robot, qinv(:,i))
  end

Ti =

    0.0123    0.4827    0.8757    0.6299
    0.2326   -0.8531    0.4670    0.3428
    0.9725    0.1979   -0.1228    1.2068
         0         0         0    1.0000


Ti =

    0.0123    0.4827    0.8757    0.6299
    0.2326   -0.8531    0.4670    0.3428
    0.9725    0.1979   -0.1228    1.2068
         0         0         0    1.0000
```

Figure 4.3: MATLAB snapshot sample of 8 solution matched with q values.

4.11 Trajectory Planning:

A trajectory is the path followed by the manipulator, plus the time profile along the path. It involves planning of the robot movement from one pose to the other.

Issues in trajectory planning include: attaining a specific target from an initial starting point, avoiding obstacles, and staying within manipulator capabilities. Generally, given two ends points, it is required to determine series of poses to be follow by the robot from starting point to the end point using either joint space trajectory, or Cartesian space trajectory approach. The detail description of the two types of trajectory planning methods can be found in the literatures[13].

The following MATLAB code is used for the joint space and Cartesian space approach, respectively as shown in figures (4.4) & (4.5).

MATLAB code for joint space and Cartesian space approach [Appendix B4].

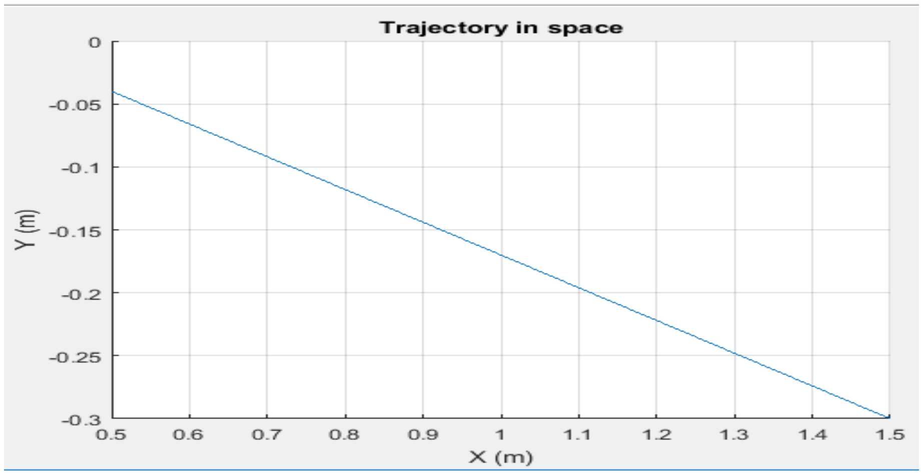


Figure 4.4: joint trajectory in joint space approach.

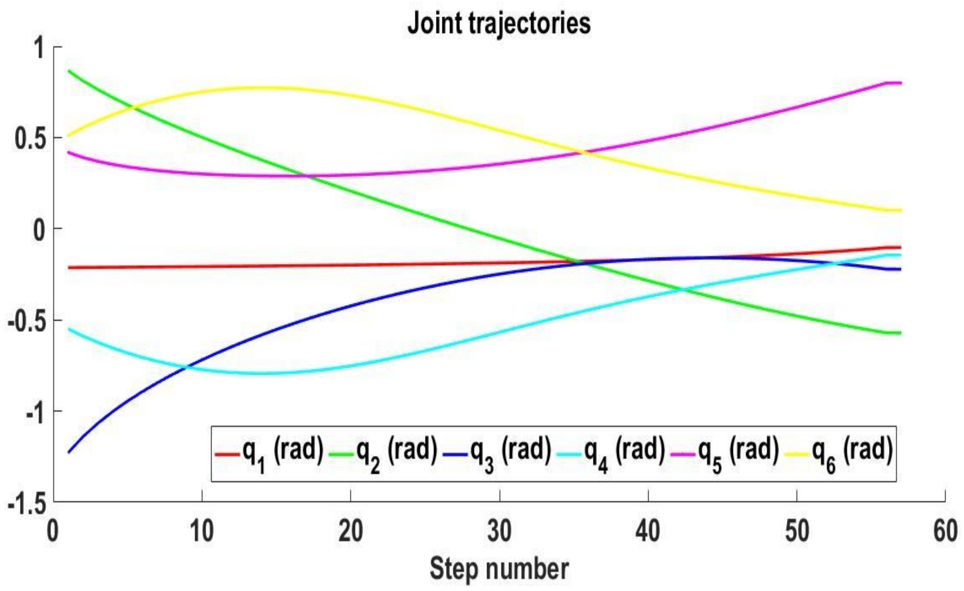


Figure 4.5: joint trajectory in Cartesian space approach.

4.12 Dynamic Modeling:

To validate the dynamic model, the input data were oblique trajectories and tasks that require the motion of the terminal element with constant speed, working load which corresponds to the maximum load recommended by KUKA Robot[4]. “forwarddynamic” function can be used to test the manipulator robot, for applied this function we should determine simulation time and values for position and joint speeds as in figures (4.6) & (4.7).

The MATLAB code for direct dynamic for KUKA KR6 robot as assumed no friction and no torque applied [Appendix B5].

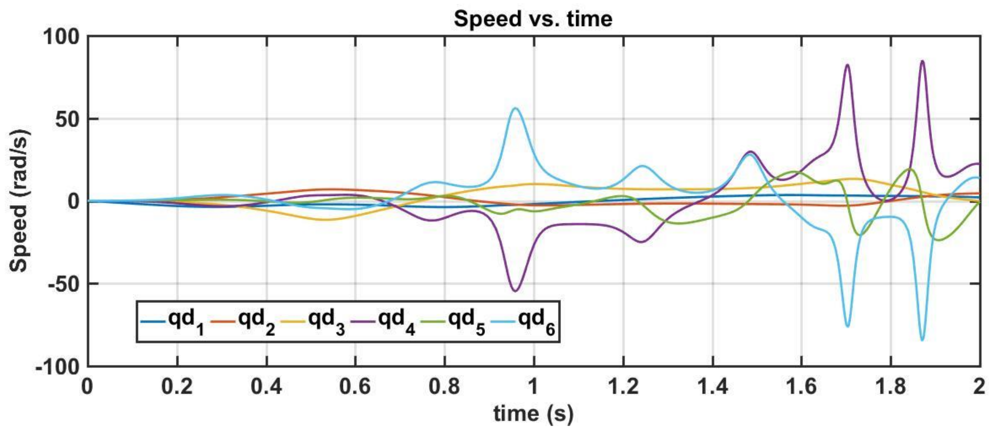


Figure (4.6): varying joints speed with time.

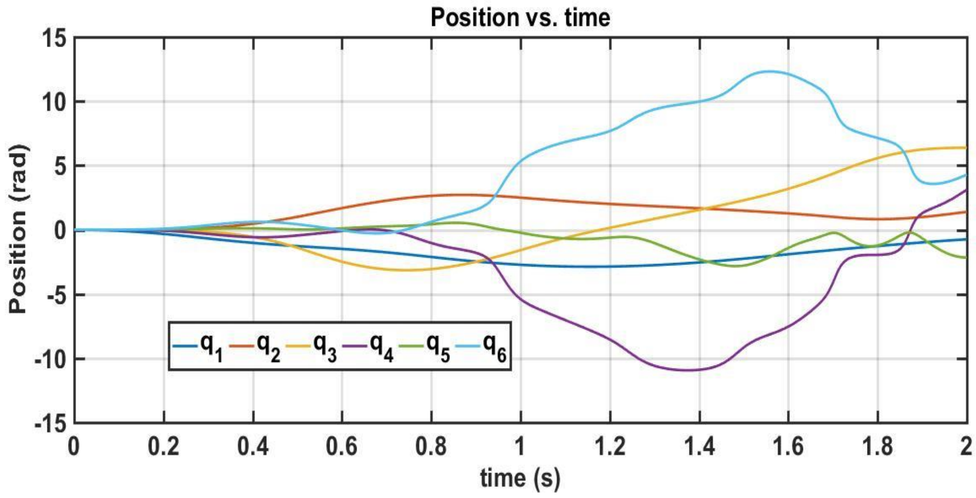


Figure (4.7): varying joints position with time.

4.13 Inverse Dynamics:

Manipulator inverse dynamics, or simply inverse dynamics, is the calculation of the forces and/or torques required at a robot's joints in order to produce a given motion trajectory consisting of a set of joint positions, velocities and accelerations. Mathematically, the inverse dynamics problem can be described by a vector equation of the form:

$$\tau = f(q, \dot{q}, \ddot{q}, \text{manipulator parameters}) \qquad (4.2)$$

Where

$\tau = torque$

$q = joint\ position$

$\dot{q} = joint\ speed$

$\ddot{q}$ =acceleration

the function “inversedynamic” used to Compute inverse dynamics via recursive Newton-Euler.

general call to inverse dynamic function:

TAU = inversedynamic(robot, Q, QD, QDD, GRAV, FEXT)

where if one of three different poses chosen then find which one the worst.

assume the poses have the following values

q1 = [0 0 0 0 0 0];

q2 = [0 pi/2 -pi/2 0 0 0];

q3 = [0 0 -pi/2 0 0 0];

for dynamic robot friction = 1

robot dynamics friction=1;

sample MATLAB code to compute torques at each pose(q1&q2) [Appendix B6] case (1) at pose q1= [0 0 0 0 0 0] as shown in figure (4.8).

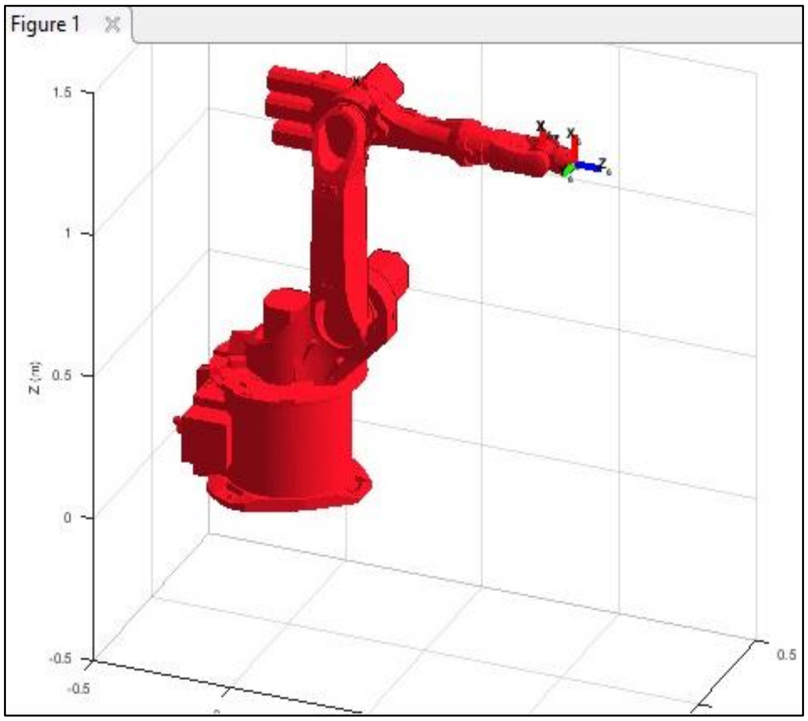


Figure 4.8 : robot 3d view for q1= [0 0 0 0 0 0].

In this case the result of MATLAB computing for inverse dynamic at q1 as following: Torques at each joint given position q1, zero speed and acceleration, standard gravity acting on Z0 computing static torques at position q1 due to gravity [0 0 9.81]

tau = -0.0000, 87.3335, 85.7443, 0.0000, 2.1631, -0.0000

Case (2) at pose q2 = [0 pi/2 -pi/2 0 0 0] as shown in figure (4.9).

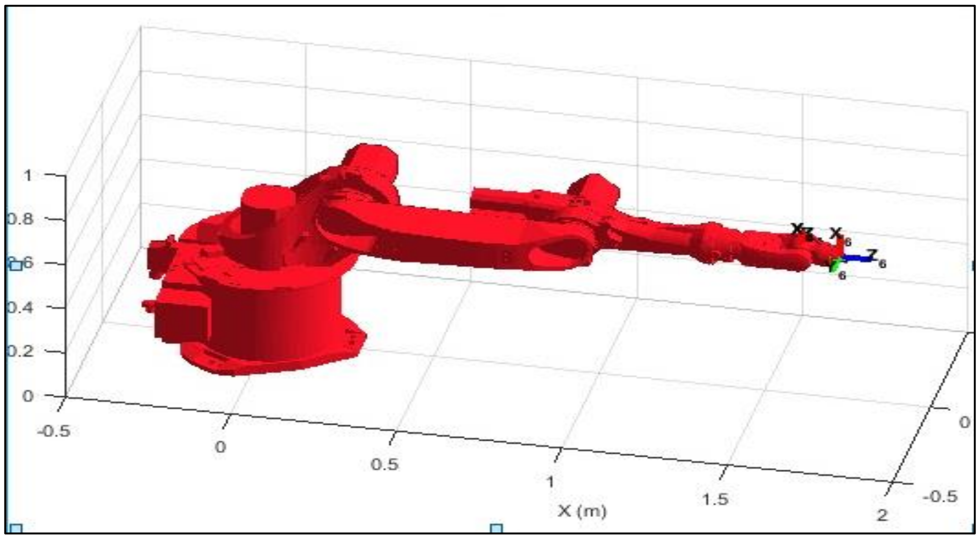


Figure 4.9: robot 3d view for q1= [0 pi/2 -pi/2 0 0 0].

In this case the result of MATLAB computing for inverse dynamic at q1 as following Torques at each joint given position q2, zero speed and acceleration, static torques at position q2 due to gravity [0 0 9.81]. The differences noted with respect to torque tau_2 at q1

tau = 0.0000, 446.0656, 85.7443, 0.0000, 2.1631, -0.0000. From previous analysis, some differences of torques can be noted respect to position values.

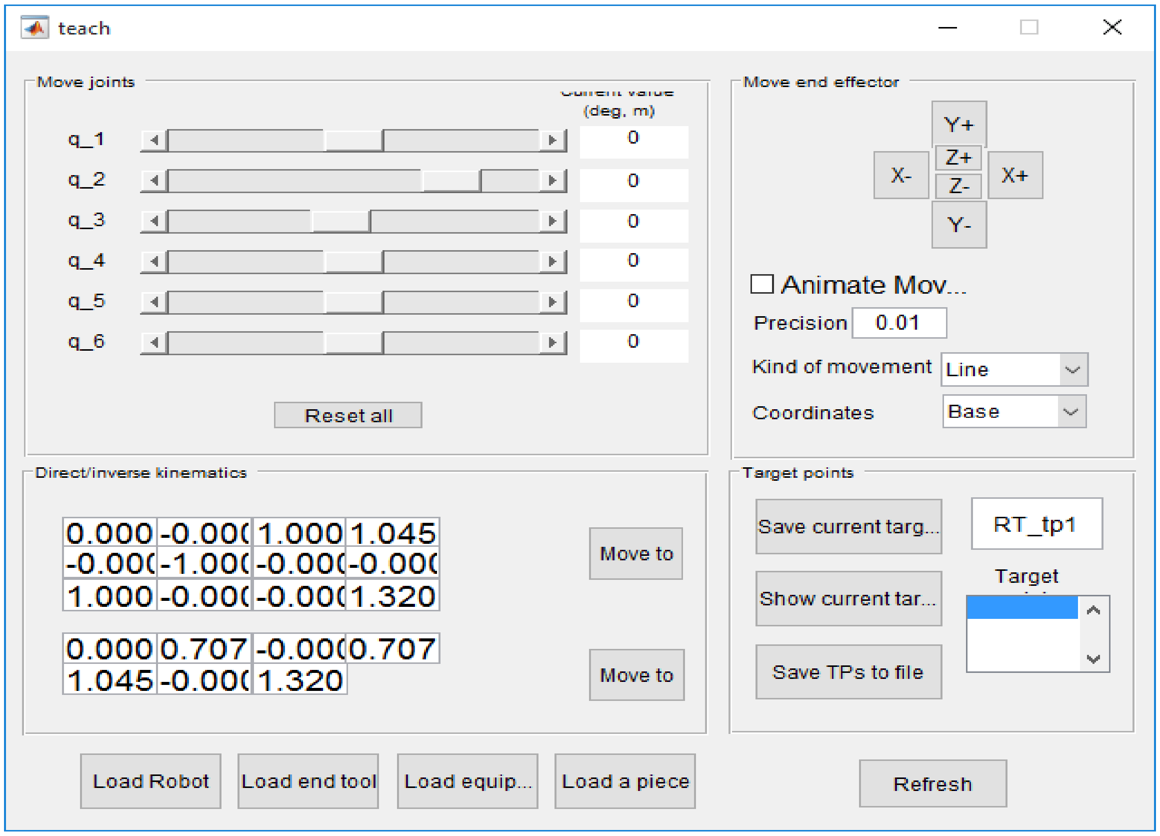
