Parametric Interpolation Algorithms for Motion Control

YUAN KAI, CHOW

October 2003

A thesis submitted to The Nottingham Trent University in partial fulfilment of the requirements for the degree of Doctor of Philosophy

The Nottingham Trent University
in collaboration with
Axiomatic Technology (UK) Ltd

Abstract

Computer numerical controlled machine tools have been used in a wide range of applications. Machine drives have used both stepper motors and servomotors to control the motion. Stepper motors are a less costly alternative and are suitable for low-cost machining. Leadscrews convert the stepper motor's rotary motion to linear motion. This project has concentrated on stepper motors but it is expected that the algorithms could be extended for use with servomotors. Most stepper motor driven machine tools are used in an open-loop system. Since there is no feedback from the actual motion of the machine, extreme care has to be taken when designing the controller.

Investigations into multi-axis stepper motor continuous path machining systems have revealed problems affecting machining performance, such as vibrations. Earlier work within the research group has demonstrated how these vibrations arise from the excitation of machine dynamics, which are sometimes due to sudden variations in pulse rates. For high speed machining, the machine must be accelerated and the problem with vibrations becomes more severe. Such rough motion can then result in unsatisfactory path following, if it increases any positional error (deviation of the machined path from the desired path). Besides vibrations, another cause of positional errors may be the approximations made by the interpolation algorithms used. Many motion control systems also suffer from the disadvantage that they are not able to maintain a constant speed round a curve.

Previous researchers in the group have addressed the above problems with some success. Their work includes smoothing of unevenly spaced pulse timings after they have been generated. However, such smoothing may increase the positional errors. The work described in this thesis addresses the problems from a more fundamental viewpoint, by generating pulse timings that are smooth initially.

New line and arc interpolation algorithms have been developed for stepper motors, which calculate the timing for every individual pulse. These algorithms exploit recent advances in microprocessor technology which allow the use of a high-speed digital signal processor. A new, more general approach to interpolation has been applied, which avoids the need for the path to pass through particular intermediate points. A novel approach is to use the distance along the path as a parameter, which ensures synchronisation of the axes. With these algorithms sudden changes in speed can be avoided. Four partial simulation techniques have been developed in order to evaluate the algorithms. Simulation results show that we can expect not only significant reduction in positional errors but also greatly diminished fluctuations in speed. Therefore vibrations are also likely to be reduced.

New acceleration algorithms have been developed, so that the new interpolation algorithms can be used for high-speed machining. The algorithms are based on linear and parabolic acceleration algorithms described by previous authors. Simulation results show that the new acceleration algorithms are expected to allow the machine to follow the required curve closely while changing speed as required. Thus vibrations are again expected to be reduced.

An initial evaluation of the practical implementation of the new algorithms has been undertaken on a CNC machine. The results are promising and broadly in agreement with the simulation results.

The work described in this thesis is the Author's own, unless otherwise stated, and it is, as far as he is aware, original.

To my mother, father and Yen San Yong for their love, support and encouragement. No words can express my deepest gratitude to them.

Acknowledgements

I would like to pay tribute to several people who have given an enormous amount of help throughout my course of research in The Nottingham Trent University. I am grateful to my supervisors, Prof. Peter Thomas, Dr. Janet Poliakoff and Dr. Paul Orton for their advice, encouragement and wholehearted support. I wish to thank my friends and colleagues, past and present, in the Intelligent Machines group and Axiomatic Technology for their help and friendly support, especially Mark Howson. Also of course, my sincere gratitude to my family for their encouragement and support.

Table of Contents

Abbreviations

APT:	Automatically Programmed Tools
CAD:	Computer Aided Design
CAM:	Computer Aided Manufacturing
CNC:	Computer Numerical Control
DDA:	Digital Differential Analyser
DSM:	Direct Search Method
DSP:	Digital Signal Processor
EDM:	Electrical Discharge Machining
HPGL:	Hewlett-Packard Graphics Language
MCU:	Machine Control Unit
NC:	Numerical Control

List of Units:

m	metres
s	seconds

Glossary of Terms

Acceleration:	The rate of change in speed as a function of time. Acceleration usually refers to increasing in speed.
Computer Aided Design:	Making use of the computer to design models of an object, normally using a graphical interface.
Computer Aided Manufacturing:	Making use of the computer to generate control of manufacturing machines in an organised and efficient way.
Computer Numerical Control:	Making use of the computer to control machine tools. The input consists of numerical data.
Critical Damping:	The value of damping that provides most rapid response to a step function without overshoot.
Damping:	Dissipation of vibratory energy with time.
Damping Factor:	A ratio of the actual damping of a system relative to its critically damped value.
Deceleration:	Rate of change in speed where the speed decreases in time.
Digital Signal Processor:	Fast processor for mathematical applications.
Feedrate:	The speed at which the machining is required to be performed whilst following the desired shape.
Interpolation:	Providing a machine with control signals (in terms

	of position or speed at given times) so that it moves in such a manner as to follow a desired path.
Position Error:	The difference between the desired shape and interpolated position.
Pull-in Speed:	Maximum demanded speed to which a motor can respond without loss of steps when starting from rest.
Pull-out Speed:	Maximum operating speed for a given load.
Pull-out Torque:	Maximum torque which the motor can develop at each operating speed.
Resolution:	The smallest positioning increment that can be achieved.
Shaft Encoder:	Digital transducer that is used for measuring angular displacements and angular velocities.
Speed:	Magnitude of the velocity.
Spline:	Composite curve comprising of various (usually) polynomial segments. Successive segments are connected with a predefined continuity.
Stepper motor:	A type of motor that can move in increments (steps) in response to command signals. One pulse normally results in movement by one step angle.
Step size:	The linear motion of the cutter corresponding to the movement of one step angle of a stepper motor.

Torque:	The moment of a system of forces tending to cause rotation.
Velocity:	Rate of change in distance (with known direction).

1 Introduction

In ancient times people started to carve on wood or stone (Figure 1-1), either as a means of communication or for decoration. In those days, all of their home equipment was made by hand. When humans invented machinery, they started to use machines to replace the jobs done by hand. Manpower was still essential until recently, because without numerical control, conventional machines are controlled by a skilled operator. The cutting tool is moved along the workpiece by the operator turning a hand wheel. With the introduction of Numerical Controlled (NC) machines, the programmer was only required to program the required path while most of the cutting operations were performed automatically by the machine tools.

Figure 1-1: Example of Ancient Egyptian Carvings on Stones [1].

NC technology has been one of manufacturing's major developments in the past 50 years. The applications of NC technology range from traditional milling, turning and drilling to today's laser cutting and water jet cutting. NC products are used in many industries, such as aerospace, engraving and sign making [2]. Since the 1970's, NC has been logically extended to Computer Numerical Control (CNC) by housing the

microprocessor on the machine tool itself. The machine drives used in today's machinery include servomotors and stepper motors. Many systems use stepper motors as their main drives because of the ease of implementation and low cost [3]. Investigations into stepper motor controlled continuous path motion control system have revealed several problems [4][5]. The Author's work includes the development of a solution to these problems. This thesis documents these ideas, experiments performed and the conclusions drawn.

This chapter provides an introduction to NC and CNC machine tools and how they are applied in the industrial environment. Section 1.1 discusses briefly how machining technology has progressed from ancient times to today's CNC. In addition, this section will also explain the importance of CNC machines and how they help in the industrial production today. The research reported in this thesis is motivated by the problems that have arisen in some existing stepper motor controlled systems. Therefore, these problems are presented in Section 1.2. The following Section, 1.3, discusses the aims and objectives of the Author's work to achieve the goal of improving the accuracy and smoothness of the path following. Section 1.4 completes this introduction chapter with an outline of the complete thesis.

1.1 Historical Development of Manufacturing Systems

Figure 1-2 illustrates another example of the hand-carving work done by ancient man. Most of such carvings required a high level of skill and were used for communication or for decoration. In those days, all the tools were also made by hand.

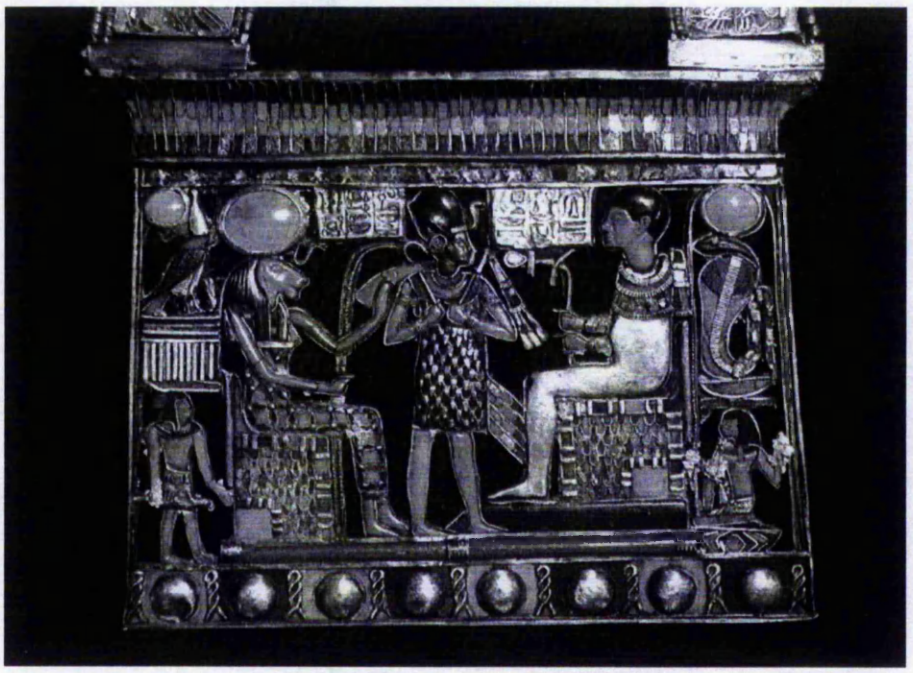


Figure 1-2: Example of Ancient Egyptian Carvings [1].

With the introduction of machines, cutting, carving and milling work have become easier. Without numerical control, conventional machining is performed by an operator, who moves the cutting tool along the workpiece by turning the hand wheel as shown in Figure 1-3. The operator counted the number of revolutions made on the hand wheel to achieve accurate positioning.

Numerically controlled systems are vital in modern manufacturing systems. Computers are one of the main resources for automation. This type of numerically controlled system has its origin in the Industrial Revolution. Since 1808, weaving machines began to use metal cards with holes punched on them to control the pattern of the cloth being produced [6]. Each needle on the machine was controlled by the presence or absence of a hole on the punched card. Another example of numerical control is the player-piano

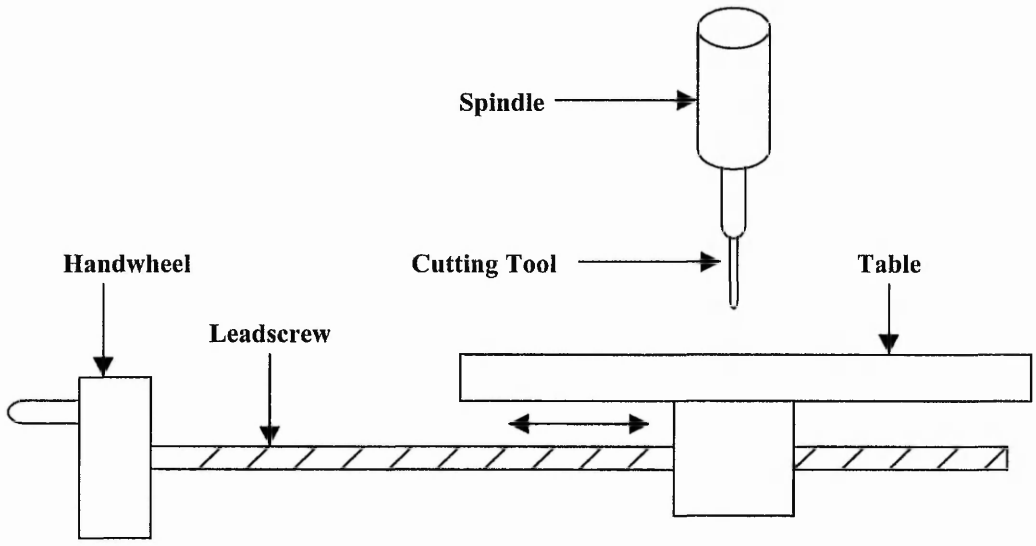


Figure 1-3: Conventional Hand-Controlled Machine.

(better known as pianola). No pianist is needed and the pianola uses a roll of paper with holes punched on it. This time the presence or absence of a hole determines whether a particular note is played.

The Numerically Controlled (NC) machine tool was invented by John Parsons in Traverse City, Michigan, and was subcontracted to the Massachusetts Institute of Technology (MIT) Servomechanism Laboratory in the early 1950's [2]. The motivation was that the U.S. Air Force needed to manufacture complex aircraft parts accurately. Such complex parts were difficult to produce using conventional machine tools. The first machine produced by Parsons and MIT was called a Cincinnati Hydrotel, a three-axis vertical spindle milling machine [7].

The part program used to control the machine tool is kept in a storage device. Traditionally, it was stored on punched tapes. The data on the punched tapes could be generated manually or using a computer with the help of a computer-assisted programming system. The commonly used programming language was the Automatically Programmed Tools (APT), which was developed in the late 1950's by MIT [7]. APT uses English-like words to describe the geometry and the tool motions in a part program.

A typical NC process is illustrated in Figure 1-4 and can be decomposed into:

- Stage 1: Part design.
- Stage 2: Translation into machine command language with additional information about cutters, etc.
- Stage 3: Translation from machine command language into machine code and storing the information on paper tape.
- Stage 4: Machine Control Unit (MCU) reads processing information and controls manufacturing process on machine tool.

In 1976, Computer Numerical Control (CNC) machines were introduced to replace the conventional NC machines. This was stimulated by the invention of microprocessors in 1974 [2]. Computers have replaced most of the digital hardware control boards of the NC machine. In contrast to the NC machine, which performs most of the data handling and control processes within the hardware circuitry, the CNC machine makes use of an on-board computer system. CNC machines allow much greater storage capacity compared to NC machines [5]. In an NC system, the controller will only accept one block of instructions and execute it before taking in any further instruction blocks. A CNC machine can store a whole program. Furthermore, program editing can be done on the machine itself.

CNC machines allow movement to be actuated by stepper motors or servomotors under signals from the controller [8][9][10][11], and guided by the part program as illustrated in Figure 1-5. The part program is normally arranged in the form of blocks of information [2], where each block contains the numerical data required to produce one segment of the workpiece. The machine takes the command from the CNC program. The drive motor is then rotated through a corresponding angle, which in turn drives the leadscrew, causing motion along the axis.

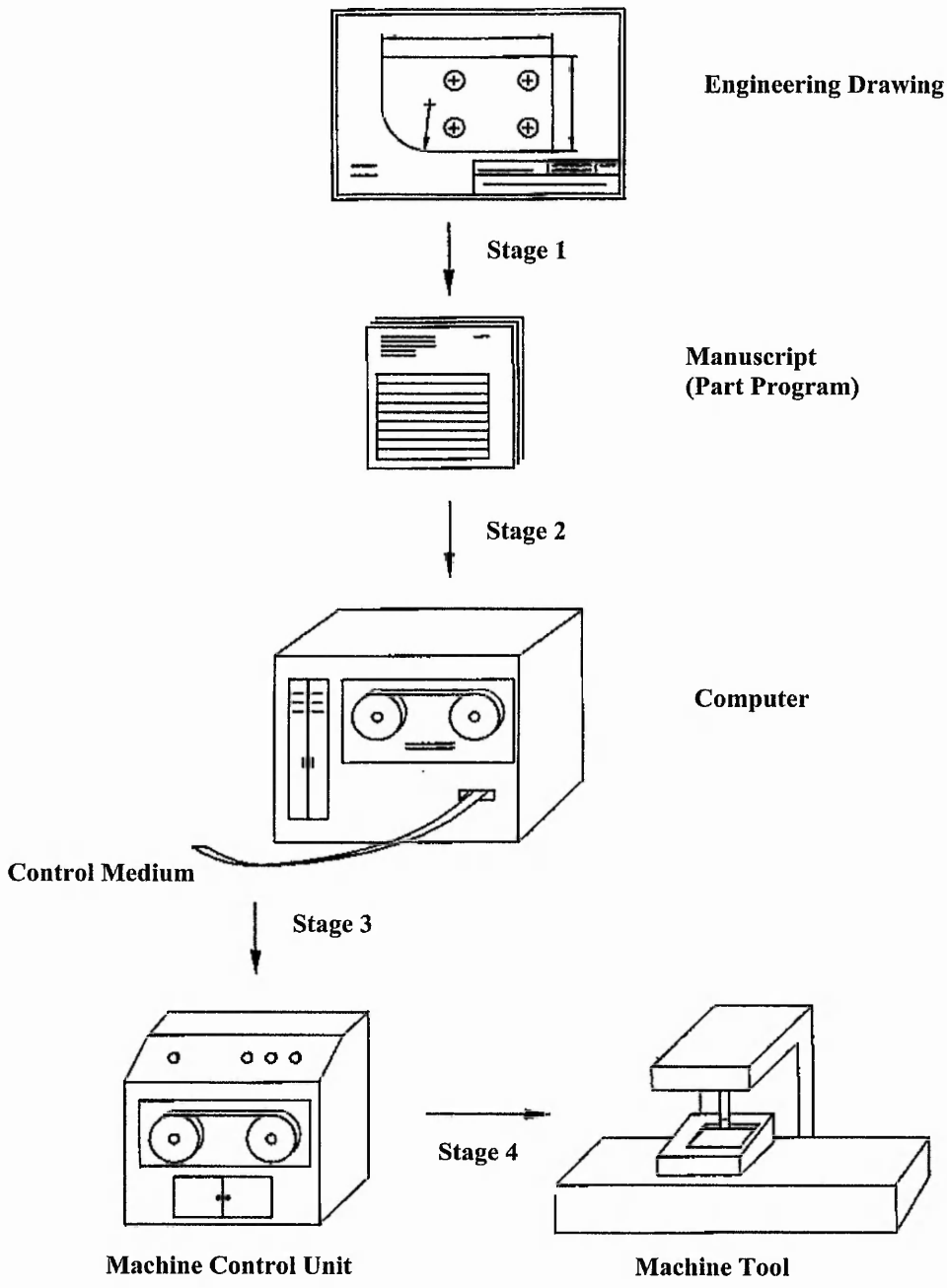


Figure 1-4: Structure of Early CNC Machining (Adapted from [4]).

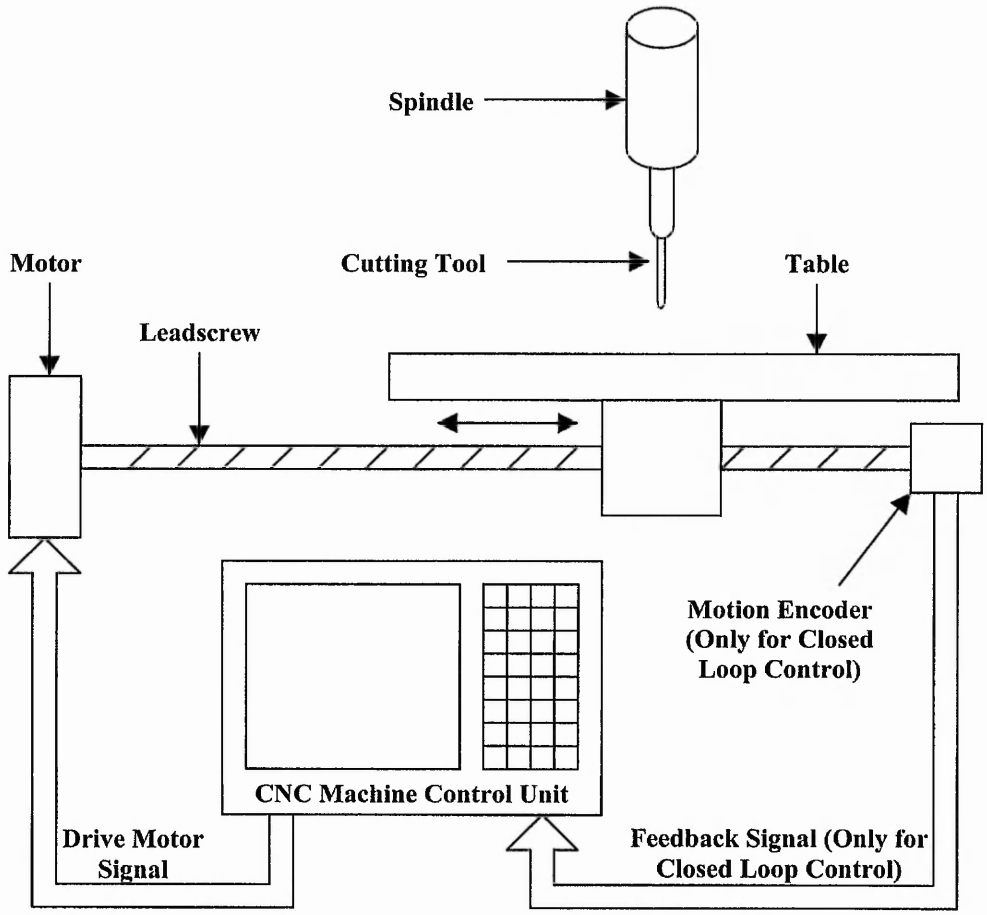


Figure 1-5: CNC Machine.

One of the CNC machines used by Pacer Systems Ltd. [12] is shown in Figure 1-6. This machine uses stepper motors in an open loop manner as the main drives. This particular machine offers an active work area of one square metre.

The advantage of using CNC machine tools is greater automation [2]. During the machining process, the machine can run unattended, enabling the operator to do other tasks. The skill level requirement for a CNC operator is also less than for a conventional machining operator, who is required to operate the hand wheel throughout the machining process. Furthermore, the machined workpieces are more uniform and accurate. In other words, much greater repeatability is possible. After verification of the part program, many identical objects can be produced with a high degree of accuracy. Thus, CNC machines are able to increase productivity.

Figure 1-6: Pacer Cadet CNC Machining [12].

Another important advantage with the CNC system is flexibility [2][6]. Machining is performed by running a part program. Therefore, a different object can be produced by loading the appropriate part program. With CNC machines, the machining process can now accommodate parts with both simple and complex geometry. Moreover, the production costs are decreased once the machine is set up.

CNC applications range from milling, turning and electric discharge machining (EDM) to laser, flame and plasma cutting, punching and nibbling, forming, bending, grinding, as well as inspection and robotics [2]. Continuing advances in computers have reduced the cost of CNC tremendously. Not only are aerospace industries able to afford such technology, but also small machine shops. CNC machines are found in automotive, electronics, engraving and sign making. An example of a complex path engraving is illustrated in Figure 1-7.

Figure 1-8 shows an example of the outline of a horse and the encircled part illustrates how part of the complex shape can be approximated by simple components, consisting of straight lines and circular arcs. The joints between segments, which are added for

Figure 1-7: Engraving Example [5].

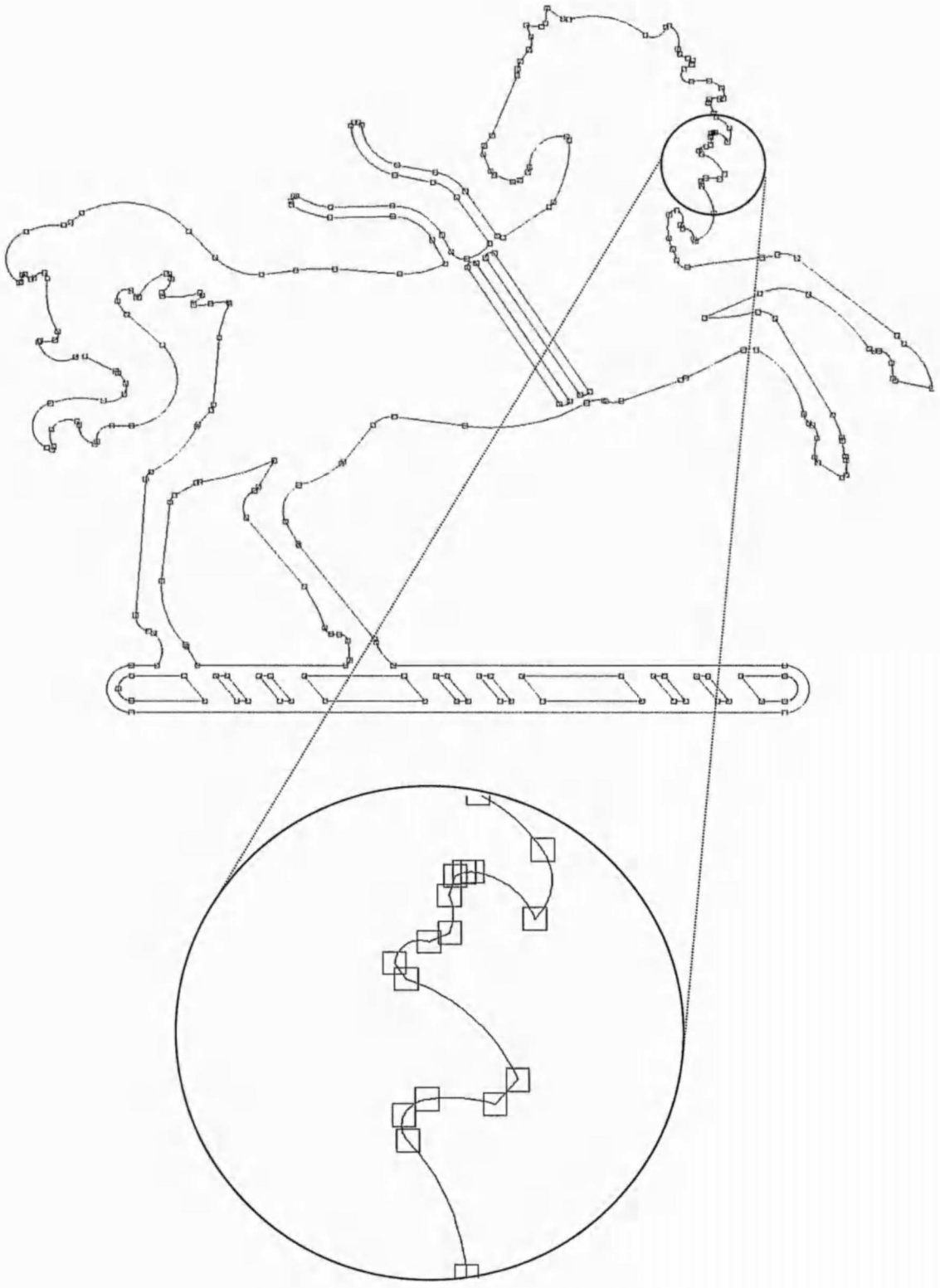


Figure 1-8: Plotting Example [13].

clarity in this example, are shown as square boxes. It can be seen that a smooth curve may be made by joining several arcs together.

One of the more advanced examples of CNC machining is the CNC laser cutting machine [6][14]. Laser cutting machines use the coherent laser light as a cutting tool and can cut plate stock into intricate shapes. Laser cutting is particularly suitable for materials such as ceramics, which are difficult to cut with normal cutters. The advantage of using laser cutting is the absence of mechanically-induced material damage, tool wear and machine vibration. Nevertheless, it can still cause damage to the material being cut.

The CNC machining centre is a machine tool capable of performing multiple operations and processes in a single set up [7]. It is typically fitted with an automatic tool changer. There are machining centres with automatic tool changers in vertical and horizontal configurations. Different machining operations can be accomplished in a single machine set up and the machine may have multiple spindles.

To maintain constant cutting width and to reduce the surface damage introduced by the machining method, the machine tool should be able to maintain smooth continuous path motion throughout the machining process [4][5]. The research described in this thesis has investigated the problems with existing continuous path motion control systems. With this knowledge, new interpolation algorithms have been developed to achieve smooth continuous path motion whilst improving the cutting accuracy. New acceleration algorithms enable smooth continuous path motion at high speed. All these are aimed at higher path quality.

1.2 Problems with Existing Systems

Most CNC machines in the industry can follow paths defined by combinations of straight lines and circular arcs [15]. If a complex path is required, it can be first approximated by straight line and circular arc segments. Then linear or circular arc interpolation is performed in the CNC controller by breaking the line or arc segment into smaller facets [15]. To maintain a higher path following accuracy, smaller facets are used [16]. However, this can result in longer machining time because the stepper motor may have to slow down at the end of each facet, if the angle between facets is too large [5]. On the other hand, if it does not slow down, this may cause vibrations. The following example, Figure 1-9, shows how a simple curve is approximated by three facets. The problem has been exaggerated here by using shorter facets for greater clarity. In these examples, the maximum change in x is 2 steps. In practice, the maximum change in x is typically about 10 steps or higher.

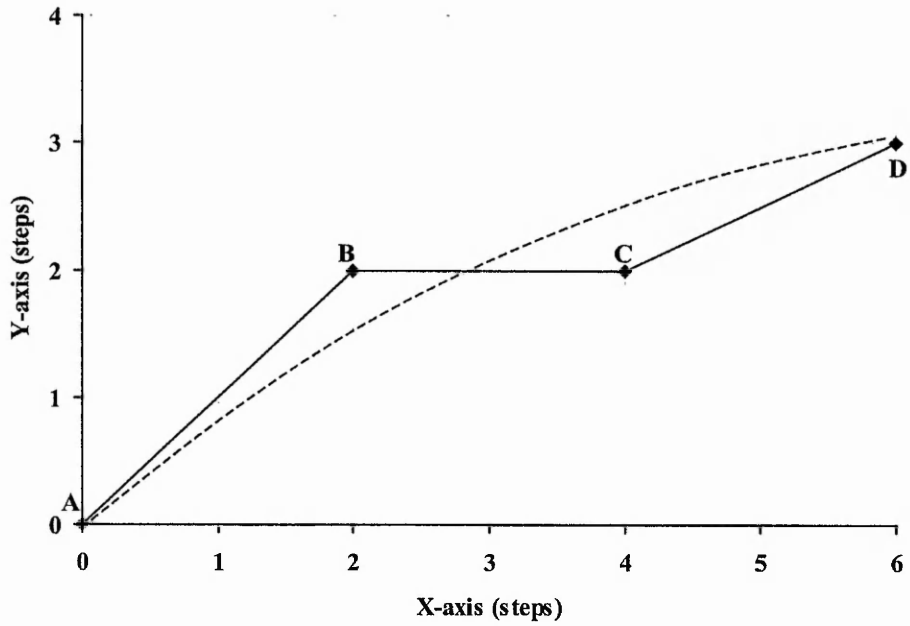


Figure 1-9: Approximation of Curve (Dashed Line) with Straight Lines (Changes in Direction: 45° and 26.6°).

The previous example involves a particularly bad case with changes of direction of 45° (changes from AB to BC) and 26.6° (changes from BC to CD). However, the changes of direction may not always be so large. Smaller changes in direction are less likely to cause vibrations on the motor. Figure 1-10 shows a less bad example, where the changes of direction are 18.4° and 0° respectively.

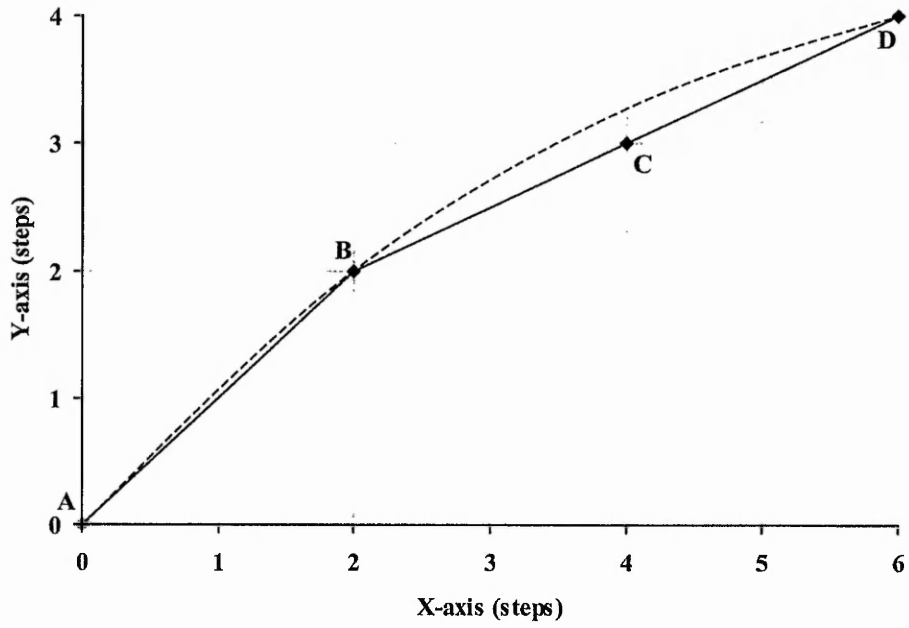


Figure 1-10: Approximation of Curve with Lines (Changes in Direction: 18.4° and 0°).

Figure 1-11 and Figure 1-12 illustrate how the interpolation can be achieved by approximating with short straight lines for a more realistic case where 10 steps in the X-axis are used for approximation. The linear interpolation example, shown as a dashed line in Figure 1-11, is for a straight line from the origin to (30,20). This straight line is approximated by three shorter straight line segments. The direction changes by 4.0° each time.

On the other hand, Figure 1-12 illustrates how the required circular arc, shown as dashed, is approximated by three straight line segments. This circular arc is from (0,20) to (20,0), centred at the origin. The changes in direction are 33.7° each time. Generally, if the straight line segments are longer, there will be a greater difference between the

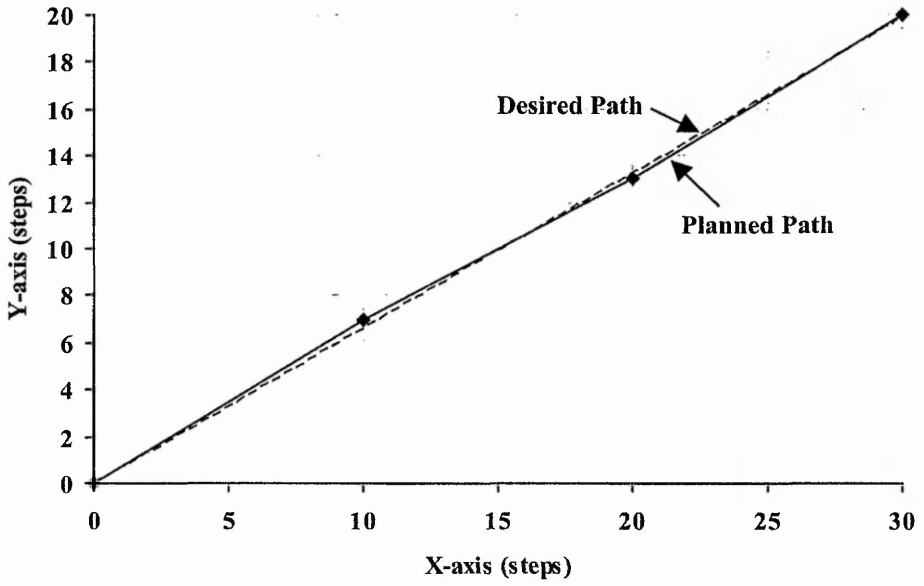


Figure 1-11: Linear Interpolation for Straight Line (0,0) to (30,20) via Points (10,7) and (20,13).

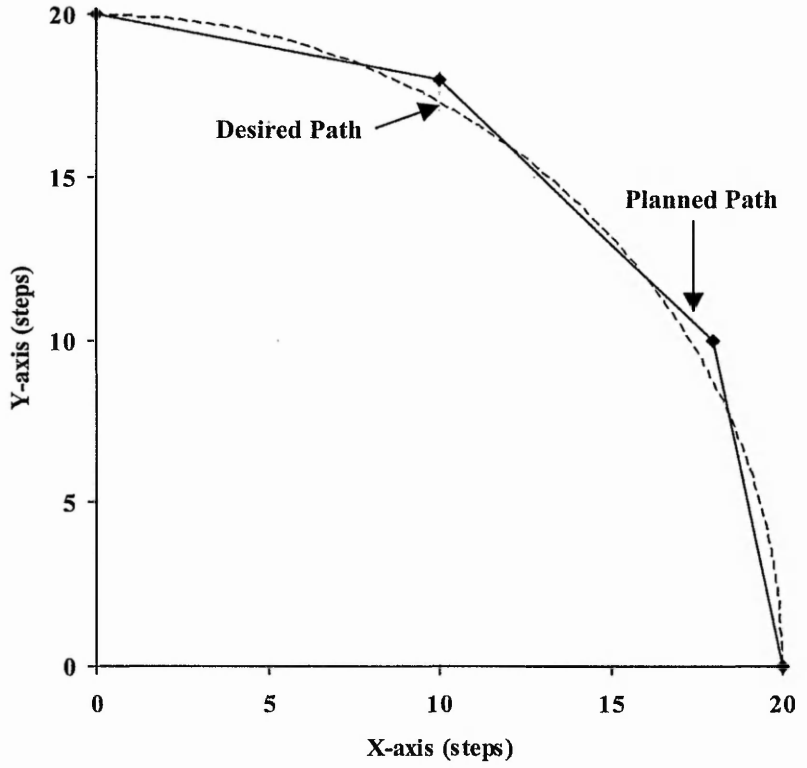


Figure 1-12: Circular Interpolation for Circular Arc from (0,20) to (20,0) with Centre (0,0) via Points (10,18) and (18,10).

desired path (arc) and the planned path (straight lines). Also, larger angles between segments could cause vibrations unless the machine slows down.

Investigations of multi-axis stepper motor motion control systems have identified three major problems:

1. Vibrations: The end effector can be subjected to vibrations [17]. Some of these are due to large variations in the rate of command pulses, as illustrated in Figure 1-13, caused by the interpolation algorithms. Others are due to the dynamic behaviour of the machine, caused for example by the friction or resonant frequencies of the machine.

2. Position errors: Significant separation between the interpolated path and the desired path can be caused by the interpolation algorithms [4]. The dynamic behaviour of the machine can sometimes increase these errors.

3. Varying resultant speed: The desired shape is required to be machined at a predefined speed (also called feedrate). However, existing systems are not always able to maintain a constant speed throughout the machining process [18].

The problems discussed above will result in unsatisfactory resultant path following. Vibrations and position errors cause small errors in the shape. Varying speed and vibrations affect the quality of cut.

The Intelligent Machines group in The Nottingham Trent University has investigated the stepper motor motion control system. Previous researchers in the group have managed to identify the source of errors generated during machining [19][20] (an example of rough motor motion during acceleration is shown in Figure 1-13) and have developed techniques to reduce the errors, such as the Variable Pulse Control algorithm with look back and look ahead feature [19] and algorithms for a DSP (Digital Signal Processor) filter [20]. These previous methods have aimed to reduce variations in pulse rates after they have been generated.

The research described in this thesis aims to generate pulses without the large variations in pulse rate. It concentrates, on the development of new smooth interpolation and acceleration algorithms to generate the appropriate pulse timings for multi-axis CNC machining in a different way. These timings are generated according to the geometry of the path. Thus, smooth streams of command pulses are maintained as much as possible in order to minimise the chance of vibrations caused by large variations in the rate of command pulses. At the same time, errors in position are kept low.

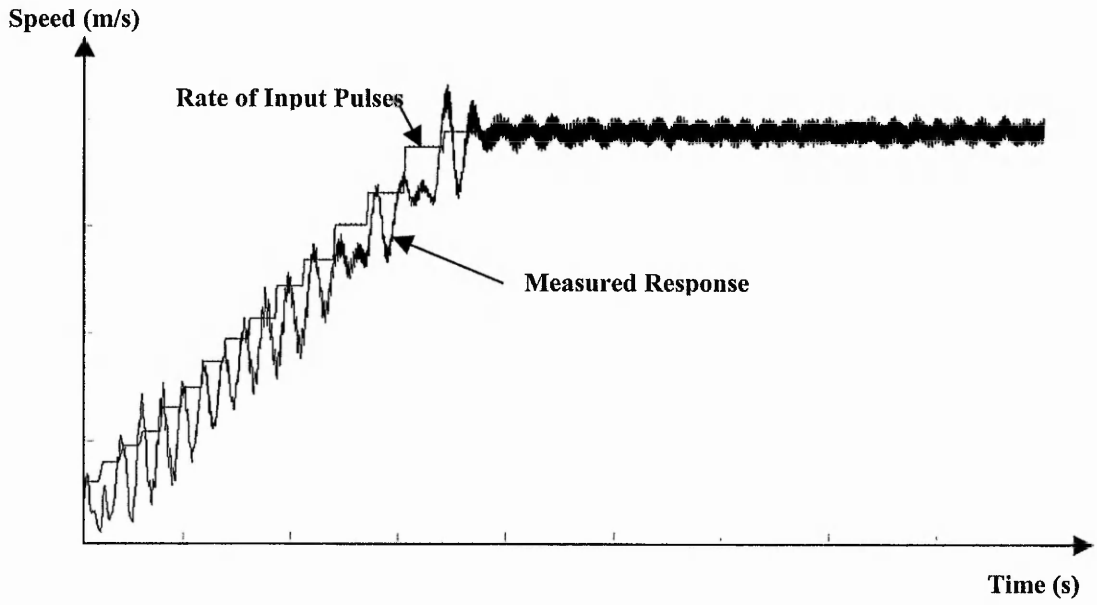


Figure 1-13: Example of Vibrations during Acceleration (Adapted from [5]). Command pulses are sent in blocks, each block at a higher rate than the previous one.

To illustrate how machine vibrations can affect the resultant path, Figure 1-14(a) shows an example of engraved shape from the Pacer Ltd CNC machine. At certain segments of the shape, Figure 1-14(b), it can be seen that machine vibrations have occurred, resulting in a rather rough finish.

