AN INNOVATIVE METHOD FOR IMPROVEMENT OF ROBOTIC SIMULATION CYCLE TIME ACCURACY

NENAD APOSTOLOVIC

AN INNOVATIVE METHOD FOR IMPROVEMENT OF ROBOTIC SIMULATION CYCLE TIME ACCURACY

by

© Nenad Apostolovic, B.A.Sc.

A thesis submitted to the

School of Graduate Studies

In partial fulfillment of the

Requirements for the degree of

Master of Engineering

Faculty of Engineering and Applied Science
Memorial University of Newfoundland

January 2003

St. John's, Newfoundland, Canada

Your file Votre référence
ISBN: 0-612-89686-2
Our file Notre référence
ISBN: 0-612-89686-2

Abstract

Robotic simulations can be classified into two groups – ones provided by the robot manufacturers and the "generic" ones provided by the simulation software companies. Both types have advantages and disadvantages with respect to cost, accuracy, functionality and integration with other virtual manufacturing tools.

Improvement of motion accuracy is one area of significant development of "generic" robotic simulations. Upon completion of the RRS I ("Realistic Robotic Simulation") project, it finally became possible to use original motion and kinematics algorithms, which minimized differences between the simulated motion and real motion.

However, RRS I Specification has several serious drawbacks. Not many robot manufacturers provide modules, functionality is limited and the price is high.

Presented material proposes a new method for improvement of simulation motion time accuracy. The method is based on the assumption about the existence of factors, whose influence on motion time of the real robot can be identified and incorporated through correction factors into the simulation motion model.

Acknowledgments

First of all, I want to thank to my supervisors Dr. Ray Gosine and Prof. Andy Fisher for guidance and encouragement during my studies. A special thank you to Prof. Andy Fisher and Dr. Mahmoud Haddara, Associated Dean of Graduate Studies of Engineering for approving internship with Flow Software Technologies.

I would like to thank Mr. David Fortin, the manager of Flow Software Technologies, for providing me both with the opportunity to enter the challenging area of industrial robotic simulation and for helping me with the resources needed for the research. A thank you as well to my team leader, Mr. Aleksandar Boskovic for many hours of discussion about robotic simulation and for being a role model for my academic and professional development. Many thanks to Jonathan Heron for all the help provided when it was necessary. Many thanks to all other employees of Flow Software Technologies and CIS Robotics, who influenced my work on the project. The quality of my work will be my gratitude.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1. Simulation

The complexity of many present-day systems, such as transportation systems, manufacturing systems, military systems is so high that successful design and implementation would be impossible without a tool such as simulation. Used in all stages of product/system development, simulation provides invaluable answers to many critical questions about the system.

A key benefit of using simulation is that a model of the real system, rather than the system itself is tested. In other words, a system can be designed and tested before it is built. The benefit of this approach is that the simulated system can be built "right the first time" with minimum (if any) problems.

At the same time, using the model of a system rather than the system itself represents the main weakness of simulation. For example, data used for building a model may be invalid, results of the analysis might be hard to understand or a proposed solution might not be the right one [1].

Yet, with the advantages and disadvantages mentioned, simulation has proven to be a valuable tool for design, analysis and control of complex present-day systems.

1.2. Robotic Simulation

The basic purpose of a robotic simulation is to provide a simulation of the actions performed by one or more robots and their interaction with other equipment in the work cell. Robotic simulation is one of the fundamental components of "virtual manufacturing", which itself represents a foundation for a new approach in the process of product design and manufacturing – concurrent engineering.

Typical applications of a robotic simulation include [2]:

1) Presentation purposes – a model of a work cell can be easily created for the purposes of concept verification and concept or marketing presentation,
2) Engineering purposes – design and verification of a work cell layout, verification of kinematic reach, path verification, singularity check, collision detection,
3) Offline programming purposes – involves the creation of a robot program using robotic simulation, verification of a created program, and its subsequent export into a proper format accepted by the targeted robot controller,
4) Process analysis and ergonomics analysis – includes throughput analysis using discrete-event simulation resulting in estimate of work cell capacity and cycle time.

Robotic simulation, just like any other simulation, uses a model of a system for testing possible scenarios rather than the system itself. Regardless of whether the analysis is performed on models of existing or non-existing systems, savings in time and material achieved by using simulation are significant (more details provided in Chapter 2).

1.3. Realism of Robot Simulation

The fundamental problems associated with simulation that were mentioned in section 1.1 can also be applied to robotic simulation. The following set of errors has been identified as important with respect to robotic simulation ([3, 4, 5, 6]):

1) Geometric errors – this type of error is based on the differences between an ideal CAD model and the real world model,
2) Dynamic errors – represent differences in motion behavior between the simulated and the real robot. Errors are a result of forces and torques not taken into account, which can significantly influence motion,
3) Thermal errors - thermal expansion due to factors such as friction in joints or temperature of environment. Typically, this type of error is not taken into account,
4) System errors – such as gear backlash and poorly tuned servos,
5) Motion behavior errors – occur due to the difference between the original robot controller motion algorithms and algorithms provided by the robotic simulation companies. This type of error results in incorrect cycle times and differences between the simulated and the real-world trajectory shapes.

Geometric errors, as well as system errors can be minimized relatively easy through the process of robot calibration. On the other hand, dynamic errors are very hard to detect. Although compensation of these errors is possible, establishment of a correct model is difficult and computationally expensive [4].

1.4. Motion Accuracy

Robotic simulations can be classified into two groups – "generic" robot simulations and robot simulations provided by the robot manufacturers. Both types of robotic simulations have their advantages and disadvantages. The simulations provided by the robot manufacturer have a high level of motion accuracy, because the simulation contains the same motion and path planning algorithms as the real robot. The key disadvantage is that typically there is no support for robots from other manufactures, as well as poor integration with other simulation tools.

On the other hand, generic robotic simulators typically contain libraries of robots from different manufacturers [7]; however the same motion and kinematics algorithms are applied to every robot regardless of the robot manufacturer [8]. In other words, a simulated work cell and a real-world work cell could show quite different behaviors with respect to cycle times and actual motion trajectories [4].

One possible solution to the problem is to develop a simulation motion model that is based on dynamics. However, the problems associated with the dynamics-based motion model are numerous. A dynamics model is typically of high complexity, computationally expensive and requires that a range of new parameters to be known prior to the computation [9, 10].

Another solution to the problem is to use the original controller software in the simulation systems; however it was not always possible to do so [9, 11], since the controller related information was kept confidential by robot manufacturers.

1.4.1. RRS Specification

In order to solve the problem of motion accuracy, a consortium of companies from the automotive industry, simulation industry, and industrial robot manufacturers was created in 1991 [6, 8]. The purpose of this consortium was to find an optimum solution for the problem described in section 1.4. The solution was found in the mid 1990s in the form of the RRS I Specification (RRS stands for "realistic robot simulation"). The RRS I specification defines a standard interface so that "the original software for motion interpolation and transformation of real controllers could be integrated in simulation systems in a standardized manner" [6]. In other words, an interface provides communication (figure 1.1.) between the simulation software and a module ("RCS module" – RCS stands for "realistic controller simulation"), which contains original motion and kinematics algorithms [9].

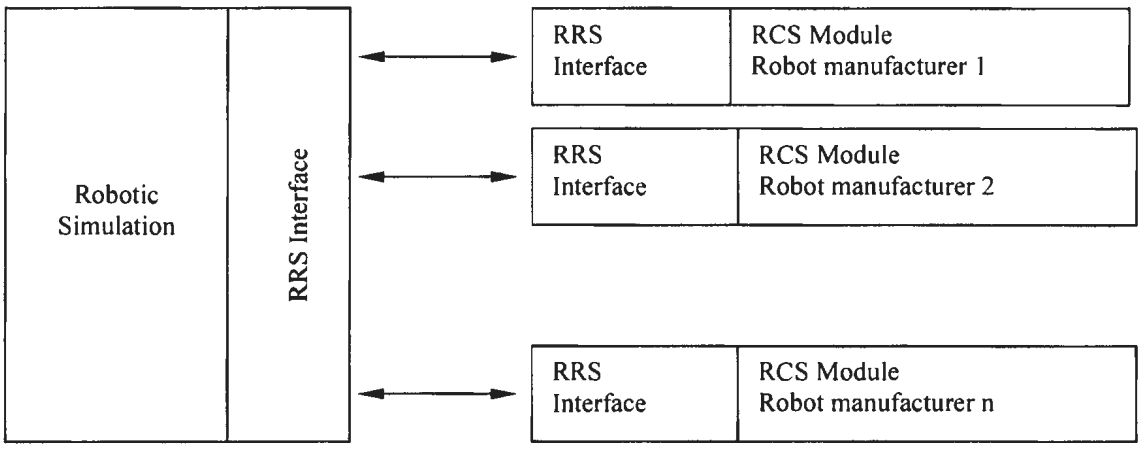


Figure 1.1 RRS Architecture

The RRS I Specification proved to be a success in several ways:

1) Targeted accuracy of cycle times was within ± 3% of actual cycle time, however, in some cases the achieved accuracy was within ± 1% [10],
2) Targeted positioning accuracy was 0.001 radians. Implemented RCS modules achieved accuracy of 0.00005 radians [10],
3) It became possible to have two or more robots of different manufacturers in a generic robot simulation, and to be sure that motion and kinematics algorithms were accurate.

The RRS I specification has limitations, such as no support for signaling, robot applications, interrupt handling, and motion coordination [8]. With these limitations in mind the consortium is currently working on a new specification. The purpose of the new specification (RRS II) is to overcome these limitations. RRS II introduces a new approach to the robotic simulation - robotic simulation consists of two separate entities, i.e. a simulator and a new type of RCS module. The purpose of a simulator is to provide a user interface and a graphical representation of the simulated work cell, while the RCS module represents the actual engine of the simulation. New RCS modules contain not only kinematics and motion algorithms, but they also contain input/output handling procedures, file system handling, interrupts, and motion synchronization [11]. In other words, the new RCS modules represent virtual robot controllers [12].

1.5. Contributions of the Thesis

Although the RRS I Specification proved to be a success, there are some serious drawbacks:

1) Although the RRS I Specification defines an interface that has 57 functions, several important issues mentioned in section 1.4 yet remain to be addressed by the new RRS II specification,
2) In order to conduct a more accurate simulation, the user needs to purchase an RCS module from the robot manufacturer, which can be a costly investment,
3) Not many robot manufacturers provide RCS modules,
4) RCS modules typically do not provide full functionality defined by the RRS I Specification.

Basically, only customers who really have a need for accurate simulation are the ones who will purchase the RCS module. A typical example is the automotive industry. Alternatively, for a small sized company that utilizes one or two robots, a highly accurate cycle time is of not primary concern. Yet, it is important to know approximately how much time certain operations might take.

The method presented in this thesis proposes a new approach to improve the cycle time accuracy of a robotic simulation without using the RCS modules and without using the dynamics motion model. This approach is based on the facts that:

1) Usage of motion and kinematics algorithms provided by the generic robotic simulation companies will rarely result in either an accurate estimate of simulation cycle time or in an accurate shape of a trajectory [9],
2) The original motion algorithms and kinematics algorithms will remain confidential,
3) Dynamics-based motion models are highly complex, computationally expensive, and still do not guarantee accurate simulation motion time.

A valid hypothesis that can be made about the motion time of the real robot is that there exists a set of factors that influence the motion time. Each of these factors affects the cycle time to a certain extent. The basic goal of this research is to verify the hypothesis about the existence of the influential factors and subsequently, to determine the level of their influence. If the hypothesis proves to be correct, then the next goal would be to find a way to integrate these influences into the existing simulation motion model so that simulation motion time accuracy is improved.

1.6. Thesis Summary

Chapter 2 provides an overview of simulation, as well as robotic simulation in particular. The overview includes a description of the role that robotic simulation has in the process of product design, types of robotic simulation and the benefits of using robotic simulation. Target areas of application such as marketing, engineering, offline programming, and process analysis are explained.

Chapter 3 provides a detailed description of the robotic simulation structure. Modules described include: CAD module, CAD data translators, built-in libraries for robots and supporting equipment, motion trajectory generator, kinematics module, and offline programming module.

Chapter 4 contains detailed mathematical descriptions of two analytical simulation motion models, one based on constant acceleration and the other one based on linear acceleration. For each stage/sub-stage of motion, a set of equations is derived for parameters such as travel time and travel distance.

Chapter 5 contains a description of the testing procedure for the two parameters that were tested, as well as the experimental results.

Chapter 6 includes conclusion and suggestions for future work.

Chapter 2

Robotic Simulation

2.1. Simulation

A simulation can be defined as "the imitation of the operation of a real-world process or system over time" [1], or as "the imitative representation of one system or process by means of functioning of another" [13]. It provides answers related to the performance of the existing system, evaluation of alternative solutions, and the quality of the design solution for a system that is to be built [1].

A typical simulation study is a process with several stages [1]:

1) Problem formulation – a clear understanding of the problem must exist,
2) Setting of objectives and overall project plan through an official proposal – the proposal has to define the objectives clearly, as well as stages of investigation and personnel that will be involved,
3) Model conceptualization – mathematical and logical representation of a real-world system,
4) Data collection – acquisition of real-world system data,
5) Model translation – creation of a computer model of a real-world system,
6) Verification – performance verification of the computer model,

7) Validation – validation of the model's accurate representation of a real-world system,
8) Experiment design – a decision on the number and duration of trial runs,
9) Production run and analysis
10) More runs – optional and based on the results from stage 9.
11) Documentation and reporting
12) Implementation – clients are introduced to the results of the simulation

It is apparent that that the most important aspect of any simulation study is the modeling of the real-world system, and this represents one of the greatest benefits of using a simulation. The model, rather than the real-world system itself, is used to prove the concept, to test new ideas and new features and to compare alternative solutions [14]. Finally, testing of a model saves both money and time. This is especially true in case of large and complex systems.

A simulation can be used both before and after the real-world system is built, however its application is most effective during the product's design stage. In the case of manufacturing, the use of simulation has become an integral part of concurrent engineering. Concurrent engineering represents a relatively new approach [15] in the product development. Unlike the traditional product development that is a sequential process with mutually isolated sequences, concurrent engineering is a parallel process, i.e. the product development activities are happening at the same time at several different levels, such as design, manufacturing, process analysis, etc. This "parallelism" provides early detection of problems associated to different aspects of the product, and in turn

provides a significant reduction of costs that would appear if the problems were detected at a later stage (figure 2.1.).

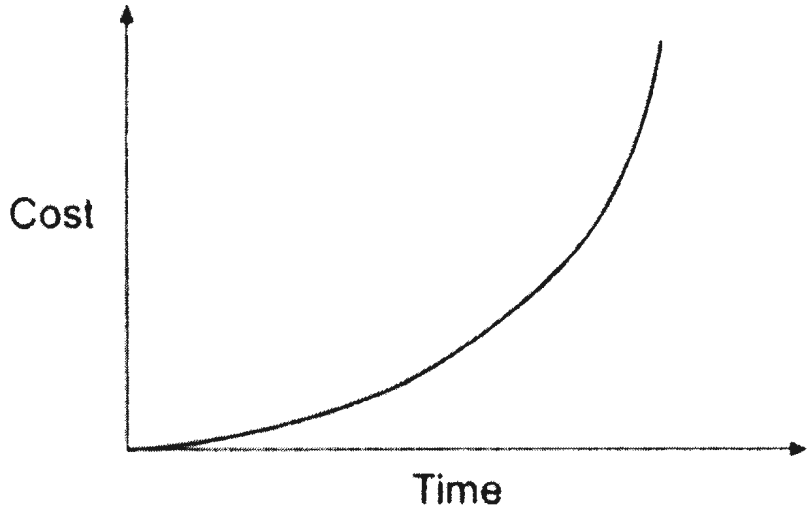


Figure 2.1 Cost-time Problem Detection Curve [15]

Additional benefits of using a simulation include faster time to market and no delays while waiting for the problems to be fixed because the potential problems are identified and resolved before the prototype is built [16].

The most important role in the process of product development belongs to the three-dimensional CAD solid model [15]. Not only does it serve as a communication tool between the members of the product development team, it is also used as an input for different simulation tools. Since the product development teams typically consist of engineers with different backgrounds [17], the tools they use for analysis are also different. While design engineers use finite element analysis to simulate mechanical

processes inside the part to determine stresses and strains, industrial engineers use material flow or process simulations to conduct the throughput analysis in order to determine the capacity of a cell, its utilization, etc. Similarly, quality engineers would use specialized simulation to verify and optimize inspection programs for CMMs and NC machine tools [18].

Simulation is also used after the deployment of a product or a system, as a tool for:

- Personnel training – in a situation where the real work is done in a dangerous environment, or when the actual facilities to be operated by the employees are too complex or too expensive to be reproduced. There is no production downtime and no costs associated with material used during the training [14, 19].
- Maintenance and support - simulation is used as an analysis tool to track down the possible bugs that might appear within the system. The bugs are typically reported by the customers and it is up to the system integrators to reproduce the state of the system that caused the bugs, and then to prevent its reappearance [14].

2.2. Development of Robotic Simulation

The development of robotic simulation began in mid 1980s when the first industrial robotic simulation provided by McAuto, a division of McDonell Douglas appeared on the market [20]. Deneb Robotics, Tecnomatix, and Silma Inc. followed with their robot simulations. Robotic simulations at the time provided only basic functionality, since they were based on CAD systems that used wire-frame and surface representation of objects in space. This limited functionality resulted in the limited role that robotic simulation had – it was used as verification tool rather than a process design tool [21].

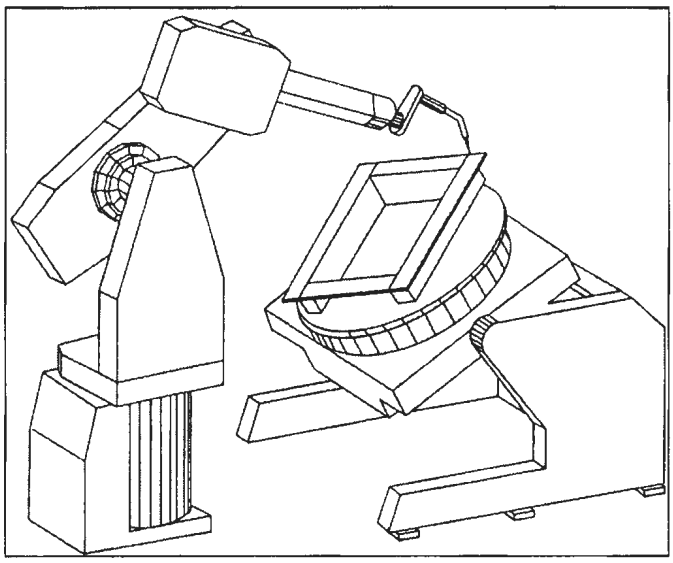


Figure 2.2 Robot Servicing a Work-piece – Surface Representation [22]

With the ever-increasing power of computers, the functionality of both CAD systems and robotic simulations also improved over time. Basic functionality that included verification of a robot's kinematic reach and work cell layout was expanded over time to incorporate collision detection, cycle time analysis, offline programming, and calibration [20].

Introduction of a solid model representation in CAD systems provided further development of robotic simulation. More complex tasks such as path planning, improved collision detection, and grasp planning are some of the features that are presently considered to be standard [7]. The realism of robotic simulations with respect to the applications utilized has also improved. Figure 2.3 shows two robots painting the body of a car. Different thicknesses of deposited paint are represented with different colors. Figure 2.4 shows two robots performing welding.

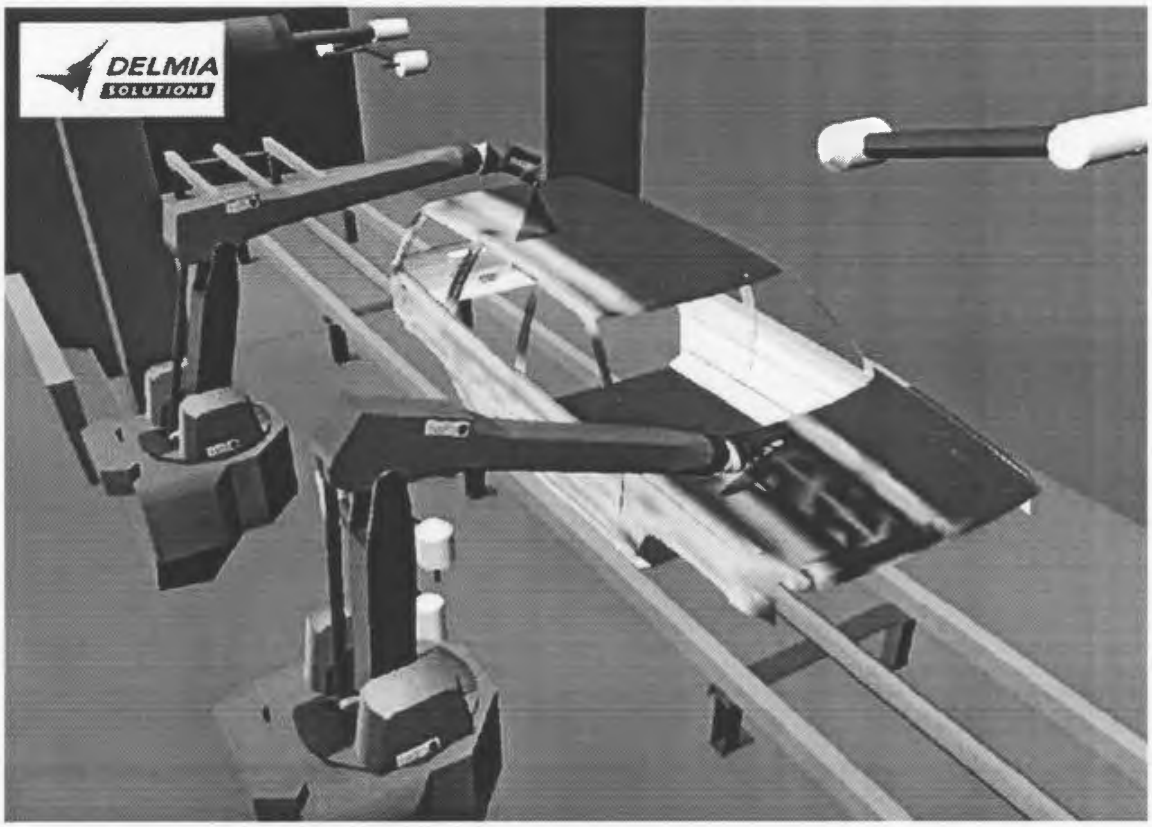


Figure 2.3 Car Body Painting Line [23]

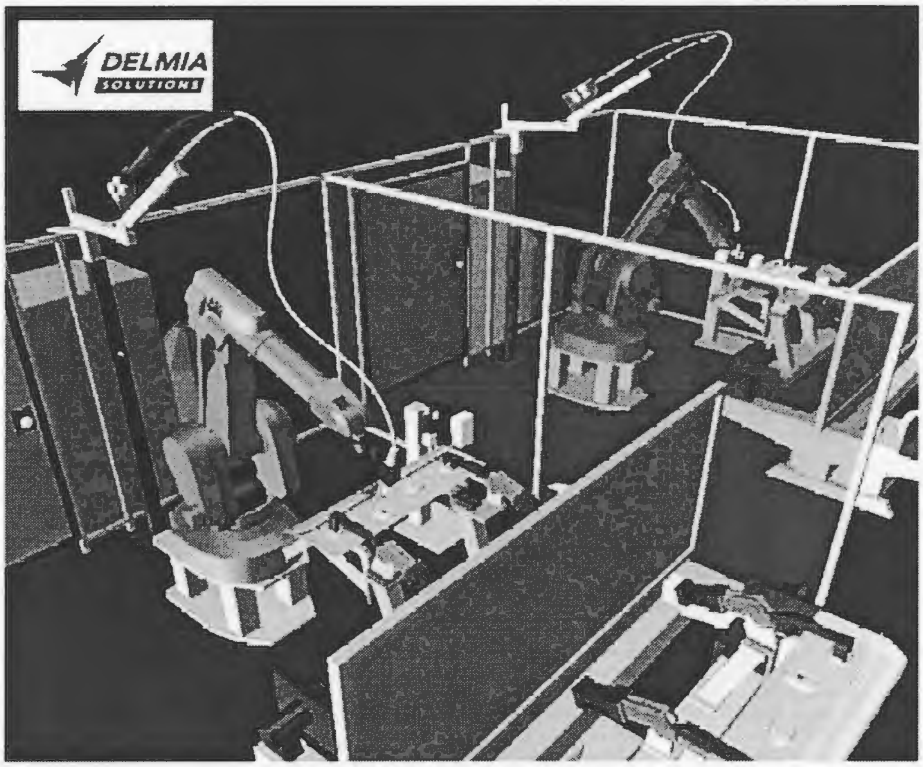


Figure 2.4 Arc Welding Application [23]

Robotic simulation is in a continuous process of development. Several trends can be noticed today. On the system level, the trend is integration of a robotic simulation into larger systems [18, 21, 23, 24]. Figure 2.5 provides the structure of a virtual manufacturing tool offered by Tecnomatix. In other words, robotic simulation today is considered as one of the many tools used in the process of product development.

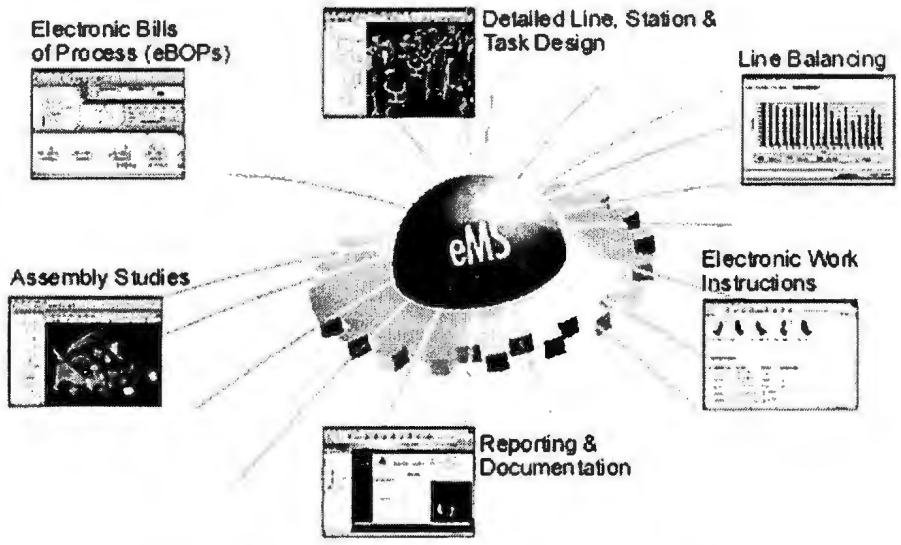


Figure 2.5 Integration of "Virtual Manufacturing" Tools [18]

The two largest manufacturers of robotic simulation, Delmia and Tecnomatix are currently introducing a whole range of new products, which include tools for process planning, Internet-based exchange of manufacturing information between plants and suppliers, quality inspection and tolerance management, analysis of ergonomics issues, and analysis of machining issues [18, 23]. The goal behind this integration is to firmly establish "virtual manufacturing" as a key link between product design and actual

production (figure 2.6). The reason behind this integration has already been mentioned – concurrent engineering. Both the design of a product and the design of the manufacturing process occur simultaneously, thus cutting the costs of production, improving the quality of the product, and getting the product to market faster [25].

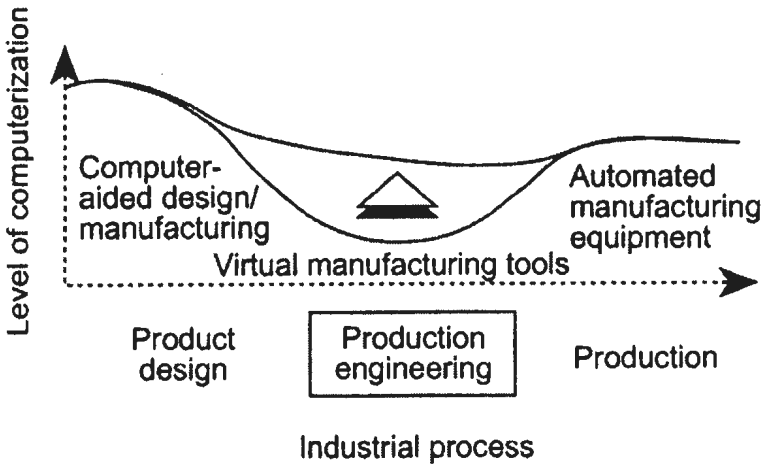


Figure 2.6 Virtual Manufacturing as a Key Link [25]

On the application level one of the major trends emerging is an improvement in robotic simulation accuracy. Details are provided in section 1.4.1.

2.2.1. Benefits of Using Robotic Simulation

The most important benefit of using a robotic simulation is the same benefit that applies to simulation in general – it is the model, a virtual model of a real-world system that is simulated, tested and modified rather than the real-world system itself. In case of a robotic simulation it means that issues related to work cell design, selection of robots and associated parameters are resolved before the actual physical model is built. The final result is that the real-system is built "right the first time".

Robot simulation can be used in one of the following contexts [2]:

- As a conceptual design and presentation tool,
- As an engineering analysis tool,
- As an offline programming tool, or
- As a process and ergonomics analysis tool.

The benefits of using a robotic simulation will be grouped and presented accordingly.

2.2.1.1. Robotic Simulation as a Conceptual Design and Presentation Tool

Robot simulation can be used as a tool that provides very effective visual presentations of concepts to customers, other engineers, or to management. A user can either create a new cell using the basic CAD functionality provided by the robot simulation, or simply import a work cell from a file created in an external CAD package [2].

Furthermore, robot simulations provide libraries of robots and supporting standard equipment [9], thus the initial design can be completed in a matter of days [14].

Another powerful feature of present-day robot simulations is the capability of saving the simulated actions occurring inside the cell to an animated media file [18, 23, 26, 27].

2.2.1.2. Robotic Simulation as Engineering Design Tool

Once the customers or the management accepts the presented concept, further development takes place. Typical activities for this stage involve detailed design of the work cell [2, 26]. Detailed design of the work cell includes the proper selection of one or more robots with respect to the task to be performed, overall dimensions of the work cell, kinematic reach of the robot, and maximum payload. Robotic simulation software typically provides libraries of robots, tools and other standard supporting equipment from different manufacturers, which significantly accelerates the design process. Design of tools and fixtures, selection of material handling systems, such as conveyors and AGVs is also a part of this stage.

The next stage includes creation of collision-free paths. The paths can be created either manually or automatically, by using a built-in utility for automatic path generation. Once the path is defined, cycle time analysis takes place. Cycle time can be broken down into the motion of the robot, motion and actions performed by other equipment in the cell, and the time spent waiting for a certain signal to change its value. When it comes to motion of the robot, typical parameters that can be set include speeds and accelerations

both for joint and linear/circular motion (figure 2.7). The time that a robot spends waiting after it gets into a target point as well as the corresponding tool and input/output actions can be specified.

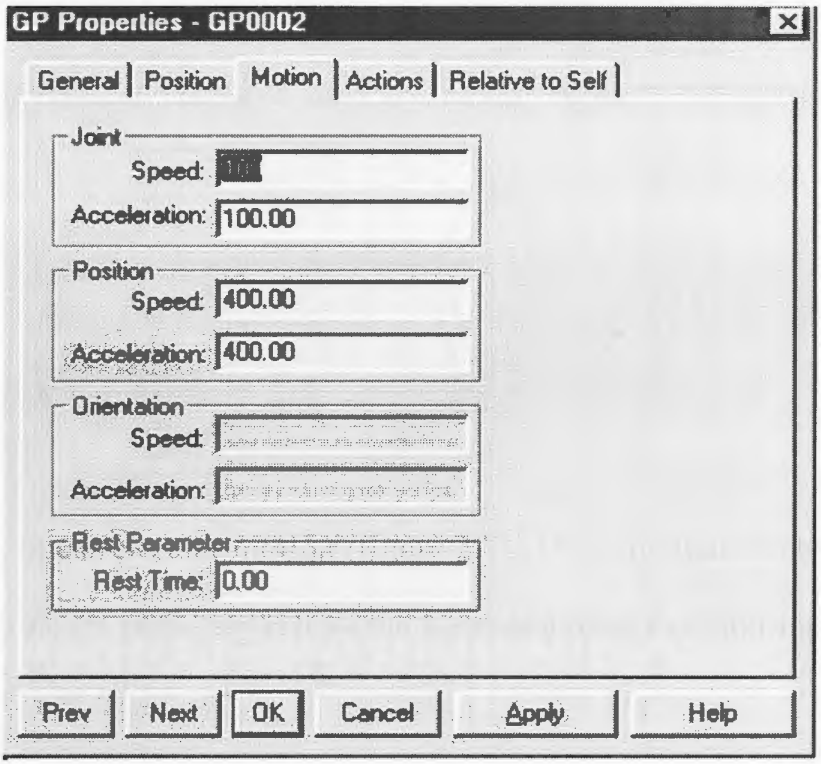


Figure 2.7 Kinematics Properties Form [28]

Benefits of using simulation as an engineering tool are numerous, especially in the case of complex products and systems [29]. Evaluation of design alternatives is performed in the virtual environment, which means:

- Problems are identified and resolved prior to the actual production,
- Overall cost reduction,
- Shorter time to market,
- Uniform quality of products.

Examples:

1) Nissan produces four types of vehicles on one production line in one of its plants in Japan [25]. The production facility was designed and verified offline by using ROBCAD (Tecnomatix). The same robot simulation is used for offline programming purposes – more than 1200 programs were created for the 117 robots employed. Furthermore, overall design time including time for design verification was reduced from five to only three months.

2) British Rover used ROBCAD in the design of its vehicle Rover 75 [30]. More than 750 modifications of the original model were made based on the offline verification. Potential savings in using simulation and getting the design "right first time" were estimated to approximately half a million pounds just for the bumper tooling alone.

3) Boeing used the robot simulation provided by Deneb Robotics (now Delmia, a part of Dassault Systemes) to verify structural design and assembly of the X-32 joint strike fighter [29]. It was all a part of a competition for a $750 billion dollar contract for the U.S. Department of Defense. Production costs were reduced by approximately 33%.

2.2.1.3. Robotic Simulation as an Offline Programming Tool

A robot performs tasks through a programmed sequence of motions or actions [31], which are stored in the memory of robot controller. Programming of robots can be done either online of offline. Online programming is typically done by a programmer, who uses a teach pendant to move the robot to different locations inside the robot's attainable workspace. A teach pendant is a hand-held device that is connected directly to the robot's controller and enables the programmer to create, modify or delete programs [31].

One of the major advantages of online programming is that it does not require a lot of skill [31]. A major drawback to online programming is that it must be done on site. In addition, online programming is typically a time intensive task and production must be halted during the programming time. According to the report [34], for a facility that has seven lines with 36 painting robots, overall downtime for online programming was estimated to over one year. Costs also included paint used in programming, as well as the vehicle prototypes. The conclusion is that online programming is effective only when the task is not too complex [3].

Offline programming is a method of creating robot programs without using a real robot [3]. Typically, offline programming is a three-step process that includes the building of a CAD model of the work cell, calibration, and program development [2].

A CAD model of the work cell can be created either in a robot simulation package, or in some other CAD software package and later imported into the robot simulation [2]. This CAD model is used both for simulation and offline programming [26].

