TRAJECTORY PLANNING FOR 6-DOF ROBOT MANIPULATOR BASED ON OFFLINE ROBOT PROGRAMMING APPROACH

Malnydelage Dimithri Maliyos Fernando

(148456D)

Degree of Master of Science

Department of Electronic and Telecommunication Engineering

University of Moratuwa

Sri Lanka

January 2019

TRAJECTORY PLANNING FOR 6-DOF ROBOT MANIPULATOR BASED ON OFFLINE ROBOT PROGRAMMING APPROACH

Malnydelage Dimithri Maliyos Fernando

(148456D)

Thesis submitted in partial fulfillment of the requirements for the degree Master of Science in Electronics and Automation

Department of Electronic and Telecommunication Engineering

University of Moratuwa

Sri Lanka

January 2019

DECLARATION, COPYRIGHT STATEMENT AND THE STATEMENT OF THE SUPERVISOR

"I declare that this is my own work and this thesis does not incorporate without acknowledgement any material previously submitted for a Degree or Diploma in any other University or institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text".

"Also, I hereby grant to University of Moratuwa the non-exclusive right to reproduce and distribute my thesis, in whole or in part in print, electronic or other medium. I retain the right to use this content in whole or part in future works (such as articles or books)".

Signature: …… Date:

The above candidate has carried out research for the Master's Thesis under my supervision.

Name of the supervisor: Prof. Rohan Munasinghe

Signature of the supervisor: …… Date:

ABTRACT

Industrial robot manipulators are highly involved in modern manufacturing industries. Robot programming is the procedure to carry out generating a sequence of robot instruction. Teaching method is highly applied where a teach pendent is used to generate the robot programme by teaching one point at a time. This process tends to consume more time and the accuracy can be varied depends on the application. Several other methods are used to program robot movement nevertheless industrial applications of these systems are still developing. Programming tends to be difficult and restricts the productivity and industrial application. Hence, requirement of flexible programming methods is still challenging for inexpert robot operators. Trajectory planning for a robot system is still a developing area where the accuracy, productivity and high quality on various operations are highly concerned. To address these limitations, off-line programming systems can be used where computer systems with realistic graphics, interfaces and features can be used to plan and program robot motions without using robot hardware. The research is aimed to present methods for finding a better mathematical way of optimized trajectory planning of 6-DOF industrial robot manipulator. Computer Aided Design software systems are used to implement off-line programming technique by developing human robot interface in order to create robot moving sequence and achieve required data for further calculations. Welding process of machine head cover using a 6 DOF robot manipulator is used to demonstrate and evaluate the proposed method. Methods for Point allocation along the robot moving path and data extraction are presented. Inverse kinematic model for the 6 DOF manipulator is developed and implemented in order to get joint space data represented by joint angles. Derived data is studied to analyze the manipulator motion behavior while moving along predefined path via points allocated. Robot path planning and trajectory planning with CAD system involvement as off-line programming technique is analyzed by comparing results in order to evaluate the performance of the proposed method.

Keywords: Offline robot programming, Computer Aided Design, 6 DOF robot manipulator, inverse kinematics, Human robot interface

ACKNOWLEDGEMENT

The basis of this master thesis is originated by an industrial project which was planned and implemented in MAS Holdings Pvt Limited. I would like to express my sincere gratitude to Mr. Randy Rajarathnam; General Manager- Technical, Mr. Maduranga Pemarathne; Manager-Innovation as the project managers and for giving me technical support to succeed this project. Further, I am sincerely grateful to other members of project team for their support throughout the project and the research.

I am profoundly grateful to my supervisor, Prof. Rohan Munasinghe who has motivated me with intense guidance and great supervision throughout the research. I am also grateful to course coordinator of the MSc programme in Electronics and Automation; Dr. Chamira Edussooriya for the guidance and advises given throughout the research. I am thankful to Mr. Damith Kandage; course assistant for providing assistances given throughout the whole duration of the research and finally I would like to thank all the academic staff members and supportive staff members who supported and inspired me by supporting whenever needed throughout the research.

Malnydelage Dimithri Maliyos Fernando

B.Sc. Eng. (Moratuwa)

Department of Mechanical Engineering,

University of Moratuwa,

Katubedda, Sri Lanka,

January 2019.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

Abbreviation	Description
DOF	Degree Of Freedom
CAD	Computer Aided Design
3D	Three Dimensional
OLP	Off Line Programming

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

Chapter 1 explains about background and motivation of the research. It outlines the foundation of the research initialization as a master thesis. The research is aimed for finding optimized solutions for trajectory planning for 6 DOF robot manipulator based on offline programming approach. Section 1.1 discusses the thesis background and motivation. Problem definition and the objectives of the thesis are defined in section 1.2. Section 1.3 describes research limitations. Research influences over the industry is discussed in section 1.4. Finally, Section 1.5 summarizes the remaining chapters of the report.

1.1 Background and Motivation

Industrial robots are highly involved and utilized in modern manufacturing industries due to its efficiency, productivity and programmability. The demand for the usage of robots in automated manufacturing systems is increasing especially in automotive systems. Performing high precision tasks with higher repeatability and quality makes them more utilized in the industries. Industrial robots can be operated continuously without taking a break. This supports manufacturers to increase the productivity and efficiency.

There are several types of industrial robots and can be simplified to five major types.

1. Cartesian
2. cylindrical
3. Delta
4. SCARA
5. Articulated robots

Each robot type has different features and specific elements which utilize them more appropriate for different applications. Main differentiators among them are speed, size and workspace.

Articulated robots which resemble a human arm in its mechanical configuration are the most commonly utilized industrial robots since the design and the configuration offer most flexibility. Articulated robot has connecting links which are connected to the base with rotary joints. Figure 1.1 shows an illustration of six degree of freedom industrial robot manipulator manufactured by Denso robotics.

Figure 1.1: Denso robotics six degree of freedom robot manipulator model: VP 6242

6 DOF robot manipulator has many advantages such as high speed operation, large work area for minimum floor space and ability to align to multiple planes. Even though it has significant advantages, challenges are being confronted since it is complicated in programming and kinematics.

Robot programming is the procedure to carry out creating a sequence of robot working instruction and work location/points that achieves the required task. Various methods and techniques are used to plan and program robot operations. Teaching method is highly applied where a teach pendent is used to generate the robot programme by teaching one point at a time. This process tends to consume more time and the accuracy

can be varied depends on the application. In some applications, technical expertise may be required to perform the task like welding, spraying etc.

Several other methods are used to program robot movement such as,

- software dedicated to a particular industrial process.
- human body attached sensors to capture arm movements
- CAD based solutions.
- vision-based interfaces.

Due to several reasons including reliability issues, applications of these systems are still developing. Thus, the teach pendant is considered as a common robot input device which grant access to robotic functionalities. The difficulty of this robot programming process limits the productivity of the robot and more widespread use of robot technology. Hence, requirement of flexible programming methods is still challenging for inexpert robot operators.

Currently CAD systems can provide more design and modelling capabilities having high precision and standard, simulation can be also done with great accuracy. Therefore, CAD systems can be utilized to give effective engineering solutions for preparing robot path and trajectories in robot programming.

Off-line programming (OLP) is introduced as a revolutionary robot programming method where the robot program can be created without considering the actual robot cell which represent the robot system and the application. The created robot program can be imported or uploaded to the robot controller for the robot manipulator execution. Simulator can be used to create the robot cell by creating comprehensive graphical three-dimension(3D) model. Currently, robot integrators use robotic simulators and off-line programming techniques to create efficient program paths for a particular robot manipulation to perform a specific task. Simulators are beneficial where robot movement and reachability analysis, collision detection and cycle time calculation can be done when robot programs are simulated.

Off-line robot programming is used to develop the robot program on an external computer system without interfering the production or the robot operation. This method improves the on-line programming abilities where programmers use teach pendants to program the robot manually. Robot programming change over time can be minimized by applying off-line robot programming techniques.

The capability to perform different operations and task is more important when considering the flexibility of industrial robot manipulators. Robot system is subjected to deal with higher degree of problems to resolve with respect to human flexibility. When considering an object to move between two space points, several factors are needed to be concerned.

1. Finding Optimum path or route.
2. Avoidance of obstacles and collisions.
3. Improving task productivity.
4. Maintaining higher efficiency.

Path planning is more important where planning of entire path from one point to another point in work space including stopping in predefined path points is executed. Path planning is a geometrical description of robot motion. Obstacles and path constraints can restrict the motion of a robot.

Considering obstacle constraints, Path planning can be done assuming that robot has to be moved by planning not moving through that obstacle takes place. Path constraints are needed to be concerned where there can be reference points that the robot must move through that points.

In the case of 6 DOF robot manipulator, physical limit of motion and mechanics constraints are appeared in most cases. Trajectory planning is planning of desired motion of a robot manipulator and depicts how well the robot manipulator operates. For optimal solution, actuator positions, velocity, acceleration, jerk and the limit of joints are needed to be considered. Energy expenditure while running the robot is more concerned. In every movement, robot has to accelerate, hold and brake hence energy is consumed. Therefore, unnecessary energy dissipation should be reduced in order to increase the energy efficiency of the manipulator. Speed is more concerned where

productivity of the robot manipulator is highly focused. Cycle time for the task is important in a high production rate hence integrator try to minimize as much as possible. Therefore, time taken for a path is more important to be considered. Path planning and trajectory planning should be highly concerned in order to get higher productivity while maintaining better robot performance.

1.2. Problem Definition

Robot manipulators are highly involved in current manufacturing industries and for planning and programming of the robot manipulator, teaching method is highly used. It is more suitable for simple operations where programmers need less time for take necessary point locations in robot moving path. In some critical operations such as welding, painting etc. time taken for teaching is much higher than usual since there can be higher number of task points hence programmer need significant time to teach and the experience and expertise is more important. If any modification has to be done, programmers face numerus problem where in some cases, the teaching of the robot path has to be done from the beginning. Offline robot programming is applied to overcome this kind of issue. Dedicated software solutions are available but can be expensive and limited to particular operations.

Computer Aided Design software is highly used in manufacturing industries since higher accuracy level can be achieved. User friendliness is the other factor that users involve CAD systems in their operations. CAD systems can be involved to develop human robot interface hence programmer can define robot path while analyzing the system graphically. Path planning can be done effectively and relevant data extraction from CAD system can be used.

Inverse kinematics of 6 DOF robot manipulator is needed to find joint space values for particular robot position in work space. Inverse kinematics is studied for many decades and solving of inverse kinematic for a particular robot system is difficult computationally and time to solve is higher. Effective inverse kinematic solution is more important to find joint angles since the actuator works in joint space. Trajectory

planning is important to improve the productivity of the robot manipulator operation. Optimization of the trajectory planning enhance the production while maintain the robot work cycle smoothly.

1.2.1 Thesis Objectives

Thesis main objective is to research and develop methods to involve CAD systems as offline robot programming technique to improve and optimize the trajectory planning of 6 DOF robot manipulator by implementing enhanced inverse kinematic model for the manipulator. In this research, methods for path planning and relevant data extraction in CAD systems, inverse kinematic model for the robot manipulator and trajectory planning optimization with CAD systems are investigated. Proposed method is validated by implementing for a practical robot application and comparing the results and its performance.

1.2.2 Goals

In this research, it is intended to present methods for optimizing trajectory planning using offline robot programming techniques in order to increase the productivity and accuracy. This research intends to find a better mathematical way of optimized trajectory planning of 6-DOF industrial robot manipulator.

1.3 Limitations

Proposed method is based on the controlling and operation of 6 DOF robot manipulator. Therefore, the development of the inverse kinematics model is limited to the type of 6 DOF manipulator which is intended to research on. Welding operation of a machine head cover using Denso VP 6242 robot manipulator is evaluated in order to apply the proposed method and analysis the result and performance. Inverse kinematic

model is subjected to change with types of robot manipulator and the work space where the robot end effector operates.

1.4 Influences to the industry

This research work demonstrates an approach to develop methods to optimize trajectory planning in order to increase the efficiency and productivity with offline programming techniques. The proposed method can be implemented for most common available 6 DOF manipulators in the industry and can be applied for other types of robot manipulators. The method uses for path planning including path creation, task point allocation and data extraction in CAD systems can be used for any kind of industrial operation for further development. Trajectory planning can be analyzed and path planning methods can be optimized according to trajectory planning requirements in order to reduce task cycle time while maintaining robot smooth movement. Integrators can develop other methods based on this proposed method for further development.

1.5. Report summary

The remaining sections is arranged as follows.

In Chapter 2,some literature is reviewed about existing methods for solving kinematics of robot manipulators, optimizing of trajectory planning for different applications and influence of current offline programming methods for best robot operation. Chapter 3 illustrates the behavior of the robot and the application which is evaluated. It also extends the background of the system that will support to develop methods for optimum solutions. In Chapter 4, an approach for developing appropriate methods for optimizing trajectory planning of 6 DOF robot manipulator is proposed. Path planning with CAD system support, mathematical model for inverse kinematic solution of 6 DOF robot manipulator and analysis of trajectory planning from derived joint space data in order to optimize of the robot programmability and operability are proposed.

Simulation model is developed in a separate robot programming software tool in order to verify the developed mathematical model results and debug for further development in chapter 5. The final results of the research are described in chapter 6 and it validates the results over the results of the simulation. It also suggests improvements as forthcoming developments. The development procedure of the mathematical model and algorithms for inverse kinematic solution of 6 DOF robot manipulator and trajectory planning are illustrated in appendix 'A'. Appendix 'B' shows the dimensions and the configuration of the Denso VP 6242. It also presents the dimensions of the work piece and the robotic cell layout. Appendix 'C', 'D' and 'E' illustrate calculated joint angles values tabulated for experiment scenario 1,3 and 4.

CHAPTER 2

LITERATURE REVIEW

Existing approaches developed for inverse kinematic solutions, path planning and trajectory planning of robot manipulators and other similar applications are discussed in this chapter. Robot manipulators are highly involved in modern manufacturing industries specially in automotive [1]. Robot programming is critical significantly in order to operate productively and various programming techniques are used. Currently sophisticated robot controllers and software interfaces are involved [2] and most of these applications are expensive in use thus teaching method is highly used. Teaching method can be imprecise, time consuming process in some applications where technical expertise of the application should be concerned in order to perform robot teaching [3]. Offline robot programming methods are introduced as a revolutionary solution for the drawbacks of current online programming techniques including teaching where the robot programs are created in separate interface independent of actual robot cell prior to upload to the robot manipulator for execution [4]. Importance of the Path planning and trajectory planning is more valuable for robot operations and maintenance in order to increase the productivity and efficiency.

2.1 Some Literature

Even though there are many types of industrial robot manipulators and systems which are capable of offline robot programming, it is still very expensive and limited to a particular industrial application which tends to cost more when changing the application. In addition, the applications where the accuracy is highly concerned need advanced robot systems with controllers [1][2].

A. Paulo Moreira ,Pedro Neto and J. Norberto Pires researchers in University of Coimbra, Portugal and University of Porto, Portugal have presented CAD-Based Off-Line Robot Programming approach in order to optimize robot programmability [3].

Currently CAD systems can provide more design and modelling capabilities having high precision and standard, simulation can be also done with great accuracy. Therefore, CAD systems can be utilized to give effective engineering solutions for preparing robot path and trajectories in robot programming.

L. Alonso Ferreira, M. Álvarez Souto, I. Fernández Iglesias and Y. Lapido Figueira, researchers present solution for offline robot programming method with CAD support to support ship building fabrications and supports [5]. They proposed a solution for a hyper-flexible welding cell with 6 DOF robot manipulator mounted on 3 axis gantry system which is programmed in a CAD environment [6] in order to prepare robot task sequences and extract relevant data for further processes. Figure 2.1 shows the application area of hyper flexible robotic cell they have implemented.

Figure 2.1 Hyper flexible robotic cell

It also emphasizes the capability of creating accurate robotic cell for complex robotic system arrangements in a CAD system which gives unexperienced programmers to program robot tasks easily.

Kinematics of a robot manipulator is a very critical problem when we deal with automatic control of the robot operation. A. Khan, C. Xiangming, Z. Xingxing and W.

Quan present method and solutions of closed form inverse kinematic modelling for a 6 DOF robot manipulator [7]. The manipulator is based on Puma560 robot which is intended to move in underwater area. They have developed a closed-form solution of inverse kinematics model for a 6-DOF manipulator and were able to validate the developed algorithm using simulation in Robotic toolbox.

Mustafa Jabbar Hayawi from Thi-Qar University has presented an Analytical Inverse kinematics Algorithm of A 5-DOF Robot Arm. Forward kinematics and a closed form inverse kinematic solution for the educational 5 DOF robot manipulator TR 4000 are presented to overcome the high number of iterative numerical solution [8].

Trajectory planning is very essential to maintain Smoothness and Ease of accurate tracking by the manipulator. Non-smooth trajectories cause problems like high torque in actuators, Vibrations, Error in path tracking, manipulator wear and low level of quality output [9]. Jerk controlling is very important in order to minimize these problems since the nature of the jerk profile predicts the behavior of the motion. Studies shows that a trajectory with controlled jerk profile is essential in order to achieve the desired features. Controlled jerk trajectory can be achieved by,

1. Maintaining zero bound starting and ending profile which reserve the smoothness at start and end motion.
2. Maintaining continuous jerk profile to avoid potential 'Infinite jerk' of non-continuous jerk profiles.
3. Limiting jerk between known values where user can compare with manufacturer's recommendations.

Trajectory planning can be executed in joint space or operational space. In joint space trajectory planning, the motion is described by joint values whereas in operational space trajectory planning, the motion is described in Cartesian space in many cases. The motion between two points is unpredictable in joint space trajectory planning while it is easy to visualize the path and the motion between the two points is known in Cartesian space trajectory planning. Collision can be prevented by proper trajectory planning in operational space but it is computationally expensive since we need to

solve inverse kinematics between discretized points along initial point to final point at each step. In joint space trajectory planning, inverse kinematics is needed to be calculated only once and constraints like joint angle and velocity can be concerned [10].

CHAPTER 3

ROBOT AND APPLICATION SYSTEM

This chapter explains the behavior of robot system and the application. Section 3.1 presents information about the robot system with relevant sub systems required for the operation. Section 3.2 presents information about the application which is evaluated by implementing proposed methods in order to optimize the output. Section 3.3 expresses the necessary details about the total system represented the entire robot working cell for the application.

3.1 Robot system

Robot system is consisted of two major sub systems.

1. Denso VP 6242 robot manipulator
2. Robot controller system

Denso VP 6242 robot [Figure 3.1] is a high speed, high accuracy robot unit produced by DENSO Robotics. It is categorized as mini sized vertical articulated robot.

Figure 3.1: Denso VP 6242 mini sized vertical articulated robot

Denso VP 6242 arm is a 6-axis type robot arm built with joints similar to a human arm, great flexibility can be achieved. 6-axis freedom of movement makes them suitable to handle a much wider application range such as assembly, Dispensing, Grinding, Laser welding, material handling, material removal, packaging, pick and place, ultrasonic welding, polishing, spot welding.

It is suitable for installations where operational space is limited. It has excellent repeatability of ±0.02 mm. It can handle payloads up to 3 kg and can reach up to 432mm maximum [Figure 3.2]. It can be ceiling or ceiling mounted with no special hardware needed [Figure 3.3].

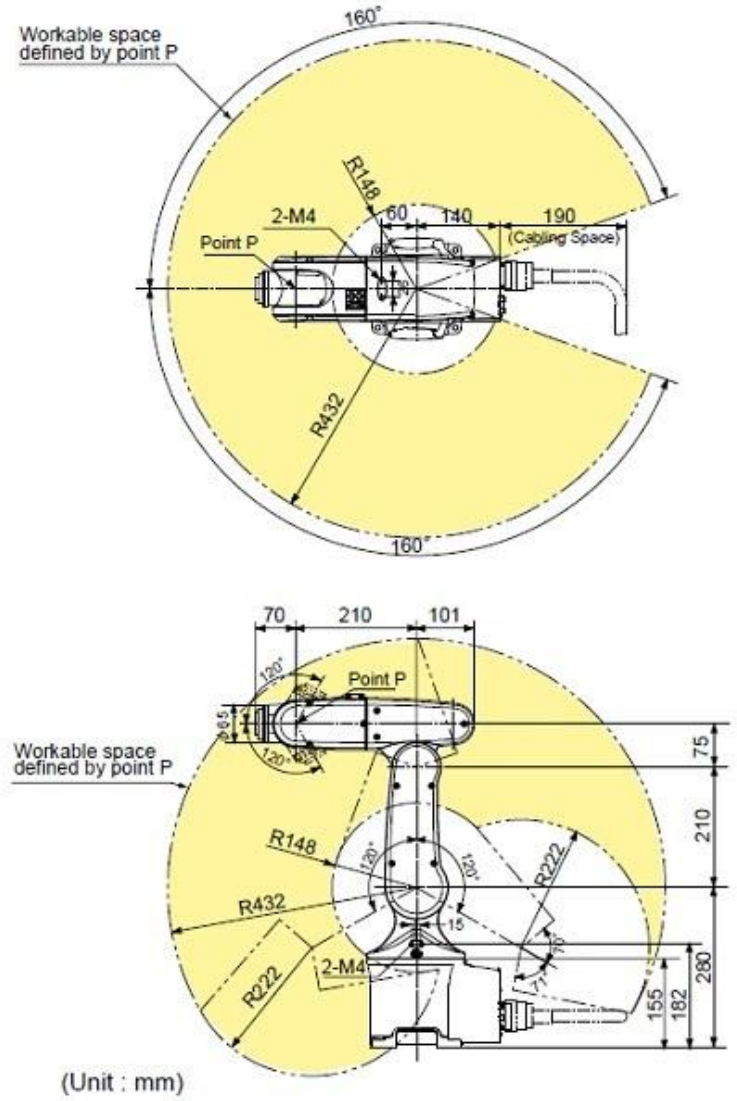


Figure 3.2: External Dimensions and Workable Space (VP-6242)

Figure 3.3: VP 6242 robot arm ceiling or ceiling mounted

VP 6242 robot arm is consisted of six joints operated with AC servomotors and brakes for all axes. Each joint has motion and speed limitation and table 1 presents the range of motion and speed limitation.

Table 1: Joint movement and speed limitation [VP 6242]

Joint	Range of motion (°)	Maximum joint speed (°/sec.)
J1	±160	250
J2	±120	187
J3	+19, +160	250
J4	±160	300
J5	±120	300
J6	±360	300

Source: Denso VP 6242 user manual [11]

Model RC8A and RC8 are the controller compatible with the VP 6242 robot system. It features and supports the Safety Motion function [Figure 3.4].

Figure 3.4: RC8A controller for VP 6242 robot system

Robot controller can be connected many sub systems such as control devices, software and other peripherals for further expansions [Figure 3.5].

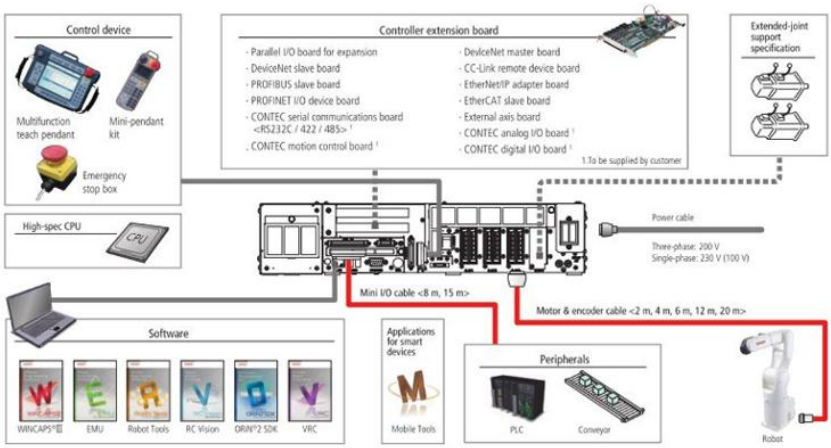


Figure 3.5: RC8A controller System configuration

The controller is the main controlling device of the industrial robotic arm which allows robot system and parts to function together. It also allows other systems to be connected with the current systems. The controller runs a set of instructions written in code named a program. The program is inputted or entered with a teach pendant [Figure 07] or software interface that built on most of operating systems. It converts the commands in the source code of the program to motion or motor drivers which are connected with robot arm joints. Internal model of kinematic structure of the robot

manipulator is inbuilt with the controller. It can coordinate the motions precisely which are commanded by each individual robot arm motor drives. Thousands of parameters are referred by the controller to ensure the robot operating precision of the required task enrolled with the application.

Wincaps III software interface is a programming tool developed by Denso corporation and it is used for various application including program developments, parameter settings, transferring relevant data between robot arm and the controller and robot posture checking on a 3D screen. Robot can be programmed in a separate area called program editing window and functions such as line number display, command color display, indentation, comment block and bookmark can be implemented. Simulation capability is more advantageous where programmer can run the program on a computer system. Cycle time measurement, interference can be checked by user prior to transfer to the controller. Program start/stop and break points with robot motion with robot trajectory display are beneficial in order to increase the productivity. 3D arm view is the capability of the displaying robots and peripheral devices three dimensionally when robot motion is simulated [Figure 3.6]. 3D graphic data, format of VRML, DirectX can be imported in to Wincaps interface. Programmer can obtain location data information [Figure 3.7] of the robot end easily in order understand and get relevant data for further calculations.

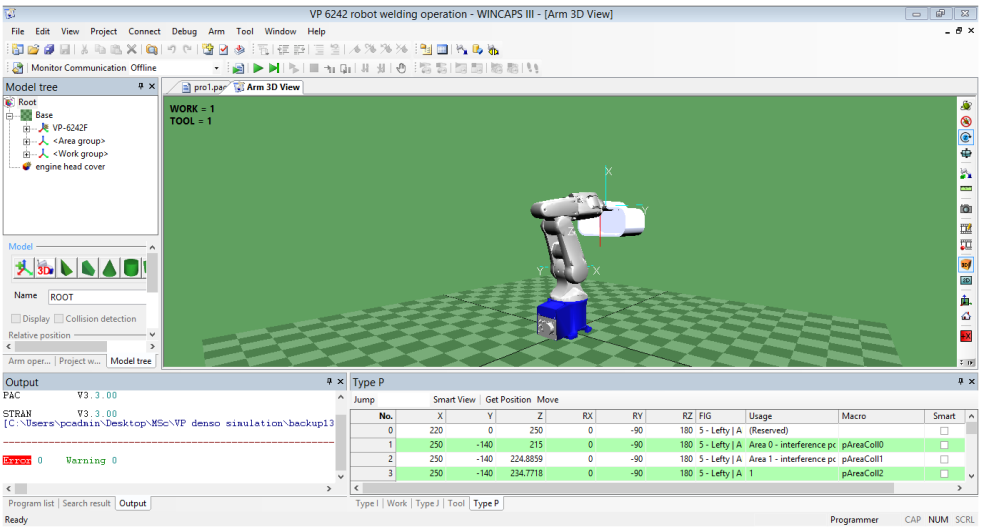


Figure 3.6: 3D arm view – Wincaps III software

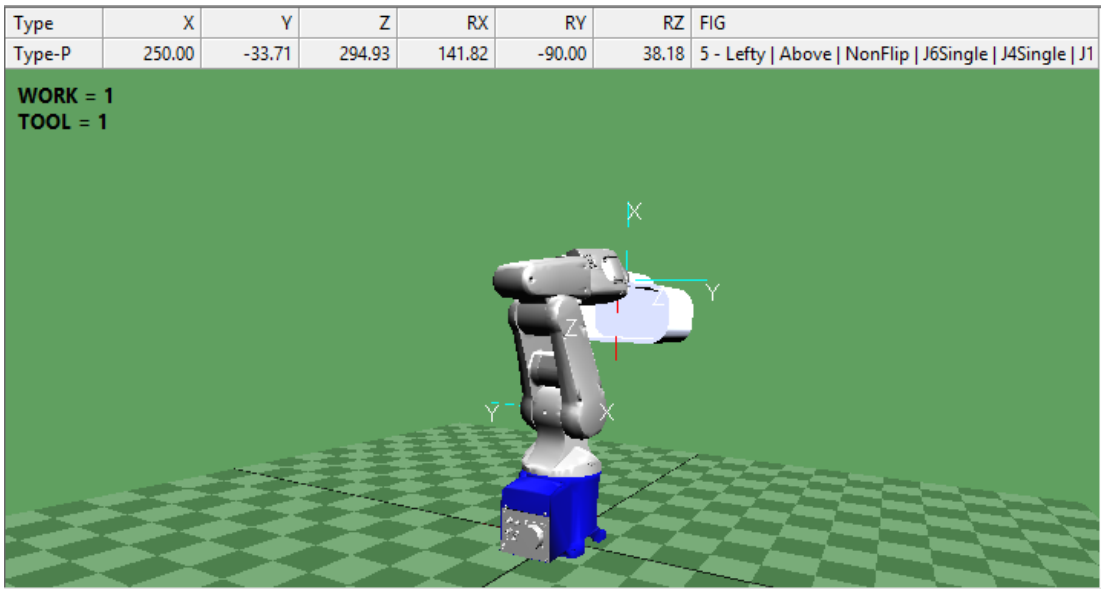


Figure 3.7: Data location data – Wincaps III software

3.2 Automated welding operation

Robot manipulators are highly involved in modern manufacturing and fabrication industries such as arc welding and spot-welding operations are included [Figure 3.8].