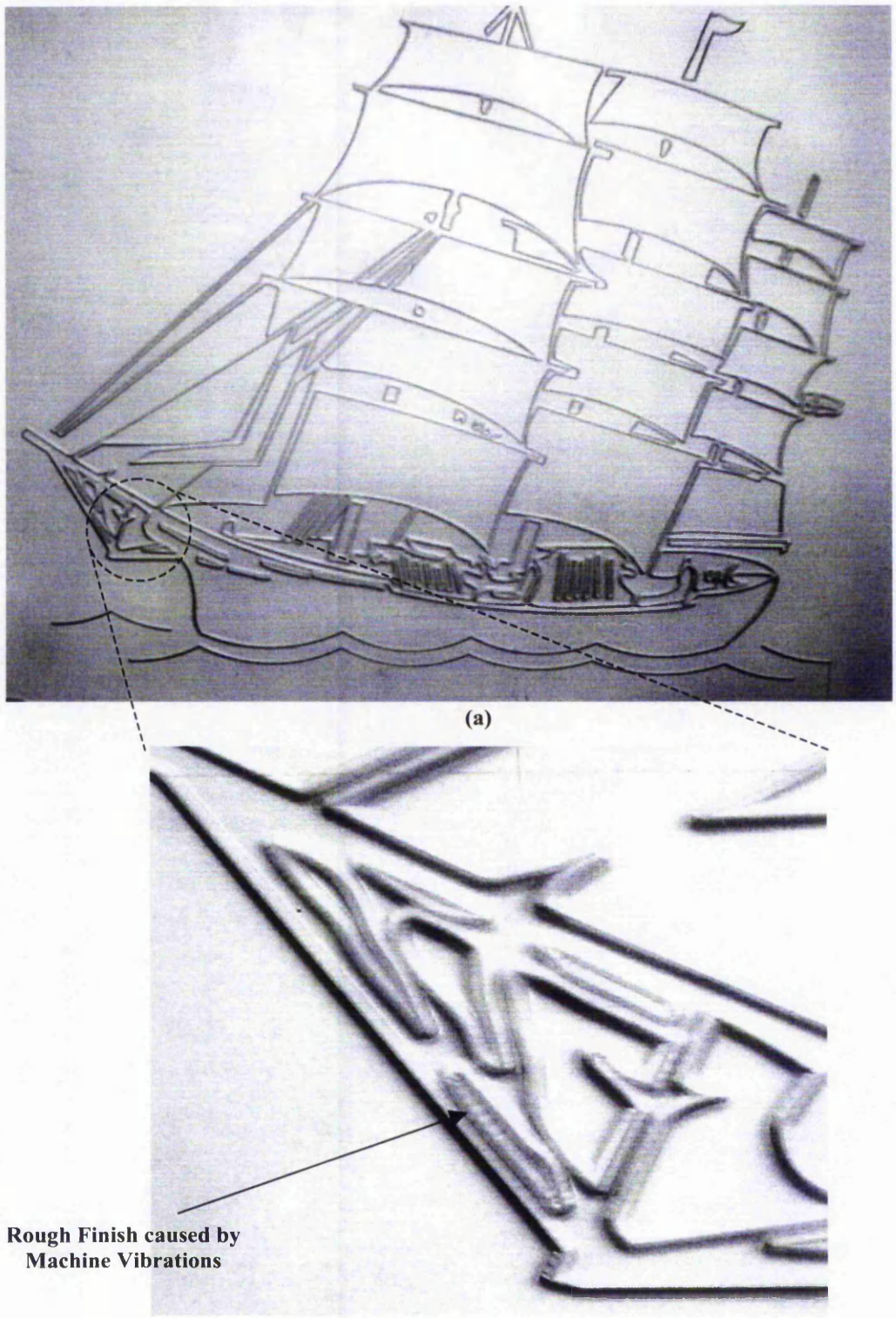


Figure 1-14: Example of Machined Shape from Pacer Ltd CNC Machine. (a) Full shape; (b) Detailed view of the shape showing rough finish.

1.3 Aims and Objectives

The work described in this thesis has investigated how to improve the motion of a multi-axis stepper motor controlled system. The aim of this research programme has been to improve the smoothness and accuracy of continuous path motion in a discrete domain.

Specifically, the aims are:

- to reduce sudden variations in pulse rate, which are a serious cause of vibrations;
- to improve the accuracy of the path following by avoiding the need for splitting the path into facets;
- to achieve smoother control of the resultant speed of the end effector.

Different components of an open-loop stepper motor controlled system have been analysed. These include the controller and the mechanical parts of the system, such as the stepper motors, etc. The research has concentrated on the controller, particularly the interpolation and the acceleration algorithms. Therefore, previous interpolation algorithms have been investigated to determine the smoothness of the resultant path following and the effect they have on the axis motion. The algorithms used by a commercial motion control system are also analysed.

Specifically, the objectives are:

- to develop understanding of the components of a stepper motor control system;
- to investigate parametrisation of curve description for smooth motion control;
- to develop algorithms to generate pulse timings for interpolation of such parameterised curves for stepper motors;
- to evaluate the interpolation algorithms at constant speed under simulation;
- to develop further algorithms for smooth acceleration and deceleration appropriate for use with the interpolation algorithms;
- to evaluate the acceleration and deceleration algorithms under simulation.
- to perform initial testing of the practical implementation of the algorithms.

The parametrisation of various curves has been investigated to assist in the development of new interpolation algorithms, which follow the path geometry closely. The pulse timings are generated according to this parametrisation technique. This is important to ensure smoothness of the train of command pulses.

Further work includes investigation of acceleration algorithms. Acceleration algorithms are important to ensure smooth motion without vibrations at the beginning or the end of the machining. Therefore, smooth acceleration and deceleration appropriate for use with the interpolation algorithms have been developed and implemented.

1.4 Thesis Outline

This thesis is structured into seven chapters. Chapter 1 provides the framework of the thesis. It gives a brief overview of Computer Numerical Control (CNC), which is used in many manufacturing systems. It discusses the historical development of manufacturing and presents some of the machining tools used in manufacturing factories today. The different components of CNC machine tools are also described. Later in the chapter, the aims of this thesis are introduced, based on the problems identified.

Chapter 2 outlines the principles of interpolation algorithms used in a machine control unit of a CNC machine. Existing algorithms are also presented in this chapter. Since the research described in this thesis concentrates on the interpolation algorithms, much emphasis is put on the advantages and disadvantages of these interpolation algorithms and how they affect the smoothness of the resultant path following. In addition, the principles of acceleration algorithms for use with high speed machining are discussed, together with the possible system architecture of such a motion control system.

Based on the results of the literature survey on the various components of the motion control system, the new interpolation algorithms for use with stepper motor controlled systems are described in Chapter 3. The algorithms developed include interpolation for straight lines and circular arcs. The principle of operation of these algorithms is presented here.

To accommodate high speed machining with the new interpolation algorithms, new linear and parabolic acceleration algorithms have been developed and implemented. The principles of these acceleration algorithms are discussed in Chapter 4. The new algorithms are compared and evaluated.

Chapter 5 outlines the simulation methods used to assist evaluation of the algorithms developed. There are four simulation methods, each with different features. These allow the demonstration of both the simulated path when using the new algorithms and a plot of the resultant position errors. One method is used to provide a plot of the axis speed.

Chapter 6 presents the results of the new algorithms from the different simulation methods developed for the evaluation platform. The results can be categorised into three categories: the interpolated path, the position errors and the variations in speed. The experiments performed include both interpolation at constant speed and interpolation including acceleration. Initial results from the practical implementation are also presented.

Chapter 7 concludes the thesis with discussion of the work conducted and the achievements of this research. Ideas for future work are also highlighted.

2 Survey of CNC Systems

This chapter presents the literature survey, which supports the Author's proposal for an improved system. The literature survey covers six distinct areas. The first, Section 2.1, includes a brief description of the machine drives and how the CNC machines can be classified. The different machine drives used are compared to justify the reason for the stepper motor being chosen as the main drive for the system.

Section 2.2 outlines the design considerations for a modern motion control system. This includes discussion of the different components contributing to the success of a machining process and a description of the most commonly used machine tools.

The interpolation algorithms already used are explained in Section 2.3. Since interpolation plays a vital role in smooth continuous path generation, special attention is given on the advantages and disadvantages of the different types of interpolation algorithm. Small errors in the interpolation algorithms can greatly affect the cut quality.

In Section 2.4, the acceleration algorithms are discussed. Because of the dynamic behaviour of the motor, it can only be given an initial start-up speed demand, which does not exceed the maximum pull-in speed. For higher speeds, the motor can be accelerated without losing synchronisation, provided that the pull-out torque limit for that particular speed is not exceeded. The pull-out torque limit depends on speed and is a characteristic of each motor or drive system.

Section 2.5 investigates the system architecture. This includes a description of the different possible system architectures employed by the industry today. This section also explains how multi-processing can be used on the machine control unit to enable a higher processing capability, and thus increase productivity.

An example of a practical stepper motor control system for smooth continuous path is also included at the end of the chapter, Section 2.6. This practical system is based on the system employed by the collaborator of this research, Axiomatic Technology [13].

2.1 CNC Systems

Computer Numerical Control (CNC) is the process of manufacturing machined parts using numerical data [21]. Production is controlled by a computerised controller, called the motion controller. The motor controller uses a motor to drive each axis of a machine tool. A program, consisting of numerical point data and motion control commands, is then loaded into the machine's computer [22].

The most common drives for machining system are the stepper motor and the servomotor. The stepper motor has the benefits of rigidity, high reliability and simplicity in construction, low cost and direct digital control [5][23]. A stepper motor controller can be implemented in open loop controlled architecture, thus reducing the cost of designing the system [24]. This type of motor is digital in nature, so, the interface is much simpler to design. They provide excellent torque at low speeds, up to 5 times the continuous torque of a brush motor of the same frame size or double the torque of the equivalent brushless motor [3]. Therefore, there has been increasing use of stepper motors in recent continuous path CNC applications. The main disadvantage is that the maximum speed depends on the pull-out torque [21].

The servomotor, on the other hand, can only be used in a closed-loop architecture [21]. Therefore, encoders have to be used to provide feedback [5] for the controller, which increases the cost. With feedback information, any errors in the machining process can be corrected in the following command. However, the servomotor can perform at higher speed than stepper motors have been able to achieve [21].

Today, the improvements in stepper motor drive design have pushed the speed limitation to a higher limit enabling it to be used for continuous path motion control system [17][25][26][27][28].

2.1.1 Open-Loop and Closed-Loop System

Closed-loop control refers to system with feedback, whereas open-loop control means that there is no feedback. In an open loop system, the controller has no information on the effect of the command pulses that it has just generated. As explained earlier, an open-loop system can only be used with a stepper motor. A stepper motor is a device that normally produces rotation through a fixed angle in response to an input pulse [29]. This causes the CNC machine to move one linear step with the help of a leadscrew. The electromagnetic effect of the combination of the stator and rotor of the stepper motor will hold the motor at the required position [29]. If a number of pulses are sent, the corresponding number of linear steps will normally be moved. The digital nature of the stepper motor makes the implementation of the drive process easier. The open-loop control system is indeed a less expensive approach when compared to its closed-loop counterpart.

An open-loop system is less costly and is easier to implement. However, the control has to be based on the assumption that the required change in position is achieved for every command pulse sent out [23][29]. Therefore, any errors are not compensated. These errors can be caused by the excitation of machine dynamics [5]. For instance, if the acceleration of the command pulses exceeds the pull-out torque limit, the result may be in loss of synchronisation (missed steps) or the motor may stall [29]. Typically, an open-loop system is only suitable for a lower torque motor, such as a stepper motor. A closed loop control system is equipped with encoders to provide the position and speed data for each axis. These data are then used as a reference to compare with the input value [21][30]. By taking the difference between the input value and the measured value, the error can be found. In the feedback process, the control program tries to reduce this error in the next command. The actuator used for a closed-loop system can be a stepper motor [31] but it is more commonly used with servomotors. In fact, a servomotor must be in a closed-loop system.

One example of a feedback device that can be used is the incremental encoder. The incremental encoder is mounted on the other end of the leadscrew and consists of a rotating disc with slots, through which light can be shone [5][21]. Light shines through

these slots and is detected with a sensor on the other side. As the disc rotates, the sensor can detect the presence or absence of the light beam. The change from absence to presence of the light beam generates a pulse. A series of pulses are generated when the disc is rotated. The angular displacement and speed can be measured by counting these pulses over time.

The main advantage of a closed-loop system is the ability to compensate for any errors by adjusting the next command. Furthermore, servomotors used in a closed-loop system have higher torque ranges when compared to stepper motors [3]. The main disadvantage that the closed-loop set-up is much more expensive than the open-loop system. Another disadvantage is that it is more complex.

2.1.2 Point-to-Point and Continuous Path Systems

With a point-to-point system, the path from the starting point to the final point is not controlled. Instead, only the numerical value of the starting and final points coordinates are provided in the part program. One example of a point-to-point system is a drilling machine. In a drilling machine, the machined object is moved until the centre of the hole to be drilled is directly beneath the drilling tool [21]. Then, the drilling tool will be moved vertically and the object is drilled to the correct depth. The exact path taken between holes does not matter.

On the other hand, with a continuous path system, the tool is usually cutting while it is moving. If the different axes are moved simultaneously, then the relative speeds determine the actual path followed [21]. If both motors move at constant speed, a straight line will be cut. For a non-linear motion, the speed for the axes will need to change during the machining process. Figure 2-1 illustrates the individual axis speed and the generated path for an anticlockwise circular-arc interpolated at constant speed.

Part (a) of Figure 2-1 illustrates the magnitude of the X-axis speed while (b) shows the magnitude of the Y-axis speed. The relative speeds for X and Y generates an

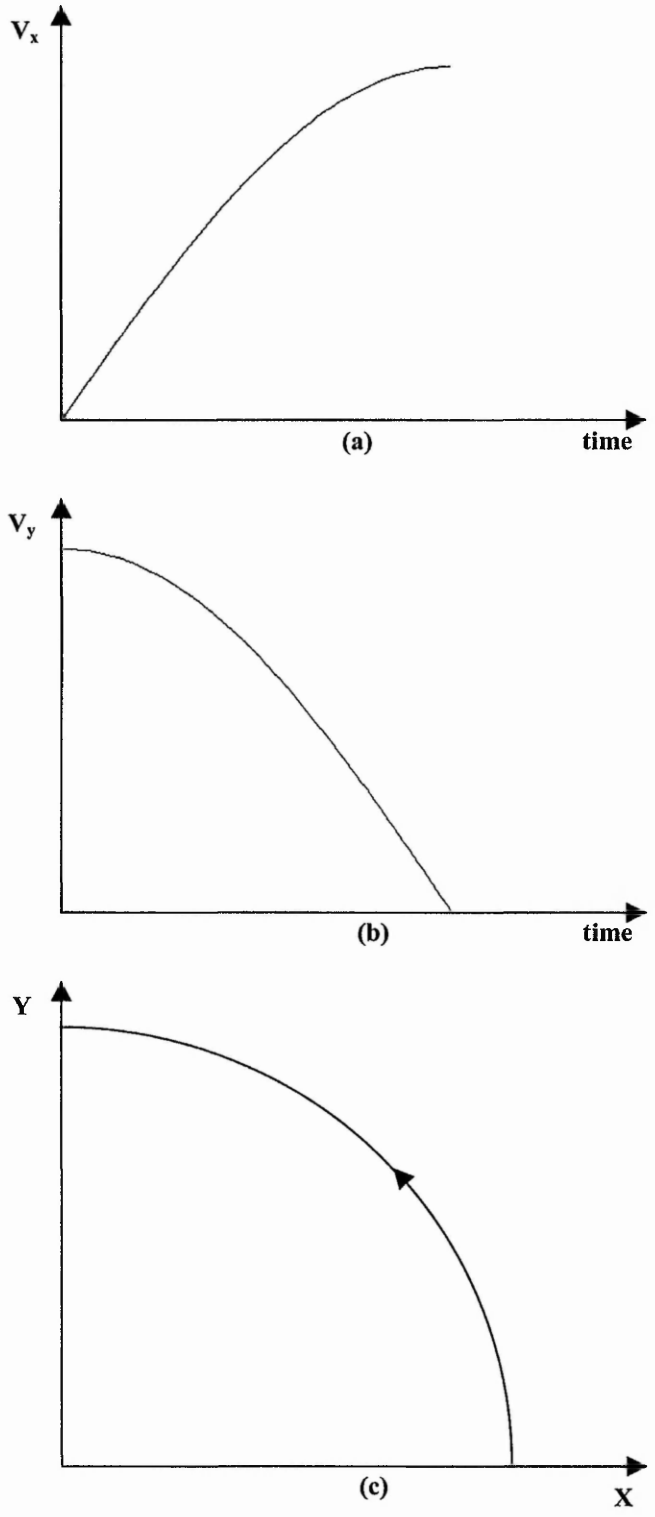


Figure 2-1: Example of Ideal Continuous Path Motion for A Circular Arc: (a) Magnitude of X speed; (b) Magnitude of Y speed; (c) Plot of Path (The direction is negative for X and positive for Y).

anticlockwise first quadrant of a circle as in (c), because they are sine and cosine waves respectively.

The programmed feedrate is the required speed for the cutting tool along the continuous path. This is used by the controller to provide appropriate command signals for each of the axes [15]. The interpolator program in the controller achieves motion appropriately between the points that are known. One example of a continuous path system is a milling machine [3].

2.1.3 Incremental and Absolute Systems

An absolute system is a system in which all motion commands are relative to one reference point [21]. This is known as the origin, or sometimes the zero point. Programmed commands refer to the position relative to this zero point.

In the incremental system, the end points for movements are specified relative to the tool's current position, rather than the zero point. Each new instruction in the part program refers to the distance from the current point to the next one.

The benefit of using an absolute system is the ease in determining the current position for any command [21]. In case of emergency, when the machine has to be stopped and the cutting tool is reset to the initial position, the current position of the cutting tool can be restored by just moving its coordinate to the required position with respect to the zero point. However, this is not possible with the incremental system, where the whole process will have to be restarted when an interruption has occurred.

In addition, the absolute system allows insertion of additional dimensional data within the part program. Any addition or modification of dimensional data does not affect the part program. Figure 2-2 illustrates a simple example of modification done to the initial machining path, where part of the inner closed curve has been removed and replaced with straight lines. The programming commands for the rest of the part program remain the same for an absolute system, including the unmodified segments of the inner curve.

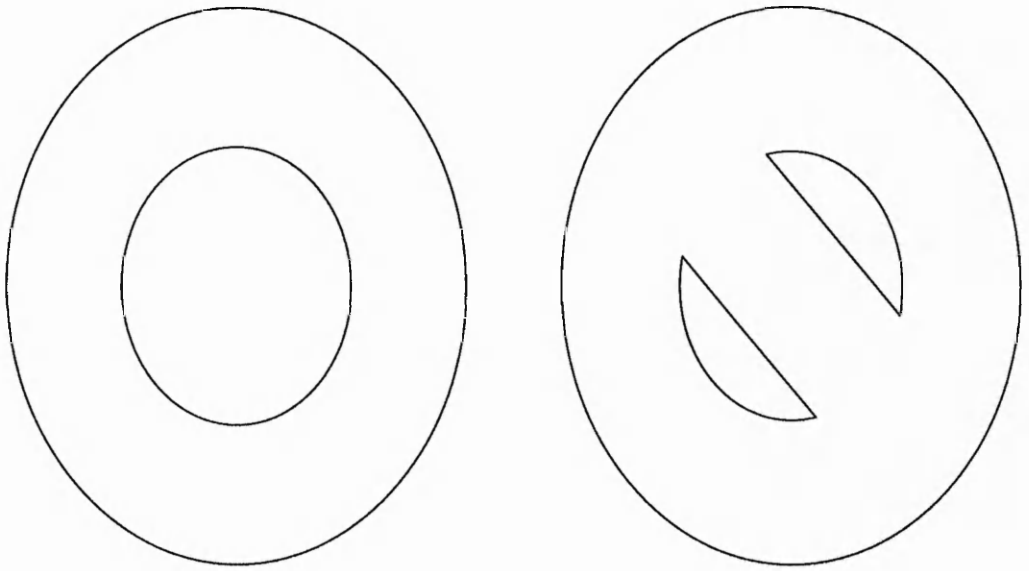


Figure 2-2: Example of Machining Path for an Absolute System: (a) Before Modification; (b) After Modification.

On the other hand, in an incremental system the part program will have to be reprogrammed from the point of the modification [21]. For the example shown above, the incremental system will require the programming commands for both the closed curves to be reprogrammed (assuming that the inner curve is machined first).

An example where it is easier to modify an incremental system is mirror image path following, where one part of the object is in symmetrical geometry to another part of the object. Only the sign of some of the corresponding commands needs to be changed. Calculation of new motion commands is not required. An example of a mirror image machining is illustrated in Figure 2-3. In an incremental system, the part program to machine Figure 2-3(b) can be similar to the part program used for machining Figure 2-3(a). The only changes will be to invert the sign for the Y-axis movements.

In practice, the tool path is offset from the desired path by the tool radius to allow for material removed during cutting. Koren [21] describes how this is performed for both the line and circular arc while Chai *et al.* [32] developed a tool-radius compensated parabolic path.

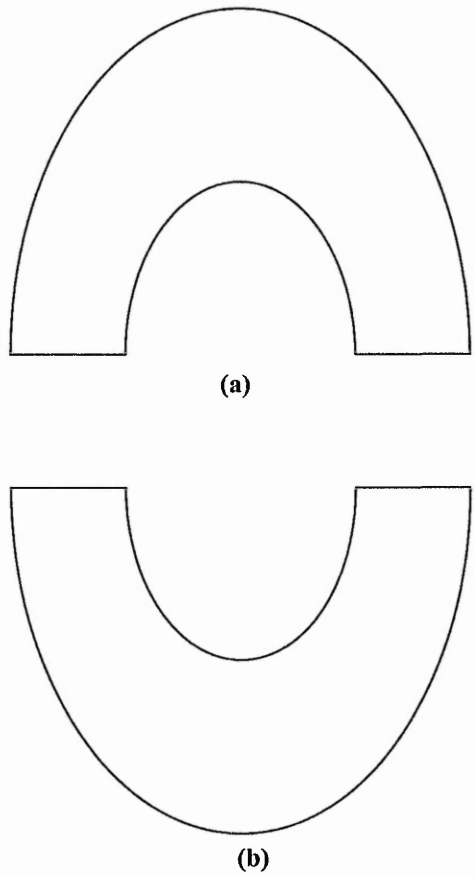


Figure 2-3: Example of Mirror Image Machining for an Incremental System. The path in (a) can be converted easily to the path in (b) by changing the sign of the movements in the Y-Axis.

2.2 Design Considerations for a Modern CNC System

After looking at the different types of CNC system and the applications involved, this section describes the different components of a CNC system.

Three critical units of a CNC system are as follows:

- Computer Aided Design / Computer Aided Manufacturing (CAD/CAM)
- Motion Controller
- Machine Tool

2.2.1 Computer Aided Design / Computer Aided Manufacturing (CAD/CAM)

Programmers use this powerful tool to simplify the task of writing and testing part programs [33]. The CAD/CAM is a technique of using a computer-aided programming language to prepare part programs.

Computer Aided Design (CAD) is a geometric modelling system used to produce engineering drawings of parts [2]. One of the important features of a CAD packages is the interactive graphics, which allows the programmer to manipulate the design directly through graphical representation of the product.

Computer Aided Manufacturing (CAM) is a way of using the computer to assist in the manufacture of a part [2]. Previously CAM systems have been designed to generate a complete word address program for machining a part on a particular CNC machine. The operations performed by the CAM system includes accepting data from CAD packages, selection of tools to be used for a particular job, specifying the feeds and speeds, and generation of the final CNC program.

Since both CAD and CAM make use of computers, their functions can be performed by the same system. CAD and CAM together create a direct link between product design

and manufacturing [6][34]. The coupling of CAD and CAM considerably shortens the time needed to bring a new product to market.

In the early years of NC programming, the part programmer extracted the appropriate dimensional data of the product to generate a sequence of program instructions combining both data and motion parameters. Today, with the advent of CAM, the part program can now be produced by computer software using the product design as input [7][35]. This permits the computer to take over most of the tedious work from the part programmer.

The system can be used for off-line checking of the program, where the resultant tool paths are graphically displayed on the computer screen. Furthermore, the system can determine the optimum tools and speeds for the material selected.

Many machines, such as Pacer machines [12], are controlled by an internal language, which will be generated by the CAD/CAM package. Thus, the commands (part program) are needed to communicate with the system motion controller. This CAD/CAM package will accept design drawing inputs that are in HPGL format and other formats.

2.2.2 Motion Controller

In the past, the part program was maintained in a storage device, normally punched tape. The information punched on the tape was inserted into the NC system by means of a tape reader. In an NC system, the tape reader reads one block of data and the instructions are then executed for that block. The same process happens for each block of data. One block of instructions corresponds to one segment of the required path. By contrast, CNC systems allow the punched tape to be read only once and then stored in the computer memory. This removed pauses between blocks of instructions. Today, the motion controller communicates with the CAD/CAM package via a parallel or serial data line, reducing the likelihood of data loss (also known as direct numerical control, DNC) [36]. The motion controller used by Pacer consists of a PC extension card [12].

The motion controller is responsible for performing the following tasks:

- Data Decoding
- Interpolation
- Acceleration and Deceleration Planning

The data received from the CAD/CAM is first decoded by the motion controller. The information extracted includes the direction of motion, the end point position and any auxiliary control signals. From the vector information, the interpolation process is executed by providing the appropriate command signals (pulses) between two successive points from the dimensional data [37]. In order words, it coordinates the motion of machine axes to generate the required machining.

The acceleration and deceleration planning process is important, in order to ensure smooth motion without too much vibration at the beginning and the end of the machining [38]. One controller may operate in conjunction with several drives and motors in a multi-axis system.

2.2.3 Machine Tool

A few of the more commonly used machine tools are described here:

Milling: Material is removed from a workpiece using a rotating cutter. Single or multiple-axis control moves can generate either simple two-dimensional patterns or more complex three-dimensional shapes [2].

Turning: Objects with rotational symmetry are produced using a cutter that moves perpendicularly through the centre plane of a rotating workpiece [2].

Wire EDM: Electrical Discharge Machining, or EDM, uses an electrical discharge from a thin wire to achieve fine cuts through hard metal parts [2]. Most EDM machines use two parallel planes in which each cutting point can move independently of the other.

This is useful in producing tapered pieces used in the production of punch dies for stamping.

Laser, Flame, and Plasma cutting: This type of machining uses a powerful light beam, a concentrated flame, or a plasma arc, to remove material [2].

Punching and nibbling: These machines are used to cut patterns in sheets of metal by the use of punch dies [2]. Repeated punches along a path achieves a nibbling effect that allows cutting of complex patterns.

2.3 Previous Interpolation Algorithms

As described in Chapter 1, the motion controller of a CNC system receives end point position information from the part program. In addition, further information about the type of continuous path to be followed is also provided. Most commonly used types are made up of linear and circular-arc segments, though higher order curve types are often available. To perform the machining, interpolation is required. Interpolation involves defining the path and rate of travel of a cutting tool when provided with a coded mathematical description of the path [39]. The motion between programmed end points of segments must be defined in order to result in smooth curves or straight lines. In other words, it must generate coordinated movement of the separate drive axes in order to achieve the desired path of the tool relative to the workpiece.

Interpolation is defined as the process of synthesising a prescribed curve from a large number of incremental steps [40]. NC systems contain hardware interpolators (which consist of digital circuits), while in CNC systems, the interpolator is mostly implemented in software. Software interpolators can be categorised into reference pulse and reference word interpolators, as explained below.

Reference pulse interpolators will output a stream of command pulses [41]. They are based upon an iterative technique controlled by an interrupt clock. A single iteration of the interpolation routine will be executed for every interrupt signal. At each stage, it is possible that an output pulse is generated. This pulse advances the corresponding machine axis by one motor step. Therefore, the maximum attainable feedrate, or the axis speed, is inversely proportional to the execution time of a single iteration. The drawback of such interpolators is that the maximum speed is limited by speed at which the computer can process the data [41].

A reference word interpolator repeatedly sends out a reference word indicating the commanded position for a certain time frame [42]. Each word normally corresponds to more than one step. Therefore, the maximum speed is not limited by the computer speed. With such interpolators, a circle is approximated with straight line segments. In

Sections 2.3.1 to 2.3.3, examples of reference pulse interpolators are described, while examples of reference word interpolators are discussed in Sections 2.3.4 and 2.3.5.

2.3.1 Digital Differential Analyser (DDA)

Digital Differential Analyser (DDA) interpolation builds upon the idea of integrating the speed to obtain the required distance [40][43][44][45]. Hardware DDA is a computer or logic circuit that uses numbers to represent analogue quantities when solving differential equations [39]. It has been found that such technique can also be employed to perform the required interpolation. In the early days of DDA interpolators, a hardware-type interpolator was normally used. It consists of a network of integrators. A schematic diagram of a DDA integrator is illustrated in Figure 2-4 and Figure 2-5 [40] the symbolic representation of a DDA integrator.

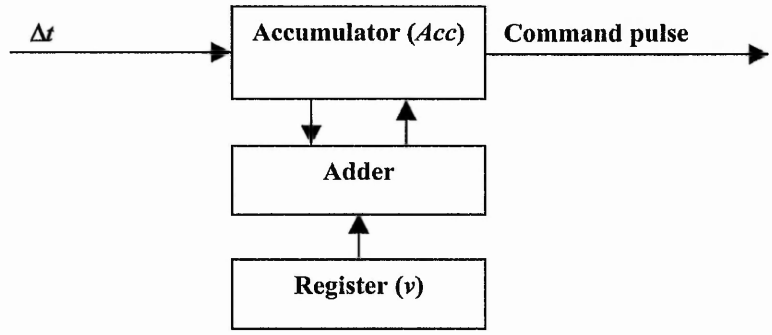


Figure 2-4: DDA Integrator.

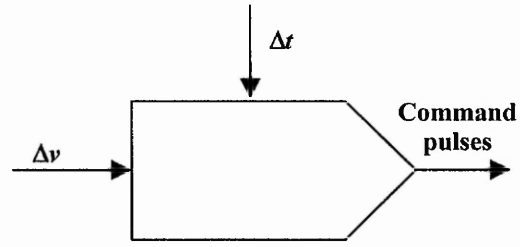


Figure 2-5: Symbol for DDA Integrator (Adapted from [40]).

The principle of operation of DDA is to perform digital integration on the speed by employing successive rectangular approximation methods [21], as illustrated in Figure 2-6 for one axis.

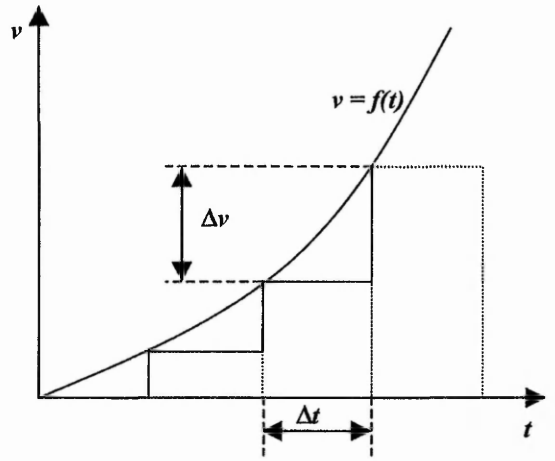


Figure 2-6: Digital Approximation of a Continuous Function (Adapted from [21]).

Let $x(t)$ be the stepper motor axis position at time t and $v(t)$ its speed. Then $x(t)$ is equal to the area under the curve $v(t)$ [21].

$$x(t) = \int_0^t v dt \tag{2-1}$$

If x_k is the value of x when $t = k\Delta t$ (after k intervals of Δt), x_k is approximated as:

$$x_k = \sum_{i=1}^{k} v_i \Delta t \tag{2-2}$$

The value of x_k in equation (2-2) can be related to the previous x_{k-1} value as follows:

$$x_k = \sum_{i=1}^{k-1} v_i \Delta t + v_k \Delta t \tag{2-3}$$

Therefore,

$$x_k = x_{k-1} + \Delta x_k \tag{2-4}$$

where $$\Delta x_k = v_k \Delta t \tag{2-5}$$

The hardware-type integrator operates in an iterative mode at a frequency f provided by an external clock, where:

$$f = \frac{1}{\Delta t} \tag{2-6}$$

The required speed, v_k, at each external interrupt clock is computed by adding the increment Δv_k to the preceding v_{k-1}, as in (2-7). The v_k value is stored in an n-bit register, limiting its range of allowable values to 2^n [40]. The variable v_k is added to the previous contents of the accumulator, Acc, as in (2-8).

$$v_k = v_{k-1} + \Delta v_k \tag{2-7}$$

$$Acc_k = Acc_{k-1} + v_k \tag{2-8}$$

If the value of Acc_k is higher than $(2^n - 1)$, an overflow is generated, which causes an axis motion command pulse.

For linear interpolation the motion can be expressed in terms of the constant velocity components, u and v, as follows:

$$\begin{aligned} x &= ut \\ y &= vt \end{aligned} \tag{2-9}$$

To design a hardware-based linear DDA interpolator, two of the DDA integrators discussed above can be used, as shown in Figure 2-7.

An example of DDA interpolated straight line is discussed below. This line is from (0,0) to (3,2) with the n value chosen to be 3, giving 2^n=8 samples. The table below lists the eight steps of the integration process.

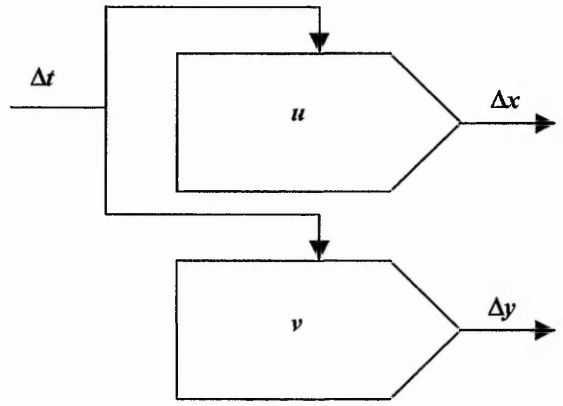


Figure 2-7: Linear DDA Interpolator.

Table 2-1: List of Interpolation Steps (Hardware DDA Interpolator).

Timing Pulse	x	Step in x	y	Step in y
Preset	0		0	
1	$\frac{3}{8}$		$\frac{2}{8}$	
2	$\frac{6}{8}$		$\frac{4}{8}$	
3	$1\frac{1}{8}$	+1	$\frac{6}{8}$	
4	$1\frac{4}{8}$		1	+1
5	$1\frac{7}{8}$		$1\frac{2}{8}$	
6	$2\frac{2}{8}$	+1	$1\frac{4}{8}$	
7	$2\frac{5}{8}$		$1\frac{6}{8}$	
8	3	+1	2	+1

Figure 2-8 is a plot of the resulting cutter path when simulated using the Zero Order simulation (which will be discussed in Chapter 5) where the machine is assumed to move the whole step instantaneously when each pulse is received. The overall speed is controlled by changing the clock frequency of the integrators.

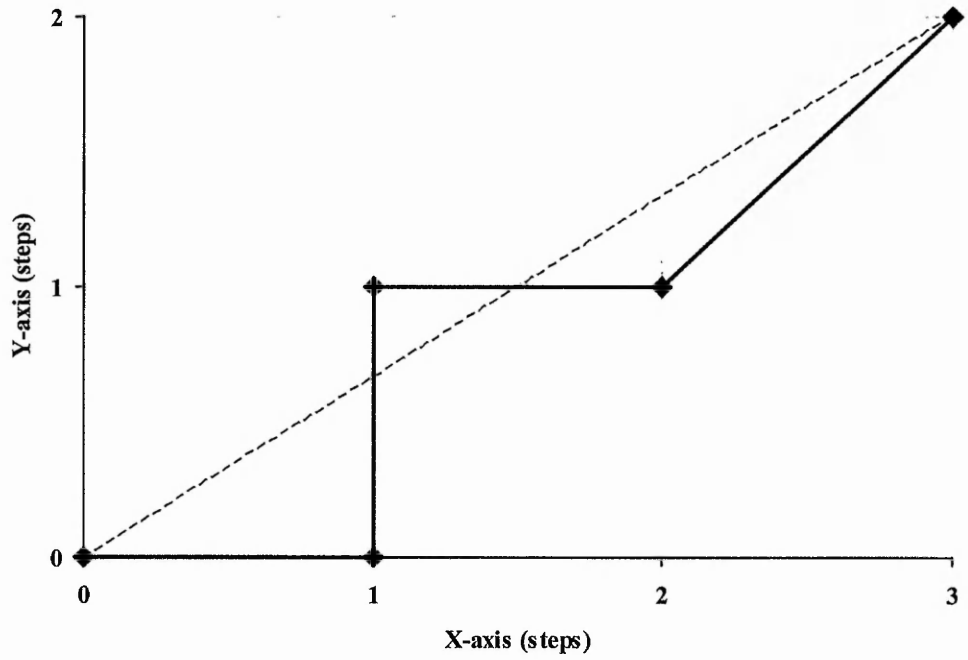


Figure 2-8: Planned DDA Linear Cutter Path (Full Line) for the Straight Line from (0,0) to (3,2) (Dashed Line). This corresponds to the steps illustrated in Table 2-1 for the DDA Hardware Interpolator.

For DDA circular arc interpolation, the speed components of the individual axis must always be generated tangentially to the arc. This circular interpolation is limited to one quadrant. For arcs that pass through more than one quadrant, this arc will be divided into successive circular arcs each within one quadrant. The discussion below is for anti-clockwise first quadrant circular arc interpolation. The circular arc centre (0,0), radius, R, can be represented as follows [21]:

$$X^2 + Y^2 = R^2 \tag{2-10}$$

where

$$X = R\cos\omega t$$

$$Y = R\sin\omega t \tag{2-11}$$

Thus, the speed commands can be calculated using differentiation:

$$V_x = \frac{dX}{dt} = -\omega R \sin \omega t$$
$$V_y = \frac{dY}{dt} = \omega R \cos \omega t \tag{2-12}$$

Thus,

$$\Delta X = -\omega R \sin \omega t \Delta t = +\Delta(R \cos \omega t)$$
$$\Delta Y = \omega R \cos \omega t \Delta t = -\Delta(-R \sin \omega t) \tag{2-13}$$

The registers are initially loaded with the following:

$$v_1 = -R \sin \omega(0)$$
$$v_2 = R \cos \omega(0) \tag{2-14}$$

The structure of the circular DDA interpolator is illustrated in Figure 2-9.

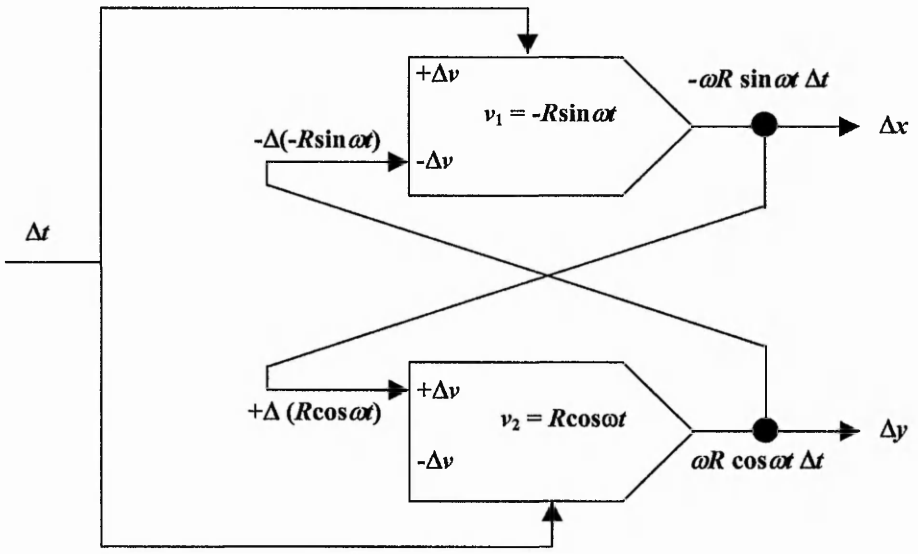


Figure 2-9: Circular DDA Interpolator (Adapted from [21]).

The software DDA interpolation [21] is also designed with a similar principle. In the hardware-based interpolator, a fixed length register (n-bit) is used. On the other hand, a

variable length register (i.e. floating point) is used in the software DDA integrator. When using a n-bit register, the ratio between the machine tool feedrate v and the frequency of the interpolator is shown below:

$$\frac{v}{f} = \frac{L}{2^n} \tag{2-15}$$

where L is the length of the line and f is the interrupt frequency.

For the software DDA interpolation, floating point values are used and the 2^n in equation (2-15) can be replaced by the actual length, L. Therefore,

$$v = f \tag{2-16}$$

The interrupt frequency is the required feedrate. The software implementations provide a higher degree of flexibility. Therefore, the progressive increase in microprocessor performance favours the software implementation of the algorithm. For the DDA interpolation used for comparison later in this thesis, the software implementation is used. The main advantage of the circular DDA interpolator is the constant resultant speed along a circular path. The disadvantage of such an interpolator is the cumulative position error, which results in the circular path not reaching its end point exactly. The cumulative position error is caused by the approximation used as follows:

$$\begin{aligned} x_{i+1} &= x_i + v_x \Delta t \\ y_{i+1} &= y_i + v_y \Delta t \end{aligned} \tag{2-17}$$

For a straight line v_x and v_y are constant, so there is no error, but for a circular arc they change over time.

2.3.2 Search-Step

The Search-Step interpolation algorithm is mainly used for open-loop continuous path motion control system [4]. The Search-Step algorithm is based upon the implicit representation of a curve. The curve is defined by a non-parametric equation of the form $f(x,y)=0$. In other words, all points lying on the required curve satisfy the equation $f(x,y)=0$, while points not lying on the curve will have a non-zero value. The sign of the non-zero value normally depends on which side of the curve the point is lying and this value is normally a good enough measure of closeness to the curve at that particular point [40][46].

Interpolation proceeds by moving the cutting tool along a single coordinate axis at a time. There are four possible direction of interpolation for the next interpolated point, X+, X-, Y+ and Y- [47]. As described earlier, points not lying on the line or curve will have a non-zero value for $f(x,y)$. Therefore, the next interpolation step is chosen to be the step movement that will try to switch the sign of the value of $f(x,y)$. For instance, if the current interpolated point has a positive value for $f(x,y)$, the next interpolated point will be chosen to be the one with a negative value. The direction will also be considered.

Linear interpolation with the Search-Step algorithm is based on the straight line function:

$$\begin{aligned} f(x,y) &= (x_e - x_s)y - (y_e - y_s)x \\ &= \Delta X y - x\Delta Y \end{aligned} \tag{2-18}$$

where (x_s, y_s) is the start point and (x_e, y_e) is the end point. ΔX and ΔY are constant for any particular line. First quadrant linear interpolation is considered in the following discussion. Taking a step forward in the X-axis,

$$\begin{aligned} f(x+1,y) &= \Delta X y - (x+1)\Delta Y \\ &= \Delta X y - x\Delta Y - \Delta Y \\ &= f(x,y) - \Delta Y \end{aligned} \tag{2-19}$$

Similarly, taking a step backward in the X-axis,

$$f(x-1,y) = f(x,y) + \Delta Y \tag{2-20}$$

When a motor step is moved in the positive direction of the Y-axis,

$$\begin{aligned} f(x,y+1) &= \Delta X(y+1) - x\Delta Y \\ &= \Delta Xy + \Delta X - x\Delta Y \\ &= f(x,y) + \Delta X \end{aligned} \tag{2-21}$$

Similarly, when a motor step is moved in the negative direction of the Y-axis,

$$f(x,y-1) = f(x,y) - \Delta X \tag{2-22}$$

As an example, a cutting tool is moved from the initial position (0,0) along a straight line until position (3,2). For this example, $f(x,y) = 3y - 2x$ and $\Delta X = 3$ and $\Delta Y = 2$. Results of the computed steps are listed in Table 2-2. The resulting cutter path (Zero Order simulation) is illustrated in Figure 2-10.