Calibration is a required step in offline programming [2], since the CAD model exists in a virtual environment where all dimensions are ideal. In the real world, accuracy associated problems can influence offline programming to the point where modification of offline created programs on the shop floor is inevitable [9]. Typical problems associated with accuracy include different lengths of robot links, incorrect placement of the robot and other equipment in the cell, environment temperature, and payload [2]. The purpose of calibration is to identify the influences mentioned and to incorporate them into the mathematical model of the simulated robot, thus preventing modification of the program on the shop floor [2, 26].

Offline programming is relatively easy, and it can be done either manually or automatically - using the advanced features of a robotic simulation. Benefits are numerous. Probably the most important benefit is a significant reduction in production downtime, in some cases up to 90%. In additional there is a reduction in the time required for the creation of a robot program [32].

Additional benefits include:

- Correct tool orientation that depends on the type of application, yet assigned automatically by the simulation software [26],
- Automatic creation of teach points both on simple and complex parts [26],
- Offline verification of created programs – in case something unpredictable happens, such as collision, or change of robot configuration, it is happening in the virtual world, therefore there is no real damage done. Correction of a program can be done relatively quickly [26],
- Usage of one system for many robots [3], i.e. the same offline programming tool can output the same program into several different robot languages,
- Improved safety of a robot programmer who is not exposed to a harmful environment [3, 26].

Once the offline created programs are verified, they are transferred to the robot controller for the test run.

2.2.1.4. Robotic Simulation as Process and Ergonomics Analysis Tool

Robotic simulation integrated with the discrete event simulation can also be used for evaluation of the work cell performance [2, 26]. Typical analysis involves:

- Justification of the number of robots in the work cell,
- Recognition of the potential production bottlenecks,
- Estimate of the cycle time – best and worst scenarios.

Several software packages meant for workplace ergonomics analysis can be found on the market. Based on the existing CAD model of a manufacturing cell, or an assembly line, for example, the following analysis can be conducted [2, 16, 33, 34]:

- Estimate of a percentage of the general population that will work comfortably can be determined,
- Evaluation of the safety hazards,
- Evaluation of worker's lifting capacity and resulting strains,
- Potential reach to certain places inside the machine,
- Evaluation of the time designated for an operation.

2.3. Chapter Summary

This chapter provided a brief description of development of a robotic simulation, the role a robotic simulation plays in the process of product development and the benefits of using a robotic simulation. Four main application areas of robotic simulation were described – presentation and marketing, engineering design and analysis, process analysis and offline programming.

The next chapter will provide more insight into robotic simulation through description of its functional structure.

Chapter 3

Robotic Simulation – Functional Structure

3.1. Introduction

Robotic simulation represents a complex and large software product. The structure of a robotic simulation is not clearly defined by a standard. Robot simulation companies themselves define the type and function of modules. Common features of a few robot simulations can be identified and organized into modules [7]:

- CAD solid modeler,
- Built-in libraries of commercially available robots,
- Data translators,
- Kinematics module,
- Motion trajectory generator,
- Offline programming module(s),
- Calibration module,
- Open development interface.

3.2. CAD Solid Modeler

Every robot simulation typically provides a CAD solid modeler that has a limited functionality. The basic idea is to use a CAD package such as AutoCAD, Pro/Engineer, CATIA, or I-DEAS to design a work cell. Once created, a work cell is typically saved in a file, which is then uploaded into the robotic simulation. CAD solid modelers provide the following functionality [35]:

- Visual presentation of the work cell layout, which typically includes a robot or robots, machines, conveyors, fixtures, tables and jigs,
- Enables the user to create, modify or delete the model of the work cell,
- Enables the user to expand the existing libraries of robots and equipment, which accelerates subsequent cell designs,
- Visual representation of the motion that takes place in the cell,
- Represents input for automatic generation of a robot path.

3.3. Built-in Libraries

Although the user can create a robot and later save it in a file for future use, robot simulation companies and robot manufacturing companies typically provide robot model information in the form of CAD files [7]. This approach provides shorter work cell development time, since the user can access the robot model data using CAD libraries provided by the robot simulation company. Users can also download the robot CAD model from the robot manufacturer's website, and insert it directly into the robot

simulation, without any modifications. The direct result is a significant decrease in work cell development time.

The same applies to the tools, fixtures, jigs, and all the other equipment typically used in a robotic work cell. Furthermore, the user can create models of custom designed or custom made equipment, save it and re-use it when the need arises.

3.4. CAD Data Translators

The CAD model of a work cell does not necessarily have to be created using the CAD capabilities of the robotic simulation software. It can also be created by some other CAD software package and imported into the robot simulation. The process of CAD model importing can create numerous problems [36, 37, 38]. The key reason lies in the different file formats in which a CAD model can be saved and in errors that occur during the conversion process of one file format into another.

The CAD file format can be either a proprietary one or a neutral one such as IGES or STEP. The conversion is performed by the translators that transfer a CAD file from one file format to another. A conversion can be [38]:

1) Direct conversion of one proprietary file format into another. The main problem is that both file formats have to be known, and that can represent a serious problem since proprietary formats are kept confidential [38]. A conversion back into the original proprietary file format can occur, thus creating the need for another translator.

2) Indirect conversion – a neutral file format is used as an intermediate step in the conversion process. The problem with this approach is that conversion errors can happen both during the conversion from the original proprietary file format into a neutral file format, and from neutral file format into the targeted proprietary file format.
3) Spatial Corporation and Unigraphics Solutions Ltd. have undertaken a different approach to the problem. The two companies developed their own CAD modeling systems (ACIS and Parasolid), which are built-in in numerous applications [38]. One CAD modeling system implies one file format, which means that no file format conversion is necessary. Furthermore, one ACIS based file can be opened and used by any other ACIS based application without any problems [38].

Robotic simulation has to be able to import the CAD model in different file formats, as well as to save the CAD model to different file formats. In other words, robot simulation software has to be able to "import" and "export" a CAD model into different file formats. For this particular purpose, specialized CAD translators have to be provided.

The CAD model translation process is not always successful. Errors occur because of different mathematical representations of 3D objects [36], thus corrective interventions are necessary. For example, Workspace 5 robotic simulation, can import files both in IGES and SAT formats. It can also perform corrections, such as "healing" of IGES files. Healing is "the process of improving the accuracy of solid models so they can be used

more effectively…" [37]. Healing is a multi-step process, which includes clean up of a translated model, geometry simplification, stitching, etc.

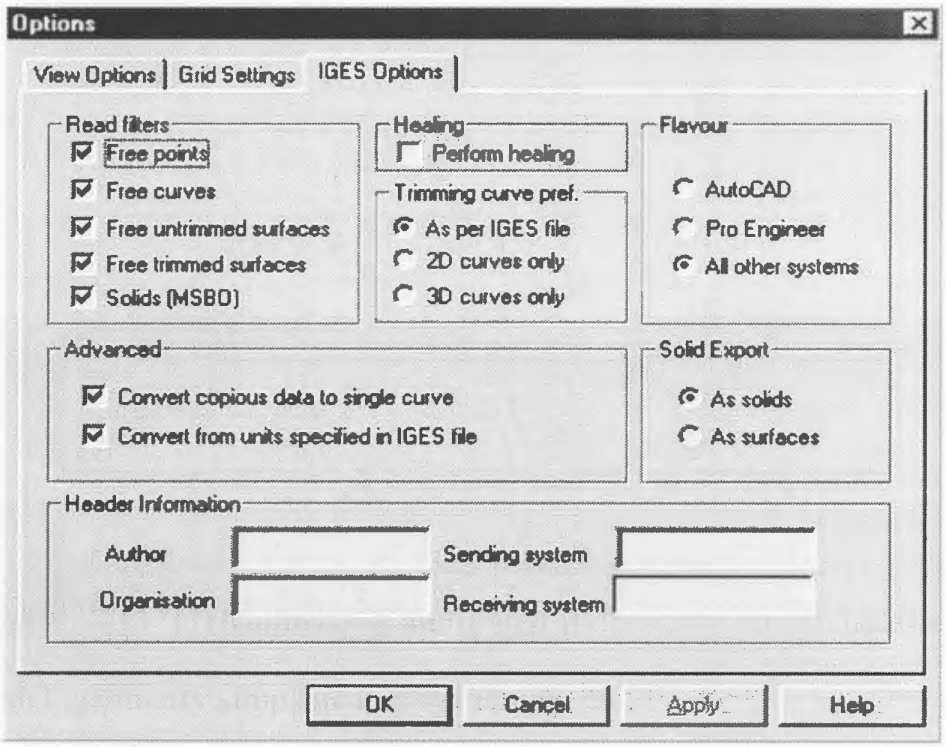


Figure 3.1 IGES Import Options [28]

3.5. Kinematics Module

An industrial robot is typically a serial link manipulator that has several joints, each of which can be either rotational or prismatic. By varying the types of joints different robot configurations can be built. Some of them are presented on the pictures below. The purpose of the kinematics module is to provide direct and inverse kinematics solutions for a range of different configurations of the robots shown on figure 3.2, figure 3.3 and figure 3.4.

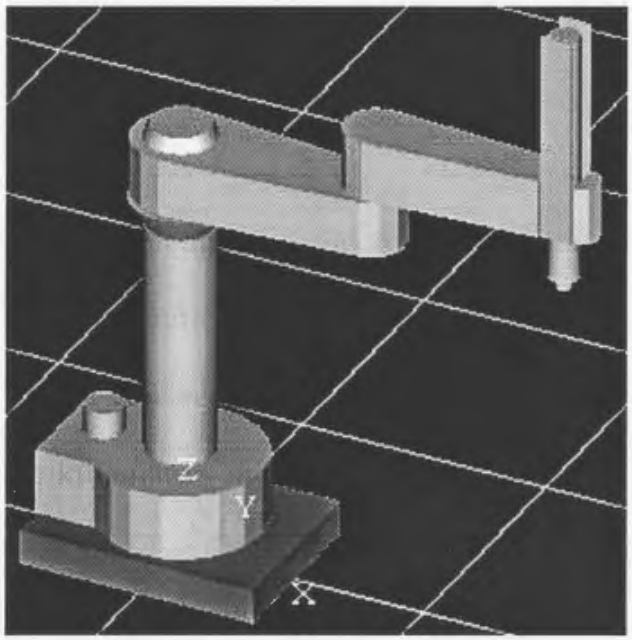


Figure 3.2 SCARA Robot Configuration [28]

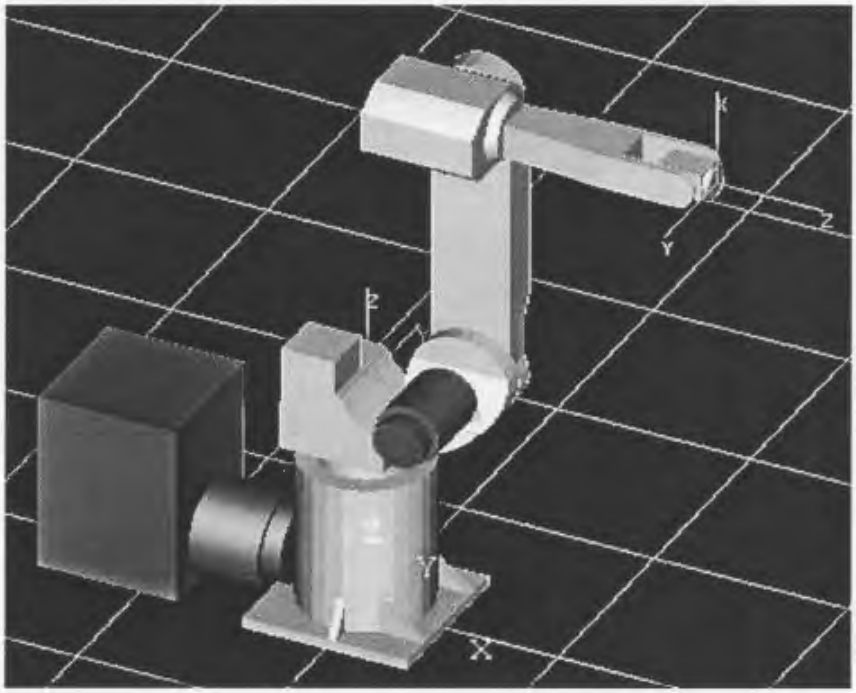


Figure 3.3 PUMA Robot Configuration [28]

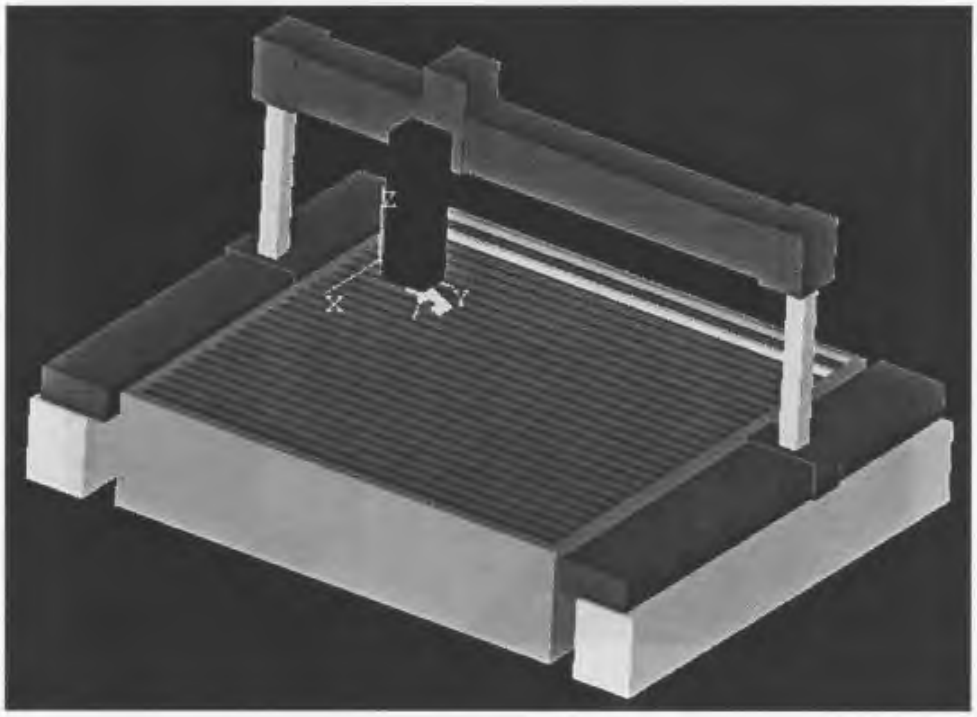


Figure 3.4 Gantry Robot Configuration [28]

Direct kinematics is typically based on the Denavit-Hartenberg convention, which represents a set of rules for establishing a geometric description of a serial link manipulator [39, 40, 41]. A set of transformation matrices is combined into one transformation matrix that represents the position and orientation transformation of the last link in the chain relative to the first link.

$$T_n^{ref} = A_1^{ref} A_2^1 A_3^2 ... A_n^{n-1} \tag{3.1}$$

Inverse kinematics represents a method of calculating the values of joint angles or distances based on the current Cartesian position and orientation of the last link. Finding solutions of the inverse kinematics problem is more complex than finding solutions for direct kinematics. Common methods for solving the inverse kinematics problem include algebraic method, geometric method and numerical methods.

The algebraic method uses equation 3.1 as a start point for inverse kinematics calculation. Derivation ultimately produces a system of twelve equations, out of which only six are independent, which means that there might be one solution, multiple solutions or no solutions at all [39]. For a manipulator that has six degrees of freedom, the number of multiple solutions can be up to 16 [40]. Ambiguity of solutions is avoided by using configuration strings [40].

Numerical methods can be applied to any kinematic structure. The problem though is that not all solutions can be computed [40].

3.6. Motion Trajectory Generator

A robot performs tasks by moving between programmed teach points. Motion control of the robot is based on the difference between the actual and the desired position. The larger the difference the larger the current sent to the servo-drives is. However, the current has to be limited in order to keep the servo-drives functional. The solution is to use large number of intermediate positions or interpolated points as an approximation of a continuous trajectory.

The points are typically supplied by the motion trajectory generator [4] or by a trajectory planner [41]. By feeding the servo-drives one interpolated point at a time, control over accelerations and velocities reached during motion between two points can be established.

Generally, robot motion is categorized in two major groups [41]:

- Joint based motion – teach points are typically defined in joint coordinates, and motion control is based on the difference between the joint angles/distances of start and end point. Joint-based motion, compared to the path-based motion is less computationally demanding [40], trajectory planning is simpler and can be done in almost real time [41]. Problems associated with joint-based motion are that the path is of an irregular shape and locations of manipulator links during motion are unknown, which represents a major disadvantage when it comes to obstacle avoidance [41].
- Path-based motion – teach points are expressed in Cartesian coordinates and the shape of the path is known. The path is typically a straight line or a circular arc, and the

TCP of the robot follows the path while moving from the start point to the end point. Path-based motion is computationally more expensive due to the fact that the location of the tool center point of the robot is expressed in Cartesian coordinates. In order to move the robot from the start point to the end point a conversion of the location expressed in Cartesian coordinates into joint coordinates is necessary.

Another important note about robot motion is that all the axes start and stop at the same time, no matter if the motion type is joint-based or path-based.

In the case of joint-based motion, motion trajectory is based on the difference in joint values between the actual and the desired position [4, 41]. Depending on the type of velocity profile (constant or linear acceleration), travel time for each joint is calculated. The reference travel time is equivalent to the time taken for the joint that takes the longest time to complete its motion. Travel times of all other joints are set to be equal to the reference travel time. Then, joint velocities and accelerations (in case of linear acceleration) are scaled so that all joints begin and complete motion at the same time.

In the case of path-based motion, the motion trajectory generator supplies servo-drives with a number of intermediate points that approximate the straight-line path or a circular-arc path. Although it has been established that the locations of interpolation points are based on the linear velocity, acceleration, and the type of the velocity profile, the exact method of generation of interpolation points has been kept confidential by the robot manufacturer companies [11]. In other words, only assumptions can be made how a particular robot controller generates interpolation points. Every interpolation point that

makes the path is expressed in Cartesian coordinates and an inverse kinematics calculation is required in order for joint angles/distances to be found, thus making path-based motion computationally more intense than the joint based motion [4].

3.7. Offline Programming

There are two methods to program an industrial robot – on-line programming and off-line programming. On-line programming is typically done through a hand-held device called a teach pendant, which is directly connected to the controller of the robot. A teach pendant enables a robot programmer to move the manipulator either in Cartesian or joint space, to memorize locations, define tool actions in those points, etc [3]. Online programming has an advantage in situations that involve simple tasks or in situations where the parts have a simple geometry. Disadvantages are numerous and include halted production so that robots can be programmed, long programming time, and scrap material generated during programming [3].

Offline programming represents an alternative to on-line robot programming. It enables the user to create robot programs without using a real robot. It is especially useful when task complexity is high, as well as when long production downtimes are not allowed, or when programming is to be done in a harmful environment [3].

The creation of robot paths is based on the geometric information of the part which action is to be performed upon [26]. A path can be created manually by selecting vertices or edges of the object and setting the values of relevant parameters such as joint

or linear velocity and acceleration, orientation speed and acceleration, wait time, and tool actions [26]. This type of programming is used for the creation of simple programs, which are typically represented as a sequential list of teach-points that the robot's TCP will acquire during motion [28].

Alternatively, a path can be created on a selected edge or surface automatically by internal algorithms. This automatic path generation uses internal analytical models of the process developed through experiments [7]. For every robotic application, such as spot welding, arc welding, or painting, an analytical model is developed [5] based on the parameters that are important for the process. Assignment of teach-point properties is also performed automatically.

Complex robot tasks require complex program logic, such as condition handlers, variables, subroutines, and interrupts [42]. This essentially means that a sequential list of teach-points does not satisfy the requirements of complex tasks. For this purpose robotic simulations typically provide a development environment as well as the simulation language, which contains a set of commands that provide access to the API of the simulation software. Workspace 5, for example uses Visual Basic for Applications as the development environment and custom developed Visual Basic commands to access the simulation API. Figure 3.5 shows some of the routines and data that can be accessed from within the Workspace 5 development environment.

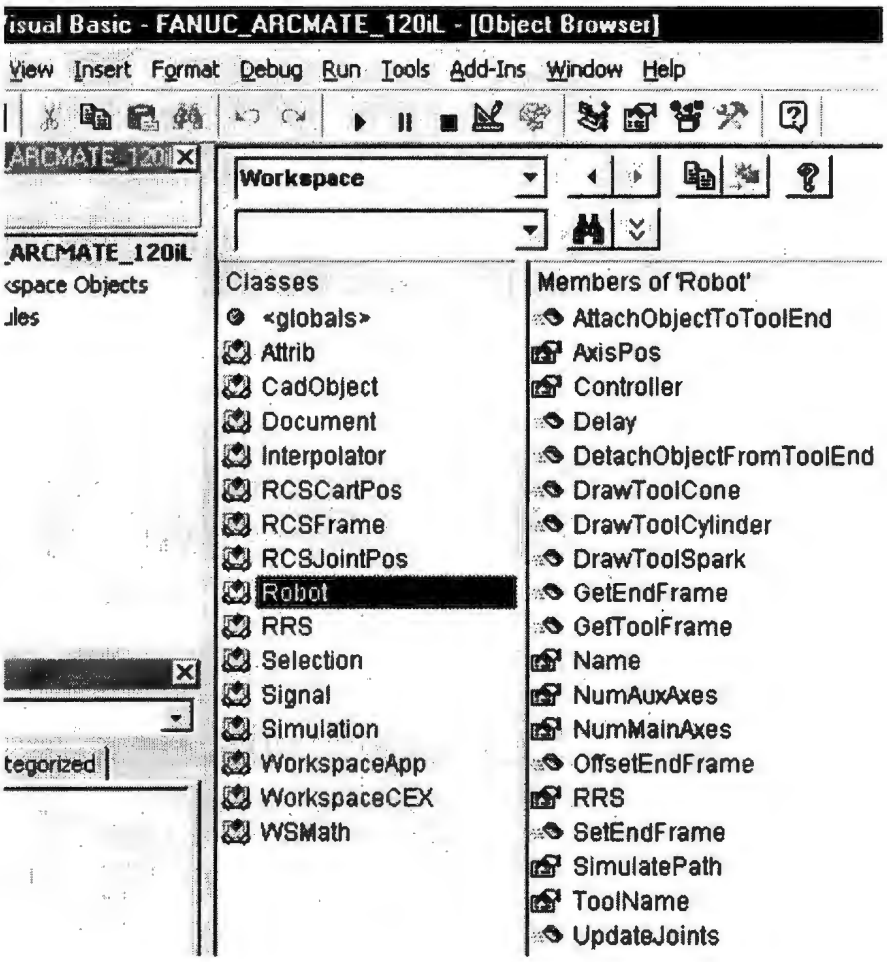


Figure 3.5 Robot Related Functions Available to the User in the VBA Environment [28]

Both simple and complex robot programs, once verified in the simulation can be converted into the native robot language format. In the case of a simple program represented by the sequential list of teach-points, conversion is quite straightforward. Parameters associated with each teach-point are read, formatted according to the syntax of the selected output language and then written into a file. Minor adjustments may be required, yet overall programming time is shorter than in case of online programming [10, 42].

Another important aspect of offline programming is the verification and modification of existing programs [26, 42]. For that purpose a set of translators or postprocessors [42] has to be developed for the robot languages supported by the robot simulation software. The purpose of the translators is to convert the program from the original robot language format into the simulation language format and back.

3.8. Calibration

The purpose of calibration is to eliminate the differences between the real world and the virtual CAD world in which all dimensions are ideal [7]. Calibration is of vital importance for the process of offline programming. It improves the accuracy of the robot and prevents its collision with other equipment in the work cell.

There are a few different types of calibration that have to be performed in order to have an accurate simulation of the work cell operation – calibration of the robot, calibration of auxiliary axes, jigs, and parts. Calibration of the robot itself can be either static or dynamic. Static calibration includes identification of static characteristics such as link lengths, joint-axis orientation, gear backlash, and coupling factors [43]. Dynamic calibration includes identification of dynamic parameters, such as forces and friction [43].

Static calibration can be done in one or two ways [43]:

- By selecting a "statistically large number of locations evenly distributed" in joint space,
- By optimizing the number of locations based on the parameters to be identified.

The result of the process of static calibration is identification of the robot's "signature" [7], which represents a set of parameters such as joint axis geometries, joint angle offsets and actuator/link compliances [43]. These parameters are incorporated into the kinematics model of the robot [7], thus improving the accuracy of the robot to approximately 1mm or less [7].

3.9. Open Development Interface

Robot simulation typically provides an open development interface, so that more advanced users can access geometric and kinematics information, as well as develop their own tools for various kinds of analysis [7]. Workspace 5 [28] provides Visual Basic for Applications (VBA) as an open development environment, which offers almost unlimited computer programming functionality. In addition to VBA, the user also has access to the Workspace component object model. Access to the component object model enables the user to retrieve information relevant to the motion of a robot and its actions. The software also provides functionality for the interfacing of custom developed dynamic link libraries (DLLs), whose basic purpose is to provide forward and inverse kinematics solutions for robots with complex structures [28].

3.10. Chapter Summary

This chapter provided more insight into robotic simulation through description of its functional structure. Each element of the structure has been explained with its advantages and disadvantages.

The next chapter will focus more on motion planning strategies. Two simple motion models will be presented as well as a method of motion tracking.

Chapter 4

Velocity Profiles and Their Impact on Cycle Times

4. 1. Introduction

The purpose of this chapter is to provide an insight into the motion strategy of a robot simulation. Equations are derived for two typical acceleration profiles – constant and linear. Each of the acceleration profiles, including the corresponding velocity profiles are analyzed in detail by each stage of motion.

4. 2. Constant Acceleration/Deceleration Motion

The basic assumption is that both the acceleration and deceleration rates are constant. Figure 4.1 provides two different velocity profiles based on the assumption that acceleration is constant.

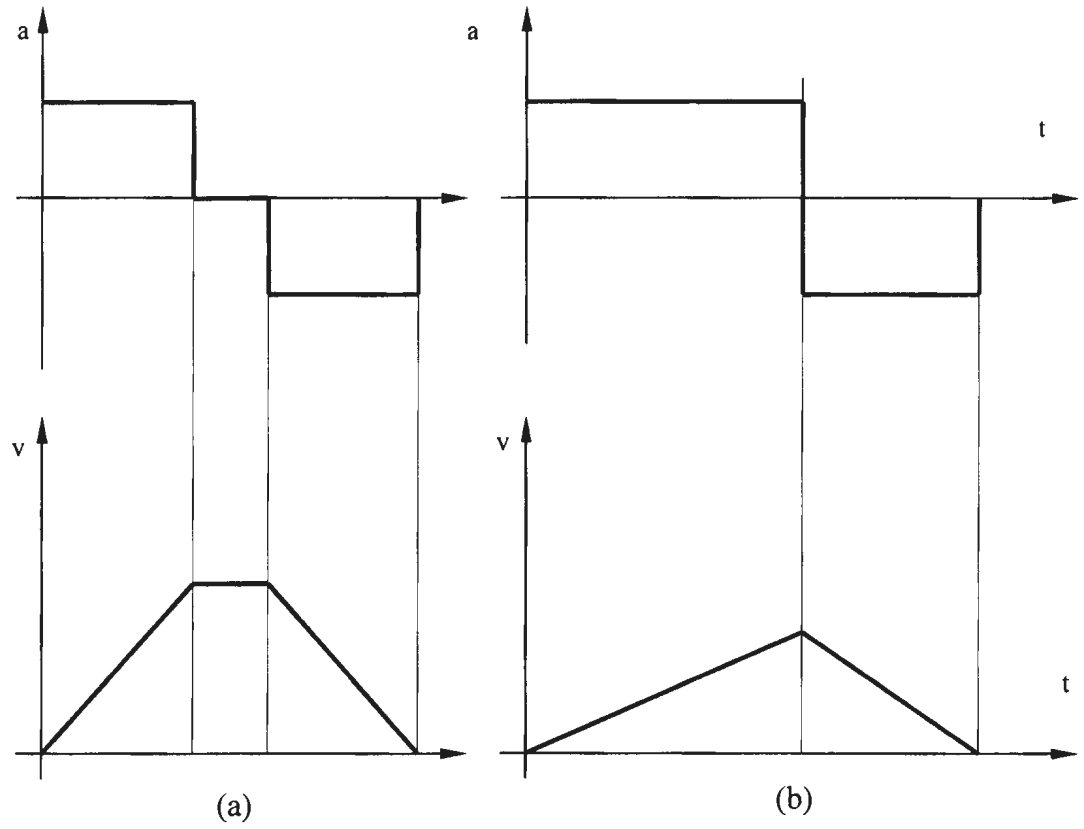


Figure 4.1 Velocity Profile for Constant Acceleration

Clearly, three stages of motion can be differentiated:

- Acceleration motion: $a = a_{accel,\max}$,
- Constant velocity motion: $a = 0$,
- Deceleration motion: $a = a_{decel,\max}$.

4.2.1. Stage 1 - Acceleration Motion

The basic assumptions related to this stage are:

- $a = a_{accel,\max}$,
- The distance between the start point and end point is large enough so that the maximum velocity can be reached,

Derivation of velocity and trajectory equations is based on the assumption about the acceleration:

$$a = a_1 = a_{accel,\max}$$

$$a_1 = \frac{dv}{dt} \Rightarrow dv = a_1 \cdot dt \quad (> 0)$$

$$\int_{v_0}^{v} dv = \int_{t_0}^{t} a_1 \cdot dt$$

$$v - v_0 = a_1 (t - t_0) \tag{4.1}$$

Boundary conditions are: $t_0 = 0$ and $v_0 > 0$.

After the boundary conditions are applied to the equation 4.1, the velocity during the stage 1 can be found as:

$$v = v_0 + a_1 \cdot t \tag{4.2}$$

In order to find the distance that the TCP travels during time t, further integration of equation 4.2 is required:

$$v = \frac{ds}{dt} = v_0 + a_1 t \Rightarrow ds = (v_0 + a_1 \cdot t)dt$$

$$\int_{s_0}^{s} ds = \int_{t_0}^{t} (v_0 + a_1 \cdot t) \cdot dt$$

$$s - s_0 = v_0(t - t_0) + a_1(\frac{t^2}{2} - \frac{t_0^2}{2}) \tag{4.3}$$

Boundary conditions: $s_0 = 0$, $t_0 = 0$, and $v_0 > 0$

After the boundary conditions are integrated into the equation 4.3, the travel distance can be found as:

$$s = v_0 \cdot t + \frac{1}{2} a_1 \cdot t^2 \tag{4.4}$$

Since the assumption is that there is enough time for TCP to accelerate to v_{max}, equations 4.2 and 4.4 can be written as:

$$t_1 = \frac{v_{\max} - v_0}{a_1} \tag{4.5}$$

$$s_1 = v_0 \cdot t_1 + \frac{1}{2} a_1 \cdot t_1^2 \tag{4.6}$$

Equation 4.5 can be integrated into the equation 4.6, by replacing parameter t_1, and that will define the travel distance during the acceleration stage:

$$s_1 = v_0 \frac{v_{max} - v_0}{a_1} + \frac{1}{2} \cdot a_1 \frac{(v_{max} - v_0)^2}{a_1^2}$$

$$s_1 = \frac{1}{2a_1}\left(v_{max}^2 - v_0^2\right). \qquad (4.7)$$

4.2.2. Stage 2 - Constant Velocity Motion

This stage of motion directly depends on the distance between the start point and end point. There are two possibilities:

1) The distance between the start point and end point is not large enough, thus the maximum velocity cannot be reached. Corresponding acceleration and velocity profiles are given on figure 4.2(b).
2) Distance between the start and end point is large enough so that there is enough time to accelerate to v_{max}. Corresponding acceleration and velocity profiles are given on the figure 4.1(a).

Equations required to define the time and distance traveled during the stage 2 cannot be calculated until the equivalent equations for stage 3 are not established.

4.2.3. Stage 3 - Deceleration Motion

During this stage, velocity linearly decreases to zero while the TCP is moving towards the target point. There are two basic assumptions about this stage of the motion:

- Acceleration is constant and has a negative sign,
- The robot stops in the target point, i.e. velocity is equal to zero in the target point.

$$a = -\frac{dv}{dt} = a_{decel,max}$$

$$a = -\frac{dv}{dt} = a_{decel,max} \Rightarrow dv = -a_3 \cdot dt$$

$$\int_{v_2}^{v} dv = -\int_{t_2}^{t} a_3 \cdot dt$$

$$v - v_2 = -a_3 \cdot (t - t_2). \tag{4.8}$$

Boundary conditions are $t_2=0$, $v_2=v_{max}$ and $v3 = 0$.

After the boundary conditions are applied to equation 4.8, the velocity during stage 3 can be found as:

$$v = v_{max} - a_3 \cdot t. \tag{4.9}$$

In order to find the travel distance during the deceleration stage, further integration of equation 4.9 is required:

$$v = \frac{ds}{dt} = v_{max} - a_3 \cdot t \Rightarrow ds = (v_{max} - a_3 \cdot t) \cdot dt,$$

$$\int_{s_2}^{s} ds = \int_{t_2}^{t} (v_{max} - a_3 \cdot t) dt ,$$

$$s - s_2 = v_{max}(t - t_2) - a_3(\frac{t^2}{2} - \frac{t_2^2}{2}) . \qquad (4.10)$$

Boundary conditions are $s_2 = 0$, $t_2 = 0$.

After the boundary conditions are integrated into the equation 4.10, the travel distance during stage 3 can be found as:

$$s = v_{max} t - \frac{1}{2} a_3 \cdot t^2 . \qquad (4.11)$$

Since the assumption is that motion will stop at the target point, equations 4.9 and 4.11 can be written as:

$$t_3 = \frac{v_{max}}{a_3} \qquad (4.12)$$

$$s_3 = v_{max} \cdot t_3 - \frac{1}{2} a_3 \cdot t_3^2 . \qquad (4.13)$$

Equation 4.12 can be integrated into the equation 4.13, by replacing the time parameter, and that will define the travel distance during the deceleration stage:

$$s_3 = v_{max} \cdot \frac{v_{max}}{a_3} - \frac{1}{2} a_3 \frac{v_{max}^2}{a_3^2} = \frac{1}{2} \frac{v_{max}^2}{a_3} . \qquad (4.14)$$

4.2.4. Stage 2 (Revisited)

As it has already been mentioned, existence of this stage depends on several parameters, such as:

- Distance between the start point and target point,
- Velocity/acceleration parameters.

It is quite simple to find relevant parameters for this stage. If the distance traveled during the acceleration and the deceleration stage is shorter than the overall distance between the start point and the target points, then the maximum velocity can be achieved. Distance between the start point and end point can easily be found as:

$$L = \sqrt{\left(x_{target} - x_{start}\right)^2 + \left(y_{target} - y_{start}\right)^2 + \left(z_{target} - z_{start}\right)^2} \qquad (4.15)$$

Using equations 4.7, 4.14 and 4.15:

$$\frac{1}{2a_1}\left(v_{max}^2 - v_0^2\right) + s_2 + \frac{1}{2}\frac{v_{max}^2}{a_3} = L,$$

$$s_2 = L - \frac{1}{2a_1}\left(v_{max}^2 - v_0^2\right) - \frac{1}{2}\frac{v_{max}^2}{a_3}. \qquad (4.16)$$

Travel time during stage 2 can be found as:

$$t_2 = \frac{s_2}{v_{max}}, \qquad (4.17)$$

where v_2 represents the maximum velocity.