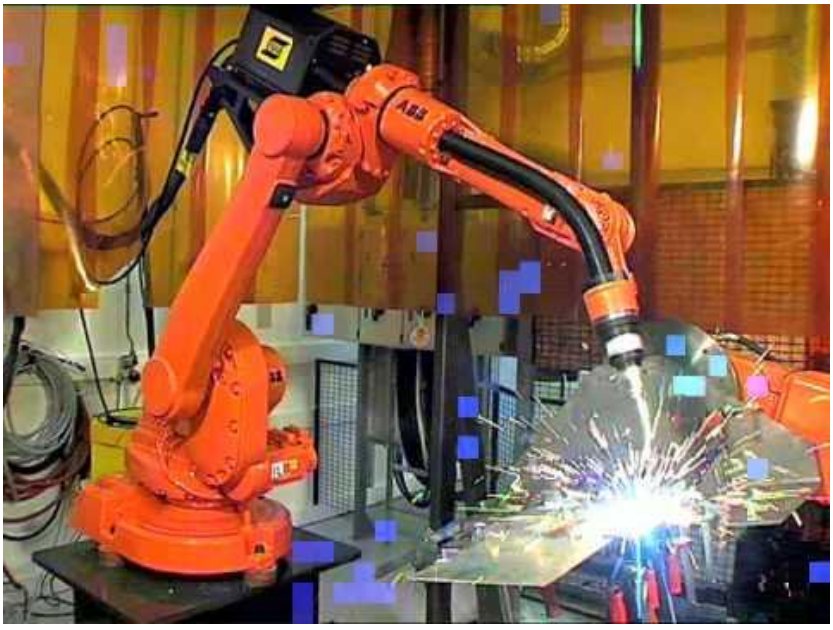


Figure 3.8: Robotic welding operation – ABB robotics [12]

For welding operation, smooth movement with constant speed of the welding torch is highly important to get higher quality welding output. Due to higher temperature at the end of the welding torch, even a small jerk at a point in the moving line causes a huge impact to the output [Figure 3.9].

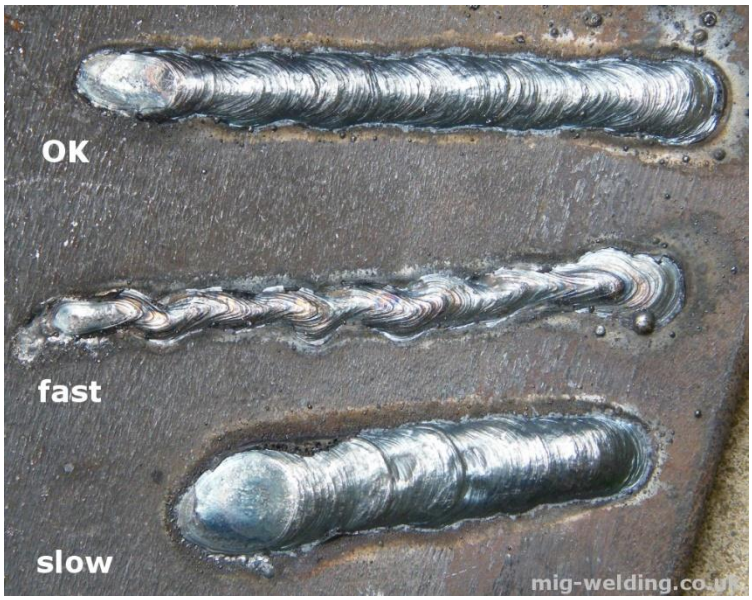


Figure 3.9: Arc welding defects due to speed changes [13]

Robotic welding operated system consists of two major parts.

1. Robot manipulator and controller system
2. Welding torch for robotic operations

Robot manipulator is used to handle the welding torch and operated as a human arm when the welding operation is done manually. Specially developed welding torch is now available for automated robotic operation since the conventional welding equipment cannot be used for robotic operation due to complexity of system components. Robotic welding torch can be attached to the robot end as an end effector without need of much effort. Figure 3.10 illustrates the differences of the design of conventional welding torch used for manual welding and robotic welding torch.

Figure 3.10: Differences of welding torch unit between Manual welding and Robotic welding [14]

3.3 Robotic welding Robot cell

Robotic welding operation done for fabricating a machine head cover by welding the edge of the head cover is evaluated as a research model in order to apply and test developed methods for trajectory planning with CAD based offline programming approach for a robot manipulator. Three-dimensional (3D) design of the overall system and dimensions are illustrated in Figure 3.11 and Figure 3.12.

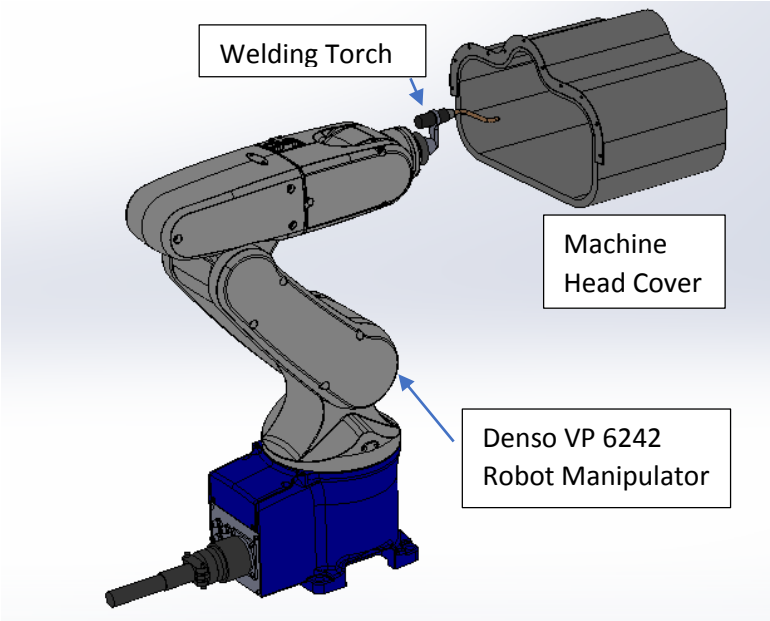


Figure 3.11: Robotic welding application – 3D model

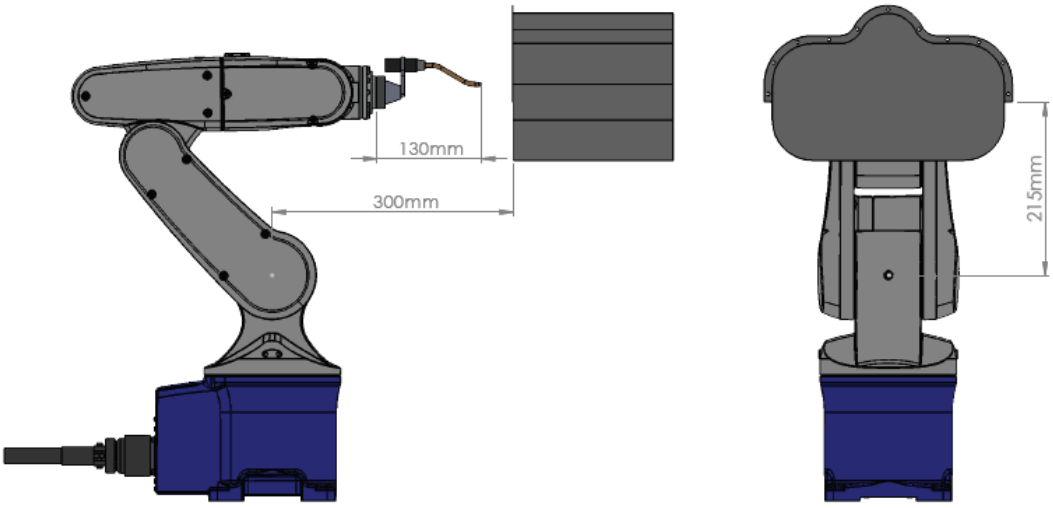


Figure 3.12: Welding application - Layout

Figure 3.13 illustrates the dimensions of the machine head cover which is confronted for welding.

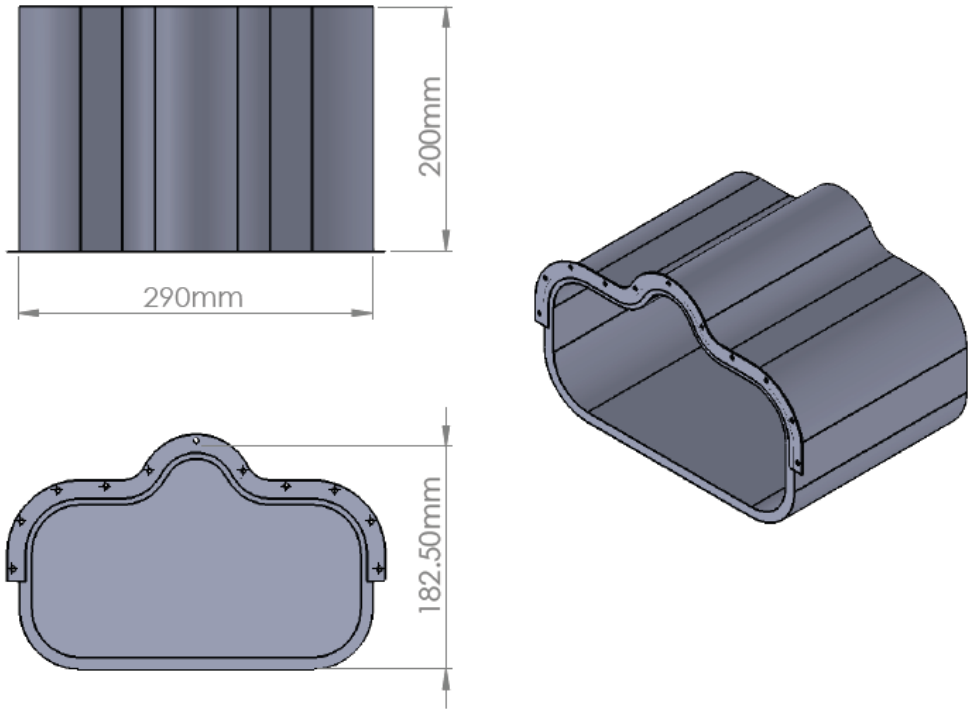


Figure 3.13: Machine Head Cover - Dimensions

Figure 3.14 shows a sample industrial Robotic welding torch used in the industry [15]. Figure 3.15 illustrates the dimensions of the welding torch attached to end of the robot end as an end effector.

Figure 3.14: Industrial Robotic welding torch [15]

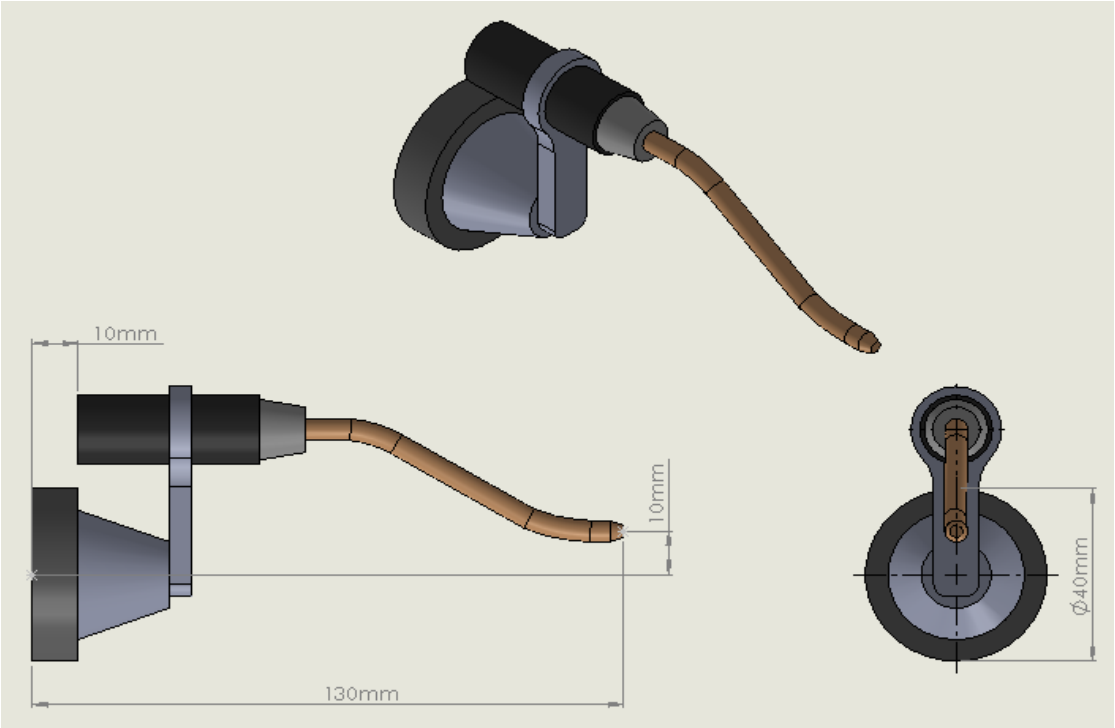


Figure 3.15: Robotic Welding Torch - Dimensions

Figure 3.16 shows the robot end moving path along the cover edge which is followed by the robot end effector to get proper welding operation.

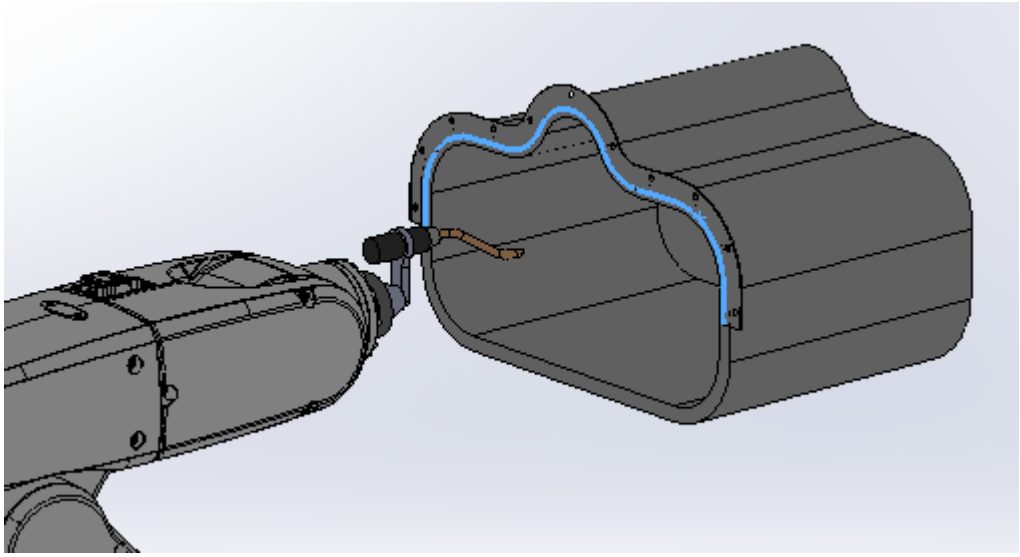


Figure 3.16: Robot End travelling path (marked in blue color)

CHAPTER 4

SYSTEM AND PROCESS DESIGN IDENTIFICATION

This chapter describes development approach over optimized methods of finding inverse kinematics of 6 DOF robot manipulator and trajectory planning done according to path planning using CAD systems. Section 4.1 illustrates the methods and mathematical approach for finding optimized inverse kinematic solution for the robot manipulator in order to find accurate joint space data of the robot while operating along the path. Section 4.2 illustrates the process of creating the robotic cell or the human robot interface in CAD environment where programmer can develop precise virtual model of the robot and the application in a software. Section 4.3 proposes methods to path planning for the particular application and information extraction in order to create robot moving sequences. Section 4.4 proposes the mathematical approach for mapping trajectories according to speed limitation with respect to path planning.

4.1 Mathematical approach for closed form solution of inverse kinematic of 6 DOF robot manipulator

Inverse kinematics problem of robot manipulator is very essential to solve to find joint angles of each joints of the robot locating at a position and orientation of the robot end effector. Robot users work in the cartesian space but the robot operates in the joint space. Therefore, inverse kinematics is defined as transformation from cartesian space to joint space. Finding joint angles is more important as robot end position is needed to be located precisely. Solving inverse kinematic is the practical complication of manipulator control and need numerical methods to solve. Inverse kinematics is more complex comparatively and numerous solutions may exist for the identical robot manipulator posture. Closed form solutions cannot be derived always since trigonometric nonlinear simultaneous equations are consisted with inverse kinematics of robot manipulator. There may not always exist solutions for inverse kinematics for a particular range of robot end effector posture. Numerical methods are used to derive

inverse kinematic solutions to find joint space values when kinematic equations are not possible to solve analytically [16].

Denso VP 6242 Robot is a 6 DOF robot manipulator consists of six revolute joints at each links. Representation of the Denavit–Hartenberg (D-H) model can be used to model connections of the robot links and joints [17]. It is needed for finding solutions for forward kinematics and inverse kinematics.

The base of the manipulator is link 0 and not considered one of the six links generally. Link 1 is connected to the base link by joint 1. Links are maintaining a fixed relationship with joints at each link end. The common normal distance is a_i(length) and α_i (twist) is the angle between the axes in a plane perpendicular to a_i. The distance between each joint is denoted as d_i and the angle between normal of each joint is denoted as θ_i. Coordinates can be assigned for the VP 6242 robot as illustrated in Figure 4.1.

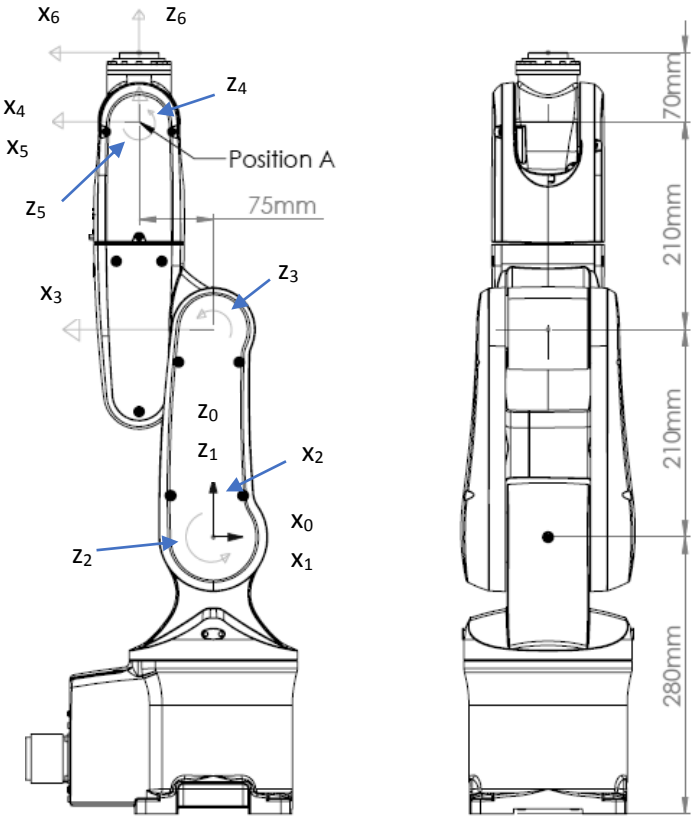


Figure 4.1: Coordinate assignment – Denso VP 6242

According the coordinates assigned for the robot manipulator, relevant parameters can be assigned [Table 2].

Table 2: Denso VP 6242 robot manipulator D-H parameters

Joint i	α_{i-1}	a_{i-1}	d_i	θ_i
1	0	0	0	θ_1
2	90^o	0	0	$90^o+\theta_2$
3	0	210	0	$90^o+\theta_3$
4	90^o	75	210	θ_4
5	-90^o	0	0	θ_5
6	90^o	0	70	θ_6

4.1.1 Robot manipulator forward kinematic

Determining forward kinematic problem is finding robot end effector position, orientation given by joint angles. Every joint is consisted of position, orientation values relative to its previous joint values. Transformation matrices denote these relations. Following equation represents a general formulation for transformation matrix calculation.

$$T_{i}^{i-1} = R_x(\alpha_{i-1})\, D_x(a_{i-1})\, R_z(\theta_i)\, D_z(d_i) \quad (1)$$

Where

$R_x(\alpha_{i-1})$ = rotation matrix about the X axis by α_{i-1}

$D_x(a_{i-1})$ = translation matrix along the X axis by a_{i-1}

$R_z(\theta_i)$ = rotation matrix about the Z axis by θ_i

$D_z(d_i)$ = translation matrix along the Z axis by d_i

a, α, θ, d are manipulator D-H parameters.

$$R_x(\alpha_{i-1}) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha_{i-1} & -\sin\alpha_{i-1} & 0 \\ 0 & \sin\alpha_{i-1} & \sin\alpha_{i-1} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2)$$

$$D_x(a_{i-1}) = \begin{bmatrix} 1 & 0 & 0 & a_{i-1} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3)$$

$$R_z(\theta_i) = \begin{bmatrix} \cos\theta_i & -sin\theta_i & 0 & 0 \\ sin\theta_i & \cos\theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4)$$

$$D_z(d_i) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5)$$

And T_{i}^{i-1} is

$$\begin{bmatrix} \cos\theta_i & -sin\theta_i & 0 & a_{i-1} \\ \cos\alpha_{i-1}\, sin\theta_i & \cos\alpha_{i-1}\cos\theta_i & -\sin\alpha_{i-1} & -d_i \sin\alpha_{i-1} \\ \sin\alpha_{i-1}\, sin\theta_i & \sin\alpha_{i-1}\cos\theta_i & \cos\alpha_{i-1} & d_i \cos\alpha_{i-1} \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (6)$$

$$T_{1}^{0} = \begin{bmatrix} C_1 & -S_1 & 0 & a_1 \\ S_1 & C_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (7)$$

$$T_{2}^{1} = \begin{bmatrix} C_2 & -S_2 & 0 & a_2 \\ 0 & 0 & -1 & 0 \\ S_2 & C_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (8)$$

$$T_3^2 = \begin{bmatrix} C_3 & -S_3 & 0 & 210 \\ S_3 & C_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (9)$$

$$T_4^3 = \begin{bmatrix} C_4 & -S_4 & 0 & 75 \\ 0 & 0 & -1 & -210 \\ S_4 & C_4 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (10)$$

$$T_5^4 = \begin{bmatrix} C_5 & -S_5 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ S_5 & C_5 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (11)$$

$$T_6^5 = \begin{bmatrix} C_6 & -S_6 & 0 & 0 \\ 0 & 0 & -1 & -70 \\ S_6 & C_6 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (12)$$

Hence, multiplication of matrices (7,8,9,10,11,12) calculates the transformation matrix T_6^0 that gives end effector position, orientation with respect to frame 0 (13).

$$T_6^0 = T_1^0 \, T_2^1 \, T_3^2 \, T_4^3 \, T_5^4 \, T_6^5 \quad (13)$$

4.1.2 Robot manipulator inverse kinematic

Determining robot manipulator inverse kinematic is finding robot joint angles when end effector position, orientation location known.

Considering Denso VP 6242 robot manipulator, last three joints axes intersect at one point and it is referred as point A [figure 19]. The point A position can be considered as independent of the consecutive last three joints θ_4, θ_5, θ_6. Hence, previous three joints are considered when determining Point A position.

End effector position and orientation = $[P_x, P_y, P_z, \gamma, \beta, \alpha]^T$

Where,

α is the angle between Z_6 and Z_0(Z axis of 6th and 1st joint)

β is the angle between Y_6 and Y_0(Y axis of 6th and 1st joint)

γ is the angle between X_6 and X_0(X axis of 6th and 1st joint)

The Point A position is denoted as $P_a = [P_{ax}, P_{ay}, P_{az}]^T$

P_a can be described as,

$$P_{ax} = P_x - d_6 * a_x$$

$$P_{ay} = P_y - d_6 * a_y$$

$$P_{az} = P_z - d_6 * a_z$$

Where

$$a_x = \bar{Z_6} . \bar{X_0} \qquad a_y = \bar{Z_6} . \bar{Y_0} \qquad a_z = \bar{Z_6} . \bar{Z_0}$$

Solution for the θ_1, θ_2 and θ_3 can be derived as follows.

P_a, Point A position can be derived from homogeneous transformation matrix T_4^0 derived from $T_1^0\ T_2^1\ T_3^2$ and T_4^3.

$$T_4^0 = T_1^0\ T_2^1\ T_3^2 T_4^3 = \begin{bmatrix} r_{11} & r_{12} & r_{13} & P_{ax} \\ r_{21} & r_{22} & r_{23} & P_{ay} \\ r_{31} & r_{32} & r_{33} & P_{az} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Where

$r_{11} = S_1S_4 - C_4(C_1C_2C_3 - C_1S_2S_3)$

$r_{12} = C_4S_1 + S_4(C_1C_2C_3 - C_1S_2S_3)$

$r_{13} = - C_1C_2S_3 - C_1C_3S_2$

$r_{21} = - C_1S_4 - C_4(C_2C_3S_1 - S_1S_2S_3)$

$r_{22} = S_4(C_2C_3S_1 - S_1S_2S_3) - C_1C_4$

$r_{23} = - C_2S_1S_3 - C_3S_1S_2$

$r_{31} = -C_4(C_2S_3 + C_3S_2)$

$r_{32} = S_4(C_2S_3 + C_3S_2)$

$r_{33} = C_2C_3 - S_2S_3$

$P_{ax} = 75\ C_1S_2S_3 - 75\ C_1C_2C_3 - 210\ C_1C_2S_3 - 210\ C_1C_3S_2 - 210\ C_1S_2$ (14)

P_{ay}= 75 $S_1S_2S_3$ – 75 $C_2C_3S_1$ – 210 $C_2S_1S_3$ – 210 $C_3S_1S_2$ – 210 S_1S_2 (15)

P_{az}= 210 C_2 + 210 C_2C_3 – 75 C_2S_3 – 75 C_3S_2 – 210 S_2S_3 (16)

Where $S_i = \sin\theta_i$ and $C_i = \cos\theta_i$

(14) × (S_1) – (15) × (C_1) = 0

Therefore,

$P_{ax} \times S_1 = P_{ay} \times C_1$

$\frac{S_1}{c_1} = \frac{P_{ay}}{P_{ax}}$

$\tan\theta_1 = \frac{P_{ay}}{P_{ax}}$ hence $\theta_1 = \tan^{-1}(\frac{P_{ay}}{P_{ax}})$ (17)

by (14) × (C_1) + (15) × (S_1) , the following is obtained.

$P_{ax}C_1 + P_{ay}S_1 = 75S_2S_3 - 75C_2C_3 - 210C_2S_3 - 210C_3S_2 - 210S_2$ (18)

By taking $P_{ax} \times C_1 + P_{ay} \times S_1 = m$, the following can be obtained.

Therefore from (18) : -75 C_{23} – 210 S_{23} – 210S_2 = m

Where $C_{23} = \cos(\theta_2+\theta_3)$

$S_{23} = \sin(\theta_2+\theta_3)$

-75 C_{23} – 210 S_{23} = m + 210S_2 (19)

From (16): 210C_{23} – 75S_{23} = P_{az} – 210C_2 (20)

From (19) and (20), the following is obtained.

$C_{23} = \frac{14P_{az}}{3315} - \frac{m}{663} - \frac{70S_2}{221} - \frac{196C_2}{221}$ (21)

$S_{23} = -\frac{P_{az}}{663} - \frac{14m}{3315} - \frac{196S_2}{221} + \frac{70C_2}{221}$ (22)

Further simplifies by substituting following,

$n = \frac{14P_{az}}{3315} - \frac{m}{663}$ and $o = \frac{-P_{az}}{663} - \frac{14m}{3315}$

then equations (21) and (22) can be simplified as follows.

$$C_{23} = n - \frac{70S_2}{221} - \frac{196C_2}{221} \quad (23)$$

$$S_{23} = o - \frac{196S_2}{221} + \frac{70C_2}{221} \quad (24)$$

substituting the equations (23) and (24) to $S_{23}^2 + C_{23}^2 = 1$

then the following can be obtained.

$$\frac{196c_2^2}{221} - \frac{392C_2n}{221} + \frac{140C_2o}{221} + n^2 - \frac{140S_2n}{221} + o^2 - \frac{392S_2o}{221} + \frac{196s_2^2}{221} = 1 \quad (25)$$

Equation (25) can be simplified by taking $p = n^2 + o^2 + \frac{196}{221}$

Substituting p then

$$-\frac{392c_2n}{221} + \frac{140c_2o}{221} - \frac{140s_2n}{221} - \frac{392s_2o}{221} = 1 - p \quad (26)$$

$a S_2 + b C_2 = c$

by substituting $a = -\frac{140n}{221} - \frac{392o}{221}$, $b = -\frac{392n}{221} + \frac{140o}{221}$ and $c = 1 - p$ for equation (26)

then the joint 2 angle θ_2 can be calculated as follows.

$$\theta_2 = \text{Atan2}(a,b) \pm \text{Atan2}((a^2 + b^2 - c^2)^{1/2}, c) \quad (27)$$

from equation (19): $-75 C_{23} - 210 S_{23} = m + 210S_2$ (28)

Considering equation $e C_{23} + f S_{23} = g$

by substituting $e = -210$, $f = -75$ and $g = m + 210S_2$ for equation (28)

then, $$\theta_2 + \theta_3 = \text{Atan2}(e,f) \pm \text{Atan2}((e^2 + f^2 - g^2)^{1/2}, g) \quad (29)$$

Hence the joint 3 angle value can be derived as follows.

$$\theta_3 = \text{Atan2}(e,f) \pm \text{Atan2}((e^2 + f^2 - g^2)^{1/2}, g) - \theta_2 \quad (30)$$

Solutions for θ_4, θ_5 and θ_6

End effector orientation is defined by R_6^0 . Since robot orientation is described by rotation matrix, position A orientation is defined by R_3^0.