Table 2-2: List of Interpolation Steps (Search-Step).

Step	$f(x,y)$	X step	Y step
0	0	0	0
1	-2	+1	0
2	+1	0	+1
3	-1	+1	0
4	+2	0	+1
5	0	+1	0

Circular interpolation using the Search-Step algorithm is based on the implicit representation of the circle. For the example of a circle with centre (0,0) and radius r,

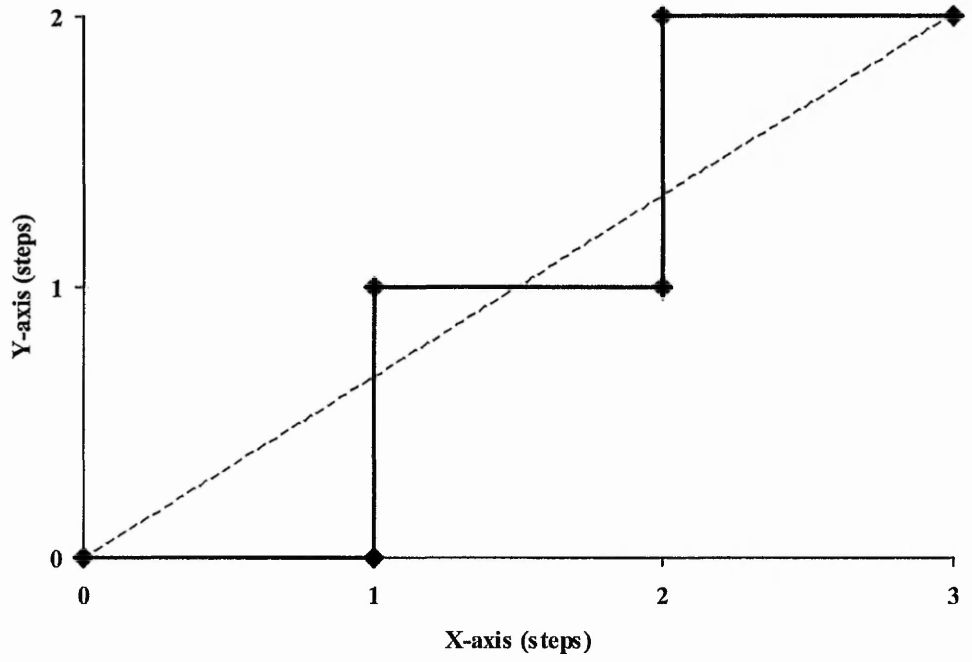


Figure 2-10: Planned Search-Step Linear Cutter Path (Full Line) for Straight Line from (0,0) to (3,2) (Dashed Line). This corresponds to the steps illustrated in Table 2-2.

$f(x,y) = x^2 + y^2 - r^2 = 0$. The following discussion is for anticlockwise first quadrant circular interpolation. After one positive step in the x direction,

$$\begin{aligned} f(x+1,y) &= (x+1)^2 + y^2 - r^2 \\ &= x^2 + 2x + 1 + y^2 - r^2 \\ &= f(x,y) + 2x + 1 \end{aligned} \tag{2-23}$$

Similarly, after one negative step in the x direction results in

$$f(x-1,y) = f(x,y) - 2x + 1 \tag{2-24}$$

On the other hand, when moving one step in the positive y direction,

$$\begin{aligned} f(x,y+1) &= x^2 + (y+1)^2 - r^2 \\ &= x^2 + y^2 + 2y + 1 - r^2 \\ &= f(x,y) + 2y + 1 \end{aligned} \tag{2-25}$$

When moving one motor step in the negative y direction,

$$f(x, y-1) = f(x, y) - 2y + 1 \tag{2-26}$$

With the Search-Step algorithm, the position errors are kept within one stepper motor step. It has a higher accuracy when following a circular path compared to the DDA algorithm. Unlike the DDA, Search-Step does not allow for constant speed during interpolation. In fact, for circular-arc Search-Step interpolation, there is a speed variation up to a factor of $\sqrt{2}$ around the curve [40]. Furthermore, each movement is constrained to one of four possible directions (parallel to one of the 2 axes). Pak *et al.* [48] suggest a special pulse filtering algorithm to solve the speed problems. Their work is mostly concerned with the approximation of the resultant speed rather than the precise speed value.

2.3.3 Direct-Search Interpolation

For Search-Step Interpolation described in the previous section, there are only four possible directions of interpolation (parallel to one of the axes). In other words, only one axis is allowed to move at any instant. The interpolated path will follow the required path more closely if it allows for diagonal interpolation to the next interpolated point. This means that the interpolation will involve movement in both axes.

The principle of operation of the Direct-Search algorithm (an improvement to the Search-Step algorithm) is similar to the Search-Step interpolation [49][50]. It allows eight different directions of motion, because diagonal movement is also allowed. Like Search-Step algorithm, it still suffer a variation of speed up to a factor of $\sqrt{2}$. The $f(x,y)$ function in the Search-Step algorithm is used as an indication of the distance of the interpolated points from the required line or curve. When $f(x,y)$ is zero, the interpolated point falls on the line or curve. The first step is to determine the direction of interpolation. With this information, the number of possible interpolated points has been reduced to three, movement in X, Y or both axes together. For each possible new point (x, y) the path error is calculated. The one yielding the least path error will be applied.

The value for $f(x,y)$ is used as an indication of the error. The three possible interpolated step results in the following changes in the errors:

Step in X-axis, $$f(x+1,y) = f(x,y) + \Delta f_x \quad (2\text{-}27)$$

Step in Y-axis, $$f(x,y+1) = f(x,y) + \Delta f_y \quad (2\text{-}28)$$

Biaxial step, $$f(x+1,y+1) = f(x,y) + \Delta f_x + \Delta f_y \quad (2\text{-}29)$$

where

$$\Delta f_x = -(y_e - y_s), \text{ and} \quad (2\text{-}30)$$

$$\Delta f_y = x_e - x_s \quad (2\text{-}31)$$

for linear interpolation. On the other hand, for circular arc interpolation,

$$\Delta f_x = 2x+1, \text{ and} \quad (2\text{-}32)$$

$$\Delta f_y = 2y+1 \quad (2\text{-}33)$$

An example of path (with Zero Order simulation) is illustrated in Figure 2-11.

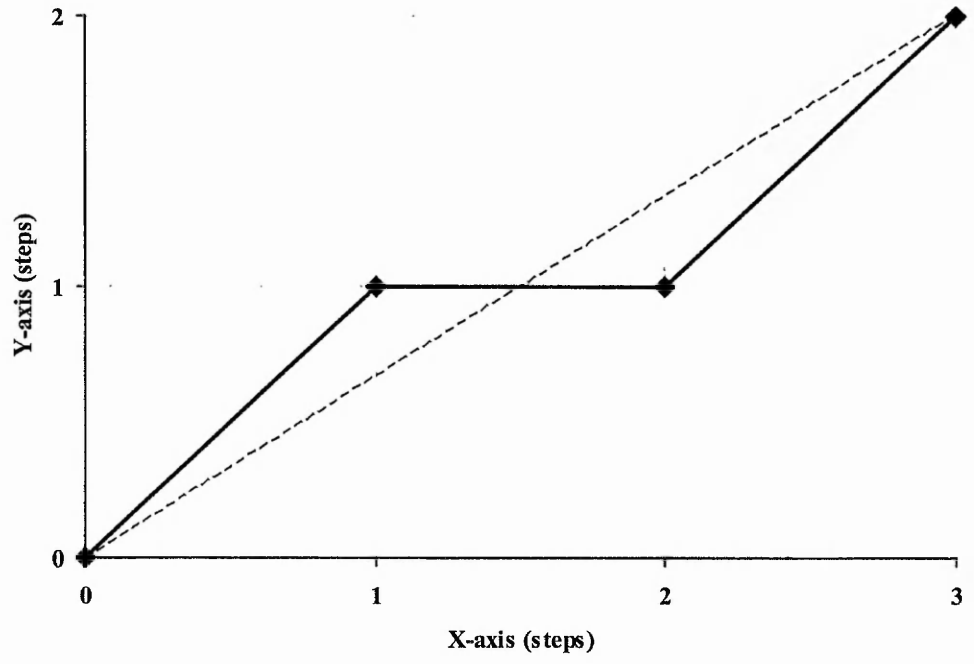


Figure 2-11: Planned Direct-Search Linear Cutter Path (Full Line) for Straight Line from (0,0) to (3,2) (Dashed Line).

2.3.4 Bresenham Interpolation

Bresenham interpolation is often used in computer control of digital plotters. The working area is divided into small squares. The side length of the square is normally selected to be the shortest possible incremental movement. A typical mesh size is 1/100th of an inch [51]. The plotter can move linearly from a point on a mesh to any adjacent point on the mesh. This allows eight possible directions of motion like Direct Search interpolation. The data to be plotted are expressed in an (x,y) rectangular coordinate system which has been scaled with respect to the mesh.

To perform Bresenham interpolation, the octant of the required path has to be first determined. By doing this, the number of possible motions is reduced to two. The example in Figure 2-12 shows the interpolation of a line in the first octant starting from (0,0) to (3,2) (Zero Order simulation). The two possible directions of interpolation are labelled M1 and M2. The selection of either M1 or M2 direction depends on the perpendicular distances of the two possible mesh cross points to the actual path [51]. The path with the shorter distance will be selected. For example for the first step, p_1 is chosen as the first step movement instead of p_2 because of its shorter perpendicular distance. For straight lines, the same result is obtained as with the Direct Search algorithm. The two algorithms work on the same principle (finding points with smallest position error) but using a slightly different approach for calculating the error. For higher order curves, the curve has to be broken up into short straight lines before performing the Bresenham interpolation.

Figure 2-13 illustrates how the two possible directions of interpolation are determined when the octant of interpolation is known. As in the case of Search-Step interpolation, the Bresenham algorithm suffers from variation of resultant speed. When travelling along a circular arc, the speed variation can be up to a factor of $\sqrt{2}$ [4]. Liu *et al.* [52] have extended the Bresenham's algorithm to spatial straight lines.

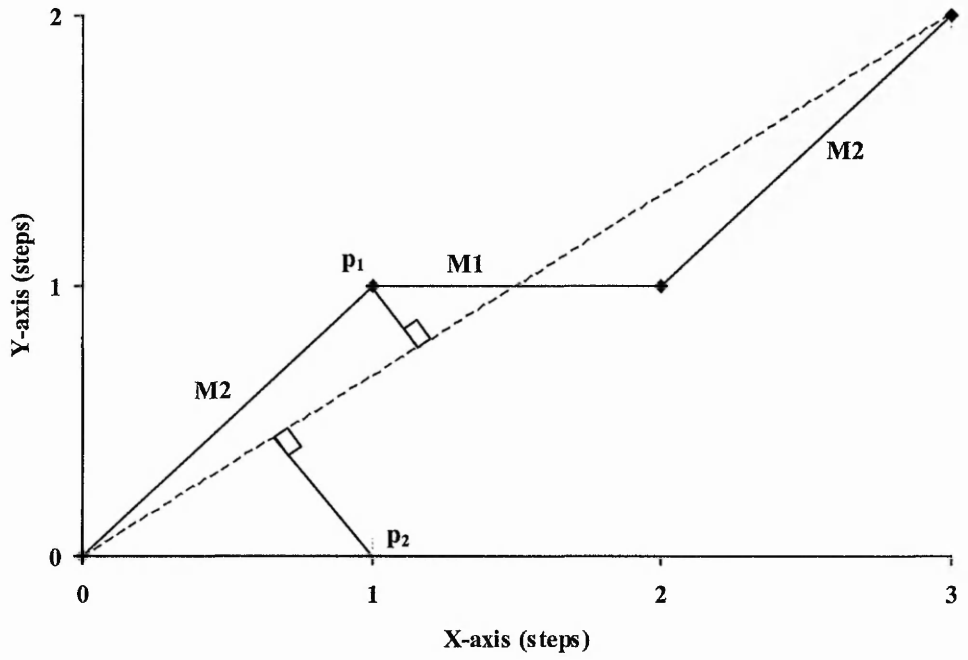


Figure 2-12: Planned Bresenham Linear Cutter Path (Full Line) for Straight Line from (0,0) to (3,2) (Dashed Line).

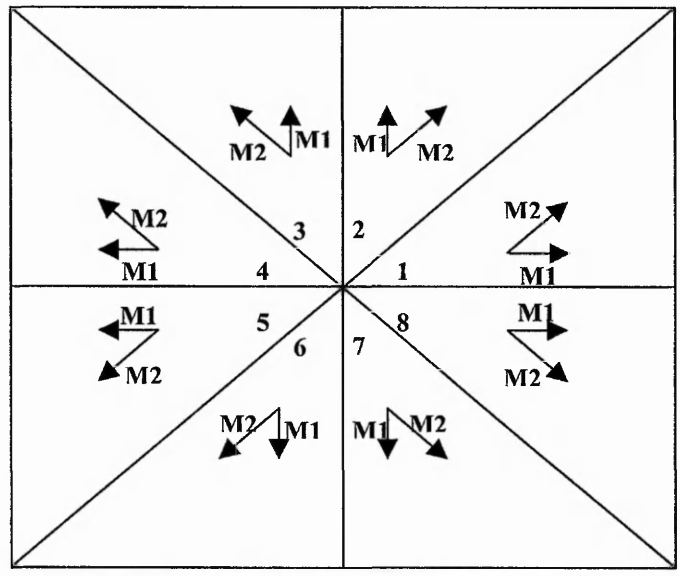


Figure 2-13: Orientation of Possible Movements Relative to the Axes.

2.3.5 Parametric Interpolation

With increasing demand for cutting of complex paths, parametric interpolation has been developed in recent years [53][54]. Complex paths can be divided into fewer segments. Compared to the conventional interpolators (where one parametric polynomial segment may become 10 linear segments), each segment can be expressed as a parametric polynomial [55]. Thus, the size of the memory required to store the segments information is reduced.

When interpolating a complex path with linear segments, there might be sudden changes in actual path direction at the junctions. This can cause vibrations and work has been done to reduce this. On the other hand, a parametric interpolator distributes the change of direction more evenly along every segment. The drawbacks are high computation power and failure to follow the entire path with constant feedrate [56].

For a cutter path $P(u)$ in two-dimensional space which can be represented as $P(u) = (x(u), y(u))$, where u is the parameter. A set of two parametric equations define $x(u)$ and $y(u)$. For example, for a parametric cubic curve, $x(u)$ and $y(u)$ are of cubic form and are given by:

$$\begin{aligned} x(u) &= a_x u^3 + b_x u^2 + c_x u + d_x \\ y(u) &= a_y u^3 + b_y u^2 + c_y u + d_y \end{aligned} \tag{2-34}$$

where a_x, b_x, c_x, d_x, a_y, b_y, c_y and d_y are constant coefficients.

The velocity along this curve, $\mathbf{V}(u)$, is expressed as:

$$\mathbf{V}(u) = \frac{dP(u)}{dt} = \frac{dP(u)}{du}\frac{du}{dt} = \left(\frac{dx}{du}\mathbf{i} + \frac{dy}{du}\mathbf{j}\right)\frac{du}{dt} \tag{2-35}$$

where $\mathbf{i}$ and $\mathbf{j}$ are the x and y components.

The function du/dt in the above equation defines the relationship between the static information represented in CAD model and the dynamic information needed to control

the CNC machine tool. Taking the magnitude of the velocity and rearranging the equation (2-19), an equation for du/dt is obtained as follows [57]:

$$|\mathbf{V}(u)| = \sqrt{\left(\frac{dx}{du}\right)^2 + \left(\frac{dy}{du}\right)^2}\,\frac{du}{dt} \tag{2-36}$$

$$\frac{du}{dt} = \frac{|\mathbf{V}(u)|}{\sqrt{\left(\frac{dx}{du}\right)^2 + \left(\frac{dy}{du}\right)^2}} \tag{2-37}$$

Therefore, from the desired cutting speed $|V(u)|$ and the path, the value of du/dt can be found. The interpolation is performed by generating intermediate parameter values at each time sample, which is then used to determine the position of the next interpolated point. The rate of change of the parameter, du/dt, is used in a first order approximation to obtain the subsequent parameter value [57][58].

$$u_{i+1} = u_i + \frac{du}{dt}\Delta t \tag{2-38}$$

The coordinates of the interpolated point are calculated by substituting u_{i+1} from equation (2-38) into equation (2-34). Equation (2-38) is executed recursively until the desired path is fully interpolated. The first-order approximation of Taylor series used here has introduced errors resulting in not being able to follow the path very close to the desired speed. The introduction of second-order approximation [18] can be a remedy to the problem but it suffers from high demand for processing power. Another alternative is to use a feedback interpolator developed by Lo [59].

2.3.6 Parametric B-Spline Interpolation

When approximating a high degree curve with straight lines and circular arcs, one of the problems introduced is the certain jump of curvature at the junction between segments [60]. With cubic spline interpolation (or higher degree), the curvature at the junction is kept continuous [60][61]. Cubic spline interpolation is made up of cubic segments. Two conditions are also set at the beginning and end whilst generating the spline [62]. For

the spline, the number of data points increases heavily in regions with large curvature. In such cases, for interpolation with lines and arcs a large number of segments is usually also required. This is particularly true if it is required to machine this section using linear interpolation. A spline curve represents the path with a smaller number of points. However, spline interpolation algorithms usually require a postprocessing unit and powerful microcomputer architecture for the machine controller. Therefore, spline interpolation is most often used in surface machining, where the sheer amount of data justifies a complex calculation algorithm. Many machine tool controllers which claim to support spline interpolation, employ a two stage interpolator. The first stage involves a coarse grain interpolator, which performs the spline interpolation. The output data (line segments) are then fed into a fine grain interpolator which usually employs the linear interpolation algorithms.

A B-Spline curve is a type of spline where the shape of the curve is controlled by a series of data points called control points or control vertices [62][61][63]. One particular property of the B-Spline curve is local control [64], by which altering a single vertex will only change the shape of a small part of the curve. An example of B-Spline interpolation is discussed below for the uniform cubic B-Spline. A cubic B-Spline is generated by a weighted sum of basis functions as follows [65]:

$$Q_i(u) = \sum_{r=-3}^{r=0} V_{i+r} B_{i+r}(u) \tag{2-39}$$

For the uniform B-Spline curve, $u_{i+1} = u_i + 1$, so the equations to define $B_{i-3}(u)$, $B_{i-2}(u)$, $B_{i-1}(u)$ and $B_{i-0}(u)$, can be replaced by $b_{-0}(i)$, $b_{-1}(i)$, $b_{-2}(i)$ and $b_{-3}(i)$ which are the 4 polynomials used to define all uniform cubic B-Splines. Each segment is re-parameterised with i where $0 \leq i \leq 1$ [62].

$$\begin{aligned}
B_{i-0}(u) &= b_{-0}(i) = \frac{1}{6} i^3 \\
B_{i-1}(u) &= b_{-1}(i) = \frac{1}{6}(1 + 3i + 3i^2 - 3i^3) \\
B_{i-2}(u) &= b_{-2}(i) = \frac{1}{6}(4 - 6i^2 + 3i^3) \\
B_{i-3}(u) &= b_{-3}(i) = \frac{1}{6}(1 - 3i + 3i^2 - i^3)
\end{aligned} \tag{2-40}$$

To perform the interpolation, the B-Spline is treated by considering each of the cubic polynomial segments in turn. The coefficients of the *k*-th cubic polynomial segment are calculated using the value of the control points/vertices [66].

$$Q_k(u) = a_k + b_k u + c_k u^2 + d_k u^3 \tag{2-41}$$

where

$$a_k = \frac{1}{6}(V_{k+1} + 4V_k + V_{k-1}),$$

$$b_k = \frac{1}{2}(V_{k+1} - V_{k-1}),$$

$$c_k = \frac{1}{2}(V_{k+1} - 2V_k + V_{k-1}),$$

$$d_k = \frac{1}{6}(V_{k+2} - 3V_{k+1} + 3V_k - V_{k-1})$$

S. Bedi [66] has used the forward difference algorithm to implement his B-Spline interpolation in software. This algorithm is used to interpolate the cubic polynomial segments into straight lines. The requirement is often to interpolate the given points [60] and normally a B-Spline curve with the given points as control points is unlikely to have the curve passing through those points. New control points for a B-Spline curve passing through the given points can be found by using the given points as B-Spline control points and then repeatedly modifying them until the curve passes through the original given points within the required tolerance.

2.4 Acceleration Algorithms

Continuous path motion involves simultaneous motion of multiple axes to machine the desired curve. Each axis is controlled by an individual stepper motor. Due to the dynamic behaviour of the motor, it can only be given an initial start-up speed demand, provided that speed does not exceed a maximum value, known as pull-in speed. For higher speeds, the motor can be accelerated without losing step (i.e. losing synchronisation), provided that the pull-out torque limit for that particular speed is not exceeded [29]. The pull-out torque limit is dependant on speed and is a characteristic of each motor/drive system. An example of a pull-out torque characteristic of stepper motor system is illustrated in Figure 2-14.

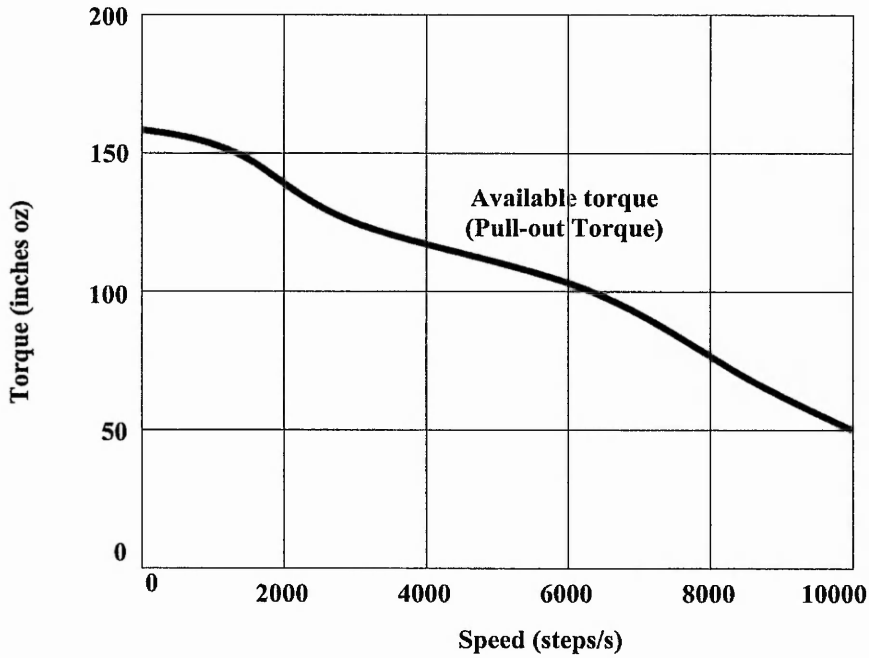


Figure 2-14: Pull-Out Torque Characteristics of a Typical Stepper Motor (Adapted from [72]).

The raw data for acceleration and desired feedrate is pre-processed by the CAD/CAM unit. The information contained in the command language for the motion control system only provides the overall requirement for the system. The motion control unit in turn has to calculate the acceleration and deceleration values in fixed time or distance

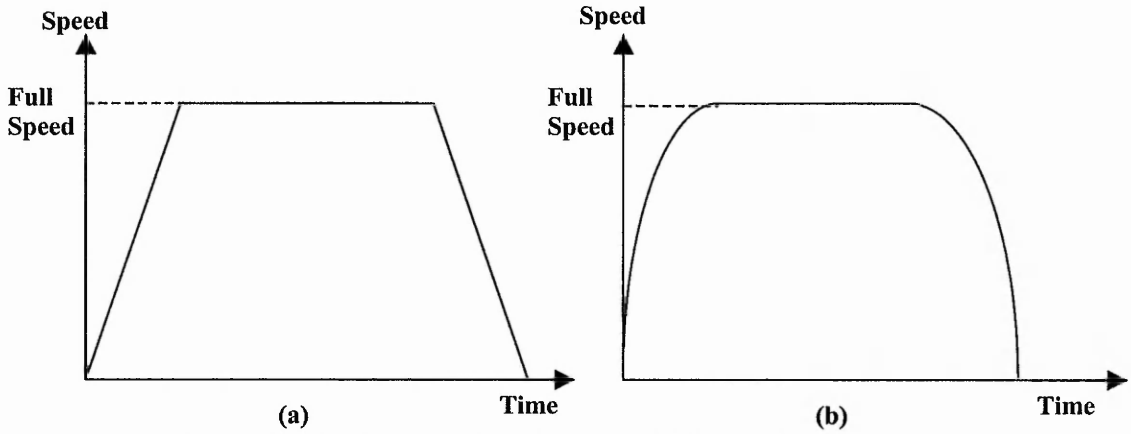


Figure 2-15: (a) Linear Acceleration; (b) Parabolic Acceleration.

intervals. Two of the most commonly used acceleration algorithms are linear acceleration (also sometimes referred as the trapezoidal profile) and parabolic acceleration [72][67][68]. For these algorithms, the speed varies with time either linearly or quadratically, respectively, as shown in Figure 2-15.

The speed and distance value are then calculated for the whole design or for a defined block [4]. The latter requires a look-ahead system, which ensures that the system can stop at the end of every block. As stated earlier, the linear and parabolic acceleration algorithms are most commonly used, so they will be discussed in more detail in the following sections.

The cutting speed, maximum fast feedrate and rate of acceleration are provided to the motion control unit. This information is precalculated by the CAM software tool using a knowledge based database system [5]. The controller will then calculate the number of steps required along a path segment for the master axis (axis with the highest speed) using an interpolation algorithm. Based on this information, the number of steps for acceleration, constant and deceleration can be determined.

2.4.1 Linear Acceleration

Linear acceleration is the most commonly used acceleration algorithm where the acceleration and deceleration phases of the speed profile are taken to be linear. Thus,

the speed changes linearly in these phases, so the acceleration or deceleration rate remains constant, as shown in Figure 2-15(a).

The linear acceleration (or trapezoidal profiling) is normally based on a master-slave axes relationship when used in a continuous path stepper motor controlled motion control system [13]. The master (or primary) axis is the axis with the highest speed during a particular segment of interpolation. This speed will be used as a reference to calculate the speed for other slave (or secondary) axes. This can cause reduced surface quality.

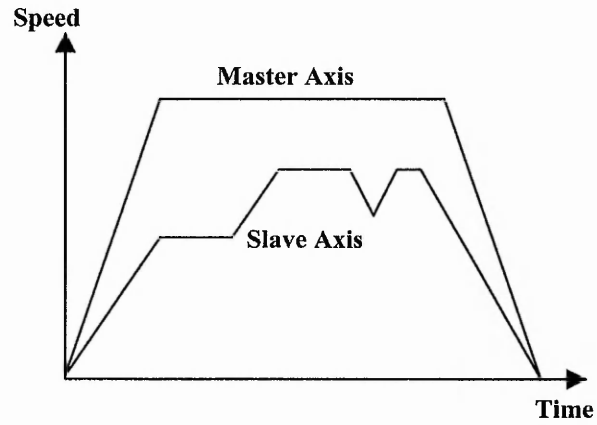


Figure 2-16: Linear Acceleration (Trapezoidal Profile).

As explained in the previous section, each stepper motor has a maximum starting and stopping speed (pull-in speed), which is dependent on the rotor inertia and the load inertia. The motor cannot be started or stopped above the pull-in speed without losing steps. In practical applications, this speed is not sufficiently fast to use for the whole shape. Therefore, there is a need for the acceleration algorithms (motion profiling). From the starting speed to the required speed is where the linear acceleration occurs. One way of implementing the linear acceleration is by linear ramping, where a block of pulses is maintained at a constant speed before the next higher constant speed block of pulses are generated. The acceleration algorithm generates the required speed profile from a set of four motion parameters. They are the base speed, maximum speed, acceleration rate and deceleration rate [69][70][71]. These motion parameters are determined in advance in a CAM software tool. These parameters are calculated taking into account the material using a knowledge-based database system [4]. This

information is included in the command language used (G-code, APT, etc) and then transferred to the machine tool controller unit.

The number of steps required for the primary axis for the required path segment is first calculated. An acceleration (look-up) table is then generated from the motion parameters information and the number of steps in the master axis [4]. A programmable timer is then used to perform the linear ramping. It is used to generate a pulse train whose frequency depends on a timer preload register. The value to be preloaded into the register is obtained from the acceleration table. The layout of the programmer timer is illustrated in Figure 2-17.

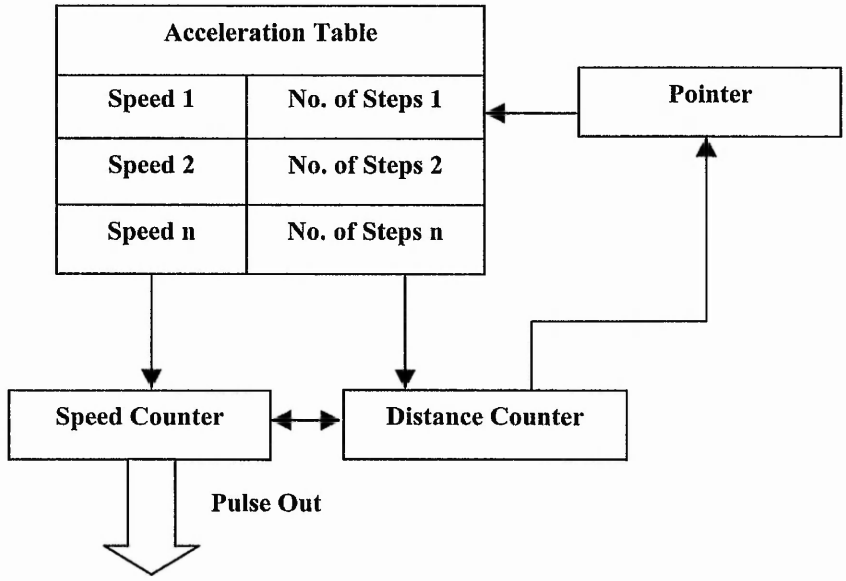


Figure 2-17: Layout of a Programmable Timer (Adapted from [70]).

The equation to calculate the number of motor steps in each time slot during acceleration or deceleration is shown in (2-42) [70].

$$S_i = \frac{V_m^2 i}{n^2 A} \tag{2-42}$$

where i = 1,2,…,n

A = Acceleration or Deceleration rate

V_m = Maximum speed

n = number of speed steps during acceleration or deceleration

These motor steps are controlled by the microprocessor. The number of discrete speed steps, n, should be large enough so that the step change in speed is not too large. The number of motor steps during the constant phase (at speed n) is given by (2-43) [70].

$$S_n = S_T - 2(S_1 + S_2 + ... + S_{n-1}) \tag{2-43}$$

where S_T is the total number of motor steps.

Since a certain number of motor steps will be made at a particular speed before incrementing or decrementing the speed, the velocity profile for the master axis will resemble the graph in Figure 2-18. In these applications, the graph is not actually linear but a series of steps.

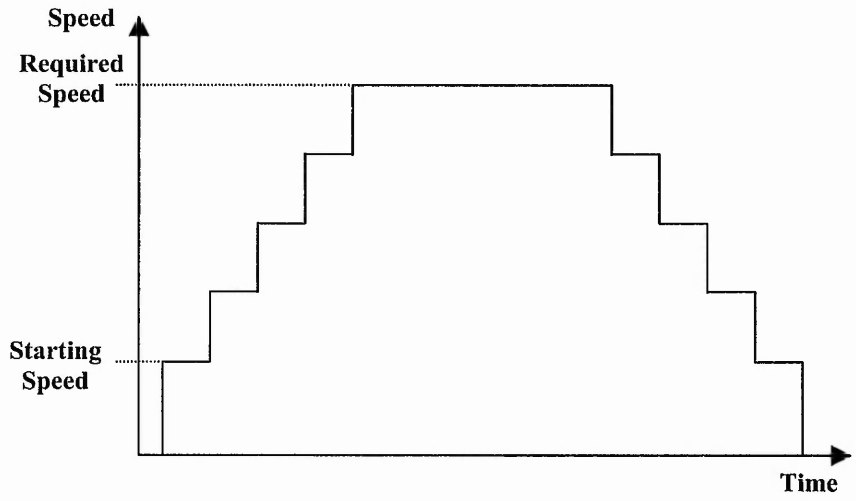


Figure 2-18: A Simplified Speed Profile for Master Axis for a Typical Commercial CNC System (see Section 2.6).

Linear acceleration results in slow acceleration and much of the available torque is not utilised. Linear acceleration in steps is the most basic, but generally results in a longer than desirable acceleration time [72] and causes vibrations [5]. Figure 2-19 illustrates an example of how vibration is caused by linear ramping.

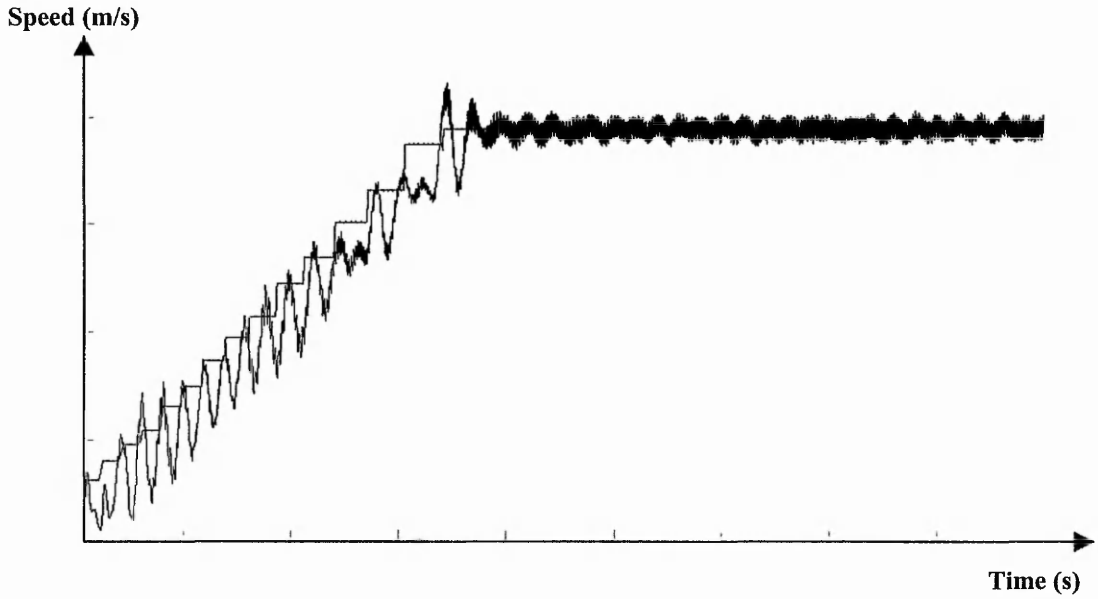


Figure 2-19: Example of Vibrations Caused by Linear Ramping (Adapted from [5]) (also shown in Figure 1-13).

2.4.2 Parabolic Acceleration

Parabolic acceleration is a technique used in a high performance stepper motor controller that allows better utilisation of available torque. The parabolic shaped acceleration graph can be seen in Figure 2-20. The 'speed profile' starts off at the standard set starting speed and then, as the motor begins to move, the speed increases following a parabolic shape. This also means that the acceleration is decreasing linearly throughout the acceleration phase, unlike the case of linear acceleration where the acceleration is constant.

A parabolic acceleration algorithm allows a higher acceleration at low motor speed and a lower rate at high speed. With this method, much more of the available motor torque can be utilised and the stepper motors can be used at higher speeds, Figure 2-21.

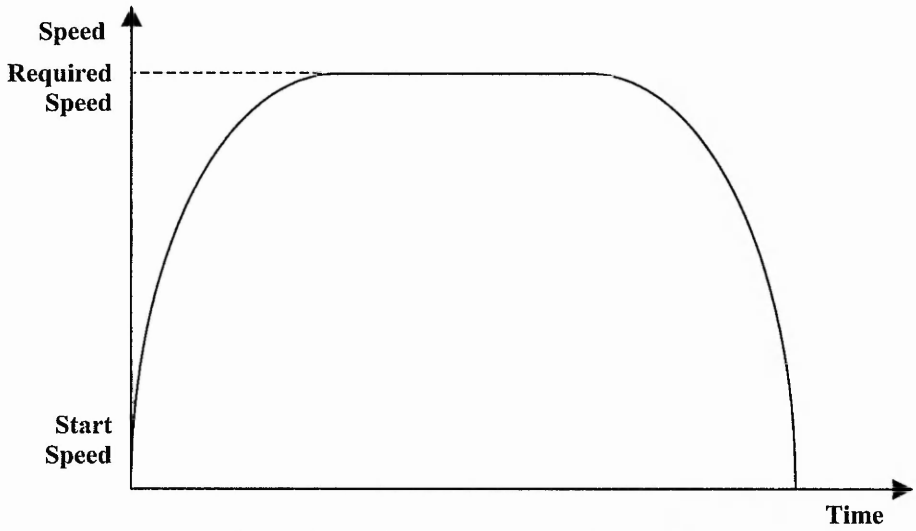


Figure 2-20: Parabolic 'Speed Profile'.

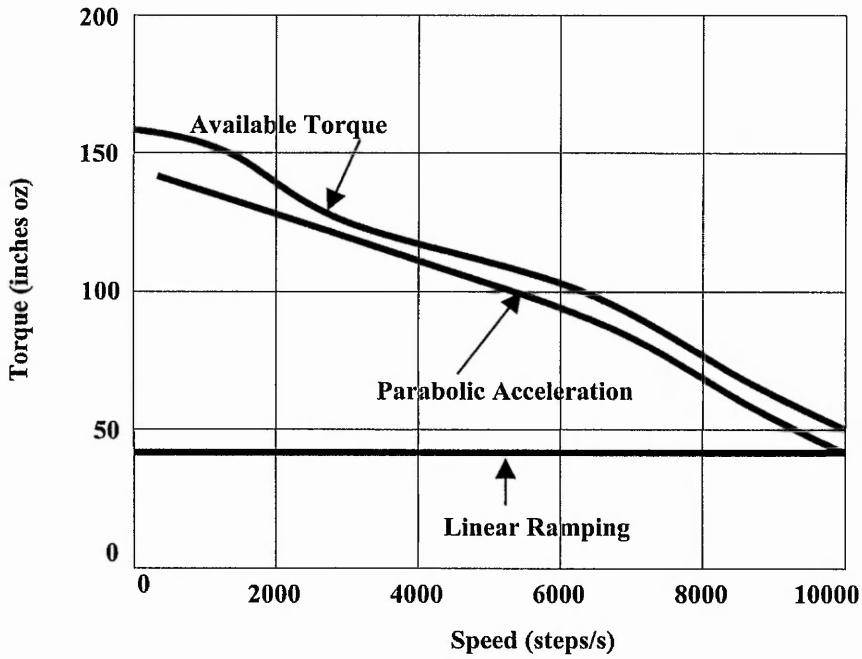


Figure 2-21: Comparison of Available Torque & Torque Utilised (Adapted from [72]) for Linear and Parabolic Acceleration.

Moreover, Dong-Il Kim, *et al.* [73] has shown that machining accuracy is improved with parabolic acceleration in comparison with linear acceleration. This is likely to be because linear acceleration involves a sharp discontinuity in the acceleration (the transition stage between the acceleration phase and the constant phase), which tend to

cause increased vibration and overshoot. Figure 2-21 illustrates the comparison of the torque utilised when using either the linear and parabolic acceleration from Palmin [72].

Palmin [72] highlighted three main benefits derived from parabolic acceleration:

- Servo performance with stepper motor – Controller systems, which integrates the powerful parabolic ramping tool, can bridge the gap between servo and stepper motors. This will reduce the development cost, as stepper motors are much cheaper than servomotors.
- Size and Weight Reduction – Maximising torque utilisation permits the application of smaller stepper motors.
- Complete Stepper Motor Utilisation – Using parabolic acceleration increases the practical speed limit for the stepper motor by using high torque available at low speed to shorten acceleration time.

2.5 System Architecture

The motion controller board has to read the geometric data and speed, and from this information it generates a command signal (pulse), which is fed to the motor driver circuit and eventually causes a motor movement of one step. This has to be done in real-time. Distributed microprocessor architectures and embedded systems are common solutions for the needs for high performance real-time control.

In the early motion control system architectures, a single microprocessor was used for the pulse generation process. With the decreasing price of microprocessors, it is common to allocate a single processor to the pulse generation task for each axis. This distributed multi-processor system can be classified into two categories based on the amount of coupling among the processors. They are:

- Tightly-Coupled
- Loosely-Coupled

Master/slave is one solution in trying to provide motion control. A master/slave relationship is seen in the distributed multi-processor environment. The master processor is responsible of controlling the communication with the slave processors (which are allocated for each of the other axes) and to distribute the pulse generation task. In the tightly-coupled system, there is no communication between the slave processors. On the contrary, the loosely-coupled system allows an independent bus connection between the slave processors.

Sherkat, *et al.* [74] have designed a loosely-coupled system architecture. Sherkat and Thomas [75] showed that the gain in independence from the master-processor is neutralised by an increase in inter-axis communication. This communication overhead is mainly due to the required periodic synchronisation process between the different axes. The synchronisation signals are used to compensate for varying execution times between the processors on the different axes. The layout of this architecture is illustrated in Figure 2-22.