If the overall travel distance required for full acceleration and deceleration stage is longer than the distance between the start point and target point, it means the following:

- Maximum velocity cannot be achieved,
- Adjustments of the acceleration and deceleration distances have to be performed,
- Velocity profile is triangular (figure 4.1(b))

Using equations 4.7, 4.14 and 4.15:

$$\frac{1}{2a_1}\left(v^2 - v_0^2\right) + \frac{1}{2}\frac{v^2}{a_3} = L. \tag{4.18}$$

The velocity that can be achieved during the motion is:

$$v^2 = \frac{2 \cdot L \cdot a_1 \cdot a_3 + v_0^2 \cdot a_3}{a_1 + a_3}. \tag{4.19}$$

By replacing v_{max} with v in equations 4.7 and 4.14, distances traveled during acceleration and deceleration stages can be found.

4.2.5. Constant Acceleration Based Motion – Summary

Table 4.1 Constant acceleration motion – summary

		Velocity Profile	
Stage	**Parameters**	**Trapezoidal**	**Triangular**
Acceleration	Time	$t_1 = \frac{v_{max} - v_0}{a_1}$	
	Velocity	$v_{max} = v_0 + a_1 \cdot t_1$	
	Acceleration	a_1	
	Distance	$s_1 = \frac{1}{2a_1}\left(v_{max}^2 - v_0^2\right)$	
Constant Velocity	Time	$t_2 = \frac{s_2}{v_{max}}$	Not applicable
	Velocity	v_{max}	
	Acceleration	0	
	Distance	$s_2 = L - \frac{1}{2a_1}\left(v_{max}^2 - v_0^2\right) - \frac{1}{2}\frac{v_{max}^2}{a_3}$	
Deceleration	Time	$t_3 = \frac{v_{max}}{a_3}$	
	Velocity	0	
	Deceleration	a_3	
	Distance	$s_3 = \frac{v_{max}^2}{2a_3}$	

4.3. Linear Acceleration/Deceleration Motion

The basic assumption of this velocity profile is that both acceleration and deceleration change linearly. The resulting velocity profile has smooth transitions between the different stages of motion, which in turn provides a more realistic representation of motion and less structural vibration of the robot due to a sudden change in acceleration/deceleration.

A key difference between the constant acceleration profile and the one with linear acceleration is in the way acceleration changes with time. As a result, two types of errors occur – motion time error and trajectory accuracy error. Since the constant acceleration velocity profile reaches maximum velocity faster than the velocity profile with linear acceleration, it means that the target location will be reached faster with the constant acceleration profile.

Similarly, for a given amount of time the constant acceleration profile means longer distance traveled than in case of the linear acceleration profile. Yet, it is not known which of the two velocity profiles resembles the motion time and trajectory accuracy closer to the ones of the real robot. One more time, it is the confidentiality of the motion planning and kinematics algorithms that represents a main obstacle in modeling of the velocity profile and leaves one only with the assumptions about the velocity profile used on the real robot.

The number of motion stages for this case is seven (see figure 4.2), while the number of possible acceleration/deceleration cases exceeds the number of cases for constant acceleration/deceleration. Each case will be described in detail here together with the supporting equations.

Figure 4.2 provides the most general case of the linear-acceleration motion between two points, with clearly defined acceleration stage, constant velocity stage and deceleration stage. The goal is the same – find the distances required by the default case, find the sum of all the distances, compare it to the real distance between the start point and end point, and then see which particular sub-case will be used for further calculations.

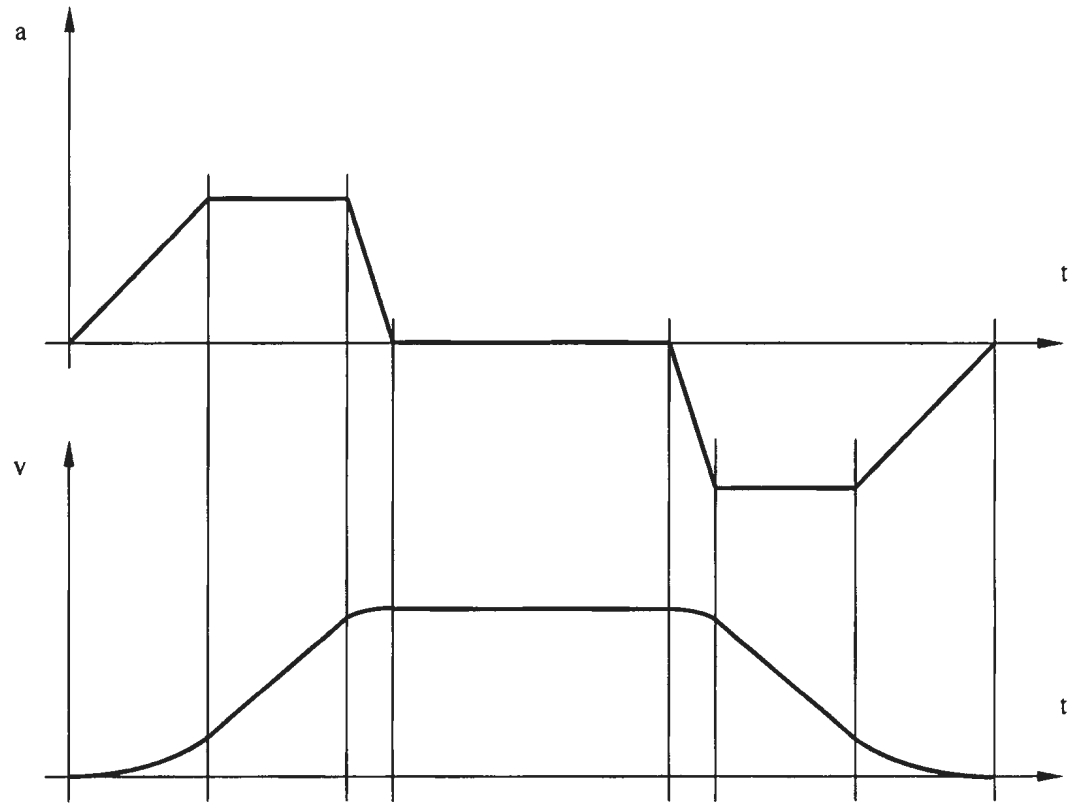
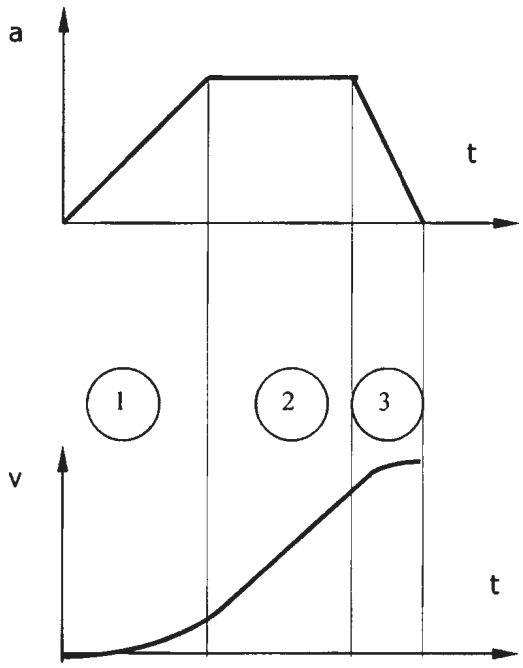


Figure 4.2 Velocity Profile for Linear Acceleration

4.3.1. Stage 1 - Acceleration Stage

Figure 4.3 represents the most general case of the acceleration stage. It is based on the assumption that the distance between the start point and target point is large enough to provide acceleration to a_{max}, and to v_{max}. The acceleration stage can be further broken down into three sub-stages:

- Sub-stage 1 – linear increase of acceleration to a_{max},
- Sub-stage 2 – constant acceleration a_{max},
- Sub-stage 3 – linear decrease of acceleration to a=0.



1 – Linear increase of acceleration
2 – Constant acceleration
3 – Linear decrease of acceleration

Figure 4.3 Acceleration Stage – General Case

4.3.1.1. Sub-Stage 1 – Linear Increase of Acceleration

Derivation begins with the following equation, which represents the formal description of the assumption that the rate of change of acceleration is positive and constant:

$$\frac{da}{dt} = k_1 > 0 \tag{4.20}$$

where k_1 represents the rate of change and is a known value.

Derivation continues with the integration of the following equation:

$$da = k_1 \cdot dt$$

$$\int_{a_0}^{a} da = k_1 \int_{t_0}^{t} dt$$

$$a - a_0 = k_1 (t - t_0) \tag{4.21}$$

Boundary conditions: $a_0=0$, $t_0=0$,

$$a = k_1 \cdot t \tag{4.22}$$

Equation 4.22 represents acceleration at any given point of time during the sub-stage 1. Acceleration time, i.e. the duration of sub-stage 1 can be found as:

$$t_1 = \frac{a_{max}}{k_1} \tag{4.23}$$

In order to find the distance traveled during the sub-stage 1, the derivation has to continue:

$$a = k_1 \cdot t$$

$$\frac{dv}{dt} = k_1 t$$

$$dv = k_1 \cdot t \cdot dt$$

$$\int_{v_0}^{v} dv = k_1 \cdot \int_{t_0}^{t} t \cdot dt$$

$$v - v_0 = k_1 \left(\frac{t^2}{2} - \frac{t_0^2}{2} \right)$$

Boundary conditions: $v_0=0$, $t_0=0$

$$v = k_1 \cdot \frac{t^2}{2} \tag{4.24}$$

The velocity reached at the end of sub-stage 1 can be found as:

$$v_1 = k_1 \cdot \frac{t_1^2}{2} \tag{4.25}$$

By replacing t_1 with equation 4.23, velocity v_1 can be found as:

$$v_1 = \frac{1}{2} \cdot \frac{a_{max}^2}{k_1} \tag{4.26}$$

Distance traveled during sub-stage 1 will be found as:

$$v = k_1 \frac{t^2}{2}$$

$$\frac{ds}{dt} = k_1 \frac{t^2}{2} \Rightarrow ds = k_1 \frac{t^2}{2} dt$$

$$\int_{s_0}^{s} ds = k_1 \int_{t_0}^{t} \frac{t^2}{2} \cdot dt$$

$$s - s_0 = k_1 \cdot \left(\frac{t^3}{6} - \frac{t_0^3}{6} \right)$$

Boundary conditions: $s_0=0$, $t_0=0$

$$s = k_1 \cdot \frac{t^3}{6} \tag{4.27}$$

Distance traveled during sub-stage 1 can be found as:

$$s_1 = k_1 \cdot \frac{t_1^3}{6} \tag{4.28}$$

By replacing t_1 with equation 4.23, travel distance s_1 can be found as:

$$s_1 = \frac{1}{6} \cdot \frac{a_{max}^3}{k_1^2} \tag{4.29}$$

4.3.1.2. Sub-Stage 2 – Constant Acceleration a_{max}

Derivation begins with the assumption that acceleration is constant and equal to a_{max}.

$$a = a_{max} = \text{const.} \tag{4.30}$$

$$\frac{dv}{dt} = a_{max} \Rightarrow dv = a_{max} \cdot dt$$

$$\int_{v_0}^{v} dv = a_{max} \int_{0}^{t} dt$$

$$v - v_0 = a_{max}(t - t_0)$$

Boundary conditions: $v_o=v_1$, $t_0=0$,

$$v = v_1 + a_{max} \cdot t \tag{4.31}$$

The velocity reached at the end of stage 2 can be found as:

$$v_2 = v_1 + a_{max} \cdot t_2 \tag{4.32}$$

By replacing v_1 with equation 4.26, velocity at the end of sub-stage 2 can be found as:

$$v_2 = \frac{a_{max}^2}{2k_1} + a_{max} \cdot t_2 \tag{4.33}$$

The distance traveled during sub-stage 2 can be found through further derivation:

$$v = v_1 + a_{max} t$$

$$\frac{ds}{dt} = v_1 + a_{max} t$$

$$\int_{s_0}^{s} ds = \int_{0}^{t} (v_1 + a_{max} t) \cdot dt$$

$$s - s_0 = v_1 \cdot (t - t_0) + a_{max} \cdot (\frac{t^2}{2} - \frac{t_0^2}{2})$$

Boundary conditions: $s_0=0$, $t_0=0$.

$$s = v_1 \cdot t + a_{max} \cdot \frac{t^2}{2} \tag{4.34}$$

The distance traveled during sub-stage 2, will be:

$$s_2 = v_1 \cdot t_2 + a_{max} \cdot \frac{t_2^2}{2} \tag{4.35}$$

4.3.1.3. Sub-Stage 3 – Linear Decrease of Acceleration

Derivation begins with the assumption that decrease of acceleration is linear and constant:

$$\frac{da}{dt} = -k_2 < 0 \tag{4.36}$$

where k_2 represents the rate of change and is a known value.

$$da = -k_2 \cdot dt$$

$$\int_{a_0}^{a} da = -k_2 \int_{t_0}^{t} dt$$

$$a - a_0 = -k_2 (t - t_0) \tag{4.37}$$

Boundary conditions: $a_0=a_{max}$, $t_0=0$,

$$a = a_{max} - k_2 \cdot t \tag{4.38}$$

Equation 4.38 represents the value of acceleration at any given point of time during the sub-stage 3. The duration of sub-stage 3 can be found as:

$$t_3 = \frac{a_{max}}{k_2}. \tag{4.39}$$

In order to find the distance traveled during the sub-stage 3, derivation has to continue:

$$a = a_{max} - k_2 \cdot t$$

$$\frac{dv}{dt} = a_{max} - k_2 t$$

$$dv = (a_{max} - k_2 t) \cdot dt$$

$$\int_{v_0}^{v} dv = \int_{t_0}^{t} (a_{max} - k_2 t) \cdot dt$$

$$v - v_0 = a_{max}(t - t_0) - k_2 \cdot \left(\frac{t^2}{2} - \frac{t_0^2}{2} \right).$$

Boundary conditions: $v_0=v_2$, $t_0=0$

$$v = v_2 + a_{max} \cdot t - k_2 \cdot \frac{t^2}{2}. \tag{4.40}$$

Velocity reached at the end of sub-stage 3 is v_{max}:

$$v_{max} = v_2 + a_{max} \cdot t_3 - k_2 \cdot \frac{t_3^2}{2}. \tag{4.41}$$

By replacing t_3 with equation 4.39, velocity v_2 can be found as:

$$v_2 = v_{max} - \frac{1}{2} \cdot \frac{a_{max}^2}{k_2} \tag{4.42}$$

Distance traveled during sub-stage 3 will be found as:

$$v = v_2 + a_{max} \cdot t - k_2 \cdot \frac{t^2}{2} \Rightarrow \frac{ds}{dt} = v_2 + a_{max} \cdot t - k_2 \cdot \frac{t^2}{2}$$

$$ds = (v_2 + a_{max} \cdot t - k_2 \cdot \frac{t^2}{2}) \cdot dt$$

$$\int_{s_0}^{s} ds = \int_{t_0}^{t} (v_2 + a_{max} \cdot t - k_2 \cdot \frac{t^2}{2}) \cdot dt$$

$$s - s_0 = v_2 \cdot (t - t_0) + a_{max} \cdot (\frac{t^2}{2} - \frac{t_0^2}{2}) - k_2 \cdot (\frac{t^3}{6} - \frac{t_0^3}{6})$$

Boundary conditions: $s_0=0$, $t_0=0$

$$s = v_2 \cdot t + a_{max} \frac{t^2}{2} - k_2 \frac{t^3}{6} \tag{4.43}$$

Distance traveled during sub-stage 3 can be found as:

$$s_3 = v_2 \cdot t_3 + a_{max} \frac{t_3^2}{2} - k_2 \frac{t_3^3}{6}. \tag{4.44}$$

By replacing t_3 with equation 4.39, and v_2 with equation 4.42 the travel distance s_3 can be found as:

$$s_3 = \frac{v_2 \cdot a_{max}}{k_2} + \frac{a_{max}^3}{3k_2^2}. \tag{4.45}$$

4.3.1.4. Sub-Stage 2 – Revisited

Now, when the velocity at the end of the sub-stage 2 is known from equation 4.42, sub-stage 2 can be revisited in order to find its travel time and distance. Using equations 4.26, 4.33 and 4.42, time t_2 can be found as:

$$t_2 = \frac{v_{max}}{a_{max}} - \frac{a_{max}}{2}\left(\frac{1}{k_1} + \frac{1}{k_2}\right). \tag{4.46}$$

By replacing t_2 in equation 4.35 with equation 4.46 and using equation 4.26, the distance traveled during sub-stage 2 can be found:

$$s_2 = \frac{v_{max}^2}{2 \cdot a_{max}} - \frac{v_{max} \cdot a_{max}}{2k_2} - \frac{1}{4}\frac{a_{max}^3}{k_1}\left(\frac{1}{k_1} + \frac{1}{k_2}\right) + \frac{a_{max}^3}{8}\left(\frac{1}{k_1} + \frac{1}{k_2}\right)^2. \tag{4.47}$$

4.3.1.5. Stage 1 – Summary

Relevant parameters for sub-stage 1:

- Motion time: $t_1 = \frac{a_{max}}{k_1}$,
- Velocity reached: $v_1 = \frac{1}{2} \cdot \frac{a_{max}^2}{k_1}$
- Distance traveled: $s_1 = \frac{1}{6} \cdot \frac{a_{max}^3}{k_1^2}$

Relevant parameters for sub-stage 2:

- Motion time: $t_2 = \frac{v_{max}}{a_{max}} - \frac{a_{max}}{2}\left(\frac{1}{k_1} + \frac{1}{k_2}\right)$,
- Velocity reached: $v_2 = v_{max} - \frac{1}{2} \cdot \frac{a_{max}^2}{k_2}$,
- Distance traveled: $s_2 = \frac{v_{max}^2}{2 \cdot a_{max}} - \frac{v_{max} \cdot a_{max}}{2k_2} - \frac{1}{4}\frac{a_{max}^3}{k_1}\left(\frac{1}{k_1} + \frac{1}{k_2}\right) + \frac{a_{max}^3}{8}\left(\frac{1}{k_1} + \frac{1}{k_2}\right)^2$.

Relevant parameters for sub-stage 3:

- Motion time: $t_3 = \frac{a_{max}}{k_2}$,
- Velocity reached: v_{max},
- Distance traveled: $s_3 = \frac{v_{max} \cdot a_{max}}{k_2} - \frac{a_{max}^3}{6k_2^2}$.

4.3.2. Stage 2 – Constant Velocity Stage

During stage 2, the value of acceleration is equal to zero. Consequently, the value of velocity remains constant - v_{max}. Motion time is calculated as:

$$t_2 = \frac{s_2}{v_{max}} \tag{4.48}$$

However, neither the travel distance s_2 nor the motion time t_2 can be calculated at this point in time. First, parameters of the deceleration stage have to be calculated, and then stage 2 will be revisited.

4.3.3. Stage 3 - Deceleration Stage

Relevant parameters of the deceleration stage can be calculated in a similar manner to that which was used for the parameters of the acceleration stage. Furthermore, an assumption can be made that the deceleration profile is symmetrical to the acceleration profile, i.e. travel times, velocities, and travel distances are exactly the same:

$$\begin{aligned} s_1 &= s_7 \\ s_2 &= s_6 \\ s_3 &= s_5 \end{aligned} \tag{4.49}$$

4.3.4. Stage 4 – Finalized Calculations

Three basic cases can be derived with respect to the distance between the start point and end point. In the first case, the distance between the start point and end point is large enough to provide both acceleration and velocity to reach their maximum values. In the second case, the distance is not large enough, and further analysis is required.

4.3.4.1. Case 1

Velocity profile and the corresponding travel distances are given on the figure 4.4.

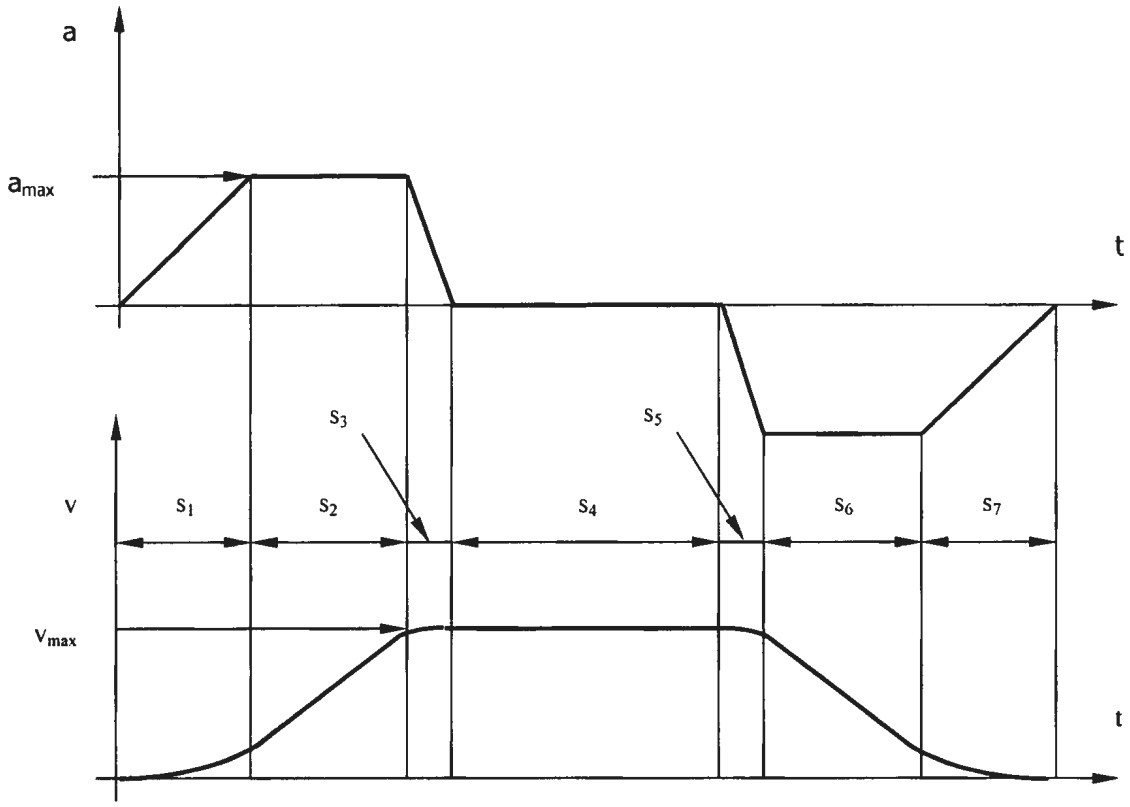


Figure 4.4 Linear Acceleration – Case 1

The distance between the start point and the end point can be found easily as:

$$L = \sqrt{\left(x_{target} - x_{start}\right)^2 + \left(y_{target} - y_{start}\right)^2 + \left(z_{target} - z_{start}\right)^2} \tag{4.50}$$

Travel distance during which the velocity has value of v_{max} can be found as:

$$s_1 + s_2 + s_3 + s_4 + s_5 + s_6 + s_7 = L \Rightarrow s_4 = L - 2 \cdot (s_1 + s_2 + s_3) \tag{4.51}$$

Travel time of sub-stage 4:

$$t_4 = \frac{s_4}{v_{max}} \tag{4.52}$$

4.3.4.2. Case 2

In this case, the distance between the start point and end point is large enough to reach the maximum value of acceleration - a_{max}, however, it is not large enough to allow the velocity to reach its maximum value – v_{max}. The velocity profile and the corresponding travel distances are given on figure 4.5.

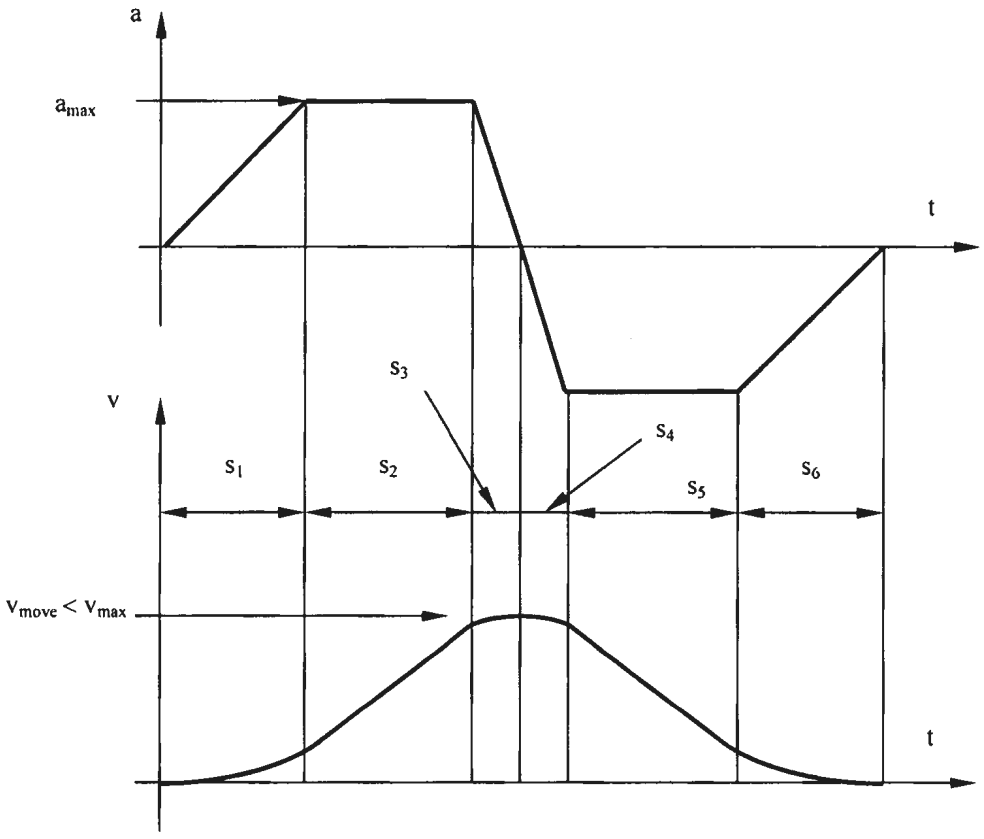


Figure 4.5 Linear Acceleration – Case 2

There are six distinct sub-stages that can be discerned. Again, assuming that the acceleration and velocity curves are symmetrical, analysis can be simplified significantly. The goal is to determine travel distance s_2, as well as the maximum velocity reached during motion.

Sub-stage 1 is defined by equations 4.23, 4.26, and 4.29. Sub-stage 2 is defined by equations 4.33 and 4.35, while sub-stage 3 is defined by equations 4.39, 4.40, and 4.44. The only difference is in the boundary condition for the sub-stage 3, during which the velocity reached is less than v_{max}.

The velocity at the end of the sub-stage 3 will be:

$$v_3 = v_2 + \cdot\frac{a_{max}^2}{2k_2} \tag{4.53}$$

Travel distance during the sub-stage 3 can be found as:

$$s_3 = v_2 \cdot \frac{a_{max}}{k_2} + \frac{a_{max}^3}{3k_2^2} \tag{4.54}$$

Using equation 4.47, travel distance s_2 can be found as:

$$s_1 + s_2 + s_3 = \frac{L}{2} \tag{4.55}$$

By replacing travel distances with corresponding expressions, equation 4.55 changes into:

$$\frac{a_{max}}{2}t^2 + \left(\frac{a_{max}^2}{2k_1} + \frac{a_{max}^2}{k_2}\right)\cdot t + \left(\frac{a_{max}^3}{6k_1^2} + \frac{a_{max}^3}{2k_1k_2} + \frac{a_{max}^3}{3k_2^2}\right) = \frac{L}{2} \tag{4.56}$$

The quadratic equation has two solutions:

$$t_{1/2} = \frac{-C_2 \pm \sqrt{C_2^2 - 4C_1C_3}}{2C_1},$$

where:

$$C_1 = \frac{a_{max}}{2};\ C_2 = \frac{a_{max}^2}{2k_1} + \frac{a_{max}^2}{k_2};\ C_3 = \frac{a_{max}^3}{6k_1^2} + \frac{a_{max}^3}{2k_1k_2} + \frac{a_{max}^3}{3k_2^2} - \frac{L}{2}$$

Two conditions have to be satisfied so that the solutions are real:

1) $C_1 > 0$ - Correct all the time, since acceleration is a positive value,

2) $C_2^2 - 4C_1C_3 \geq 0$ - In order to avoid complex solutions.

After replacing C_1, C_2 and C_3 with the corresponding equations, condition 2 can be written as:

$$a_{max}^4\left(-\frac{1}{12k_1^2} + \frac{1}{3k_2^2}\right) - a_{max}L > 0. \qquad (4.57)$$

Further derivation transforms equation 4.57 into:

$$\frac{a_{max}^3}{L}\frac{4k_1^2 - k_2^2}{12k_1^2k_2^2} > 0.$$

Since $a_{max} > 0$, as well as k_1^2 and k_2^2, only the numerator has to be larger than zero. The numerator can be represented as a product of two elements:

$$(2k_1 - k_2)(2k_1 + k_2) > 0. \qquad (4.58)$$

This equation will be satisfied if:

$$((2k_1 - k_2) > 0 \wedge (2k_1 + k_2) > 0) \vee ((2k_1 - k_2) < 0 \wedge (2k_1 + k_2) < 0). \qquad (4.59)$$

Graphical presentation of the solution is given on figure 4.6.

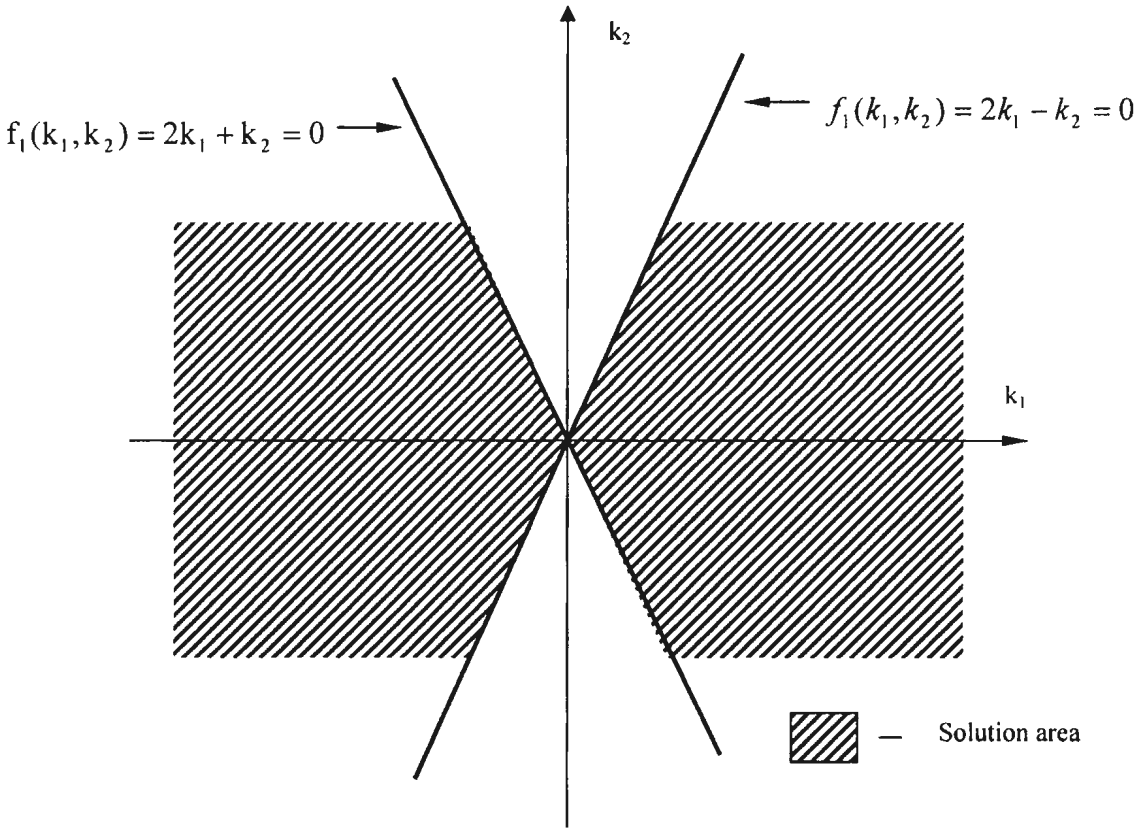


Figure 4.6 Condition 2 – Graphical Representation of the Solution

4.3.4.3. Case 3

The third general case of possible velocity profiles happens when neither maximum acceleration nor maximum velocity is reached. The corresponding velocity profile is given on figure 4.7.