R_6^0 can be described as $R_6^0 = R_3^0 \ R_6^3$

Matrix R_6^3 can be described as

$$R_6^3 = R_4^3 \ R_5^4 \ R_6^5 = \begin{bmatrix} c_4c_5^2 - s_4s_6 & -s_4c_5 - c_4c_5s_6 & c_4s_5 \\ s_5c_5 & -s_5s_6 & -c_5 \\ s_4c_5^2 + c_4s_6 & c_4c_5 - s_4c_5s_6 & s_4s_5 \end{bmatrix} = \begin{bmatrix} n_{11} & n_{12} & n_{13} \\ n_{21} & n_{22} & n_{23} \\ n_{31} & n_{32} & n_{33} \end{bmatrix}$$

Where $S_i = \sin(\theta_i)$ and $C_i = \cos(\theta_i)$

Then the joint 4 angle can be calculated as follows.

$\theta_4 = \tan^{-1} \frac{n_{33}}{n_{13}}$ when $\theta_5 \neq 0$ (31)

when $\theta_5 = 0$, the link axes are in collinear which the arms are at singular position. In this condition, there is only one motion of robot end effector orientation which can be calculated by sum or difference of θ_4 and θ_6. In most cases, current θ_4 value is considered.

T_6^4 Can be derived as follows.

$T_6^4 = T_4^{0\ -1} \ T_6^0$ since θ_1, θ_2 , θ_3 , θ_4 and T_6^0 are known.

T_6^4 can be described as

$$T_6^4 = T_5^4 \ T_6^5 = \begin{bmatrix} c_5c_6 & -c_5s_6 & s_5 & 70s_5 \\ s_6 & c_6 & 0 & 0 \\ -s_5c_6 & s_5s_6 & c_5 & 70c_5 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Where $S_i = \sin(\theta_i)$

$C_i = \cos(\theta_i)$

Therefore joint 5 angle θ_5 can be calculated as follows.

$\theta_5 = \sin^{-1} T_6^4{}_{(1,3)}$ (32)

T_6^5 Can be derived as follows.

$T_6^5 = T_5^{0\,-1}\, T_6^0$ since $\theta_1, \theta_2, \theta_3, \theta_4, \theta_5$ and T_6^0 are known.

T_6^5 can be described as follows.

$$T_6^5 = \begin{bmatrix} c_5 & -s_5 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ s_5 & c_5 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Therefore, joint 6 angle θ_6 can be calculated as,

$\theta_6 = \sin^{-1} T_6^5{}_{(1,2)}$ (33)

4.2 The Human Robot interface

Denso VP 6242 robot manipulator can be virtually developed in a three dimensional graphical user interface using available CAD software interfaces which is high accurate and user friendly. Solidworks [18] is used to develop the CAD models. Model for each links of the robot manipulator can be designed and modelled [Figure 4.2]. 3D design is developed on 1:1 scale since it is used for creating simulation model in the simulation interface.

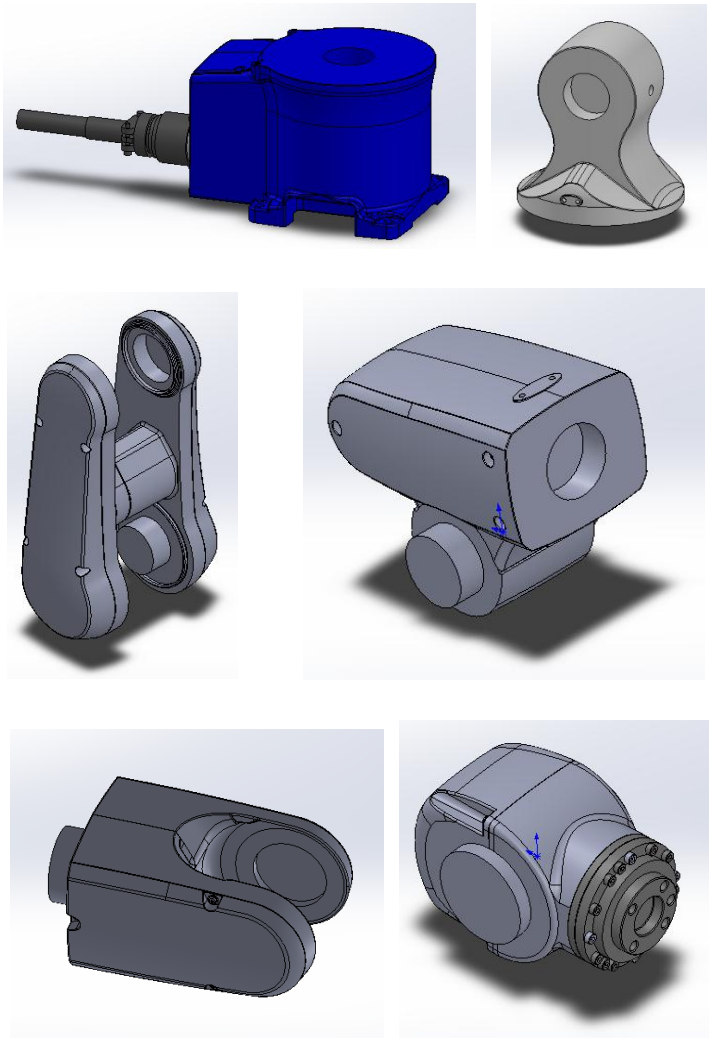


Figure 4.2: Robot links 3D design models

Created 3D link models can be assembled in Solidworks in order to create the complete robot assembly [Figure 4.3].

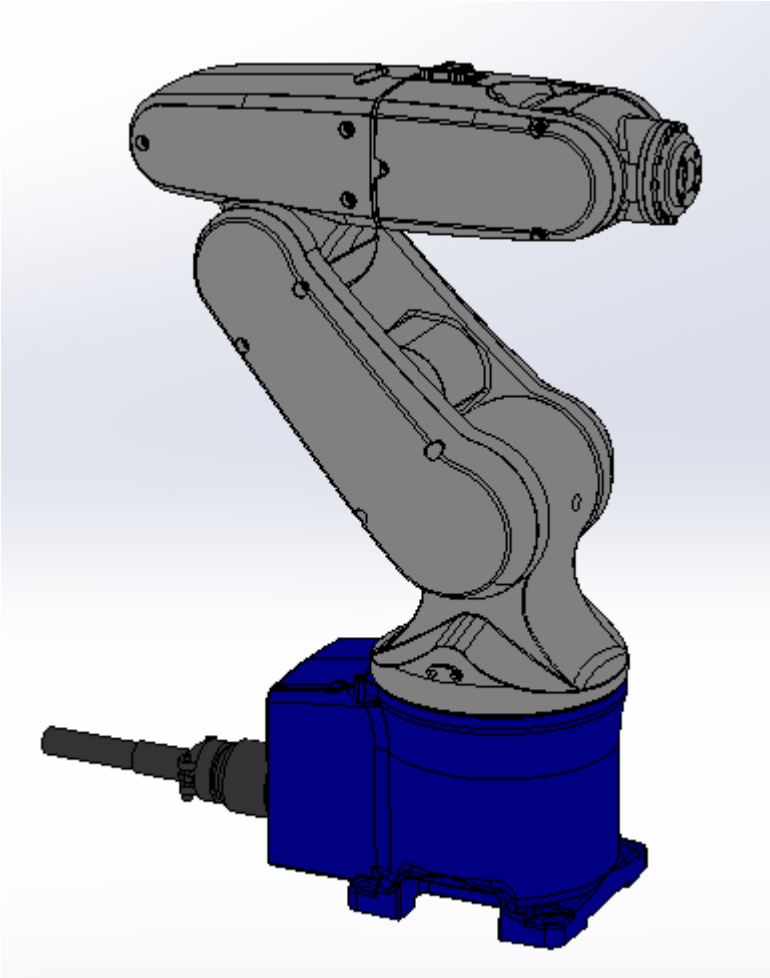


Figure 4.3: Robot manipulator assembly in Solidworks

Coordinate systems can be assigned to the relevant joint axes which are needed to describe the relative position and orientation of each link [Figure 4.4]. Work coordinate system can be assigned relative to developed mathematical model in section 4.1.

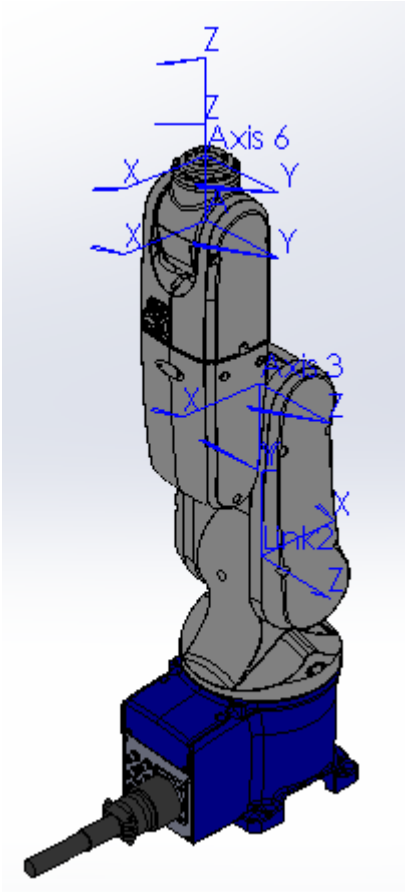


Figure 4.4: Coordinate system assignment in 3D model

Tool which is attached to the robot end can be modelled on 1:1 scale [Figure 4.5].

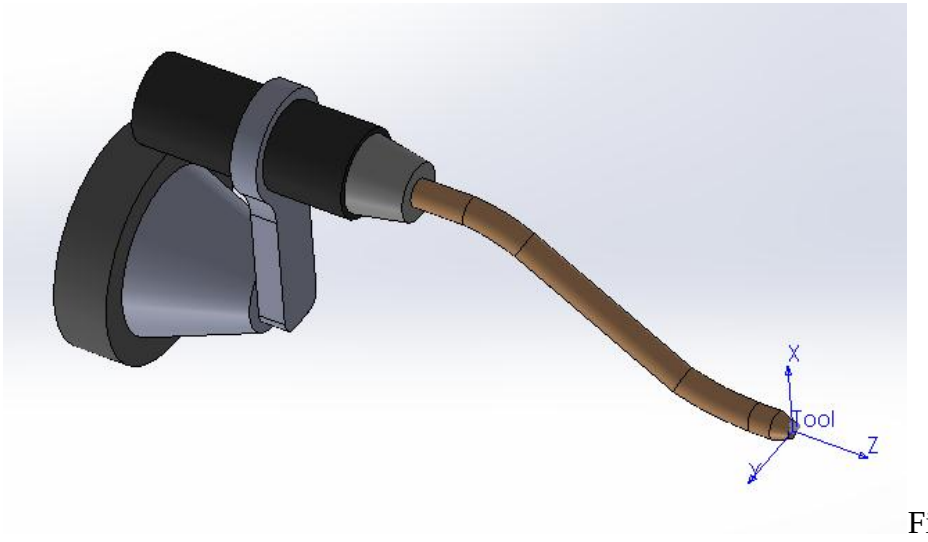


Figure 4.5: Tool design – Welding torch

The machine head cover can be modelled on 1:1 scale [Figure 4.6].

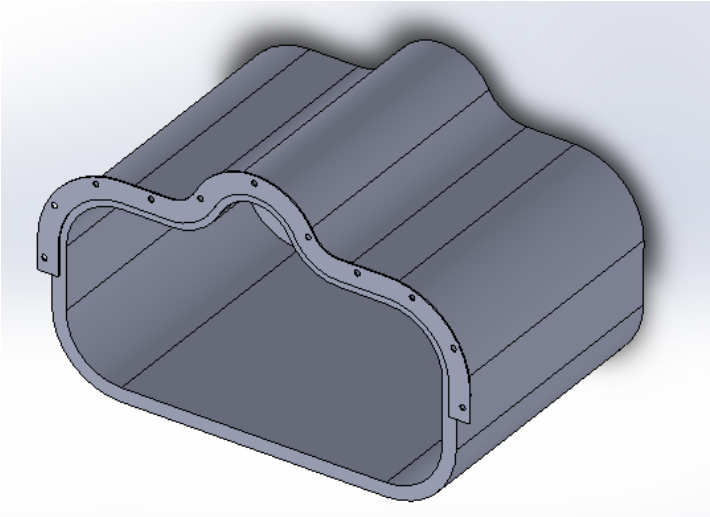


Figure 4.6: Machine head cover – 3D design

The head cover can be located according the actual robot cell dimensions [Figure 4.7].

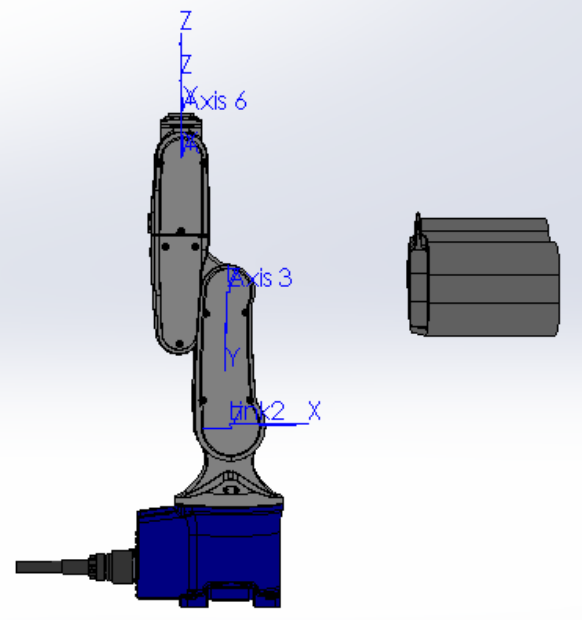


Figure 4.7: Machine Head cover location

Complete robot cell is illustrated in Figure 4.8.

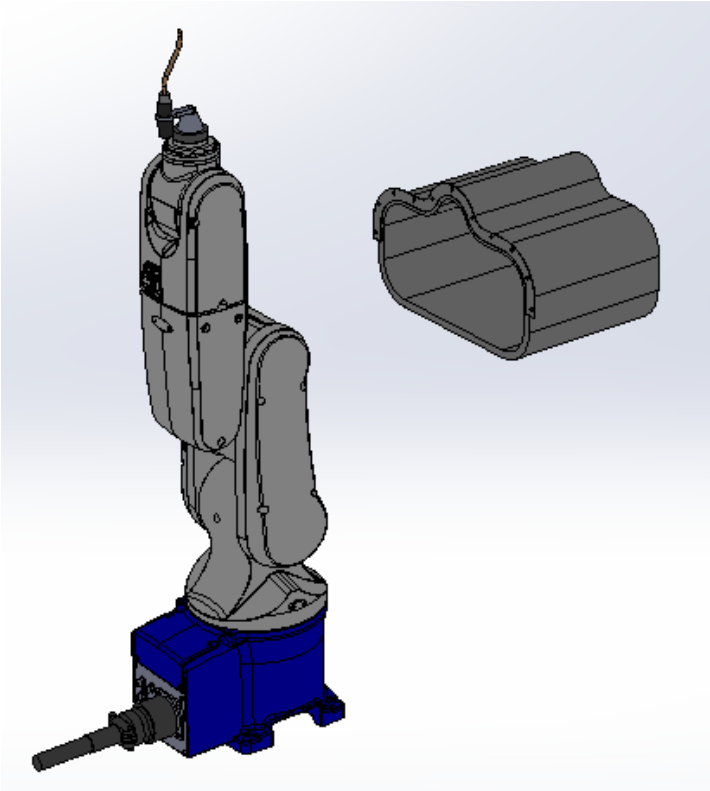


Figure 4.8: Complete 3D design for the robot cell

4.3 Path planning and information extraction

Robot end moving line can be defined using a spline which is accurate to describe the path efficiently [Figure 4.9].

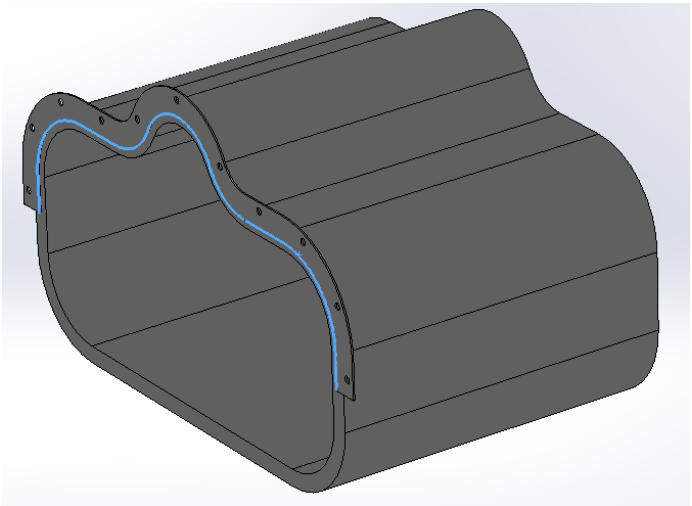


Figure 4.9: Spline feature for robot moving path definition

Points are located along the line which assigned to represent robot moving sequence [Figure 4.10].

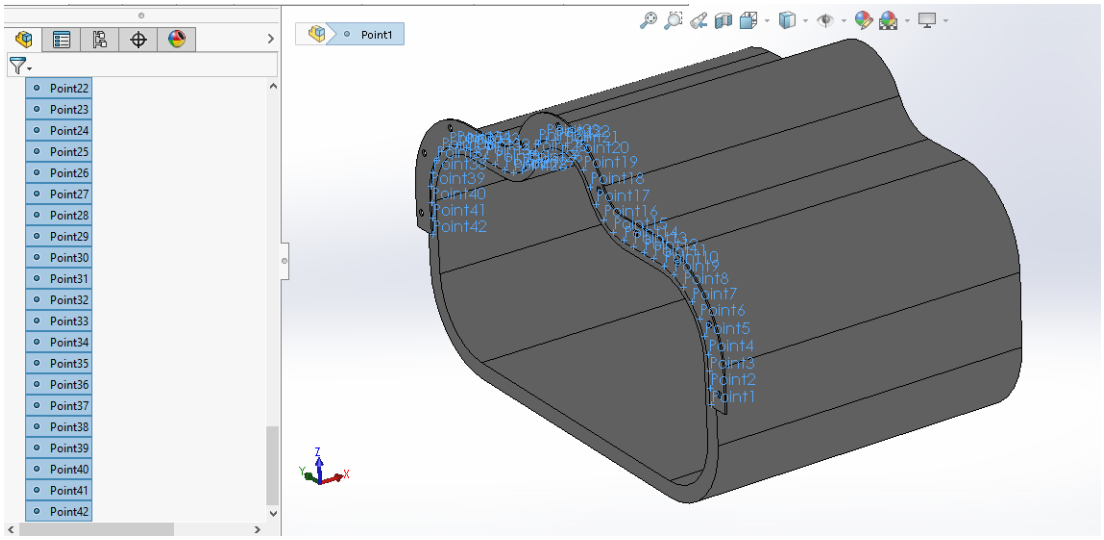
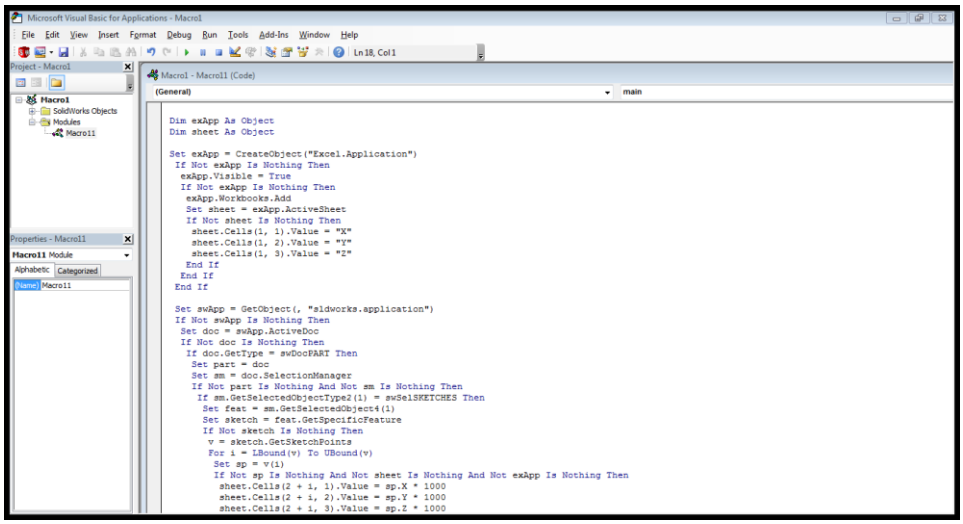


Figure 4.10: Point allocation along the robot path

Position values of these points relative to origin can be used for calculating forward and inverse kinematics [Section 4.1]. X, Y and Z coordinate values are extracted from the CAD spline. VB (Visual Basic) based macro is developed and run in Solidworks to generate the point coordinate values [Figure 4.11]. VB based macro is also capable to generate excel sheet with x y z point values [Figure 4.12].



```
Dim exApp As Object
Dim sheet As Object

Set exApp = CreateObject("Excel.Application")
 If Not exApp Is Nothing Then
  exApp.Visible = True
  If Not exApp Is Nothing Then
  exApp.Workbooks.Add
  Set sheet = exApp.ActiveSheet
  If Not sheet Is Nothing Then
   sheet.Cells(1, 1).Value = "X"
   sheet.Cells(1, 2).Value = "Y"
   sheet.Cells(1, 3).Value = "Z"
  End If
 End If
End If

Set swApp = GetObject(, "sldworks.application")
If Not swApp Is Nothing Then
 Set doc = swApp.ActiveDoc
 If Not doc Is Nothing Then
  If doc.GetType = swDocPART Then
   Set part = doc
   Set sm = doc.SelectionManager
   If Not part Is Nothing And Not sm Is Nothing Then
    If sm.GetSelectedObjectType2(1) = swSelSKETCHES Then
     Set feat = sm.GetSelectedObject4(1)
     Set sketch = feat.GetSpecificFeature
     If Not sketch Is Nothing Then
      v = sketch.GetSketchPoints
      For i = LBound(v) To UBound(v)
       Set sp = v(i)
       If Not sp Is Nothing And Not sheet Is Nothing And Not exApp Is Nothing Then
        sheet.Cells(2 + i, 1).Value = sp.X * 1000
        sheet.Cells(2 + i, 2).Value = sp.Y * 1000
        sheet.Cells(2 + i, 3).Value = sp.Z * 1000
```

Figure 4.11: VB based macro for data extraction

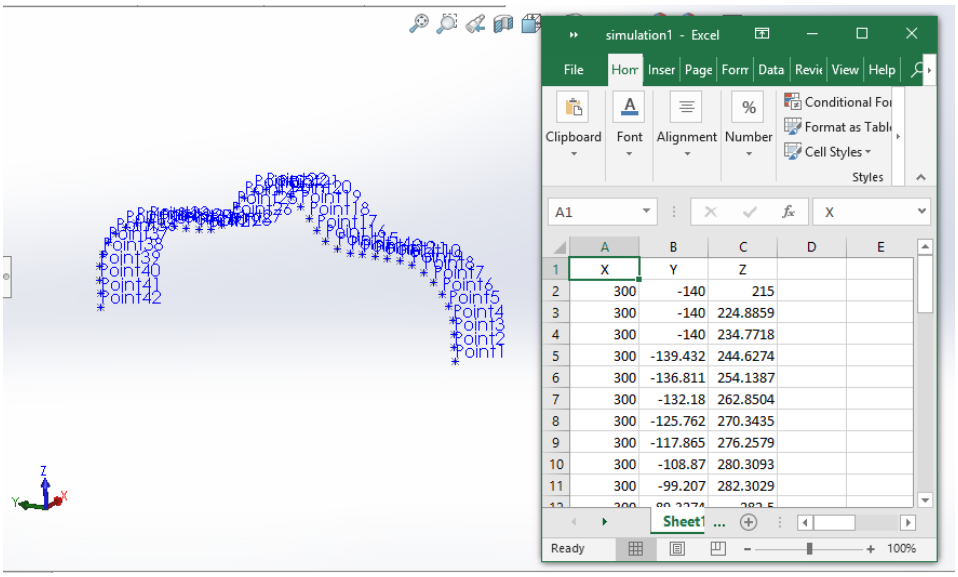


Figure 4.12: Data extracted values in Microsoft excel

4.4 Mathematical approach for mapping trajectories

Joint space trajectory generation in robotic field is commonly using for arranging smooth motion between one set of joint angles with another set such as for travelling between two specific cartesian postures having two joint angles sets for each posture. This has to be done simultaneously while generating for all joints independently.

Generally initial assumptions are considered as two discrete joint value sets are known and the requirement is to move between those two joint angles sets smoothly in joint space. The velocity and acceleration need to vary smoothly in order to maintain optimized robot movement without existing infinite jerks which make robot manipulators inaccurate operations, high vibration and wear.

Each of joints has a motor drive which is connected to rotate the joint according to control input by the robot controller system. The motor should be controlled under technical specifications such as maximum angular velocity, joint motion range, maximum inertia movement [11]. Each motor is limited with maximum angular velocity and controllers should manipulate the robot arm under these limitations.

A trajectory can be specified by assigning initial and final conditions on a time period, position, velocity, acceleration etc. Then, trajectory planning can be determined as a function so that the required conditions are satisfied. This is considered as a boundary condition problem which can be explained by considering polynomial functions such as:

$q(t) = a_0 + a_1t + a_2t^2 + \ldots + a_nt^n$

The degree n of the polynomial depends on the number of boundary conditions which should be determined by the trajectory smoothness we require. Given an initial and a final time period i.e. t_i and t_f, a trajectory segment can be specified by assigning initial and final conditions:

initial position and velocity q_i, $\dot{q}_i$

final position and velocity q_f, $\dot{q}_f$

A polynomial of degree 3 can be considered since there are four boundary conditions. From the studies [10], it may be noticed that position and velocity profiles are continuous functions of time but cannot be true for the acceleration so that discontinuities among different segments can be occurred. Besides, there is no possibility to specify initial and final values for each segment. In some cases, this is not a major problem and it is enough to have smooth trajectories. But for most cases, acceleration initial and final values for obtaining acceleration profiles is required for avoiding possible infinite jerks of non-continuous jerk profiles. Therefore, fifth order polynomial functions should be considered.

$q(t) = a_0 + a_1t + a_2t^2 + a_3t^3 + a_4t^4 + a_5t^5$

Boundary conditions are defined as follows:

$q(t_i) = q_i$: initial position at time t_i $q(t_f) = q_f$: final position at time t_f

$\dot{q}(t_i) = \dot{q}_i$: initial velocity at time t_i $\dot{q}(t_f) = \dot{q}_f$: final velocity at time t_f

$\ddot{q}(t_i) = \ddot{q}_I$: initial acceleration at time t_i $\ddot{q}(t_f) = \ddot{q}_f$: final acceleration at time t_f

The coefficient of the polynomial can be derived.

In this case, the coefficients of the polynomial are derived as follows.

$$a_0 = q_i \quad (34)$$

$$a_1 = \dot{q}_i \quad (35)$$

$$a_2 = \frac{1}{2}\ddot{q}_i \quad (36)$$

$$a_3 = \frac{1}{2T^3}[20(q_f - q_i) - (8\dot{q}_f + 12\dot{q}_i)T - (3\ddot{q}_f - \ddot{q}_i)T^2] \quad (37)$$

$$a_4 = \frac{1}{2T^4}[30(q_i - q_f) + (14\dot{q}_f + 16\dot{q}_i)T + (3\ddot{q}_f - 2\ddot{q}_i)T^2] \quad (38)$$

$$a_5 = \frac{1}{2T^5}[12(q_f - q_i) - 6(\dot{q}_f + \dot{q}_i)T - (\ddot{q}_f - \ddot{q}_i)T^2] \quad (39)$$

where $T = t_f - t_i$

Velocity, acceleration and jerk profile are derived as follows.

$$\dot{q}(t) = a_1 + 2a_2t + 3a_3t^2 + 4a_4t^3 + 5a_5t^4 \text{ (fourth order polynomial)} \quad (40)$$

$$\ddot{q}(t) = 2a_2 + 6a_3t + 12a_4t^2 + 20a_5t^3 \text{ (third order polynomial)} \quad (41)$$

$$\dddot{q}(t) = 6a_3 + 24a_4t + 60a_5t^2 \text{ (second order polynomial)} \quad (42)$$

Robot position is intended to travel point to point movement along the path via assigned positions. Boundary conditions of the trajectories are defined as follows.