4.14 Robot Programming Tools:

programming tools are set of functions allow us to build our desired application such as robot for pick and place application. The ARTE library includes a subset of instructions of the ABB RAPID language that specialized for robotic programming. Programming in ARTE will be done in the following way[15]:

a) The teach graphical user interface (GUI) allows to simulate the robot and program target points. The user will place the robot in different points in the workspace that will be needed, for example, to pick a piece or place it inside a box figure (4.10).

b) These points will be defined in a m-file in order to do this, the points created in the teach application can be exported to a m-file by using “teach” function as shown in figure (4.10). Next, the user should write a program using the equivalent MATLAB functions provided, such as MoveJ(), MoveL() or MoveAbsJ().

c) The program can be simulated under MATLAB. By using its debugging tools, you may execute the program step by step or even look into the MATLAB’s functions.



q_1: 0
q_2: 0
q_3: 0
q_4: 0
q_5: 0
q_6: 0

☐ Animate Mov...
Precision: 0.01
Kind of movement: Line
Coordinates: Base

0.000	-0.00(	1.000	1.045
-0.00(	-1.00(	-0.00(	-0.00(
1.000	-0.00(	-0.00(	1.320

0.000	0.707	-0.00(	0.707
1.045	-0.00(	1.320	

RT_tp1

Figure 4.10: the usage of the teach graphical user interface (GUI).

4.15 Allumium Milling Application Using Kuka Kr6 Robot:

There are many functions used to program the robot and make control motion of the robot, such as following function with their tasks. *MoveL*: Make a linear planning in space.

MoveC: Make a circular path in space.

Matlab code for the application [Appendix B7] and the result as shown in figure (4.11).

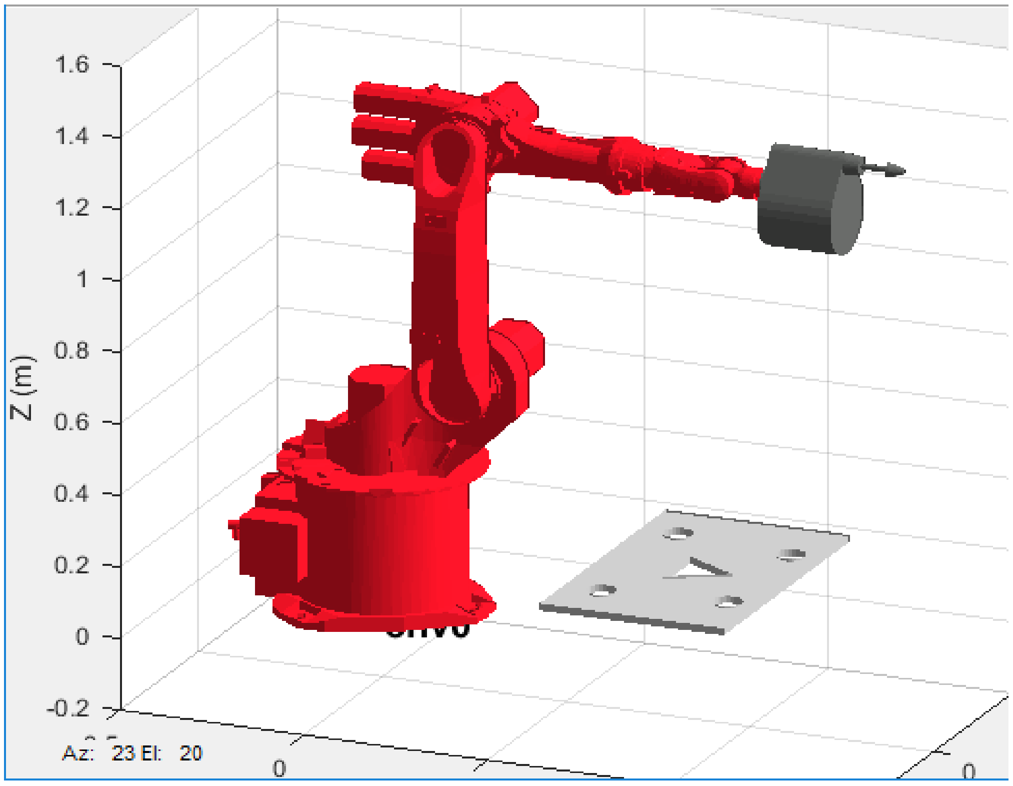


Figure 4.11: 3d view for aluminum plate milling application.

Chapter Five

Results and Discussions

Chapter Five

Results and Discussions

5.1 Introduction:

As the basic principle of the manipulator robots modelling illustrated in chapter tow, these principles implemented on the target manipulator robot KUKA KR 6 as a practical study to support the research by applicable real scientific sides as much as possible. The KUKA KR 6 robot subjected to mathematical analysis, where the kinematic and dynamic formulations for robot derived as shown in chapter three.

The transformation matrix based on DH parameters used to obtain the forward kinematic equations and the invers kinematic equations derived by using geometrical method. The dynamic modeling analysis based on Euler-Lagrange equations and recursive Newton-Euler formulations.

New software tools work under MATLAB environment called ARTE (a robotic toolbox for education) used to simulate the robot in advanced way to get modeling analysis. The results of the modeling and MATLAB simulation that obtained in chapter three and chapter four can be discussed as the following:

5.2 Kinematic Modeling and Simulation Results:

The direct kinematic represented by transformation matrix as:

$$^{n-1}T_n = Rot_{x_{n-1}}(\alpha_{n-1}).Trans_{Z_{n-1}}(\alpha_{n-1}).Rot_{Z_n}(\theta_n).Trans_{Z_n}(d_n) \quad (5.1)$$

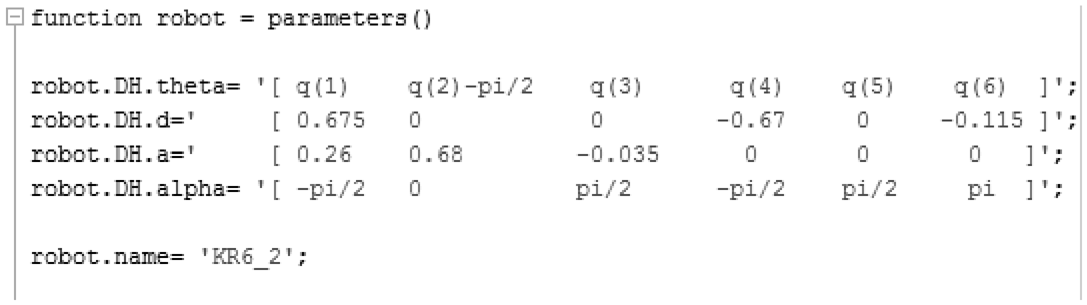
$$^{n-1}T_n = \begin{bmatrix} C\theta_{n+1} & -S\theta_{n+1}C\alpha_{n+1} & S\theta_{n+1}S\alpha_{n+1} & \alpha_{n+1}C\theta_{n+1} \\ S\theta_{n+1} & C\theta_{n+1}C\alpha_{n+1} & -C\theta_{n+1}S\alpha_{n+1} & \alpha_{n+1}S\theta_{n+1} \\ 0 & S\alpha_{n+1} & C\alpha_{n+1} & d_{n+1} \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5.2)$$

The direct kinematic can be obtained by this formulation but in manually calculation it needs to exert more time and work because the complicated calculations and the results may be not accurate as well. Where we need to calculate the transformation matrix for each link T1, T2...etc. then aggregate the total matrix.

$$^{R}T_{H} = \mathrm{T1.T2.T3.T4.T5.T6} \quad (5.3)$$

A computer simulation program and software tools help to get the accurate analysis and calculation, such as ARTE toolbox that used in this research where this toolbox has many features such as inclusion the important functions that called to do some complex mathematic operations and the important robotic parameters that are needed to analysis such as DH parameters.

By using ARTE tool, the visualizations geometry of the robot presents in 3d based on DH parameters as following:



```
function robot = parameters()

robot.DH.theta= '[ q(1)    q(2)-pi/2    q(3)      q(4)    q(5)    q(6)  ]';
robot.DH.d='     [ 0.675   0            0        -0.67     0      -0.115 ]';
robot.DH.a='     [ 0.26    0.68        -0.035      0       0        0   ]';
robot.DH.alpha= '[ -pi/2   0            pi/2      -pi/2    pi/2     pi   ]';

robot.name= 'KR6_2';
```

Figure 5.1: DH parameters represented in MATLAB.

The function “robot = load_robot(manufacturer, version)” used to visualize and animate the robot. For KUKA KR 6 robot we can use the function as the following:

robot=load_robot('kuka', 'kr6_2')

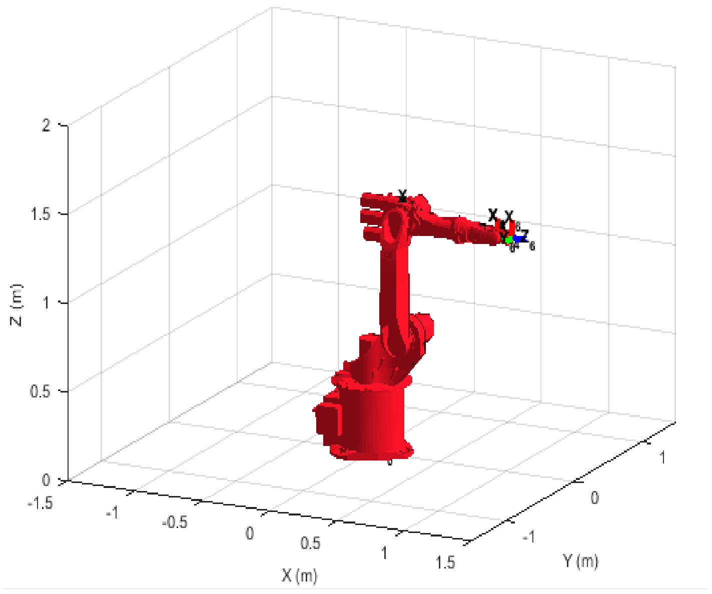


Figure 5.2: KUKA KR6 view.

The forward kinematics are computed using the toolbox function, “directkinematic” if we assume initial values for each joint (6 joints) q = [0.5 -0.5 pi/6 0.1 0.1 0.1]

to compute direct kinematics for this position q

T = directkinematic(robot, q) figure(5.2).

In inverse kinematic it considers there are eight possible solutions for the inverse kinematic problem for most of these robots. not all the eight possible solutions will be feasible for an anthropomorphic 6R robot.

A call the “inversekinematic” for this robot. All the possible solutions are stored at qinv. At least, one of the possible solutions should match q

qinv = inversekinematic(robot, T).

5.2.1 KUKA KR 6 Joints Trajectory Planning:

ARTE toolbox contain some functions and MATLAB commands specified for determine and plot the paths of robot joints position and velocity changes with time based on space trajectory, or Cartesian space trajectory methods.

The aim of the trajectory generation: to generate inputs to the motion control system which ensures that the planned trajectory is executed. The user describes the desired trajectory by some parameters, usually:

• Initial and final point (point-to-point control).

• Finite sequence of points along the path (motion through sequence of points).

The figure (5.3) shows path followed by the manipulator joints, plus the time profile along the path according to input positions (initial and final position).

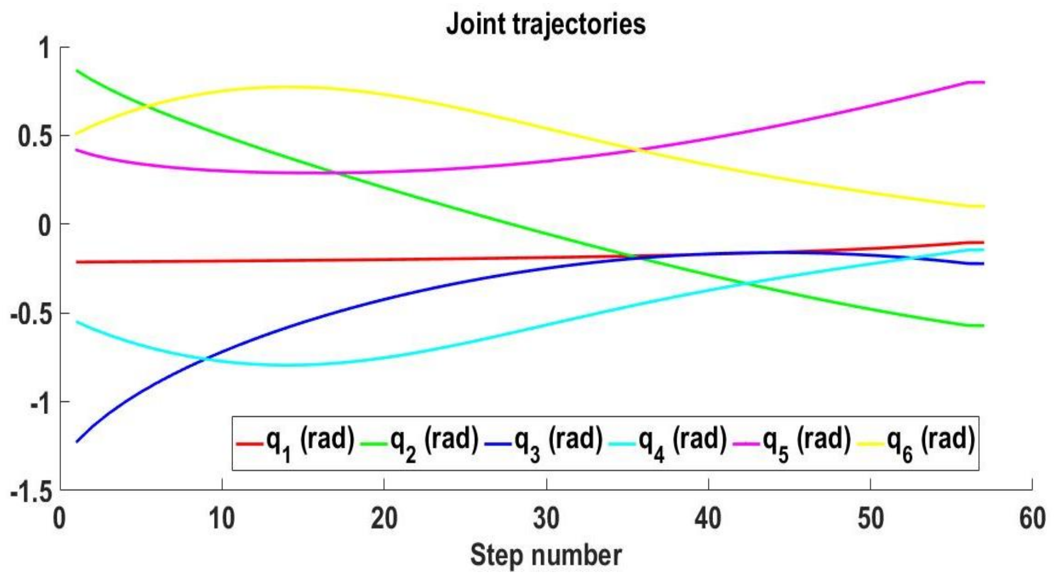


Figure 5.3: the robot joints trajectory.

5.3 Dynamic Modeling and Simulation Results:

The dynamic based on Euler-Lagrange equations and recursive Newton-Euler formulations, as illustrated in previous chapters, then the main equation used to analysis dynamic is:

$$T(t) = D\big(\theta(t)\big)\ddot{\theta}(t) + h\left(\theta(t), \dot{\theta}(t)\right) + c\big(\theta(t)\big) \tag{5.4}$$

According to this equation some parameters assumes to be determine such as inertia matrix as explained in chapter three, the following matrix equation used to find inertia matrix.

$$D_{ik} = \sum_{j=\max(i,k)}^{n} Tr\left(U_{jk} J_j U_{ji}^T\right) \ \text{i, k} = 1,2,\dots,\text{n Inertia matrix, size nxn}$$

$$J_i = \begin{bmatrix} \frac{-I_{xx}+I_{yy}+I_{zz}}{2} & I_{xy} & I_{xz} & m_i\overline{x_i} \\ I_{xy} & \frac{I_{xx}-I_{yy}+I_{zz}}{2} & I_{yz} & m_i\overline{y_i} \\ I_{xz} & I_{yz} & \frac{I_{xx}+I_{yy}-I_{zz}}{2} & m_i\overline{z_i} \\ m_i\overline{x_i} & m_i\overline{y_i} & m_i\overline{z_i} & m_i \end{bmatrix} \text{Inertia Tensor, size 4x4}$$

The following matrix shows the values of inertia tensor matrix

$$\begin{bmatrix} 2,295 & 4,062 & 7,170 & 0 \\ 4,062 & 53,840 & 22,715 & 7,753 \\ 7,170 & 22,715 & 1,111 & 14,195 \\ 0 & 7,753 & 14,195 & 38,767 \end{bmatrix}$$

5.3.1 Forward Dynamic Results Analysis:

The equation represented on forward dynamics is:

$$\ddot{q} = h(q, \dot{q}, \tau, \text{manipulator parameters})$$

In MATLAB ARTE toolbox the function “forwarddynamic” called as shown in MATLAB code [appendix B5] and the result figures following show values for position and joints speed with time for KUKA KR 6 robot as assumed no friction and no torque applied and the initial position and joint speeds as following:

```
q0 = [0 0 0 0 0 0]
qd0 = [0 0 0 0 0 0]
g= [0 0 -9.81]
```

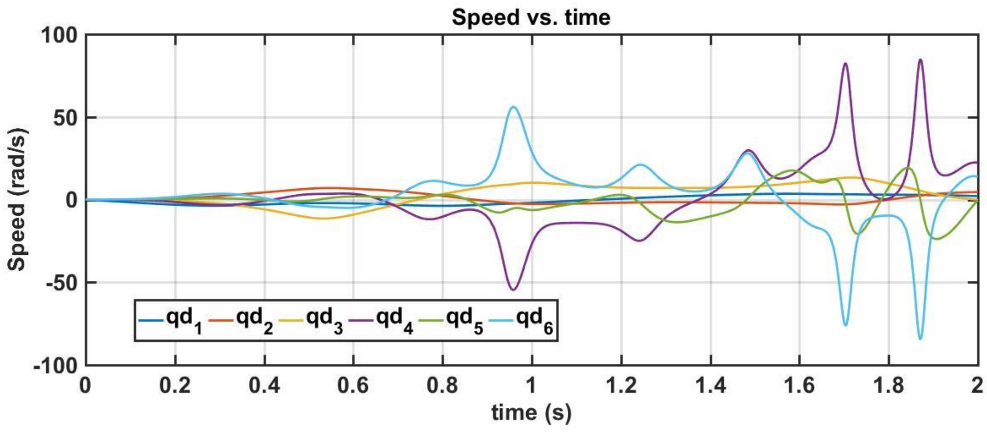


figure 5.4: the values of varying joints speeds with time.

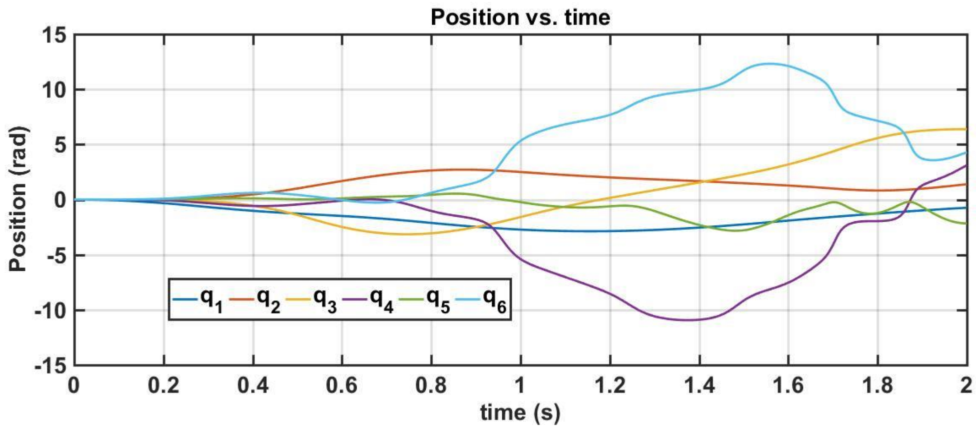


figure 5.5: the values of varying joints positions with time.

As result of applying “forwarddynamic” function the figures (5.4) & (5.5) show the values of varying joints speed with time where (qd1, qd2,…qd6) represent on speed of joints and varying position of joints with time where (q1,q2,…q6) represent on a position of joints.

5.3.2 Inverse Dynamics Results Analysis:

Mathematically, the inverse dynamics problem can be described by a vector equation of the form:

$$\tau = f(q, \dot{q}, \ddot{q}, \text{manipulator parameters})$$

In MATLAB ARTE toolbox the function “inversedynamic” used to Compute inverse dynamics via recursive Newton-Euler. to applying the invers dynamic function, we should choose different positions then compare the result torques for each position as:

q1 = [0 0 0 0 0 0];

q2 = [0 pi/2 -pi/2 0 0 0];

If we choose q = [0 pi/2 -pi/2 0 0 0] as example to test the position and result torque for each joints the following figure show view of KUKA KR6 robot at q = [0 pi/2 -pi/2 0 0 0] as shown in figure (5.6).

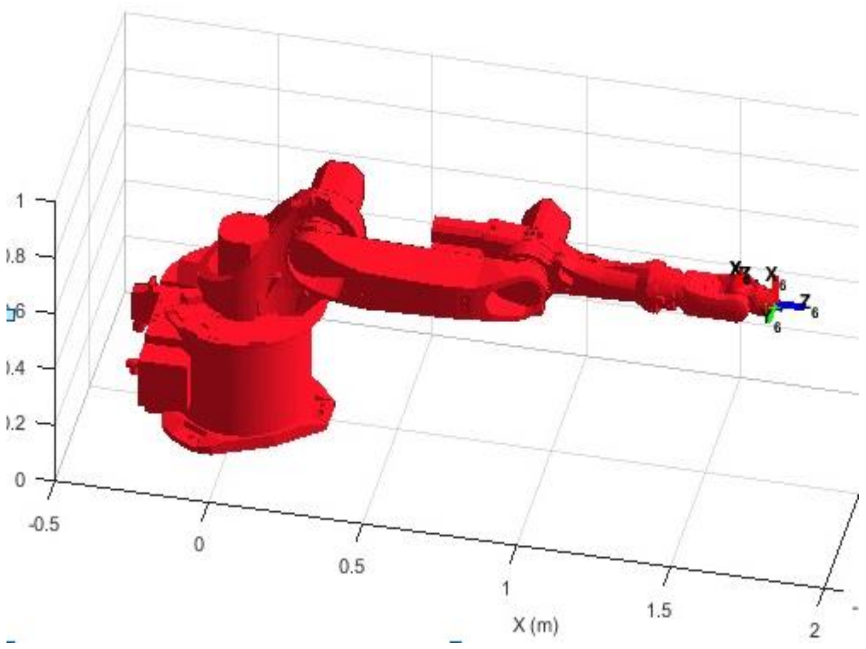


Figure 5.6: robot 3d view for q1= [0 pi/2 -pi/2 0 0 0].

And the following values represent the torque at each joints of the robot tau = 0.0000, 446.0656, 85.7443, 0.0000, 2.1631, -0.0000.

5.4 Robot Programming Tools:

As ARTE toolbox has feature of robot program tool, it is give the ability to test the manipulator robots for any chosen application. In this research the robot tested as aluminum milling machine application where the result of MATLAB Simulink showed in 3d animation.

5.5 Results discussions:

The experimental result obtained by feedback testing showed these solutions are less erroneous and more accurate. In the simulated programming application which with this method has been tested, all the steps have been implemented and therefore the result is based on the accuracy of the models in the simulation environment. Even the results are based upon simulation;
one can conclude that the measurement has enough accuracy for practical usage.

Chapter six

Conclusion and Recommendations

Chapter six

Conclusion and Recommendations

6.1 Conclusion:

In this research modeling and simulation of 6-DOF KUKA KR 6 manipulator robot have been performed using theoretical analysis and ARTE toolbox that work under MATLAB program.

The Denavit-Hartenberg and inertia parameters and also kinematic and dynamic analysis of robot were exploited.

The theoretical approach only gives the forward kinematics values and inverse kinematics values. but to find individual velocity, force, torque values of each link and joint it is complicated.

By using ARTE toolbox in MATLAB we can easily identify velocity acceleration graphs and their values regarding the joints and links and simulation of robot end effector can be done.

Simulation results show that the modeling method is effective and it lays a solid foundation for designing and manufacturing the real assistant robot.

Graphical programming languages like MATLAB can be utilized as powerful tools for simulating a robotic system as it is reported in this research.

With the aid of the MATLAB robotic simulation toolboxes, the core ideas of the coordinate transformation, forward and inverse kinematics, dynamic, control and robot programming are conveyed vividly.