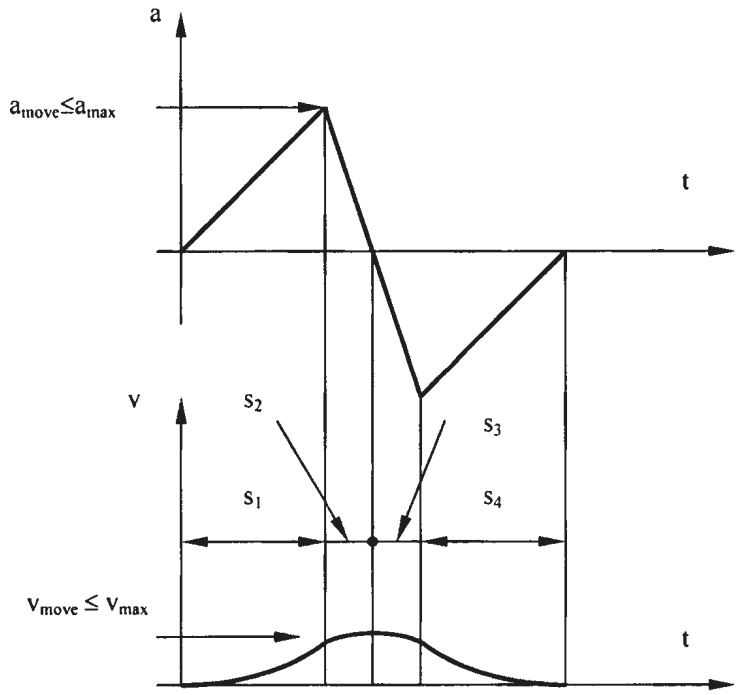


Figure 4.7 Linear Acceleration – Case 3

Four sub-stages can be noticed on the graph – s_1, s_2, s_3 and s_4. Assuming that the velocity profile is symmetric, i.e. that $s_1=s_4$ and $s_2=s_3$, significant simplification can be made. The motion equations for sub-stage 1 are:

- Travel time: $$t_1 = \frac{a}{k_1} \tag{4.60}$$

- Maximum velocity: $$v_1 = \frac{a^2}{2k_1} \tag{4.61}$$

- Travel distance: $$s_1 = \frac{a^3}{6k_1^2} \tag{4.62}$$

At the end of sub-stage 2, the velocity reached is *v*. By using equations 4.39, 4.42 and 4.59, equations for sub-stage 2 can be found as:

- Travel time: $$t_2 = \frac{a}{k_2}, \tag{4.63}$$

- Maximum velocity: $$v_2 = \frac{a^2}{2}\left(\frac{1}{k_1} + \frac{1}{k_2}\right) \tag{4.64}$$

- Travel distance: $$s_2 = \frac{a^3}{k_2}\left(\frac{1}{2k_1} + \frac{1}{3k_2}\right) \tag{4.65}$$

The maximum acceleration reached during the motion can be found as:

$$a = \sqrt[3]{\frac{3Lk_1^2k_2^2}{(2k_1 + k_2)(k_1 + k_2)}}. \tag{4.66}$$

Another set of conditions that parameters k_1 and k_2 have to satisfy can be derived using equation 4.66, i.e. the denominator has to be larger than zero:

$$(2k_1 + k_2)(k_1 + k_2) > 0 \tag{4.67}$$

The condition set in equation 4.67 is satisfied if:

$$((2k_1 + k_2) > 0 \wedge (k_1 + k_2) > 0) \vee ((2k_1 + k_2) < 0 \wedge (k_1 + k_2) < 0) \tag{4.68}$$

Figure 4.8 provides a graphical representation of the equation 4.68 solution:

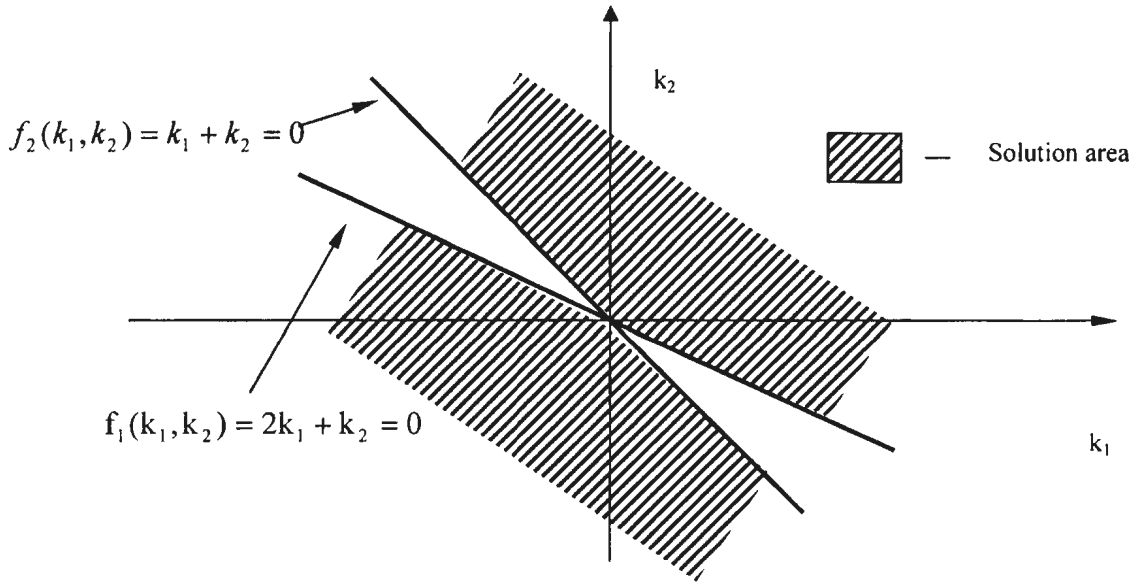


Figure 4.8 Linear Acceleration – Case 3

4.3.4.4. Special Cases

There are two special cases with respect to the velocity and acceleration reached during the motion:

1) Acceleration reaches its maximum value. Velocity also reaches its maximum value, however deceleration follows immediately. Unknowns in this particular case are related to sub-stage 2 and sub-stage 4, during which acceleration has a constant value (figure 4.9).
2) Acceleration reaches its maximum value and immediately starts decreasing (figure 4.10.) Calculations in this particular case are straightforward.

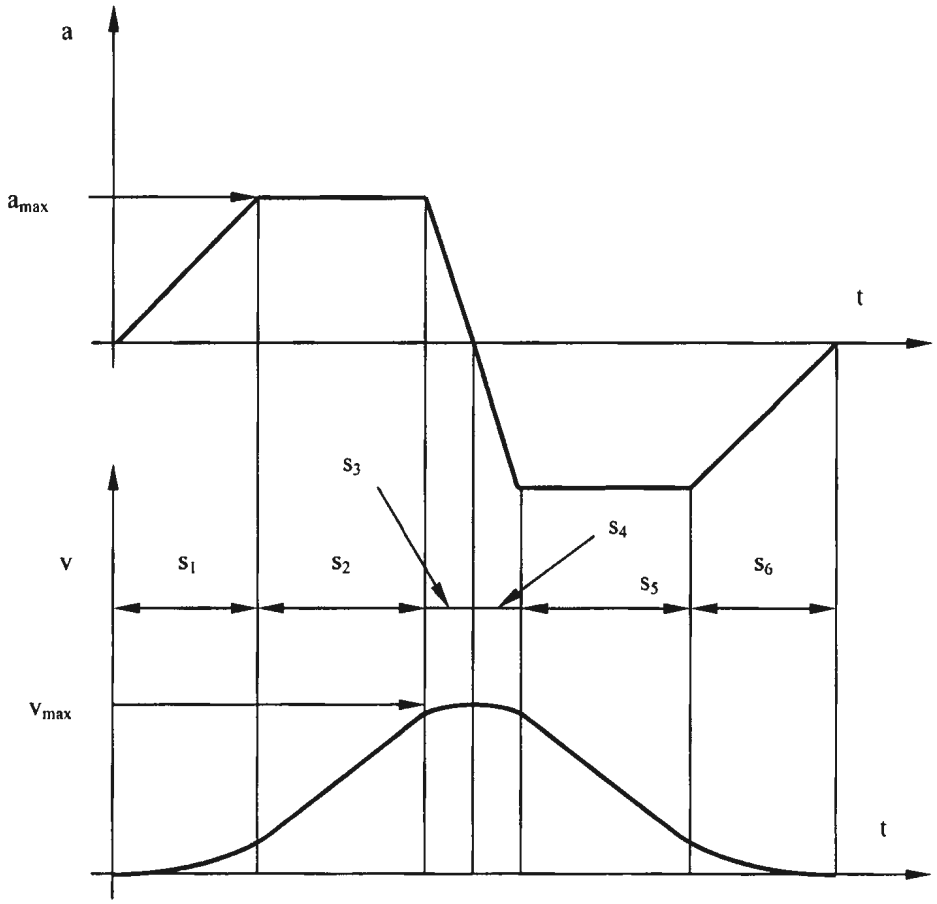


Figure 4.9 Linear Acceleration – Special Case 1

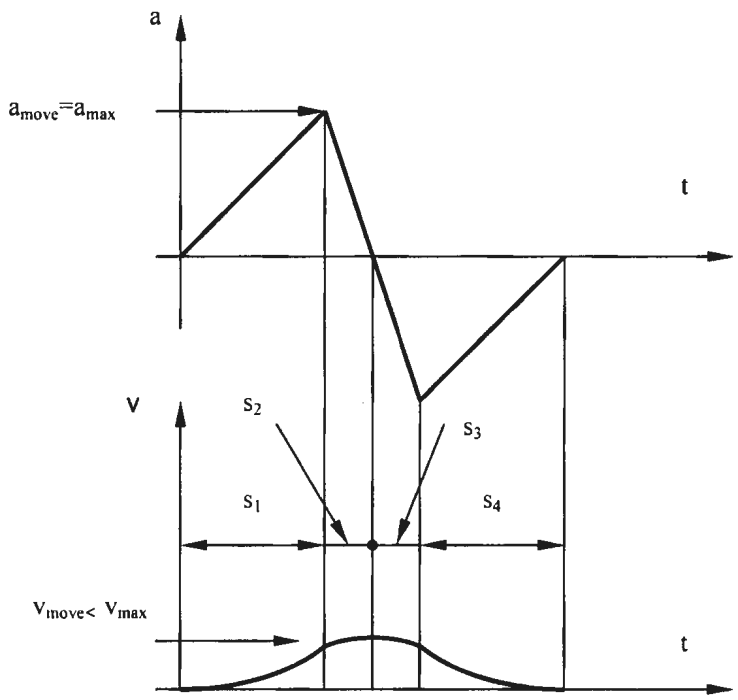


Figure 4.10 Linear Acceleration – Special Case 2

4.3.4.5. Finalized Rules for Parameters k_1 and k_2

By combining the conditions that parameters k_1 and k_2 have to satisfy, a set of solutions can be found. The conditions are:

1. $k_1 > 0$ and $k_2 > 0$
2. $((2k_1 - k_2) > 0 \wedge (2k_1 + k_2) > 0) \vee ((2k_1 - k_2) < 0 \wedge (2k_1 + k_2) < 0)$
3. $((2k_1 + k_2) > 0 \wedge (k_1 + k_2) > 0) \vee ((2k_1 + k_2) < 0 \wedge (k_1 + k_2) < 0)$

All three conditions will be satisfied if the values of parameters k_1 and k_2 are within the range:

$$k_1 > 0; k_2 > 0;$$
$$k_1 > \frac{1}{2} k_2$$

Graphical representation of the solution is given on the figure 4.11.

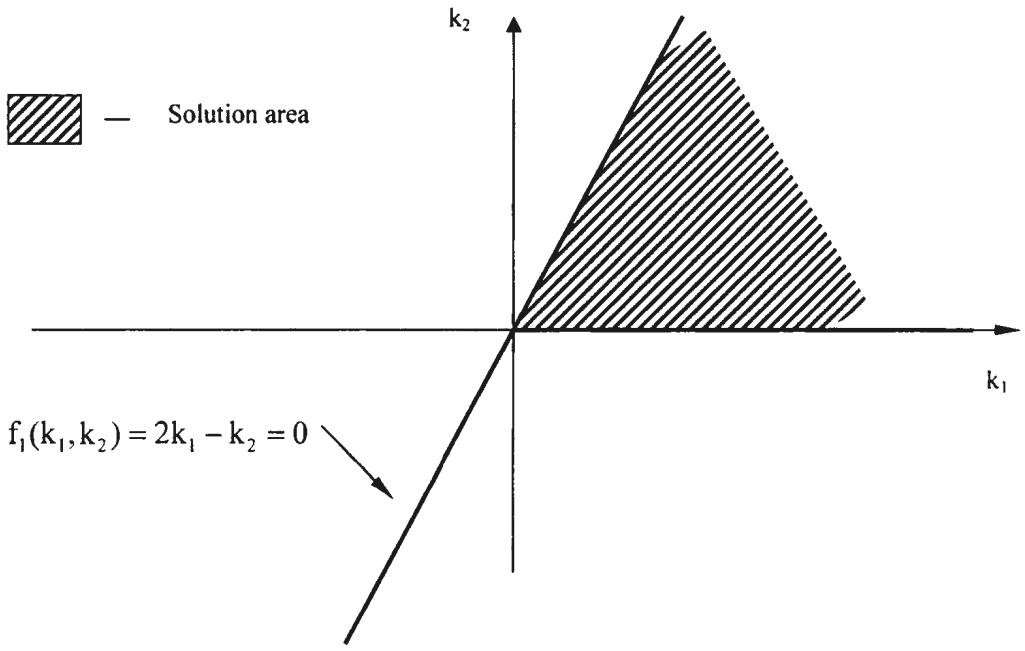


Figure 4.11 Parameters k_1 and k_2 – Solution Range

4.4. Motion Tracking

There are two motion models provided in the thesis - one based on the assumption that acceleration is constant and the other one based on the assumption that acceleration changes linearly during motion. The model used for testing is the one based on the assumption that acceleration remains constant during motion – either equal to a_{max} or to zero, depending on the motion stage (acceleration, deceleration or constant velocity motion).

Table 4.1 in section 4.2.5 gives equations that define all the parameters relevant for the constant acceleration motion model. Motion tracking can be done by monitoring the value of one of the parameters of the motion model:

- Elapsed time,
- Distance from the current TCP position to the target point,
- Distance traveled from the start teach-point to the target teach-point.

The process of robot motion tracking can be described as a series of steps:

- Step 1: Initial state – robot is in the start teach-point and motion velocity is zero,
- Step 2: Calculation of parameters for motion stages – based on the distance between the start teach-point and the target teach-point, and motion parameters associated with the target teach-point. Parameters of each motion stage can be calculated by using equations given in table 4.1.

- Step 3: Tracking of the robot's motion during the acceleration stage - by comparing the value of the elapsed time with the corresponding value calculated in step 2, or by calculating the distance between the current position/orientation of the robot's TCP, it can be concluded whether the robot is in the acceleration stage or in the stage that follows the acceleration stage (constant velocity stage or deceleration stage). The general form of the tracking equations are:
    - Motion velocity: $v_i = v_i + a_i \Delta t$, where i represents a coordinate (X, Y, Z, A, B or C) or a joint value,
    - Travel distance: $q_i = q_i + v_i \Delta t + \frac{1}{2} a_i (\Delta t)^2$, where q_i represents a generalized coordinate.

- **Step 4:** Tracking of the robot's motion during the constant velocity stage, when this stage exists:
    - Motion velocity: $v_i = v_{max}$, where i represents a coordinate (X, Y, Z, A, B, or C) or a joint value
    - Travel distance: $q_i = v_i \Delta t$
- **Step 5:** Tracking of the robot's motion during the deceleration stage:
    - Motion velocity: $v_i = v_i - a_i \Delta t$, where i represents a coordinate (X, Y, Z, A, B or C) or a joint,

- Travel distance: $q_i = q_i + v_i \Delta t - \frac{1}{2}(\Delta t)^2$, where q_i represents a generalized coordinate (X, Y, Z, A, B or C) or a joint angle/distance.

Once the target teach-point is reached, a set of parameters associated with motion to the new target teach-point is calculated, and the motion process is repeated.

4.5. Chapter Summary

This chapter provided a formal description of two basic motion models – one based on constant acceleration, the other one based on linear acceleration. Each model was determined fully with respect to the key motion parameters – acceleration, velocity, time and travel distance.

The next chapter will provide a detailed description of the motion planning problem, which results in an inaccurate estimate of motion time. Test procedure and the corresponding assumptions will also be provided in the next chapter.

Chapter 5

Innovative Method for Improvement of Simulation Motion Time Accuracy

5.1. Description of the Problem

The simple motion models used in the simulation are presented in chapter 4. The models were derived under the assumption that acceleration is either constant or changes linearly. Corresponding velocity profiles are given on figures 5.1 and 5.2.

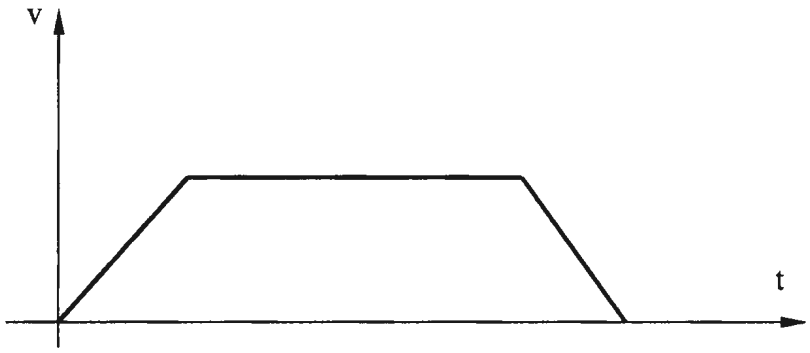


Figure 5.1 Velocity Profile for Constant Acceleration

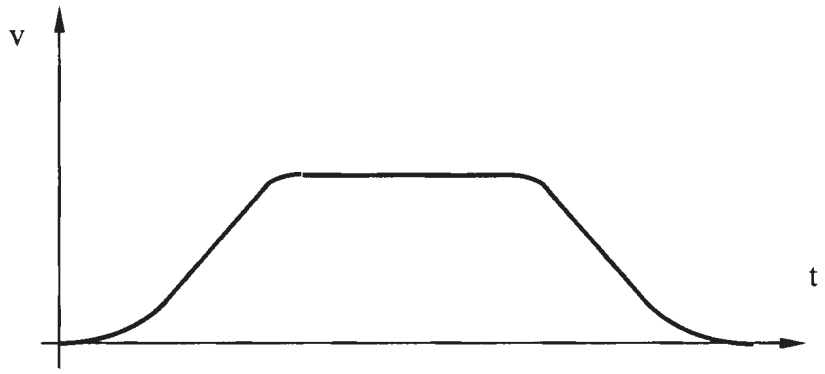


Figure 5.2 Velocity Profile for Linear Acceleration

The key problem with both models is that a number of important parameters such as mass, friction, forces and torques are not taken into consideration. Essentially, this means that both the motion time and the velocity profile will be the same, regardless of the mass of the manipulated object or the applied torques (figure 5.3). Consequently, the simulation positioning accuracy and the simulation motion time accuracy will be different from the corresponding parameters of the real robot.

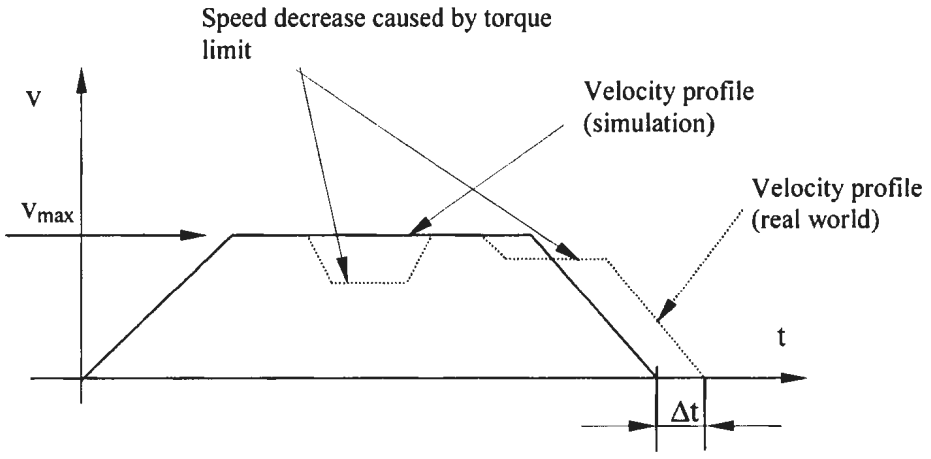


Figure 5.3 Velocity Profiles – Simulation vs. Real-World

5.2. Methods for Improvement of Simulation Motion Accuracy

Improvement of the positioning accuracy and the simulation time accuracy can be achieved in one of the two following ways:

- By integrating the original motion algorithms and kinematics algorithms through an RCS module into the simulation, which is the RRS Specification approach. However, not all the robot manufacturers provide RCS modules, the RCS module approach can be costly and the functionality implemented in an RCS module can be quite limited,
- By using a dynamics motion model instead of a simple kinematics motion model presented in chapter 4. The discussion of the dynamics motion model follows.

5.2.1. Dynamics Motion Model

Usage of dynamics equations introduces a range of new problems. The general dynamics equation of motion is:

$$\tau = M(q)\ddot{q} + h(q,\dot{q}) + V\dot{q} + g(q), \qquad (5.1)$$

where:

τ	torque matrix,
$M(q)$	inertia matrix,
$h(q,\dot{q})$	vector representing centrifugal and Coriolis forces,
$V(\dot{q})$	joint friction matrix,
$q(q)$	gravity load vector,
q	generalized coordinate.

The complexity of the dynamics motion model is very high regardless of the method used for its derivation.

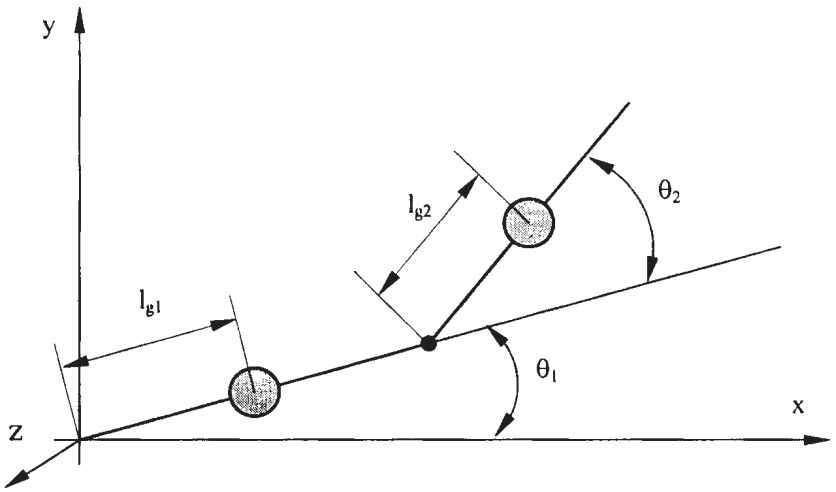


Figure 5.4 A Simple Two-link Planar Manipulator

For example, Lagrange dynamics equations for a two- link robot shown on figure 5.4 are:

- Joint 1 torque:

$$\tau_1 = [m_1 l_{g1}^2 + I_1 + m_2(l_1^2 + l_{g2}^2 + 2l_1 l_{g2}\cos\theta_2) + I_2]\ddot{\theta}_1 + [m_2(l_{g2}^2 + l_1 l_{g2}\cos\theta_2) + I_2]\ddot{\theta}_2 -$$

$$m_2 l_1 l_{g2}\sin\theta_2(2\dot{\theta}_1\dot{\theta}_2 + \dot{\theta}_2^2) + m_1 g l_{g1}\cos\theta_1 + m_2 g(l_1\cos\theta_1 + l_{g1}\cos(\theta_1 + \theta_2)$$

- Joint 2 torque:

$$\tau_1 = [m_2(l_{g2}^2 + l_1 l_{g2}\cos\theta_2) + I_2]\ddot{\theta}_1 + (m_2 l_{g2}^2 + I_2)\ddot{\theta}_2 + m_2 l_1 l_{g2}\sin\theta_2\theta_1^2 +$$

$$m_2 g l_{g2}\cos(\theta_1 + \theta_2)$$

Equations become significantly more complex for six-link robots.

There are a few important aspects of the dynamics motion model that should be mentioned:

- The number of operations that need to be performed is very large. Using the Lagrange method, 66271 multiplications and 51548 additions need to be computed so that the torque matrix for a six-link robot can be found. The Newton-Euler method requires 852 multiplications and 738 additions. The Reibert-Horn method requires 468 multiplications and 264 additions [45, 46].

- Even with the significantly reduced number of operations to be performed, another important problem remains unsolved – the frequency of performing the calculations. Calculations have to be performed for every single interpolation point, which makes the model computationally expensive [4]. The problem is compounded if there are two or more robots used in the simulation.

- The values of the parameters included in the dynamics motion model must be known in order to be used in the simulation. The task of identifying parameters such as inertia and friction that are used to create the corresponding matrices represents a challenge, since parameters such as friction are coupled with other dynamic parameters [9]. Furthermore, separate modeling and measurement of the dynamics parameters is needed, which makes identification even more difficult and time consuming [9].

- Real robots carry various tools and cables required for performing the task that they are programmed for. Welding cables and painting cables can be quite heavy, thus they influence the dynamics motion model, too. So, in order to have the dynamic motion model accurately represent the real robot, the influence of cables and tools also has to be incorporated.

Ultimately, even if the dynamics equations accurately described motion of the real robot, the simulation motion time would not be the same as the motion time of the real robot because of the internal robot controller algorithms, which are confidential.

5.3. The Description of the Proposed Method

The method proposed in this thesis can be classified as an inverse calibration method. Inverse calibration requires neither the identification of the form of the error, nor the source of the error, but a way to compensate the errors. Although more measurements are required, a better match to the real system may be achieved [46].

Various parameters, both known and unknown can influence motion of a real robot. A goal of the method proposed is not to identify them all but to express their influence through a limited set, whose influence can be determined with a relative ease. Influence of each parameter, i.e. an error generated by the influence of each parameter

will be compensated through inclusion of a corresponding correction factor in one of two motion models presented in Chapter 4.

Approximation of the real robot's motion model is based on the following parameters:

- Incline angle – motion of robot's TCP in vertical plane,
- Bearing – motion of robot's TCP in horizontal plane,
- Radial distance of the start point from the coordinated system located into the base of the robot,
- Tool orientation.
- Mass of the manipulated object,
- Configuration of the robot,

Selection of the influential parameters is based on the simple kinematics motion model parameters presented in chapter 4. Although the key parameters of the two motion models presented are travel distance, velocities and accelerations, both models recognize that position data and orientation data of teach-points as well as the motion direction are known. The influence of inertia caused by the robot's own mass and the mass of the tool or the object is also incorporated through parameters such as the mass of the tool or of the object, distance of the start teach-point away from the base of the robot, and the configuration of the robot.

Another way to justify the selection of the influential parameters is based on the fact that a robot performs tasks by moving a tool or an object through a set of teach-points that make a trajectory. Thus, radial distance and tool orientation define the influence of the location of a teach point, while the incline angle and bearing define the influence that motion direction has on motion time. Configuration of the robot defines the influence of the robot structure, while the mass defines influence of the manipulated object/tool on motion time.

The influence of each parameter can be established by performing simple motion tests with a real robot. Based on the results of the tests, functional relationships between the motion time of the real robot and the listed influential parameters can be established. Those functional relationships serve as a basis for determining the values of correction parameters, which will compensate the error values and provide a more accurate simulation motion time (figure 5.5).

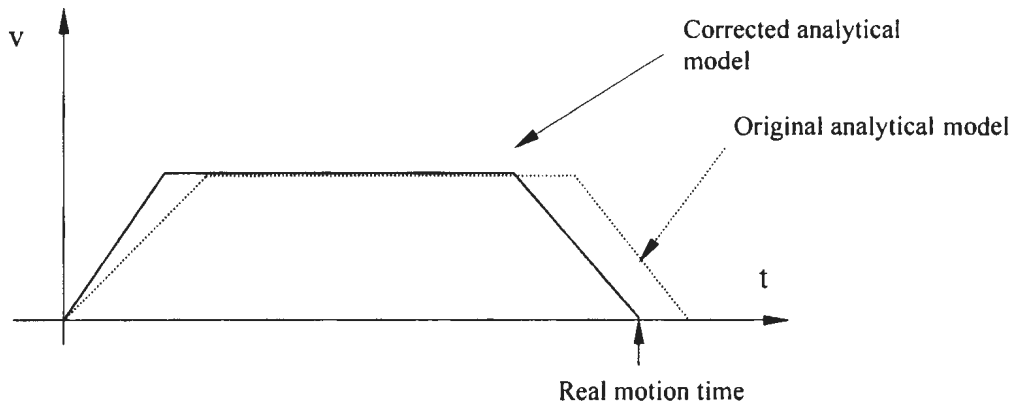


Figure 5.5 Simulation Velocity Profiles – Original vs. Corrected

A valid question can be raised about the identification of parameters other than those that have been previously mentioned, which could influence motion of the real robot and their subsequent incorporation in the motion model of the robotic simulation. As mentioned earlier, identification of the influence of some parameters is not easy. However, the method presented in this thesis incorporates both the known parameters, whose influence can be identified easily and the "hidden" parameters, which are difficult to be identified and integrates their influence through a set of parameters listed in the section 5.3. This approach represents one of the highlights of the proposed method.

Other benefits of the method proposed in the thesis are:

- There is no need to build highly complex motion models. Influence of the parameters such as cables, tools, motion algorithms and kinematics algorithms is compensated through usage of the correction factors,
- The correction factor database is established by the simulation user and is based on the results of the experiments. It is up to the user to decide how fine the approximation will be,
- Once the correction factor database is established, it can be used for a certain period of time until the need for its revision arises.
- Computation-wise, the correction factor method is supreme compared to the method of dynamic equations. The values of the correction factors associated to each teach-point of the trajectory are typically retrieved from the database prior to the actual simulation of motion. The constant acceleration motion model used in the simulation is computationally inexpensive and tracking of robot motion requires only the comparison of the elapsed time to the time needed for a particular motion stage (e.g. acceleration or deceleration).

5.4. Test Assumptions

The experiment was conducted under the following set of assumptions:

1) During the test, the robot remained in its "natural" configuration. This configuration is similar to the one that the robot shown on figure 5.6 has. The goal of this assumption is to eliminate variability resulting from the nature of a robot as a serial link manipulator which allows for a teach point to be reached in several different configurations.

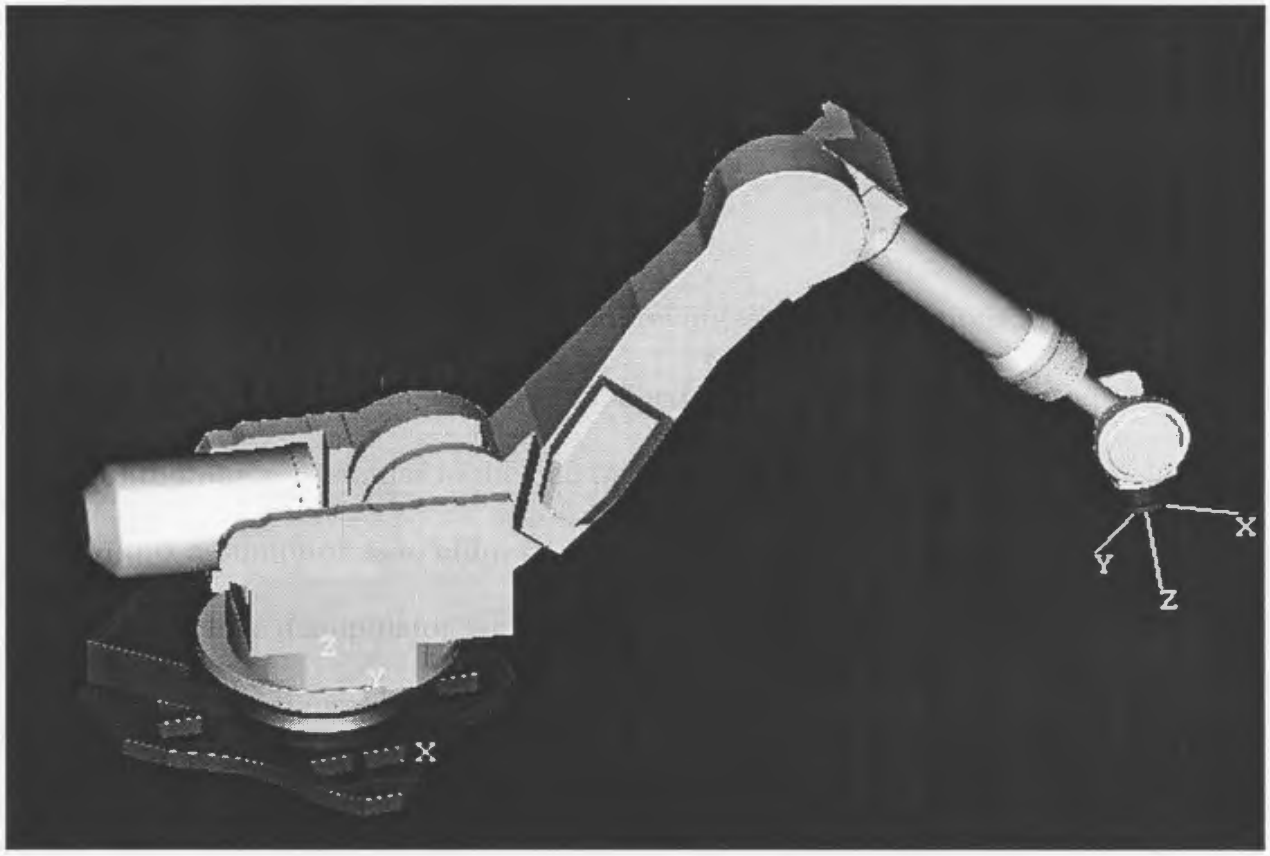


Figure 5.6 "Natural" Robot Configuration [28]

2) Mass of the tool attached to the flange is constant – although the mass of the tool and/or the object carried in the gripper represents an influential parameter, it is kept constant during the testing.
3) Teach points used for testing are far enough from each other to allow the robot to reach maximum linear velocity.
4) Teach points are located directly in front of the robot. The assumption is that most of the tasks are performed in this part of robot's envelope.
5) Orientation of the tool is kept constant during the test.
6) Motion type used during the test is linear.

5.5. The Test

For research purposes of this thesis, the influence of only two out of the six parameters listed in section 5.3 were tested – bearing and incline angle. These two parameters can be understood as the "basic" parameters, because the influence of every other assumed influential parameter is tested and identified based on the known influence of bearing and incline angle.

Two experiments were performed. In the first experiment, the motion direction of the robot's TCP was kept horizontal, i.e. motion was performed in a horizontal plane that was a fixed distance away from the base frame of the robot (figure 5.7). The key idea behind this test was to establish the influence of bearing on motion time of the real robot.

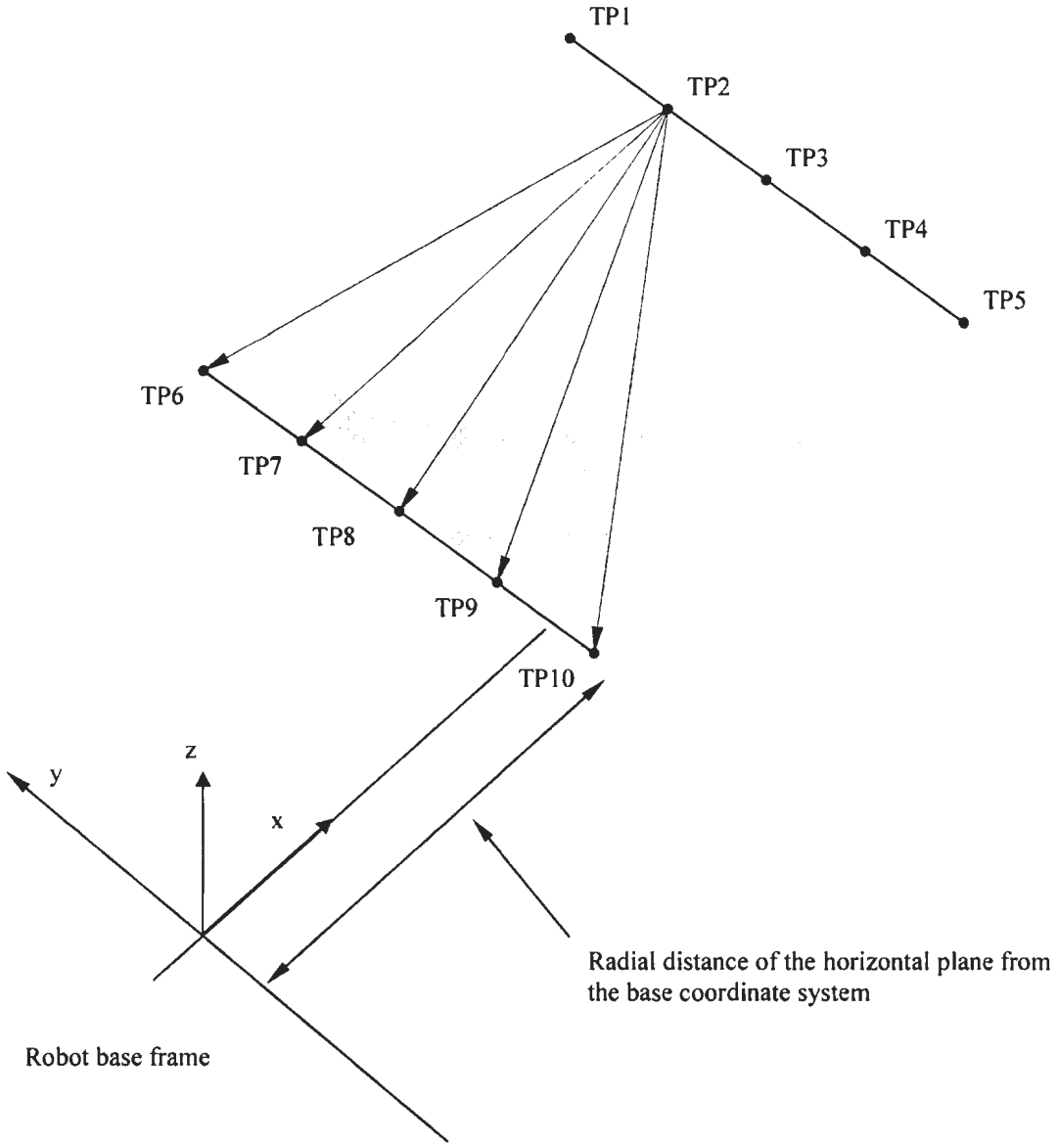


Figure 5.7 Test Description – Motion in Horizontal Plane

In the second experiment, motion direction was kept vertical, i.e. motion of the robot's TCP was performed in a vertical plane (figure 5.8). Similar to the horizontal plane motion test, it is the influence of the incline angle whose influence on motion time of the real robot was established.