$\dot{q}_i = 0$: initial velocity at time t_i $\dot{q}_f = 0$: final velocity at time t_f

$\ddot{q}_i = 0$: initial acceleration at time t_i $\ddot{q}_f = 0$: final acceleration at time t_f

Figure 4.13 illustrates Typical trajectory profile for position variation.

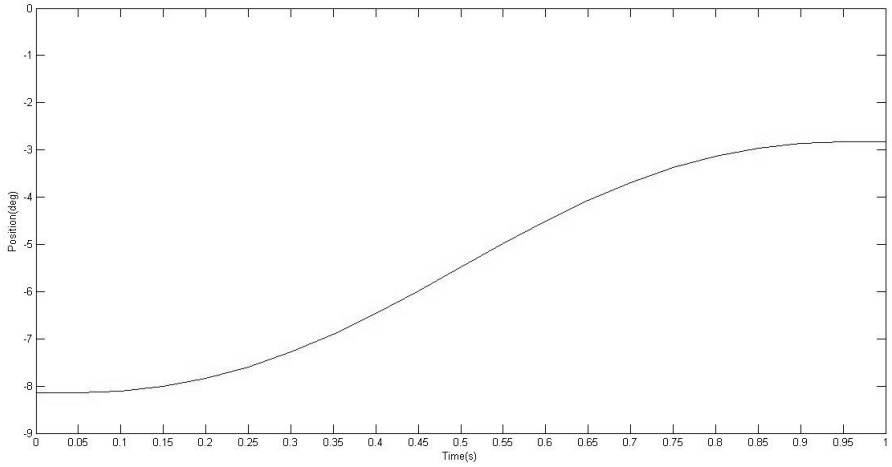


Figure 4.13: Typical trajectory profile for position variation

Figure 4.14 illustrates typical trajectory profile for velocity variation.

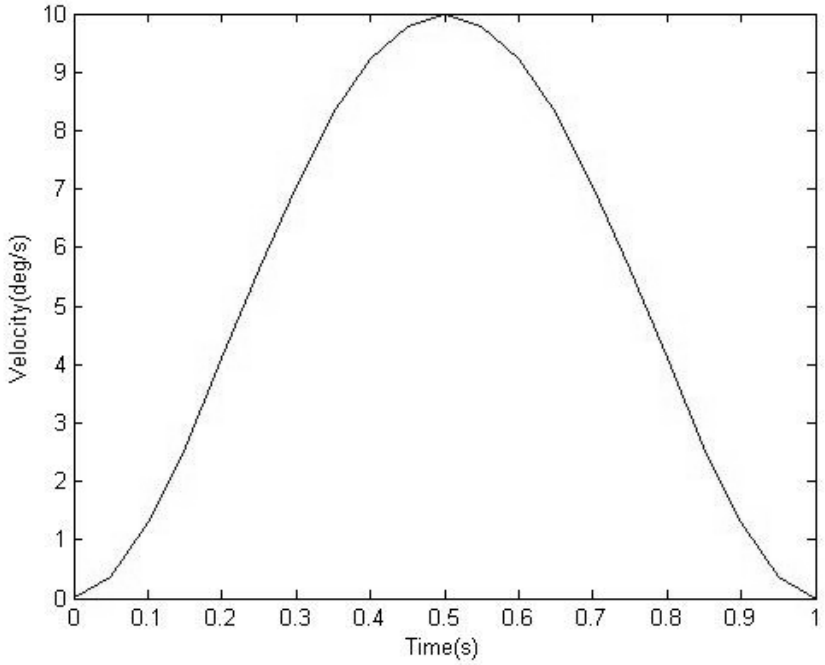


Figure 4.14: Typical trajectory profile for velocity variation

Figure 4.15 illustrates typical trajectory profile for acceleration variation.

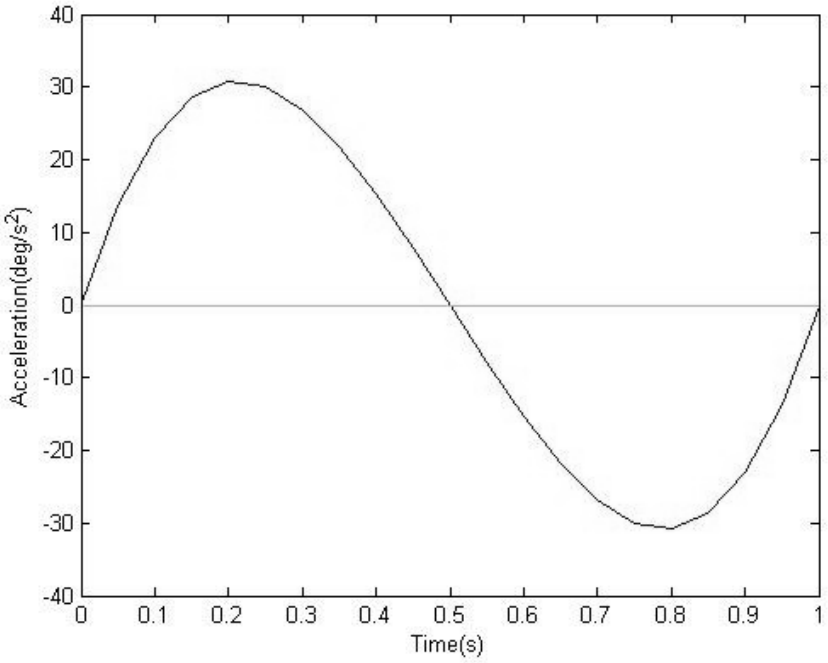


Figure 4.15: Typical trajectory profile for acceleration variation

Figure 4.16 illustrates typical trajectory profile for Jerk variation.

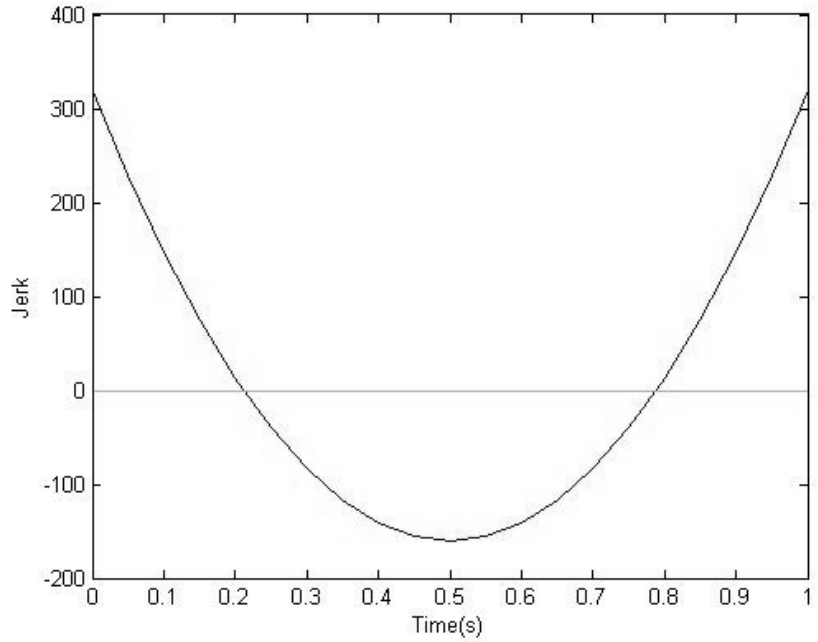


Figure 4.16: Typical trajectory profile for jerk variation

From the equations (34) to (39) with boundary conditions assigned, following expressions can be derived.

$a_0 = q_i$ (43)

$a_1 = 0$ (44)

$a_2 = 0$ (45)

$a_3 = \frac{1}{2T^3}[20(q_f - q_i)]$ (46)

$a_4 = \frac{1}{2T^4}[30(q_i - q_f)]$ (47)

$a_5 = \frac{1}{2T^5}[12(q_f - q_i)]$ (48)

Maximum velocity of a particular joint is reached in the middle of the time duration [Figure 32]. Therefore, maximum velocity can be derived as follows.

Applying solutions to the equation (40):

$\dot{q}(t) = 3\frac{1}{2T^3}[20(q_f - q_i)]t^2 + 4\frac{1}{2T^4}[30(q_i - q_f)]t^3 + 5\frac{1}{2T^5}[12(q_f - q_i)]t^4$

when $t = \frac{T}{2}$

Then,

$\dot{q}(t)_{max} = 3\frac{1}{2T^3}[20(q_f - q_i)]\left(\frac{T}{2}\right)^2 + 4\frac{1}{2T^4}[30(q_i - q_f)]\left(\frac{T}{2}\right)^3 + 5\frac{1}{2T^5}[12(q_f - q_i)]\left(\frac{T}{2}\right)^4$

$\dot{q}(t)_{max} = \frac{15(q_f - q_i)}{8T}$ (49)

therefore, T (time period from one position to another) can be calculated for a particular speed limit.

$T \geq \frac{15(q_f - q_i)}{8\dot{q}(t)_{max}}$ (50)

Maximum velocity should be considered when the robot manipulator is operated with a significant operating speed such that when robot is moving from one cartesian location to another position in a limited time period. Variation of each joint speed is varied depends on the location and robot arm configuration and speed limitation is

defined in order protect from possible breakdowns and enhance the robot operation. The robot user can realize the possible speed limitations by mapping maximum speed of each joint while moving along the path via assigned position points. Then the position points can be allocated according to the observations in order to minimize the joint speed variation and optimize the system in order to increase the productivity.

CHAPTER 5

SYSTEM DESIGN AND SIMULATION

Mathematical models developed for inverse kinematics are described in this chapter and trajectory planning solutions proposed in Chapter 4. Section 5.1 presents the procedure to implement inverse kinematic solution for the 6 DOF robot manipulator which is needed to derive joint space data for further development of trajectory planning. Section 5.2 presents the procedure to implement trajectory planning scheme developed for moving robot arm effectively. Development of robot cell to trial in a Simulation environment is illustrated in Section 5.3 which clarifies the accuracy and the effectiveness of the mathematical solutions and schemes developed.

5.1 Design and Implementation of Inverse kinematic model

Algorithms can be developed using derived mathematical models developed in section 4. Matlab is a software tool used for analyzing data, developing algorithms, or creating models [19].

The inverse kinematic model for the Denso VP 6242 robot arm has been developed considering without end effector attachment. But we can see tools attached to the robot arm end for many industrial uses such as painting nozzle, welding torch, cutter unit. In this application, welding torch has been applied as an end effector. Therefore, the tool end is subjected to move along the path. Path and Points can be assigned according to user's requirements [Figure 5.1].

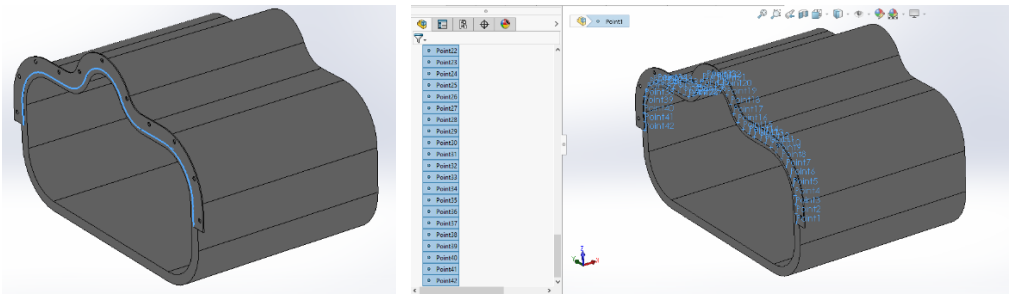


Figure 5.1: Path and point allocation along the robot moving path

Robot end tool position and orientation are needed to define for calculating and perform developed algorithm. Tool position is same as the point location of the defined path. Orientation is defined as follows and the orientation is needed to persist the same along the path [Figure 5.2 and Figure 5.3].

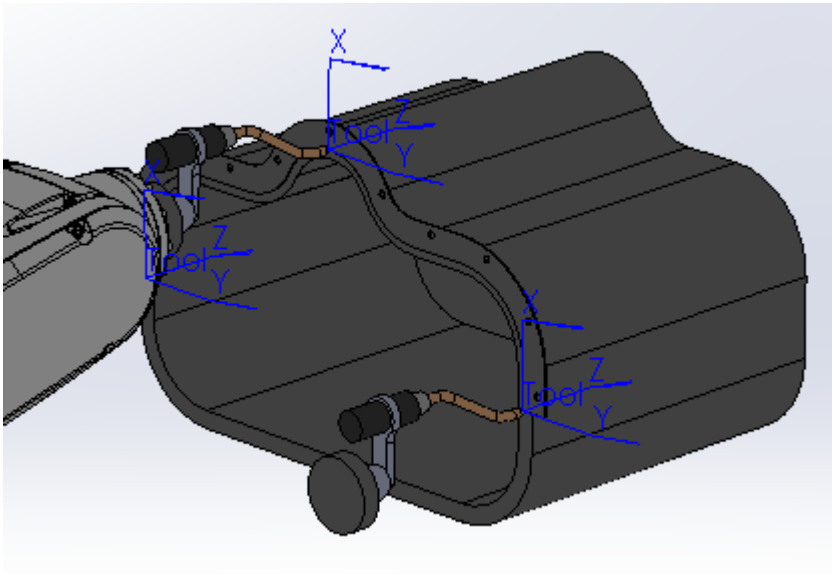


Figure 5.2: Tool orientation along the path

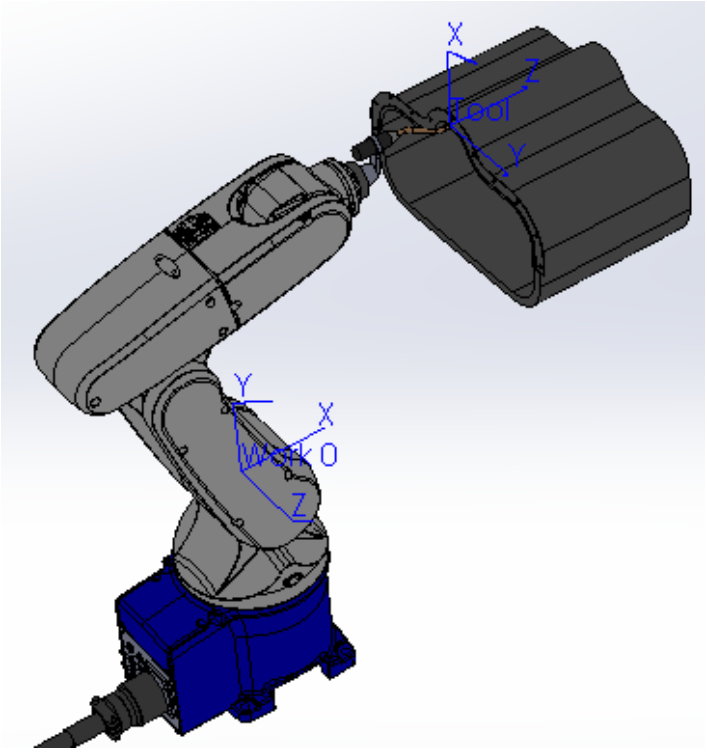


Figure 5.3: Tool orientation with respect to work 0 coordinate

The end-effector(tool) position, orientation with respect to base, known as T_{Tool}^{0} is defined as follows.

$$T_{Tool}^{0} = \begin{bmatrix} R_{Tool}^{0} & P_{Tool}^{0} \\ 0\ 0\ 0 & 1 \end{bmatrix} \quad (51)$$

Where P_{Tool}^{0} = $[X_{tool}, Y_{tool}, Z_{tool}]^T$ position coordinate and R_{Tool}^{0} can be derived as follows.

$$R_{Tool}^{0} = R_Z(180^0)\, R_Y(-90^0)\, R_X(0^0) \quad (52)$$

Then T_{Tool}^{0} can be derived as $T_{Tool}^{0} = T_{6}^{0}\ T_{Tool}^{6}$ (53)

T_{Tool}^{6} Tool position, orientation with respect to joint coordinate 6 is derived as follows.

$$T_{Tool}^{6} = \begin{bmatrix} 1 & 0 & 0 & 10 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 130 \\ 0 & 0 & 0 & 1 \end{bmatrix} \text{ [Figure 5.4]} \quad (54)$$

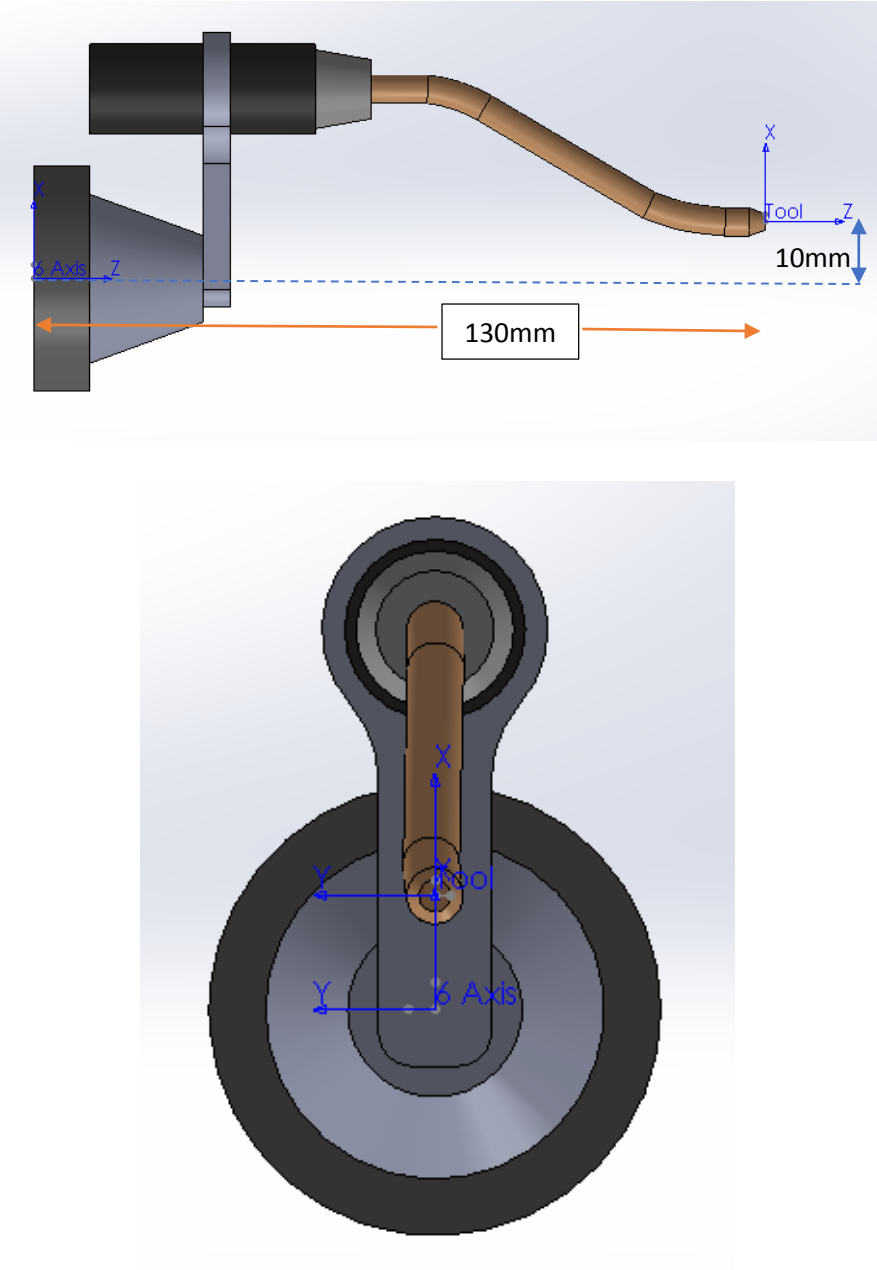
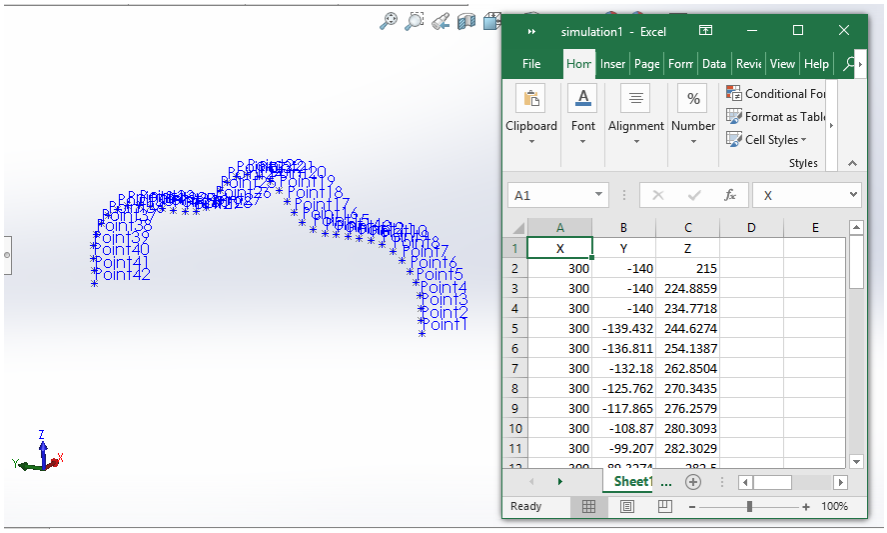


Figure 5.4: Tool dimensions

From equation (53), T_6^0 can be derived as $T_6^0 = T_{Tool}^0 \, T_{Tool}^{6\,-1}$ (55)

X, Y, Z cartesian point data can be derived from the Solidworks CAD environment by running a Visual Basic based macro [Section 4.3]. This macro exports cartesian point values of each point location only and we need to manually input the Rx, Ry, Rz rotation angles (around axes X ,Y, Z) for each point location or sequence which will

be needed for finding joint angles of the robot arm for each point location by developing algorithms in Matlab software. This rotation angles are the angles values defined in equation (52) [Figure 5.5].



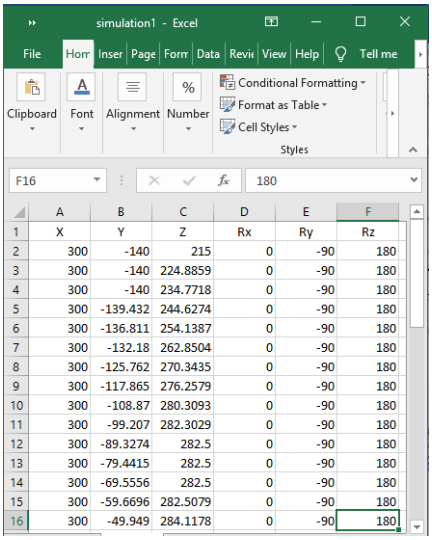


Figure 5.5: Orientation defined by manually

The position of point A in the robot arm is needed to find in order to implement the developed algorithms in section 4.1.2. Matlab codes can be written to implement this procedure in order to find the P_a location.

Importing data extracted from the CAD features

```
C = xlsread('C:\Users\pcadmin\research\simulation1.xlsx','Sheet1');
tam = size(C);

Pa = [];
for i = 1 : tam

    T6EF = [1 0 0 10;0 1 0 0;0 0 1 130;0 0 0 1];

    X(i,1) = C(i,1);%X value
    Y(i,2) = C(i,2);%Y value
    Z(i,3) = C(i,3);%Z value

    al = C(i,6); %angle arond Z axis
    be = C(i,5);%angle around Y axis
    ga = C(i,4);%angle around X axis

    %Eular angle Z,Y,X
    Rz = [  cosd(al) -sind(al)      0;
            sind(al) cosd(al)       0;
            0           0           1;];

    Ry = [  cosd(be)    0           sind(be);
            0           1           0;
            -sind(be)   0           cosd(be);];

    Rx = [  1           0           0;
            0           cosd(ga)    -sind(ga);
            0           sind(ga)    cosd(ga);];

    %Rotation Matrix
    A_R_B = Rz*Ry*Rx;

TT(:, :, i) = [A_R_B(1,1) A_R_B(1,2) A_R_B(1,3) X(i,1); A_R_B(2,1)
A_R_B(2,2) A_R_B(2,3) Y(i,2); A_R_B(3,1) A_R_B(3,2) A_R_B(3,3)
Z(i,3);
0 0 0 1];

T6EFT = ((T6EF)^(-1));
T(:, :, i) = TT(:, :, i)* T6EFT;

%Pa position
x = T(1,4,i)- (70)*T(1,3,i);%Pa x position
y = T(2,4,i)- (70)*T(2,3,i);%Pa y position
z = T(3,4,i)- (70)*T(3,3,i);%Pa z position
end
```

Algorithms for finding joint angles θ_1, θ_2 and θ_3

Implementing equation (17) from section 4.1.2

```
%Calculating theta 1
theta1 = atan2d(y,x);

if theta1 == 180
    theta1 = 0;
end


if (90<theta1) && (theta1<180)
    theta1 = -(180 - theta1);
end

if (-180>theta1) && (theta1>-90)
    theta1 = 180 + theta1;
end
%}
```

Implementing equation (27) and (30)

```
%Calculating theta 2 and theta 3
c1 = cosd(theta1);
s1 = sind(theta1);

m = (x*c1) + (y*s1);
n = (-1/663) + (14*z/3315);
o = ((-14*l)/3315) - (z/663);
p = (196/221) + (n^2) + (o^2);

a_ = ((-140*n) - (392*o))/221;
b_ = ((140*o) - (392*n))/221;

c = 1 - p;

theta2 = atan2(a_,b_) + atan2(((a_^2 + b_^2 - c^2))^(1/2),c);
theta2 = (theta2*180)/pi;

if theta2 ==360
    theta2 = 0;
end

if (180<theta2) && (theta2<360)
theta2 = 360 - theta2;
end

%Theta 3

e = -210.00;
f = -75.00;
g = l+(210*(sind(theta2)));
```

```
h = ((e^2 + f^2 - g^2))^(1/2);

if isreal(h) == 1
    theta23_1 = atan2(e,f)+ atan2(h,g);
    theta23_1 = (theta23_1*180)/pi;

    theta_23_2 = atan2(e,f)- atan2(h,g);
    theta_23_2 = (theta_23_2*180)/pi;

    if (theta_23_2>-360) && (theta_23_2<-180)
        theta_23_2 = theta_23_2 + 360;
    end

    theta_3_1 = - theta2 + theta23_1;

    theta_3_2 = - theta2 + theta_23_2;


theta2_ = atan2(a_,b_) - atan2((a_^2 + b_^2 - c^2)^(1/2),c);
theta2_ = (theta2_*180)/pi;

if ( theta2_ > -360) && (theta2_ < -180)
    theta2_ = 360 + theta2_;
end


e = -210.00;
f = -75.00;
g_ = l+(210*(sind(theta2_)));
h_ = (e^2 + f^2 - g_^2)^(1/2);

if isreal(h_) == 1
    theta23_3 = atan2(e,f)+ atan2(h_,g_);
    theta23_3 = (theta23_3*180)/pi;

    theta23_4 = atan2(e,f)- atan2(h_,g_);
    theta23_4 = (theta23_3*180)/pi;

    theta_3_3 = - theta2_ + theta23_3;

    theta_3_4 = - theta2_ + theta23_4;

else
    theta_3_3 = 0 - theta2_;

    theta_3_4 = 0 - theta2_;
end
```

Algorithms for finding joint angles θ_4, θ_5 and θ_6

```
%DH parameters
al_0=0;al_1=90,al2=0;al_3=90;al_4=-90;al_5=90;
a_0=0;a_1=0;a_2=210;a_3=75;a_4=0;a_5=0;
d_1=0;d_2=0;d_3=0;d_4=210;d_5=0;d_6=70;
t_1=theta_1;t_2=90+theta_2;t_3=90+theta_3;

T1_0 = [ cosd(t_1),           -sind(t_1),              0,
a_0           ;
        sind(t_1)*cosd(al_0), cosd(t_1)*cosd(al_0),     -sind(al_0),
-sind(al_0)*d_1;
        sind(t_1)*sind(al_0), cosd(t_1)*sind(al_0),     cosd(al_0),
cosd(al_0)*d_1 ;
        0,                  0,                      0,          1
;   ]

T2_1 = [ cosd(t_2),           -sind(t_2),              0,
a_1           ;
        sind(t_2)*cosd(al_1), cosd(t_2)*cosd(al_1),     -sind(al_1),
-sind(al_1)*d_2;
        sind(t_2)*sind(al_1), cosd(t_2)*sind(al_1),     cosd(al_1),
cosd(al_1)*d_2 ;
        0,                  0,                      0,          1
;   ]

T3_2 = [ cosd(t_3),           -sind(t_3),              0,
a_2           ;
        sind(t_3)*cosd(al_2), cosd(t_3)*cosd(al_2),     -sind(al_2),
-sind(al_2)*d_3;
        sind(t_3)*sind(al_2), cosd(t_3)*sind(al2)_     cosd(al_2),
cosd(al_2)*d_3 ;
        0,                  0,                      0,          1
;   ]
```

Implementing equation (31)

```
%Calculating theta4
T3_0 = T1_0*T2_1*T3_2;


R03 = [T3_0(1,1:3);
        T3_0(2,1:3);
        T3_0(3,1:3);];

R03T = transpose(R03);

R06 = [T(1,1:3);
        T(2,1:3);
        T(3,1:3);];

R6_3 =  R03T*R06;

theta4 = atan2d(R6_3(3,3),R6_3(1,3));
```

```
if theta4 == 180
    theta4 = 0;
end
```