By using ARTE toolbox programing tools the KUKA KR6 tested as aluminum milling machine application so this tool can be used to simulate and animate manipulator robots in many different applications and study their behaviors and kinematic and dynamic properties and get best approaches for development and enhancement the manipulator robots performance.

6.2 Recommendations:

A number of issues with respect to absolute safety, accuracy, cost-effectiveness etc., still remains unaddressed which are up to the engineers to consider, before feasibly manufacturing it.

With this approach, we sincerely hope to simplify the complexity further and attain a complete solution for the kinematic and dynamic of a robotic manipulator using MATLAB and other help tools.

References

1. Díaz, J.F.A., F.A.d.N.C. Pinto, and M.S. Dutra, Kinematical and dynamical models of KR 6 KUKA robot, including the kinematic control in a parallel processing platform. 2010: INTECH Open Access Publisher.

2. Tokhi, M.O. and A.K. Azad, Flexible robot manipulators: modelling, simulation and control. Vol. 68. 2008: Iet.

3. Dombre, E. and W. Khalil, Modeling, Performance Analysis and Control of Robots Manipulators. 2006: Iste.

4. Paul, R.P., Robot manipulators: mathematics, programming, and control: the computer control of robot manipulators. 1981: Richard Paul.

5. Saha, S.K., Introduction to robotics. 2014: Tata McGraw-Hill Education.

6. Sciavicco, L. and B. Siciliano, Modelling and control of robot manipulators. 2012: Springer Science & Business Media.

7. Hayat, A.A., et al. Identification of Denavit-Hartenberg parameters of an industrial robot. in Proceedings of Conference on Advances In Robotics. 2013. ACM.

8. Khatamian, A. Solving Kinematics Problems of a 6-DOF Robot Manipulator. in Proceedings of the International Conference on Scientific Computing (CSC). 2015. The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp).

9. Høifødt, H., Dynamic modeling and simulation of robot manipulators: the Newton-Euler formulation. 2011.

10. Niku, S., Introduction to robotics. 2010: John Wiley & Sons.

11. Balafoutis, C.A. and R.V. Patel, Dynamic analysis of robot manipulators: A Cartesian tensor approach. Vol. 131. 1991: Springer Science & Business Media.

12. Qassem, M.A., I. Abuhadrous, and H. Elaydi. Modeling and Simulation of 5 DOF educational robot arm. in Advanced Computer Control (ICACC), 2010 2nd International Conference on. 2010. IEEE.

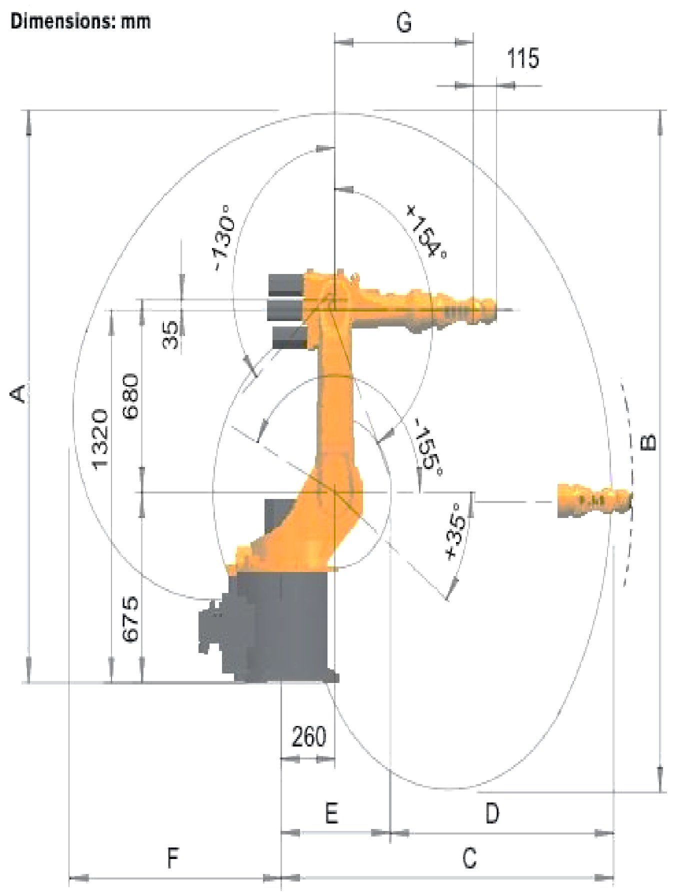
13. Tijani, I.B. Teaching fundamental concepts in robotics technology using MATLAB toolboxes. in 2016 IEEE Global Engineering Education Conference (EDUCON). 2016. IEEE.

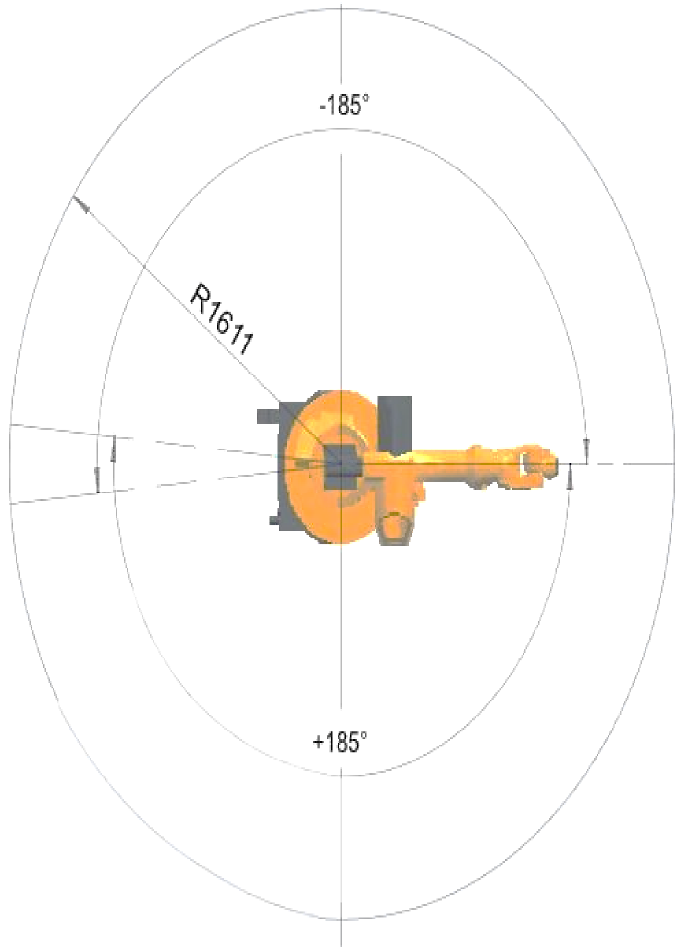
14. Gil, A., et al., Development and deployment of a new robotics toolbox for education. Computer Applications in Engineering Education, 2015. 23(3): p. 443-454.

15. ARTE website (http://arvc.umh.es/arte/index_en.html).

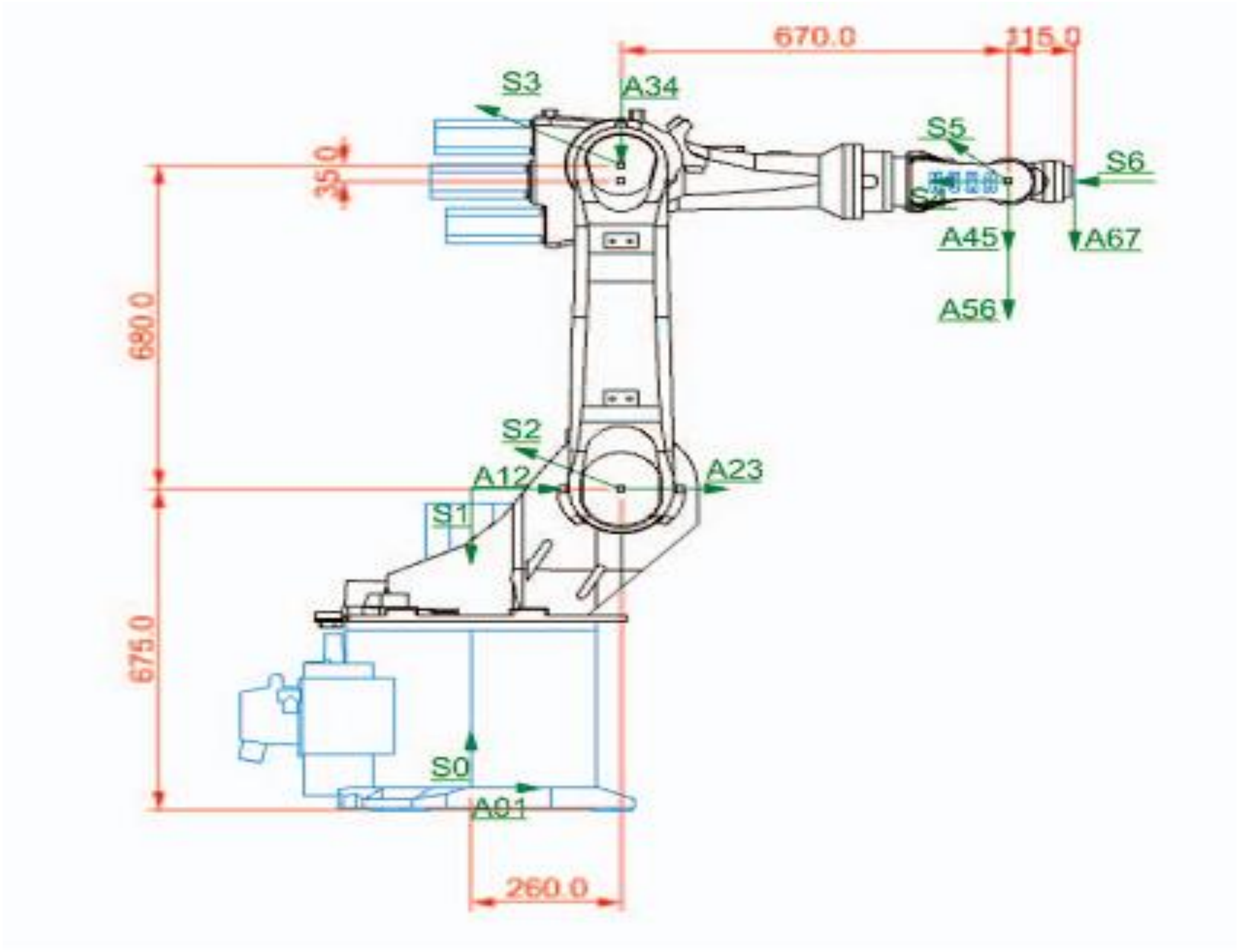
Appendix A: KUKA KR 6 manipulator robot specifications and dimentions

Dimensions: mm





Work envelope	Dimensions[1]							Volume
	A	B	C	D	E	F	G	
KR 6-2	2,026 mm	2,412 mm	1,611 mm	1,081 mm	530 mm	1,027 mm	670 mm	14.5 m³


670.0
115.0
S3
A34
35.0
S5
S6
S4
A45
A67
A56
680.0
S2
A12
A23
S1
675.0
S0
A01
260.0

Appendix B: MATLAB codes:

B1: MATLAB code for Visualization robot