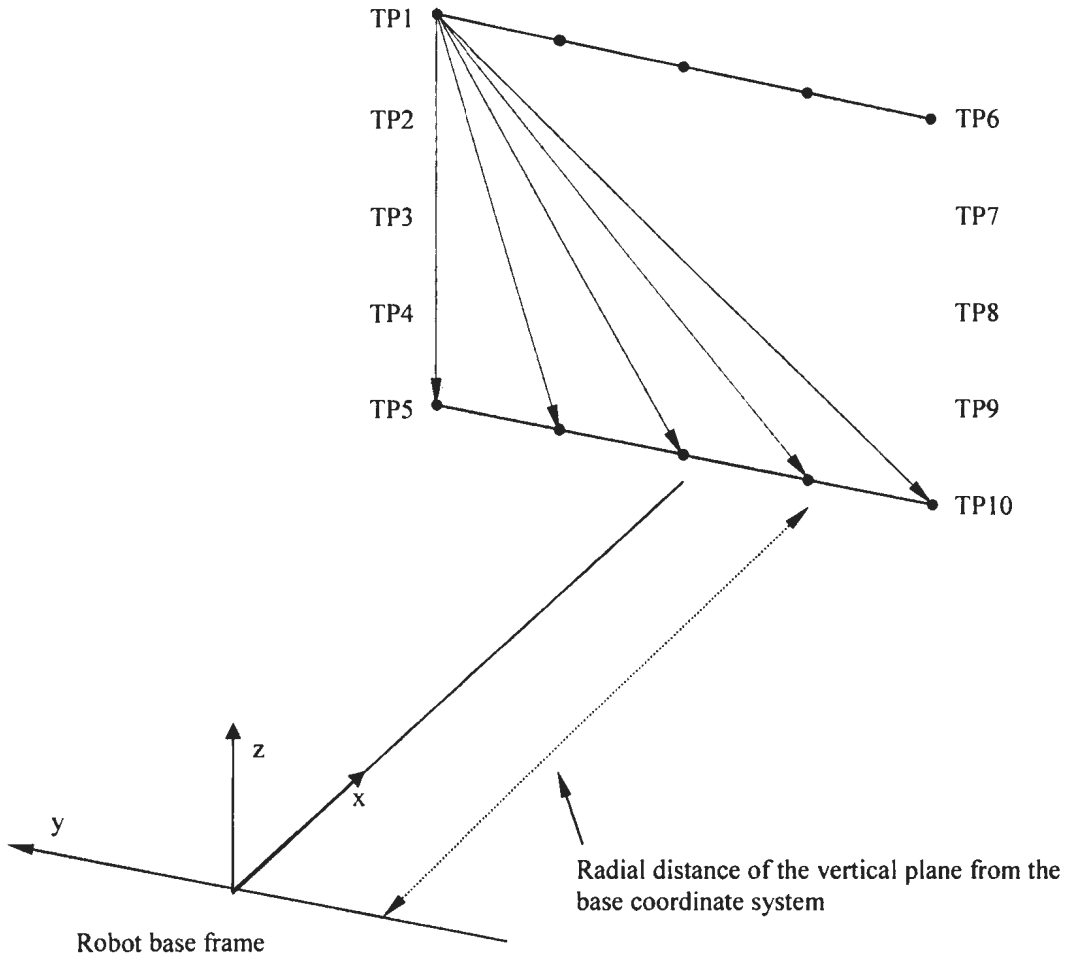


Figure 5.8 Test Description - Motion in Vertical Plane

The model of the robot used for testing was a MOTOMAN UP20 [47] with an XRC controller [48]. Both the manipulator and the controller were never used in service before, which means a reduction of the potential errors caused by the electro-mechanical systems. Another important fact is that the teach-pendant used with the XRC controller has the functionality of displaying the actual motion time of the robot, which made the measurements accurate. Furthermore, the repeated tests for the same trajectory resulted in the same motion time. Other relevant test data include:

- Linear velocity: 300mm/s (used both on the real robot and in the simulation),
- Simulation acceleration: 400mm/s^2

5.5.1. Bearing – Approach Motion

The purpose of the approach motion test was to establish the values of the motion times between the teach-points while the TCP was moving linearly towards the base of the robot (figure 5.9). All teach-points used in the test belong to the same horizontal plane, which can be seen on figure 5.10 and figure 5.11. Coordinates of the start teach-points are given in tables A.1 and A.2, while the corresponding motion times are given in tables A.9, A.10, A.11 and A.12.

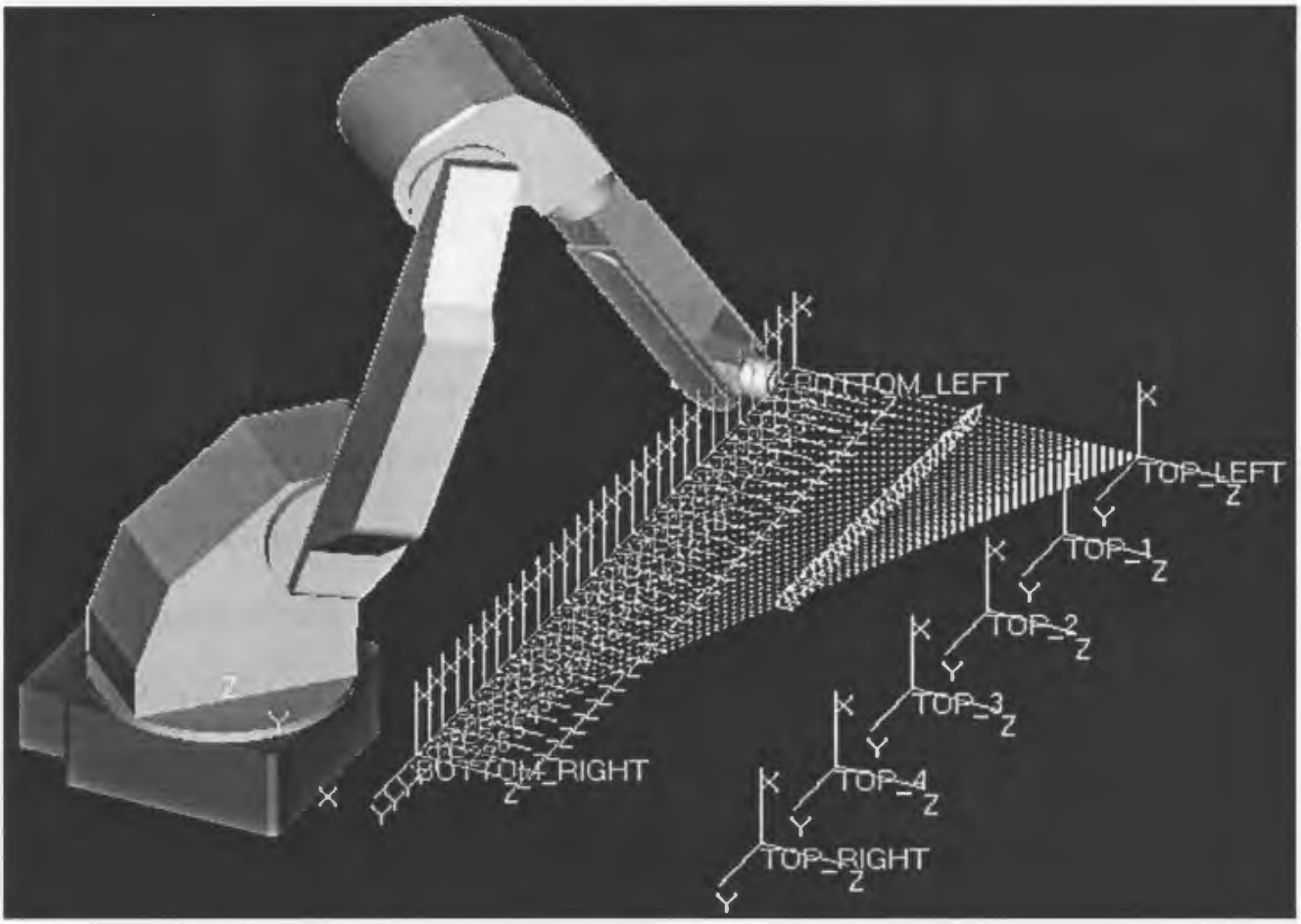


Figure 5.9 Approach Motion Test – Isometric View

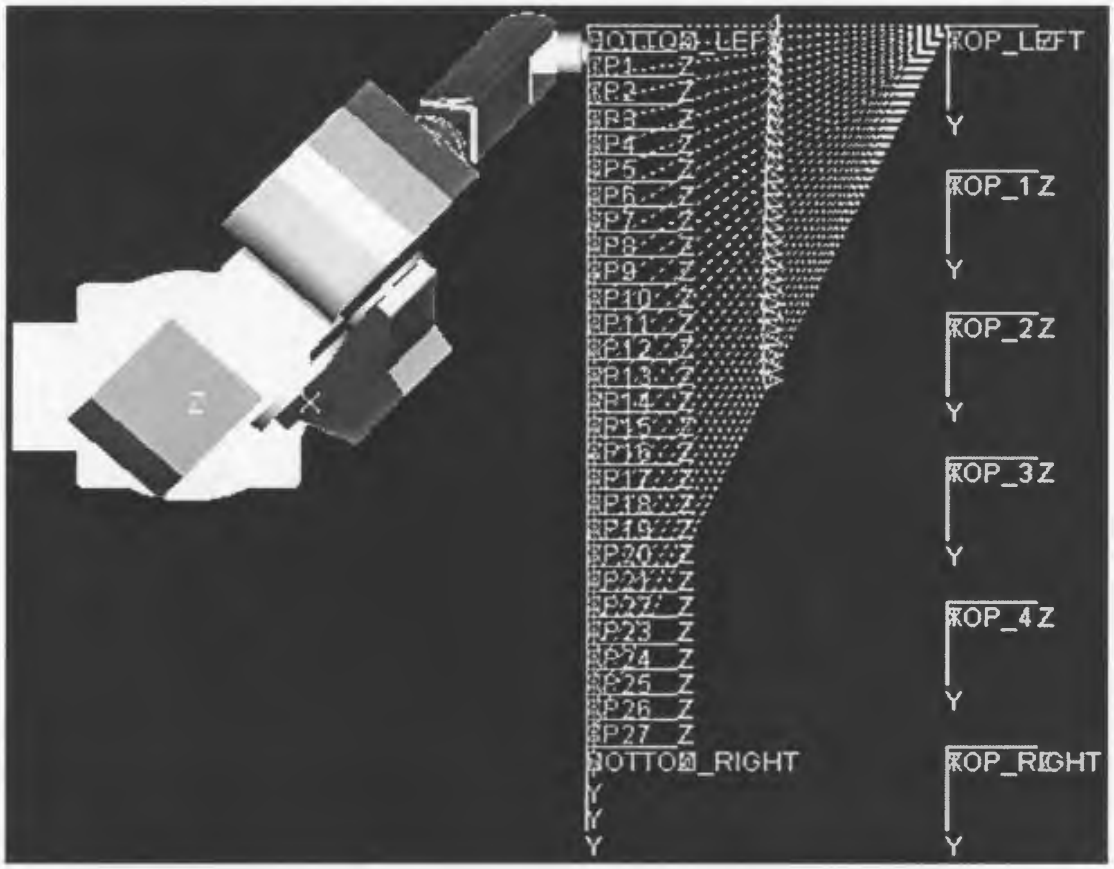


Figure 5.10 Approach Motion Test – Top View

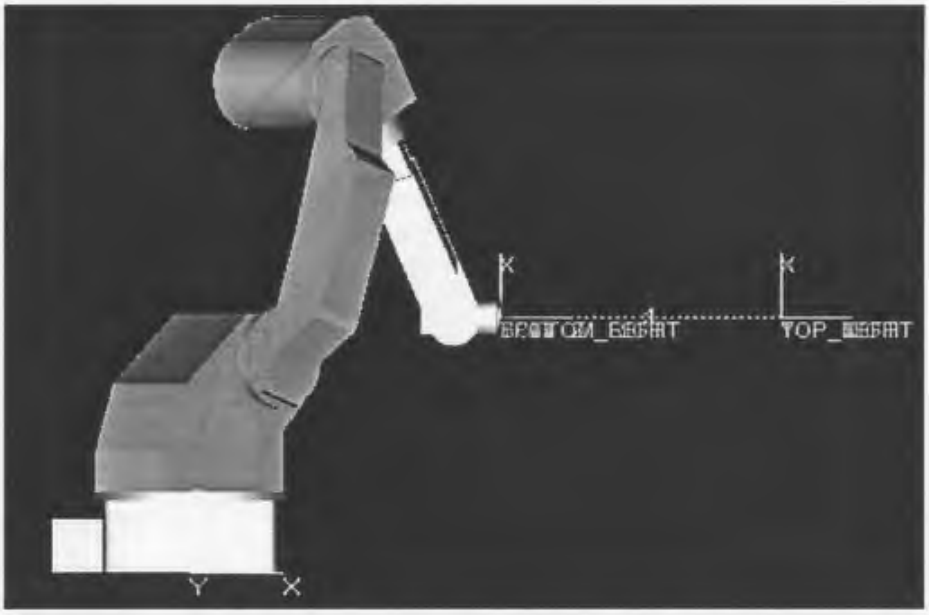


Figure 5.11 Approach Motion Test – Side View

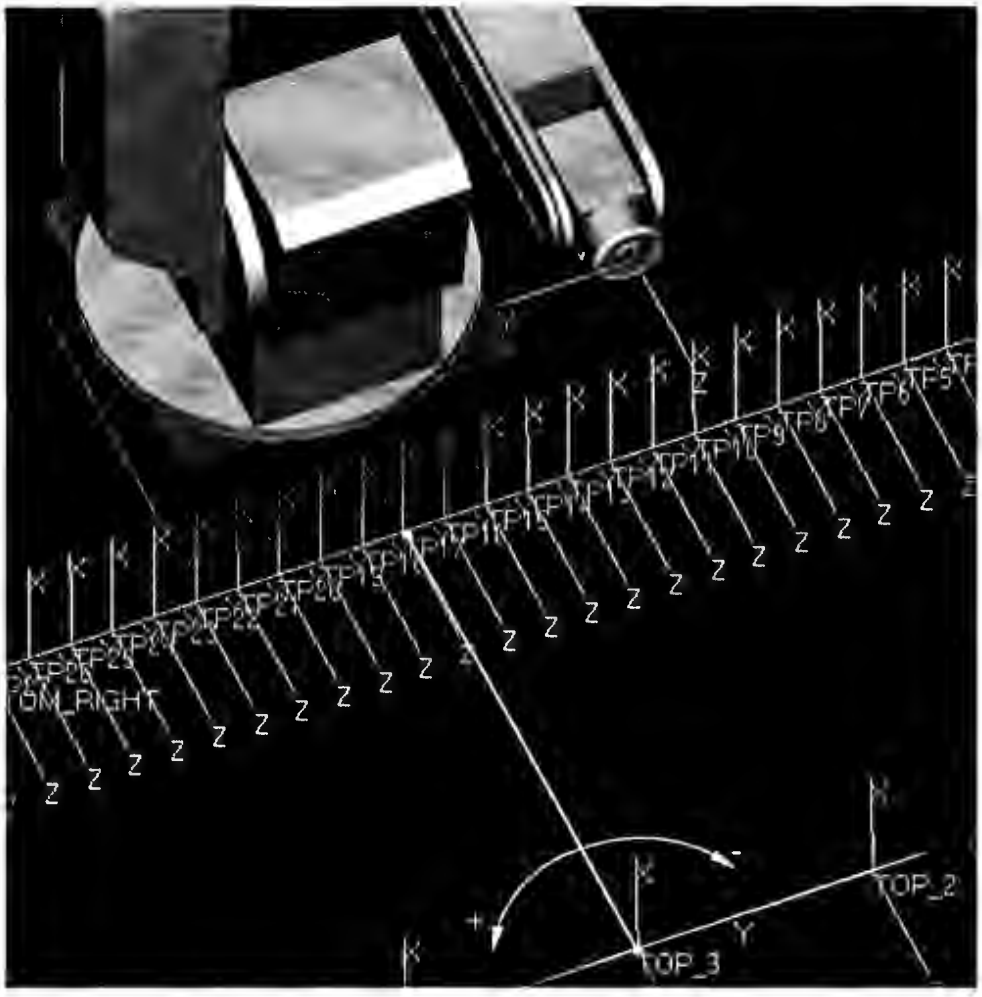


Figure 5.12 Bearing Angle for Approach Motion

Values of the bearing angle (figure 5.12) were calculated by using the following formula:

$$\text{Bearing} = \text{atn}\left(\frac{Y_{TP} - Y_{TOP}}{X_{TP} - X_{TOP}}\right), \tag{5.2}$$

where X_{TP}, Y_{TP} represent the position of a target teach-point TP_i relative to the base frame of the robot, and X_{TOP}, Y_{TOP} represent the position of a start teach-point.

5.5.2. Bearing – Depart Motion

The depart motion test was performed using a similar approach to that of the approach motion test. Figures 5.13, 5.14 and 5.15 provide the visual presentation of the performed test while figure 5.16 provides a definition of the bearing angle for depart motion. Coordinates of the teach-points can be found in tables A.3 and A.4 in the appendix, while the motion times of the real robot and of the simulation, and the corresponding error values can be found in tables A.13, A.14 and A.15.

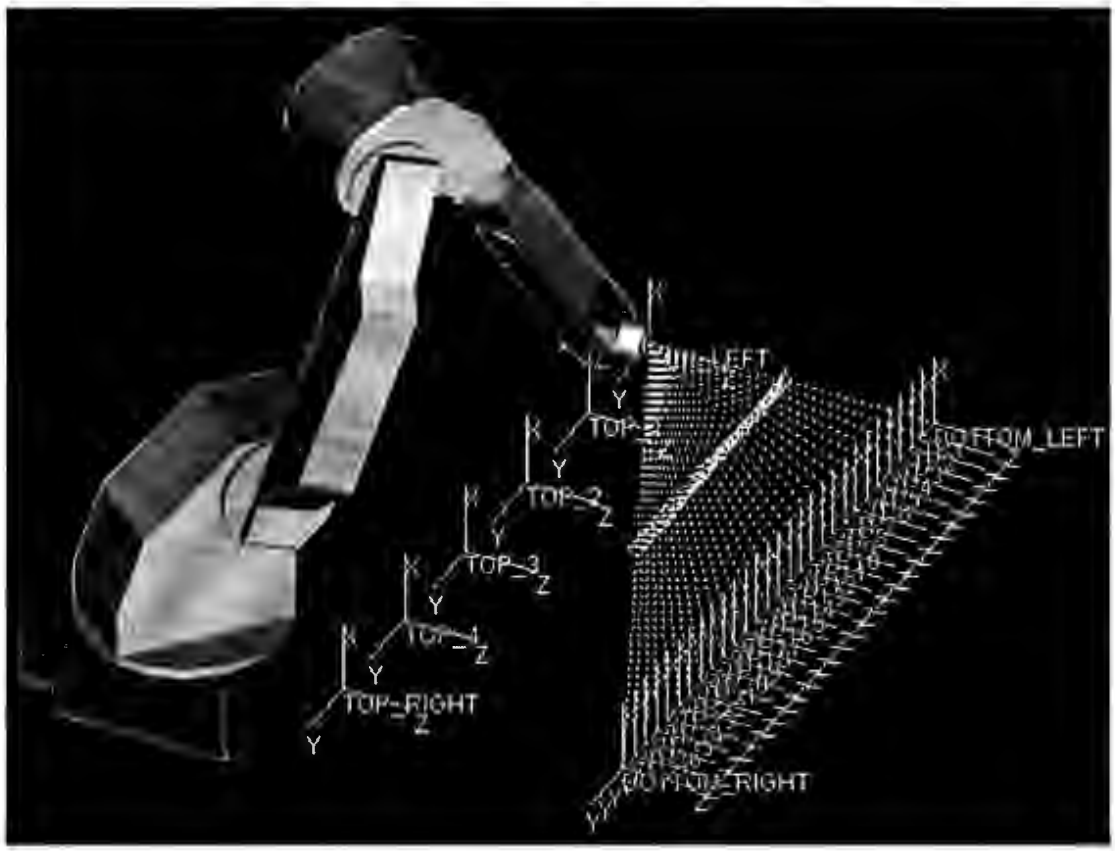


Figure 5.13 Depart Motion Test – Isometric View

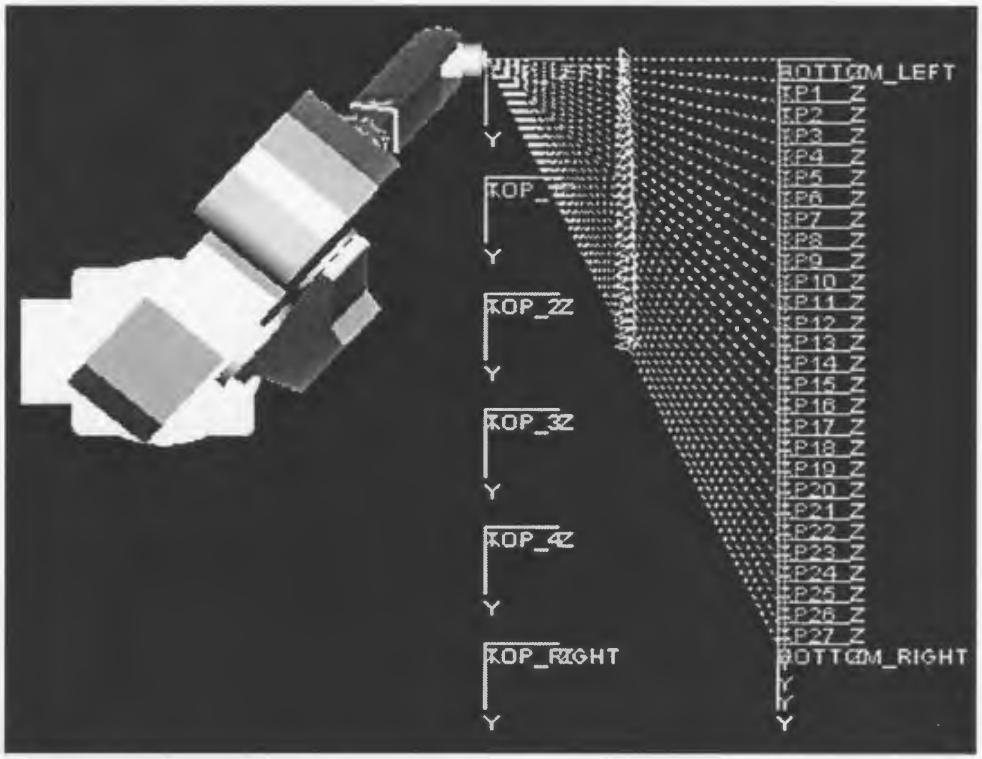


Figure 5.14 Depart Motion Test – Top View

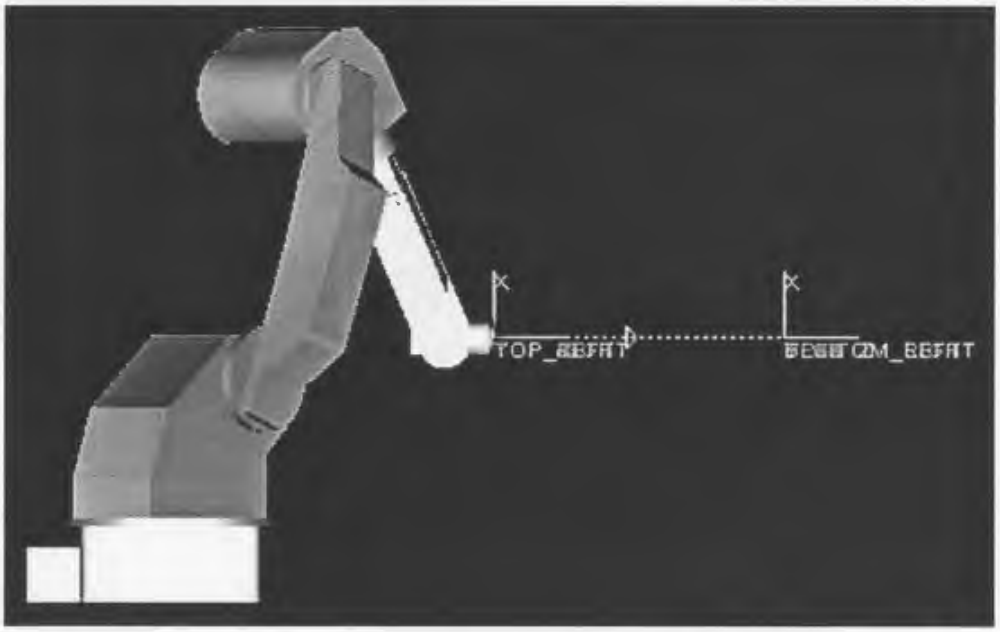


Figure 5.15 Depart Motion Test – Side View

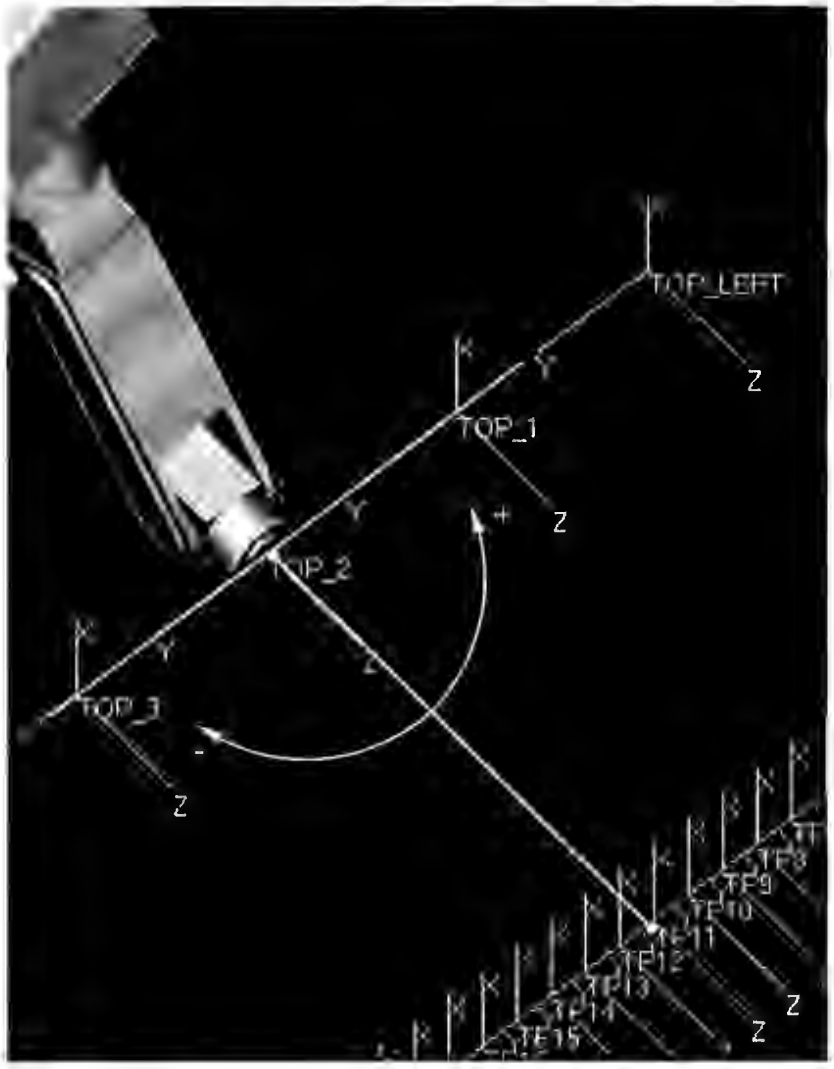


Figure 5.16 Bearing Angle for Depart Motion

The value of the bearing angle can be calculated by using the equation 5.2.

5.5.3. Incline Angle – Downward Motion

The downward motion test consists of sets of motion between the start teach-points and target teach-points, all belonging to the same vertical plane (figure 5.17). The top view and the side view are given on figure 5.18 and on figure 5.19. Coordinates of the start teach-points and of the target teach-points are given in the table A.5 and A.6. Motion times of the real robot, of the simulation, and the corresponding error values are given in tables A.16 and A.17.

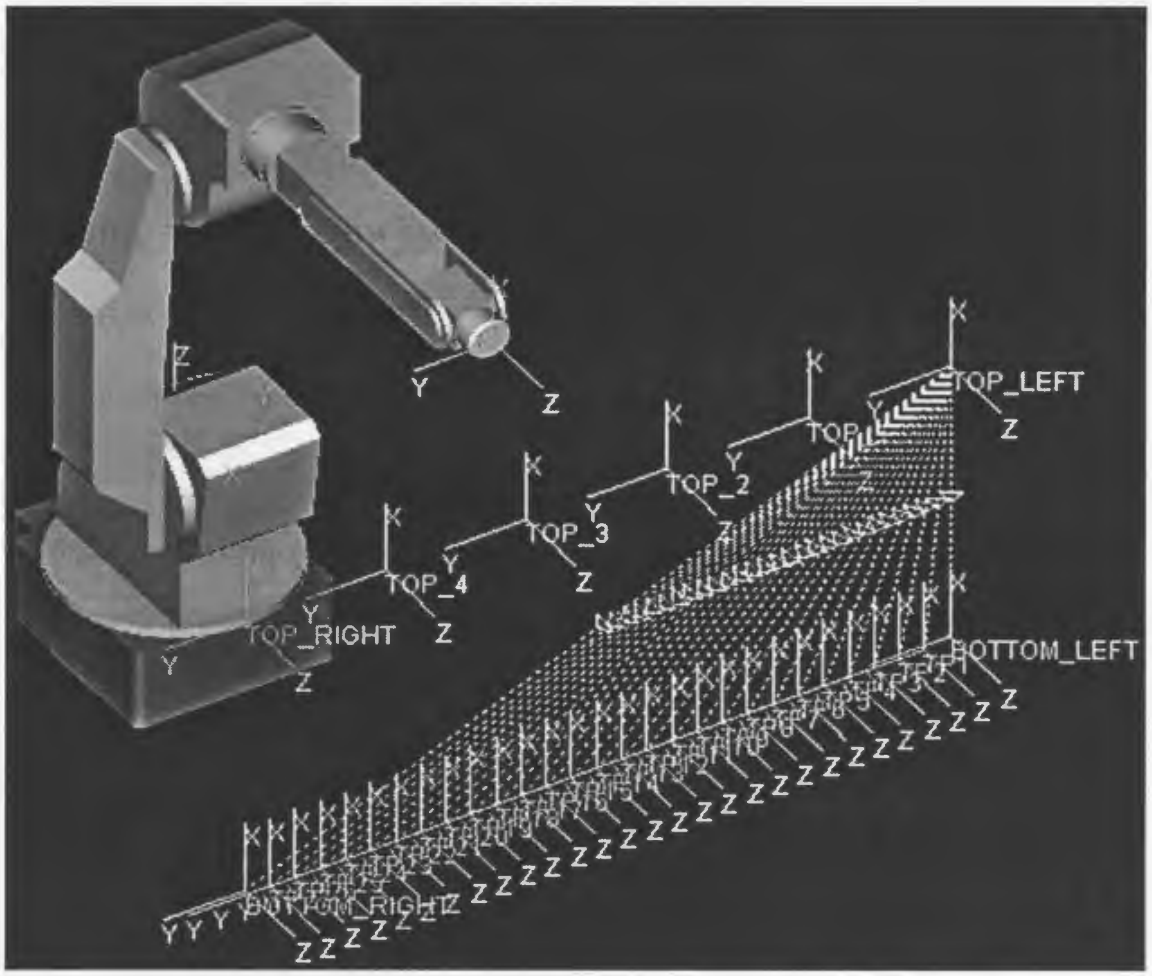


Figure 5.17 Downward Motion Test – Isometric View

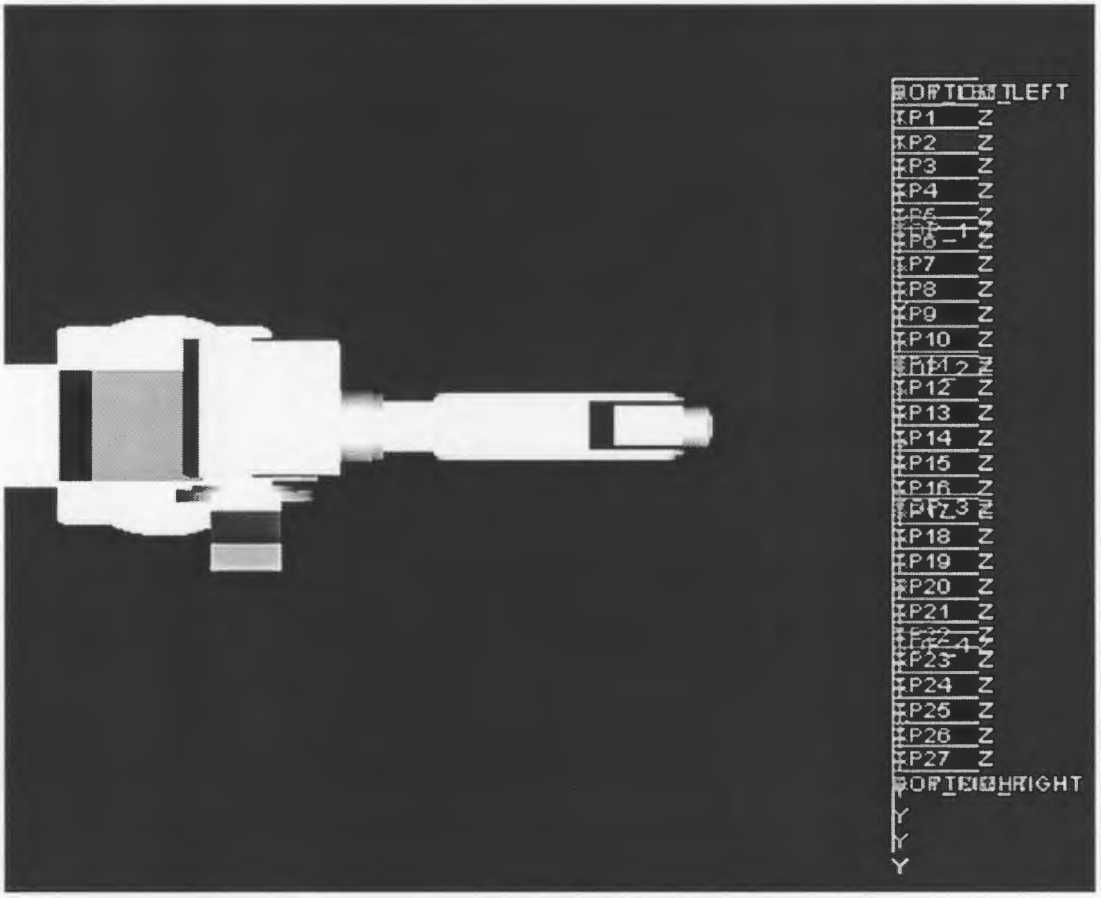


Figure 5.18 Downward Motion Test – Top View

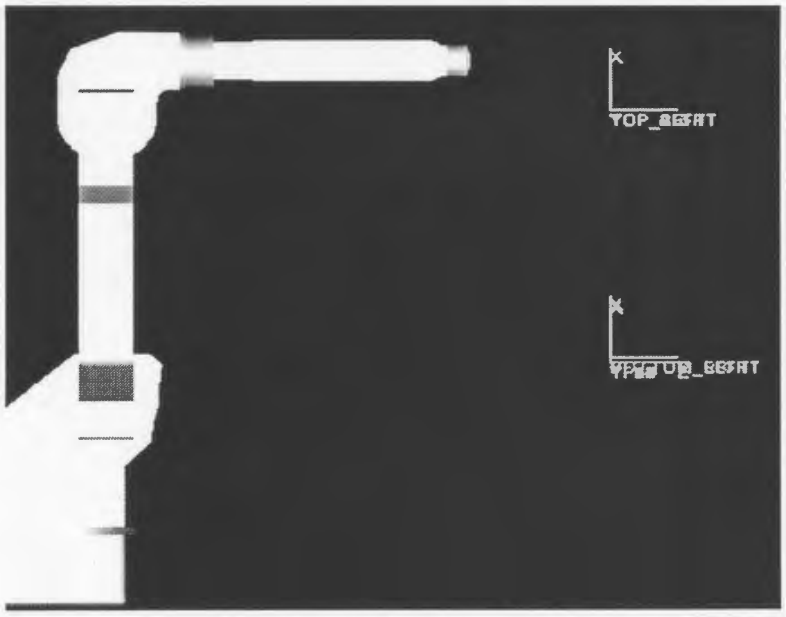


Figure 5.19 Downward Motion Test – Side View

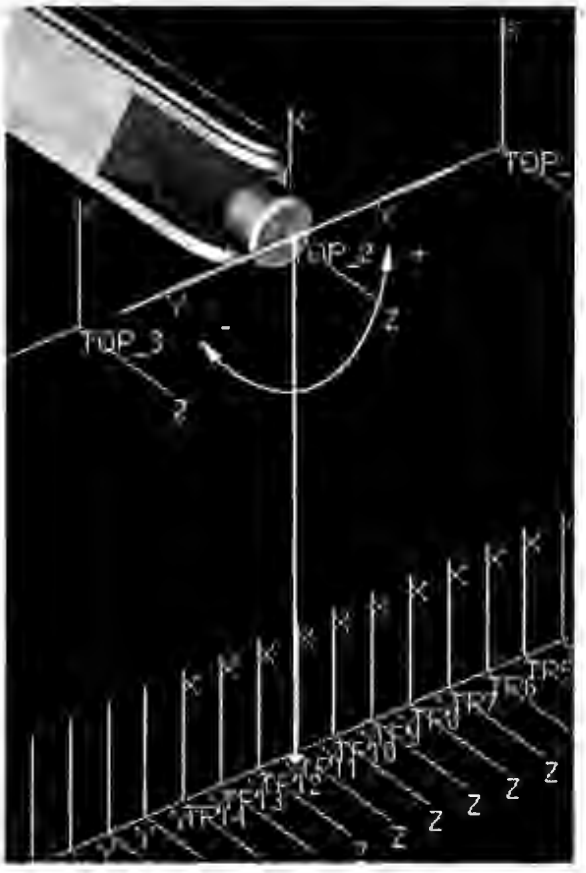


Figure 5.20 Incline Angle for Downward Motion

Figure 5.20 provides information about the sign of the incline angle for downward motion. The value of the incline angle can be found in the same way that the bearing angle value was found.