Implementing equation (32)

```
%Calculating theta 5
T_4 = theta4;
T4_3 = [ cosd(t_4),           -sind(t_4),              0,
a_3           ;
        sind(t_4)*cosd(al_3), cosd(t4)*cosd(al_3),     -sind(al_3),
-sind(al_3)*d_4;
        sind(t_4)*sind(al_3), cosd(t_4)*sind(al_3),     cosd(al_3),
cosd(al_3)*d_4 ;
        0,                  0,                      0,          1
;   ]

T4_0 = T3_0 * T4_3;


T4_0T = transpose(T4_0);
T46 = T4_0T * T;
theta5 = asind(T46(1,3));
```

Implementing equation (33)

```
%Calculating theta 6
t_4 = theta4;
t_5 = theta5;

T4_3 = [ cosd(t_4),           -sind(t_4),              0,
a_3           ;
        sind(t_4)*cosd(al_3), cosd(t_4)*cosd(al_3),     -sind(al_3),
-sind(al_3)*d_4;
        sind(t_4)*sind(al_3), cosd(t_4)*sind(al_3),     cosd(al_3),
cosd(al_3)*d_4 ;
        0,                  0,                      0,          1
;   ]

T5_4 = [ cosd(t_5),           -sind(t_5),              0,
a_4           ;
        sind(t_5)*cosd(al_4), cosd(t_5)*cosd(al_4),     -sind(al_4),
-sind(al_4)*d_5;
        sind(t_5)*sind(al_4), cosd(t_5)*sind(al_4),     cosd(al_4),
cosd(al_4)*d_5 ;
        0,                  0,                      0,          1
;   ]

T5_0 = T3_0*T4_3*T5_4;


R05 = [T5_0(1,1:3);
        T5_0(2,1:3);
        T5_0(3,1:3);];
```

```
R5_0T = transpose(R05);

R6_0 = [T(1,1:3);
        T(2,1:3);
        T(3,1:3);];

R6_5 =  R5_0T*R6_0;

%Calculating theta6
theta6 = asind(-R6_5(1,2));

end
```

5.2 Algorithms for developing trajectory planning schemes in section 4.4

```
initial_time=0;
final_time=time;
timestep=.05;
x=initial_time:timestep:final_time;%time 0 to 20 seconds

t = size(x,2);

jtable(t,7)=0;%joint table
vtable(t,7)=0;%velociy table
atable(t,7)=0;%angular table
qtable(t,7)=0;%jerk table
jtable(:,1)=x;%table with time
vtable(:,1)=x;%table with time
atable(:,1)=x;%table with time
qtable(:,1)=x;%table with time
c = jtable(1:t,1);%time from 0 to 10seconds
t1 = size(c,1);

%joint parameters

    if line == 1

    jointi=[51.84,12.298,-119.428,-103.03,-53.81,111.40];%home
position [230,140,215,0,-90,180]

jointf=[t_t(line,1),t_t(line,2),t_t(line,3),t_t(line,4),t_t(line,5),
t_t(line,6)];
    else
    jointi=[t_t(line-1,1),t_t(line-1,2),t_t(line-1,3),t_t(line-
1,4),t_t(line-1,5),t_t(line-1,6)];

jointf=[t_t(line,1),t_t(line,2),t_t(line,3),t_t(line,4),t_t(line,5),
t_t(line,6)];
    end
%Joint boundary conditions
for k=1:6

ji=jointi(k); %Initial position
jf=jointf(k); % Final position
```

```
j_vi=0; % Initial Angular velocity is assumed as zero
j_vf=0; % Final Angular velocity is assumed as zero
j_ai=0;% Initial acceleration is assumed as zero
j_af=0; % Final acceleration is assumed as zero

% Assume arm comes to the end position position with time
t_i=0;%initial time
t_f=x(1,t);%final time
T=t_f-t_i;%time difference

%Coefficients calculation
for i = 1:t

a0=ji;
a1=j_vi;
a2=j_ai/2;
a3=[20*(jf-ji)-(8*j_vf+12*j_vi)*T-(3*j_af-j_ai)*T.^2]/(2*T.^3);
a4=[30*(ji-jf)+(14*j_vf+16*j_vi)*T+(3*j_af-2*j_ai)*T.^2]/(2*T.^4);
a5=[12*(jf-ji)-6*(j_vf+j_vi)*T-(j_af-j_ai)*T.^2]/(2*T.^5);
Coefficients_J1=[a_0 a_1 a_2 a_3 a_4 a_5]

jtable(i,k+1)=
a_0+(a_1*x(1,i))+(a_2*(x(1,i))^2)+(a_3*(x(1,i))^3)+(a_4*(x(1,i))^4)+
(a_5*(x(1,i))^5);
vtable(i,k+1)=
a_1+(2*a_2*x(1,i))+(3*a_3*(x(1,i)^2))+(4*a_4*(x(1,i)^3))+(5*a_5*(x(1
,i)^4));
atable(i,k+1)=
(2*a_2)+(6*a_3*x(1,i))+(12*a_4*(x(1,i)^2))+(20*a_5*(x(1,i)^3));
qtable(i,k+1)= 6*a_3+(24*a_4*x(1,i))+(60*a_5*(x(1,i)^2))

end
```

Implementation of maximum velocity mapping

```
for i = 1:tam

[jtable vtable atable qtable] = trajectory(i,1);%line number,time

vsize = size(vtable);

    for q = 2:7
        vmax(i,q-1) = max(abs(vtable(:,q)))
    end
end

vmaxtable =
array2table(vmax(2:tam,:),'VariableNames',{'J1','J2','J3','J4','J5',
'J6'});

figure;
plot(2:tam,vmax(2:tam,1:6))
```

```
ylabel('Maximum velocity(deg/sec)');xlabel('Line
number');set(gca,'XTick',[0:1:line]);
legend('joint 1','joint 2','joint 3','joint 4','joint 5','joint 6')
```

sample graph derived for the maximum velocity mapping is shown in Figure 5.6.

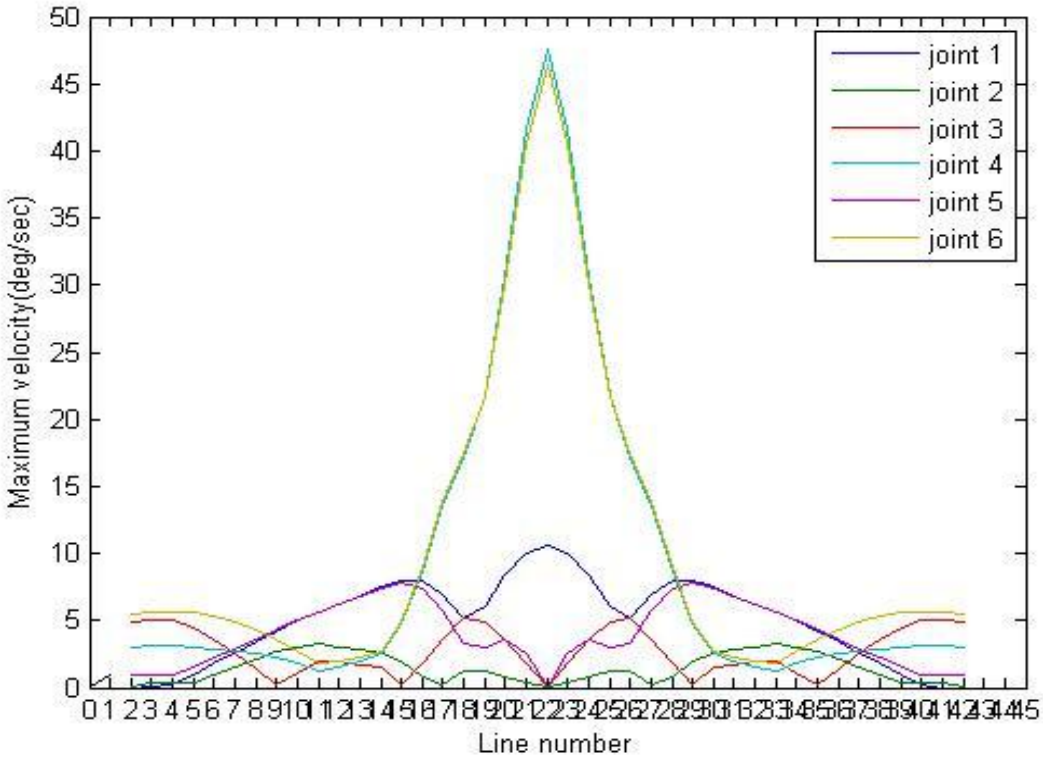


Figure 5.6: Maximum velocity mapping

Time calculation algorithm for reaching maximum velocity developed in section 4.4.

```
initial_time=0;
final_time=2;
timestep=.05;
x=initial_time:timestep:final_time;%time 0 to 20 seconds

t = size(x,2);

if joint == 1
    v_max = 250;
else
    if joint == 2
        v_max = 187;
    else
        if joint == 3
            v_max = 250;
        else
```

```
            if joint == 4
                v_max = 300;
            else
                if joint == 5
                    v_max = 300;
                else
                    if joint == 6
                        v_max = 300;
                    end
                end
            end
        end
    end
end


if line == 1

jointi=[51.84,12.298,-119.428,-103.03,-53.81,111.40];

jointf=[t_t(line,1),t_t(line,2),t_t(line,3),t_t(line,4),t_t(line,5),
t_t(line,6)];
    else
    jointi=[t_t(line-1,1),t_t(line-1,2),t_t(line-1,3),t_t(line-
1,4),t_t(line-1,5),t_t(line-1,6)];

jointf=[t_t(line,1),t_t(line,2),t_t(line,3),t_t(line,4),t_t(line,5),
t_t(line,6)];
end

ji = jointi(joint);
jf = jointf(joint);
dj = jf - ji;
j_vi=0; % Initial Angular velocity is assumed as zero
j_vf=0; % Final Angular velocity is assumed as zero
j_ai=0;% Initial acceleration is assumed as zero
j_af=0; % Final acceleration is assumed as zero


%Coefficients calculation

a_0=ji;
a_1=j_vi;
a_2=j_ai/2;
a_3=[20*(dj)-(8*j_vf+12*j_vi)*T-(3*j_af-j_ai)*T.^2]/(2*T.^3);
a_4=[30*(-dj)+(14*j_vf+16*j_vi)*T+(3*j_af-2*j_ai)*T.^2]/(2*T.^4);
a_5=[12*(dj)-6*(j_vf+j_vi)*T-(j_af-j_ai)*T.^2]/(2*T.^5);
Coefficients_J1=[a_0 a_1 a_2 a_3 a_4 a_5]

t_max = (15*dj)/(v_max*8);
time = abs(t_max);
end
```

5.3 Simulation model

Results taken by running developed algorithms are required to be verified in order to troubleshoot the system and its outcome. Verification of Algorithms developed for forward kinematics and inverse kinematics can be done by performing simulations in a separate simulating software tool. Wincaps III robot programming tool from Densowave corporation is used here for developing robot cell graphically and simulating the experiments to get better results [20].

5.3.1 Creating Robot cell

Robot working cell can be developed virtually in Wincaps III and the summarized procedure is illustrated below.

In Wincaps III, Robot arm can be selected and imported to the simulation module [Figure 5.7].

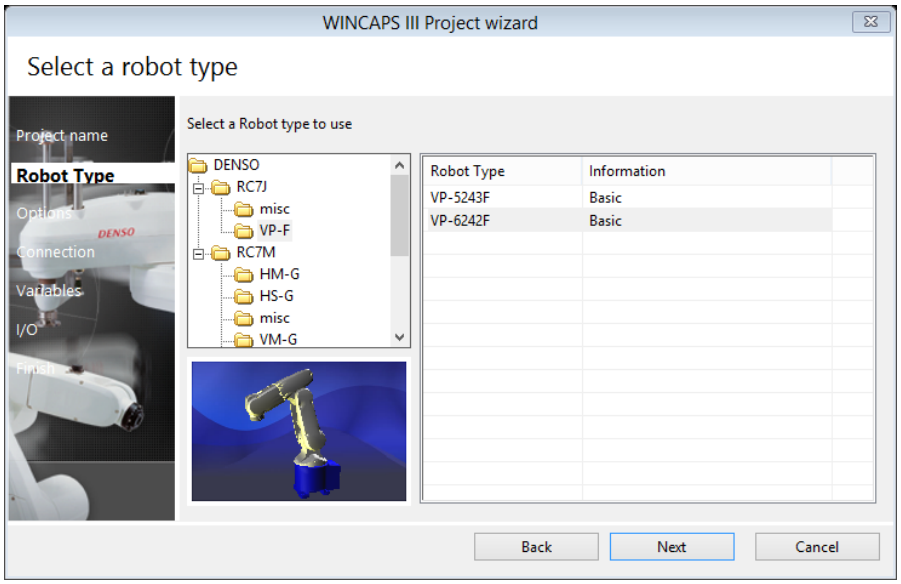


Figure 5.7: Robot Arm selection in Wincaps III

Default configuration is shown in Figure 5.8.

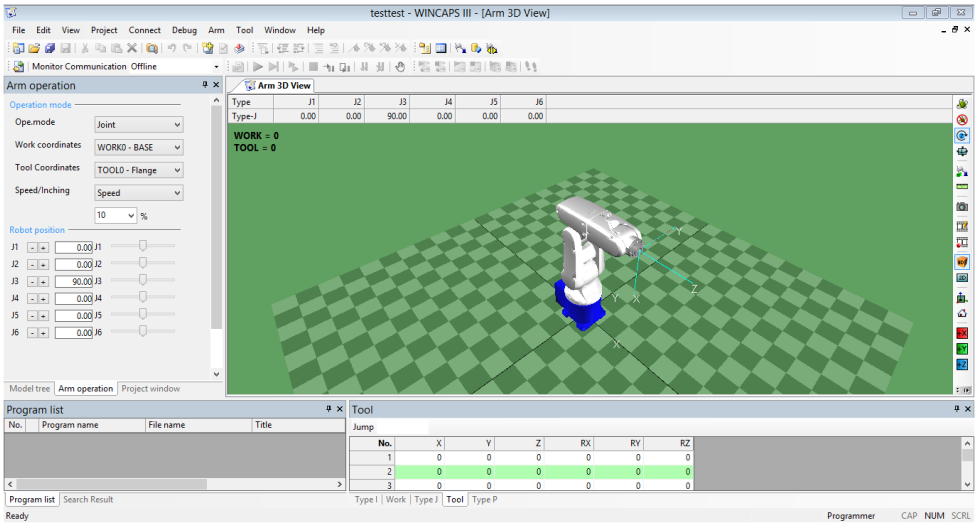


Figure 5.8: Default Robot configuration in Wincaps III

Designs of Machine head cover and the tool can be imported to Wincaps III working module and located according to actual working dimensions [Figure 5.9].

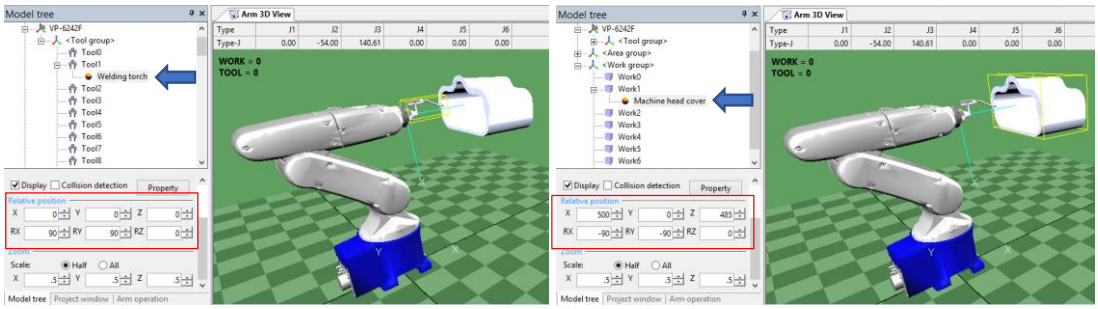


Figure 5.9: Machine head cover and Tool placement in the simulation module

Work and Tool coordinates are defined according to the developed coordinate assignment [Figure 5.10].

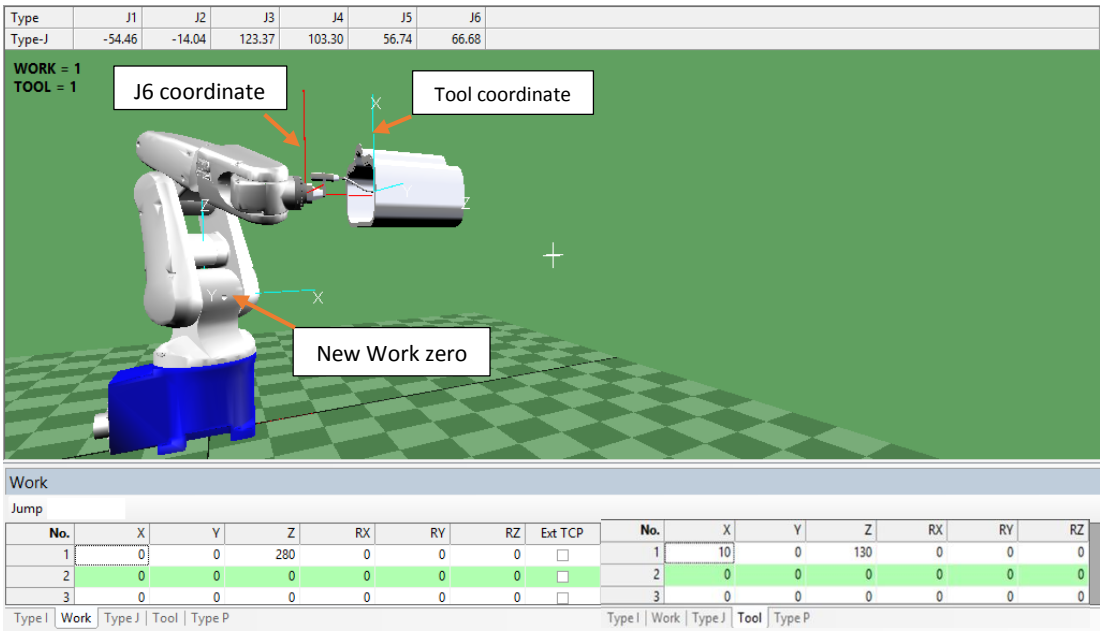


Figure 5.10: Work and Tool coordinate assignment

Robot program is written and executed according to robot movement along the robot end points positioned in the line [Figure 5.11].

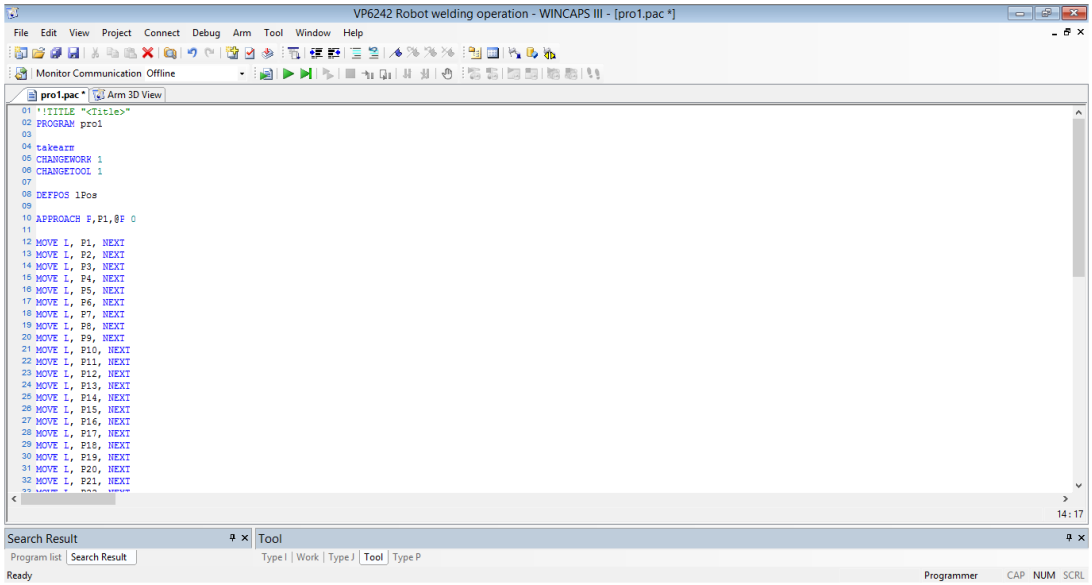


Figure 5.11: Robot programs for simulating robot movement via points

Position and orientation data can be imported to the simulation module in order to run the simulation [Figure 5.12].

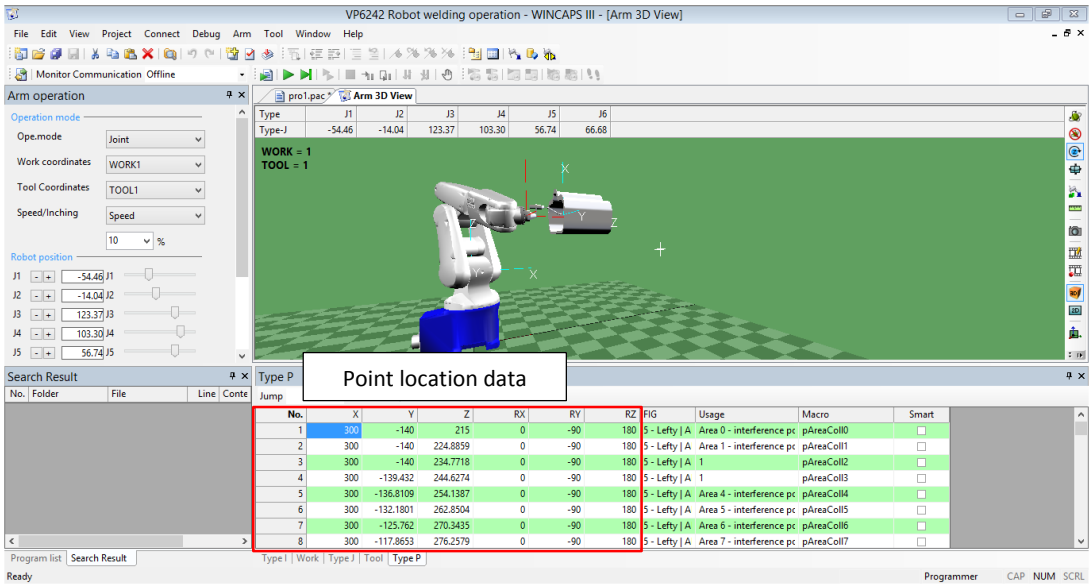


Figure 5.12: Import Point location data into the program

Execution of the program and the simulation can be done according to the user's speed requirement. Joint angle values of each joint can be observed at each robot end point while moving via that point [Figure 5.13]. By observing these joint angles values, experiments can be done to troubleshoot the algorithms and developed methods. Optimization of the solution can be done in order to increase the accuracy.

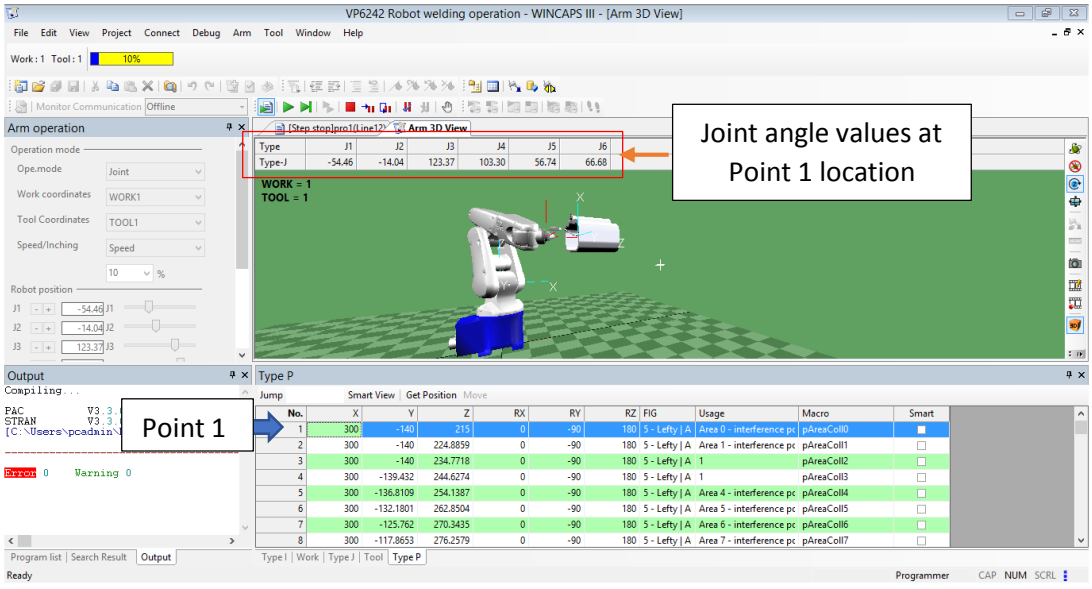


Figure 5.13: Joint angles of the robot arm at point 1 position

CHAPTER 6

RESULTS AND CONCLUSION

Validations of implemented algorithms are discussed in this chapter and systems in chapter 4 and 5. Section 6.1 illustrates simulations done for various point cloud allocation along the robot moving path. Optimization of path planning with trajectory planning is validated by performing several trials. Suitability of using Computer Aided Design features used as an offline robot programming approach for optimizing trajectory planning of 6 DOF robot manipulator in order to enhance the productivity is discussed in the conclusion in section 6.2. The thesis suggests additional developments of the proposed trajectory and path planning scheme which can be implemented in other robotic applications as the upcoming work in section 6.3.

6.1 Various point cloud simulation Results

Simulations for various point cloud of the path validate the efficiency of using offline robot programming techniques for path planning and trajectory planning. By trialing several scenarios with various point to point locations along the path, programmer can conclude an optimized path planning scenario for the application which enhance the productivity of the robot operation.

Simulation objectives:

- Observe and analyze on various cartesian point allocation along a predefined path.
- Identification of Velocity behavior and limits.
- Identification of maximum velocity along task points/sequence.
- Optimization of point allocation in order to get smooth velocity variations along the task sequence.

Procedure

1. Point allocation done as per the user's parameters.
2. Point located data is extracted by running developed VB based macro.
3. Find joint angles for each point location.
4. Verify scenario joint angles results with simulation in Wincaps III.
5. Mapping maximum velocity variation of robot arm joint movement.

Robot task and operation requirements

Path length is 405.33mm and the target cycle time for the operation is needed to be maintained below 60 seconds. [Figure 6.1]

Variation of the robot joint speeds should be low in order to get smooth operation while optimizing the cycle time.

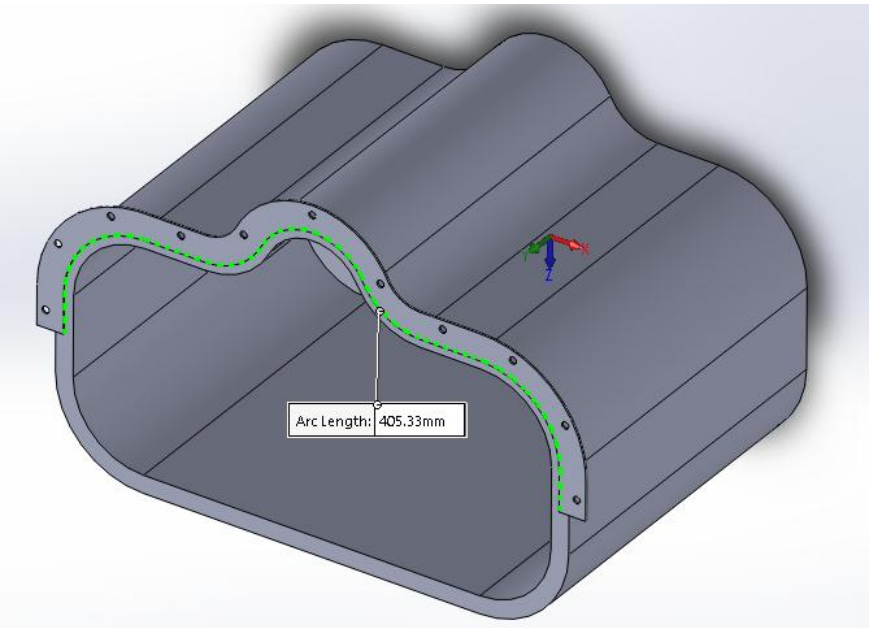


Figure 6.1: Robot path length

Scenario 1

Initial point to point distance can be calculated by taking the average point to point distance which is 6.75mm (405.33mm / 60 point to point intervals) considering point to point time duration as 1 second. These parameters can be considered as base parameters to commence the simulation.