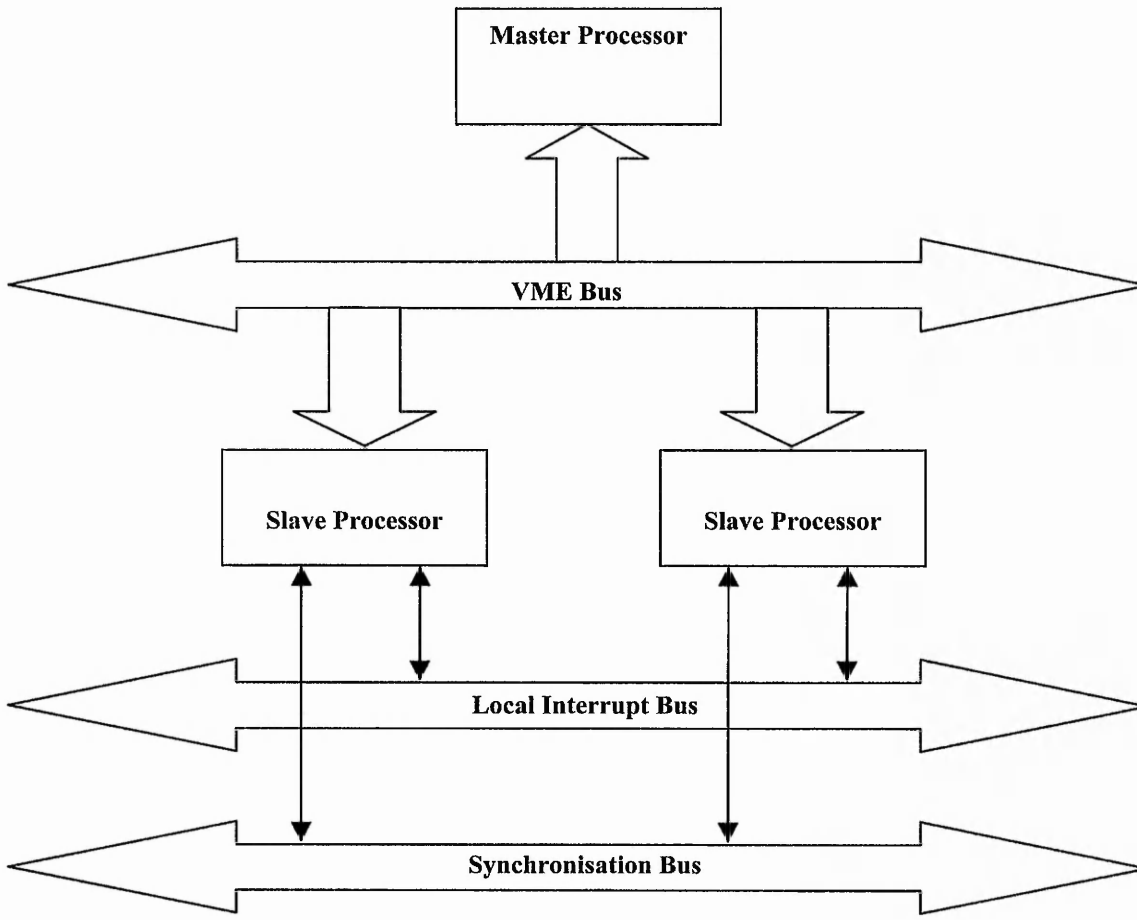


Figure 2-22: Loosely-Coupled Motion Control System Architecture.

Advances in DSP design, i.e. low-level parallel architecture (i.e. Harvard) allows performance that could previously only be gained by a multi-processor system to be gained on a single cost-effective DSP without the communication overhead [5][76][77].

2.6 Review of A Commercial Motion Control System

A typical commercial interpolation technique has been investigated in this research (Axiomatic Technology Ltd) [13]. The path is split into a number of short straight line segments and for each such segment a predefined block of pulses is sent out in the master axis. The master axis is the axis closest in direction to the desired path. The size of the blocks can vary and in practice, each block size is about 25 pulses or more. Figure 2-23 illustrates a simple example taken from Figure 1-9, which has X-axis as the master axis and a block size of 2 pulses for clarity. In this example, the block size is kept constant.

On the other hand, when each block is sent, a number of pulses are sent out in the secondary axis as a block during the same time interval at a constant rate. Usually the number of pulses is smaller, so they are sent at a lower rate, but it can be the same. The number of pulses on the secondary axis must be chosen so that the direction followed is as required. This is illustrated in Figure 2-23 for a simple example (from Figure 1-9). There are three pulse generation intervals in this example (from point A to B, from

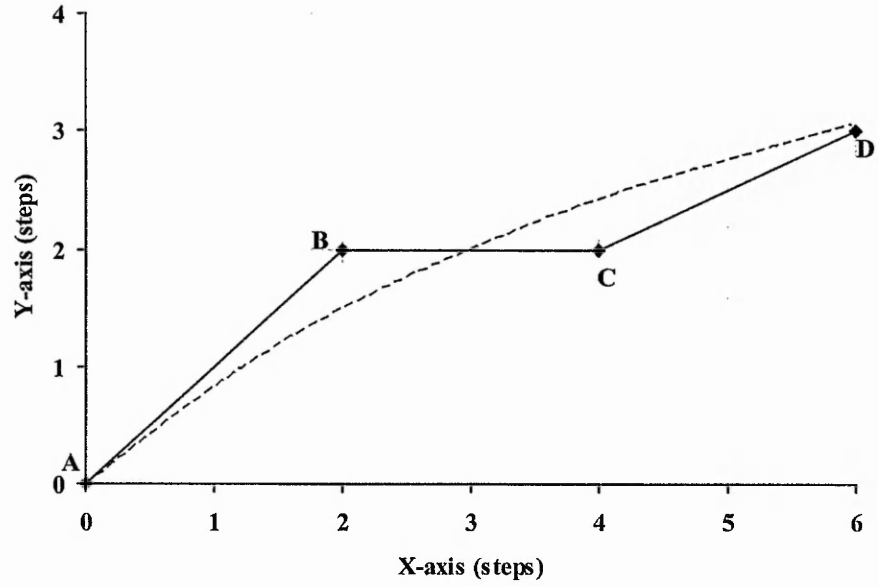


Figure 2-23: Example of Commercial Circular-Arc Interpolation (Full Line) for a Small Segment of an Arc (Dashed Line) (as shown in Figure 1-9) (a Block Size of 2 is used for Clarity).

point B to C and from point C to D). During each of the intervals, 2 pulses in the master axis (X-axis) are sent out. On the other hand, in the secondary axis (Y-axis), the number of pulses are 2, 0 and 1 respectively so the number is the same as for X from A to B and then smaller for B to C and C to D. The maximum number of pulses in a block depends on the curvature of the desired path at that particular segment (high curvature requires a shorter block size to prevent positional errors).

When motion at constant speed is required for the curve in Figure 2-23, the time interval for each block is the same (Figure 2-24 and Figure 2-25) and the 2 axes are synchronised because they use the same block timing. When acceleration is required, ideally the rate at which pulses are sent would be gradually increased. However, in practice, this has not been done due to the lack of computing power. Instead, one block of pulses in the master axis is sent at a constant rate. Then another block is sent at a higher rate, and so on (as discussed in Section 2.4.1). When these data for constant speed are analysed using a speed-time diagram, Figure 2-24, the primary axis can be seen to have constant speed, but for the secondary axis the speed is highest during the first time interval (2 motor steps movement) and drops to zero in the second interval (no motor movement). A movement of one motor step during the third time interval results in a speed of half the speed for the first time interval because the time interval is the same.

During acceleration, the time interval for any subsequent block is shorter than the time interval for the previous block (Figure 2-26 and Figure 2-27). Again the two axes are synchronised by using the same block timing intervals for both. From Figure 2-26, the X-axis speed increases gradually as expected while the Y-axis speed drops in the second block before increasing suddenly at the third block of pulses.

A large number of pulses per block will result in a finer achievable angle between end points during one time interval. For instance, a block size of 2 pulses makes it possible to resolve only to an angle of 26.57° ($\tan^{-1}$ 0.5). If the block size is 10, it would be 5.71° ($\tan^{-1}$ 0.1). However, when the number of pulses in a block is increased, the time interval for sending out those pulses will also increase. During motion planning for

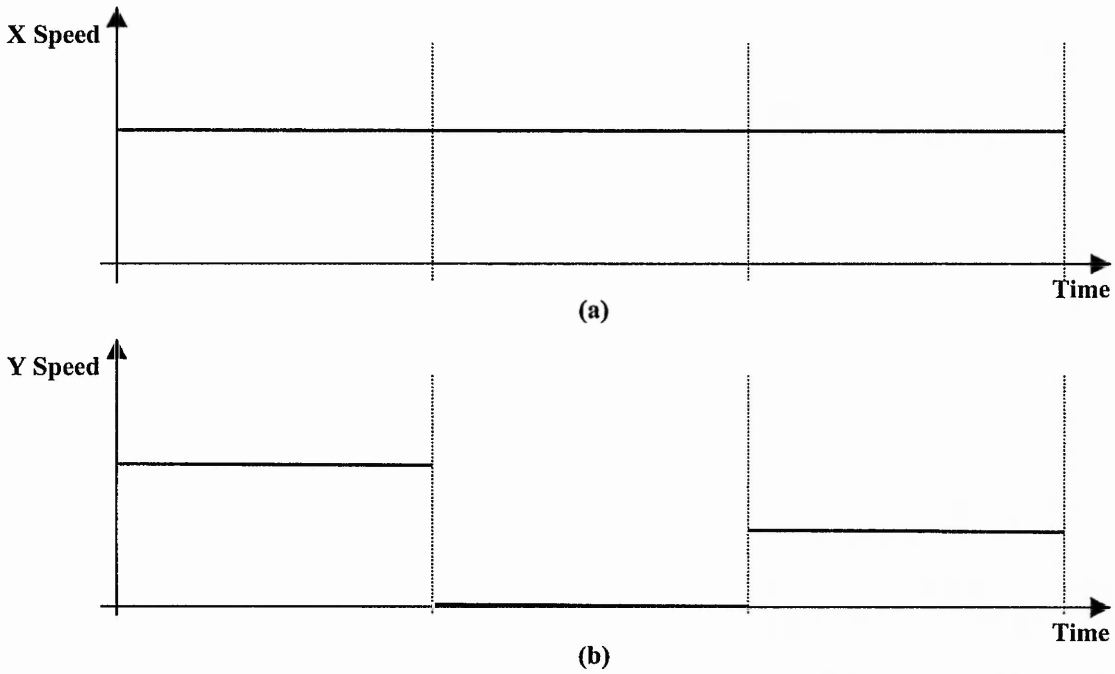


Figure 2-24: Speed-Time Analysis of Constant Speed for the Curve in Figure 2-23: (a) X-axis speed; (b) Y-axis speed.

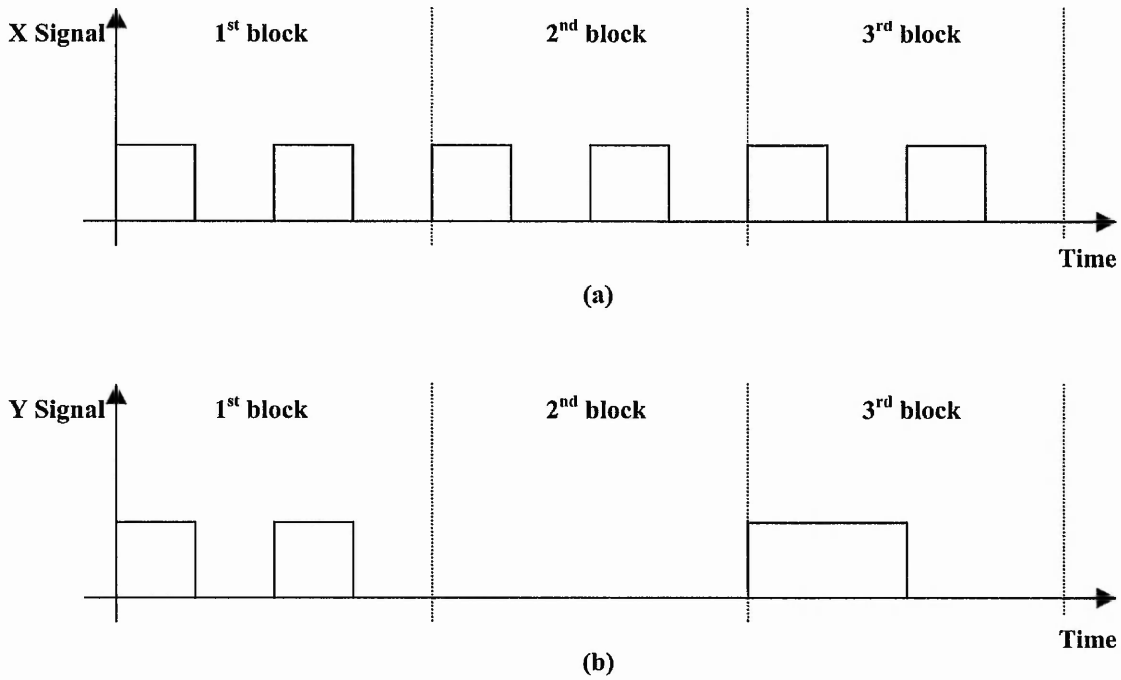


Figure 2-25: Pulses for Motion at Constant Speed for the Curve in Figure 2-23: (a) X-axis Pulses; (b) Y-axis Pulses.

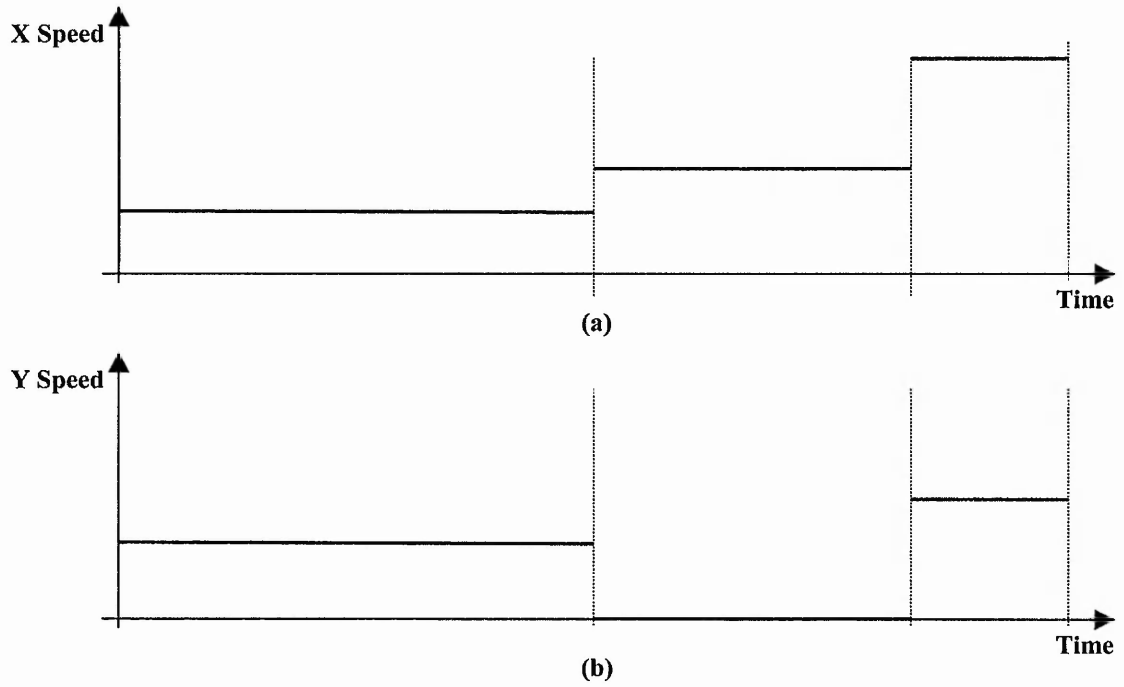


Figure 2-26: Speed-Time Analysis of Accelerating Speed for the Curve in Figure 2-23: (a) X-axis speed; (b) Y-axis speed.

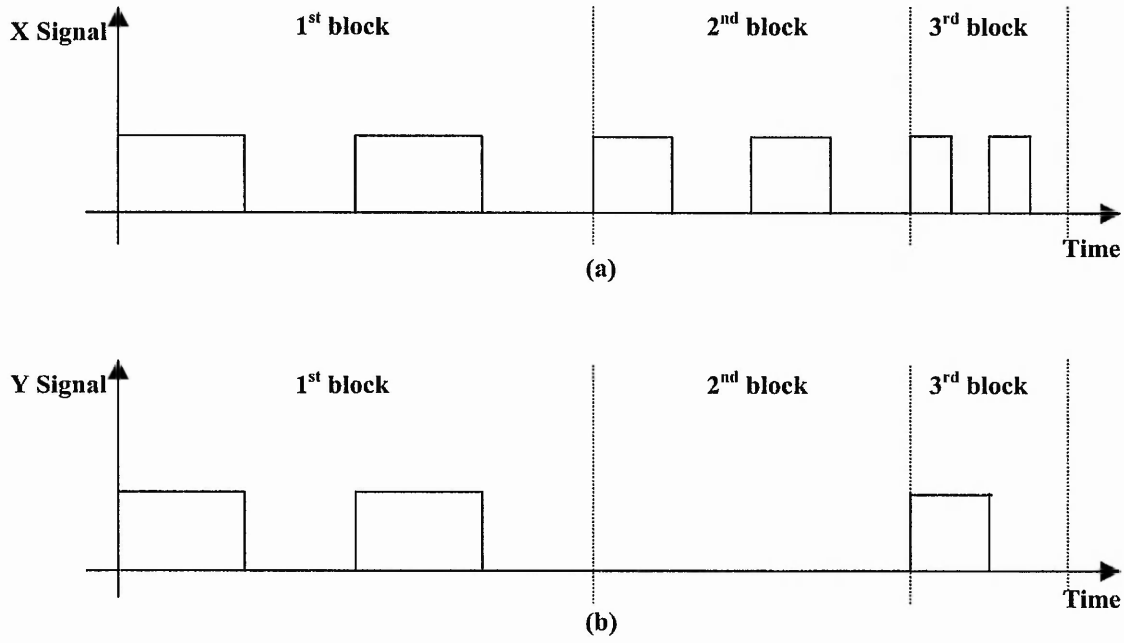


Figure 2-27: Pulses for Motion when Accelerating for the Curve in Figure 2-23: (a) X-axis Pulses; (b) Y-axis Pulses.

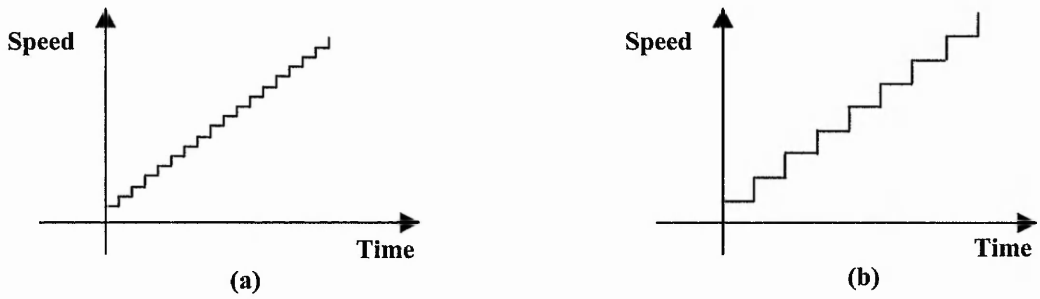


Figure 2-28: Illustration of Motion Planning with Acceleration for Block Size of (a) 2 steps; (b) 4 steps.

acceleration, each block of pulses will be sent out at a constant rate. Therefore, during the acceleration/deceleration phase of the motion planning, the stepper motor will stay at each designated speed for a longer time resulting in a higher jump of speed between the blocks of pulses, as illustrated in Figure 2-28. This will be more likely to cause machine vibrations resulting in unsatisfactory cut quality and position errors.

For circular arcs, many of the stepper motor motion control systems today compromise between large block size to avoid large angles and position errors (see Section 1.2) and smaller block size to avoid vibrations. Figure 2-29 illustrates how a straight line is approximated by short straight line segments. The X-axis speed is constant as can be seen in Figure 2-30(a) but not the Y-axis speed, Figure 2-30(b). A similar example for the circular arc is illustrated in Figure 2-31 and Figure 2-32. This time the speed changes in steps for both axes. An alternative, which has been investigated by the Author of this report, is to incorporate the motion planning for acceleration in the interpolation process by generating a separate timing for every single pulse. In other words, greater control over the speed profile is possible. Sometimes loss of synchronisation can occur, which will affect the remaining part of the path. For a commercial system, it is essential to set acceleration and planned speed values so that loss of synchronisation does not occur.

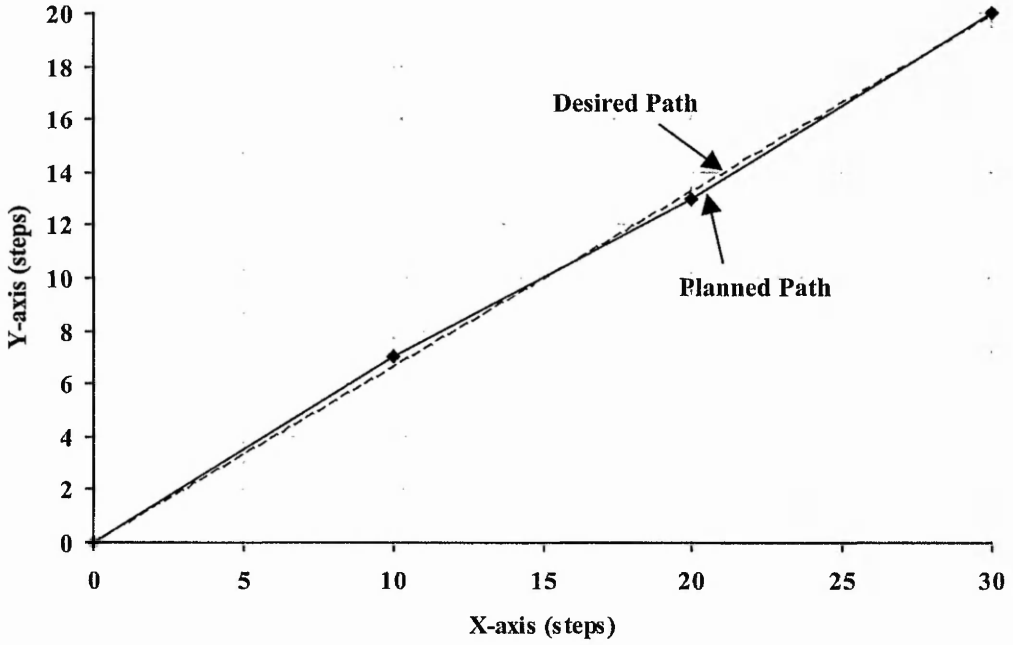


Figure 2-29: Linear Interpolation for Straight Line (0,0) to (30,20) via Points (10,7) and (20,13) (also shown in Figure 1-11).

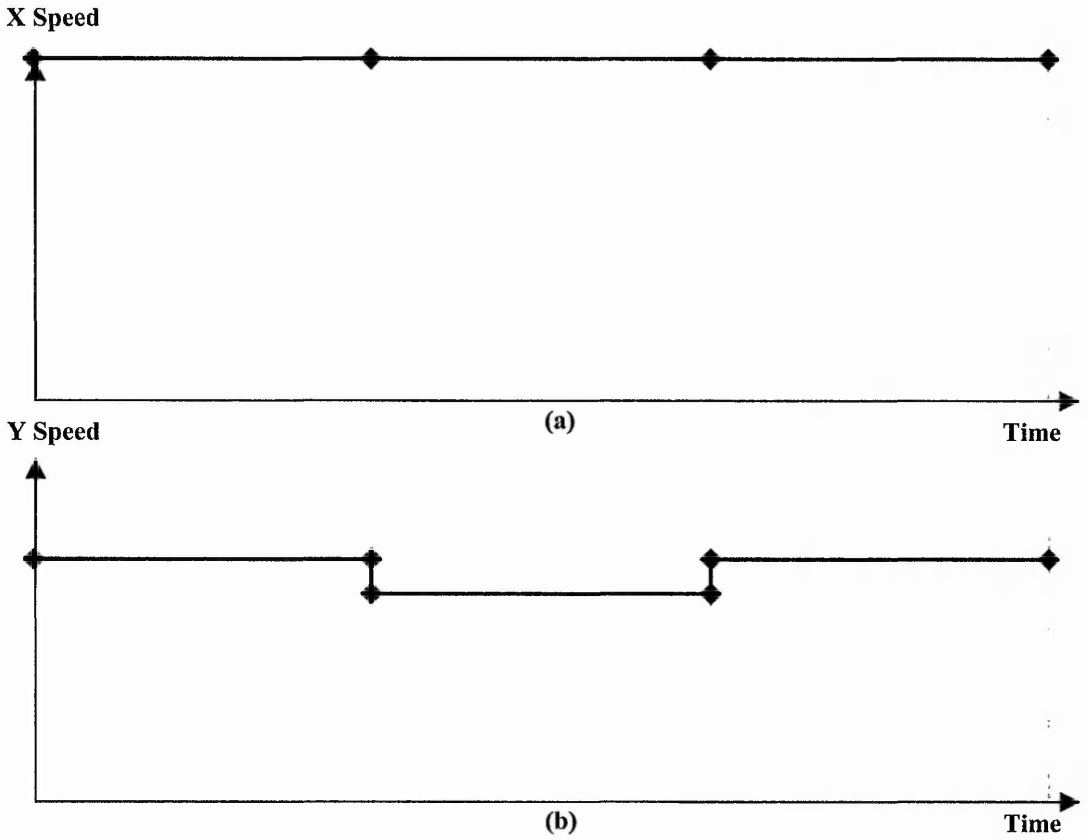


Figure 2-30: Speeds for Straight Line in Figure 2-29: (a) X-axis; (b) Y-axis.

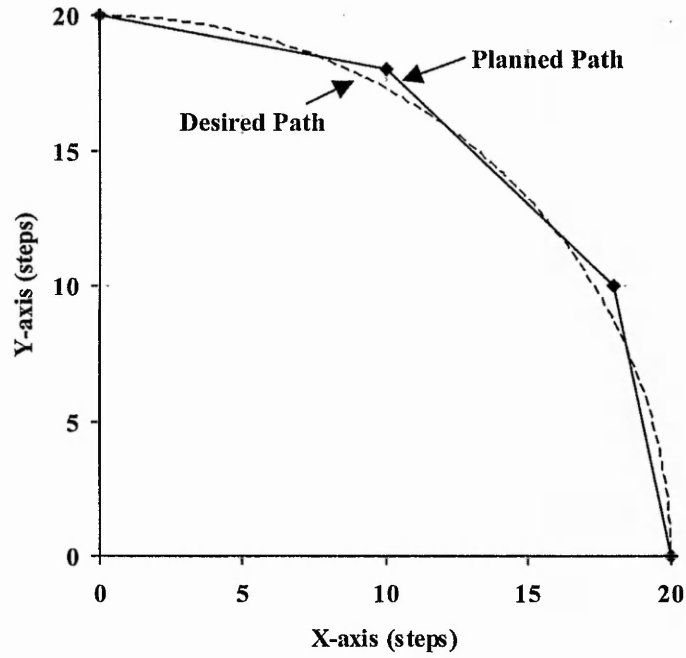


Figure 2-31: Circular Interpolation for Anticlockwise Circular Arc from (0,20) to (20,0) with Centre (0,0) via Points (10,18) and (18,10) (also shown in Figure 1-12).

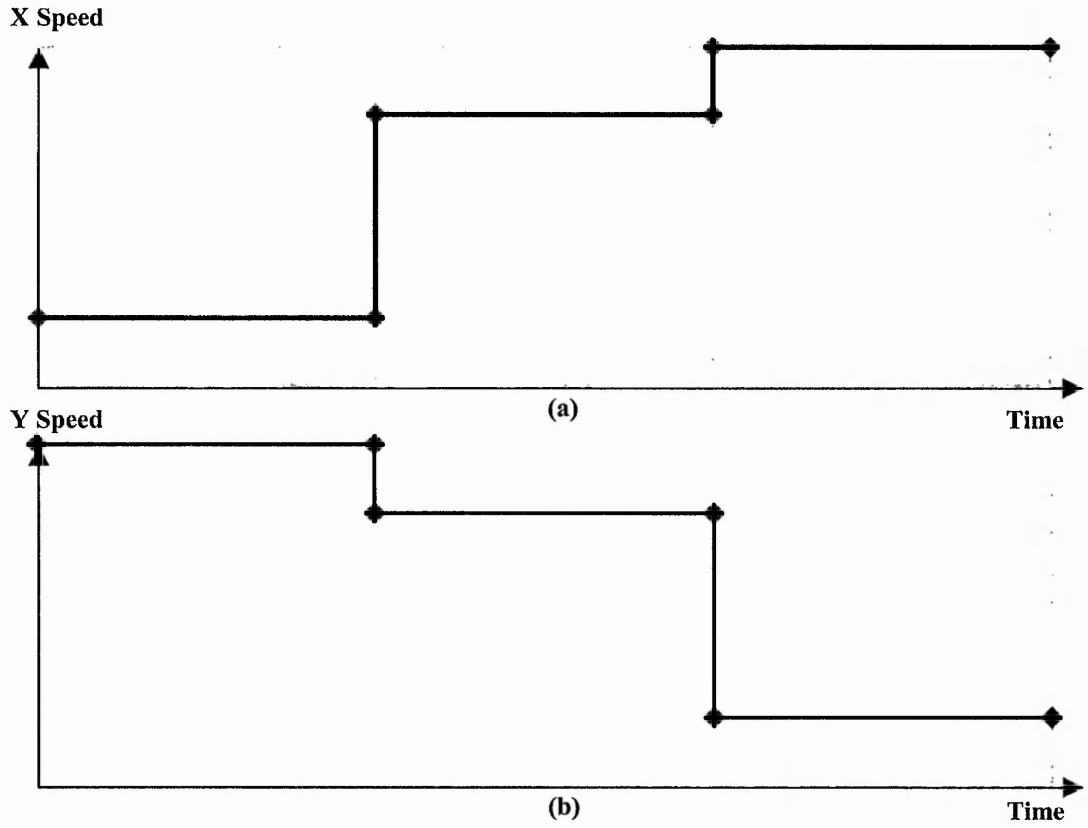


Figure 2-32: Speeds for Circular Arc in Figure 2-31: (a) X-axis; (b) Y-axis.

2.7 Summary

The literature survey conducted and presented in this chapter demonstrates the wide range of different approaches when designing a CNC machining system. Each of the approaches has their advantages and disadvantages. The choice of approach depends on the requirements of the machining process.

The different types of CNC systems were reviewed highlighting the possible motors for the machining. The chosen system to be investigated in the research described in this report is a stepper motor driven continuous path motion control system, which has the benefits of simplicity in construction, lower cost and direct digital control. The major components of a typical CNC machining system, namely the CAD, machine control unit and machine tools, have also been presented. Each of these components contributes to the end result of the machined path. This thesis concentrates only on the machine control unit, which is the core of the machining system. The main tasks of a machine control unit include interpolation of the required path while implementing acceleration and deceleration.

Through investigations, a large number of interpolation algorithms have been found. Some of the most commonly used algorithms have been presented and described in detail in this chapter. Only the circular DDA algorithm maintains constant resultant speed along a circular path. However, this advantage is neutralised by the deviation from the original path. The Direct Search interpolation algorithm is more appropriate when used in an open-loop stepper motor driven machinery because deviation from the required path cannot be compensated.

Two common acceleration algorithms are the linear and parabolic acceleration. Linear acceleration is a better choice when simplicity is the machining criterion. On the other hand, when performance is of major importance, parabolic acceleration is more appropriate because it is able to utilise more of the available torque. A typical commercial CNC motion control system has been investigated. The pulse generation process is analysed in detail. The results show that problems arise in certain aspects of the process and need to be solved.

3 Timing-Based Interpolation for Stepper Motors

Chapter 2 described the properties of various components involved in the continuous path motion generation process in respect of smoothness of motor motion. In-depth investigations into these components reveal several shortcomings, particularly in the interpolation and acceleration algorithms used. In Section 2.6, the problems with existing stepper motor driven multi-axis motion control system are discussed. These problems manifest themselves as unsatisfactory path following.

In order to overcome or at least minimise the problems with stepper motor driven motion control system, new interpolation algorithms have been developed. Many of the interpolation algorithms that are available in the industry today can be thought of as position-based interpolation algorithms. The coordinates of the end points of the vector are first determined. For circular arc interpolation, additional information is needed (e.g. circle centre). From this information, new interpolated points are calculated by breaking the vector to a number of short line segments, as illustrated in Figure 2-29 and Figure 2-31. Then each line segment is interpolated by generating pulses, at a different constant rate on each axis. This line is the same direction as earlier examples (Figure 2-8, Figure 2-10, Figure 2-11 and Figure 2-12) but the line has double the length to give more idea of what happens with a longer line.

Another very different approach to interpolation has been proposed by the Author, timing-based interpolation. The idea is partly based on the Axiomatic approach but has been extended to whole vectors and extended to curves as well as straight lines. The idea is to derive an equation for the timing of every command pulse on an individual axis according to the path geometry. Ultimate control is gained of every individual pulse. For lines, the pulses are each needed to be sent at a different constant rate. When interpolating arcs, the rate of the pulses can be increased or decreased gradually on an individual axis. This reduces the chances of vibrations caused by sudden changes in pulse rates from the interpolation algorithm. The feedrate information is also used when generating the timing so that the resultant path will be machined at the required speed.

The motion on different axes is synchronised by calculating the time when each step should be achieved.

Examples of paths generated when interpolating using the new algorithms are described in Section 3.4. A more detailed evaluation of the new algorithms will be discussed in Chapter 6.

The algorithms developed require heavy floating-point calculations. Therefore, these algorithms are only possible with the current advances in microprocessor technology. The core processor used to run the new algorithms is a Digital Signal Processor (DSP), which is capable of intensive floating-point operations.

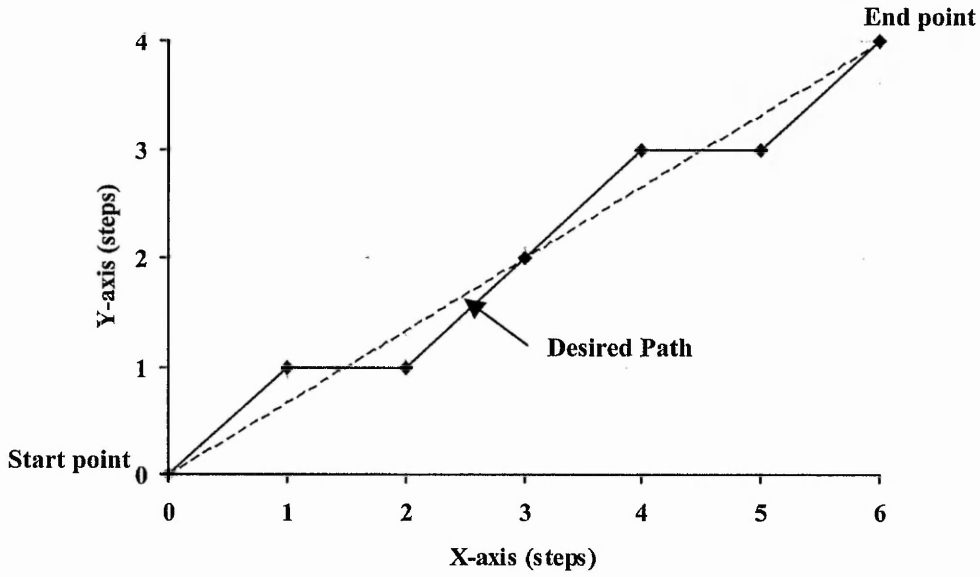


Figure 3-1: Position-Based Direct-Search Interpolation for Line from (0,0) to (6,4).

3.1 Position-Based Interpolation versus Timing-Based Interpolation

Most of the interpolation algorithms that are available in industry today are position-based interpolation algorithms. The coordinates of the end points are first determined, which are obtained from the part program. From this information, new interpolated points are calculated. The example in Figure 3-1 illustrates how position-based interpolation is generated for a straight line from (0,0) to (6,4) using Direct-Search interpolation algorithm.

With position-based interpolation, one interpolated point (shown as a black circle in Figure 3-1) is generated at each fixed time sample. The next interpolated point is usually taken as the point closest to the required path. By doing this, the generated path can follow the required one closely but the speed on the different axes will vary as shown in Figure 3-3. In this example, the speed varies in the middle part of the line. The X-axis speed stays constant while the Y-axis speed jumps to a higher value in the middle of the path before decreasing to the initial speed. For a longer line, the speed will continually jump up and then down again. This variation of speed may cause vibrations.

The generated command pulses are shown in Figure 3-2 while the speeds are illustrated in Figure 3-3. The speed at each pulse is calculated using the time interval after the previous pulse, as explained in Section 6.1.2. Therefore, there will not be any speed data for the first command pulse.

On the other hand, the timing-based interpolation algorithms are able to reduce or sometimes eliminate the sudden changes of speed. For the line from Figure 3-1, the pulses are distributed evenly in the time domain, as shown in Figure 3-4. The speeds on both axes are now constant throughout the interpolation of the line, Figure 3-5.

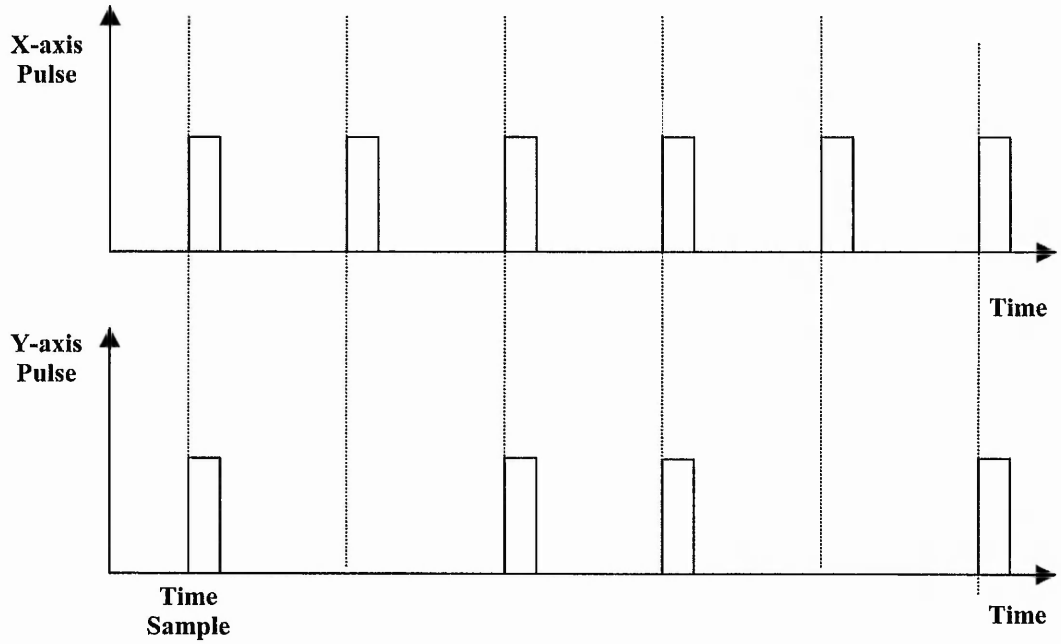


Figure 3-2: Command Pulses for Position-Based Interpolation for the Line in Figure 3-1.

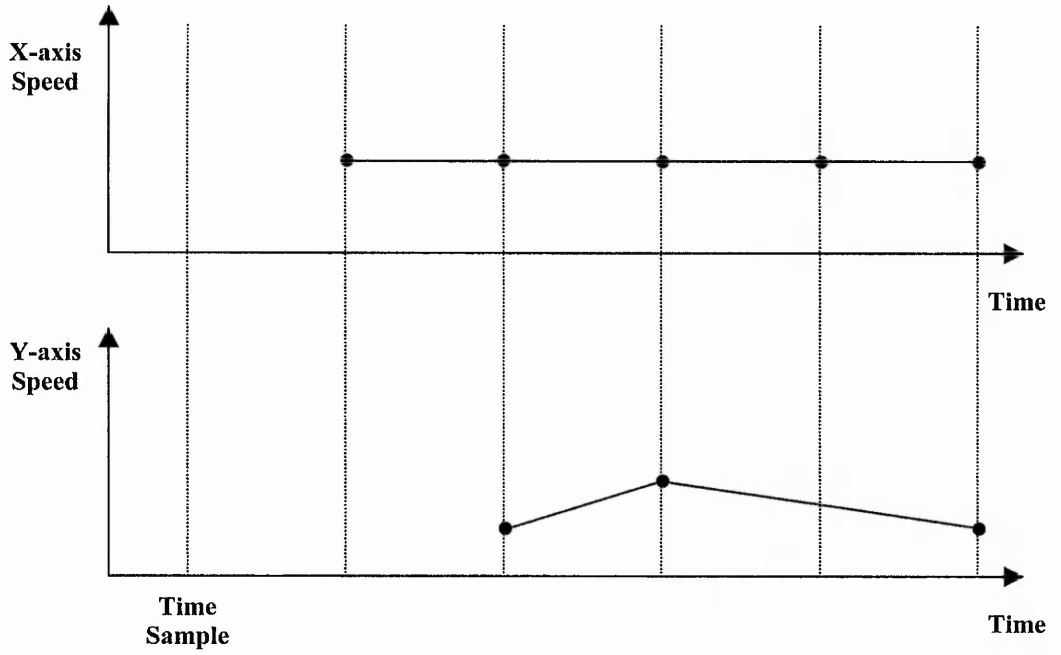


Figure 3-3: Illustration of Axis Speed for Position-Based Interpolation for the Line in Figure 3-1.

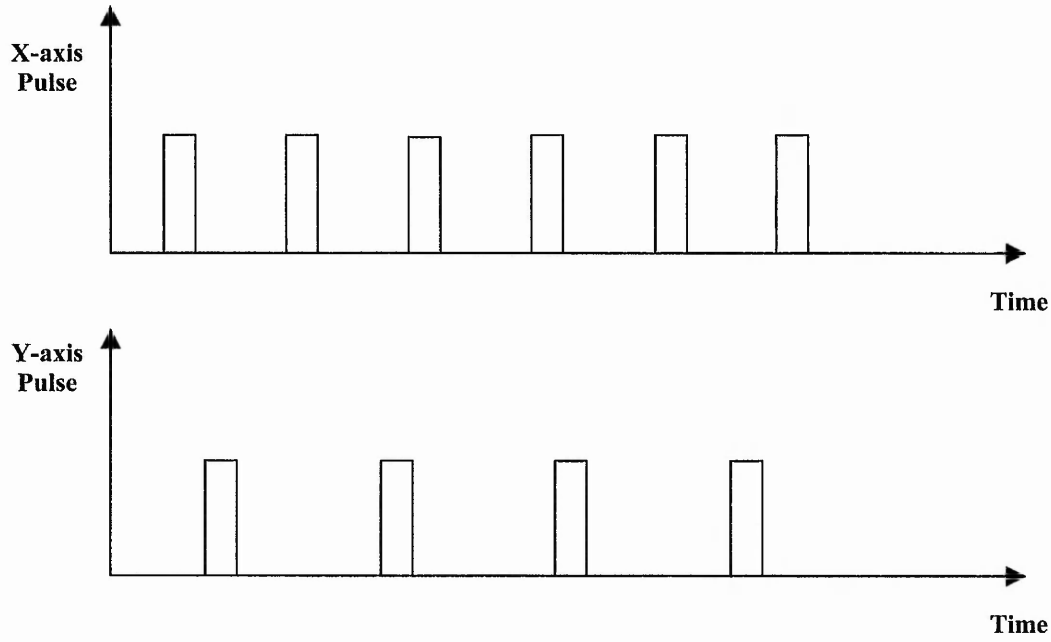


Figure 3-4: Command Pulses for Timing-Based Interpolation for the Line in Figure 3-1.