5.6. Chapter Summary

This chapter provided a formal description of the problem of inaccurate motion time estimate. Causes of the incorrect estimate were provided and a method for improvement of motion time accuracy was suggested. A detailed description of the testing procedure was given, too.

The next chapter will provide analysis of the results followed by the integration of correction factors into a motion model described in Chapter 4. Finally, a conclusion and suggestions for future work will be given.

Chapter 6

Analysis, Conclusion and Future Work

6.1. Analysis

Three tests were performed with a real robot – approach motion test, depart motion test and the downward motion test. For each of the motion tests, several bearing angle and incline angle values were tested and the corresponding motion times were recorded and analyzed. The results of the analysis revealed that the simulation motion time was longer than the motion time of the real robot in all the tests performed. The influence of the parameters listed in the section 5.3 was determined. Furthermore, the tests revealed the existence of unknown factors whose influence on the motion time of the real robot is significant.

6.1.1. Horizontal Motion Plane – Approach Motion

Four approach motion tests were performed during the experiment and the corresponding motion times were recorded. Based on the results of the experiment, the plots representing motion time curves of the real robot for different start teach-points (tables A.7, A.8, A.8, and A.10) were created. A minor difference among the motion time curves can be noticed. Considering that no parameter other than the distance of the start

teach-point from the base of the robot changed, the assumption made about the distance as an influential parameter can be considered as correct.

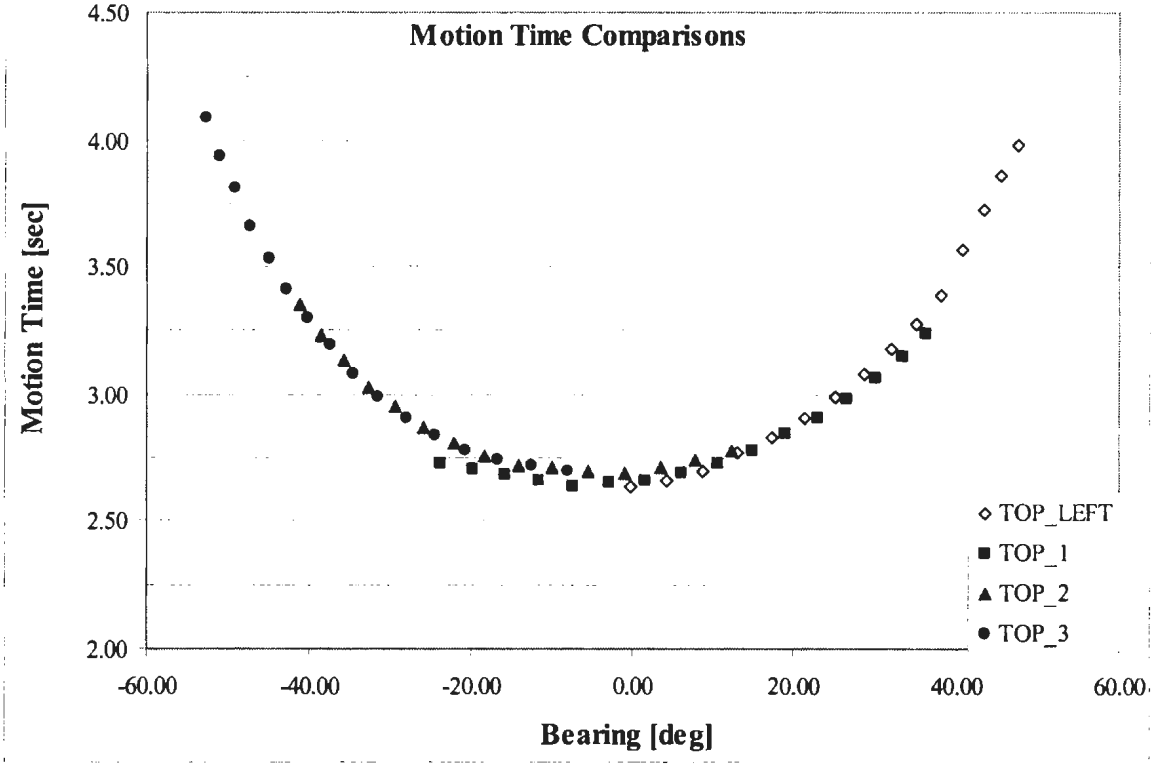


Figure 6.1 Motion Time Curves of the Real Robot

6.1.1.1. Start Teach-point "TOP_LEFT"

Figure 6.2 represents plots of motion time of the real robot and of the motion time produced by the simulation for start teach-point "TOP_LEFT". Both the plot on figure 6.2 and on figure 6.3 reveal that for the values of the bearing angle between 0 and 50 degrees

the difference between the motion time of the real robot and the simulation motion time decreases as the bearing increases, and that the decrease of the error suggests a linear trend.

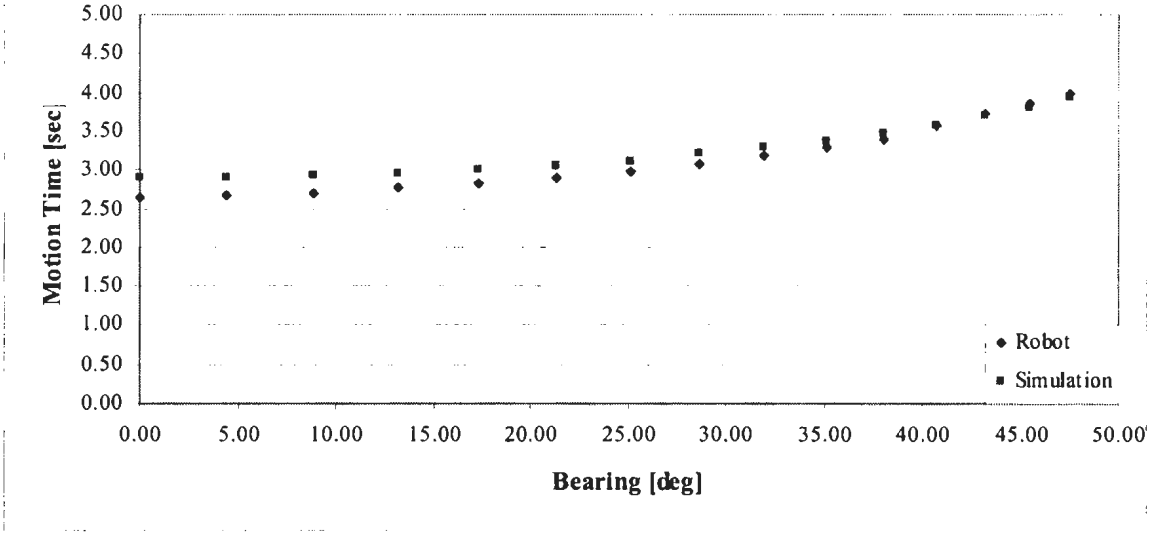


Figure 6.2 Motion Time Curves of the Start Teach-point "TOP_LEFT"

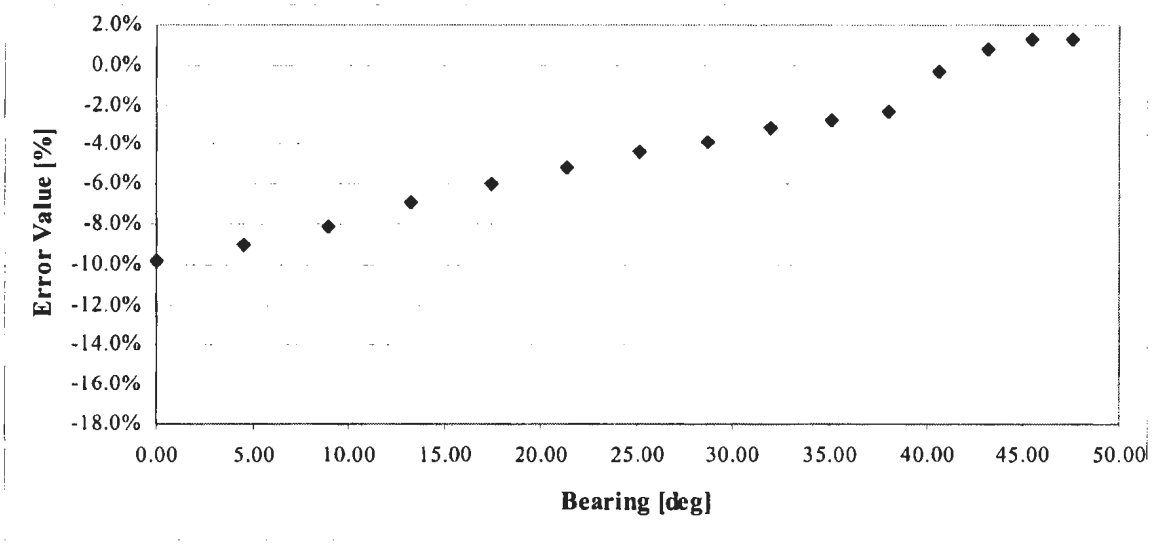


Figure 6.3 Error Plot for Start Teach-point "TOP_LEFT"

6.1.1.2. Start Teach-point "TOP_1"

Motion time plots given on figure 6.4 reveal a trend similar to the one given on figure 6.2. The error trend given on figure 6.5 reveals that with the increase of the bearing, the difference between the simulation motion time and the motion time of the real robot decreases almost linearly. Small bearing angle values, both positive and negative also mean short travel distances. By following that logic, a conclusion can be made that the error values should reach a minimum at a bearing angle value of zero degrees, which is not the case. The plot given on figure 6.5 shows an almost linear increase of error values for the range of bearing values between -30 degrees and 0 degrees. One possible reason for such behavior could be that there are factors other than the ones listed in the section 5.3 that influence motion of the real robot.

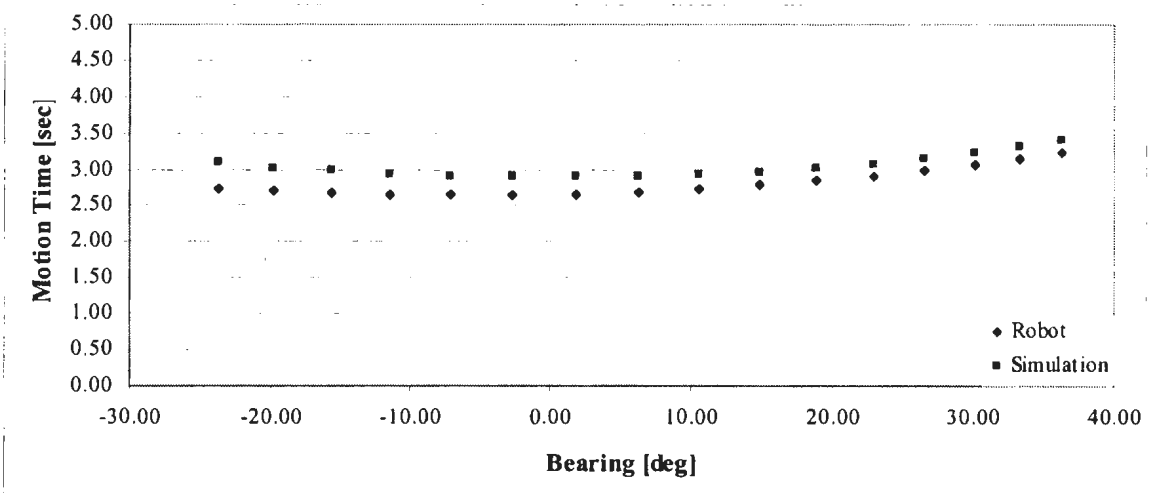


Figure 6.4 Motion Time Curves of the Start Teach-point "TOP_1"

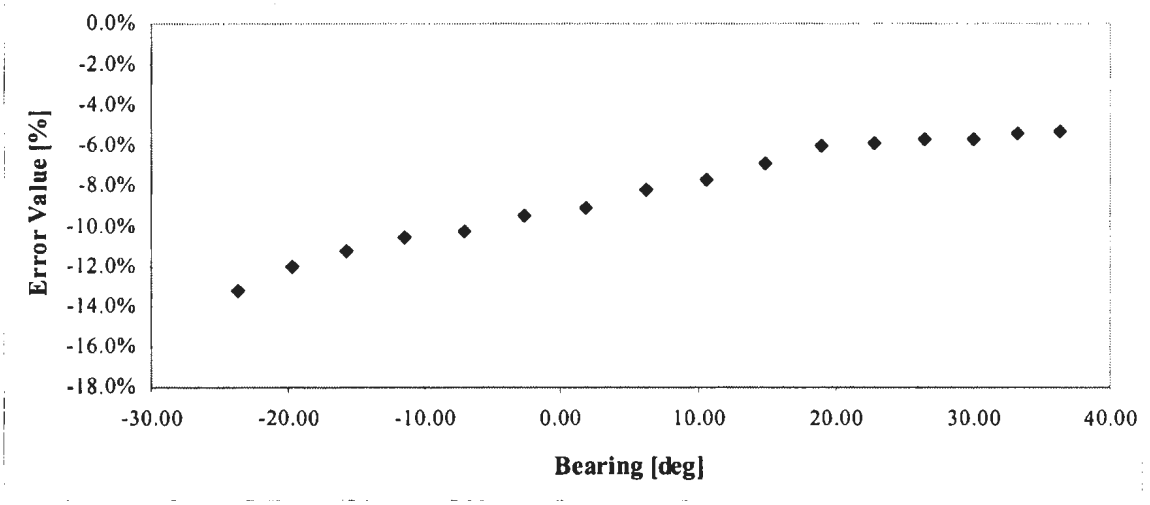


Figure 6.5 Error Plot for Start Teach-point "TOP_1"

6.1.1.3. Start Teach-points "TOP_2" and "TOP_3"

Motion that originates in teach-points "TOP_2" and "TOP_3" will be combined into one, since both teach-points are of an equal distance from the base of the robot. According to one of the original assumptions of the proposed method, error plots should be similar with respect to the trend and error values if the teach-points are an equal distance from the base frame of the robot. Based on the plots presented on figure 6.6 and on the figure 6.7 it can be concluded that the assumption made is valid for teach-points "TOP_2" and "TOP_3".

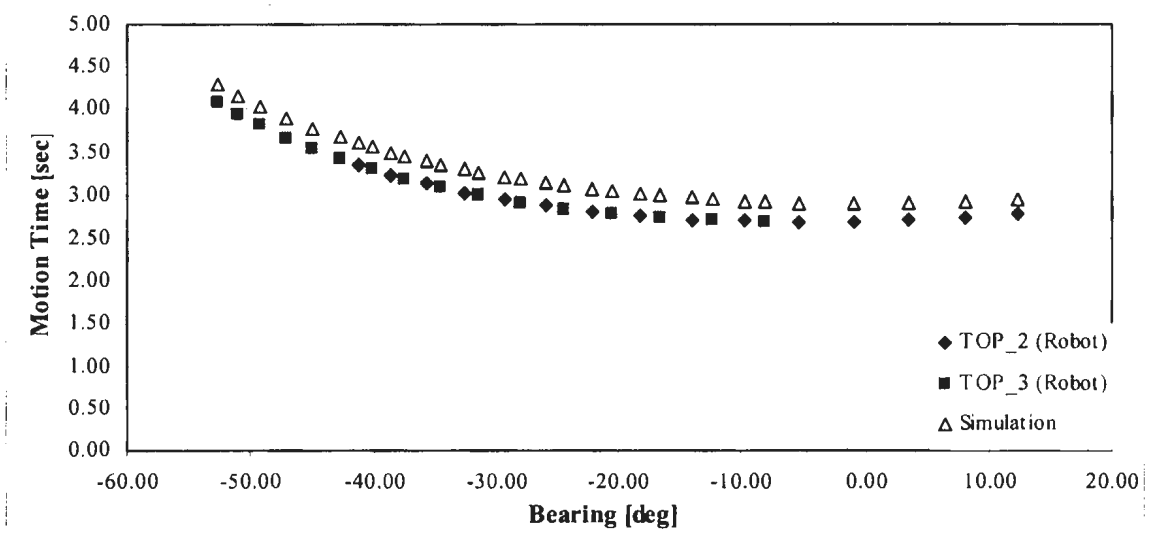


Figure 6.6 Motion Time Curves for Start Teach-point "TOP_2" and "TOP_3"

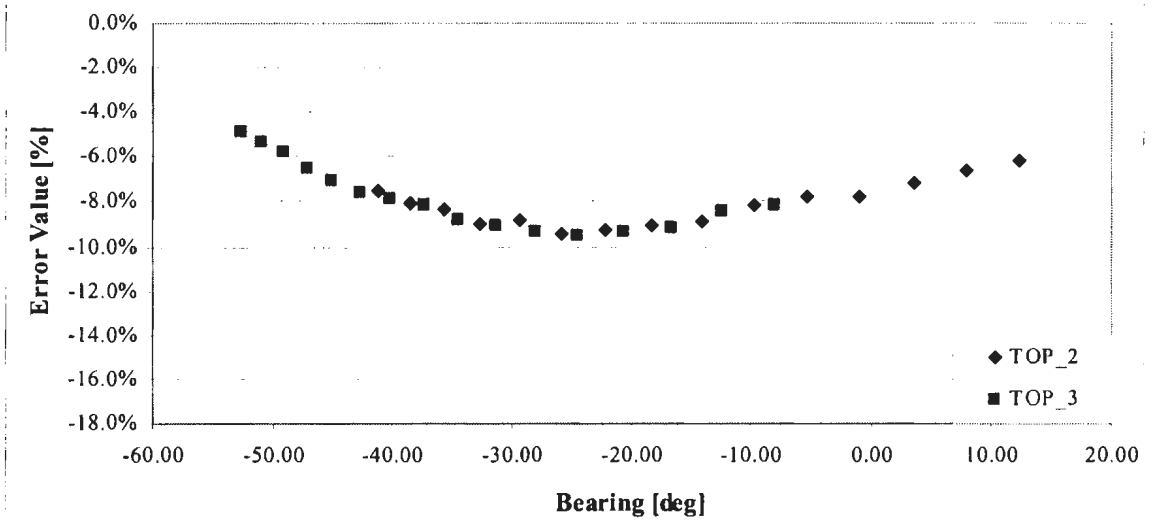


Figure 6.7 Error Plot for Start Teach-points "TOP_2" and "TOP_3"

6.1.2. Horizontal Motion Plane – Depart Motion

The plots given on figure 6.8 reveal that there is only a minor difference among the motion time curves. Considering that no other parameters changed except the distance of a start teach-point from the base of the robot, the assumption that distance is a factor that causes differences among the motion time values can be considered as valid for the teach-points tested.

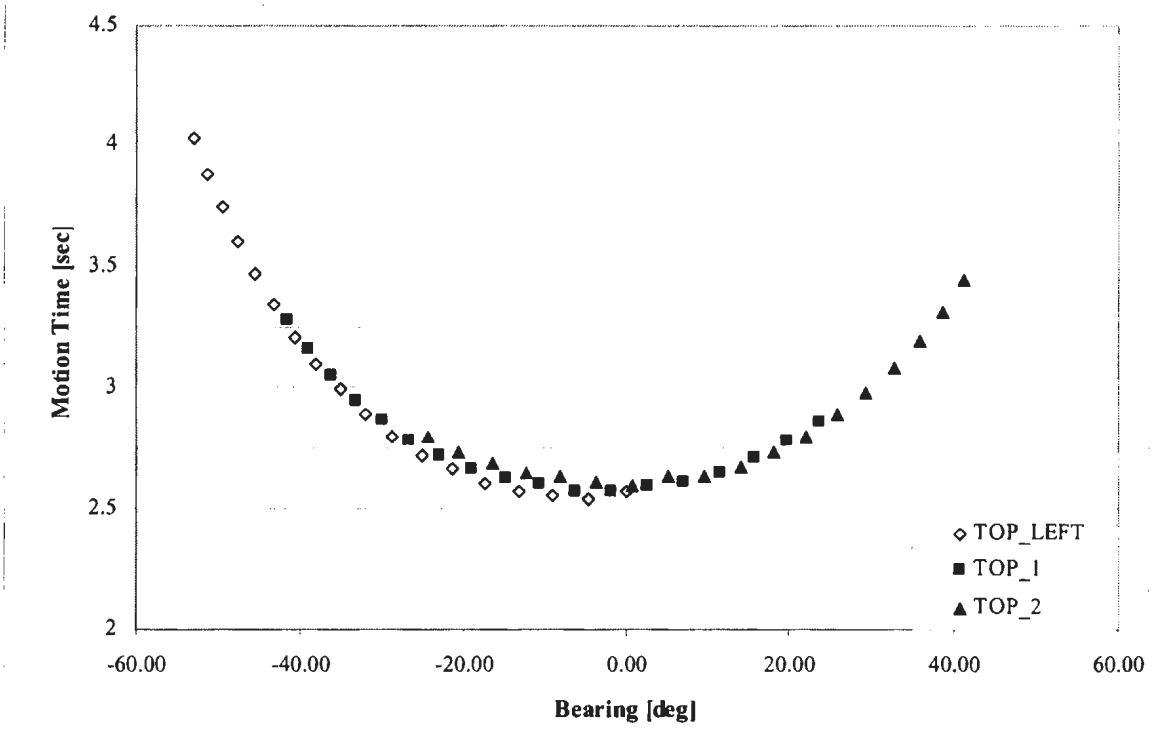


Figure 6.8 Motion Time Curves for Depart Motion.

Motion time curves are compared to the simulation motion time curve separately, because the distances of the teach-points "TOP_LEFT", "TOP_1" and "TOP_2" from the base frame of the robot are different.

6.1.2.1. Start Teach-point "TOP_LEFT"

The motion time curve of the real robot and the simulation are given on figure 6.9, while the error plot is given on figure 6.10. Clearly, the error value decreases as the value of the bearing angle increases both in the positive direction and in the negative direction. Large bearing angle values also mean longer travel distances. Considering that the largest error values were recorded neither for the shortest nor for the longest travel distances, but for the bearing angle values between -15 degrees and -20 degrees a conclusion can be made that there are factors other than the ones listed in section 5.3 that are influencing motion of the real robot.

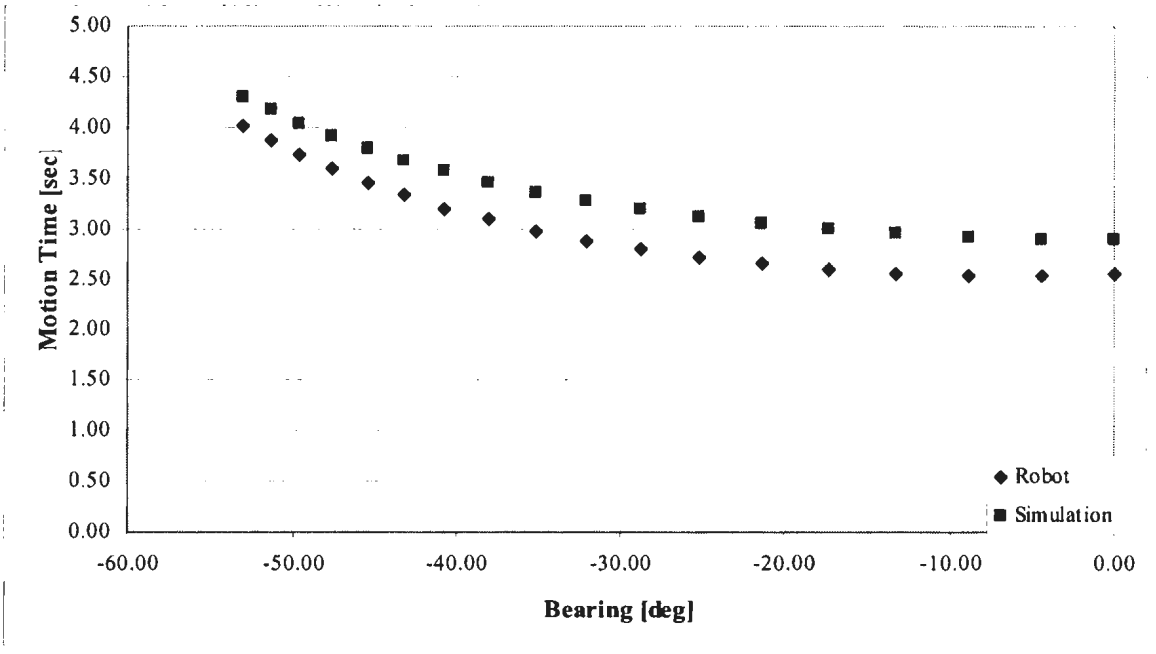


Figure 6.9 Motion Time Curve for Start Teach-point "TOP_LEFT"

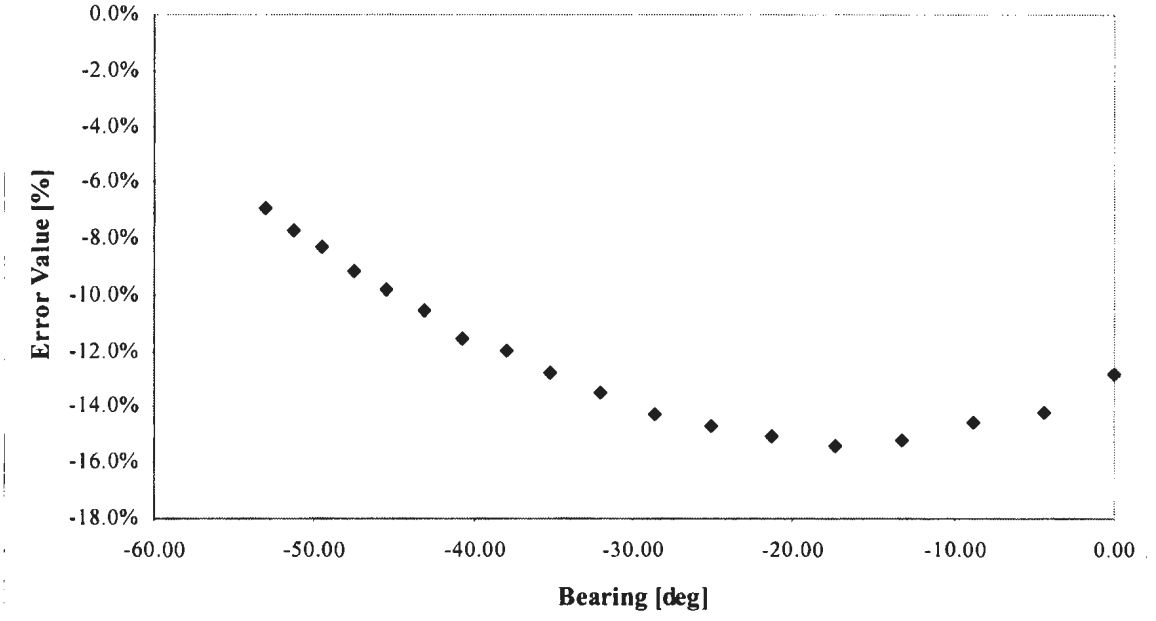


Figure 6.10 Error Plot for Start Teach-point "TOP_LEFT"

6.1.2.2. Start Teach-point "TOP_1"

Motion time curves of the real robot and of the simulation are given on figure 6.11, while the error plot is given on figure 6.12. Similarly to the case described in section 6.1.2.1, it can be assumed that the hidden factors caused the error values to reach a maximum for the bearing angle values of approximately -20 degrees.

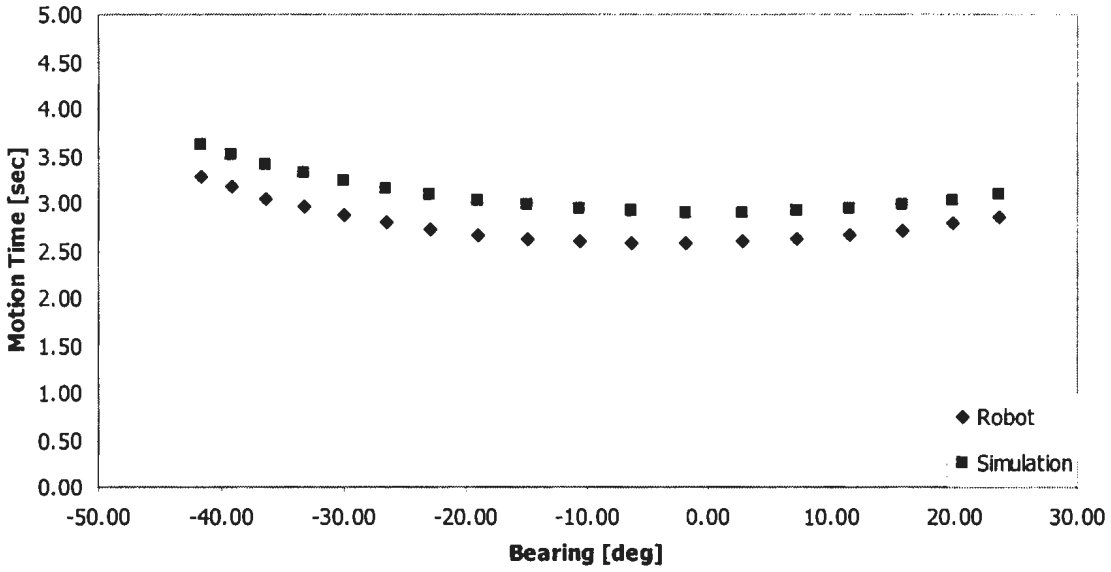


Figure 6.11 Motion Time Curve for Start Teach-point "TOP_1"

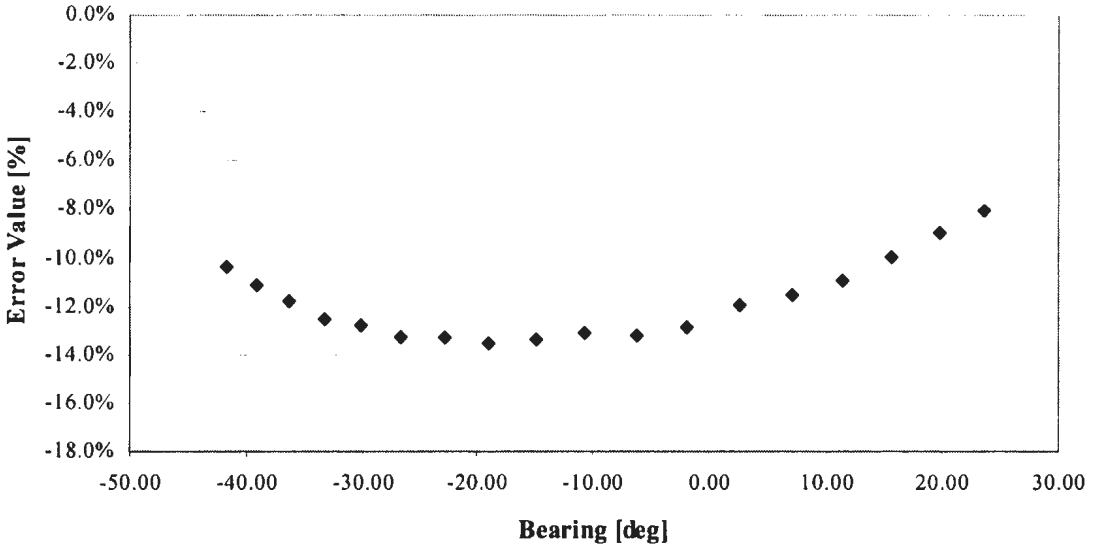


Figure 6.12 Error Plot for Start Teach-point "TOP_1"

6.1.2.3. Start Teach-point "TOP_2"

Motion time curves of the real robot and of the simulation are given on figure 6.13, while the error plot is given on figure 6.14. The error plot strongly suggests a linear decrease of error values for bearing values that range from +10 degrees to +40 degrees. Since large bearing values mean longer travel distances, one possible conclusion could be that the simulation motion model approximates motion of the real robot better for larger travel distances. However, the error value for bearing angles between -30 degrees and 0 degrees is almost constant, which suggests that the travel distance has little or no influence on error values. Similarly to the previous two cases described in sections 6.1.2.1 and 6.1.2.2, it can be assumed that there are other, unknown factors that are influencing motion of the real robot.