Parameters:

Point to point distance	: 6. 75mm	
Number of points	: 61	
Point to point time duration	: 1 Second	[Figure 6.2]

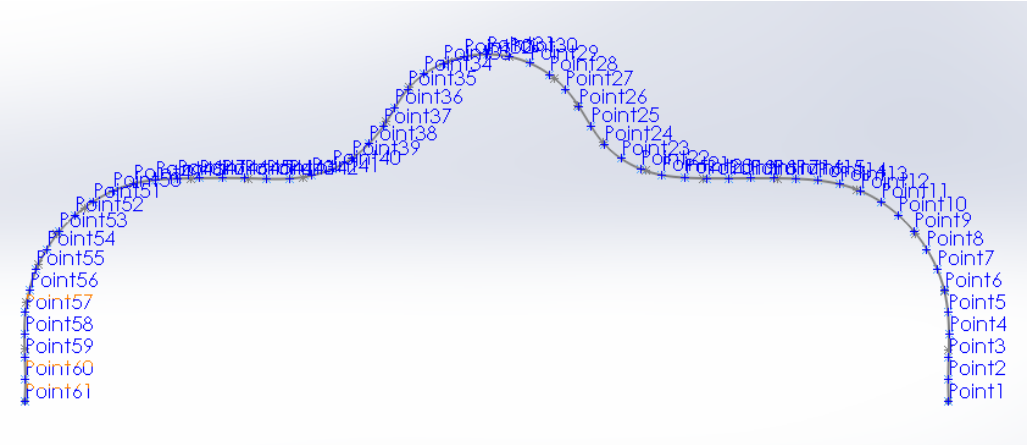


Figure 6.2: Point allocation scenario 1

Scenario 1 results

1. Joint angles can be calculated using developed algorithms as depicted in Appendix C.
2. Simulation verifies that the resulted joint angles values are accurate and set in order [Figure 6.3].

Maximum velocity of joint movement is generated and mapped in order to analyze the robot movement [Figure 6.4].

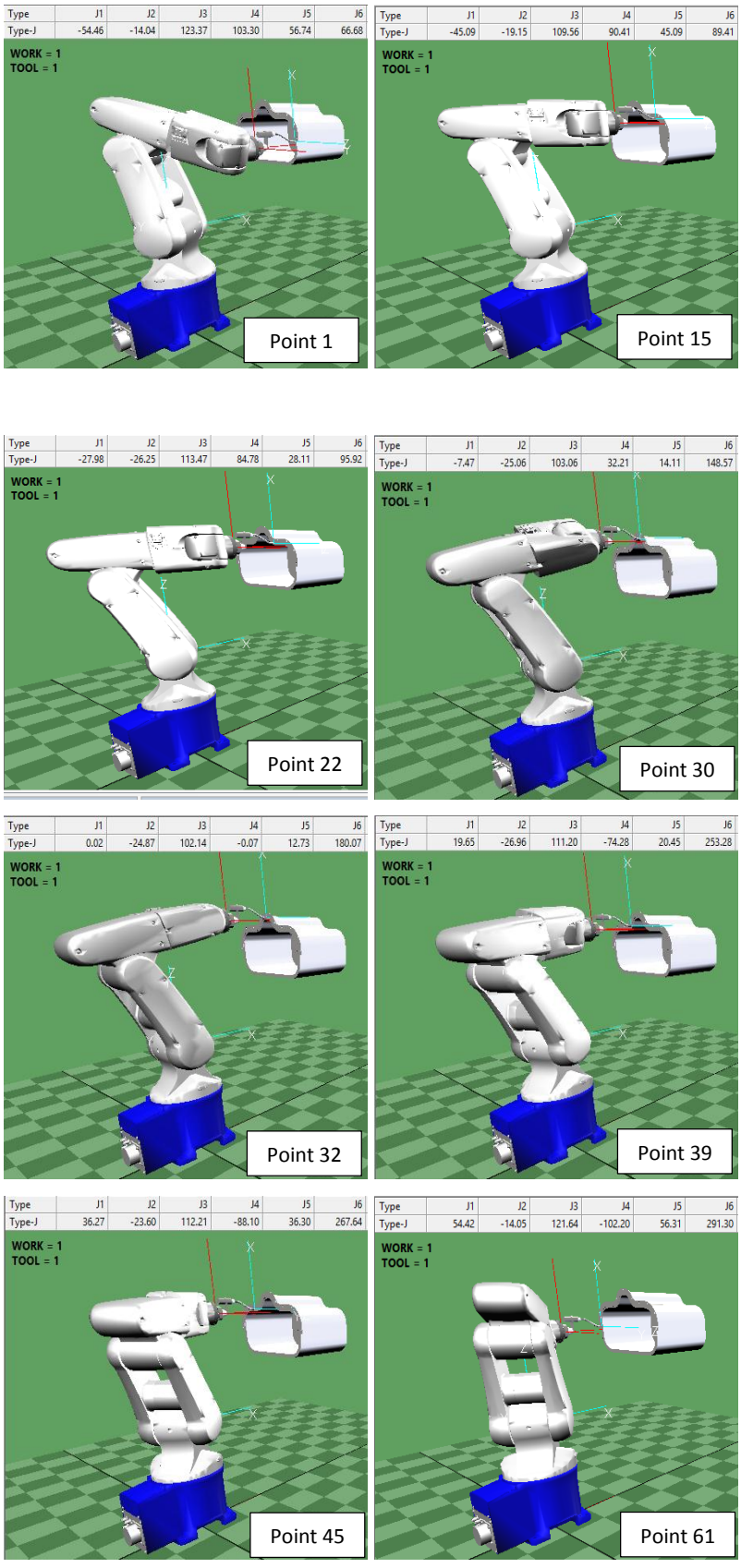


Figure 6.3: Scenario 1 simulation results

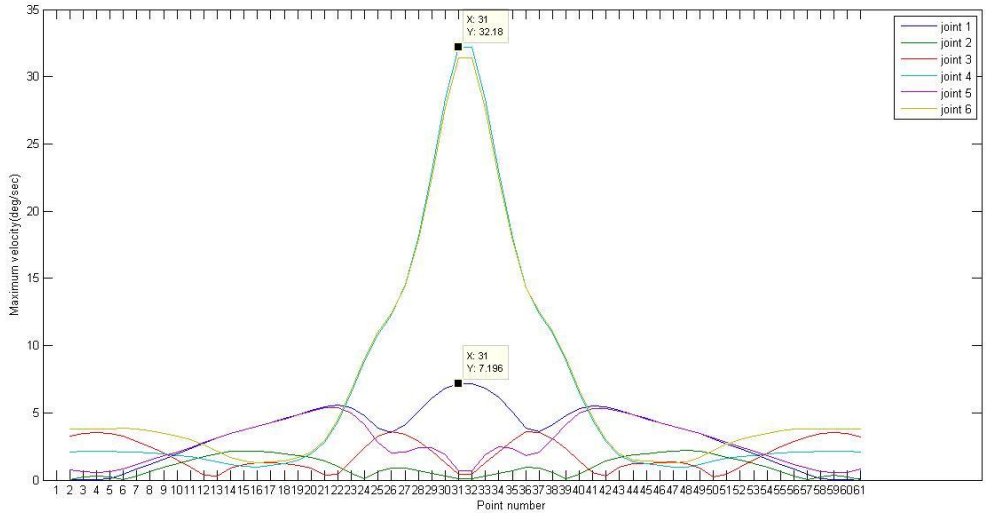


Figure 6.4: Maximum velocity of joint rotation scenario 1

Observations:

- Smooth velocity profiles can be observed.
- Significant variation between joint 4, 6 and other joints can be observed. Maximum velocity is 32.18 deg/sec (joint 4) and the maximum velocity of joint 1 is 7.196 deg/sec. Joint 2, 3 and 5 velocities are varied below joint 1 maximum velocity. Maximum velocity difference between those joints is 24.984 deg/sec.

Scenario 2

In order to reduce the cycle time, number of points has to be reduced by increasing point to point distance. Point to point distance is increased to 9.88mm from 6.75mm.

Parameters:

Point to point distance : 9.88mm (10mm approximately)

Number of points : 42

Point to point time duration : 1 Second [Figure 6.5]

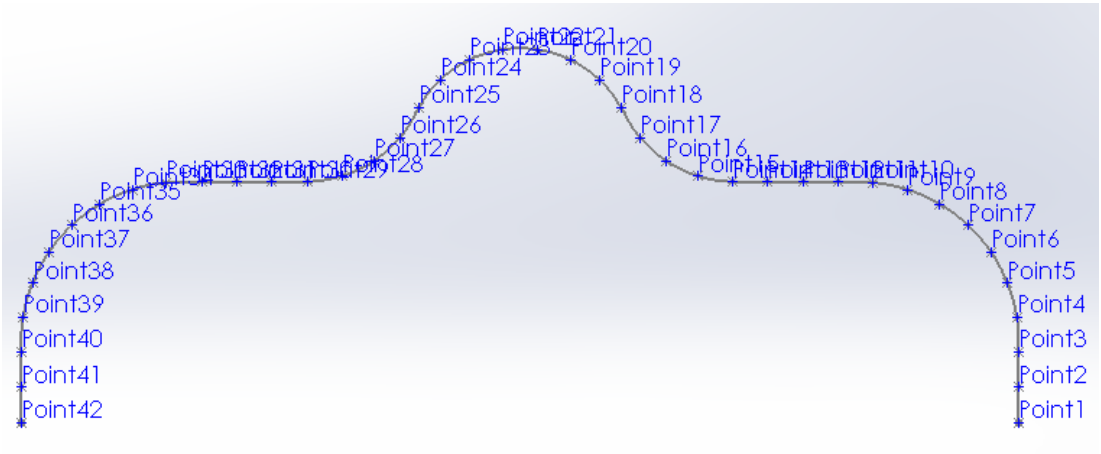


Figure 6.5: Point allocation scenario 2

Scenario 2 results

3. Joint angles can be calculated using developed algorithms as depicted in Table 3.
4. Simulation verifies that the resulted joint angles values are accurate and set in order [Figure 6.6].
5. Maximum velocity of joint movement is generated and mapped in order to analyze the robot movement [Figure 6.7].

Table 3: Scenario 2 joint angles calculation results

Line No:	Joint					
	J1	J2	J3	J4	J5	J6
1	-54.462	-14.039	123.366	103.300	56.737	66.684
2	-54.462	-13.962	120.765	101.667	56.191	69.640
3	-54.462	-13.766	118.067	100.006	55.721	72.606
4	-54.352	-13.581	115.373	98.339	55.215	75.591
5	-53.836	-13.742	113.023	96.723	54.382	78.557
6	-52.891	-14.259	111.135	95.175	53.202	81.402
7	-51.510	-15.112	109.795	93.714	51.662	84.026
8	-49.688	-16.271	109.066	92.370	49.746	86.336
9	-47.432	-17.690	108.984	91.188	47.445	88.244
10	-44.772	-19.312	109.552	90.242	44.772	89.659
11	-41.774	-21.034	110.626	89.543	41.775	90.612
12	-38.464	-22.665	111.647	88.720	38.476	91.635
13	-34.821	-24.170	112.549	87.671	34.854	92.837
14	-30.824	-25.532	113.329	86.315	30.895	94.293
15	-26.542	-26.546	113.443	83.815	26.709	96.917
16	-22.324	-27.005	112.498	79.167	22.751	101.723
17	-18.630	-26.882	110.572	71.942	19.634	109.095
18	-15.867	-26.233	107.829	62.790	17.905	118.383
19	-12.626	-25.553	105.180	51.206	16.288	129.945
20	-8.149	-25.108	103.260	34.894	14.347	145.953
21	-2.819	-24.889	102.251	12.683	12.944	167.629
22	2.819	-24.889	102.251	-12.683	12.944	192.371
23	8.149	-25.108	103.260	-34.894	14.347	214.047
24	12.626	-25.553	105.180	-51.206	16.288	230.055
25	15.867	-26.233	107.829	-62.790	17.905	241.617
26	18.630	-26.882	110.572	-71.942	19.634	250.905
27	22.324	-27.005	112.498	-79.167	22.751	258.277
28	26.542	-26.546	113.443	-83.815	26.709	263.083
29	30.824	-25.532	113.329	-86.315	30.895	265.707
30	34.821	-24.170	112.549	-87.671	34.854	267.163
31	38.464	-22.665	111.647	-88.720	38.476	268.365
32	41.774	-21.034	110.626	-89.543	41.775	269.388
33	44.772	-19.312	109.552	-90.242	44.772	270.341
34	47.432	-17.690	108.984	-91.188	47.445	271.756
35	49.688	-16.271	109.066	-92.370	49.746	273.664
36	51.510	-15.112	109.795	-93.714	51.662	275.974
37	52.891	-14.259	111.135	-95.175	53.202	278.598
38	53.836	-13.742	113.023	-96.723	54.382	281.443
39	54.352	-13.581	115.373	-98.339	55.215	284.409
40	54.462	-13.766	118.067	-100.006	55.721	287.394
41	54.462	-13.962	120.765	-101.667	56.191	290.360
42	54.462	-14.039	123.366	-103.300	56.737	293.316

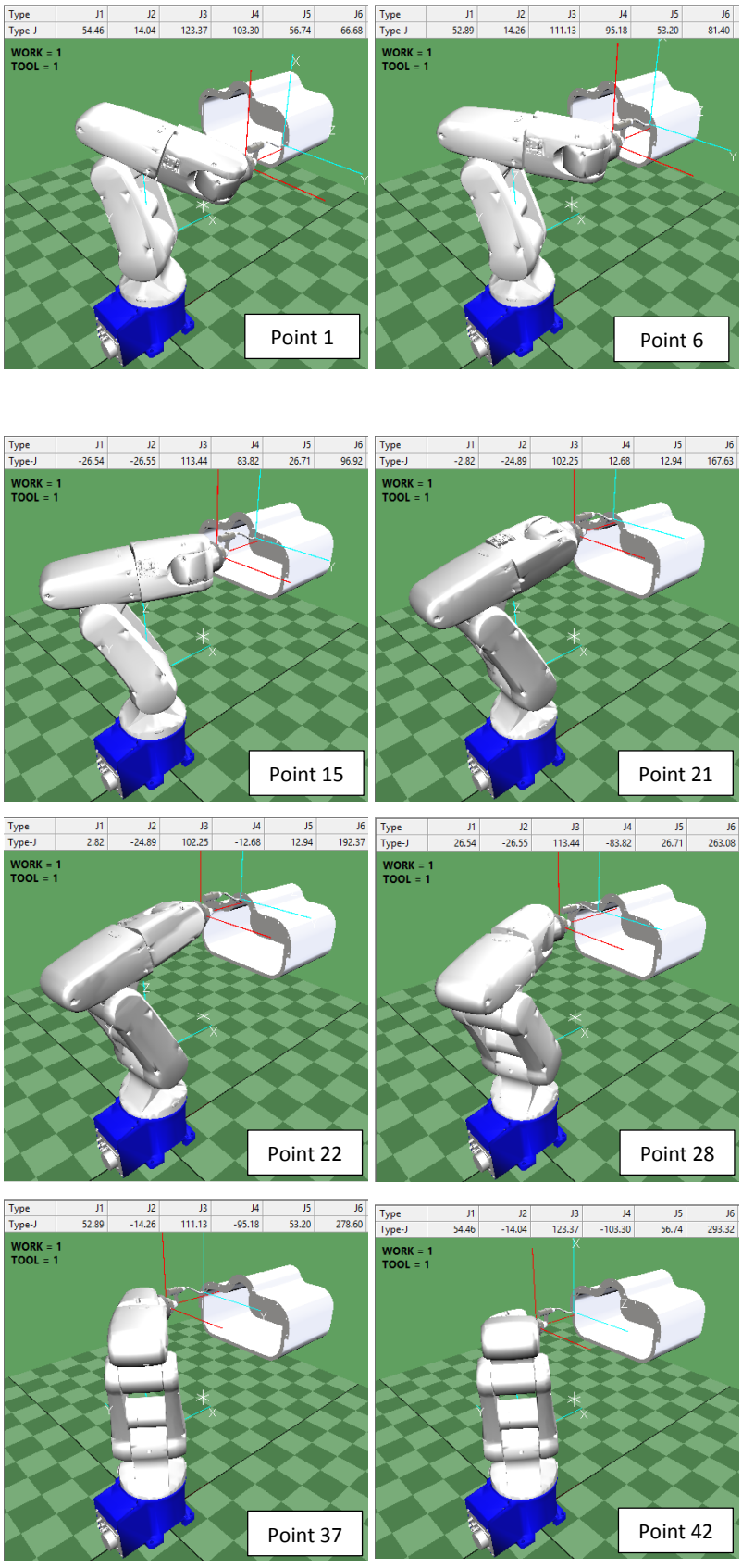


Figure 6.6: Scenario 2 simulation results

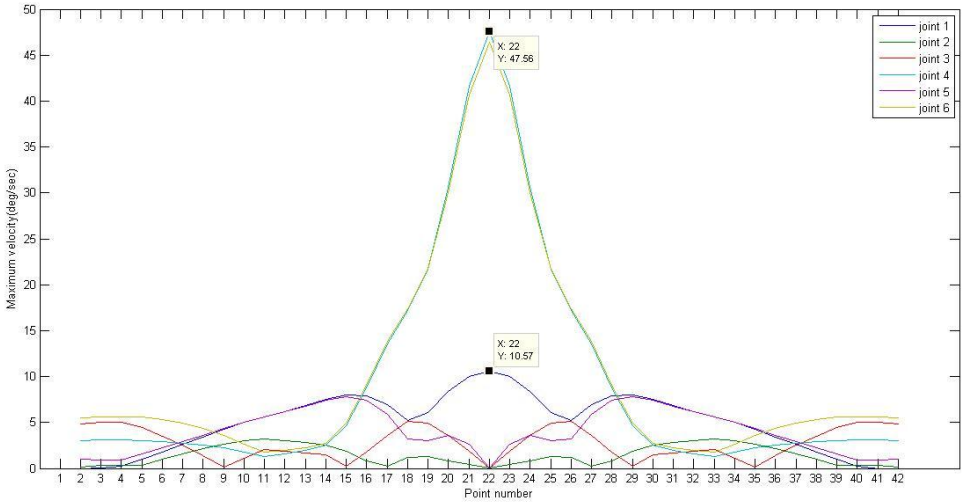


Figure 6.7: Maximum velocity of joint rotation scenario 2

Observations:

- Smooth velocity profiles can be observed.
- Significant variation between joint 4, 6 and other joints can be observed. Maximum velocity is 47.56 deg/sec in joint 4 increased from 32.18 deg/sec and the maximum velocity of joint 1 is 10.57 deg/sec from 7.196 deg/sec. Joint 2, 3 and 5 velocities are varied below joint 1 maximum velocity. Maximum velocity difference between those joints is 36.99 deg/sec increased from 24.984 deg/sec.

Scenario 3

It is realized that the after point 14 up to 30, there is a significant variance in velocities of joint 4 and joint 6 [Figure 6.8]. This variance can be reduced by reducing the distance between points within that area. Therefore, point to point distance between that area has been reduced up to 5mm.

Figure 6.8: Point identification of the velocity variance begins

Parameters:

Point to point distance : 5 and 10mm

Number of points : 57

Point to point time duration : 1 Second [Figure 6.9]

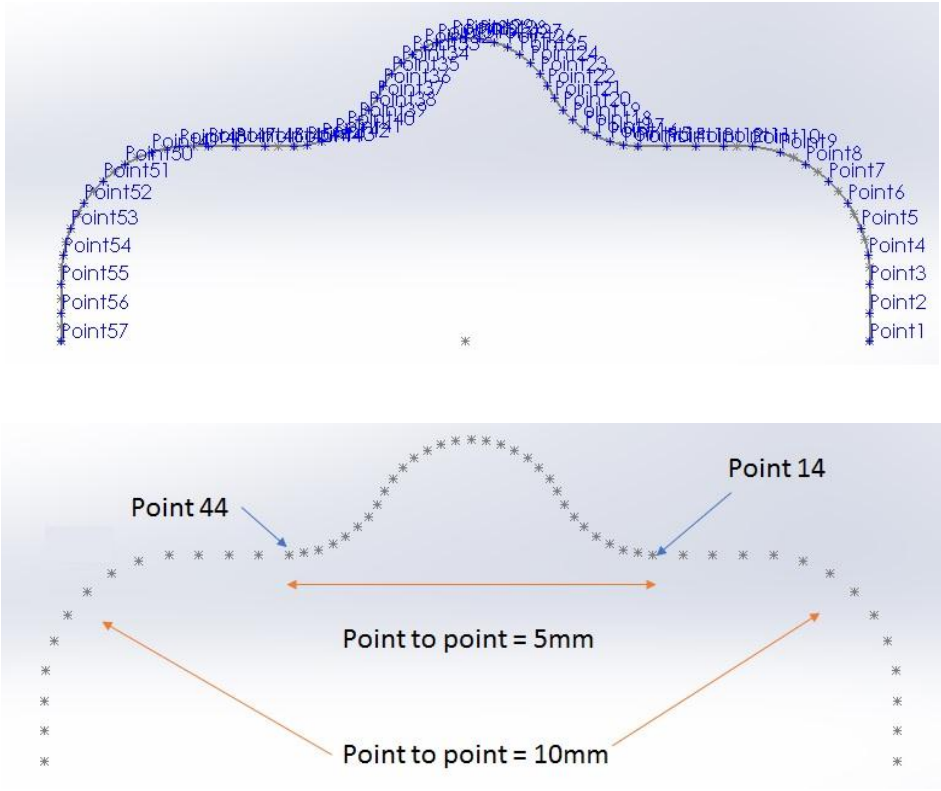


Figure 6.9: Point allocation scenario 3

Scenario 3 results

1. Joint angles can be derived by running the algorithms developed [Appendix D].
2. Simulation verifies that the resulted joint angles values are accurate and set in order [Figure 6.10].
3. Maximum velocity of joint movement is generated and mapped in order to analyze the robot movement [Figure 6.11].

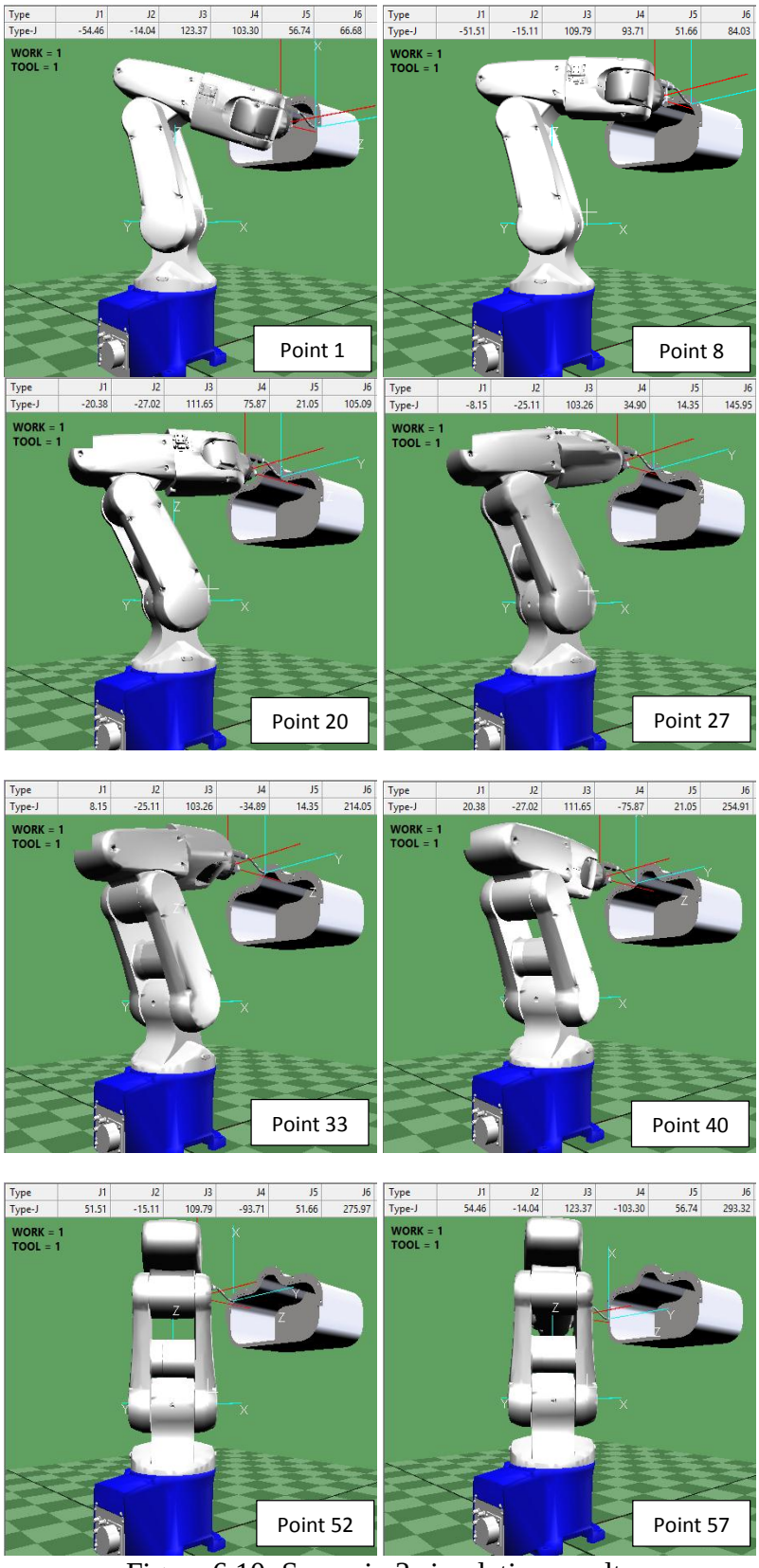


Figure 6.10: Scenario 3 simulation results

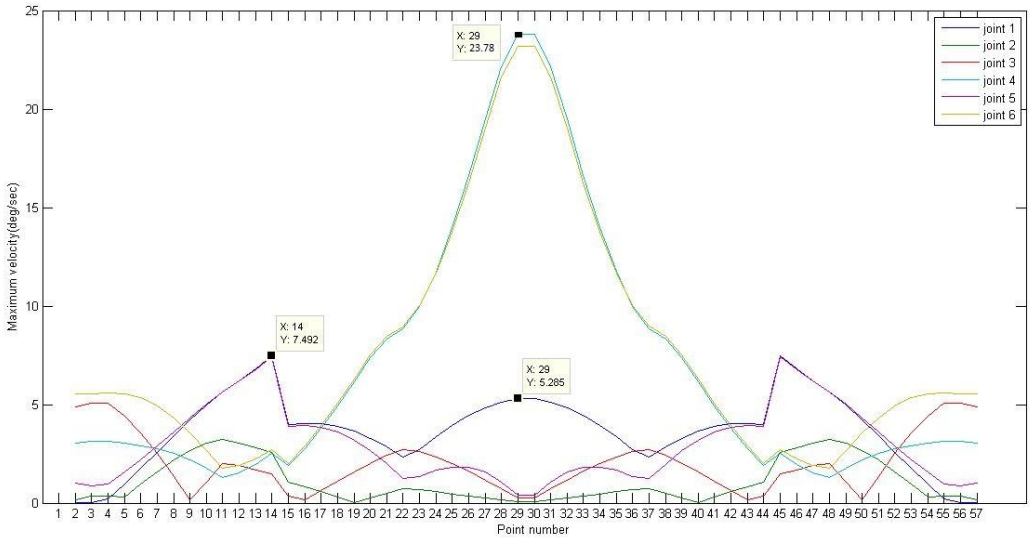


Figure 6.11: Maximum velocity of joint rotation scenario 3

Observations

- Smooth velocity profiles can be observed.
- Sudden changes of the joint 1 and 5 profile can be observed.
- Variation between joint 4, 6 and other joints is reduced than scenario 2. Maximum velocity is 23.78 deg/sec in joint 4 decreased from 47.56 deg/sec and the maximum velocity of joint 1 is 7.492 deg/sec decreased from 10.57 deg/sec. Joint 2, 3 and 5 velocities are varied below joint 1 maximum velocity. Maximum velocity difference between those joints is 16.288 deg/sec decreased from 36.99 deg/sec deg/sec.

Scenario 4

It is realized that the variance of maximum velocities in joint 4 and joint 6 has been reduced and velocities in joint 1 and joint 5 between position 14 to 15 and 44 to 45 are subjected to sudden changes which make robot arm to sudden movement. This incident can be occurred as the point to point distance changes from 10mm to 5mm between

those point locations. This effect can be minimized by allocating points which are located its point to point distances from 10mm to 5mm gradually.