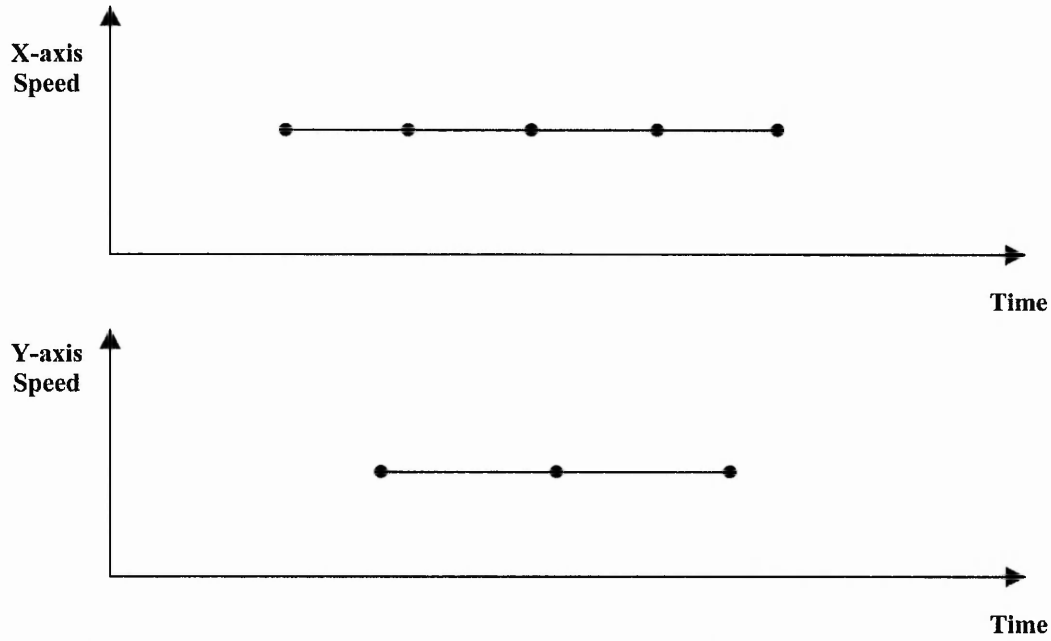


Figure 3-5: Illustration of Axis Speed for Timing-Based Interpolation for the Line in Figure 3-1.

For curves of second order or higher, the changes of speed are necessary but they are done gradually, without abrupt changes.

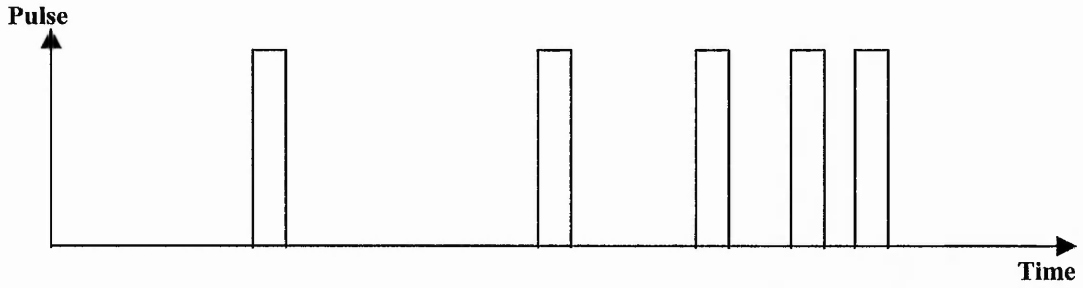


Figure 3-6: Example of Timings for Non-Linear Interpolation.

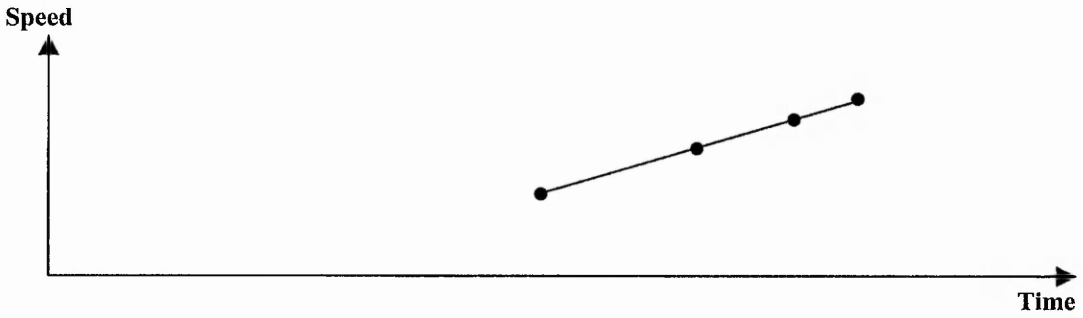


Figure 3-7: Illustration of Speed vs Time for the Timings in Figure 3-6.

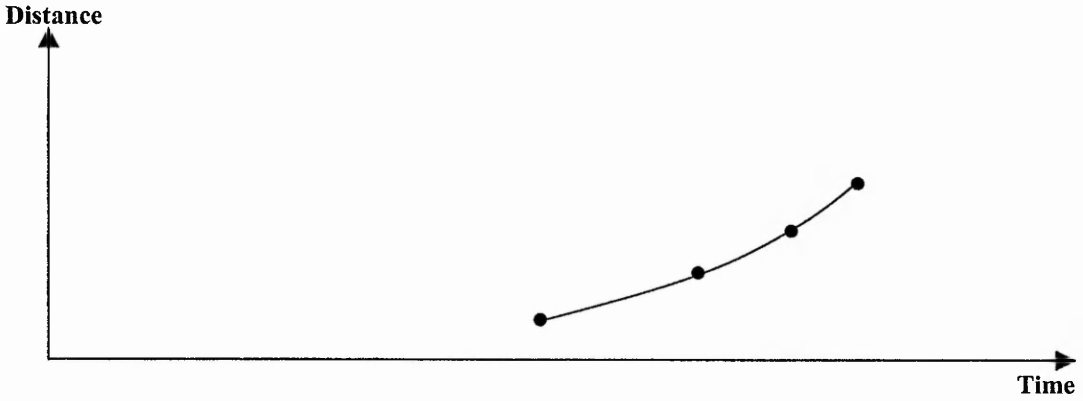


Figure 3-8: Illustration of Distance Travelled in Individual Axis for the example in Figure 3-6.

Figure 3-6 above illustrates an example of timings generated for an individual axis when following a curve. The speed of the pulses is increasing gradually, as shown in Figure 3-7. Therefore, there will not be any abrupt changes of speed, which could cause vibrations. The resultant distance travelled (single axis) for this example is illustrated in Figure 3-8.

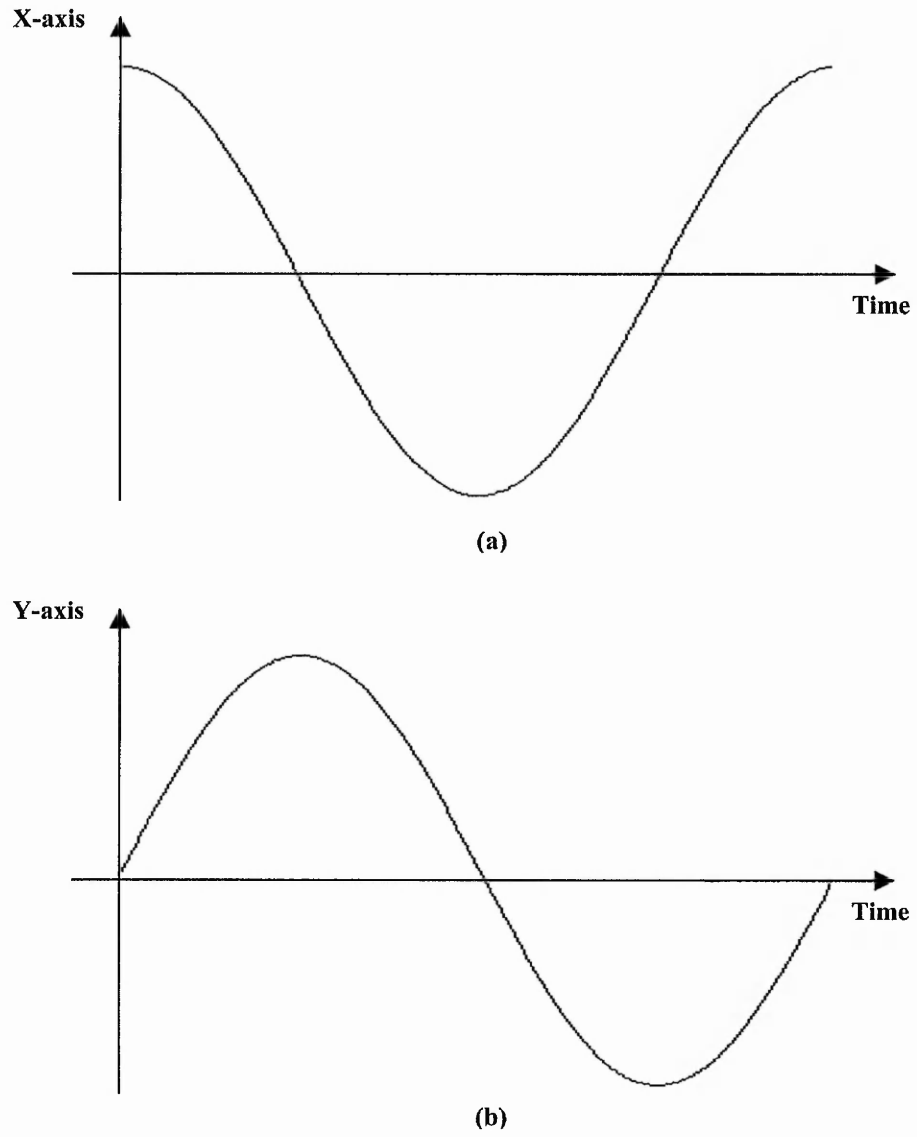


Figure 3-9: Illustrations of Required Axis Motion for Circle Interpolation: (a) X-axis Position; (b) Y-axis Position.

In the case of a circular arc, the speed plots will need to resemble the shape of sine or cosine waves, so they will also have gradual changes of speed. For constant motion for a circle, the x and y-axis position variations in time are illustrated in Figure 3-9. Figure 3-10 shows the required axis speed to generate this circle.

The research described in this report is based on the generation of interpolation using individual pulse timings (timing-based interpolation). The avoidance of vibrations is

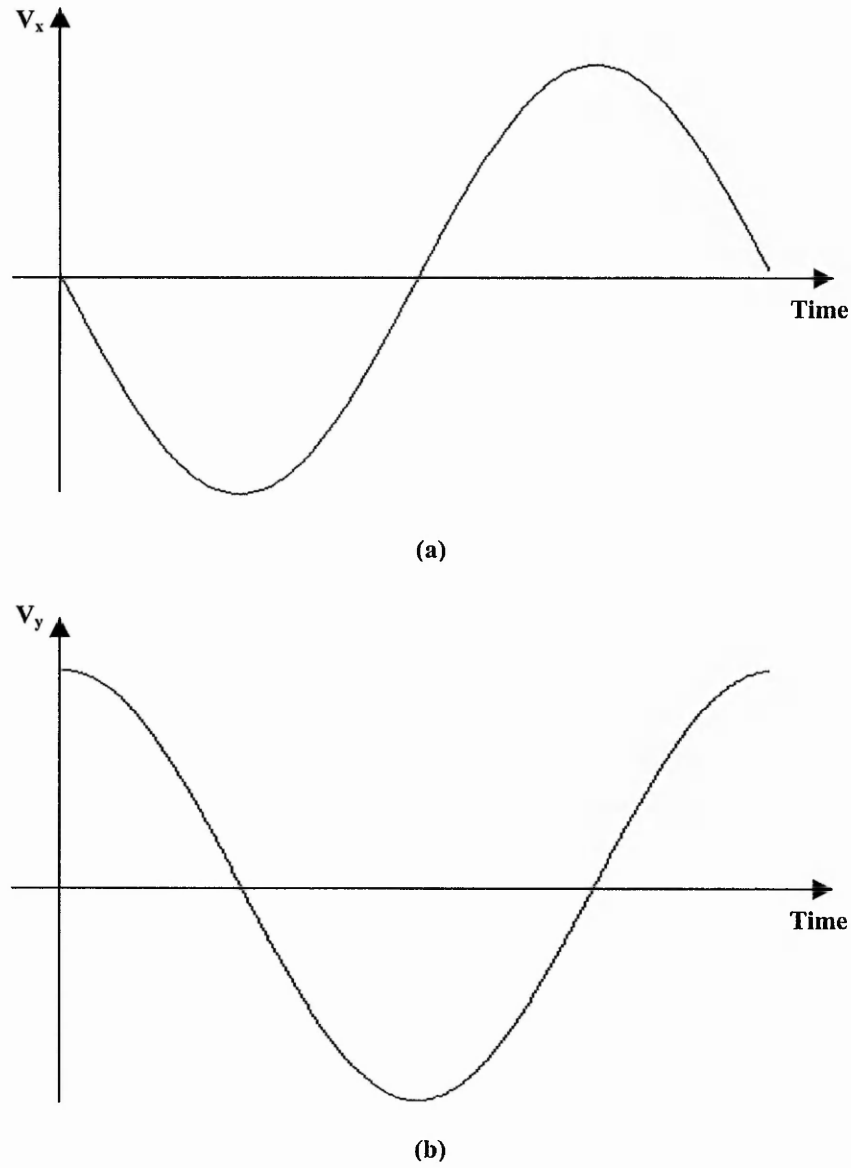


Figure 3-10: Illustration of Speed for Circle Interpolation shown in Figure 3-9: (a) X-axis Speed; (b) Y-axis Speed.

very important when using the stepper motor driven open loop control system, because there is no feedback to correct any errors made.

Axiomatic Technology Ltd use a master/slave position-based technique for their motion control system. Each block of pulses is sent at a fixed rate in the master axis. During the same time interval, a block of pulses is sent in the slave axis. The number of pulses in the slave axis block is calculated to achieve the required direction. It can be considered that the interpolation within each block is timing-based, although actual individual

timing calculations are not involved but a pulse rate is calculated for the whole block. A pulse generator is used to generate the pulses at the calculated rate. The speed is constant within a block but there can still be a sudden change between blocks, especially for curves, as illustrated earlier in Figure 2-31 and Figure 2-32. The speed changes between blocks are still high, so are likely to cause vibrations. Even for straight lines, the blocks are needed so that the acceleration algorithm can change the rate to increase the speed.

The following sections explain how the new timing-based interpolation is achieved. Section 3.2 describes the new timing-based linear interpolation and Section 3.3 discussed the new circular arc interpolation.

3.2 The New Timing-Based Linear Interpolation

Interpolation within the motion controller is responsible for the generation of a stream of command pulses for the machine tool. Instead of using conventional position interpolation, a new timing interpolation algorithm is used to generate the timing for every individual command pulse. A method is needed to synchronise the different axes. In existing algorithms, the generation of pulses is synchronised at predetermined positions. This requires points to be used where both x and y are exactly a whole number of steps. Very often such points do not lie on the line or arc, as can be seen in Figure 3-1.

Both the linear and circular arc interpolation algorithms described in this chapter make use of the parametric representation of the line or arc. The parameter chosen for the new linear interpolation is distance travelled along the path because distance and time are closely related. When the speed is constant then they are proportional, with distance equal to speed multiplied by time. For constant speed interpolation, time could equally well be used. However distance has been chosen, because it will enable the development of the acceleration algorithms in Chapter 4. Therefore, path distance is used to synchronise the motion of the different axes throughout the machining process.

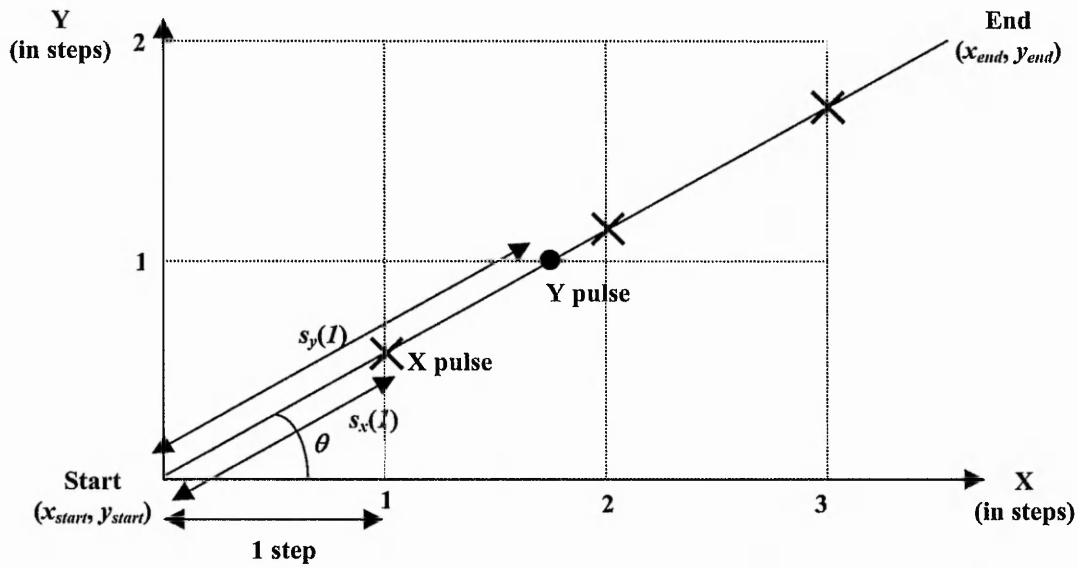


Figure 3-11: Straight Line Representation for a Line in the 1st Quadrant in the case starting from (0,0).

3.2.1 Algorithm for First Quadrant

The algorithm will be described initially for the first quadrant with speed assumed to be constant. Then the extension to other quadrants will be explained. The straight line in Figure 3-11 can be expressed in a parametric form as follows:

$$x(s) = s\cos\theta$$
$$y(s) = s\sin\theta \tag{3-1}$$

where

s = distance travelled from the starting point
θ = (constant) angle between the straight line and X-axis
$x(s)$ = distance travelled in x direction = $|x\text{-}x_{start}|$
$y(s)$ = distance travelled in y direction = $|y\text{-}y_{start}|$
x_{start} = X-axis position for starting point
y_{start} = Y-axis position for starting point
x_{end} = X-axis position for destination point
y_{end} = Y-axis position for destination point

Equation (3-1) shows how the distance in the X or Y-axis can be related to the linear distance followed thus far. The inputs for linear interpolation are the starting and destination positions. From these data, the angle, θ, is derived as in (3-2).

$$\theta = \tan^{-1}\left(\frac{|y_{end} - y_{start}|}{|x_{end} - x_{start}|}\right) \tag{3-2}$$

For straight lines, the angle, θ, in Figure 3-11 is constant throughout the interpolation. Therefore, cos θ and sin θ are also constant. On the other hand, s varies throughout the interpolation.

The distance travelled along the desired line can be expressed in terms of the distance travelled in the X or Y-axis, by rearranging (3-1), as shown in the following equation:

$$s_x = \frac{|x - x_{start}|}{\cos\theta}$$
$$s_y = \frac{|y - y_{start}|}{\sin\theta} \tag{3-3}$$

The distance travelled along the required line for every individual pulse can be determined by putting the x or y value in equation (3-3). Every command pulse corresponds to the movement of one stepper motor step. In other words, one command pulse causes movement equal to the step size of the stepper motor. Therefore, the relation between the X-axis position after n_x steps and the stepper motor step size, D, is shown in (3-4) and similarly for the Y-axis.

$$|x - x_{start}| = n_x D$$
$$|y - y_{start}| = n_y D \tag{3-4}$$

for $n_x = 1,2,3,\ldots$, destination X step & $n_y = 1,2,3,\ldots$, destination Y step

To determine the distance travelled in the X-axis after the n-th pulse, the n value is substituted into equation (3-4).

Substituting (3-4) into (3-3) yields:

$$s_x(n_x) = \frac{n_x D}{\cos\theta}$$
$$s_y(n_y) = \frac{n_y D}{\sin\theta} \tag{3-5}$$

With the feedrate information, V, assumed to be constant, this distance travelled along the desired path can be related to the time elapsed since the start of interpolation, t, as in equation (3-6).

$$s(t) = Vt \tag{3-6}$$

Substituting equations (3-6) into (3-5), the relation between the X and Y-axis position and the elapsed time becomes clearer.

$$Vt_x = \frac{n_x D}{\cos\theta}$$
$$Vt_y = \frac{n_y D}{\sin\theta} \tag{3-7}$$

Rearranging (3-7),

$$t_x(n_x) = \frac{n_x D}{V\cos\theta}$$
$$t_y(n_y) = \frac{n_y D}{V\sin\theta} \tag{3-8}$$

Both equations in (3-8) can be used to find the exact timing for each pulse. For instance, the timing for the 8th pulse in x to be generated is calculated by replacing the n_x in the first equation of (3-8) with the value 8.

As discussed earlier, sin θ and cos θ are constant. Therefore, part of (3-8) can be calculated once at the beginning of the interpolation process, as in (3-9), and storing the result in memory.

$$A_x = \frac{D}{V\cos\theta}$$
$$A_y = \frac{D}{V\sin\theta} \tag{3-9}$$

At all other iterations of the interpolation steps, only one addition operation is required, as in (3-10).

$$t_x(n_x) = t_x(n_x - 1) + A_x$$
$$t_y(n_y) = t_y(n_y - 1) + A_y \tag{3-10}$$

with $t_x(0) = 0$ and $t_y(0) = 0$.

3.2.2 Algorithm for All Quadrants

Figure 3-11 illustrates a quadrant 1 linear interpolation. However, the equations presented in this section apply to any quadrants of interpolation. The angle, θ, calculated in equation (3-2) for different quadrants are illustrated in Figure 3-12. Therefore, the interpolation for quadrants 2, 3 and 4 can be treated in a similar way as the interpolation in the first quadrant. To determine which quadrant the required interpolation belongs to, the signs of $x_{end} - x_{start}$ and $y_{end} - y_{start}$ can be used. To allow for different quadrants the direction in x is set according to the sign of $x_{end} - x_{start}$. Similarly the direction in y depends on the sign of $y_{end} - y_{start}$. This is summarised in Table 3-1.

Table 3-1: Table of Line Directions.

X-axis		Y-axis	
x_{end} - x_{start}	**x direction**	y_{end} - y_{start}	**y direction**
+ve	+1	+ve	+1
-ve	-1	-ve	-1

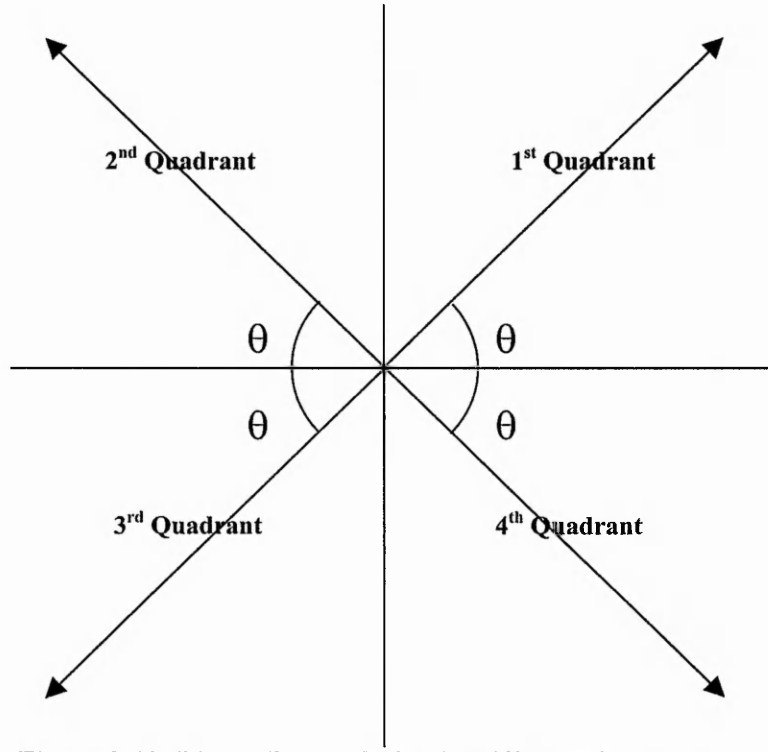


Figure 3-12: Linear Interpolation in Different Quadrants.

3.3 The New Timing-Based Circular Interpolation

The curve that is most commonly used for interpolation is the circular interpolation. As in the case for linear interpolation, a parameter is used to synchronise the different axes, in contrast to conventional synchronisation at predetermined positions. For linear interpolation, the parameter used to define the line is distance travelled. The advantage is because of the close relationship between the distance travelled and the elapsed time. This parameter can also be applied easily to circular arc interpolation. Hence, the distance has also been used as the parameter for the new circular arc interpolation.

3.3.1 Algorithm for First Quadrant Anti-Clockwise

Interpolation of an anti-clockwise first quadrant circular arc is explained in detail throughout this section. The equations for circular interpolation in other quadrants or in other directions are summarised in Table 3-2 in Section 3.3.2. Again the speed is assumed to be constant.

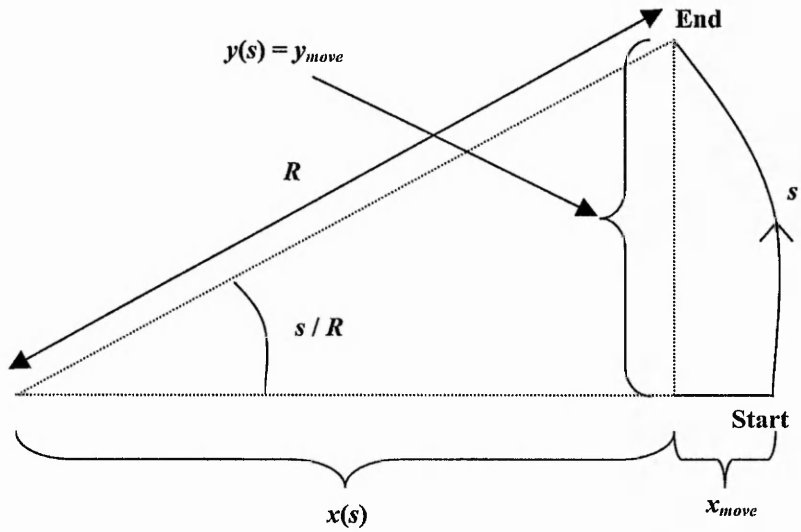


Figure 3-13: Circular Arc Representation for Anti-Clockwise Arc with Radius R in 1st Quadrant.

For a circular arc of length, s, and radius, R, the angle turned through is s/R radians, as shown in Figure 3-13. A 90° anti-clockwise circular arc starting on the positive X-axis in a two-dimensional space can be expressed as follows:

$$x(s) = R\cos\frac{s}{R}$$
$$y(s) = R\sin\frac{s}{R} \tag{3-11}$$

where

R = radius of circle

s = distance along the arc from the X-axis base

$x(s)$= position in x with respect to the circle centre = $x - x_{centre}$

$y(s)$= position in y with respect to the circle centre = $y - y_{centre}$

It should be noted that $x(s)$ is not the distance travelled on the X-axis but it is the X-coordinate position (relative to the centre of the circle) after a distance s along the arc has been travelled. On the other, $y(s)$ coincides with the distance travelled on the Y-axis for this case, because $y_{start} = 0$. The distance travelled on the X-axis in this case can be expressed as follows:

$$x_{move} = x(s) - x_{start} = x(s) - R$$
$$y_{move} = y(s) \tag{3-12}$$

The distance travelled along the desired circular arc can be expressed in terms of the position along the X or Y-axis by rearranging (3-11).

$$s_x = R\cos^{-1}\left(\frac{x - x_{centre}}{R}\right)$$
$$s_y = R\sin^{-1}\left(\frac{y - y_{centre}}{R}\right) \tag{3-13}$$

From equation (3-12), the relationship between the X and Y-axis position and the stepper motor step size is as follows:

$$\begin{aligned} x - x_{centre} &= x_{move} + R \\ &= (n_x(-D)) + R \\ y - y_{centre} &= n_y D \end{aligned} \tag{3-14}$$

for n_x = 1,2,3,..., destination X step and n_y = 1,2,3,..., destination Y step where D = motor step size

Substituting x and y from (3-14) into (3-13) yields

$$\begin{aligned} s_x(n_x) &= R\cos^{-1}\left(\frac{-n_x D + R}{R}\right) \\ &= R\cos^{-1}\left(\frac{-n_x D}{R} + 1\right) \\ s_y(n_y) &= R\sin^{-1}\left(\frac{n_y D}{R}\right) \end{aligned} \tag{3-15}$$

With the feedrate information, V (assumed constant), the path distance travelled can be related to the time elapsed since the start of interpolation, t, as in (3-16).

$$s(t) = Vt \tag{3-16}$$

Substituting s in (3-16) into equation (3-15) yields

$$\begin{aligned} Vt_x &= R\cos^{-1}\left(\frac{-n_x D}{R} + 1\right) \\ Vt_y &= R\sin^{-1}\left(\frac{n_y D}{R}\right) \end{aligned} \tag{3-17}$$

Rearranging equation (3-17), the time elapsed, t, can be expressed in terms of the X or Y- axis position:

$$\begin{aligned} t_x(n_x) &= \frac{R}{V}\cos^{-1}\left(\frac{-n_x D}{R} + 1\right) \\ t_y(n_y) &= \frac{R}{V}\sin^{-1}\left(\frac{n_y D}{R}\right) \end{aligned} \tag{3-18}$$

To determine the appropriate timing for every individual pulse, the command pulse number, denoted by n_x and n_y, are substituted into equation (3-18). For example, to calculate the timing of the 8th pulse on each of the two axes, $t_x(8)$ and $t_y(8)$, equation (3-18) is used with n_x and n_y each assigned the value 8.

To minimise the processing required for every command pulse, two parts of the division operation in equation (3-18) can be precalculated at the start of a circular interpolation and storing them in memory because they are constant throughout the interpolation.

$$A = \frac{R}{V}$$
$$B = \frac{D}{R} \tag{3-19}$$

The equation above assumes that the feedrate is constant throughout the interpolation process. Equations (3-18) with A and B in (3-19) now become

$$t_x = A\cos^{-1}(-n_x B + 1)$$
$$t_y = A\sin^{-1}(n_y B) \tag{3-20}$$

The new circular-arc interpolation algorithm described earlier in this section is for interpolation of the anticlockwise first quadrant circular arc. Interpolation for other quadrants and directions will be discussed in the next section.

3.3.2 Algorithm for All Quadrants in Both Directions

For the other quadrants and directions, the appropriate equations are summarised in Table 3-2. The circular arc explained thus far starts interpolating from the X-axis base line. It is also possible for the interpolation to start at a particular angle from the X-axis base line, α_{start}. A circular arc of radius, R, centre, (x_{centre}, y_{centre}) and start angle, α_{start} (angle from the X-axis base line), in the first quadrant is expressed in a parametric form (with the plus sign is for anticlockwise and the minus sign for clockwise) as:

$$x(s) = x_{centre} + R\cos\left(\alpha_{start} \pm \frac{s}{R}\right)$$
$$y(s) = y_{centre} + R\sin\left(\alpha_{start} \pm \frac{s}{R}\right) \quad (3\text{-}21)$$

Table 3-2: Equations for New Circular Arc Interpolation.

Direction	Quadrant	X-Axis		Y-Axis	
		Pulse Timings	Direction	Pulse Timings	Direction
Clockwise	1st	$t = A\sin^{-1}(n_x B)$	+1	$t = A\cos^{-1}(-n_y B + 1)$	-1
	2nd	$t = A\cos^{-1}(-n_x B + 1)$	+1	$t = A\sin^{-1}(n_y B)$	+1
	3rd	$t = A\sin^{-1}(n_x B)$	-1	$t = A\cos^{-1}(-n_y B + 1)$	+1
	4th	$t = A\cos^{-1}(-n_x B + 1)$	-1	$t = A\sin^{-1}(n_y B)$	-1
Anti-Clockwise	1st	$t = A\cos^{-1}(-n_x B + 1)$	-1	$t = A\sin^{-1}(n_y B)$	+1
	2nd	$t = A\sin^{-1}(n_x B)$	-1	$t = A\cos^{-1}(-n_y B + 1)$	-1
	3rd	$t = A\cos^{-1}(-n_x B + 1)$	+1	$t = A\sin^{-1}(n_y B)$	-1
	4th	$t = A\sin^{-1}(n_x B)$	+1	$t = A\cos^{-1}(-n_y B + 1)$	+1

3.4 Simulation Examples

As described in earlier chapters, one generated command pulse will result in the linear motion of one stepper motor step on that particular axis. For motion planning, it can sometimes be useful to imagine that the motor moves instantaneously when a command pulse is received, resulting in a movement of one motor step. However, this is not physically possible. Various more realistic simulation techniques have been used to simulate the possible motion. These simulation techniques will be discussed in more detail in Chapter 5. The simulation methods used for simulation of position are:

- Zero Order (Instantaneous case);
- Varying Rate First Order;
- Constant Rate First Order; and
- Second Order simulation.

For the Zero Order simulation, the stepper motor is assumed to move to the desired position instantaneously when a command pulse is received.

There are two different First Order simulation methods used. The Varying Rate First Order simulation assumes that the motor has always completed the movement for the pulse exactly when the next pulse is sent. The Constant Rate First Order simulation assumes not only that the motor has completed one motor step movement before the next command pulse is received but also that the motion varies linearly with time at a fixed constant rate until that step is completed (and then waits for the next pulse). Therefore the time for the movement (response time) is always the same.

The Second Order simulation is based on a mass-spring model of the stepper motor system where damping factor and the natural frequency of the system are used. All these simulation methods will be discussed in detail in Chapter 5 but they are used here to give an idea of the results of the interpolation algorithms.

In addition to the position simulation described above, a speed simulation method has also been used to simulate the speed variations throughout the interpolation process.

Examples of the speed simulation have been illustrated previously in Figure 3-3, Figure 3-5 and Figure 3-7. As explained earlier, Section 3.1, the speed data is calculated by assuming that the distance travelled for one motor step takes place during the time interval between the pulses. This means that there will not be any speed data for the first command pulse. The speed is therefore calculated in effect by assuming that the motor moves as in the Varying Rate First Order simulation. In all the simulation results, the number of steps is used instead of the actual distance for clarity. The motor step size used in a typical system is 0.01 mm.

3.4.1 Simulation of Linear Interpolation

Figure 3-14 shows the path generated by the new linear interpolation algorithm using the Zero Order simulation method for a line from (0,0) to (30,20). The speed used in this linear interpolation example is 500 steps/s (0.3 m/min). On the other hand, Figure 3-15 shows the same interpolation using the Second Order simulation method, which will be discussed in Section 5.2.4. The dashed line is the required path while the generated path is shown in full line. The simulated path can be seen to be able to follow the required path closely.

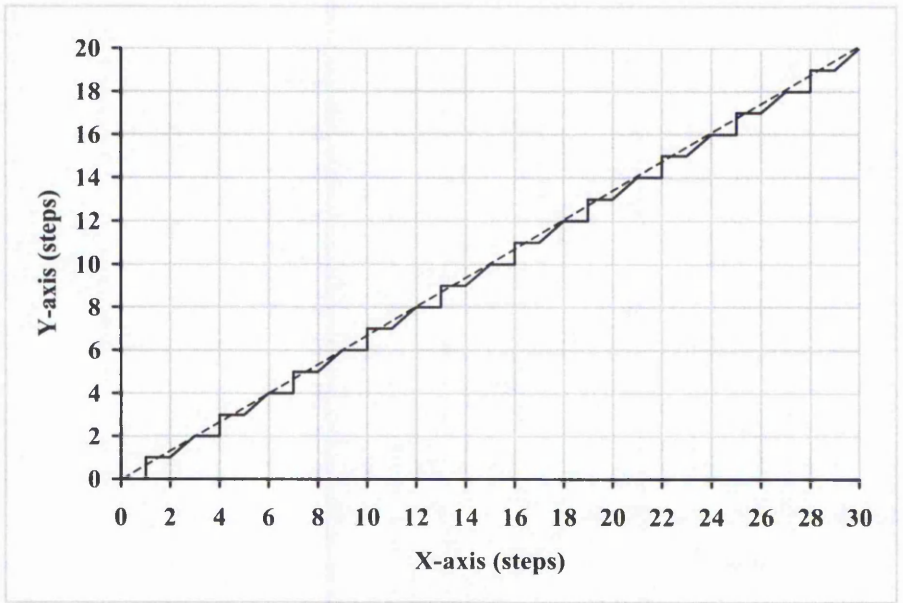


Figure 3-14: Plot of New Linear Interpolation (Zero Order Simulation). Full Line is simulated path. Dashed Line is the path required.

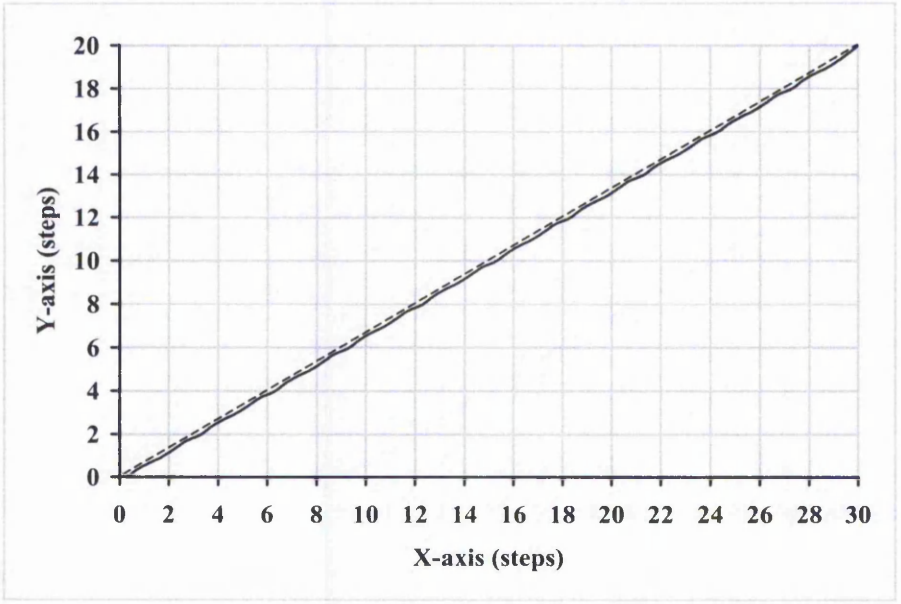


Figure 3-15: Plot of New Linear Interpolation (Second Order Simulation). The damping factor is 0.7, typical values for a stepper motor control system. Full Line is simulated path. Dashed Line is the path required.

The position error for the two simulations is plotted in Figure 3-16 and Figure 3-17. The error is plotted as positive if it is above the required line and negative if otherwise. This gives greater information about the motion. From Figure 3-14 (Zero Order simulation), we can see that in this example the simulated path repeats itself after every three steps in the X-axis. Therefore, position error too follows a similar pattern. After three steps in the X-axis, the position error is zero because the third X steps touches the required line.

Figure 3-17 illustrates the position error when simulated using the Second Order simulation. The position error is smaller than that for Zero Order simulation in Figure 3-16. This is because the dynamics of the stepper motor smooth out the motion, thus bringing the generated path closer to the required one. In practice, the Second Order simulation is likely to be closer to the actual generated path.

The largest position error for the Zero Order simulation is 0.55 of a step while the Second Order simulation produces largest position error of 0.19 of a step. Therefore, the generated path is likely to follow the required line at an accuracy within half the stepper motor step according to these two simulation methods. The average error value

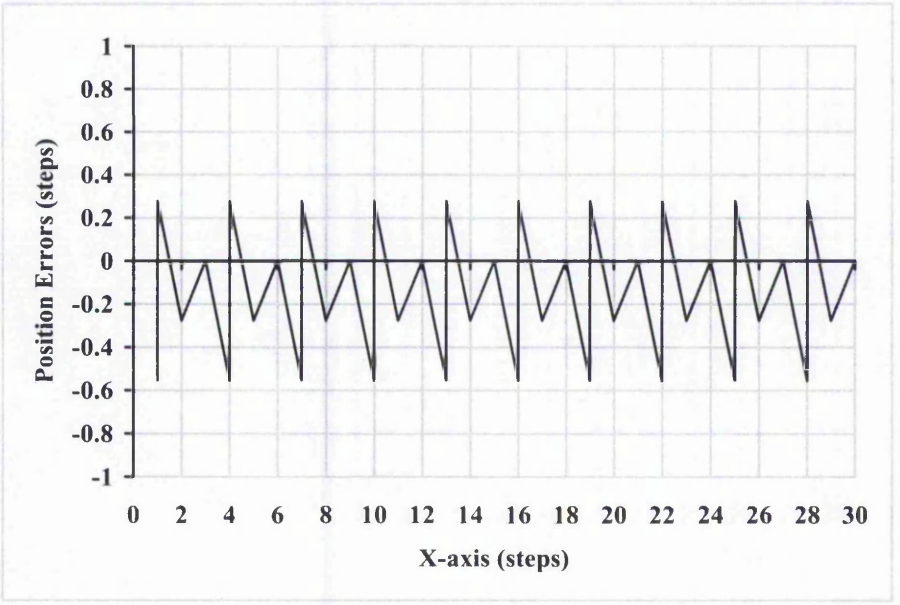


Figure 3-16: Position Error of New Linear Interpolation (Zero Order Simulation) shown in Figure 3-14. The largest position error here is 0.55. The error is positive if it is above the required line and negative below.