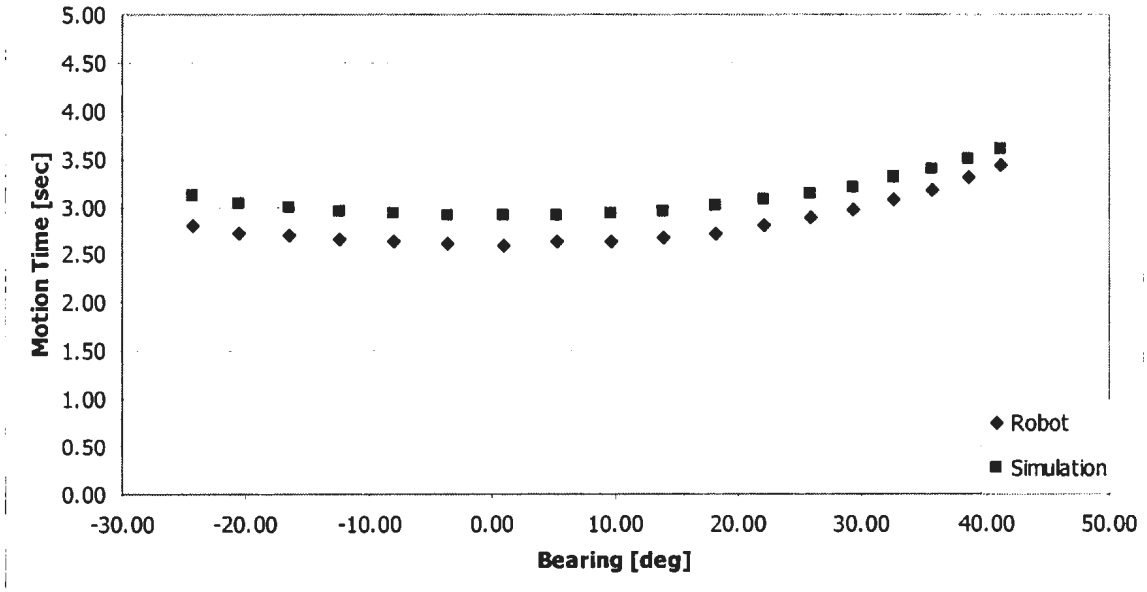


Figure 6.13 Motion Time Curve for Start Teach-point "TOP_2"

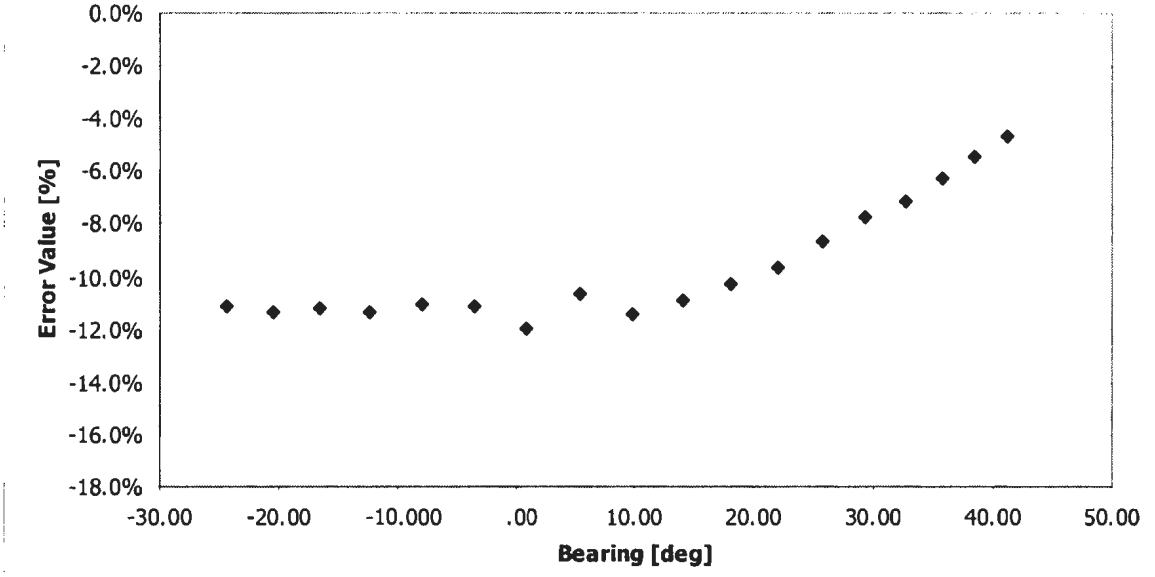


Figure 6.14 Error Plot for Start Teach-point "TOP_2"

6.1.3. Vertical Motion Plane – Downward Motion

Motion time curves for two different start teach-points "TOP_LEFT" and "TOP_2" are given on figure 6.15. Motion time curves reveal a slight difference in shape and trend, which can be explained by the fact that the teach-points have different distances from the base frame of the robot.

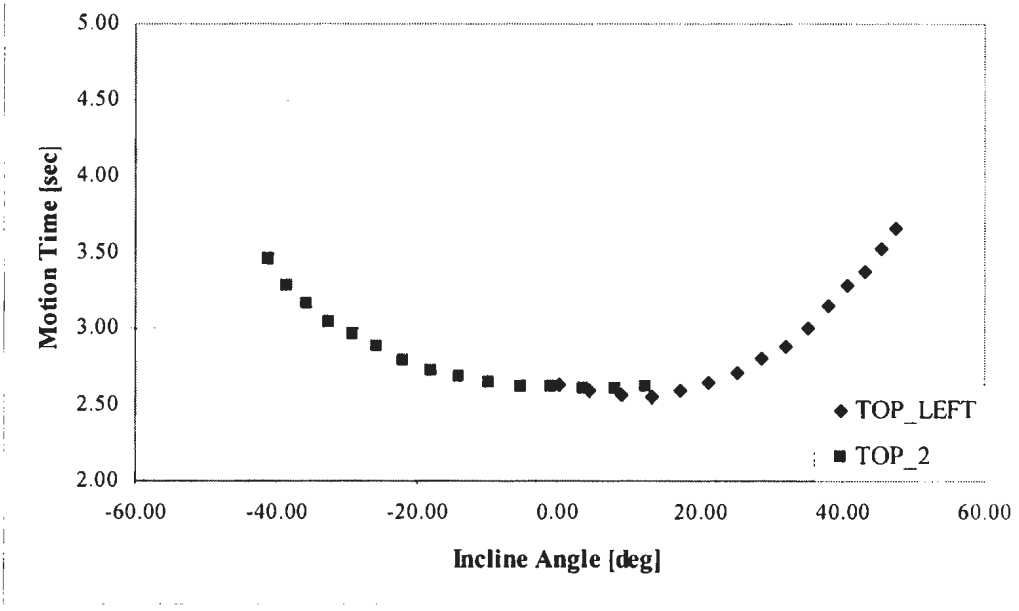


Figure 6.15 Motion Time Curves for Downward Motion

6.1.3.1 Start Teach-Point "TOP_LEFT"

Motion curve for teach-point "TOP_LEFT" presented on figure 6.16 shows the shortest motion time for the incline angle values of approximately 15 degrees. Since the small incline angle value also means a short travel distance, it could be expected that the shortest travel time would take place when the incline angle value equals zero degrees. The motion time curve for "TOP_LEFT" teach-point suggests that there are other, unknown factors that are influencing the motion of the real robot.

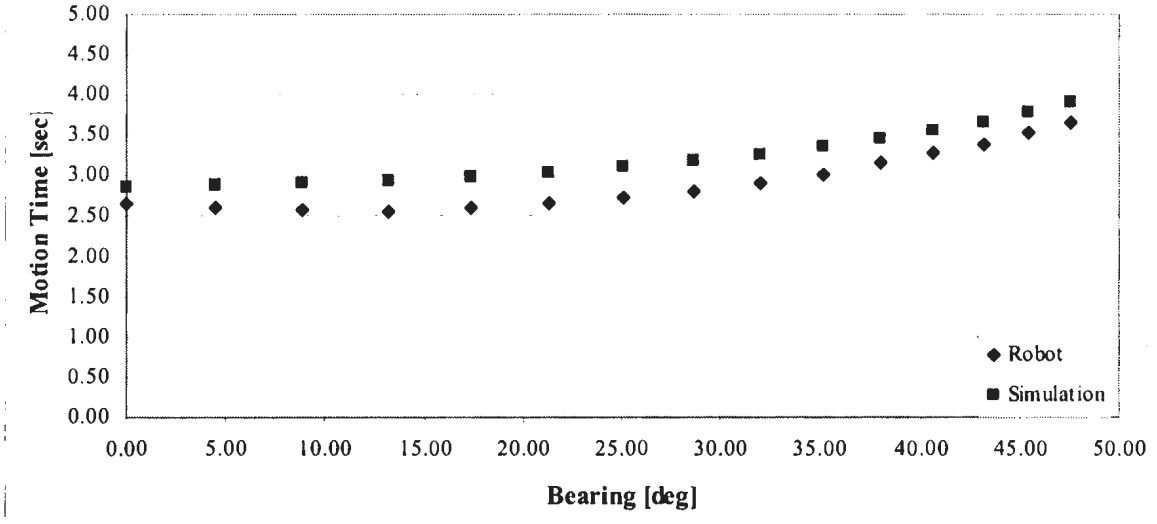


Figure 6.16 Motion Time Curve for Start Teach-point "TOP_LEFT"

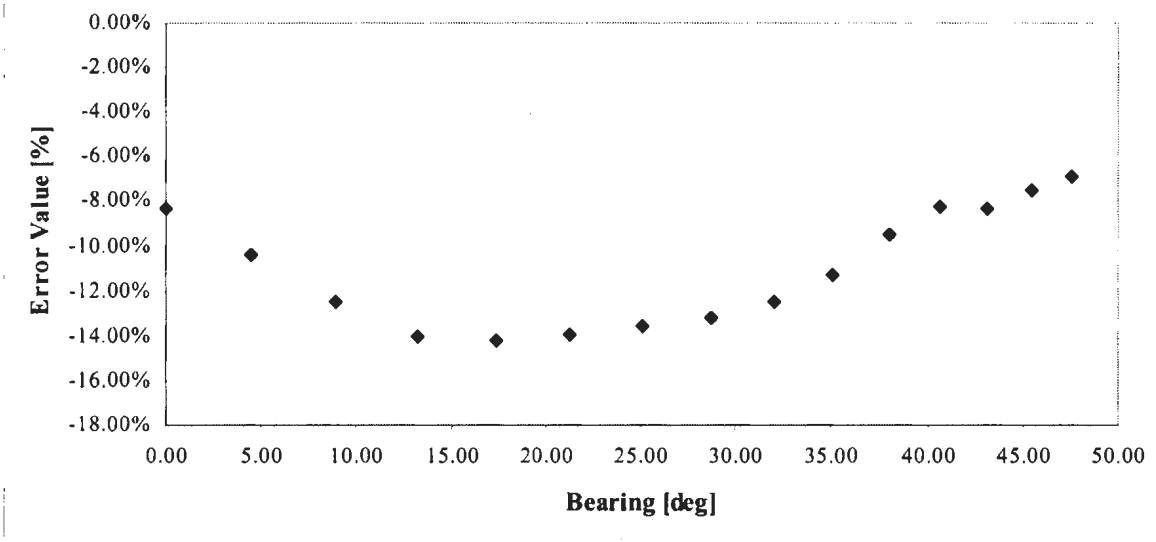


Figure 6.17 Error Plot for Start Teach-point "TOP_LEFT"

6.1.3.2. Start Teach-Point "TOP_2"

Motion time curves of the simulation and of the real robot are given on figure 6.18, while the error plot is given on figure 6.19. The error values decrease with a decrease in the incline angle value, which suggests a better approximation of the real robot's motion at higher negative values of the incline angle. However, the error plot also shows that for the range of incline angle values between 0 degrees and 20 degrees the error trend remains the same, which is opposite from what was expected. On the other hand, there is no information about the error values for incline angle values of 10 degrees and more, which means that the trend could change its direction like in case of the error plot given on figure 6.12 and figure 6.17.

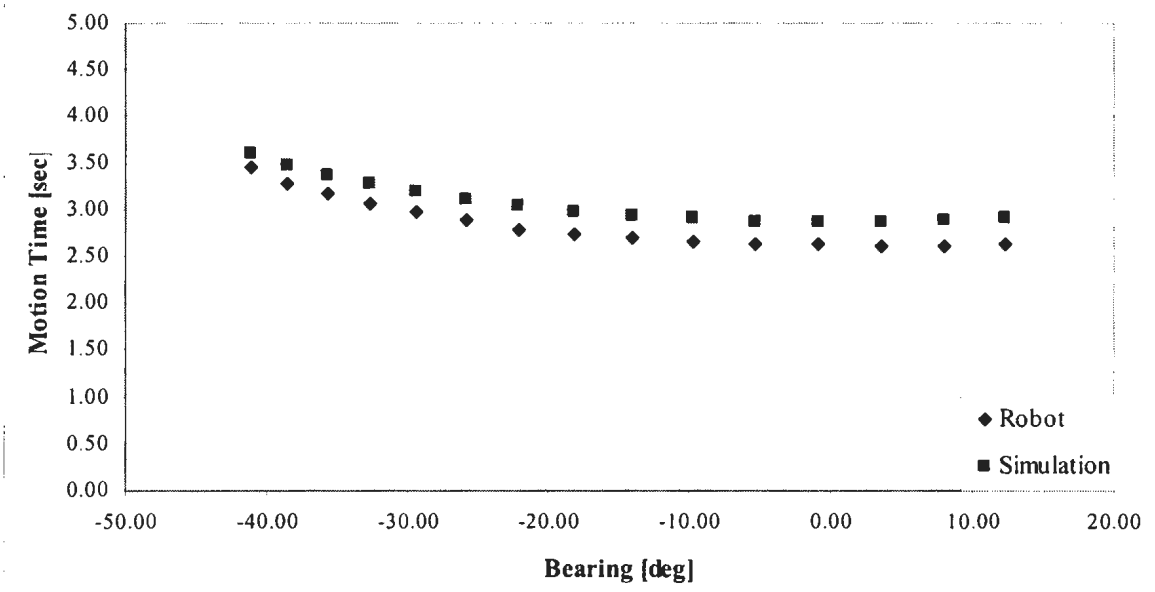


Figure 6.18 Motion Time Curve for Start Teach-point "TOP_2"

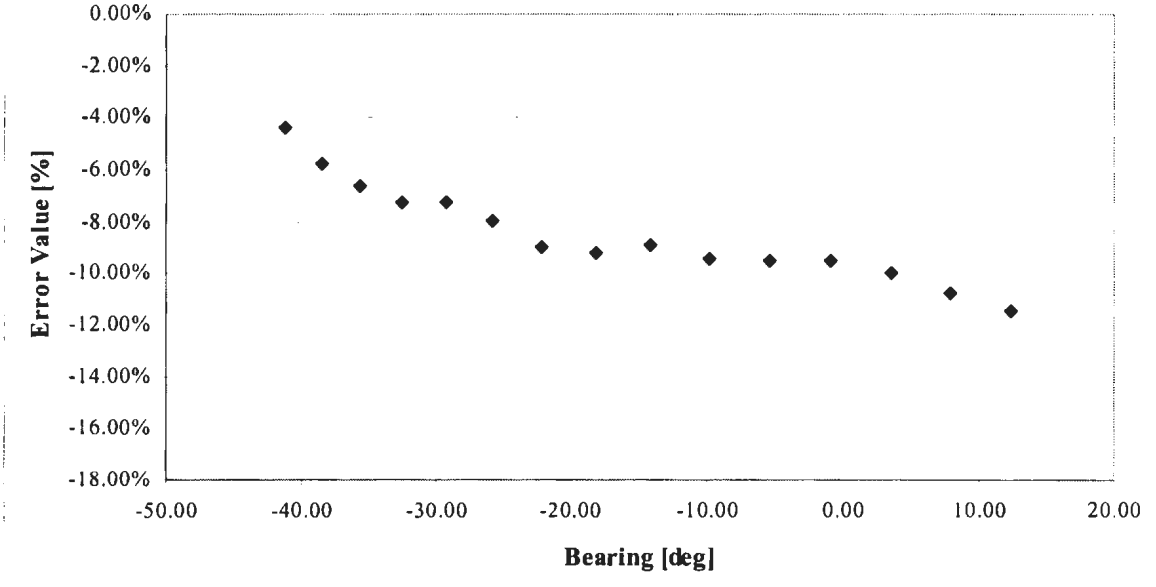


Figure 6.19 Error Plot for Start Teach-point "TOP_2"

6.2. Correction Factors

Motion plots presented in section 6.1 reveal a difference between the simulation motion model and the motion model of the real robot. In the case of an ideal robotic simulation that difference would not exist. However, no method including the RRS I Specification and the dynamics based motion model presented in previous chapters is able to provide simulation motion times that are identical to the motion time of the real robot.

The method proposed in the thesis assumes that through the integration of the correction factors into the simulation motion model it becomes possible to achieve simulation motion times identical to the motion times of the real robot for incline angle values or bearing values tested. There is one correction factor per influential parameter. In this particular case only two correction factors will be analyzed - one for the bearing and the other one for the incline angle.

A correction factor can be integrated into the simulation motion model through one of the parameters that define motion models described in chapter 4:

1) Distance between the start point and the target point,
2) Maximum velocity that is to be reached during motion between the two points,
3) Acceleration/deceleration

A change of distance is unacceptable, because it results in an incorrect trajectory. A change of velocity is also unacceptable, because maximum velocity is the key parameter for actions such as welding or painting. The only parameter that can be changed is acceleration. Although important in the overall motion calculation process, the

acceleration value is not a key parameter in most of the processes performed by the robot. Therefore, acceleration is a parameter that is going to be used for modification of the simulation motion models.

Derivation of correction factors is based on the assumption that after the correction, simulation motion time will become equal to the motion time of the real robot. Adjusted acceleration of the simulation motion model that results in the motion time equal to the average experimental motion time of the real robot can be found as:

$$a_{adjusted} = \frac{v_{max}^2}{T \cdot v_{max} - L}, \tag{6.1}$$

where:

T [s]	motion time (experiment),
v_{max} [mm/s]	maximum velocity (simulation),
L [mm]	distance between the start teach-point and end teach-point.

The value of the adjusted acceleration can be either larger or smaller than the nominal value of acceleration. Consequently, the maximum velocity of linear motion in a simulation will be reached either faster or slower than it would be reached in the case of nominal acceleration (figure 6.20).

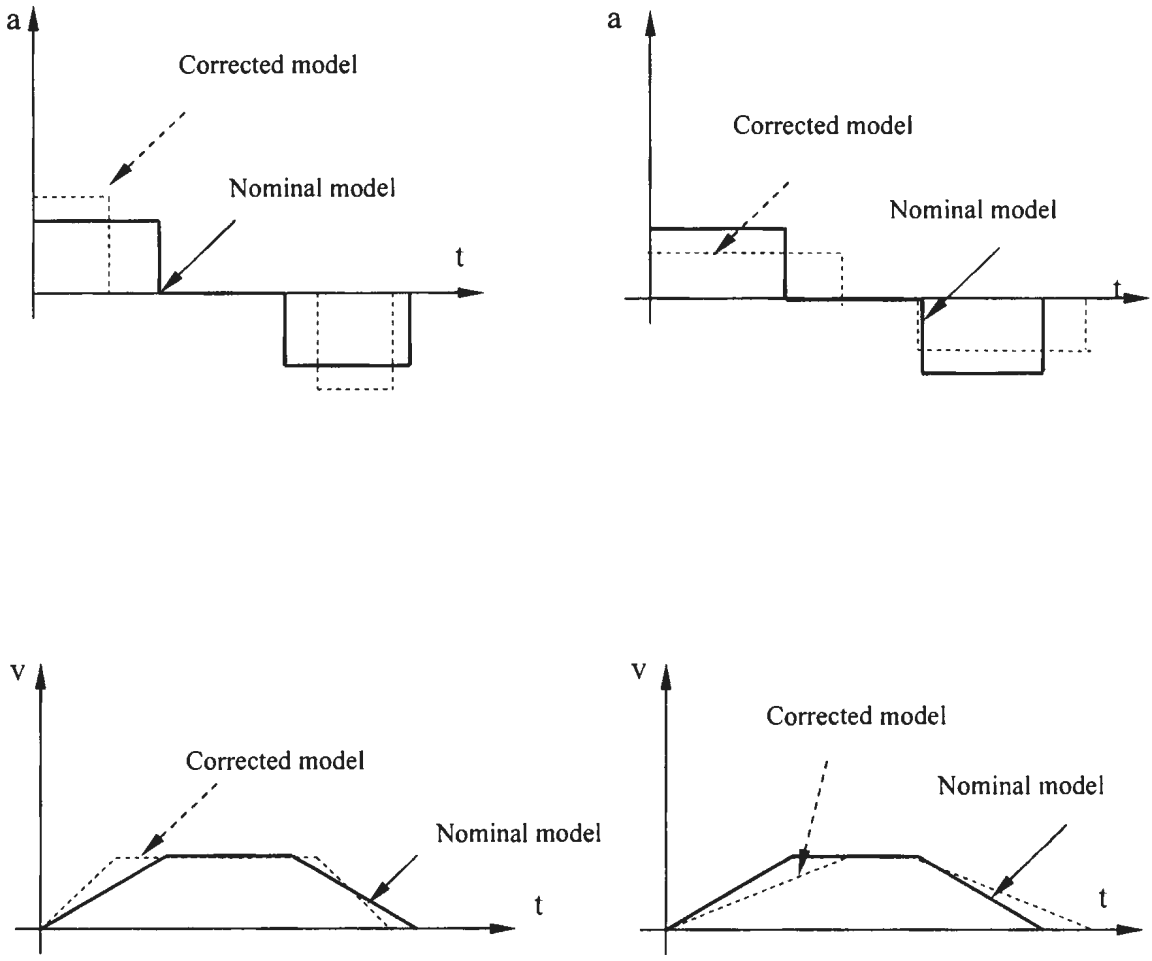


Figure 6.20 Constant Acceleration Models – Nominal model and Corrected model

6.2.1. Correction Factor for Horizontal Motion

The correction factor value for a horizontal motion plane can be found as a ratio:

$$C_H = \frac{a_{adjusted}}{a_{nominal}} \tag{6.2}$$

For example, when motion originates from the point TOP_LEFT (coordinates given in the table A.1) and when bearing angle value is zero degrees, the value of acceleration that a simulation motion model must have in order to achieve the same motion time as the real robot is calculated using equation 6.1 and is 592.11 mm/s^2. The correction factor in this case is calculated as:

$$C_H = \frac{a_{adjusted}}{a_{nominal}} = \frac{592.11}{400.0} = 1.48$$

Similarly, the correction factor value can be calculated for any combination of the bearing angle value and the distance of a teach-point from the base of the robot. Based on the results given in table A.16 and A.17, plots given on figures 6.21 and 6.22 were created. Both the figure 6.21 and the figure 6.22 show that the correction factor curves for different teach-points do differ. That outcome was expected considering that the factors associated with the teach-points, such as the distance from the base of the robot are different. In the case of teach-points TOP_2 and TOP_3, which are equally distant from the base of the robot, the correction factor curves show a similarity of values and trends, which was expected.

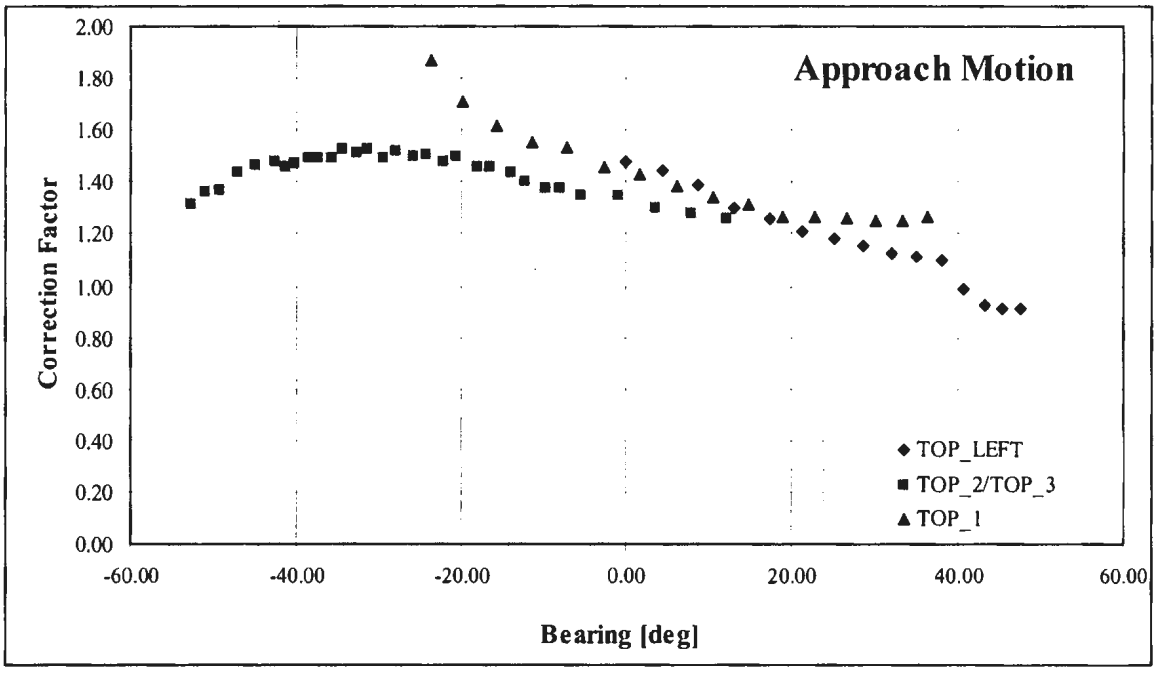


Figure 6.21 Correction Factor Curves for Approach Motion

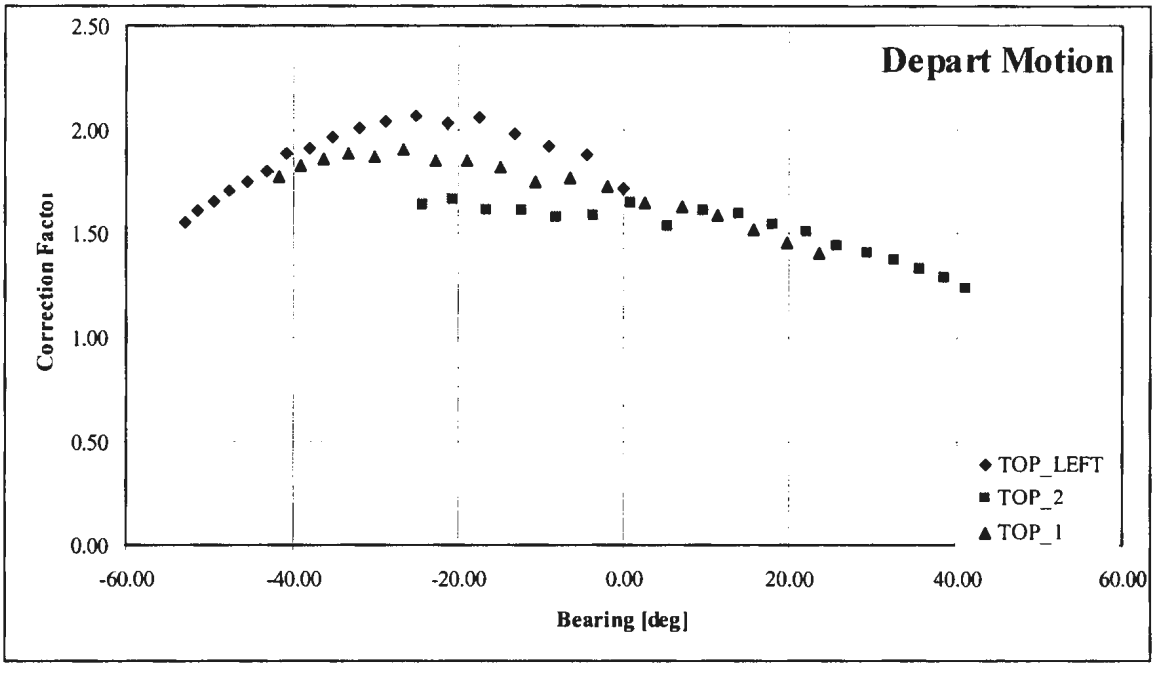


Figure 6.22 Correction Factor Curves for Depart Motion

Values of the correction factor for bearing angle values other than the ones given in the table 6.5 can be found by using the law of proportion:

$$C_H(\alpha) = C_{H,\alpha_{lower}} + \frac{\alpha - \alpha_{lower}}{\alpha_{upper} - \alpha_{lower}}(C_{H,\alpha_{upper}} - C_{H,\alpha_{lower}}), \qquad (6.3)$$

where:

$C_H(\alpha)$ correction factor value for bearing α,

α_{lower} lower limit of the bearing range to which α belongs,

α_{upper} upper limit of the bearing range to which α belongs,

$C_{V,\alpha_{lower}}$ correction factor value for the bearing α_{lower},

$C_{H,\alpha_{upper}}$ correction factor value for the bearing α_{upper}.

For example, for approach motion that originates from "TOP_LEFT" teach-point having the value of bearing angle of 7 degrees, the corresponding value of the correction factor C_H will be:

$$C_H = 1.44 + \frac{7 - 4.47}{8.88 - 4.47}(1.39 - 1.44) = 1.411$$

6.2.2. Correction Factor for Vertical Motion

The correction factor for vertical motion can be found in exactly the same manner that the correction factor for horizontal motion was found – as a ratio of adjusted and nominal value of acceleration.

$$C_V = \frac{a_{adjusted}}{a_{nominal}}, \tag{6.4}$$

where:

$a_{adjusted}$ – acceleration value that results in the simulation motion time equal to the motion time of the real robot,

$a_{nominal}$ – nominal value of acceleration used in the simulation motion model. Typically, $a_{nominal}$ is a value set by the user.

Using equation 6.1, the correction factor values C_V can found for motion in the vertical plane. Correction factor values are presented in the table A.18, while the corresponding graph is presented on figure 6.23.

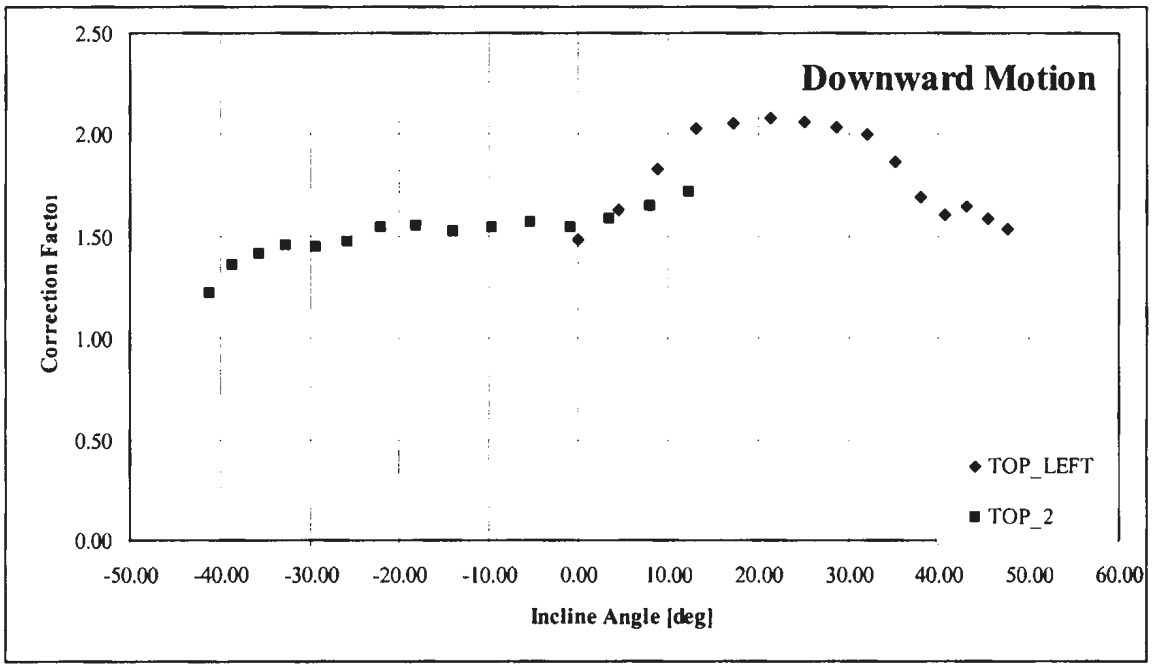


Figure 6.23 Correction Factor Curves for Downward Motion

Correction factor curves differ in shape and value, which is expected since the influential factors associated to the two teach-points tested are different.

6.3. Conclusion

Based on the results of the experiment motion time curves of the real robot and of the simulation were created. Simulation motion time curves and the corresponding motion time curves of the real robot show similarities in trend, which means that the simulation motion model does resemble the motion model of the real robot. Motion time curves also show that the bearing and the incline angle do influence motion time of the real robot. The error plots revealed that the error values in some cases reached 15%, which means that approximation of the real robot's motion model is not close enough and that the level of influence that bearing and the incline angle have on real robot's motion time is significant.

Furthermore, all of the motion time plots show that besides the influential parameters tested in the experiment, there is a set of unknown parameters significantly influencing the motion of the real robot. Identification of the hidden parameters themselves is of no importance for the method proposed in the thesis. It is their influence that is important and that can be identified through the parameters such as bearing and incline angle, whose influence can be established easily.

Integration of the influence of both known and unknown parameters is done through a set of correction factors. The values of the correction factors presented earlier in this chapter range between 1 and 2. In other words, the value of acceleration used in the simulation sometimes needs to be almost twice as large as the nominal value set by the user. The increased value of acceleration results in shorter amount of time spent

moving between the teach-points, which means a shorter overall travel time. All motion time plots presented show that the simulation motion time is longer than the motion time of the real robot, thus by increasing the value of acceleration used in the simulation the overall simulation motion time becomes shorter. Furthermore, the simulation motion time between the two teach-points would be shortened to the exact motion time of the real robot, thus making the simulation motion time accurate.

Values of the correction factors can be established by using the error plots similar to the ones presented earlier in this chapter. Once integrated into the simulation motion model, the correction factors should make the motion time curves of the real robot and of the simulation overlap. Although the approximation is better if the number of teach-points tested is larger, testing does not have to include teach points throughout the whole work envelope of the robot. Instead, the tests can be made only in the part of the robot's envelope in which the task will be performed.