Parameters:

Point to point distance : 10mm (point 1 to 13 and 44 to 56)
5mm (point 17 to 40)

Number of points : 56

Point to point time duration : 1 Second [Figure 6.12]

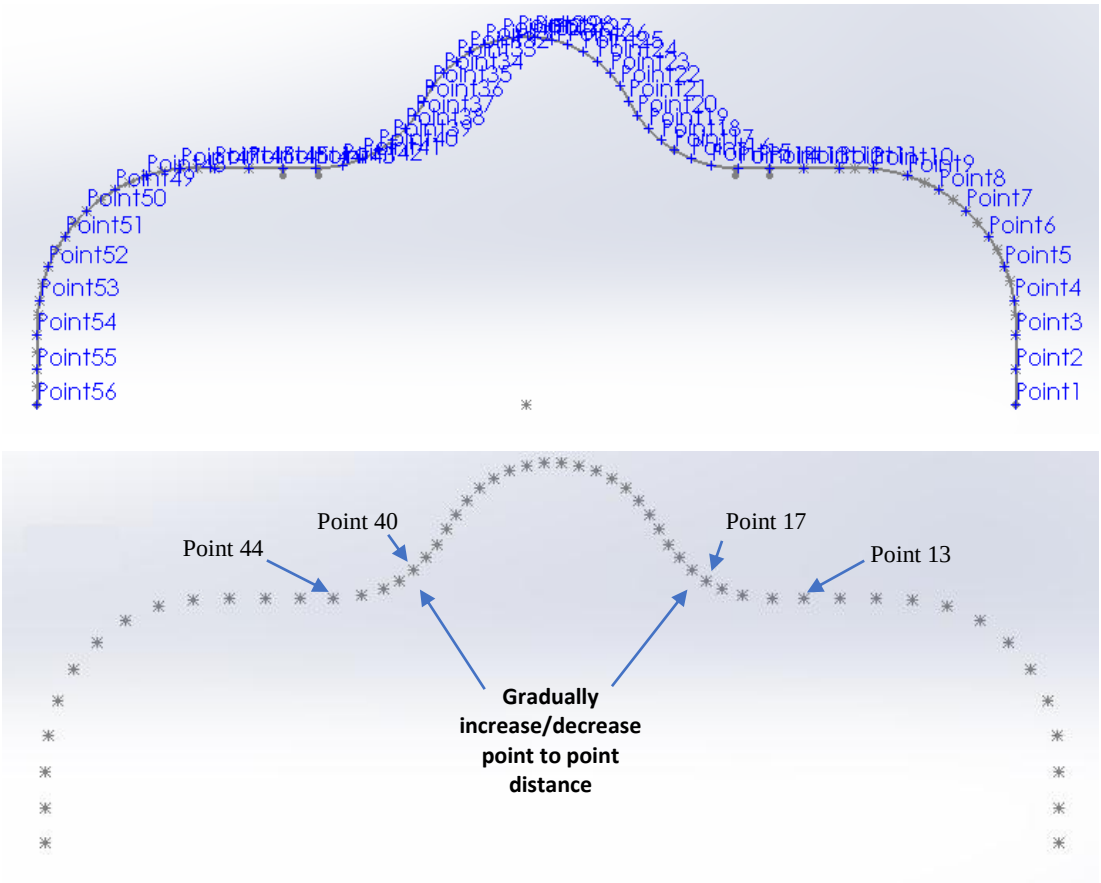


Figure 6.12: Point allocation scenario 4

Scenario 4 results

1. Joint angles can be derived by running the algorithms developed [Appendix E].
2. Simulation verifies that the resulted joint angles values are accurate and set in order [Figure 6.13].
3. Maximum velocity of joint movement is generated and mapped in order to analyze the robot movement [Figure 6.14].

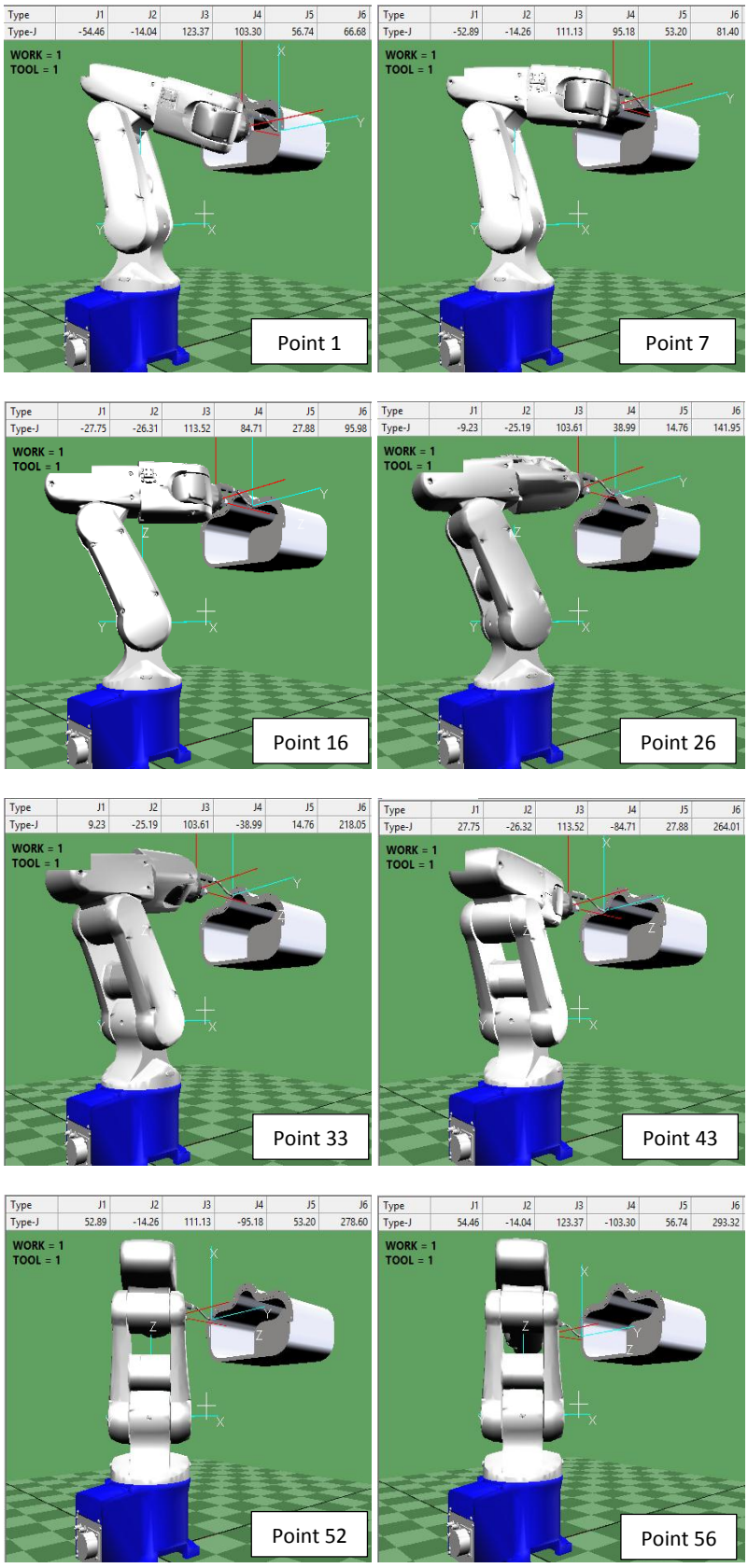


Figure 6.13: Scenario 4 simulation results

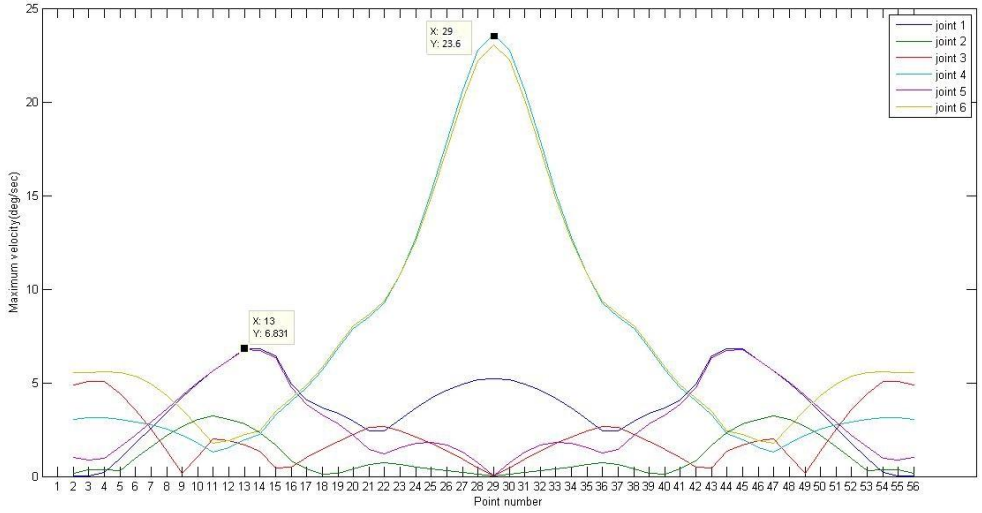


Figure 6.14: Maximum velocity of joint rotation scenario 4

Observations:

- Smooth velocity profiles can be observed.
- Sudden changes of the joint 1 and 5 has been improved. Maximum velocity is 23.6 deg/sec in joint 4 and the maximum velocity of joint 1 is 6.831 deg/sec decreased from 7.492 deg/sec. Joint 2, 3 and 5 velocities are varied below joint 1 maximum velocity. Maximum velocity difference between those joints is 16.769 deg/sec nearly same to scenario 3 results.

Total cycle time is 55 seconds which is below than target cycle time 60 seconds. Maximum velocity variation between joints has been minimized up to 16.769 deg/sec which is a significant improvement from the results of initial simulation and scenarios.

From the above simulations, it is clear that trajectory planning for the robot arm moving along the path can be optimized by realizing the joint speed variations and allocating task points with proper evaluation in order to increase the productivity. This method is introduced as an offline programming technique where programmer can generate the robot programs with optimum output in a different environment separately and the robot operation is not needed to stop for programming.

6.2 Conclusion

SolidWorks features with computer aided design capabilities were used to create the robot cell and generate programs for robot operation with significant accuracy. Path planning of a robot movement can be done efficiently using Solidworks design features and proposed method proves its capability to create the robot working cell virtually. Developed design models can be used frequently for different type of robot cells unless the robot, tool and work piece are not changed.

The proposed method for finding inverse kinematic solutions of the Denso VP 6242 robot manipulator can be used in many working areas in the robot cell and implemented for other types of robot manipulators available in the industry.

Various point clouds with different point to point distances were tested and simulated in Denso Wincaps III software. Optimized point cloud allocation along the robot moving path, was derived gradually by simulating various point to point distances while maintaining smooth robot joint movement. Total cycle time is reduced to 55 seconds which is below required maximum cycle time (60 seconds). Maximum velocity difference between joint 1 and joint 4 has been reduced to 16.769 deg/sec from 24.984 deg/sec which is a significant improvement from the results of initial simulation and scenarios.

Mapping maximum velocity of joint variation between robot moving from one point to another, is very effective to understand overall behavior of the robot motion along the whole path. Simulation 3 Result demonstrates possible changes which are not suitable for better robot operation can be identified and user can improve the path planning according to its outcome needed. Simulation 4 result validates the possibility of arranging robot smooth movement by allocating proper task point cloud. Simulations validate that the point cloud allocation along the robot moving path can be optimized by running several simulations with various point to point distances.

6.3 Future works

Computer Aided Design software and tools are still developing gradually with the development of current manufacturing industries and its requirements hence the efficiency of its utilization can be further improved. Developed inverse kinematic model can be further developed for other types of robot manipulators and can be used for other types of robotic applications.

BIBLIOGRAPHY

[1]"Automobile | Search by industry | usage | industrial robots | DENSO WAVE", *Denso-wave.com*, 2018. [Online]. Available: https://www.denso-wave.com/en/robot/katsuyou/automobile.html. [Accessed: 01- Oct- 2018].

[2]"RC8A controller | Robot controller | products | industrial robots | DENSO WAVE", *Denso-wave.com*, 2018. [Online]. Available: https://www.denso-wave.com/en/robot/product/controller/rc8a.html. [Accessed: 02- Oct- 2018].

[3]P. Neto, J. Pires and A. Moreira, "CAD-based off-line robot programming", *2010 IEEE Conference on Robotics, Automation and Mechatronics*, 2010.

[4]"Robot Offline Programming - Delfoi", *Delfoi.com*, 2018. [Online]. Available: https://www.delfoi.com/web/solutions/robotiikka/en_GB/offline/. [Accessed: 02- Oct- 2018].

[5]L. Ferreira, Y. Figueira, I. Iglesias and M. Souto, "Offline CAD-based Robot Programming and Welding Parametrization of a Flexible and Adaptive Robotic Cell Using Enriched CAD/CAM System for Shipbuilding", *Procedia Manufacturing*, vol. 11, pp. 215-223, 2017.

[6]"FreeCAD: Your Own 3D Parametric Modeler", *Freecadweb.org*, 2018. [Online]. Available: https://www.freecadweb.org/. [Accessed: 02- Oct- 2018].

[7]A. Khan, C. Xiangming, Z. Xingxing and W. Quan, "Closed form inverse kinematics solution for 6-DOF underwater manipulator", *2015 International Conference on Fluid Power and Mechatronics (FPM)*, 2015.

[8]M. Jabbar Hayawi, "Analytical Inverse kinematics Algorithm Of A 5-DOF Robot Arm", *Journal of College of Education for Pure Science*, vol. 1, no. 4, pp. 92-104, 2011.

[9]S. Macfarlane and E. Croft, "Jerk-bounded manipulator trajectory planning: design for real-time applications", *IEEE Transactions on Robotics and Automation*, vol. 19, no. 1, pp. 42-52, 2003.

[10]J. Craig, *Introduction to robotics*. Upper Saddle River, NJ: Pearson Education, 2005.

[11]"VP-6242", *Densorobotics.com*, 2018. [Online]. Available: https://densorobotics.com/content/user_manuals/19/005929.html. [Accessed: 05- Oct- 2018].

[12]"ABB Arc Welding Robots", *RobotWorx*, 2018. [Online]. Available: https://www.robots.com/applications/arc-welding/brands/abb. [Accessed: 23- Oct- 2018].

[13]"Arc Welding Faults - Examples of Speed, Arc Length, and Current Problems", *Mig-welding.co.uk*, 2018. [Online]. Available: https://www.mig-welding.co.uk/arc-welding-faults.htm. [Accessed: 23- Oct- 2018].

[14]M. sources, "Manual welding power sources | Panasonic Industry Europe", *Eu.industrial.panasonic.com*, 2018. [Online]. Available: https://eu.industrial.panasonic.com/products/robot-welding-system-solutions/power-sources/manual. [Accessed: 23- Oct- 2018].

[15]"Aristo® RT - Arc welding torch / air-cooled / robotic by ESAB | DirectIndustry", *Directindustry.com*, 2018. [Online]. Available: http://www.directindustry.com/prod/esab/product-18224-1722887.html. [Accessed: 23- Oct- 2018].

[16]S. Kucuk and Z. Bingul, "Robot Kinematics: Forward and Inverse Kinematics", *Industrial Robotics: Theory, Modelling and Control*, 2006. Available: 10.5772/5015.

[17]J. Denavit and R. Hartenberg, "A Kinematic Notation for Lower-pair Mechanisms Based on Matrices", *Journal of Applied Mechanics*, vol. 22, pp. 215-221, 1955.

[18]"3D CAD Design Software", *Solidworks.com*, 2018. [Online]. Available: https://www.solidworks.com/. [Accessed: 23- Oct- 2018].

[19]"MATLAB - MathWorks", Mathworks.com, 2018. [Online]. Available: https://www.mathworks.com/products/matlab.html. [Accessed: 23- Oct- 2018].

[20]"WINCAPSⅢ | Software | products | industrial robots | DENSO WAVE", Denso-wave.com, 2018. [Online]. Available: https://www.denso-wave.com/en/robot/product/software/wincaps3.html. [Accessed: 23- Oct- 2018].

APPENDICES

Appendix A

- Algorithms for inverse kinematic solution of 6 DOF robot manipulator and trajectory planning in Matlab
- Machine head cover dimensions
- Robotic cell layout

Appendix B - Dimensions and the configuration of the Denso VP 6242

Appendix C - Scenario 2 joint angles calculation results

Appendix D - Scenario 3 joint angles calculation results

Appendix A

Algorithms for inverse kinematic solution of 6 DOF robot manipulator and trajectory planning in Matlab

The main program

```
clear all;
clc;

syms c1 s1 d3 c23 s23 s2 c2 al be t l
syms m n p
syms s1 s2 s3 s4 s5 s6 c1 c2 c3 c4 c5 c6;
global tt;



%position read

C = xlsread('C:\Users\pcadmin\Desktop\MSc\research
matlab\simulation3.xlsx','Sheet1');
tam = size(C);

Pa = [];
for i = 1 : tam

    T6EF = [1 0 0 10;0 1 0 0;0 0 1 130;0 0 0 1];

    X(i,1) = C(i,1);%X value
    Y(i,2) = C(i,2);%Y value
    Z(i,3) = C(i,3);%Z value

    al = C(i,6); %angle arond Z axis
    be = C(i,5);%angle around Y axis
    ga = C(i,4);%angle around X axis

    %Eular angle Z,Y,X
    Rz = [  cosd(al) -sind(al)      0;
            sind(al) cosd(al)       0;
            0           0           1;];

    Ry = [  cosd(be)    0           sind(be);
            0           1           0;
            -sind(be)   0           cosd(be);];

    Rx = [  1           0           0;
            0           cosd(ga)    -sind(ga);
            0           sind(ga)    cosd(ga);];
    %Rotation Matrix
    ARB = Rz*Ry*Rx;

TT(:, :, i) = [A_R_B(1,1) A_R_B(1,2) A_R_B(1,3) X(i,1); A_R_B(2,1)
A_R_B(2,2) A_R_B(2,3) Y(i,2); A_R_B(3,1) A_R_B(3,2) A_R_B(3,3)
Z(i,3);
0 0 0 1];
T6EFT = ((T6EF)^(-1));
T(:, :, i) = TT(:, :, i)* T6EFT;
%Pa position
x = T(1,4,i)- (70)*T(1,3,i);%Pa x position
y = T(2,4,i)- (70)*T(2,3,i);%Pa y position
```

```
z = T(3,4,i)- (70)*T(3,3,i);%Pa z position

distance = (x^2 + y^2 + z^2)^(1/2);
if distance > 420
    disp('error');
    pause;
end

%Calculating theta 1
theta1 = atan2d(y,x);


if theta1 == 180
    theta1 = 0;
end


if (90<theta1) && (theta1<180)
    theta1 = -(180 - theta1);
end

if (-180>theta1) && (theta1>-90)
    theta1 = 180 + theta1;
end
%}


%Calculating theta 2 and theta 3
c1 = cosd(theta1);
s1 = sind(theta1);

l = (x*c1) + (y*s1);
m = (-l/663) + (14*z/3315);
n = ((-14*l)/3315) - (z/663);
p = (196/221) + (m^2) + (n^2);

a_ = ((-140*m) - (392*n))/221;
b_ = ((140*n) - (392*m))/221;

c = 1 - p;

theta2 = atan2(a_,b_)+ atan2(((a_^2 + b_^2 - c^2))^(1/2),c);
theta2 = (theta2*180)/pi;

if theta2 ==360
    theta2 = 0;
end

if (180<theta2) && (theta2<360)
theta2 = 360 - theta2;
end

%Theta 3
```

```
e = -210.00;
f = -75.00;
g = l+(210*(sind(theta2)));
h = ((e^2 + f^2 - g^2))^(1/2);

if isreal(h) == 1
    theta_23_1 = atan2(e,f)+ atan2(h,g);
    theta_23_1 = (theta_23_1*180)/pi;

    theta_23_2 = atan2(e,f)- atan2(h,g);
    theta_23_2 = (theta_23_2*180)/pi;

    if (theta_23_2>-360) && (theta_23_2<-180)
        theta_23_2 = 360 + theta_23_2;
    end

    theta_3_1 = - theta2 + theta_23_1;

    theta_3_2 = - theta2 + theta_23_2;

else
    theta_3_1 = 0 - theta2;

    theta_3_2 = 0 - theta2;
end

theta2_ = atan2(a,b)- atan2((a_^2 + b_^2 - c^2)^(1/2),c);
theta2_ = (theta2_*180)/pi;

if ( theta2_ > -360) && (theta2_ < -180)
    theta2_ = 360 + theta2_;
end


e = -210.00;
f = -75.00;
g_ = l+(210*(sind(theta2_)));
h_ = (e^2 + f^2 - g_^2)^(1/2);

if isreal(h_) == 1
    theta_23_3 = atan2(e,f)+ atan2(h_,g_);
    theta_23_3 = (theta_23_3*180)/pi;

    theta_23_4 = atan2(e,f)- atan2(h_,g_);
    theta_23_4 = (theta_23_3*180)/pi;

    theta_3_3 = - theta2_ + theta_23_3;

    theta_3_4 = - theta2_ + theta_23_4;

else
    theta_3_3 = 0 - theta2_;

    theta_3_4 = 0 - theta2_;
```

```
end

%calculating theta 4,5,6 - four solutions

[theta_4_1 theta_5_1 theta_6_1] =
theta456calcalternative(theta1,theta2,theta_3_1,T(:,:,i))

[Orientation1 position1] = forwardkinamatic(theta1, theta2,
theta_3_1, theta_4_1, theta_5_1, theta_6_1,T(:,:,i))



[theta_4_2 theta_5_2 theta_6_2] =
theta456calcalternative(theta1,theta2,theta_3_2,T(:,:,i))

[Orientation2 position2] = forwardkinamatic(theta1, theta2,
theta_3_2, theta_4_2, theta_5_2, theta_6_2,T(:,:,i))


[theta_4_3 theta_5_3 theta_6_3] =
theta456calcalternative(theta1,theta2_,theta_3_3,T(:,:,i))

[Orientation3 position3] = forwardkinamatic(theta1, theta2_,
theta_3_3, theta_4_3, theta_5_3, theta_6_3,T(:,:,i))


[theta_4_4 theta_5_4 theta_6_4] =
theta456calcalternative(theta1,theta2_,theta_3_4,T(:,:,i))

[Orientation4 position4] = forwardkinamatic(theta1, theta2_,
theta_3_4, theta_4_4, theta_5_4, theta_6_4,T(:,:,i))

%four solutions in table
[table(:,1,i)] =
[position1;theta1;theta2;theta_3_1;theta_4_1;theta_5_1;theta_6_1];
[table(:,2,i)] =
[position2;theta1;theta2;theta_3_2;theta_4_2;theta_5_2;theta_6_2];
[table(:,3,i)] =
[position3;theta1;theta2_;theta_3_3;theta_4_3;theta_5_3;theta_6_3];
[table(:,4,i)] =
[position4;theta1;theta2_;theta_3_4;theta_4_4;theta_5_4;theta_6_4];

for k = 1:4

        dx(i,k)= table(1,k,i)-T(1,4,i);
        dy(i,k)= table(2,k,i)-T(2,4,i);
        dz(i,k)= table(3,k,i)-T(3,4,i);


if (abs(dx(i,k))>0) && (abs(dx(i,k))< 0.001)
    dx(i,k)= 0;
else
    dx(i,k)= dx(i,k);
end
```

```
if (abs(dy(i,k))>0) && (abs(dy(i,k))< 0.001)
    dy(i,k)= 0;
else
    dy(i,k)= dy(i,k);
end

if (abs(dz(i,k))>0) && (abs(dz(i,k))< 0.001)
    dz(i,k)= 0;
else
    dz(i,k)= dz(i,k);
end


end
end;

%Selecting correct solution
tt(1,1:6)=[0 0 0 0 0 0];

for o = 1:tam
for j = 1:4
if dx(o,j)== 0 && dy(o,j) == 0 && dz(o,j) == 0
    tt(o,1) = table(4,j,o);
    tt(o,2) = table(5,j,o);
    tt(o,3) = table(6,j,o);
    tt(o,4) = table(7,j,o);
    tt(o,5) = table(8,j,o);
    tt(o,6) = table(9,j,o);

end
end
end;


for aa = 1:tam
    for bb = 1:6
        if (abs(tt(aa,bb))>0) && (abs(tt(aa,bb))< 0.001)

            tt(aa,bb)=0;
        else
            tt(aa,bb) = tt(aa,bb);
        end
    end
end;


for ii = 1:tam
    if Y(ii,2)<0
    tt(ii,4) = 180+tt(ii,4);
    else
      tt(ii,4) = -180+tt(ii,4);
    end

    if tt(ii,4) > 270
```

```
        tt(ii,4) = tt(ii,4) - 360;
    end

    if tt(ii,4) < -270
        tt(ii,4) = tt(ii,4) + 360;
    end

    tt(ii,5) = -tt(ii,5);

    tt(ii,6) = -180+tt(ii,6);

end

for i = 1:tam
    if 90<tt(i,4) && tt(i,4)<180
        tt(i,6) = tt(i,6);
    else if 0<tt(i,4) && tt(i,4)<90
            tt(i,6) = -90 - (tt(i,6)+90);
            else if 0>tt(i,4) && tt(i,4)>-90
                tt(i,6) = -90 - (tt(i,6)+90);
                    else if -90>tt(i,4) && tt(i,4)>-180
                        tt(i,6) = 90 + (270 + tt(i,6));
                        end
                end
        end
    end
end

for i= 1:tam
    tt(i,6) = 180 + tt(i,6);
end

tttable =
array2table(tt,'VariableNames',{'J1','J2','J3','J4','J5','J6'});

%Denso simulation 1
Densot(:,1) = tt(:,1);
Densot(:,2) = -tt(:,2);
Densot(:,3) = -tt(:,3);
Densot(:,4) = tt(:,4);
Densot(:,5) = -tt(:,5);
Densot(:,6) = tt(:,6);

Densottable =
array2table(Densot,'VariableNames',{'J1','J2','J3','J4','J5','J6'});

for i = 1:tam

[jtable vtable atable qtable] = trajectory(i,1);%line number,time

vsize = size(vtable);

    for q = 2:7
        vmax(i,q-1) = max(abs(vtable(:,q)))
    end
```

```
end



vmaxtable =
array2table(vmax(2:tam,:),'VariableNames',{'J1','J2','J3','J4','J5',
'J6'});

figure;
plot(2:tam,vmax(2:tam,1:6))
ylabel('Maximum velocity(deg/sec)');xlabel('Line
number');set(gca,'XTick',[1:1:tam]);
legend('joint 1','joint 2','joint 3','joint 4','joint 5','joint 6')
%maxx = max(abs(vtable(:,3)));


for ii = 1:tam
    for jj=1:6
        if vmax(ii,1)>250
            vmaxd(ii,1)=vmax(ii,1)-250;
        else
            vmaxd(ii,1)= 0;
        end
        if vmax(ii,2)>187
            vmaxd(ii,2)=vmax(ii,2)-187;
        else
            vmaxd(ii,2)= 0;
        end
        if vmax(ii,3)>250
            vmaxd(ii,3)=vmax(ii,3)-250;
        else
            vmaxd(ii,3)= 0;
        end
        if vmax(ii,4)>300
            vmaxd(ii,4)=vmax(ii,4)-300;
        else
            vmaxd(ii,4)= 0;
        end
        if vmax(ii,5)>300
            vmaxd(ii,5)=vmax(ii,5)-300;
        else
            vmaxd(ii,5)= 0;
        end
        if vmax(ii,6)>300
            vmaxd(ii,6)=vmax(ii,6)-300;
        else
            vmaxd(ii,6)= 0;
        end
    end
end

for ii=1:tam

        vd(ii,1) = max(vmaxd(ii,:));
        for jj=1:6
            if vd(ii,1)== vmaxd(ii,jj)
```

```
                vd(ii,2) = jj;
            end
                if vd(ii,1) == 0
                vd(ii,2) = 0;
                end

        end
end

%Displaying velocity max values in same vmax table
for i = 1:tam
        if vmax(i,1)>250
            vmax(i,7)=vmax(i,1);
        end
        if vmax(i,2)>187
            vmax(i,8)=vmax(i,2);
        end
        if vmax(i,3)>250
            vmax(i,9)=vmax(i,3);
        end
        if vmax(i,4)>300
            vmax(i,10)=vmax(i,4);
        end
        if vmax(i,5)>300
            vmax(i,11)=vmax(i,5);
        end
        if vmax(i,6)>300
            vmax(i,12)=vmax(i,6);
        end
end
```

Sub functions 1

```
function [Orientation position] =
forwardkinamatic(theta_1,theta_2,theta_3,theta_4,theta_5,theta_6,T)

format short g
al_0=0;al_1=90,al_2=0;al_3=90;al_4=-90;al_5=90;
a_0=0;a_1=0;a_2=210;a_3=75;a_4=0;a_5=0;
d_1=0;d_2=0;d_3=0;d_4=210;d_5=0;d_6=70;
t_1=theta_1;
t_2=90+theta_2;
t_3=90+theta_3;
t_4=theta_4;
t_5=theta_5;
t_6=theta_6;
T01 = [cosd(t_1),         -sind(t_1),              0,           a_0
;
sind(t_1)*cosd(al_0), cosd(t_1)*cosd(al_0),     -sind(al_0), -
sind(al_0)*d_1;
sind(t_1)*sind(al_0), cosd(t_1)*sind(al_0),     cosd(al_0),
cosd(al_0)*d_1 ;
0,                    0,                        0,           1
;   ]

T12 = [ cosd(t_2),           -sind(t_2),            0,           a_1
;
sind(t2)*cosd(al_1), cosd(t_2)*cosd(al_1),     -sind(al_1), -
sind(al_1)*d_2;
sind(t_2)*sind(al_1), cosd(t_2)*sind(al_1),     cosd(al_1),
cosd(al_1)*d_2 ;
0,                    0,                        0,           1
;   ]

T23 = [ cosd(t_3),           -sind(t_3),              0,
a_2           ;
sind(t_3)*cosd(al_2), cosd(t_3)*cosd(al_2),     -sind(al_2), -
sind(al2)*d_3;
sind(t_3)*sind(al_2), cosd(t_3)*sind(al_2),     cosd(al_2),
cosd(al_2)*d_3 ;
0,                    0,                        0,           1
;   ]

T34 = [ cosd(t_4),           -sind(t_4),              0,
a_3           ;
sind(t_4)*cosd(al_3), cosd(t_4)*cosd(al_3),     -sind(al_3), -
sind(al3)*d_4;
sind(t_4)*sind(al_3), cosd(t_4)*sind(al_3),     cosd(al_3),
cosd(al_3)*d_4 ;
0,                    0,                        0,           1
;   ]

T45 = [ cosd(t_5),           -sind(t_5),              0,
a_4           ;
```

```
sind(t_5)*cosd(al4), cosd(t_5)*cosd(al_4),     -sind(al_4), -
sind(al_4)*d_5;
sind(t_5)*sind(al_4), cosd(t_5)*sind(al_4),     cosd(al_4),
cosd(al_4)*d_5 ;
          0,                  0,                      0,
1              ;   ]

T56 = [ cosd(t_6),           -sind(t_6),              0,
a_5           ;
sind(t_6)*cosd(al_5), cosd(t_6)*cosd(al_5),     -sind(al_5), -
sind(al_5)*d_6;
sind(t_6)*sind(al_5), cosd(t_6)*sind(al_5),     cosd(al_5),
cosd(al_5)*d_6 ;
         0,                  0,                      0,
1              ;   ]


T_06 = T01 * T12 * T23 * T34 * T45 * T56 ;


Orientation = T_06(1:3,1:3)

position = T_06(1:3,4)

end
```