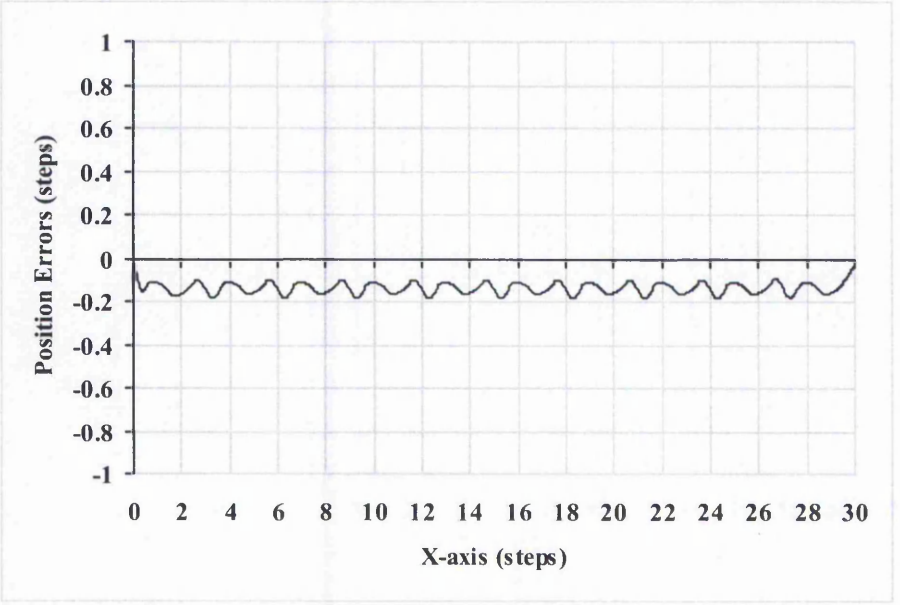


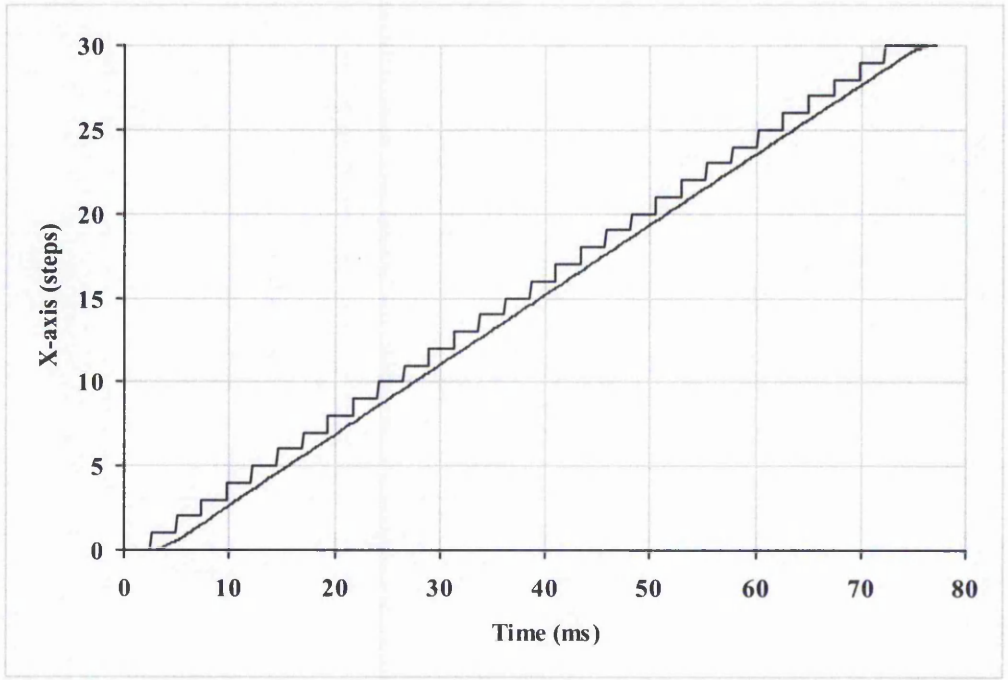
Figure 3-17: Position Error of New Linear Interpolation (Second Order Simulation) shown in Figure 3-15. The largest position error here is 0.19. The error is positive if it is above the required line and negative below.

is –0.14 step. This average error is reduced when employing the Half-Step Technique, which will be discussed in Section 3.5.

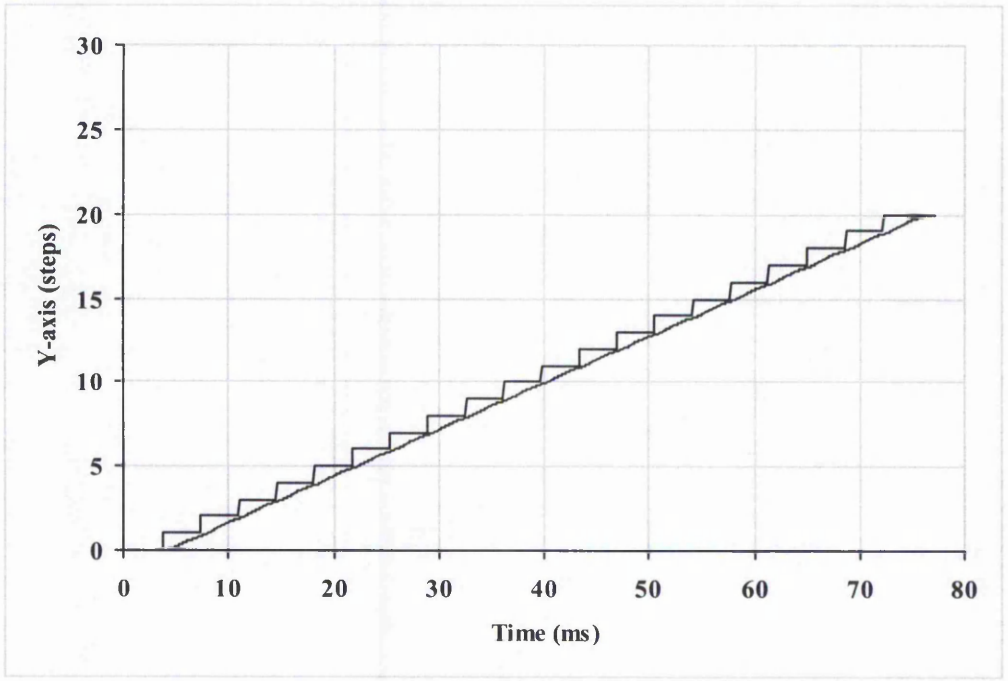
The Zero Order simulation assumes that the motor step happens at the instant of the arrival of the command pulse. In reality, the response of the motor is smoother but it can be oscillatory. The frequency of this oscillation depends on the natural frequency of the stepper motor. This can be seen when one motor step is moved, which will be discussed in more detail in Section 5.1.

The motor will take some time before it settles at the next step position. When more than one step is moved, the next step comes before the motor has settled at the previous step. This, in fact can smooth out the motion with the rotor lagging the stator. This is fine as long as the lagging is not more than two steps away [29]. Otherwise, the motor steps will lose synchronisation, causing missed steps.

Figure 3-18 illustrates the behaviour of the motor in each axis separately in the example from Figure 3-15. The "steps" in the diagrams show the motion in Zero Order case, while the curves show the position from the Second Order simulation. To illustrate the changes in motor position in more detail, the axis position changes in Figure 3-18 have been enlarged for the first 20 ms of the motion and are shown in Figure 3-19. A single command pulse will result in oscillation before it settles at the next motor step. It can be seen from Figure 3-19 that the next command pulse comes and reinforces the motion from the first, resulting in a smoothing effect. It can be noted that the motion in X-axis is smoother than in the Y-axis. This is because of the higher pulse rate in the X-axis, resulting in a smaller time interval between command pulses. However, the lag is greater with the higher pulse rate.

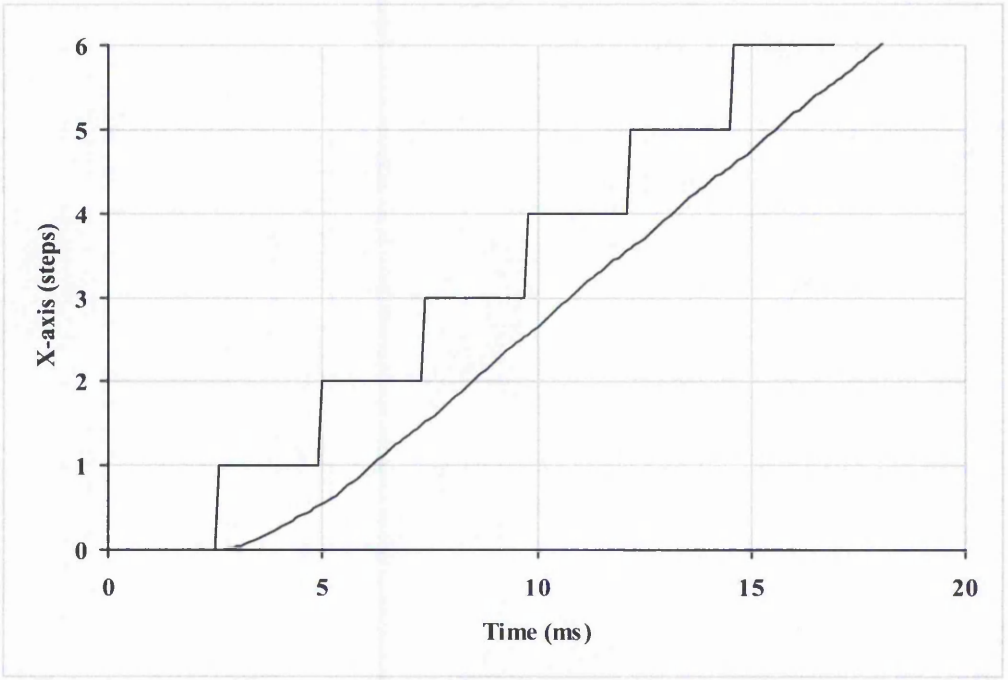


(a)

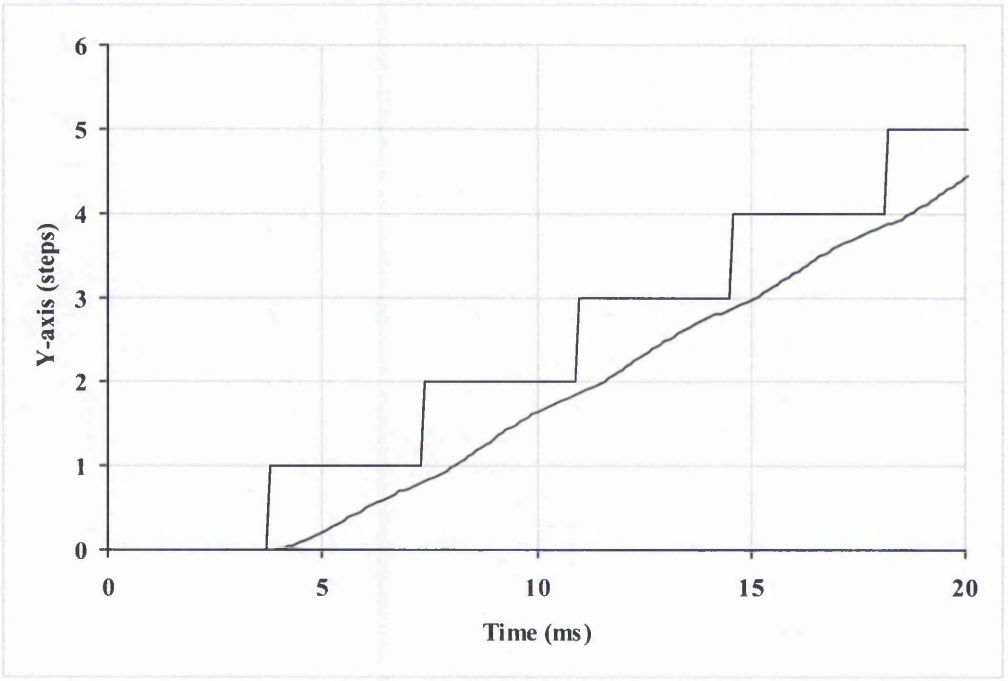


(b)

Figure 3-18: Plot of Motion along the 2 Axes of the New Linear Interpolation (Second Order Simulation) shown in Figure 3-15. The Zero Order simulation is also shown to indicate the times of pulses.



(a)



(b)

Figure 3-19: Detail of Plot of Motion along the 2 Axes of the New Linear Interpolation (Second Order Simulation) from Figure 3-18 for 20 ms.

3.4.2 Simulation of Circular Arc Interpolation

Figure 3-20 and Figure 3-21 show an example of the path generated using the new circular interpolation. The parameters used for a quarter of a circle are as follows:

start	= (20,0)
end	= (0,20)
direction	= anticlockwise
centre	= (0,0)

The speed used is 500 steps/s (0.3 m/min). Figure 3-20 shows the path generated by the new circular interpolation algorithm when using the Zero Order simulation. From this figure, we can tell that the new interpolation still able to follow the required circle fairly closely. In fact, the largest position error in this case is 0.97 of a step.

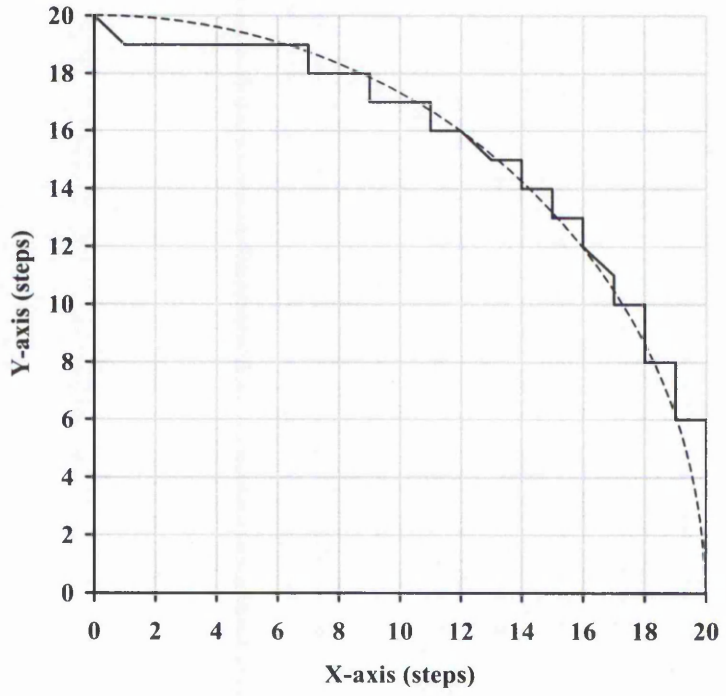


Figure 3-20: Plot of New Circular Arc Interpolation (Zero Order Simulation). Full Line is simulated path. Dashed Line is the required path.

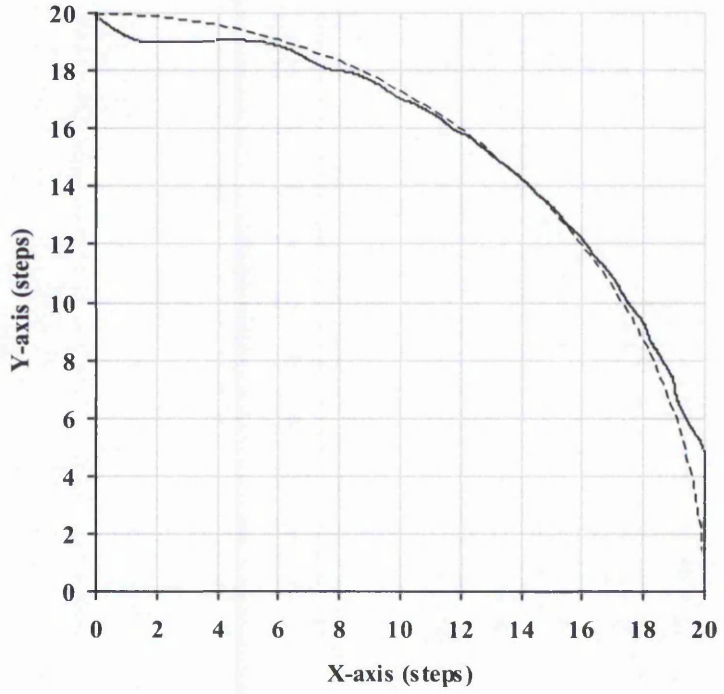


Figure 3-21: Plot of New Circular Arc Interpolation (Second Order Simulation). The damping factor is 0.7, which are typical for a stepper motor control system. Full Line is simulated path. Dashed Line is the required path.

Figure 3-21 shows the path for the interpolated circular arc when using the Second Order simulation. When one step in one axis is followed quickly by one step in the other, it looks like a corner when using the Zero Order simulation. In the actual machining, this is smoothed out by the dynamics of the machine. This can be seen from Figure 3-21, when Second Order simulation takes into consideration the dynamics of the stepper motor.

As explained earlier, the largest position error is within the resolution of the stepper motor. This means that the new circular interpolation can still follow the required circular arc closely. Figure 3-22 shows the position error throughout the whole interpolation process for the Zero Order case. Although the size of the position error is close to one motor step at the beginning and end of the motion, it is below 0.6 of a step in the middle of the motion. The error is plotted as positive if it is outside the circle and negative if inside.

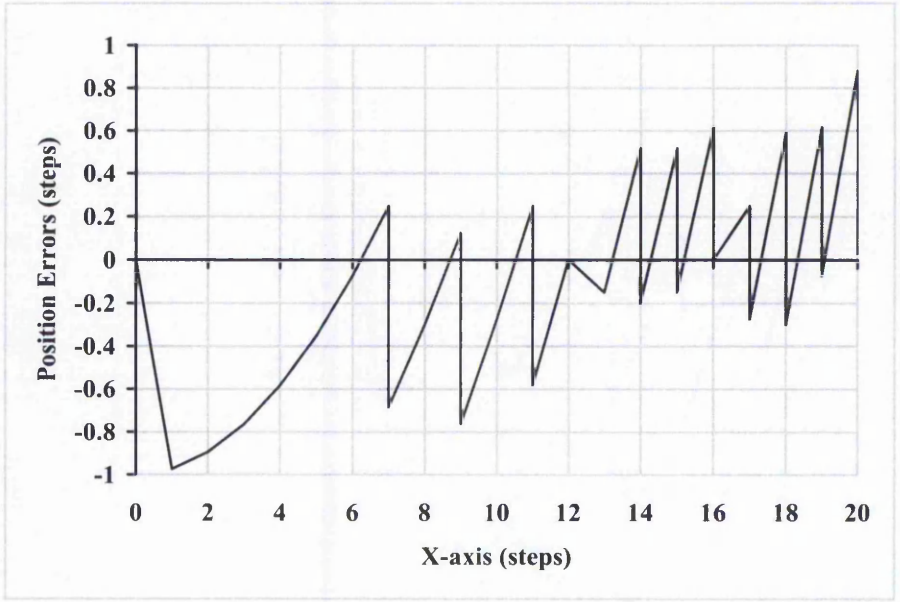


Figure 3-22: Position Error of New Circular Arc Interpolation shown in Figure 3-20 (Zero Order Simulation). The largest position error here is 0.97. The error is positive if it is outside the required circle and negative inside.

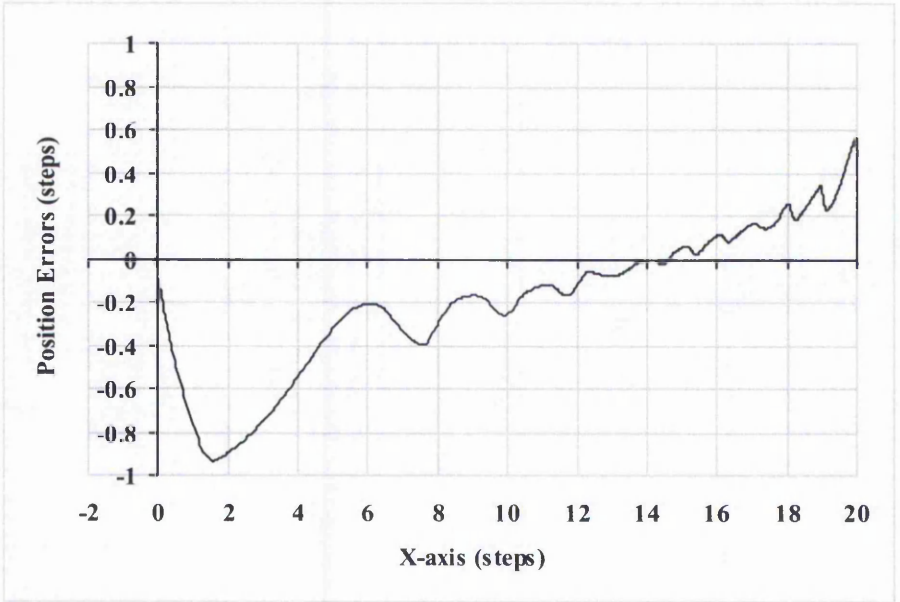
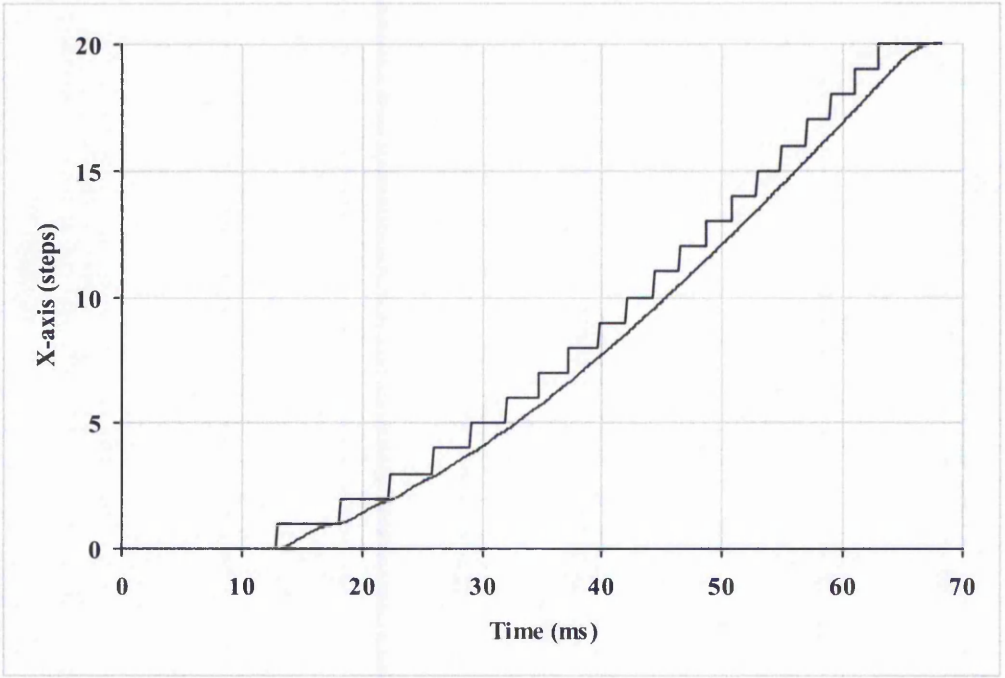
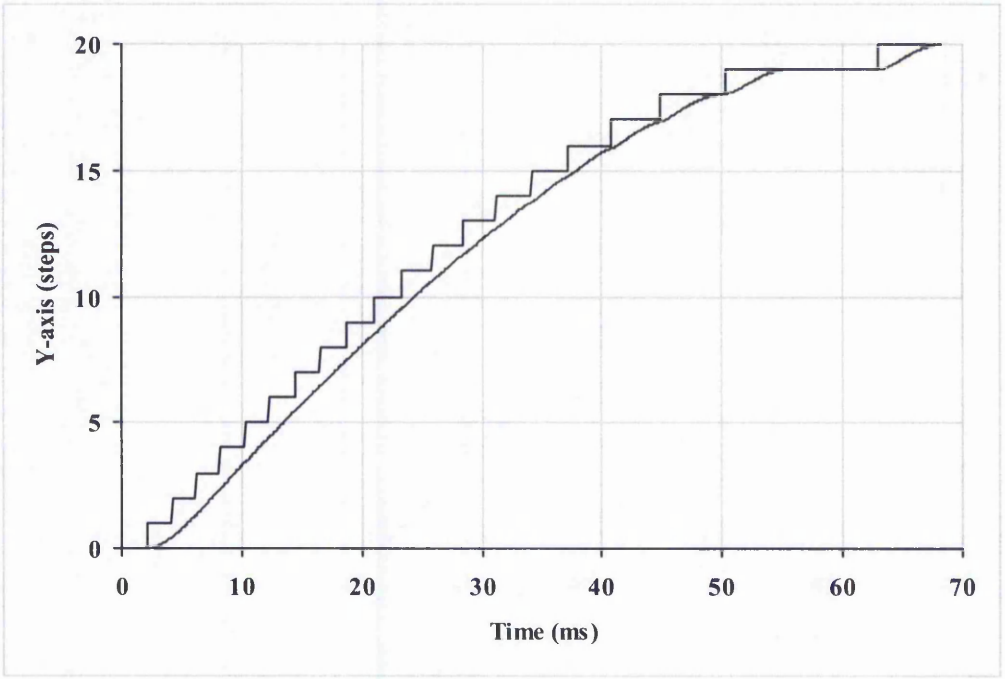


Figure 3-23: Position Error of New Circular Arc Interpolation shown in Figure 3-21 (Second Order Simulation). The largest position error here is 0.93. The error is positive if it is outside the required circle and negative inside.



(a)



(b)

Figure 3-24: Plot of Motion along the 2 Axes of the New Circular Interpolation shown in Figure 3-21 (Second Order Simulation). The zero order simulation is also shown to indicate the times of pulses.

From Figure 3-23, it can be seen that the largest position error for Second Order simulation is 0.93 of a step. Most of the time throughout the interpolation the position error is within half of a stepper motor step. The largest position error occurs at the end of the interpolation (near x=0) and it is also 0.6 of a step at the beginning. An improvement to the algorithm, the Half Step technique, is able to reduce the largest position error considerably, and is explained in detail in the next section.

As in the straight line example, the relation between the Zero Order case and Second Order simulation can be seen clearly with a diagram of the X and Y-axis movement with respect to time. Figure 3-24 shows the X and Y-axis movements when interpolated using the new circular interpolation algorithm. The "steps" in the diagrams show the motion in a Zero Order case, while the curves show the position simulated using the Second Order simulation.

3.5 The New Half Step Technique

It has been noted from the simulation diagrams of the new algorithms, particularly the position error diagrams, that the algorithms generate a smooth path but the accuracy of the circular arc is not very good at the beginning and end of the arc. The problems are highlighted in Section 3.5.1 while the developed solution, the Half-Step Technique, will be discussed in detail in Section 3.5.2 and Section 3.5.3.

3.5.1 Problems with the New Circular Arc Interpolation Algorithm

In Figure 3-25 and Figure 3-26, the simulated output of the path generated using the new interpolation algorithms show two sections of the path that have been analysed in more detail, because the errors are greatest there. The sections are encircled with dotted lines. Section A shows that at the start of motion the Y-axis motor has moved several steps before the X-axis motor starts to move, whereas in fact it would be better for the X-axis to start to move a little earlier. From Section B, it can be seen that the end position is reached only at the last pulse interval for both axes, even though the actual value on the Y-axis for the required path is very close to the final value during the last few steps on the X-axis. The last pulses for each axis are simultaneous, giving the diagonal line at 135° in the path when Zero Order simulation is used. Even with the Second Order simulation the last part of the path is close to the same diagonal line.

Figure 3-25: Plot of New Circular Arc Interpolation (Zero Order Simulation) (as shown in Figure 3-20).

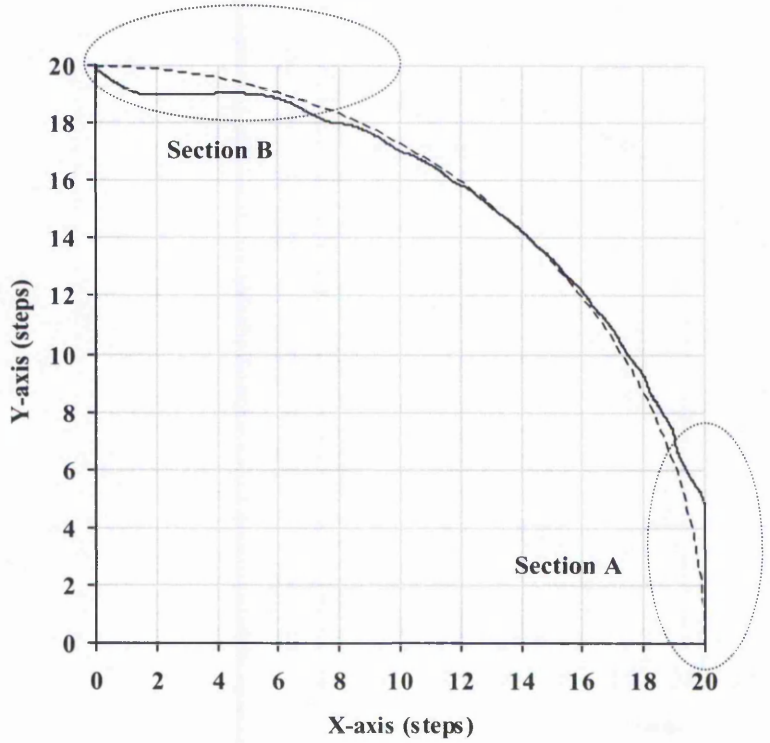


Figure 3-26: Plot of New Circular Arc Interpolation (Second Order Simulation) (as shown in Figure 3-21).

3.5.2 The Half-Step Technique for Circular Arc Interpolation

Investigations into the new interpolation algorithms have revealed that a possible solution to reduce the problems at Sections A and B is to adjust the pulse timings. The new interpolation algorithm relies on the geometry of the path. At every increment of one step in any particular axis, a command pulse is generated. A modification of the algorithm is to adjust the generation of pulse timings so that each pulse occurs half way between the two steps instead of at the end of each whole step, as illustrated in Figure 3-27.

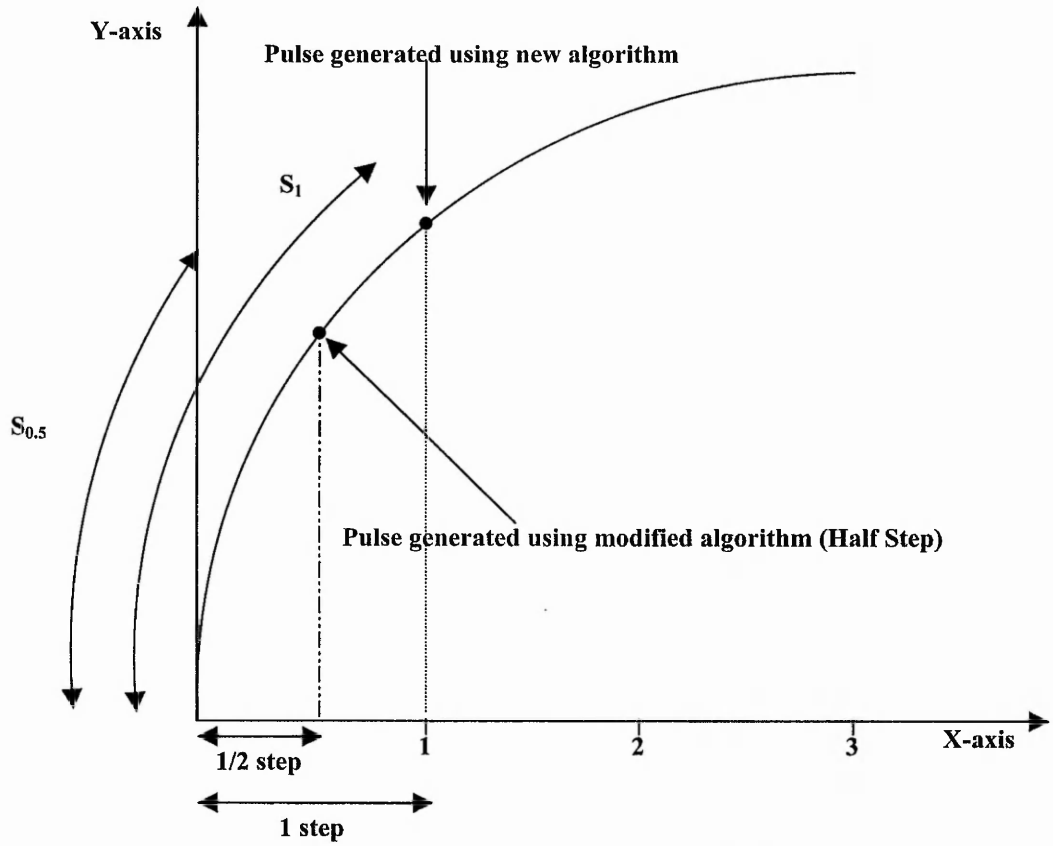


Figure 3-27: Illustration of Half-Step vs Full-Step.

With the new interpolation algorithm, the first command pulse timing is generated when the value of S (distance along the curve) reaches S_1, as in equation (3.15). Instead of using S_1 to calculate the first X-axis pulse timing, the half-step technique makes use of $S_{0.5}$, which is the distance along the arc when the X-axis position has been changed by a total of half a step. The next command pulse uses the distance at one and a half step in

X-axis, $S_{1.5}$. The equation for the half-step technique is obtained by replacing n_x in equations 3-15 by n_x - 0.5, as follows:

$$s_x(n_x) = R\cos^{-1}\left(\frac{(n_x - 0.5)(-D)}{R} + 1\right)$$
$$s_y(n_y) = R\sin^{-1}\left(\frac{(n_y - 0.5)D}{R}\right) \tag{3-22}$$

Thus, to replace equation (3-18), the pulse timings are calculated as:

$$t_x(n_x) = \frac{R}{V}\cos^{-1}\left(\frac{(n_x - 0.5)(-D)}{R} + 1\right)$$
$$t_y(n_y) = \frac{R}{V}\sin^{-1}\left(\frac{(n_y - 0.5)D}{R}\right) \tag{3-23}$$

for n_x = 1,2,3,…,destination X step and n_y = 1,2,3,…,destination Y step

Figure 3-28 and Figure 3-30 show the simulation results of the New Half Step technique after Zero Order simulation and Second Order simulation respectively. The same parameters have been used as in Figure 3-25 and Figure 3-26.

From Figure 3-28, the X motor moves earlier in Section A while the last Y step comes earlier in Section B, resulting in the simulated path being closer to the required one in both sections. This confirms that the errors are smaller in Sections A and B using this simulation. The position errors throughout the interpolation are plotted in Figure 3-29. The largest position error is 0.62 step, comparing with the new algorithm without the half step technique which has largest position error of 0.97 step. Because there is no motion in both axes at the same time using the Zero Order simulation, the New Half Step algorithm has a larger average error, 0.06, comparing to the Full Step new algorithm which has a value of 0.03 step. This will not be the case for other simulation results, which will be discussed in Chapter 6.

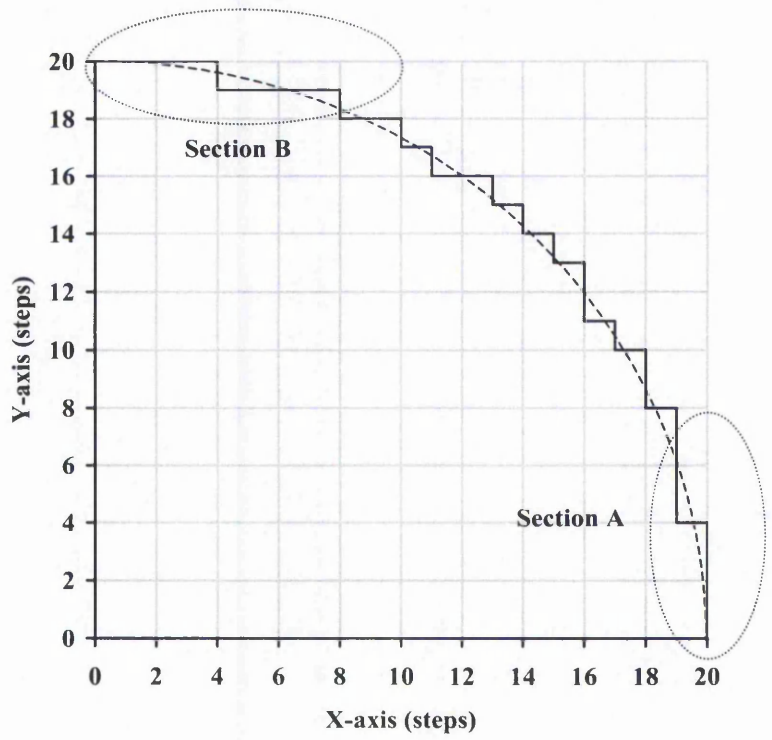


Figure 3-28: Plot of New Circular Arc Interpolation (with Half-Step Technique) (Zero Order Simulation). Full Line is simulated path. Dashed Line is required path. Position errors in Sections A and B can be seen to be smaller than in Figure 3-25.

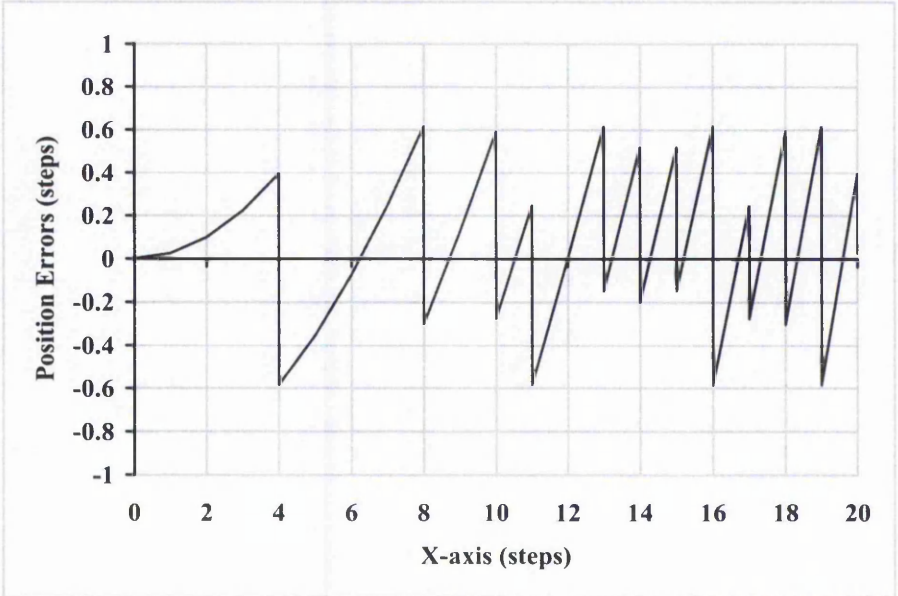


Figure 3-29: Position Error of New Circular Arc Interpolation with Half-Step Technique (Zero Order Simulation). The largest position error here is 0.62, which is smaller than the largest error in Figure 3-22 (0.97). The error is positive if it is outside the required circle and negative inside.

In Figure 3-30 (the Half Step technique using Second Order simulation), it again appears that the generated path follows the required arc more closely than for the Full Step case in Figure 3-21. The largest position error with the Full Step interpolation is 0.93 step while this error is reduced to 0.28 step with the Half Step interpolation, showing an improvement of 0.65 step. The position error throughout the circular arc interpolation is shown in Figure 3-31. The average error for the Half Step algorithm is indeed smaller (0.02 step) than with the Full Step algorithm (-0.07 step). With the Half Step algorithm, the error fluctuates around zero more closely than for Full Step. Chapter 6 will include a full evaluation of the improvement of this Half Step technique when compared with the Full Step interpolation algorithms.

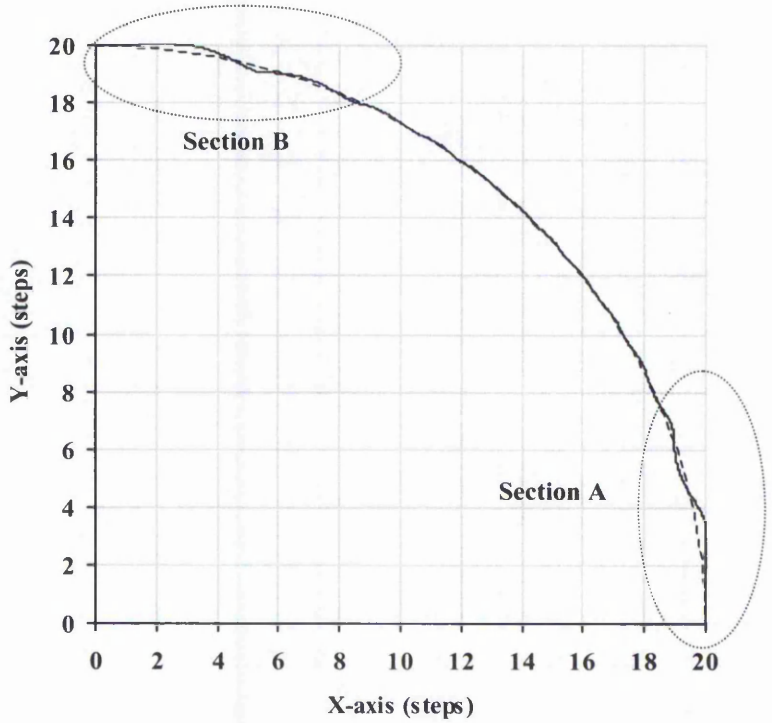


Figure 3-30: Plot of New Circular Arc Interpolation with Half-Step Technique (Second Order Simulation). The damping factor is 0.7, which are typical for a stepper motor control system. Full Line is simulated path. Dashed Line is path required. It can be seen that at both the beginning and end of the path the simulated path is much closer to the desired path than with the full-step algorithm (compare with Figure 3-21).

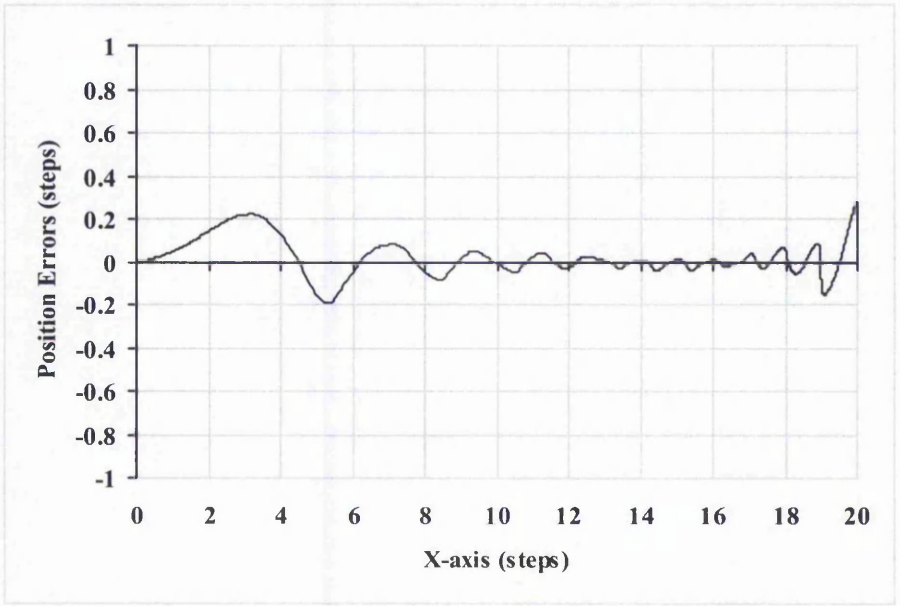


Figure 3-31: Position Error for New Circular Arc Interpolation with Half-Step Technique (2^{nd} Order Simulation). The largest position error here is 0.28 step. The error is positive if it is outside the required circle and negative otherwise. It shows a big improvement when compared to its full-step counterpart (largest position error = 0.93 step) in Figure 3-23.

Table 3-3 summarises the largest position errors for both the New Full and Half Step circular arc interpolation algorithms. The average errors are also included. The Second Order Simulation is expected to be closer to the actual value. Therefore, it can be concluded that the New Half Step circular arc interpolation algorithm is able to follow the arc more closely than its Full Step counterpart.