Another important aspect of the proposed method is the introduction of robot dynamics in the calculation process through the correction factors. Correction factors are robot-specific parameters. Their values depend not only on the factors already mentioned, but also on the factors that have not been mentioned – such as the mechanical structure of a robot, tool geometry, friction and torque characteristics of servo-drives. Through correction factors the influence of those "hidden" parameters or hard to identify parameters are incorporated into the existing, simple kinematics model, thus bringing more realism into the simulated motion time.

6.4 Future Work

Future work has two basic goals with respect to the time frame: an immediate goal and a long-term goal. The immediate goal includes verification of the results presented in this thesis, while the long-term goal includes expansion of correction formula on other influential parameters.

6.4.1. Immediate Goal

The immediate goal includes verification of the proposed method on an inclined plane, i.e. on a plane that would include both vertical and horizontal motion. Derivation of the integration formula follows.

Integration of acceleration correction factors is based on the two parameters – the direction of motion and the value of linear acceleration. Direction of linear motion can be found by using start and end point coordinates:

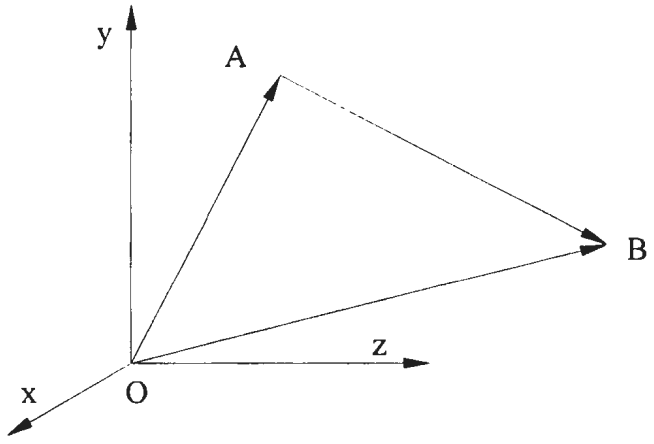


Figure 6.13 Motion Direction Derivation Using Basic Vector Calculus

Vector calculations are given by the following equations:

$$\overrightarrow{AB} = \overrightarrow{OB} - \overrightarrow{OA} = (x_B - x_A)\vec{i} + (y_B - y_A)\vec{j} + (z_B - z_A)\vec{k} \tag{6.5}$$

Angles that vector $\overrightarrow{AB}$ makes with the reference coordinate axes can be found as:

$$\cos\alpha_X = \frac{x_B - x_A}{\sqrt{(x_B - x_A)^2 + (y_B - y_A)^2 + (z_B - z_A)^2}} \tag{6.6}$$

$$\cos\alpha_y = \frac{y_B - y_A}{\sqrt{(x_B - x_A)^2 + (y_B - y_A)^2 + (z_B - z_A)^2}} \tag{6.7}$$

$$\cos\alpha_z = \frac{z_B - z_A}{\sqrt{(x_B - x_A)^2 + (y_B - y_A)^2 + (z_B - z_A)^2}} \tag{6.8}$$

A vector of linear acceleration is directed along the vector $\overrightarrow{AB}$, and it too can be decomposed into three components a_x, a_y and a_z using angles calculated in equations 6.6, 6.7 and 6.8:

$$\vec{a} = a_x\vec{i} + a_y\vec{j} + a_z\vec{k} \tag{6.9}$$

$$a_x = a \cdot \cos\alpha_x \tag{6.10}$$

$$a_y = a \cdot \cos\alpha_y \tag{6.11}$$

$$a_z = a \cdot \cos\alpha_z \tag{6.12}$$

Acceleration value in the horizontal plane can be found as:

$$a_H = \sqrt{a_x^2 + a_y^2} \tag{6.13}$$

Acceleration value in the vertical plane can be found as:

$$a_V = \sqrt{a_y^2 + a_z^2} \tag{6.14}$$

Now, when acceleration values of both horizontal and vertical plane motion are known, the correction calculation can take place. The result of the correction is the following set of equations:

$$a_{H,corr} = C_H \cdot a_H = C_H \sqrt{a_x^2 + a_y^2} \tag{6.15}$$

Similarly:

$$a_{V,corr} = C_V \cdot a_V = C_V \sqrt{a_y^2 + a_z^2} \tag{6.16}$$

Corrected acceleration values have to be integrated back into the analytical model. In order to do so, equation 6.9 needs to be written in a different format:

$$a^2 = a_x^2 + a_y^2 + a_z^2 \tag{6.17}$$

The equation can be derived further:

$$a^2 = a_x^2 + a_y^2 + a_y^2 + a_z^2 - a_y^2$$

$$a^2 = a_H^2 + a_V^2 - a_y^2$$

By replacing a_H and a_V with equations 6.15 and 6.16, a corrected overall linear acceleration value can be found as:

$$a_{corr}^2 = C_H^2 a_x^2 + C_H^2 a_y^2 + C_V^2 a_y^2 + C_V^2 a_z^2 - a_y^2$$

$$a_{corr}^2 = C_H^2 a_x^2 + (C_H^2 + C_V^2 - 1) \cdot a_y^2 + C_V^2 a_z^2 \tag{6.18}$$

Once the corrected value of linear acceleration a_{corr} is known, it can be used for calculations that are described in the chapter 4.

6.4.2. Long-Term Goals

There are three goals of future work that require a significant amount of time, thus can be considered as long-term goals:

1) The experiments described in this chapter determined the influence that motion direction has on motion time of the real robot. Parameters such as the mass of the tool or manipulated object, configuration of the robot, type of motion interpolation, as well as the radial distance of both the vertical and horizontal plane were kept constant during the experiment. Future work should be focused on identification of influence the mentioned parameters have on motion time.
2) Functional dependency of the corrected acceleration value should be expressed in a simpler manner than what is described in equation 6.18. Ideally, a_{corr} could be expressed in the following format:

$$a_{corr} = a \cdot C_V \cdot C_H \cdot C_m \ldots \cdot C_n \tag{6.19}$$

Where:

- a – nominal acceleration,
- C_V – correction factor for vertical motion plane
- C_H – correction factor for horizontal motion plane
- C_m – correction factor for mass compensation,
- C_n – n-th influential factor

3) Ultimately, if further tests confirm the results of the test made for the research purposes of this thesis, a standard method similar to the process of static robot calibration could be developed. The result of the method's application would be the "signature" of the tested robot model. The "signature" would represent nothing but the description of the functional relationship between the correction parameters and motion time. A functional relationship could be provided either by the robot manufacturers or established by the users themselves through a set of simple standardized tests like the ones described in this chapter. Once known, a "signature" would be imported into the simulation system and applied to the underlying motion model of the simulation.

References

[1] Jerry Banks. "Principles of Simulation". In Jerry Banks (Ed.) *Handbook of Simulation*, John Willey & Sons, Inc. 1998.

[2] Onur Ulgen, Ali Gunal. "Simulation in the Automobile Industry". In Jerry Banks (Ed.) *Handbook of Simulation*, John Wiley & Sons, Inc. 1998.

[3] Y. F. Yong, M. C. Bonney. "Off-line Programming". In Shimon Y. Nof (Ed.) *Handbook of Industrial Robotics, Second Edition*, John Wiley & Sons, Inc. 1999.

[4] Bryan Greenway. "Robot Accuracy". *Industrial Robot: An International Journal*, Vol. 27, Number 4, pp. 257-265, 2000.

[5] John Owens. "Robot Simulation - Seeing the Whole Picture". *Industrial Robot*, Vol. 18, No. 4. pp. 10-12, 1991.

[6] R. Bernhardt, G. Schreck, C. Willnow. "Realistic Robot Simulation". *Computing and Control Engineering Journal*, pp. 174-176, August 1995.

[7] Jacob Rabinovitz. "CAD and Graphic Simulators/Emulators of Robotic Systems". In Shimon Y. Nof (Ed.) *Handbook of Industrial Robotics, Second Edition*, John Wiley & Sons, Inc. 1999.

[8] RRS Interface Specification, Version 1.3, September 23, 1997.

[9] Jack Hollingum. "Simulation, Calibration and Offline Programming". *Industrial Robot*, Volume 21, Number 5, pp. 20-21, 1994.

[10] Rolf Bernhardt. "Approaches for Commissioning Time Reduction". *Industrial Robot*, Volume 24, Number 1, pp.62-71, 1997.

[11] RRS II - Reference Group Meeting Notes, June 13, 2000, IPK Berlin

[12] Presented Slides - RRS II Reference Group Meeting at IPK Fraunhofer, Berlin, June 13, 2000.

[13] Merriam-Webster Dictionary. Homepage. 17 July 2001. <http://www.m-w.com>

[14] Russel Stringham. "Simulation Tools Ease and Speed Assembly Cell Development". *Assembly Automation*, Volume 19, Number 2, pp.121-125, 1999.

[15] David Bradley. "Concurrent Engineering for Bespoke Products". *Assembly Automation*, Volume 15, Number 1, pp.35-37, 1995.

[16] Brian Rooks. "A Shorter Product Development Time with Digital Mock-Up". *Assembly Automation*, Volume 18, Number 1, pp. 34-38, 1998.

[17] Paul G. Ranky. "Concurrent Engineering and Enterprise Modeling". *Assembly Automation*, Volume 14, Number 3, pp.14-21, 1994.

[18] Tecnomatix Corporation. Homepage. 17 July 2001. <http://www.tecnomatix.com>

[19] Gunter Wittenberg."Training with Virtual Reality". *Assembly Automation*, Volume 15, Number 3, pp. 12-14, 1995.

[20] John Craig. "Past, Present and Future". *Robotics World*, pp. 40-41. November/December 1999.

[21] Brian Rooks. "Tecnomatix Weaves its e-Manufacturing Web". *Assembly Automation*, Volume 20, Number 3, pp.213 - 216, 2000.

[22] Gutter Wittenberg. "Development in Offline Programming: An Overview". *Industrial Robot*, Volume 22, Number 3, pp. 21-23, 1995.

[23] Delmia Corporation. Homepage. 17 July 2001. <http://www.delmia.com>

[24] Anna Kochan. "Introducing Coherence in Virtual Manufacturing". *Automotive Manufacturing Solutions*, Volume 1, Number 1, pp. 70-73, 2000.

[25] Gilad Lederer. "Making Virtual Manufacturing a Reality". *Industrial Robot*, Volume 22, Number 4, pp. 16-17, 1995

[26] Paul Abbott, John Owens. "Simulation and Offline Programming in the Real World". *Robotics World*, pp. 40-43, Fall 1998.

[27] Flow Software Technologies. Homepage. 17 July 2001. <http://www.workspace5.com>

[28] Workspace (Version 5.01 for Windows) [Computer software], 2001

[29] Austin Weber. "Virtual Tools Optimize Assembly". *Assembly*, pp.44-52, April 2000.

[30] Anna Kochan. "Rover's E-Process Assembles Cars in Virtual World". *Assembly Automation*, Volume 19, Number 2, 1999, pp.118-120.

[31] Michael P. Deisenroth, Krishna K. Krishnan. "On-line Programming". In Shimon Y. Nof (Ed.) *Handbook of Industrial Robotics, Second Edition*, John Wiley & Sons, Inc. 1999.

[32] Brian W. Rooks. "Off-line Programming: A Success for Automotive Industry". *Industrial Robot*, Volume 24, Number 1, pp.30-34, 1997.

[33] Anna Kochan. "Land Rover Uses Jack and ROBCAD to Perfect Production Processes". *Assembly Automation*, Volume 18, Number 2, pp.129-131, 1998.

[34] Narinder Nayar. "Workspace Ergonomics and Simulation". *Assembly Automation*, Volume 16, Number 1, pp.25-28, 1996.

[35] S. Zeghloul, B. Blanchard, M. Ayrault. "SMAR: A Robot Modeling and Simulation System". *Robotica*, Volume 15, pp.63-73, 1997.

[36] "ACIS Interoperability Components". Home page. Spatial Corporation. 11 Jul. 2001 <http://www.spatial.com/products/Interop/ACIS_interop.htm>

[37] "ACIS Healing". Home page. Spatial Corporation. 11 Jul. 2001 <http://www.spatial.com/products/Interop/healing.htm>

[38] "Data Exchange White Paper". Home page. Theorem Solutions Ltd. 11 Jul. 2001 <http://www.theorem.co.uk/docs/whitep.htm>

[39] Andrew A. Goldenberg, Mohammad, R. Emami. "Kinematics and Dynamics of Robot Manipulators". In Shimon Y. Nof (Ed.). *Handbook of Industrial Robotics, Second Edition*, John Wiley & Sons, Inc. 1999.

[40] L. Sciavicco, B. Siciliano. *Modeling and Control of Robot Manipulators*, The Mc Graw Hill Companies, Inc, 1996.

[41] K. S. Fu, C. S. Lee, R. Gonzales. *Robotics: Control, Sensing, Vision & Intelligence*, McGraw-Hill Ryerson, 1987.

[42] John Owens. "Task planning in Robot Simulation". *Industrial Robot*, Volume 23, Number 5, pp. 21-24, 1996.

[43] R. Bernhardt, S.L. Albright. *Robot Calibration*, Chapman & Hall, 1993.

[44] Tsuneo Yoshikawa. *Foundation of Robotics - Analysis and Control*. MIT Press, 1990.

[45] Brady M, Hollerbach J, Johnson T, Lozano-Perez T, Mason M. *Robot Motion: Planning and Control*, MIT Press, 1982.

[46] Everett J. L. "Research Topics in Robot Calibration", In R Bernhardt (Ed.) *Robot Calibration*, Chapman & Hall, 1993.

[47] "Motoman UP20". Home page. Motoman Corporation. 10 January. 2003 <http://www.motoman.com/robots/models/up20.htm>

[48] "Motoman Controllers". Home page. Motoman Corporation. 10 January. 2003 <http://www.motoman.com/controller/controllers.htm>

Appendix A - Experimental Results

A.1. Teach-points Coordinates

A.1.1. Approach Motion

Table A.1 Coordinates of the start teach-points for approach motion

Teach-point	X [mm]	Y [mm]	Z [mm]	A [deg]	B [deg]	C [deg]
TOP_LEFT	1350	700	100	180	-90	0
TOP_1	1350	420	100	180	-90	0
TOP_2	1350	140	100	180	-90	0
TOP_3	1350	-140	100	180	-90	0
TOP_4	1350	-420	100	180	-90	0
TOP_RIGHT	1350	-700	100	180	-90	0

Table A.2 Coordinates of the target teach-points for approach motion

Teach-point	X [mm]	Y [mm]	Z [mm]	A [deg]	B [deg]	C [deg]
BOTTOM_LEFT	710	700	100	180	-90	0
TP1	710	650	100	180	-90	0
TP2	710	600	100	180	-90	0
TP3	710	550	100	180	-90	0
TP4	710	500	100	180	-90	0
TP5	710	450	100	180	-90	0
TP6	710	400	100	180	-90	0
TP7	710	350	100	180	-90	0
TP8	710	300	100	180	-90	0
TP9	710	250	100	180	-90	0
TP10	710	200	100	180	-90	0
TP11	710	150	100	180	-90	0
TP12	710	100	100	180	-90	0
TP13	710	50	100	180	-90	0

Table A.2 Coordinates of the target teach-points (continued)

Teach-point	X [mm]	Y [mm]	Z [mm]	A [deg]	B [deg]	C [deg]
TP14	710	0	100	180	-90	0
TP15	710	-50	100	180	-90	0
TP16	710	-100	100	180	-90	0
TP17	710	-150	100	180	-90	0
TP18	710	-200	100	180	-90	0
TP19	710	-250	100	180	-90	0
TP20	710	-300	100	180	-90	0
TP21	710	-350	100	180	-90	0
TP22	710	-400	100	180	-90	0
TP23	710	-450	100	180	-90	0
TP24	710	-500	100	180	-90	0
TP25	710	-550	100	180	-90	0
TP26	710	-600	100	180	-90	0
TP27	710	-650	100	180	-90	0
BOTTOM_RIGHT	710	-700	100	180	-90	0

A.1.2. Depart Motion

Table A.3 Coordinates of the start teach-points for depart motion

Teach-point	X [mm]	Y [mm]	Z [mm]	A [deg]	B [deg]	C [deg]
TOP_LEFT	710	700	100	180	-90	0
TOP_1	710	420	100	180	-90	0
TOP_2	710	140	100	180	-90	0
TOP_3	710	-140	100	180	-90	0
TOP_4	710	-420	100	180	-90	0
TOP_RIGHT	710	-700	100	180	-90	0

Table A.4 Coordinates of the target teach-points for depart motion

Teach-point	X [mm]	Y [mm]	Z [mm]	A [deg]	B [deg]	C [deg]
BOTTOM_LEFT	1350	700	100	180	-90	0
TP1	1350	650	100	180	-90	0
TP2	1350	600	100	180	-90	0
TP3	1350	550	100	180	-90	0
TP4	1350	500	100	180	-90	0
TP5	1350	450	100	180	-90	0
TP6	1350	400	100	180	-90	0
TP7	1350	350	100	180	-90	0
TP8	1350	300	100	180	-90	0
TP9	1350	250	100	180	-90	0
TP10	1350	200	100	180	-90	0
TP11	1350	150	100	180	-90	0
TP12	1350	100	100	180	-90	0
TP13	1350	50	100	180	-90	0
TP14	1350	0	100	180	-90	0
TP15	1350	-50	100	180	-90	0
TP16	1350	-100	100	180	-90	0
TP17	1350	-150	100	180	-90	0
TP18	1350	-200	100	180	-90	0
TP19	1350	-250	100	180	-90	0
TP20	1350	-300	100	180	-90	0
TP21	1350	-350	100	180	-90	0
TP22	1350	-400	100	180	-90	0
TP23	1350	-450	100	180	-90	0
TP24	1350	-500	100	180	-90	0
TP25	1350	-550	100	180	-90	0
TP26	1350	-600	100	180	-90	0
TP27	1350	-650	100	180	-90	0
BOTTOM_RIGHT	1350	-700	100	180	-90	0

A.1.3. Downward Motion

Table A.5 Coordinates of the start teach-points for downward motion

Teach-point	X [mm]	Y [mm]	Z [mm]	A [deg]	B [deg]	C [deg]
TOP_LEFT	1350	700	740	180	-90	0
TOP_1	1350	420	740	180	-90	0
TOP_2	1350	140	740	180	-90	0
TOP_3	1350	-140	740	180	-90	0
TOP_4	1350	-420	740	180	-90	0
TOP_RIGHT	1350	-700	740	180	-90	0

Table A.6 Coordinates of the target teach-points for downward motion

Teach-point	X [mm]	Y [mm]	Z [mm]	A [deg]	B [deg]	C [deg]
BOTTOM_LEFT	1350	700	100	180	-90	0
TP1	1350	650	100	180	-90	0
TP2	1350	600	100	180	-90	0
TP3	1350	550	100	180	-90	0
TP4	1350	500	100	180	-90	0
TP5	1350	450	100	180	-90	0
TP6	1350	400	100	180	-90	0
TP7	1350	350	100	180	-90	0
TP8	1350	300	100	180	-90	0
TP9	1350	250	100	180	-90	0
TP10	1350	200	100	180	-90	0
TP11	1350	150	100	180	-90	0
TP12	1350	100	100	180	-90	0
TP13	1350	50	100	180	-90	0
TP14	1350	0	100	180	-90	0
TP15	1350	-50	100	180	-90	0
TP16	1350	-100	100	180	-90	0
TP17	1350	-150	100	180	-90	0
TP18	1350	-200	100	180	-90	0
TP19	1350	-250	100	180	-90	0
TP20	1350	-300	100	180	-90	0
TP21	1350	-350	100	180	-90	0
TP22	1350	-400	100	180	-90	0
TP23	1350	-450	100	180	-90	0

Table A.6 Coordinates of the target teach-points (continued)

Teach-point	X [mm]	Y [mm]	Z [mm]	A [deg]	B [deg]	C [deg]
TP24	1350	-500	100	180	-90	0
TP25	1350	-550	100	180	-90	0
TP26	1350	-600	100	180	-90	0
TP27	1350	-650	100	180	-90	0
BOTTOM_RIGHT	1350	-700	100	180	-90	0

A.2. Motion Times

A.2.1. Approach Motion

Table A.7 Approach motion times – start teach-point "TOP_LEFT"

Target Teach-point	Travel Distance	Bearing	Motion time		Error	
			Robot	Simulation		
	[mm]	[deg]	[sec]	[sec]	[sec]	[%]
BOTTOM_LEFT	640	0.00	2.64	2.90	-0.26	-9.8
TP1	641.95	4.47	2.66	2.90	-0.24	-9.0
TP2	647.76	8.88	2.7	2.92	-0.22	-8.1
TP3	657.34	13.19	2.77	2.96	-0.19	-6.9
TP4	670.52	17.35	2.83	3.00	-0.17	-6.0
TP5	687.09	21.34	2.91	3.06	-0.15	-5.2
TP6	706.82	25.11	2.99	3.12	-0.13	-4.3
TP7	729.45	28.67	3.08	3.20	-0.12	-3.9
TP8	754.71	32.01	3.18	3.28	-0.10	-3.1
TP9	782.36	35.11	3.28	3.37	-0.09	-2.7
TP10	812.15	38.00	3.39	3.47	-0.08	-2.4
TP11	843.86	40.67	3.57	3.58	-0.01	-0.3
TP12	877.26	43.15	3.73	3.70	0.03	0.8
TP13	912.19	45.44	3.86	3.81	0.05	1.3
TP14	948.47	47.56	3.98	3.93	0.05	1.3

Table A.8 Approach motion times – start teach-point "TOP_1"

Target Teach-point	Travel Distance	Bearing	Motion time		Error	
			Robot	Simulation		
	[mm]	[deg]	[sec]	[sec]	[sec]	[%]
BOTTOM_LEFT	698.57	-23.63	2.73	3.09	-0.36	-13.2
TP1	680.07	-19.77	2.70	3.03	-0.32	-12.0
TP2	664.83	-15.71	2.68	2.98	-0.3	-11.2
TP3	653.06	-11.48	2.66	2.94	-0.28	-10.5
TP4	644.98	-7.13	2.64	2.91	-0.27	-10.2
TP5	640.70	-2.68	2.65	2.90	-0.25	-9.4
TP6	640.31	1.79	2.66	2.90	-0.24	-9.0
TP7	643.81	6.24	2.69	2.91	-0.22	-8.2
TP8	651.15	10.62	2.73	2.94	-0.21	-7.7
TP9	662.19	14.88	2.78	2.97	-0.19	-6.8
TP10	676.75	18.97	2.85	3.02	-0.17	-6.0
TP11	694.62	22.87	2.91	3.08	-0.17	-5.8
TP12	715.54	26.57	2.98	3.15	-0.17	-5.7
TP13	739.25	30.03	3.065	3.24	-0.17	-5.7
TP14	765.50	33.27	3.15	3.32	-0.17	-5.4
TP15	794.04	36.29	3.24	3.41	-0.17	-5.2

Table A.9 Approach motion times – start teach-point "TOP_2"

Target Teach-point	Travel Distance	Bearing Angle	Motion time		Error	
			Robot	Simulation		
	[mm]	[deg]	[sec]	[sec]	[sec]	[%]
BOTTOM_LEFT	850.41	-41.19	3.35	3.60	-0.25	-7.5
TP1	818.35	-38.55	3.23	3.49	-0.26	-8.0
TP2	788.16	-35.71	3.13	3.39	-0.26	-8.3
TP3	760.07	-32.64	3.03	3.30	-0.27	-8.9
TP4	734.30	-29.36	2.95	3.21	-0.26	-8.8
TP5	711.13	-25.84	2.87	3.14	-0.27	-9.4
TP6	690.80	-22.11	2.81	3.07	-0.26	-9.3
TP7	673.57	-18.17	2.76	3.01	-0.25	-9.1
TP8	659.70	-14.04	2.72	2.96	-0.24	-8.8
TP9	649.38	-9.75	2.71	2.93	-0.22	-8.1
TP10	642.81	-5.36	2.70	2.91	-0.21	-7.8
TP11	640.08	-0.90	2.69	2.90	-0.21	-7.8
TP12	641.25	3.58	2.72	2.91	-0.20	-7.2
TP13	646.30	8.00	2.74	2.92	-0.18	-6.6
TP14	655.13	12.34	2.78	2.95	-0.17	-6.1

Table A.10 Approach motion times – start teach-point "TOP_3"

Target Teach-point	Travel Distance	Bearing Angle	Motion time		Error	
			Robot	Simulation		
	[mm]	[deg]	[sec]	[sec]	[sec]	[%]
BOTTOM_LEFT	1056.03	-52.70	4.09	4.29	-0.20	-4.9
TP1	1016.71	-50.99	3.94	4.15	-0.21	-5.3
TP2	978.37	-49.14	3.81	4.03	-0.22	-5.8
TP3	941.12	-47.15	3.66	3.90	-0.24	-6.6
TP4	905.10	-45.00	3.53	3.78	-0.25	-7.1
TP5	870.46	-42.67	3.41	3.67	-0.26	-7.6
TP6	837.38	-40.16	3.30	3.56	-0.26	-7.9
TP7	806.04	-37.44	3.19	3.45	-0.26	-8.2
TP8	776.66	-34.51	3.08	3.35	-0.27	-8.8
TP9	749.47	-31.36	2.99	3.26	-0.27	-9.0
TP10	724.71	-27.98	2.91	3.18	-0.27	-9.3
TP11	702.64	-24.38	2.84	3.11	-0.27	-9.5
TP12	683.52	-20.56	2.78	3.04	-0.26	-9.4
TP13	667.61	-16.53	2.74	2.99	-0.25	-9.1
TP14	655.13	-12.34	2.72	2.95	-0.23	-8.5
TP15	646.30	-8.00	2.70	2.92	-0.22	-8.1

A.2.2. Depart Motion

Table A.11 Depart motion times – start teach-point "TOP_LEFT"

Target Teach-point	Travel Distance	Bearing Angle	Motion time		Error	
			Robot	Simulation		
	[mm]	[deg]	[sec]	[sec]	[sec]	[%]
BOTTOM_LEFT	640.00	0.00	2.57	2.90	-0.33	-12.8
TP1	641.95	-4.47	2.54	2.90	-0.36	-14.2
TP2	647.77	-8.88	2.55	2.92	-0.37	-14.5
TP3	657.34	-13.19	2.57	2.96	-0.39	-15.2
TP4	670.52	-17.35	2.60	3.00	-0.40	-15.4
TP5	687.10	-21.34	2.66	3.06	-0.40	-15.0
TP6	706.82	-25.11	2.72	3.12	-0.40	-14.7
TP7	729.45	-28.67	2.80	3.20	-0.40	-14.3
TP8	754.72	-32.01	2.89	3.28	-0.39	-13.5
TP9	782.37	-35.11	2.99	3.37	-0.38	-12.7
TP10	812.16	-38.00	3.10	3.47	-0.37	-11.9
TP11	843.86	-40.67	3.21	3.58	-0.37	-11.5
TP12	877.27	-43.15	3.34	3.69	-0.35	-10.5
TP13	912.20	-45.44	3.47	3.81	-0.34	-9.8
TP14	948.47	-47.56	3.60	3.93	-0.33	-9.2
TP15	985.95	-49.52	3.74	4.05	-0.31	-8.3
TP16	1024.50	-51.34	3.88	4.18	-0.30	-7.7
TP17	1064.00	-53.02	4.03	4.31	-0.28	-6.9

Table A.12 Depart motion times – start teach-point "TOP_1"

Target Teach-point	Travel Distance	Incline Angle	Motion time		Error	
			Robot	Simulation		
	[mm]	[deg]	[sec]	[sec]	[sec]	[%]
BOTTOM_LEFT	698.57	23.63	2.86	3.09	-0.23	-8.0
TP1	680.07	19.77	2.78	3.03	-0.25	-9.0
TP2	664.83	15.71	2.71	2.98	-0.27	-10.0
TP3	653.07	11.48	2.65	2.94	-0.29	-10.9
TP4	644.98	7.13	2.61	2.91	-0.30	-11.5
TP5	640.70	2.68	2.59	2.90	-0.31	-12.0
TP6	640.31	-1.79	2.57	2.90	-0.33	-12.8
TP7	643.82	-6.24	2.57	2.91	-0.34	-13.2
TP8	651.15	-10.62	2.60	2.94	-0.34	-13.1

Table A.12 Depart motion times – start teach-point "TOP_1" (continued)

Target Teach-point	Travel Distance	Incline Angle	Motion time		Error	
			Robot	Simulation		
	[mm]	[deg]	[sec]	[sec]	[sec]	[%]
TP9	662.19	-14.88	2.62	2.97	-0.35	-13.4
TP10	676.76	-18.97	2.66	3.02	-0.36	-13.5
TP11	694.62	-22.87	2.72	3.08	-0.36	-13.2
TP12	715.54	-26.57	2.78	3.15	-0.37	-13.3
TP13	739.26	-30.03	2.87	3.23	-0.37	-12.7
TP14	765.51	-33.27	2.95	3.32	-0.37	-12.5
TP15	794.04	-36.29	3.05	3.41	-0.36	-11.8
TP16	824.62	-39.09	3.16	3.51	-0.35	-11.1
TP17	857.03	-41.69	3.28	3.62	-0.34	-10.4

Table A.13 Depart motion times – start teach-point "TOP_2"

Target Teach-point	Travel Distance	Incline Angle	Motion time		Error	
			Robot	Simulation		
	[mm]	[deg]	[sec]	[sec]	[sec]	[%]
BOTTOM_LEFT	850.41	41.19	3.44	3.60	-0.16	-4.7
TP1	818.35	38.55	3.31	3.49	-0.18	-5.4
TP2	788.16	35.71	3.19	3.39	-0.20	-6.3
TP3	760.07	32.64	3.08	3.30	-0.22	-7.1
TP4	734.30	29.36	2.98	3.21	-0.23	-7.7
TP5	711.13	25.84	2.89	3.14	-0.25	-8.7
TP6	690.80	22.11	2.80	3.07	-0.27	-9.6
TP7	673.57	18.17	2.73	3.01	-0.28	-10.3
TP8	659.70	14.04	2.67	2.96	-0.29	-10.9
TP9	649.38	9.75	2.63	2.93	-0.30	-11.4
TP10	642.81	5.36	2.63	2.91	-0.28	-10.6
TP11	640.08	0.90	2.59	2.90	-0.31	-12.0
TP12	641.25	-3.58	2.61	2.90	-0.29	-11.1
TP13	646.30	-8.00	2.63	2.92	-0.29	-11.0
TP14	655.13	-12.34	2.65	2.95	-0.30	-11.3
TP15	667.61	-16.53	2.69	2.99	-0.30	-11.2
TP16	683.52	-20.56	2.73	3.04	-0.31	-11.4
TP17	702.64	-24.38	2.80	3.11	-0.31	-11.1

A.2.3. Downward Motion

Table A.14 Downward motion times – start teach-point "TOP_LEFT"

Target Teach-point	Travel Distance	Incline Angle	Motion time		Error	
			Robot	Simulation		
	[mm]	[deg]	[sec]	[sec]	[sec]	[%]
BOTTOM_LEFT	640.00	0.00	2.64	2.86	-0.22	-8.33
TP1	641.95	4.47	2.60	2.87	-0.27	-10.38
TP2	647.77	8.88	2.57	2.89	-0.32	-12.45
TP3	657.34	13.19	2.56	2.92	-0.36	-14.06
TP4	670.52	17.35	2.60	2.97	-0.37	-14.23
TP5	687.10	21.34	2.65	3.02	-0.37	-13.96
TP6	706.82	25.11	2.72	3.09	-0.37	-13.60
TP7	729.45	28.67	2.80	3.17	-0.37	-13.21
TP8	754.72	32.01	2.89	3.25	-0.36	-12.46
TP9	782.37	35.11	3.01	3.35	-0.34	-11.30
TP10	812.16	38.00	3.15	3.45	-0.30	-9.52
TP11	843.86	40.67	3.28	3.55	-0.27	-8.23
TP12	877.27	43.15	3.38	3.66	-0.28	-8.28
TP13	912.20	45.44	3.52	3.78	-0.27	-7.54
TP14	948.47	47.56	3.65	3.90	-0.25	-6.85

Table A.15 Downward motion times – start teach-point "TOP_2"

Target Teach-point	Travel Distance	Incline Angle	Motion time		Error	
			Robot	Simulation		
	[mm]	[deg]	[sec]	[sec]	[sec]	[%]
BOTTOM_LEFT	850.41	-41.19	3.45	3.60	-0.15	-4.35
TP1	818.35	-38.55	3.28	3.47	-0.19	-5.79
TP2	788.16	-35.71	3.16	3.37	-0.21	-6.65
TP3	760.07	-32.64	3.05	3.27	-0.22	-7.21
TP4	734.30	-29.36	2.97	3.18	-0.22	-7.25
TP5	711.13	-25.84	2.88	3.11	-0.23	-7.99
TP6	690.80	-22.11	2.79	3.04	-0.25	-8.96
TP7	673.57	-18.17	2.73	2.98	-0.25	-9.16
TP8	659.70	-14.04	2.69	2.93	-0.24	-8.92
TP9	649.38	-9.75	2.65	2.90	-0.25	-9.43
TP10	642.81	-5.36	2.62	2.87	-0.25	-9.54

Table A.15 Motion Times – Start Teach-point "TOP_2" (continued)

Target Teach-point	Travel Distance	Incline Angle	Motion time		Error	
			Robot	Simulation		
	[mm]	[deg]	[sec]	[sec]	[sec]	[%]
TP11	640.08	-0.90	2.62	2.87	-0.25	-9.54
TP12	641.25	3.58	2.61	2.87	-0.26	-9.96
TP13	646.30	8.00	2.61	2.89	-0.28	-10.73
TP14	655.13	12.34	2.62	2.92	-0.30	-11.45

A.3. Correction Factors

A.3.1. Horizontal Motion Plane

Table A.16 Correction factor values C_H for approach motion

"TOP_LEFT"		"TOP_1"		"TOP_2"		"TOP_3"	
Bearing [deg]	C_H	Bearing [deg]	C_H	Bearing [deg]	C_H	Bearing [deg]	C_H
0.00	1.48	-23.63	1.87	-41.19	1.46	-52.70	1.32
4.47	1.44	-19.77	1.71	-38.55	1.49	-50.99	1.36
8.88	1.39	-15.71	1.62	-35.71	1.49	-49.14	1.37
13.19	1.30	-11.48	1.55	-32.64	1.51	-47.15	1.43
17.35	1.26	-7.13	1.53	-29.36	1.49	-45.00	1.46
21.34	1.21	-2.68	1.46	-25.84	1.50	-42.67	1.48
25.11	1.18	1.79	1.43	-22.11	1.48	-40.16	1.47
28.67	1.16	6.24	1.38	-18.17	1.46	-37.44	1.49
32.01	1.13	10.62	1.34	-14.04	1.44	-34.51	1.53
35.11	1.12	14.88	1.31	-9.75	1.38	-31.36	1.53
38.00	1.10	18.97	1.26	-5.36	1.35	-27.98	1.52
40.67	0.99	22.87	1.26	-0.90	1.35	-24.38	1.51
43.15	0.93	26.57	1.26	3.58	1.30	-20.56	1.50
45.44	0.92	30.03	1.25	8.00	1.28	-16.53	1.46
47.56	0.92	33.27	1.25	12.34	1.26	-12.34	1.40
		36.29	1.26			-8.00	1.37

Is it live?
DISC
MEMOREX
CD-R
PLATINUM
700MB/80min