```
function robot = parameters()

robot.DH.theta= '[ q(1)    q(2)-pi/2    q(3)      q(4)    q(5)    q(6)  ]';
robot.DH.d='     [ 0.675   0             0        -0.67     0     -0.115 ]';
robot.DH.a='     [ 0.26    0.68         -0.035      0       0       0   ]';
robot.DH.alpha= '[ -pi/2   0            pi/2      -pi/2    pi/2     pi  ]';

robot.name= 'KR6_2';

robot.inversekinematic_fn = 'inversekinematic_kuka_kr6_2(robot, T)';

%number of degrees of freedom
robot.DOF = 6;

%rotational: 0, translational: 1
robot.kind=['R' 'R' 'R' 'R' 'R' 'R'];

%minimum and maximum rotation angle in rad
robot.maxangle =[deg2rad(-185) deg2rad(185); %Axis 1, minimum, maximum -185 a
185
                deg2rad(-155) deg2rad(35); %Axis 2, minimum, maximum
                deg2rad(-130) deg2rad(154); %Axis 3
                deg2rad(-350) deg2rad(350); %Axis 4: Unlimited (400⁏ default)
                deg2rad(-130) deg2rad(130); %Axis 5
                deg2rad(-350) deg2rad(350)]; %Axis 6: Unlimited (800⁏ default)

%maximum absolute speed of each joint rad/s or m/s
robot.velmax = [deg2rad(156); %Axis 1, rad/s
                deg2rad(156); %Axis 2, rad/s
                deg2rad(156); %Axis 3, rad/s
                deg2rad(343); %Axis 4, rad/s
                deg2rad(362); %Axis 5, rad/s
                deg2rad(659)];%Axis 6, rad/s
% end effectors maximum velocity
robot.linear_velmax = 1.0; %m/s, not specified
robot.accelmax=robot.velmax/0.1; % 0.1 is here an acceleration time

%base reference system
robot.T0 = eye(4);

%INITIALIZATION OF VARIABLES REQUIRED FOR THE SIMULATION
%position, velocity and acceleration
robot=init_sim_variables(robot);



% GRAPHICS
robot.graphical.has_graphics=1;
robot.graphical.color = [255 20 40]./255;
%for transparency
robot.graphical.draw_transparent=0;
%draw DH systems
robot.graphical.draw_axes=1;
%DH system length and Font size, standard is 1/10. Select 2/20, 3/30 for
%bigger robots
robot.graphical.axes_scale=1;
```

```
%adjust for a default view of the robot
robot.axis=[-1.5 1.5 -1.5 1.5 0 2];
%read graphics files
robot = read_graphics(robot);

%DYNAMICS
robot.has_dynamics=1;

%consider friction in the computations
robot.dynamics.friction=0;

%link masses (kg)
robot.dynamics.masses=[25 27 15 10 2.5 1.5];

%COM of each link with respect to own reference system
robot.dynamics.r_com=[0       0          0; %(rx, ry, rz) link 1
                     -0.05   0.006   0.1; %(rx, ry, rz) link 2
                    -0.0203 -0.0141  0.070;  %(rx, ry, rz) link 3
                     0       0.019       0;%(rx, ry, rz) link 4
                     0       0           0;%(rx, ry, rz) link 5
                     0       0         0.032];%(rx, ry, rz) link 6

%Inertia matrices of each link with respect to its D-H reference system.
% Ixx   Iyy Izz Ixy Iyz Ixz, for each row
robot.dynamics.Inertia=[0      0.35 0       0   0   0;
    .13     .524    .539    0   0   0;
    .066    .086    .0125   0   0   0;
    1.8e-3  1.3e-3  1.8e-3  0   0   0;
    .3e-3   .4e-3   .3e-3   0   0   0;
    .15e-3  .15e-3  .04e-3  0   0   0];




robot.motors=load_motors([5 5 5 4 4 4]);
%Speed reductor at each joint
robot.motors.G=[300 300 300 300 300 300];
```

B2: MATLAB code for forward kinematic of the KUKA KR 6 robot:

```
%load robot parameters
 robot=load_robot('kuka', 'kr6_2');



total_simulation_time = 0.5 %simulate for .5 second

%initial position and joint speeds
q0 = [0 0 0 0 0 0]';
qd0 = [0 0 0 0 0 0]';

g=[0   0 -9.81]'; %Z0 axis

drawrobot3d(robot, q0);
adjust_view(robot);

%try both
%tau = [0 0 0 0 0 0]';%no torques applied
tau = [0 200 1 1 1 1]';
%tau = [20 20 21 21 21 21]';

%no friction
robot.friction = 0;

fprintf('\nCOMPUTING FORWARD DYNAMICS (this may take a while)')

%this may take a while, since it requires integration
%of the acceleration at each time step
%[t q qd] = forwarddynamic(robot, total_simulation_time, q0, qd0, tau, [0 0
9.81]);
            %forwarddynamic(robot, time_end, q0, qd0, tau, g, torqfun,
varargin)
[t q qd] =  (robot, total_simulation_time, q0, qd0, tau, g, []);

%animate it!!
animate(robot, q)

figure, plot(t, q), grid, title('Position vs. time')
xlabel('time (s)'), ylabel('Position (rad)')
legend('q_1', 'q_2', 'q_3', 'q_4', 'q_5', 'q_6');

figure, plot(t, qd), grid, title('Speed vs. time')
xlabel('time (s)'), ylabel('Speed (rad/s)')
legend('qd_1', 'qd_2', 'qd_3', 'qd_4', 'qd_5', 'qd_6');
```

B3: MATLAB code for inverse kinematic of the KUKA KR 6 robot

```
%there are eight possible solutions for the inverse kinematic problem for most
of these robots
n_solutions = 8;

%Try different configurations beware that, depending on the robot's topology
%not all the eight possible solutions will be feasible for an antropomorphic 6R
robot.
q=[0.5 -0.5 pi/6 0.1 0.1 0.1]
%q = [0.1 -pi/4 pi/4 0.1 0.1 0.1];

%load robot parameters. You can try different robots
%robot=load_robot('ABB', 'IRB140'); n_solutions = 8;
%robot=load_robot('ABB', 'IRB120'); n_solutions = 8;
%robot=load_robot('ABB', 'IRB1600_6_120'); n_solutions = 8;
%robot=load_robot('ABB', 'IRB1600_X145_M2004'); n_solutions = 8;



%adjust 3D view as desired
adjust_view(robot)

%there are just 2 solutions for these robots and 4 DOF
%q = [pi/2 0.2 0.8 pi/4]
%q = [-pi/4 pi/2 0.5 pi]
%robot=load_robot('kuka', 'KR5_scara_R350_Z200'); n_solutions = 2;
%robot=load_robot('example', 'scara'); n_solutions = 2;
%robot=load_robot('example', '2dofplanar'); n_solutions = 2;
%robot=load_robot('example', '3dofplanar'); n_solutions = 2;
%robot=load_robot('example', 'prismatic');n_solutions = 1; %just one possible
solutions for this case



%draw the robot
drawrobot3d(robot, q)

%Now compute direct kinematics for this position q
T = directkinematic(robot, q)

%Set to zero if you want to see the robot transparent
robot.graphical.draw_transparent=0;

%Set to one if you want to see the DH axes
%abb.graphical.draw_axes=1;

%Call the inversekinematic for this robot. All the possible solutions are
%stored at qinv. At least, one of the possible solutions should match q
qinv = inversekinematic(robot, T);



fprintf('\nNOW WE CAN REPRESENT THE DIFFERENT SOLUTIONS TO ACHIEVE THE SAME
POSITION AND ORIENTATION\n')
fprintf('\nNot that some solutions may not be feasible. Some joints may be out
of range\n')
correct=zeros(1,n_solutions);
%check that all of them are possible solutions!
for i=1:size(qinv,2),
  Ti = directkinematic(robot, qinv(:,i)) %Ti is constant for the different
solutions

    % Note that all the solutions may not be feasible. Some of the joints may
    % be out of range. You can test this situation with test_joints
    test_joints(robot, qinv(:,i));



    %now draw the robot to see the solution
    drawrobot3d(robot, qinv(:,i))

    pause(1);

    k=sum(sum((T-Ti).^2));
    if k < 0.01 % a simple threshold to find differences in the solution
        correct(1,i)= 1;
    else
        correct(1,i)= 0; %uncorrect solution
        fprintf('\nERROR: One of the solutions seems to be uncorrect. Sum of
errors: %f', i, k);
    end
end
%Display a message if any of the solutions is not correct
if sum(correct)==n_solutions
    fprintf('\nOK: Every solution in qinv yields the same position/orientation
T');
else
    fprintf('\nERROR: One or more of the solutions seems to be uncorrect.');
end

%Now, test if any of the solutions in qinv matches q
%find the solution that matches the initial q
%delta is just a squared sum of errors at each of the columns of the matrix
%which store the different solutions of qinv
delta=(repmat(q',[1 n_solutions])-qinv).^2;
i=find(sum(delta,1)<0.01);
if ~isempty(i)
    fprintf('\nOK!: Found a matching solution:\n');
    qinv(:,i)
else
    fprintf('\nERROR: Did not find a matching solution for the initial q');
end
```

B4: Matlab code for joint space and Cartesian space approach

```
robot=load_robot('kuka', 'kr6_2');
%NOA matrix initial point
T1=[0 0 1 1.5;
    0 -1 0 -0.3;
    1 -0 0 1.320;
    0 0.707 0  1.320]
%NOA matrix end point
T2=[0.325 -.776 0.541 0.5;
    0.9 -.42 -0.05 -0.04;
    0.27 0.47 0.839 1.7;
    0.5 0.2 0.12  0.8]

%distancia entre puntos consecutivos
delta = 0.02;

punto_inicial = T1(1:3,4);
punto_final = T2(1:3,4);

v=(punto_final-punto_inicial);
v=delta*v/norm(v); %vector normalizado en la direcci◌n de la
recta
distancia = sqrt((punto_final-punto_inicial)'*(punto_final-
punto_inicial));
%Generaci◌n de puntos en la trayectoria
num_points = floor(distancia/delta);
puntos = punto_inicial;
for i=1:num_points,
    puntos=[puntos i*v+punto_inicial];
end
puntos=[puntos punto_final];
```

```
figure, hold on, grid,
plot3(puntos(1,:),puntos(2,:),puntos(3,:)), title('Trajectory in
space'), xlabel('X (m)'), ylabel('Y (m)')

qs=[];
for i=1:length(puntos),
    T1(1:3,4)=puntos(1:3,i);
    qinv = inversekinematic(robot, T1);

    %select the joint coordinates in qinv which are closest to
the
    %current joint position robot.q
    q=select_closest_joint_coordinates(qinv, robot.q);
    qs=[qs q];
    robot.q=q;%update robot.q here
end

drawrobot3d(robot, qs(:,1))
adjust_view(robot)
drawrobot3d(robot, qs(:,end))

%Now, animate the robot in 3D
animate(robot, qs);

figure, hold, plot(qs(1,:), 'r'),plot(qs(2,:), 'g'),
plot(qs(3,:), 'b'), plot(qs(4,:), 'c'),
plot(qs(5,:), 'm.'), plot(qs(6,:), 'y.'),
legend('q_1 (rad)','q_2 (rad)','q_3 (rad)', 'q_4 (rad)', 'q_5
(rad)', 'q_6 (rad)' ), title('Joint trajectories'), xlabel('Step
number')
```

B5: The following matlab code for forward dynamic for kuka kr6 robot as assumed no friction and no torque applied.

```
%load robot parameters
 robot=load_robot('kuka', 'kr6_2');
total_simulation_time = 0.5 %simulate for .5 second
%initial position and joint speeds
q0 = [0 0 0 0 0 0]';
qd0 = [0 0 0 0 0 0]';
g=[0   0 -9.81]'; %Z0 axis
drawrobot3d(robot, q0);
adjust_view(robot);
%try both
%tau = [0 0 0 0 0 0]';%no torques applied
tau = [0 200 1 1 1 1]';
%tau = [20 20 21 21 21 21]';
%no friction
robot.friction = 0;
fprintf('\nCOMPUTING FORWARD DYNAMICS (this may take a while)')
%this may take a while, since it requires integration
%of the acceleration at each time step
%[t q qd] = forwarddynamic(robot, total_simulation_time, q0, qd0,
tau, [0 0 9.81]);
           %forwarddynamic(robot, time_end, q0, qd0, tau, g,
torqfun, varargin)
[t q qd] =  (robot, total_simulation_time, q0, qd0, tau, g, []);
%animate it!!
animate(robot, q)
figure, plot(t, q), grid, title('Position vs. time')
xlabel('time (s)'), ylabel('Position (rad)')
legend('q_1', 'q_2', 'q_3', 'q_4', 'q_5', 'q_6');
figure, plot(t, qd), grid, title('Speed vs. time')
xlabel('time (s)'), ylabel('Speed (rad/s)')
legend('qd_1', 'qd_2', 'qd_3', 'qd_4', 'qd_5', 'qd_6');
```

B6: sample matlab code to compute torques at each pose(q1&q2)