Table 3-3: Position Error Comparison for Full and Half Step Circular Arc Algorithm.

	Zero Order Simulation		Second Order Simulation	
	Largest Error	**Average Error**	**Largest Error**	**Average Error**
Full Step	0.97	-0.03	0.93	-0.07
Half Step	0.62	0.06	0.28	0.02

The speed variations for both Full and Half Step are similar because the times between pulses are very similar (the Full Step speed plots are shown in Chapter 6). The speed simulation assumes that one motor step is completed exactly when the next command pulse is received. It should be noted that one motor step is of 0.01 mm for a typical

stepper motor system. The speed used here is steps/s rather than m/s. The top speed for Figure 3-32 is 500 steps/s which is equivalent to 5 mm/s for the typical motor. Figure 3-32 and Figure 3-33 show a smooth speed variation for each axis. The speed changes happen gradually, reducing the likelihood of vibrations caused by abrupt changes in speed.

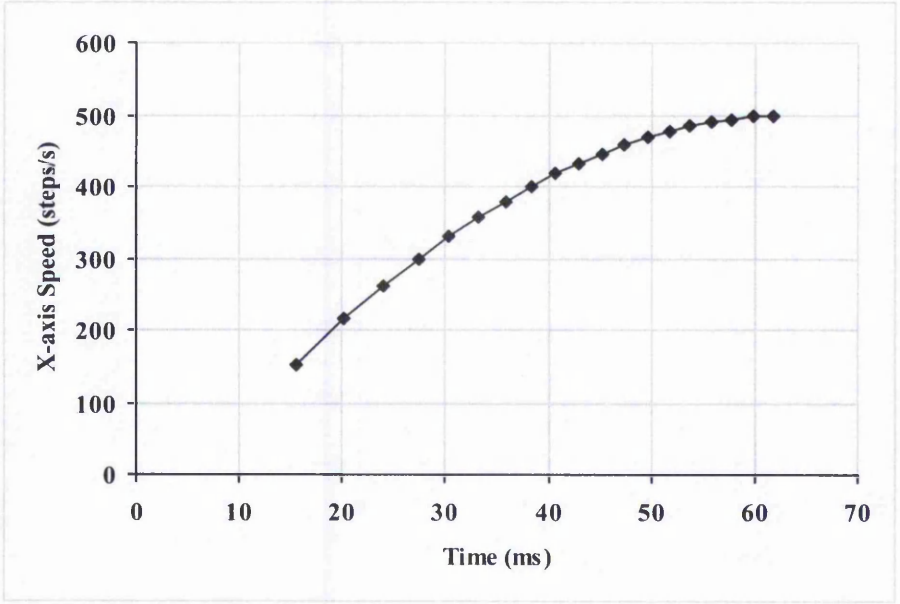


Figure 3-32: X-axis Speed for Anti-Clockwise Circular Arc in 1st Quadrant (Half Step).

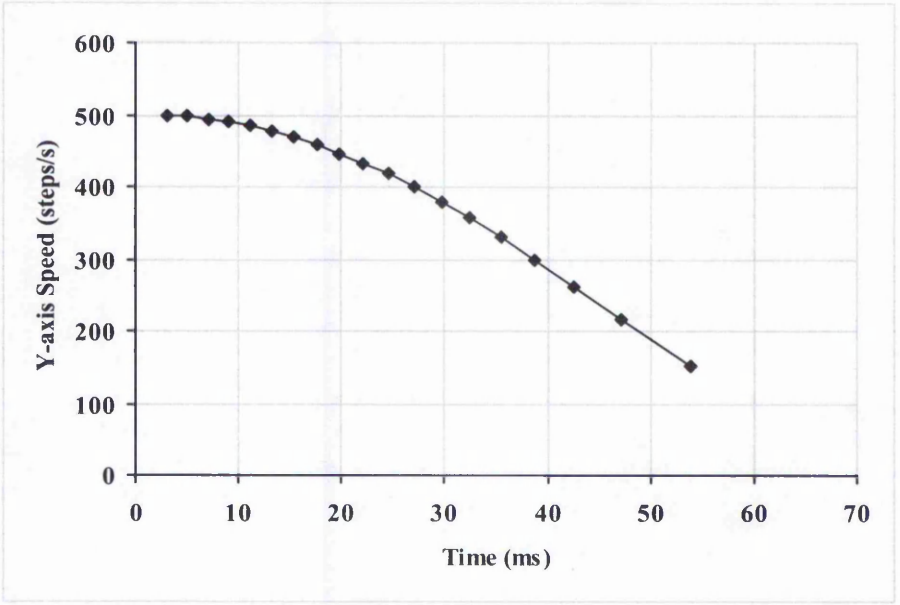


Figure 3-33: Y-Axis Speed for Anti-Clockwise Circular Arc in 1st Quadrant (Half Step).

3.5.3 The Half Step Technique for Linear Interpolation

The Half Step technique for the new interpolation algorithm can equally well be implemented with the linear interpolation. The Half Step technique was first developed for the circular arc to reduce the position error at the beginning and the end of the interpolation motion, as discussed in Section 3.5.2. A further investigation has been made to check the effect of this technique on the linear interpolation. It is found that the Half Step technique helps to reduce the position errors and the average position error is found to be closer to zero.

The equation for the Half Step linear interpolation is obtained again by replacing n_x in equations 3-5 by n_x - 0.5:

$$s_x(n_x) = \frac{(n_x - 0.5)D}{\cos\theta}$$
$$s_y(n_y) = \frac{(n_y - 0.5)D}{\sin\theta} \qquad (3\text{-}24)$$

Thus, to replace equations (3-8), the pulse timings are calculated as:

$$t_x(n_x) = \frac{(n_x - 0.5)D}{V\cos\theta}$$
$$t_y(n_y) = \frac{(n_y - 0.5)D}{V\sin\theta} \qquad (3\text{-}25)$$

for n_x = 1,2,3,...,destination X step and n_y = 1,2,3,...,destination Y step

Figure 3-34 and Figure 3-35 shows the simulated path from the Zero Order and Second Order simulation for the New Half Step linear interpolation. From Figure 3-34, there is no simultaneous motion at any instant. This does not cause problem for the position error because the timing for both axes can be very close together but they do not coincide. The Second Order simulation shows that the path is expected to follow the required one very closely.

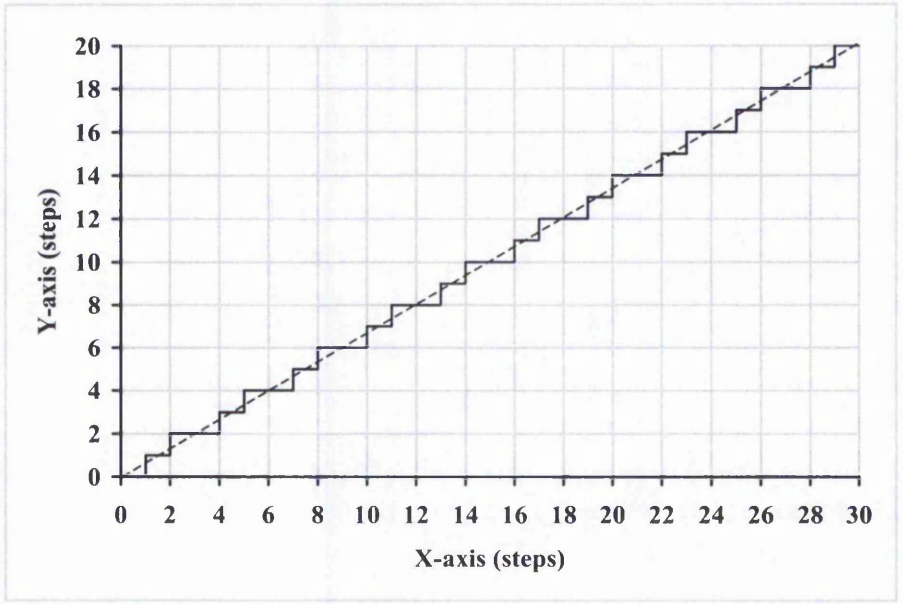


Figure 3-34: Plot of New Half-Step Linear Interpolation (Zero Order Simulation). Full Line is simulated path. Dashed Line is required path (compare with Figure 3-14).

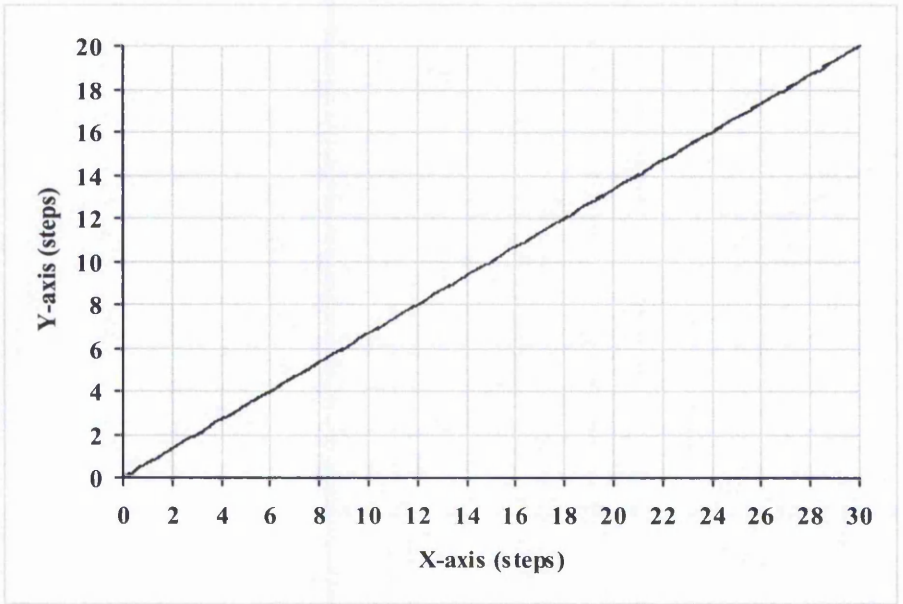


Figure 3-35: Plot of New Half-Step Linear Interpolation (2nd Order Simulation). The damping factor is 0.7, which are typical for a stepper motor control system. Full Line is simulated path. Dashed Line is required path. The two lines are very close together.

From Figure 3-36, the largest position error for the Zero Order simulation is 0.55 of a step. With the Zero Order simulation, there is no difference in the largest position error

between the Full and Half Step interpolation (compare Figure 3-16). However, the average position error is smaller for the Half Step (0.00 step) when compared to its Full Step counterpart (-0.14 step). From Figure 3-37, a lower value for the largest position error can be seen with the Half Step interpolation when simulated using the Second Order simulation, giving 0.06 step. The largest position error for the Full Step linear interpolation using the Second Order simulation is 0.19 step from Figure 3-17.

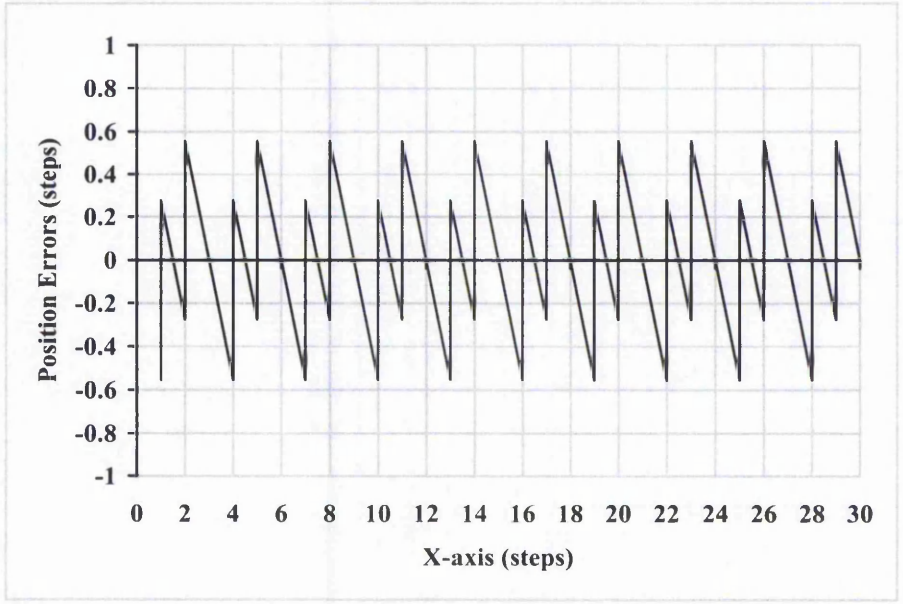


Figure 3-36: Position Error of New Half Step Linear Interpolation (Zero Order Simulation). The largest position error here is 0.55 (same as the error for full-step interpolation in Figure 3-16). The error is positive if it is above the required line and negative below.

Figure 3-37 shows the position error when using the Second Order simulation. The position error is smaller than that for Zero Order simulation in Figure 3-36. This is because the simulated dynamics of the stepper motor smooth out the motion, thus bringing the generated path closer to the required one. In practice, the Second Order simulation is likely to be closer to the actual path generated. It is noted that the Half Step technique not only benefits the circular arc interpolation but also the linear interpolation, by reducing the position error. Table 3-4 summarises the largest position errors and average errors for both the New Full and Half Step linear interpolation algorithms. The largest position error for the Half Step linear interpolation is 0.06

compared to 0.19 for the Full Step interpolation in Figure 3-17. The average error is again 0.00 (compare with the average error in Figure 3-17, -0.13 step). It can be concluded from the table that the New Half Step linear interpolation algorithm is able to follow the line more closely than its Full Step counterpart. The X and Y axis speed variations are shown in Figure 3-38 and Figure 3-39. The speeds for both axes are constant, reducing the likelihood of any vibrations.

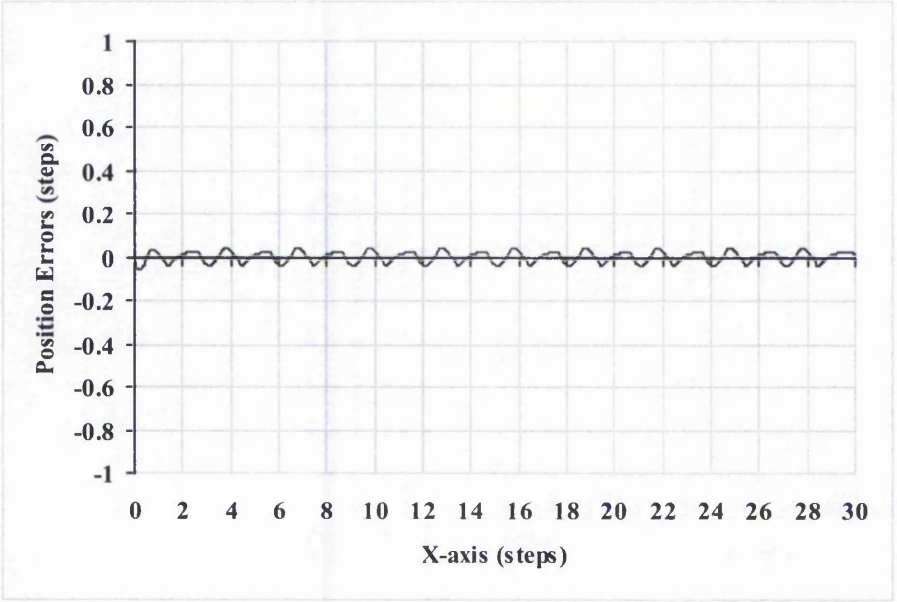


Figure 3-37: Position Error of New Half Step Linear Interpolation (Second Order Simulation). The largest position error here is 0.06 (compared with the full-step interpolation which has a largest error of 0.19 in Figure 3-17). The error is positive if it is above the required line and negative below.

Table 3-4: Position Error Comparison for Full and Half Step Linear Algorithm.

	Zero Order Simulation		Second Order Simulation	
	Largest Error	**Average Error**	**Largest Error**	**Average Error**
Full Step	0.55	-0.14	0.19	-0.13
Half Step	0.55	0.00	0.06	0.00

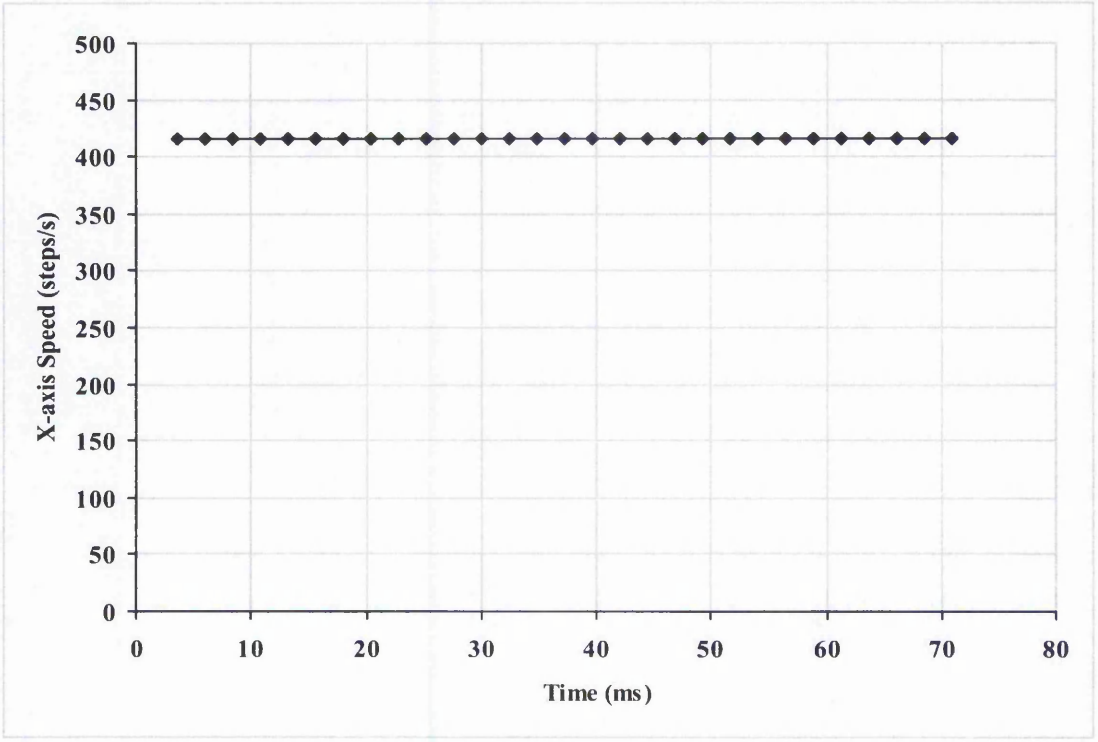


Figure 3-38: X-axis Speed for Straight Line (Half Step).

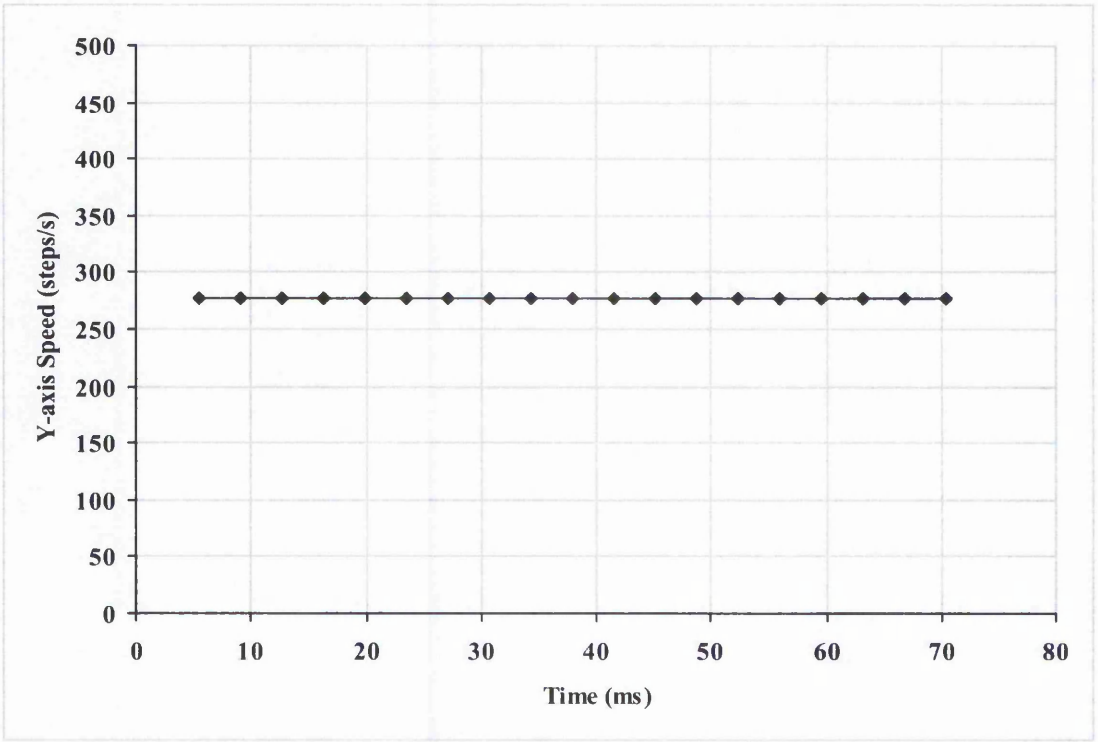


Figure 3-39: Y-axis Speed for Straight Line (Half Step).

3.6 Summary

The three main problems with the existing interpolation and the machining process in general have been described in Chapters 1 and 2, namely the vibrations, position errors and varying resultant speed. Based on these findings, the Author has proposed and developed new interpolation algorithms to reduce these problems for lines and arcs. This chapter has described the new algorithms in detail.

Most existing interpolation algorithms use position interpolation, where the coordinates of the end points are first obtained, and from this information, new interpolated points are calculated. The new algorithms can be categorised as timing interpolation. The idea is to derive an equation for the timing of every individual command pulse on each axis according to the path geometry, thus gaining ultimate control over the smoothness of the stream of command pulses. From equations (3-8) and (3-18), it is clear that the feedrate is included in the pulse timing generation formulae. These formulae have been chosen so that the resultant speed is able to keep close to the desired feedrate while allowing smooth motion on each individual axis. Axiomatic Technology Ltd uses part of these ideas by generating short straight line segments in this way. Since each shape is made of a number of short line segments, the resultant shape will not be smooth when they are joined, particularly for shapes with high curvature.

To simulate the path generated, four simulation methods have been used. Two of these methods have been presented briefly in this chapter together with examples (they are the Zero Order simulation and a Second Order simulation). Chapter 5 explains all these simulation methods in detail. A further analysis of the new interpolation algorithms using simulation has revealed that a further improvement is possible. Therefore, a New Half Step approach has been developed. In the Half Step technique, the pulse timing is adjusted so that the pulse occurs half way between two steps instead of the end of each step. With the New Half Step technique, based on simulation results, the position errors are reduced further. From the simulation of speed on each axis, it can be seen that the pulses are generated smoothly and the pulse rate either stays constant or changes gradually, as required for the shape.

4 The New Acceleration Algorithms

The new interpolation algorithms presented in Chapter 3 are appropriate for low speed machining, where the speed for each individual axis is still within the pull-in speed of the stepper motor. However, in order to perform high-speed machining, an acceleration algorithm is needed. The motor can be accelerated without losing step, provided that the pull-out torque limit at any particular speed is not exceeded. The pull-out torque limit depends on speed and is a characteristic of each motor/drive system, as explained in Section 4.1. The acceleration and deceleration planning process is important to ensure smooth motion at the beginning and the end of the machining with low chance of vibrations.

This chapter describes the two new acceleration algorithms developed by the Author. These two acceleration algorithms, together with the new interpolation algorithms, form a major part of the motion controller. As explained in Section 3.1, the new interpolation algorithms make use of a novel approach of calculating the pulse timing for every individual pulse. Thus, the new acceleration algorithms involve modifications of the pulse timings to accommodate the need for acceleration and deceleration.

Two common types of acceleration algorithm used for high-speed machining are linear and parabolic acceleration, where the speed changes, respectively, linearly or parabolically with time. The minimum speed of the motor depends on the rotor and load inertia [72] and this will also be the speed at which the motor can be planned to start moving from rest.

4.1 Linear Vs Parabolic Acceleration

An acceleration/deceleration process must be included in the interpolation routine for a machine tool in order to avoid shock and vibration, which would otherwise occur at the beginning and end of the machining of the given shape [73]. In addition, the acceleration/deceleration time is also an important criterion, because it affects the entire machining time.

First of all, the acceleration time can be compared for the linear and parabolic acceleration for the same path. Figure 4-1 shows examples of the speed profiles for the machining process comparing the linear and parabolic acceleration for different values of linear acceleration. The areas below the two curves in each case represent the distance travelled, so they will be the same for both. Figure 4-1(a) illustrates the case where linear acceleration is equal to the initial acceleration for parabolic. Figure 4-1(b) shows the case when linear acceleration is slightly lower than the initial acceleration for parabolic. For both of these, the machining time for the linear acceleration is shorter than parabolic acceleration. In Figure 4-1(c) where linear acceleration is much less than initial parabolic acceleration, it can be seen that the parabolic acceleration has reached the required maximum speed long before the linear acceleration is able to achieve that speed. This depends on being able to start with a much higher acceleration. In addition, the deceleration procedure can be started earlier than for its linear counterpart, resulting in shorter overall machining time. As the linear acceleration rate decreases from (a) to (b) and then to (c), it will reach a situation somewhere between (b) and (c) for which the two machining times are equal.

The dynamic performance of a motor can be described by the torque-speed curve as shown in Figure 4-2. It shows the maximum possible torque at each speed. From this figure, it is clear that as the operating speed (maximum speed) increases, the pull-out torque becomes lower. Acceleration is proportional to torque. Therefore, as the operating speed increases, there is a decrease in the acceleration achievable for the stepper motor at that speed. Above that acceleration, the motor loses synchronisation. With linear acceleration the acceleration has a constant value, whereas for parabolic

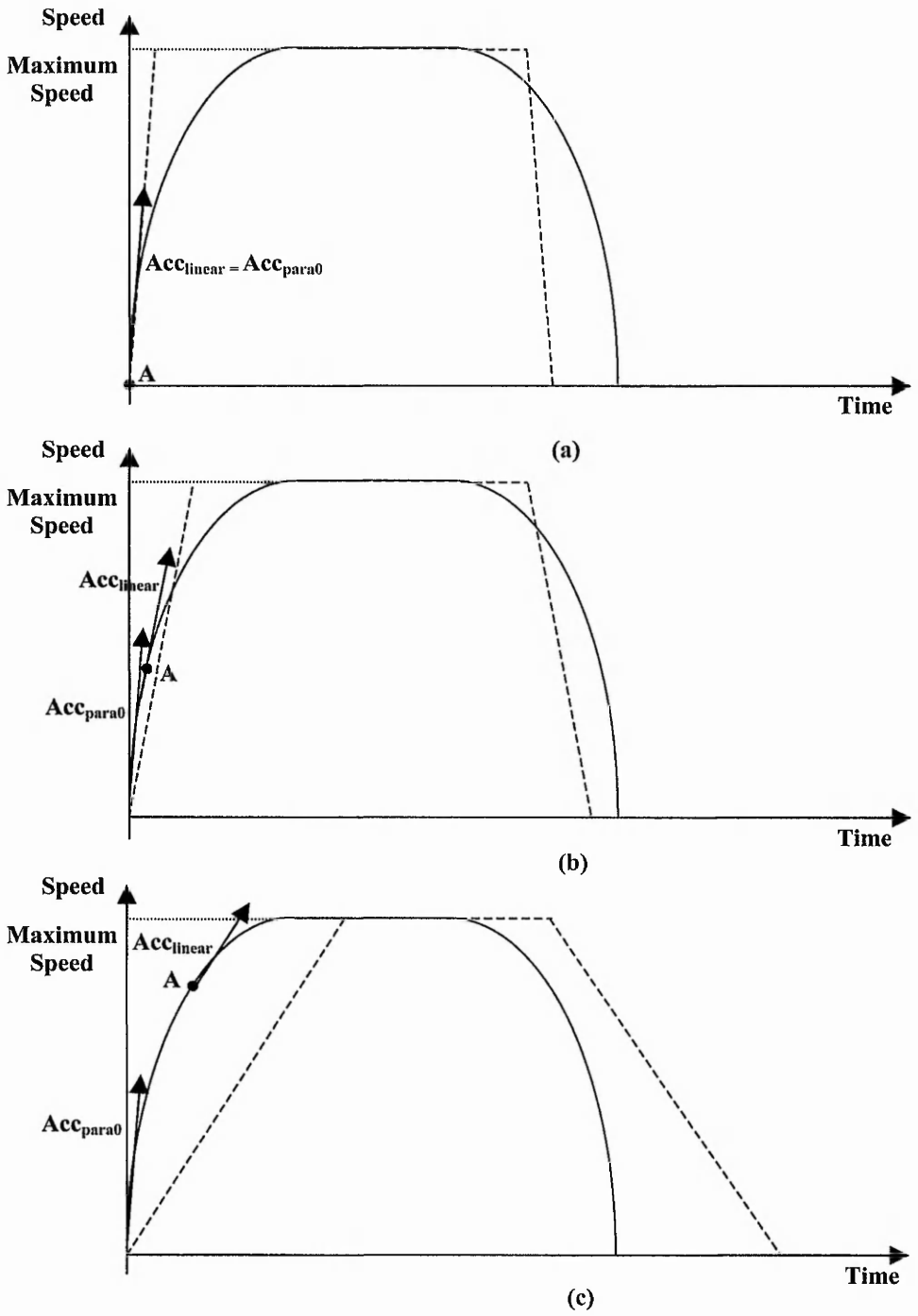


Figure 4-1: Illustration of Comparison of the 2 Speed Profiles: Linear (Dashed Line) and Parabolic (Full Line) for the same maximum speed and same distance but different values of linear acceleration (Acc_{linear}) and maximum parabolic acceleration (Acc_{para0}). (a) $Acc_{para0} = Acc_{linear}$; (b) $Acc_{para0} > Acc_{linear}$; (c) $Acc_{para0} >> Acc_{linear}$. At point A the parabolic acceleration has the same value as the linear acceleration. In cases (a) and (b), the linear acceleration is quicker whereas in case (c) the parabolic acceleration is quicker.

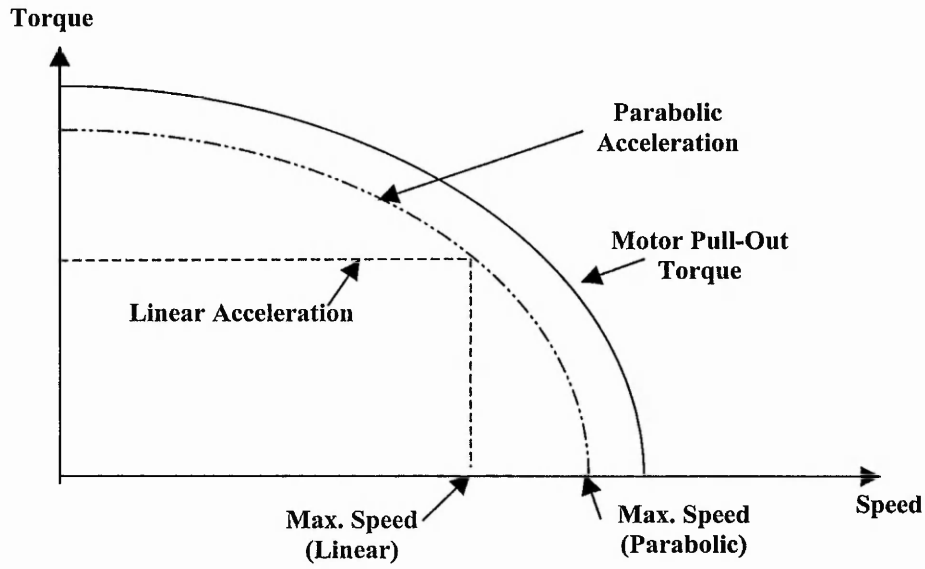


Figure 4-2: Typical Pull-out Torque/Speed Characteristic Curve (Full Line) together with Graphs of Torque used by Linear and Parabolic Acceleration. In the linear case, less of the available torque is used.

acceleration the graph of acceleration against speed is a parabola on its side [78]. Therefore from Figure 4-2, it is expected that, because it occurs at low speeds, the maximum possible acceleration for parabolic acceleration is higher than the maximum possible acceleration for linear acceleration.

As discussed earlier in this section, higher acceleration is achieved at lower speeds because the torque-speed characteristic in Figure 4-2 indicates that this can be achieved. In other words, the parabolic acceleration is able to make greater use of the motor torque capability. It is illustrated in Figure 4-2 that the torque achievable with parabolic acceleration is closer to the motor characteristic.

4.2 The New Linear Acceleration Algorithm

Linear acceleration is the simplest acceleration algorithm where the acceleration and deceleration phase of the speed profile is taken to be linear with respect to time. That is the speed changes linearly with time in these phases while the acceleration or deceleration rate is constant, as shown in Figure 4-3 and Figure 4-4. In practice, the deceleration rate, A', is not always the same as the acceleration rate, A. Linear acceleration is the simplest, but it generally results in a longer acceleration time. The minimum speed of a motor depends on the rotor inertia and load inertia, and is the

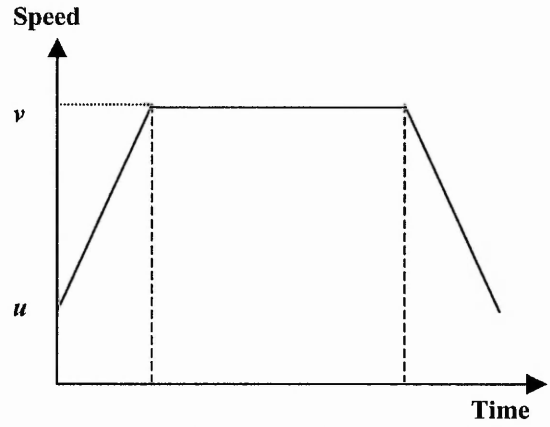


Figure 4-3: Illustration of Linear Acceleration (Speed against Time).

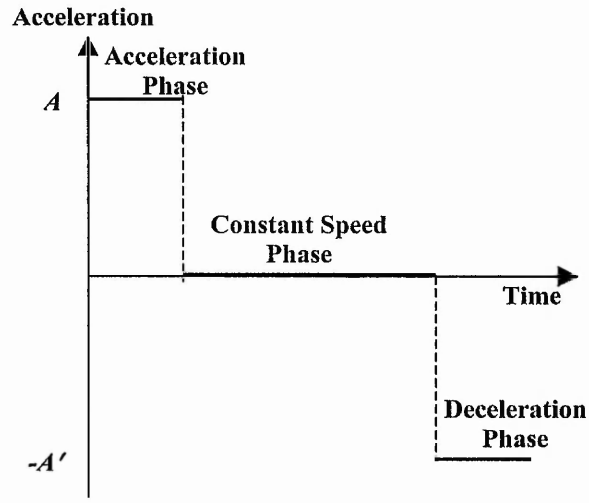


Figure 4-4: Illustration of Linear Acceleration (Acceleration against Time).

speed at which the motor can be run without needing acceleration or deceleration time. From that to the desired speed is where linear acceleration occurs.

As discussed in Section 2.4.1, many existing systems generate the linear ramping of the speed in block of pulses, where one block of pulses is dealt with each time. For example, during the acceleration phase, the first block of pulses will have a constant low speed while the following block of pulses are of slightly higher constant speed, and so on. The sudden transition in speed between these two blocks of pulses is likely to contribute to vibrations. Therefore, vibrations are expected to be reduced by using the new linear acceleration algorithm discussed here, which allows every individual pulse to be dealt with separately.

4.2.1 Calculation of Pulse Timings for Linear Acceleration

The first step for the derivation of the pulse timing formula whilst accelerating is to relate the time and distance travelled, when the acceleration is constant. This is done by the following equation:

$$s = ut + \frac{1}{2}at^2 \tag{4-1}$$

where

u = initial speed (pull-in speed)
a = acceleration (constant)
s = distance travelled

Solving equation (4-1) gives:

$$t = \frac{-u + \sqrt{u^2 + 2as}}{a} \tag{4-2}$$

where the positive root has been chosen. (If the negative root is used, the time will be negative which is not the required time, as illustrated in Figure 4-5.) The distance

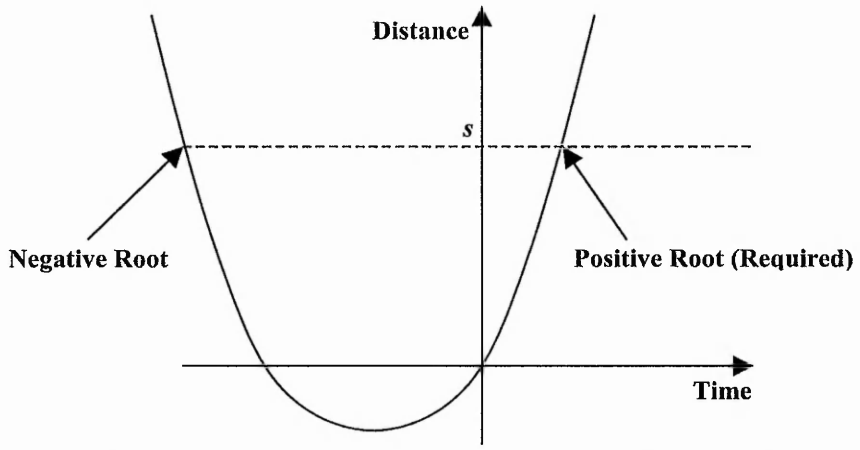


Figure 4-5: Illustration of Linear Acceleration (Distance against Time).

travelled is checked to determine whether the next command pulse should be accelerating or maintaining constant speed. To do this, the distance travelled throughout acceleration is equal to the area (only the acceleration phase) under the curve in Figure 4-3.

On the other hand, the deceleration phase is similar to the acceleration phase, just that the initial and final speed is switched and the addition process is changed to subtraction operation. (s and t are zero at beginning of deceleration).

$$s = vt - \frac{1}{2}at^2 \tag{4-3}$$

Solving equation (4-3) gives:

$$t = \frac{v \pm \sqrt{v^2 - a(2s)}}{a} \tag{4-4}$$

Equation (4-4) will generate a smaller time, t, when subtraction is performed rather than addition. The two times are both positive. The smaller one is the time when the position is first reached. (If the motion continued with negative acceleration, then it would eventually move backwards till the same position was reached again but this is not what is required. This is illustrated in Figure 4-6.)

$$t = \frac{v - \sqrt{v^2 - 2as}}{a} \tag{4-5}$$

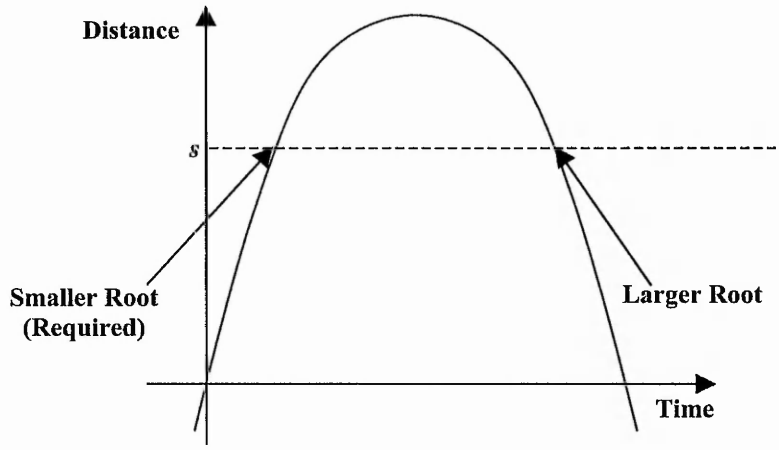


Figure 4-6: Illustration of Linear Deceleration (Distance against Time).

However, the time, t, in equation (4-5) is relative to the start of the deceleration phase. Thus, the pulse timing at the end of the constant motion phase (t_{last_cons}) has to be added to the pulse time in equation (4-5) to obtain the actual required pulse time. The time, t_{last_cons}, can be calculated using the total distance of the entire path and then finding the position, s_{last_cons}, at which the deceleration should start (by subtracting the deceleration distance).

$$t = t_{last\_cons} + \frac{v - \sqrt{v^2 - 2a(s - s_{last\_cons})}}{a} \tag{4-6}$$

The distance travelled, s, in equation (4-2) is to be obtained from the developed interpolation algorithm, equation (3-24) for half-step linear interpolation and (3-22) for half-step circular arc interpolation, which will be:

$$\begin{aligned} s_x(n_x) &= \frac{(n_x - 0.5)D}{\cos\theta} \\ s_y(n_y) &= \frac{(n_y - 0.5)D}{\sin\theta} \end{aligned} \tag{4-7}$$

for n_x = 1,2,3,…,destination X step and n_y = 1,2,3,…,destination Y step (Half-Step Linear Interpolation); and

$$s_x(n_x) = R\cos^{-1}\left(-\frac{(n_x - 0.5)D}{R} + 1\right)$$
$$s_y(n_y) = R\sin^{-1}\left(\frac{(n_y - 0.5)D}{R}\right) \quad (4\text{-}8)$$

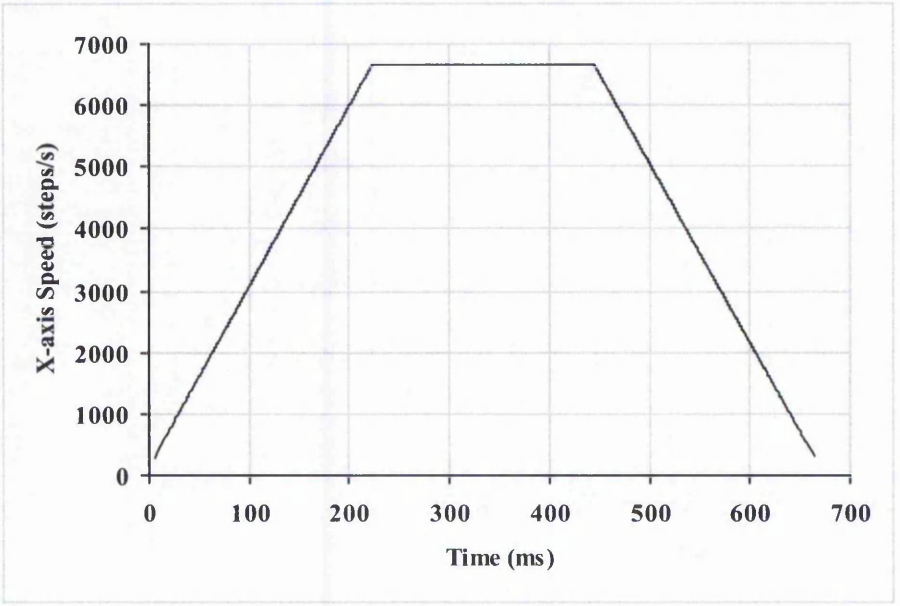
for n_x = 1,2,3,...,destination X step and n_y = 1,2,3,...,destination Y step (Half-Step Circular Arc Interpolation)

As explained in Sections 2.4 and 4.1, with linear acceleration it may take a longer time to reach the required speed, although it is the simplest acceleration. When using linear acceleration, much of the available torque is not utilised. Linear acceleration was the Author's first approach of generating acceleration algorithm for the new interpolation algorithms. Due to the disadvantages highlighted above, a parabolic acceleration algorithm has been developed. Parabolic acceleration is more desirable in terms of the acceleration time and the use of available motor torque.