Sub function 2

```
function [theta4 theta5 theta6 R6_3, T4_0] =
theta456calc(theta_1,theta_2,theta_3,T)

%DH parameters
al_0=0;al_1=90,al_2=0;al_3=90;al_4=-90;al_5=90;
a_0=0;a_1=0;a_2=210;a_3=75;a_4=0;a_5=0;
d_1=0;d_2=0;d_3=0;d_4=210;d_5=0;d_6=70;
t_1=theta_1;t_2=90+theta_2;t_3=90+theta_3;

T1_0 = [ cosd(t_1),           -sind(t_1),              0,
a_0           ;
        sind(t_1)*cosd(al_0), cosd(t_1)*cosd(al_0),     -sind(al_0),
-sind(al_0)*d_1;
        sind(t_1)*sind(al_0), cosd(t_1)*sind(al_0),     cosd(al_0),
cosd(al_0)*d_1 ;
        0,                  0,                      0,          1
;   ]

T2_1 = [ cosd(t_2),           -sind(t_2),              0,
a_1           ;
        sind(t_2)*cosd(al_1), cosd(t_2)*cosd(al_1),     -sind(al_1),
-sind(al_1)*d_2;
        sind(t_2)*sind(al_1), cosd(t_2)*sind(al_1),     cosd(al_1),
cosd(al_1)*d_2 ;
        0,                  0,                      0,          1
;   ]

T3_2 = [ cosd(t_3),           -sind(t_3),              0,
a_2           ;
        sind(t_3)*cosd(al_2), cosd(t_3)*cosd(al_2),     -sind(al_2),
-sind(al_2)*d_3;
        sind(t_3)*sind(al_2), cosd(t_3)*sind(al_2),     cosd(al_2),
cosd(al2)*d_3 ;
        0,                  0,                      0,          1
;   ]

%Calculating theta4
T3_0 = T1_0*T2_1*T3_2;


R03 = [T3_0(1,1:3);
        T3_0(2,1:3);
        T3_0(3,1:3);];

R03T = transpose(R03);

R06 = [T(1,1:3);
        T(2,1:3);
        T(3,1:3);];

R6_3 =  R03T*R06;
```

```
theta4 = atan2d(R6_3(3,3),R6_3(1,3));


if theta4 == 180
    theta4 = 0;
end


%

%Calculating theta 5
T_4 = theta4;
T4_3 = [ cosd(t_4),           -sind(t_4),              0,
a_3           ;
        sind(t_4)*cosd(al_3), cosd(t_4)*cosd(al_3),     -sind(al_3),
-sind(al_3)*d_4;
        sind(t_4)*sind(al_3), cosd(t_4)*sind(al_3),     cosd(al_3),
cosd(al_3)*d_4 ;
        0,                  0,                      0,          1
;   ]

T4_0 = T3_0 * T4_3;


T4_0T = transpose(T4_0);
T46 = T4_0T * T;
theta5 = asind(T46(1,3));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Calculating theta 6
t_4 = theta4;
t_5 = theta5;

T4_3 = [ cosd(t_4),           -sind(t_4),              0,
a_3           ;
        sind(t_4)*cosd(al_3), cosd(t_4)*cosd(al_3),     -sind(al_3),
-sind(al_3)*d_4;
        sind(t_4)*sind(al_3), cosd(t_4)*sind(al_3),     cosd(al_3),
cosd(al_3)*d_4 ;
        0,                  0,                      0,          1
;   ]

T5_4 = [ cosd(t_5),           -sind(t_5),              0,
a_4           ;
        sind(t_5)*cosd(al_4), cosd(t_5)*cosd(al_4),     -sind(al_4),
-sind(al_4)*d_5;
        sind(t_5)*sind(al_4), cosd(t_5)*sind(al_4),     cosd(al_4),
cosd(al_4)*d_5 ;
        0,                  0,                      0,          1
;   ]

T5_0 = T3_0*T4_3*T5_4;
```

```
R05 = [T5_0(1,1:3);
       T5_0(2,1:3);
       T5_0(3,1:3);];

R5_0T = transpose(R05);

R6_0 = [T(1,1:3);
       T(2,1:3);
       T(3,1:3);];

R6_5 =  R5_0T*R6_0;

%Calculating theta6
theta6 = asind(-R6_5(1,2));

end
```

Sub function 3

```
function time = timecalc(line,joint)


global tt;

syms a1 a2 a3 a4 a5 v T dj ji
eqn = a1+(2*a2*T)+(3*a3*(T^2))+(4*a4*(T^3))+(5*a5*(T^4))-v == 0;
solt = solve(eqn,T);

initial_time=0;
final_time=2;
timestep=.05;
x=initial_time:timestep:final_time;%time 0 to 20 seconds

t = size(x,2);

if joint == 1
    v_max = 250;
else
    if joint == 2
        v_max = 187;
    else
        if joint == 3
            v_max = 250;
        else
            if joint == 4
                v_max = 300;
            else
                if joint == 5
                    v_max = 300;
                else
                    if joint == 6
                        v_max = 300;
                    end
                end
            end
        end
    end
end


if line == 1

    jointi=[51.84,12.298,-119.428,-103.03,-53.81,111.40];%home

jointf=[t_t(line,1),t_t(line,2),t_t(line,3),t_t(line,4),t_t(line,5),
t_t(line,6)];
    else
    jointi=[t_t(line-1,1),t_t(line-1,2),t_t(line-1,3),t_t(line-
1,4),t_t(line-1,5),t_t(line-1,6)];
```

```
jointf=[t_t(line,1),t_t(line,2),t_t(line,3),t_t(line,4),t_t(line,5),
t_t(line,6)];
end

ji = jointi(joint);
jf = jointf(joint);
dj = jf - ji;
j_vi=0; % Initial Angular velocity is assumed as zero
j_vf=0; % Final Angular velocity is assumed as zero
j_ai=0;% Initial acceleration is assumed as zero
j_af=0; % Final acceleration is assumed as zero


%Coefficients calculation

a0=ji;
a1=j_vi;
a2=j_ai/2;
a3=[20*(dj)-(8*j_vf+12*j_vi)*T-(3*j_af-j_ai)*T.^2]/(2*T.^3);
a4=[30*(-dj)+(14*j_vf+16*j_vi)*T+(3*j_af-2*j_ai)*T.^2]/(2*T.^4);
a5=[12*(dj)-6*(j_vf+j_vi)*T-(j_af-j_ai)*T.^2]/(2*T.^5);
Coefficients_J1=[a0 a1 a2 a3 a4 a5]

vmaximum = (15*dj)/(8*T);

t_max = (15*dj)/(v_max*8);
time = abs(t_max);
end
```

sub function 4

```
function [jtable vtable atable qtable] = trajectory(line,time)



global tt;
initial_time=0;
final_time=time;
timestep=.05;
x=initial_time:timestep:final_time;%time 0 to 20 seconds

t = size(x,2);

jtable(t,7)=0;%joint table
vtable(t,7)=0;%velociy table
atable(t,7)=0;%angular table
qtable(t,7)=0;%jerk table
jtable(:,1)=x;%table with time
vtable(:,1)=x;%table with time
atable(:,1)=x;%table with time
qtable(:,1)=x;%table with time
c = jtable(1:t,1);%time from 0 to 10seconds
t1 = size(c,1);

%joint parameters


    if line == 1
    %jointi=[0,-5.43,-22.0404,0,27.4438,0];%home position
[50,0,500,0,0,0]
    jointi=[51.84,12.298,-119.428,-103.03,-53.81,111.40];%home
position [230,140,215,0,-90,180]

jointf=[t_t(line,1),t_t(line,2),t_t(line,3),t_t(line,4),t_t(line,5),
t_t(line,6)];
    else
    jointi=[t_t(line-1,1),t_t(line-1,2),t_t(line-1,3),t_t(line-
1,4),t_t(line-1,5),t_t(line-1,6)];

jointf=[t_t(line,1),t_t(line,2),t_t(line,3),t_t(line,4),t_t(line,5),
t_t(line,6)];
    end
%Joint boundary conditions
for k=1:6

ji=jointi(k); %Initial position
jf=jointf(k); % Final position
j_vi=0; % Initial Angular velocity is assumed as zero
j_vf=0; % Final Angular velocity is assumed as zero
j_ai=0;% Initial acceleration is assumed as zero
j_af=0; % Final acceleration is assumed as zero

% Assume arm comes to the end position position with time
```

```
t_i=0;%initial time
t_f=x(1,t);%final time
T=t_f-t_i;%time difference

%Coefficients calculation
for i = 1:t

a0=ji;
a1=j_vi;
a2=j_ai/2;
a3=[20*(jf-ji)-(8*j_vf+12*j_vi)*T-(3*j_af-j_ai)*T.^2]/(2*T.^3);
a4=[30*(ji-jf)+(14*j_vf+16*j_vi)*T+(3*j_af-2*j_ai)*T.^2]/(2*T.^4);
a5=[12*(jf-ji)-6*(j_vf+j_vi)*T-(j_af-j_ai)*T.^2]/(2*T.^5);
Coefficients_J1=[a0 a1 a2 a3 a4 a5]

jtable(i,k+1)=
a0+(a1*x(1,i))+(a2*(x(1,i))^2)+(a3*(x(1,i))^3)+(a4*(x(1,i))^4)+(a5*(
x(1,i))^5);
vtable(i,k+1)=
a1+(2*a2*x(1,i))+(3*a3*(x(1,i)^2))+(4*a4*(x(1,i)^3))+(5*a5*(x(1,i)^4
));
atable(i,k+1)=
(2*a2)+(6*a3*x(1,i))+(12*a4*(x(1,i)^2))+(20*a5*(x(1,i)^3));
qtable(i,k+1)= 6*a3+(24*a4*x(1,i))+(60*a5*(x(1,i)^2))

end

end
```

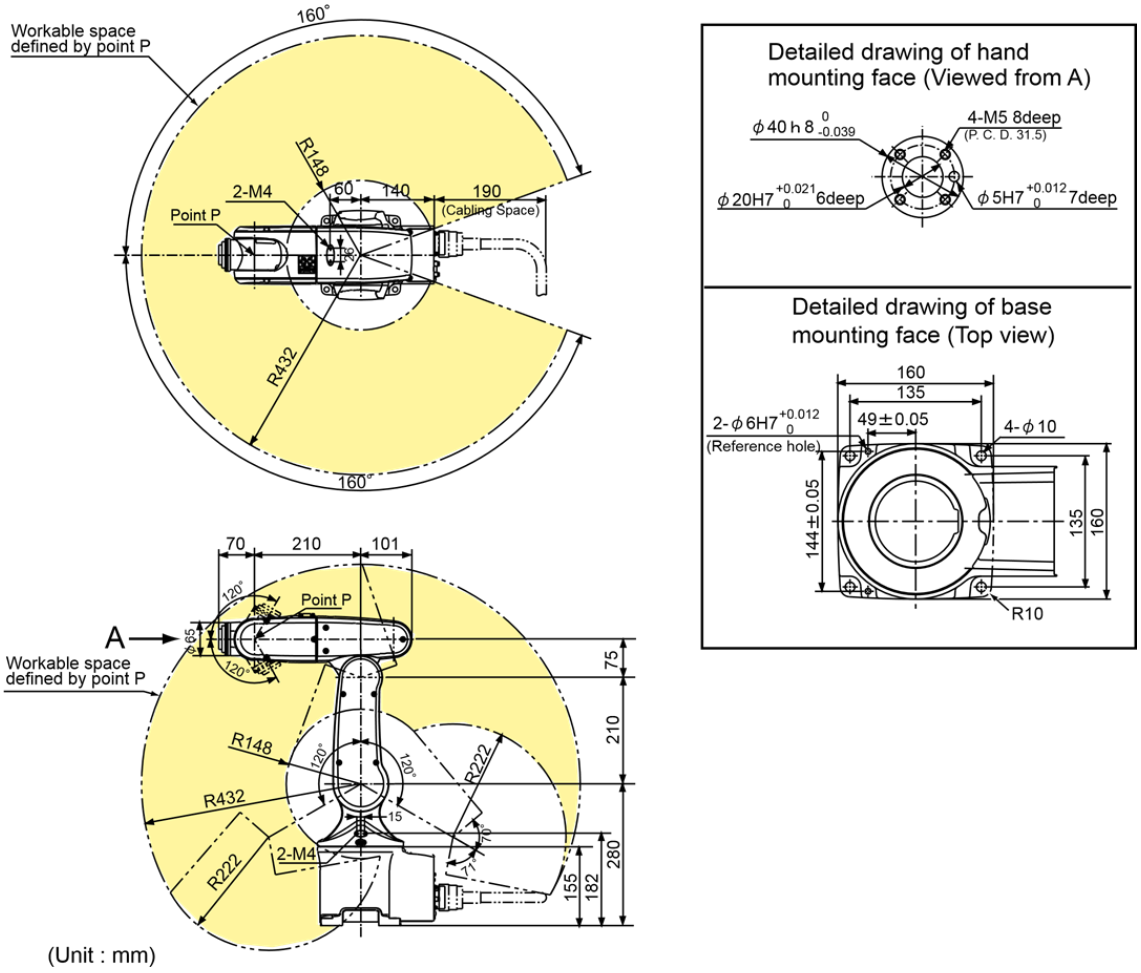
Appendix B

Dimensions and the configuration of the Denso VP 6242

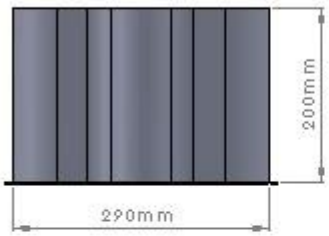
Machine head cover dimensions

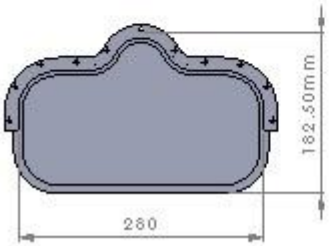
Robotic cell layout

Dimensions and the configuration of the Denso VP 6242



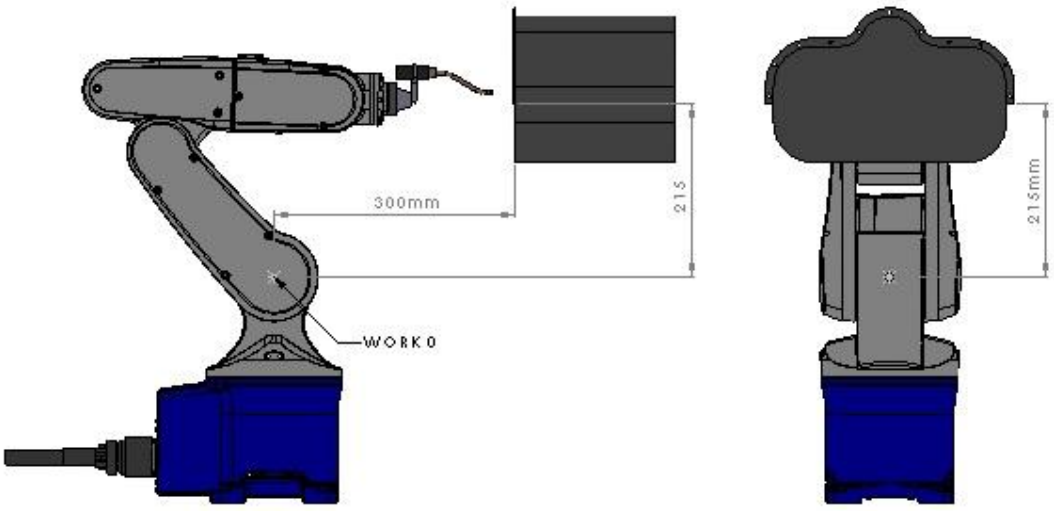
MACHINE HEAD COVER





ALL DIMENSIONS ARE IN MILLIMETERS

ROBOTIC CELL LAYOUT



ALL DIMENSIONS ARE IN MILLIMETERS

Appendix C

Scenario 1 joint angles calculation results

Line No:	Joint					
	J1	J2	J3	J4	J5	J6
1	-54.462	-14.039	123.366	103.300	56.737	66.684
2	-54.429	-14.039	121.631	102.197	56.323	68.703
3	-54.453	-13.914	119.798	101.065	56.002	70.724
4	-54.470	-13.745	117.926	99.923	55.707	72.751
5	-54.407	-13.613	116.081	98.784	55.368	74.789
6	-54.190	-13.603	114.346	97.660	54.911	76.832
7	-53.764	-13.775	112.812	96.567	54.283	78.845
8	-53.134	-14.117	111.504	95.507	53.489	80.795
9	-52.304	-14.614	110.438	94.484	52.533	82.655
10	-51.275	-15.257	109.635	93.502	51.409	84.396
11	-50.038	-16.041	109.120	92.577	50.108	85.986
12	-48.589	-16.964	108.933	91.736	48.619	87.375
13	-46.931	-18.016	109.105	91.018	46.940	88.510
14	-45.093	-19.149	109.564	90.413	45.094	89.414
15	-43.105	-20.308	110.193	89.878	43.105	90.168
16	-40.972	-21.454	110.890	89.351	40.975	90.860
17	-38.691	-22.560	111.579	88.775	38.702	91.570
18	-36.255	-23.612	112.226	88.110	36.278	92.344
19	-33.657	-24.596	112.800	87.305	33.700	93.238
20	-30.893	-25.493	113.259	86.273	30.966	94.345
21	-27.979	-26.248	113.465	84.778	28.106	95.916
22	-25.007	-26.782	113.242	82.459	25.240	98.326
23	-22.142	-27.031	112.488	78.986	22.580	101.903
24	-19.588	-26.979	111.218	74.246	20.385	106.749
25	-17.526	-26.639	109.486	68.480	18.887	112.624
26	-15.656	-26.161	107.570	61.941	17.806	119.242
27	-13.467	-25.696	105.746	54.188	16.689	126.990
28	-10.733	-25.320	104.201	44.505	15.408	136.543
29	-7.465	-25.062	103.059	32.213	14.106	148.572
30	-3.822	-24.912	102.360	17.088	13.112	163.333
31	0.016	-24.867	102.136	-0.073	12.731	180.071
32	3.851	-24.919	102.379	-17.226	13.109	196.803
33	7.499	-25.064	103.068	-32.348	14.118	211.559
34	10.786	-25.313	104.189	-44.640	15.447	223.586
35	13.488	-25.696	105.752	-54.246	16.702	233.067
36	15.532	-26.204	107.656	-61.860	17.678	240.693
37	17.442	-26.672	109.553	-68.471	18.797	247.378
38	19.650	-26.963	111.196	-74.283	20.447	253.284
39	22.213	-27.010	112.458	-79.003	22.651	258.109

40	25.062	-26.767	113.229	-82.482	25.295	261.696
41	28.012	-26.257	113.515	-84.862	28.135	264.178
42	30.914	-25.515	113.350	-86.390	30.983	265.791
43	33.676	-24.606	112.851	-87.369	33.717	266.838
44	36.272	-23.601	112.206	-88.101	36.296	267.644
45	38.707	-22.545	111.549	-88.757	38.718	268.407
46	40.987	-21.444	110.877	-89.348	40.990	269.136
47	43.119	-20.294	110.170	-89.867	43.119	269.817
48	45.108	-19.125	109.505	-90.379	45.110	270.536
49	46.944	-17.993	109.045	-90.983	46.953	271.440
50	48.590	-16.962	108.926	-91.732	48.620	272.619
51	50.031	-16.051	109.141	-92.587	50.101	274.028
52	51.270	-15.264	109.650	-93.510	51.404	275.615
53	52.306	-14.614	110.441	-94.486	52.534	277.349
54	53.135	-14.116	111.505	-95.509	53.491	279.208
55	53.763	-13.777	112.817	-96.569	54.283	281.159
56	54.191	-13.602	114.349	-97.662	54.913	283.172
57	54.417	-13.602	116.073	-98.783	55.378	285.214
58	54.474	-13.740	117.923	-99.923	55.712	287.251
59	54.447	-13.922	119.805	-101.067	55.995	289.276
60	54.415	-14.055	121.645	-102.202	56.310	291.297
61	54.462	-14.039	123.366	-103.300	56.737	293.316

Appendix D

Scenario 3 joint angles calculation results

Line No:	Joint					
	J1	J2	J3	J4	J5	J6
1	-54.462	-14.039	123.366	103.300	56.737	66.684
2	-54.462	-13.962	120.765	101.667	56.191	69.640
3	-54.462	-13.766	118.067	100.006	55.721	72.607
4	-54.352	-13.581	115.373	98.339	55.215	75.591
5	-53.836	-13.742	113.023	96.723	54.382	78.557
6	-52.891	-14.259	111.134	95.175	53.202	81.402
7	-51.510	-15.112	109.794	93.714	51.662	84.026
8	-49.688	-16.271	109.066	92.369	49.746	86.336
9	-47.431	-17.691	108.984	91.188	47.445	88.244
10	-44.772	-19.312	109.552	90.242	44.772	89.659
11	-41.773	-21.034	110.626	89.543	41.775	90.612
12	-38.464	-22.665	111.647	88.719	38.475	91.636
13	-34.820	-24.170	112.549	87.671	34.853	92.837
14	-30.824	-25.532	113.329	86.315	30.895	94.293
15	-28.701	-26.103	113.519	85.293	28.807	95.368
16	-26.541	-26.546	113.443	83.815	26.709	96.917
17	-24.396	-26.849	113.100	81.797	24.665	99.013
18	-22.323	-27.005	112.498	79.166	22.751	101.724
19	-20.382	-27.015	111.650	75.873	21.047	105.092
20	-18.630	-26.882	110.572	71.941	19.634	109.096
21	-17.116	-26.616	109.288	67.500	18.576	113.605
22	-15.867	-26.233	107.829	62.789	17.904	118.384
23	-14.418	-25.864	106.427	57.472	17.176	123.725
24	-12.628	-25.553	105.178	51.208	16.290	129.943
25	-10.523	-25.302	104.116	43.756	15.311	137.278
26	-8.149	-25.108	103.260	34.895	14.347	145.952
27	-5.560	-24.971	102.633	24.493	13.515	156.108
28	-2.819	-24.889	102.251	12.682	12.944	167.630
29	0.000	-24.862	102.123	0.000	12.739	180.000
30	2.818	-24.889	102.251	-12.681	12.944	192.369

31	5.560	-24.971	102.633	-24.492	13.514	203.891
32	8.149	-25.108	103.260	-34.895	14.347	214.048
33	10.522	-25.302	104.115	-43.755	15.311	222.722
34	12.627	-25.553	105.179	-51.207	16.289	230.056
35	14.418	-25.864	106.427	-57.471	17.177	236.275
36	15.867	-26.233	107.829	-62.789	17.904	241.616
37	17.118	-26.616	109.287	-67.501	18.578	246.397
38	18.630	-26.882	110.572	-71.941	19.634	250.905
39	20.382	-27.015	111.650	-75.874	21.047	254.909
40	22.324	-27.005	112.498	-79.166	22.751	258.276
41	24.396	-26.849	113.100	-81.798	24.665	260.987
42	26.541	-26.546	113.443	-83.815	26.709	263.083
43	28.701	-26.103	113.520	-85.293	28.807	264.632
44	30.824	-25.532	113.329	-86.315	30.895	265.707
45	34.821	-24.170	112.549	-87.671	34.853	267.163
46	38.464	-22.665	111.647	-88.719	38.475	268.365
47	41.773	-21.034	110.626	-89.543	41.775	269.388
48	44.772	-19.312	109.552	-90.242	44.773	270.341
49	47.432	-17.690	108.984	-91.188	47.445	271.756
50	49.688	-16.271	109.066	-92.369	49.746	273.664
51	51.510	-15.112	109.795	-93.714	51.662	275.974
52	52.891	-14.259	111.135	-95.175	53.202	278.598
53	53.836	-13.742	113.023	-96.723	54.382	281.443
54	54.352	-13.581	115.373	-98.339	55.215	284.409
55	54.462	-13.766	118.067	-100.006	55.721	287.394
56	54.462	-13.962	120.765	-101.667	56.191	290.360
57	54.462	-14.039	123.366	-103.300	56.737	293.316

Appendix E

Scenario 4 joint angles calculation results

Line No:	Joint					
	J1	J2	J3	J4	J5	J6
1	-54.462	-14.039	123.366	103.300	56.737	66.684
2	-54.462	-13.962	120.765	101.667	56.191	69.640
3	-54.462	-13.766	118.067	100.006	55.721	72.606
4	-54.352	-13.581	115.373	98.339	55.215	75.591
5	-53.836	-13.742	113.023	96.723	54.382	78.557
6	-52.891	-14.259	111.134	95.175	53.202	81.402
7	-51.510	-15.112	109.795	93.714	51.662	84.026
8	-49.688	-16.271	109.066	92.369	49.746	86.336
9	-47.431	-17.691	108.984	91.188	47.444	88.244
10	-44.772	-19.312	109.552	90.242	44.773	89.659
11	-41.774	-21.034	110.626	89.543	41.775	90.612
12	-38.464	-22.665	111.647	88.720	38.475	91.635
13	-34.821	-24.170	112.549	87.671	34.854	92.837
14	-31.197	-25.419	113.275	86.464	31.264	94.134
15	-27.754	-26.315	113.521	84.707	27.884	95.983
16	-25.135	-26.760	113.249	82.564	25.363	98.219
17	-22.947	-26.973	112.709	80.041	23.319	100.825
18	-20.991	-27.028	111.950	77.010	21.570	103.931
19	-19.202	-26.943	110.962	73.344	20.078	107.669
20	-17.623	-26.727	109.770	69.127	18.906	111.952
21	-16.326	-26.382	108.382	64.585	18.133	116.564
22	-15.025	-26.001	106.958	59.649	17.483	121.546
23	-13.397	-25.672	105.666	53.889	16.667	127.289
24	-11.451	-25.401	104.542	47.078	15.731	134.014
25	-9.226	-25.186	103.610	38.989	14.764	141.948
26	-6.769	-25.027	102.891	29.447	13.871	151.273
27	-4.135	-24.921	102.401	18.442	13.174	162.011
28	-1.391	-24.869	102.154	6.295	12.789	173.860
29	1.390	-24.869	102.154	-6.293	12.789	186.138
30	4.134	-24.921	102.401	-18.441	13.174	197.987

31	6.768	-25.027	102.891	-29.446	13.871	208.725
32	9.226	-25.186	103.610	-38.988	14.763	218.051
33	11.451	-25.401	104.543	-47.077	15.731	225.985
34	13.397	-25.672	105.666	-53.889	16.667	232.711
35	15.024	-26.001	106.959	-59.647	17.482	238.452
36	16.330	-26.382	108.380	-64.588	18.136	243.438
37	17.623	-26.727	109.770	-69.127	18.906	248.048
38	19.202	-26.943	110.962	-73.344	20.078	252.331
39	20.991	-27.028	111.950	-77.010	21.570	256.069
40	22.947	-26.973	112.709	-80.041	23.319	259.175
41	25.133	-26.760	113.249	-82.562	25.362	261.779
42	27.752	-26.315	113.520	-84.705	27.882	264.014
43	31.197	-25.418	113.272	-86.460	31.263	265.861
44	34.821	-24.170	112.549	-87.671	34.854	267.163
45	38.464	-22.665	111.647	-88.720	38.475	268.365
46	41.774	-21.034	110.626	-89.543	41.775	269.388
47	44.772	-19.312	109.552	-90.242	44.773	270.341
48	47.432	-17.690	108.984	-91.188	47.445	271.756
49	49.688	-16.271	109.066	-92.370	49.746	273.664
50	51.510	-15.112	109.795	-93.714	51.662	275.974
51	52.891	-14.259	111.135	-95.175	53.202	278.598
52	53.836	-13.742	113.023	-96.723	54.382	281.443
53	54.352	-13.581	115.373	-98.339	55.215	284.409
54	54.462	-13.766	118.067	-100.006	55.721	287.394
55	54.462	-13.962	120.765	-101.667	56.191	290.360
56	54.462	-14.039	123.366	-103.300	56.737	293.316