```
fprintf('\nTorques at each joint given position q1, zero speed and
acceleration, standard gravity acting on Z0')
fprintf('\nComputing static torques at position q1 due to gravity [0 0 9.81]')
tau = inversedynamic(robot, q1, [0 0 0 0 0 0], [0 0 0 0 0 0], [0  0 9.81]', [0 0
0 0 0 0]')
drawrobot3d(robot,q1)
disp('press any key to continue')
pause
fprintf('\nTorques at each joint given position q2, zero speed and
acceleration, standard gravity acting on Z0')
fprintf('\nComputing static torques at position q2 due to gravity [0 0 9.81]');
fprintf('\nPLEASE note the differences with respect to torque tau_2 at q1');
tau = inversedynamic(robot, q2, [0 0 0 0 0 0], [0 0 0 0 0 0], [0  0 9.81]', [0 0
0 0 0 0]')
drawrobot3d(robot,q2)
disp('press any key to continue')
pause
```

B7: Matlab code for the application:

```
function adept_simulation
    global robot
    global qua
    global milling_tool

    % Para que la ejecucion no sea tan lenta
    configuration.delta_time=0.04;

    q=[0 0 0 0 pi/2 0];

    % Configuraci◌n inicial
   robot=load_robot('kuka', 'kr6_2');
    punto=directkinematic(robot, [0 0 0 0 pi/2 0]);
    qua=T2quaternion(punto);

    robot.tool=load_robot('equipment/end_tools', 'milling_machine');
    robot.equipment[1]=load_robot('equipment', 'aluminum_plate');
    drawrobot3d(robot, q);
    robot.graphical.draw_axes = 0;
    robot.tool.graphical.draw_axes = 0;

    adjust_view(robot)


milling_tool=[1,[[robot.tool.TCP(1,4),robot.tool.TCP(2,4),robot.tool.TCP(3,4)],
[1,0,0,0]],[0,[0,0,0],[1,0,0,0],0,0,0]];

    main;
end

function main()
    circleA;
    vel=obtain_joint_speed(robot, speeddata);
    circleB;
    triangle;
    circleC;
```

```
 circleD;
    endpos;
end



function circleA()
    global qua milling_tool

    E=[[0.7 0.2033 0.2; 0.66 0.2433 0.2; 0.7 0.2833 0.2;0.74 0.2433 0.2]];

    RT_tp0=[[E(1,1:2) 0.4],[qua], [0, -1, -1,
0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    RT_tp1=[[E(1,:)],[qua], [0, -1, -1,
0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    RT_tp2=[[E(2,:)],[qua], [-1, -1, -2,
0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    RT_tp3=[[E(3,:)],[qua], [-1, -1, -2,
0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    RT_tp4=[[E(4,:)],[qua], [-1, -1, -2,
0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];

    MoveL(RT_tp0, 'vmax' , 'z100' , milling_tool, 'wobj0');
    MoveC(RT_tp1,RT_tp2, 'vmax' , 'fine' , milling_tool, 'wobj0');
    MoveC(RT_tp2,RT_tp3, 'vmax' , 'fine' , milling_tool, 'wobj0');
    MoveC(RT_tp3,RT_tp4, 'vmax' , 'fine' , milling_tool, 'wobj0');
    MoveC(RT_tp4,RT_tp1, 'vmax' , 'fine' , milling_tool, 'wobj0');
    MoveJ(RT_tp1, 'vmax' , 'fine' , milling_tool, 'wobj0');
    MoveL(RT_tp0, 'vmax' , 'z100' , milling_tool, 'wobj0');
end



function vel=obtain_joint_speed(robot, speeddata)



if strncmp(speeddata, 'vmax',4);
    vel=robot.velmax;
else
    [tag,remain] = strtok(speeddata, 'v');
    vel = robot.velmax.*str2num(tag)/6000;
```

```
end
end

function circleB()
    global qua milling_tool

    E=[[1.025 0.16 0.2; 0.985 0.2 0.2; 1.025 0.24 0.2;1.065 0.2 0.2]];

    RT_tp0=[[E(1,1:2) 0.4],[qua], [0, -1, -1,
0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    RT_tp1=[[E(1,:)],[qua], [0, -1, -1,
0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    RT_tp2=[[E(2,:)],[qua], [-1, -1, -2,
0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    RT_tp3=[[E(3,:)],[qua], [-1, -1, -2,
0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    RT_tp4=[[E(4,:)],[qua], [-1, -1, -2,
0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];

    MoveL(RT_tp0, 'vmax' , 'z100' , milling_tool, 'wobj0');
    MoveC(RT_tp1,RT_tp2, 'vmax' , 'fine' , milling_tool, 'wobj0');
    MoveC(RT_tp2,RT_tp3, 'vmax' , 'z100' , milling_tool, 'wobj0');
    MoveC(RT_tp3,RT_tp4, 'vmax' , 'z100' , milling_tool, 'wobj0');
    MoveC(RT_tp4,RT_tp1, 'vmax' , 'fine' , milling_tool, 'wobj0');
    MoveJ(RT_tp1, 'vmax' , 'fine' , milling_tool, 'wobj0');
    MoveL(RT_tp0, 'vmax' , 'z100' , milling_tool, 'wobj0');
end

function circleC()
    global qua milling_tool

    E=[[0.7 -0.2033 0.2; 0.74 -0.2433 0.2; 0.7 -0.2833 0.2;0.66 -0.2433 0.2]];

    RT_tp1=[[E(1,:)],[qua], [0, -1, -1,
0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    RT_tp2=[[E(2,:)],[qua], [-1, -1, -2,
0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    RT_tp3=[[E(3,:)],[qua], [-1, -1, -2,
0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
```

```
 RT_tp4=[[E(4,:)],[qua], [-1, -1, -2,
0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    RT_tp0=[[E(1,1:2) 0.4],[qua], [0, -1, -1,
0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];


    MoveL(RT_tp0, 'vmax' , 'z100' , milling_tool, 'wobj0');
    MoveC(RT_tp1,RT_tp2, 'vmax' , 'fine' , milling_tool, 'wobj0');
    MoveC(RT_tp2,RT_tp3, 'vmax' , 'z100' , milling_tool, 'wobj0');
    MoveC(RT_tp3,RT_tp4, 'vmax' , 'z100' , milling_tool, 'wobj0');
    MoveC(RT_tp4,RT_tp1, 'vmax' , 'fine' , milling_tool, 'wobj0');
    MoveJ(RT_tp1, 'vmax' , 'fine' , milling_tool, 'wobj0');
    MoveL(RT_tp0, 'vmax' , 'z100' , milling_tool, 'wobj0');
end


function circleD()
    global qua milling_tool


    E=[[1.025 -0.16 0.2; 1.065 -0.2 0.2; 1.025 -0.24 0.2;0.985 -0.2 0.2]];


    RT_tp1=[[E(1,:)],[qua], [0, -1, -1,
0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    RT_tp2=[[E(2,:)],[qua], [-1, -1, -2,
0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    RT_tp3=[[E(3,:)],[qua], [-1, -1, -2,
0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    RT_tp4=[[E(4,:)],[qua], [-1, -1, -2,
0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    RT_tp0=[[E(1,1:2) 0.4],[qua], [0, -1, -1,
0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];


    MoveL(RT_tp0, 'vmax' , 'z100' , milling_tool, 'wobj0');


    MoveC(RT_tp1,RT_tp2, 'vmax' , 'fine' , milling_tool, 'wobj0');
    MoveC(RT_tp2,RT_tp3, 'vmax' , 'z100' , milling_tool, 'wobj0');
    MoveC(RT_tp3,RT_tp4, 'vmax' , 'z100' , milling_tool, 'wobj0');
    MoveC(RT_tp4,RT_tp1, 'vmax' , 'fine' , milling_tool, 'wobj0');
    MoveJ(RT_tp1, 'vmax' , 'fine' , milling_tool, 'wobj0');
    MoveL(RT_tp0, 'vmax' , 'z100' , milling_tool, 'wobj0');
end
```

```
function triangle()
    global qua milling_tool


    E=[[0.8 -0.0866 0.2; 0.8 0.0866 0.2; 0.95 0 0.2; 0.8 -0.0866 0.5]];


    RT_tp1=[[E(1,:)],[qua], [0, -1, -1,
0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    RT_tp2=[[E(2,:)],[qua], [-1, -1, -2,
0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    RT_tp3=[[E(3,:)],[qua], [-1, -1, -2,
0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    RT_tp4=[[E(4,:)],[qua], [0, -1, -1,
0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];


    MoveL(RT_tp4, 'vmax' , 'z100' , milling_tool, 'wobj0');
    MoveL(RT_tp1, 'vmax' , 'fine' , milling_tool, 'wobj0');
    MoveL(RT_tp2, 'vmax' , 'fine' , milling_tool, 'wobj0');
    MoveL(RT_tp3, 'vmax' , 'fine' , milling_tool, 'wobj0');
    MoveL(RT_tp1, 'vmax' , 'fine' , milling_tool, 'wobj0');
    MoveL(RT_tp4, 'vmax' , 'z100' , milling_tool, 'wobj0');
end

function endpos()
    global qua milling_tool


    RT_tpend=[[0.7 0.2033 0.4],[qua], [0, -1, -1,
0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];


    MoveL(RT_tpend, 'vmax' , 'z100' , milling_tool, 'wobj0');
end
```