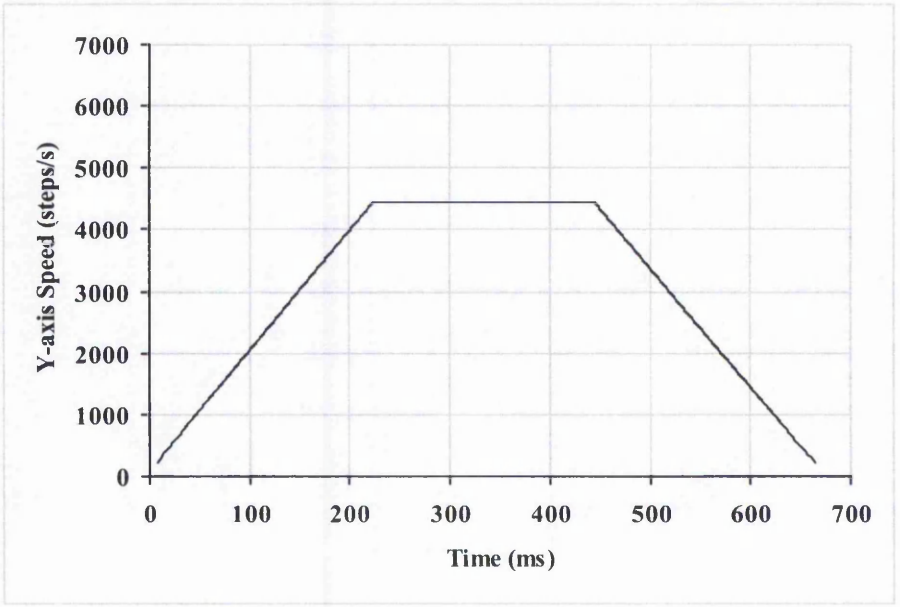
4.2.2 Simulation of Speed with Linear Acceleration

A speed diagram illustrating an example of the new linear acceleration when implemented with the new linear interpolation algorithm is shown in Figure 4-7, while Figure 4-8 for linear acceleration on the new circular arc interpolation algorithm. The acceleration rate is 35,000 steps/s^2 (0.35 m/s^2).

The X-axis speed for circular arc is composed of the X-axis speed without acceleration (follows part of a sine wave curve, as shown in Figure 3-32) and the linearly accelerating speed. The result is a rather gentle increase in speed at the beginning while the drop in speed during deceleration is steeper due to the linearly decreasing speed. The reverse is true for the Y-axis speed because without acceleration it follows part of a cosine wave, as in Figure 3-33.

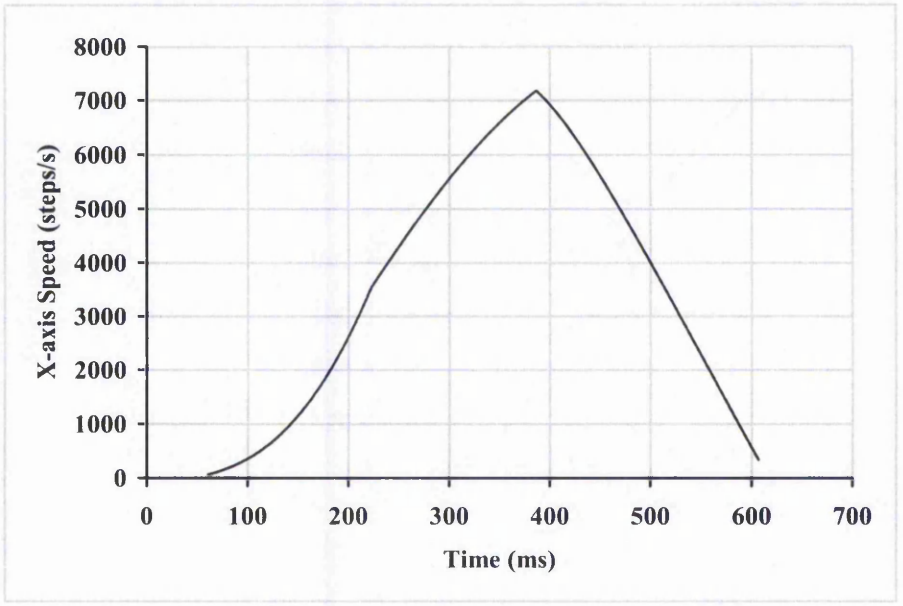


(a)

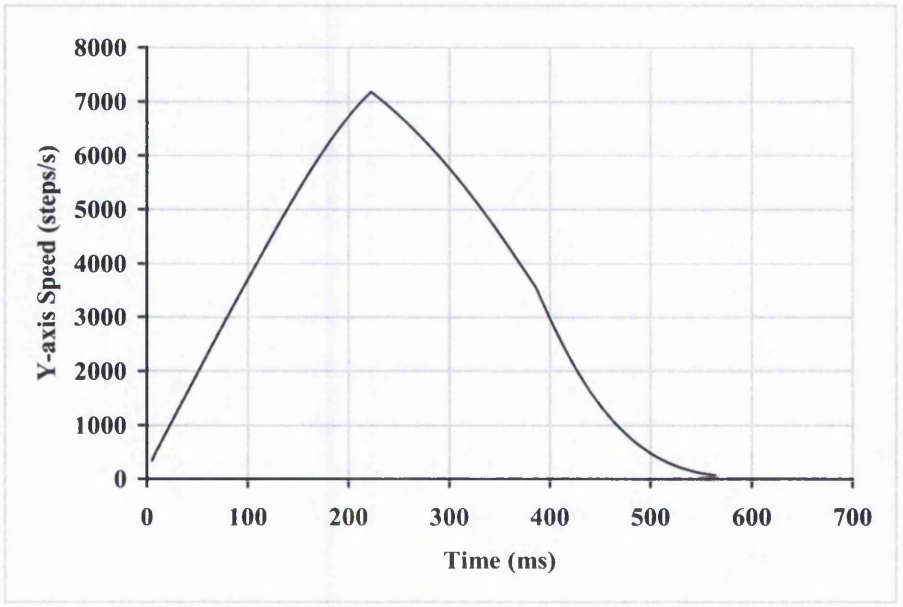


(b)

Figure 4-7: Simulation of Speed with Linear Acceleration and Deceleration for the New Half Step Linear Interpolation for (a) X-axis; (b) Y-axis. (X-axis = 3000 steps, Y-axis = 2,000 steps. Acceleration = 35,000 pulses/s^2. Maximum Resultant Speed = 8,000 steps/s. Acceleration ends at 220 ms. Deceleration starts at approximately 445 ms).



(a)



(b)

Figure 4-8: Simulation of Speed with Linear Acceleration and Deceleration for the New Half Step Circular Arc Interpolation for (a) X-axis; (b) Y-axis. (Arc = Anticlockwise 1st Quadrant with radius of 2,000 steps. Acceleration = 35,000 pulses/s^2. Maximum Resultant Speed = 8,000 steps/s. Acceleration ends at 220 ms. Deceleration starts at approximately 390 ms).

4.3 The New Parabolic Acceleration Algorithm

Investigations into the linear acceleration algorithm have revealed problems with the acceleration/deceleration time and torque utilisation. Therefore, a new parabolic acceleration algorithm has been developed for use with the new interpolation algorithms. The output is again pulse timings which allow the stepper motors to accelerate or decelerate smoothly.

Basically, the rate of acceleration is changed continuously with each and every step pulse by adjusting the timing between pulses. In fact, the acceleration rate decreases linearly to zero during the acceleration phase. One of the reasons to justify this selection is because stepper motor provides excellent torque at low speeds, which is up to 5 times the continuous torque of a brush motor of the same frame size or double the torque of the equivalent brushless motor. Palmin [72] found that acceleration time can be reduced by up to 90 percent of that of the conventional linear ramping technique.

Figure 4-9 illustrates how the speed changes over time when employing the parabolic acceleration. The shape of the curve during the acceleration and deceleration phases is each part of a parabola. Figure 4-10 shows the plot of the acceleration values for the situation in Figure 4-9.

4.3.1 Calculation of Pulse Timings for Parabolic Acceleration

The equation for speed during the acceleration and deceleration phases were developed by Palmin [72] as follows:

$$v = pt^2 + qt + v_0 \tag{4-9}$$

where

$$p = \frac{-(v_m - v_0)}{T^2}; \tag{4-10}$$

$$q = -2pT \tag{4-11}$$

v_0 is the minimum speed, v_m is the maximum speed and T is the time taken for acceleration or deceleration.

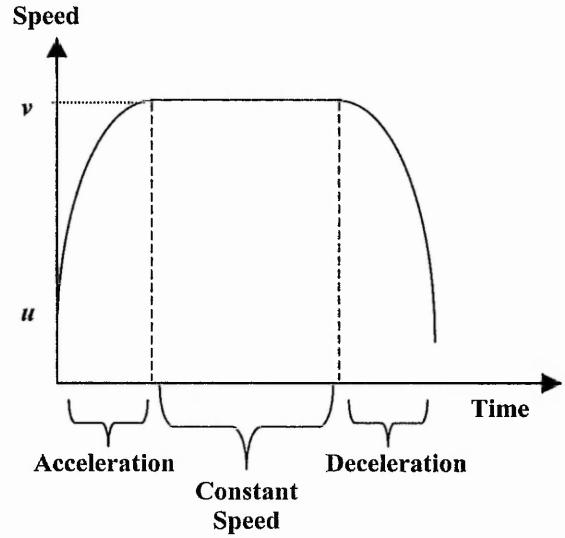


Figure 4-9: Parabolic Acceleration (Speed vs Time).

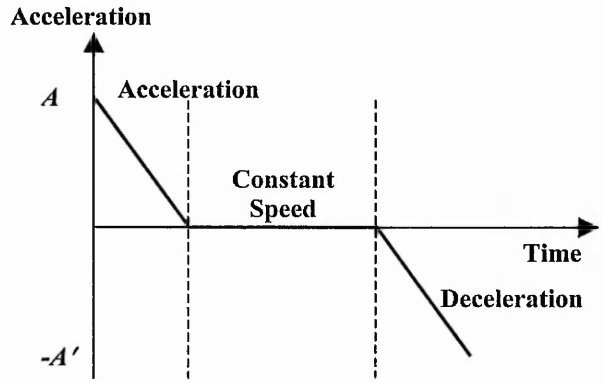


Figure 4-10: Parabolic Acceleration (Acceleration vs Time).

Palmin uses an approximation technique to find the pulse timings for acceleration.

$$v_n = v_{n-1} + \Delta v t \tag{4-12}$$

However, this approximation gives pulse timings which may have varying resultant speed. Building upon equation (4-9), the Author has related the pulse timings to distance travelled in order to simplify the task of finding the pulse timings. Distance travelled, s, can be obtained by integrating equation (4-9) to give equation (4-13),

$$s(t) = \int v dt = \frac{pt^3}{3} + \frac{qt^2}{2} + v_0 t \tag{4-13}$$

The plot of distance against time is illustrated in Figure 4-11 for the case where there is no constant phase. Now that the relationship between the distance travelled and time has been obtained, the next step is to find the time in terms of the distance travelled. To do this, the cubic equation (4-14) has to be solved to find a solution to $s(t) - s(n) = 0$. The value $s(n)$ is obtained from the New Half Step interpolation (for line or circular arc) for the nth pulse.

$$f(t) = s(t) - s(n) = \frac{pt^3}{3} + \frac{qt^2}{2} + v_0 t - s(n) = 0 \tag{4-14}$$

The root required must be greater than 0 and before the end of the motion. Therefore it must be the smaller of the two positive roots. To solve the cubic equation in (4-14), the Author has used the Newton-Raphson iteration. A description of this application of the Newton-Raphson method is given in Appendix A.

The Newton-Raphson method is a way of finding a root of a complicated function which is difficult to solve algebraically. It uses an iterative process to approach one root of equation [79][80].

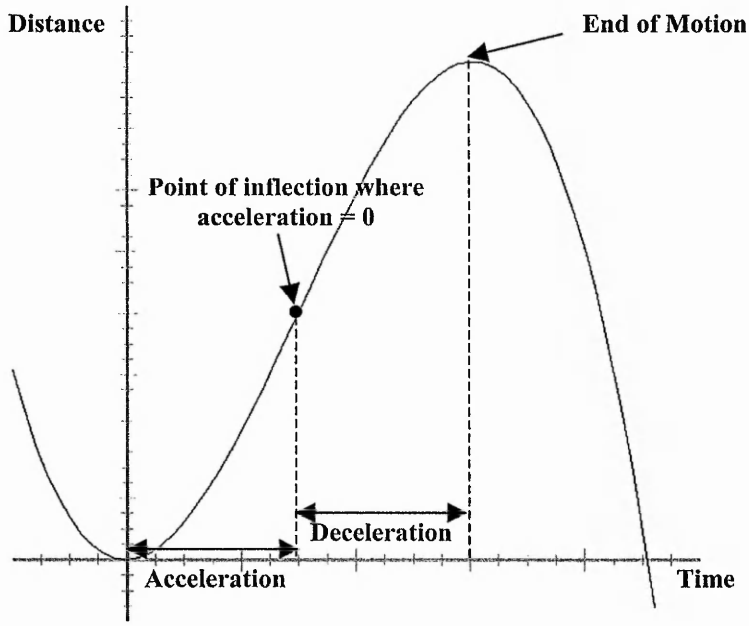
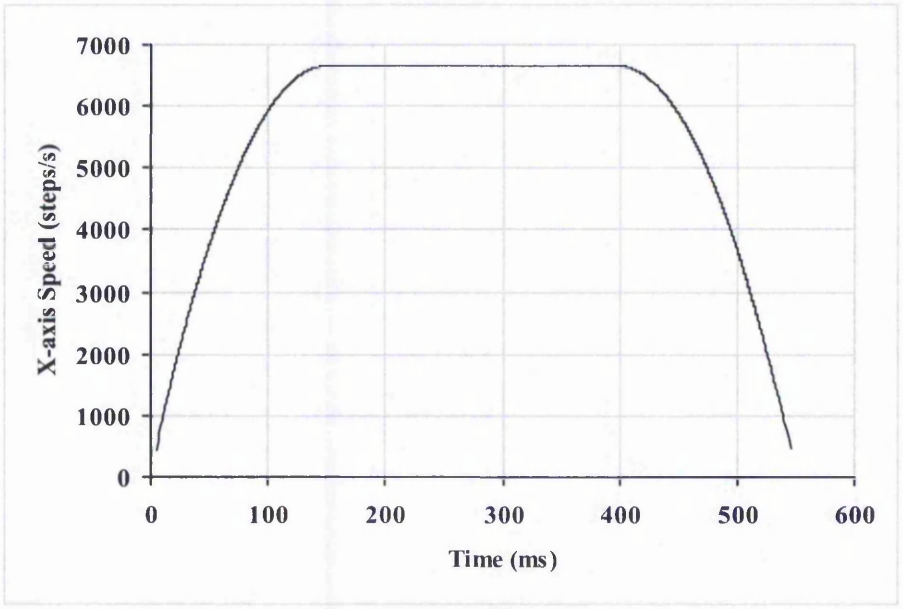


Figure 4-11: Illustration of Distance Travelled in Time. Constant phase has time length of 0. The starting speed, V_0, used in this graph is 0.

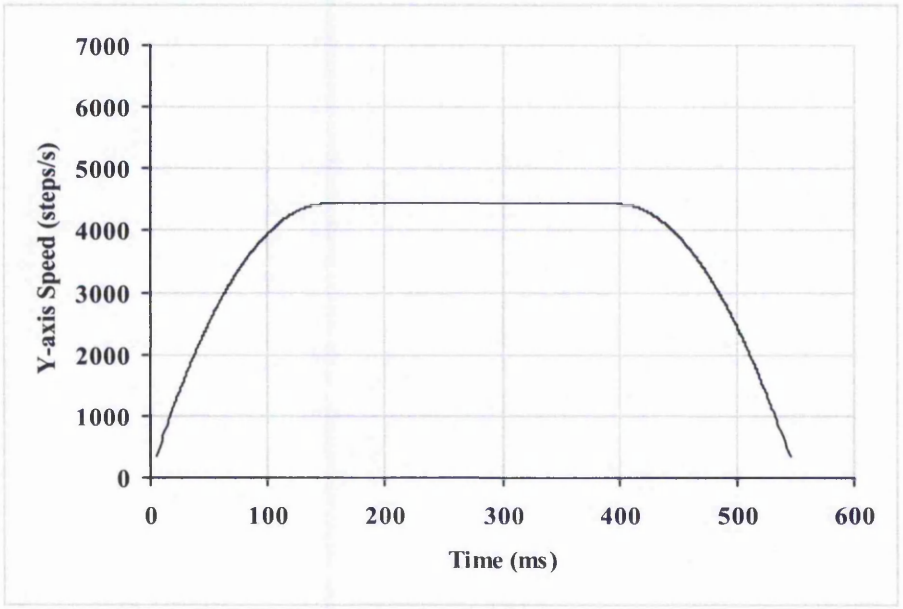
4.3.2 Simulation of Speed with Parabolic Acceleration

A speed simulation of the new parabolic acceleration when implemented with the new linear interpolation algorithm is shown in Figure 4-12. From this figure, it can be noted that the acceleration and deceleration follow a parabolic shape.

Similarly the speed simulation when parabolic acceleration is employed with the new circular arc interpolation is illustrated in Figure 4-13. Without the acceleration, between 150 ms and 340 ms, the X-axis speed follows part of a sine wave, as shown in Figure 3-32. Together with the parabolic acceleration profile, a rather gentle increase in speed is noticed at the beginning while the drop in speed during deceleration is steeper. The reverse is true for the Y-axis speed because without acceleration it follows part of a cosine wave, as shown in Figure 3-33.

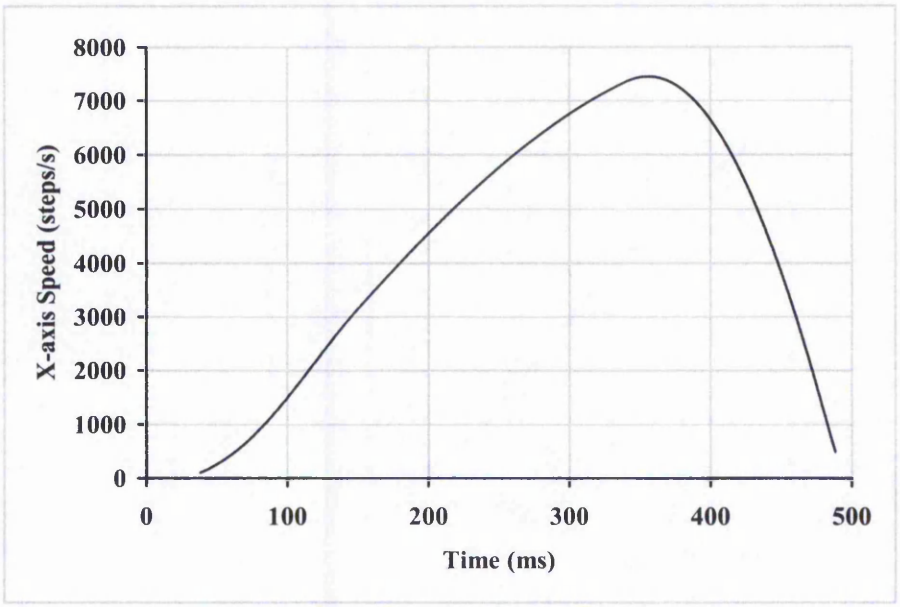


(a)

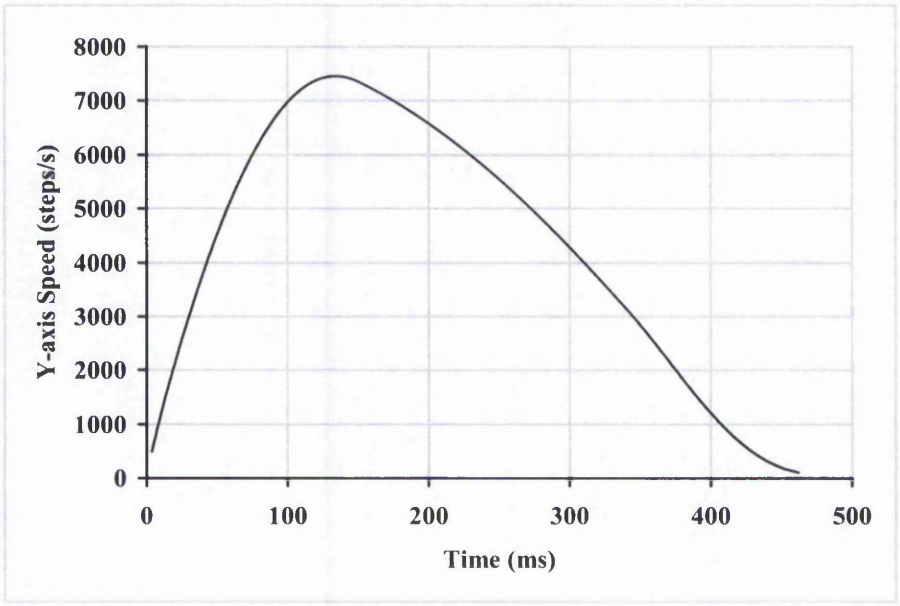


(b)

Figure 4-12: Simulation of Speed with Parabolic Acceleration and Deceleration for New Half Step Linear Interpolation for (a) X-axis; and (b) Y-axis. (X-axis = 3000 steps, Y-axis = 2000 steps. Maximum Resultant Speed = 8000 steps/s. Acceleration ends at 150 ms. Deceleration starts at approximately 395 ms).



(a)



(b)

Figure 4-13: Simulation of Speed with Parabolic Acceleration and Deceleration for New Half Step Circular Interpolation for (a) X-axis; and (b) Y-axis (Arc = Anticlockwise 1st Quadrant with radius of 2000 steps. Maximum Resultant Speed = 8000 steps/s. Acceleration ends at 150 ms. Deceleration starts at approximately 340 ms).

4.3.3 Explanation of the Shape of the Speed Curves

In some cases the curves obtained under acceleration may be difficult to understand. The speeds for interpolation of a line with the linear and parabolic acceleration are shown in Figure 4-7 and Figure 4-12, respectively. It can be seen that the acceleration changes linearly or parabolically. This is because the speed without acceleration is constant on each axis. On the other hand, the speeds during acceleration in Figure 4-8 and Figure 4-13 do not look linear or parabolic, because the speed on each axis changes even without acceleration. In fact, without acceleration, each would follow part of either a sine or cosine wave. To explain Figure 4-13, it has been deduced that increasing in speed in X-axis for the circular arc without the acceleration algorithm together with increasing in overall speed used for the acceleration algorithm will result in a very gently increasing in speed in the X-axis (Figure 4-14(a)(c)(e)). When the speed in the Y-axis is almost constant without the acceleration algorithm and the speed is increasing for the acceleration algorithm, the result is increasing in the resultant speed in the Y-axis (Figure 4-14(b)(d)(f)). A similar explanation can be given for Figure 4-8.

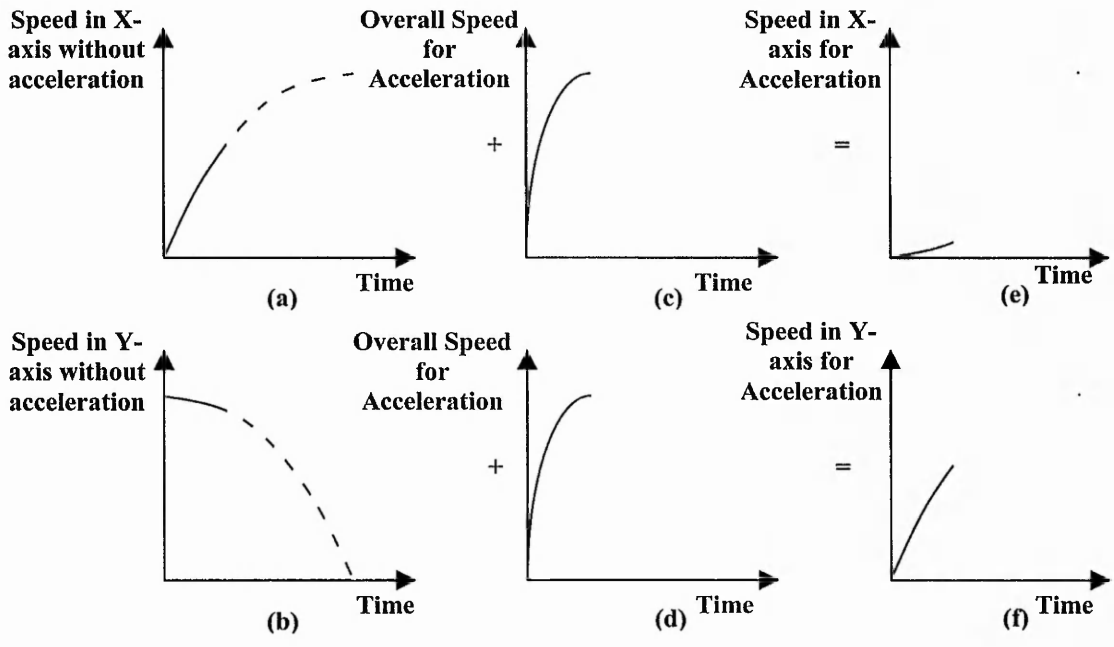


Figure 4-14: Combining Speeds: (a) Increasing in speed in X-axis without acceleration algorithm; (b) Initial speed in Y-axis without acceleration algorithm; (c)(d) Increasing in speed used for acceleration algorithm; (e)(f) Resulting speeds.

4.4 Comparison of Acceleration Algorithms

Section 4.1 explained the advantages of the parabolic acceleration over the linear acceleration. This section makes use of the new acceleration algorithms and an example to check the validity of the statements presented in Section 4.1. This resulting line has been plotted in Figure 4-15 and Figure 4-16. The parameters used for the example are as follows:

$X_{start} = 0$
$Y_{start} = 0$
$X_{end} = 3000$ steps (0.03m)
$Y_{end} = 2000$ steps (0.02m)
Feedrate = 8000 steps/s (4.8m/min or 0.08 m/s)
Starting speed = 200 steps/s (2 mm/s)
Acceleration Rate (Linear) = 0.35 m/s^2
Maximum Acceleration Rate (Parabolic) = 1.04 m/s^2

The characteristic to be analysed is the machining time. To do this, the X and Y axis positions in time have been calculated and are shown in Figure 4-17 and Figure 4-18. The total machining time with the linear acceleration is 665 ms. On the other hand, performing a parabolic acceleration on the same line will require only 545 ms. These results demonstrate that for this example parabolic acceleration will indeed require a shorter machining time than linear acceleration.

However, this advantage depends on the values of the constant acceleration (linear acceleration) and the maximum acceleration (parabolic acceleration). For example, if the two values are the same, then parabolic will take a longer time. However, this does not have to happen in practice, because the acceleration used for the linear acceleration must be within the pull-out torque limit of the motor throughout the acceleration and the pull-out torque decreases with speed, as illustrated in Figure 4-2. In practice, the machining time relationship between both accelerations is likely to be similar to the one shown in Figure 4-1(c).

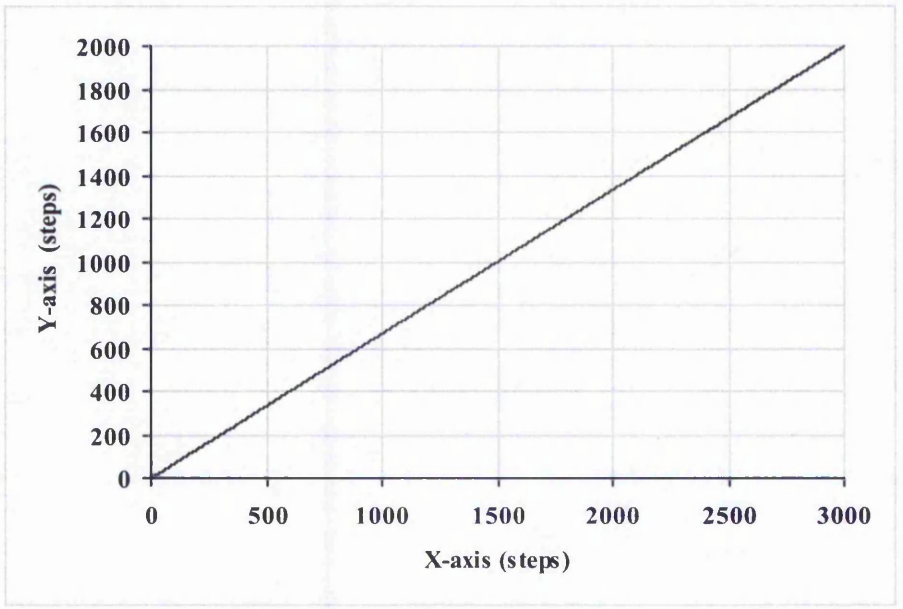


Figure 4-15: Position Plot for Linear Acceleration with Linear Interpolation for Line (X-axis = 3000 steps, Y-axis = 2000 steps, Feedrate = 8000 steps/s).

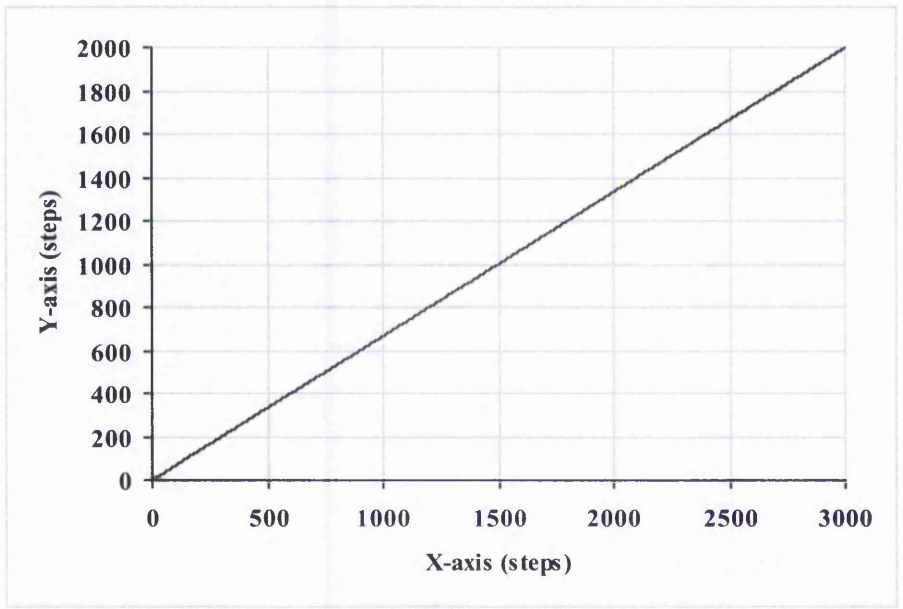
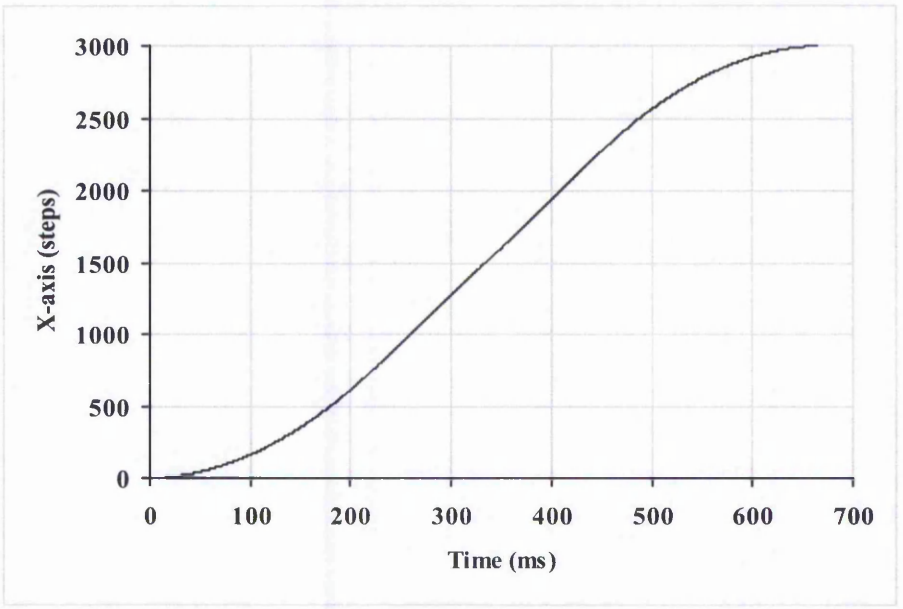
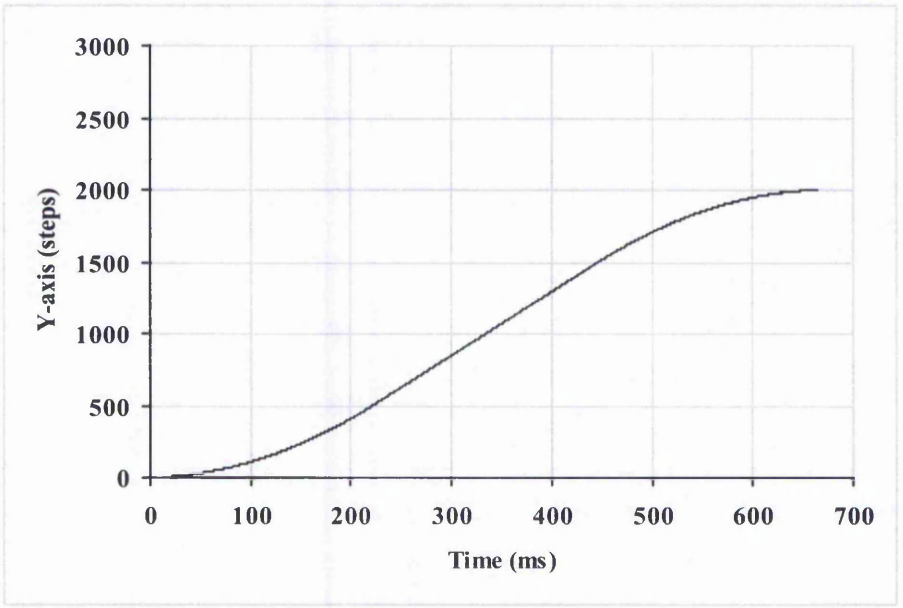


Figure 4-16: Position Plot for Parabolic Acceleration with Linear Interpolation for Line (X-axis = 3000 steps, Y-axis = 2000 steps, Feedrate = 8000 steps/s).

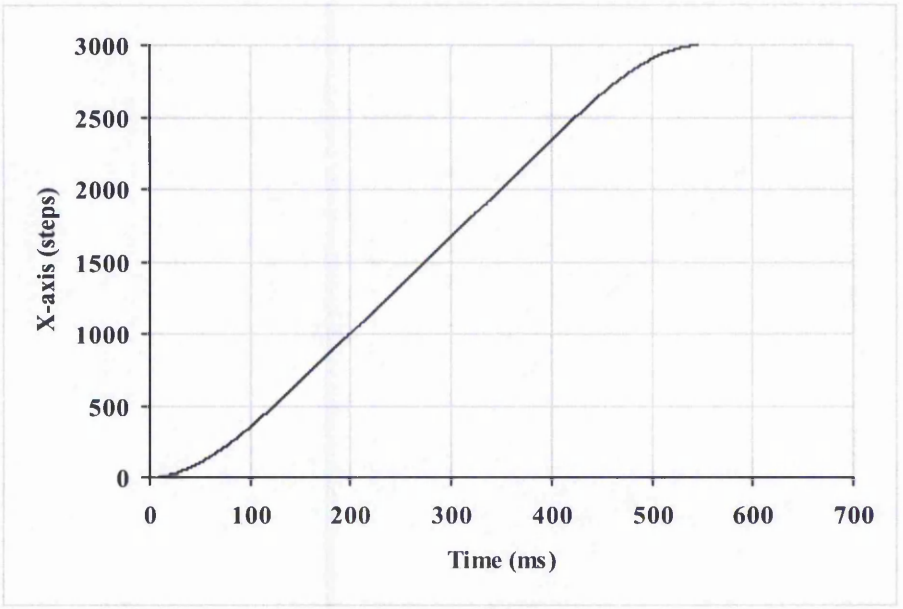


(a)

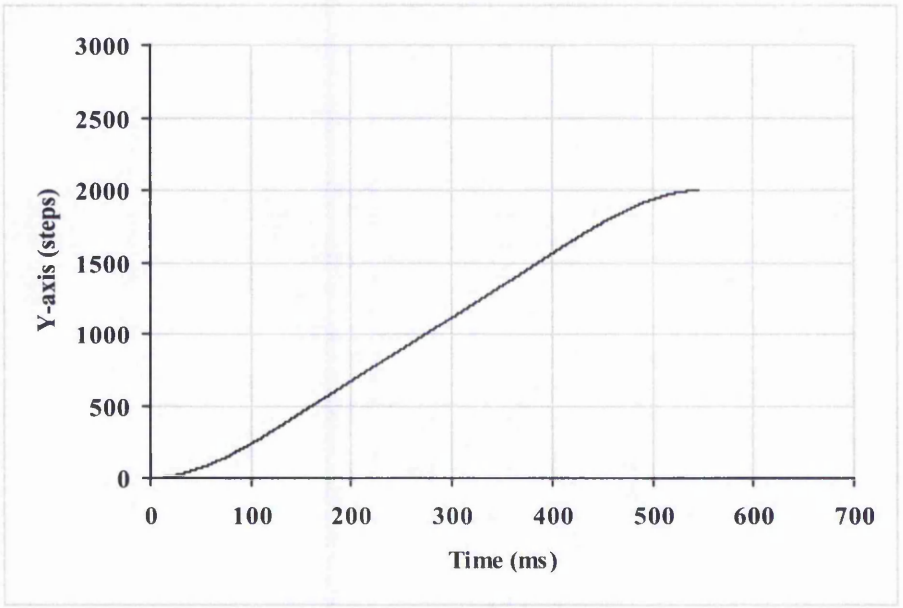


(b)

Figure 4-17: Position Plot for (a) X-axis; and (b) Y-axis, using Linear Acceleration with the Linear Interpolation for Line (X-axis = 3000 steps, Y-axis = 2000 steps, Feedrate = 8000 steps/s).



(a)



(b)

Figure 4-18: Position Plot for (a) X-axis; and (b) Y-axis, using Parabolic Acceleration for the Linear Interpolation for Line (X-axis = 3000 steps, Y-axis = 2000 steps, Feedrate = 8000 steps/s).

4.5 Summary

The new interpolation algorithms require acceleration algorithms to enable them to perform high-speed machining. In fact, it would not be very useful for a controller to have just the interpolation algorithm and no acceleration algorithm. Without acceleration, the speed during the whole machining has to be kept below the pull-in speed of the motor. Otherwise, the motor will lose step or stall. A typical pull-in speed is 0.13 m/min. Therefore, to machine a length of 1m on a single axis will take approximately 8 mins. When acceleration is used, the acceleration at each motor speed has to be kept below a certain limit which is constrained by the pull-out torque limit at that particular speed.

Both linear and parabolic acceleration have been explained in this chapter. Since the new interpolation algorithms deal with every individual pulse, the existing acceleration algorithms cannot be used. Instead, new linear and new parabolic acceleration algorithms have been developed for use with the new interpolation algorithms. Linear acceleration is more commonly used due to its simplicity of implementation but it has a drawback. The linear acceleration algorithm has a constant acceleration making the acceleration time longer than for the parabolic acceleration, provided that the starting acceleration (parabolic) can be made high enough. This is likely to be possible because the allowable acceleration at lower speed is high, can be seen in Figure 4-2.

Another advantage of the parabolic acceleration is that it makes more use of the available motor torque. The motor torque capability decreases with respect to speed. Therefore, higher acceleration can be used at lower speed but must be reduced for higher speeds. Parabolic acceleration has been developed to have high acceleration at lower speeds and lower acceleration at higher speeds.

Chapter 6 will include a more detailed evaluation of the acceleration algorithms. It should also be noted that both the acceleration algorithms can be used with any interpolation algorithms which produce pulse timings for constant speed interpolation.

5 Evaluation Platform Using Simulation

This chapter presents the four simulation methods that have been used to simulate the possible position of the generated path. They are the Zero Order simulation, Varying Rate First Order simulation, Constant Rate First Order simulation and Second Order simulation. Each of these simulation methods has its own advantage in simulating the resultant generated path. They all help to give an idea of the possible motion. The simulation algorithms have been discussed briefly in Section 3.4 but are now presented in detail.

5.1 Behaviour of a Stepper Motor

When a stepper motor performs a single step, the nature of the response is oscillatory, as illustrated in Figure 5-1. The overshoot is exaggerated here to make clear the oscillating nature of the resulting motion. This system can be likened to a mass which is located by a "magnetic spring", so the behaviour resembles the classic mass-spring characteristic [81]. Therefore, the Second Order simulation builds upon such mass-spring system.

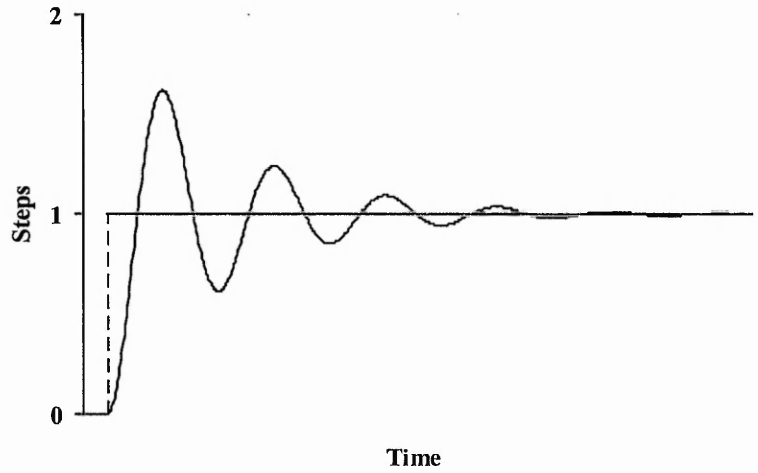


Figure 5-1: Typical Single Step Response for an Underdamped System (Damping factor = 0.15).

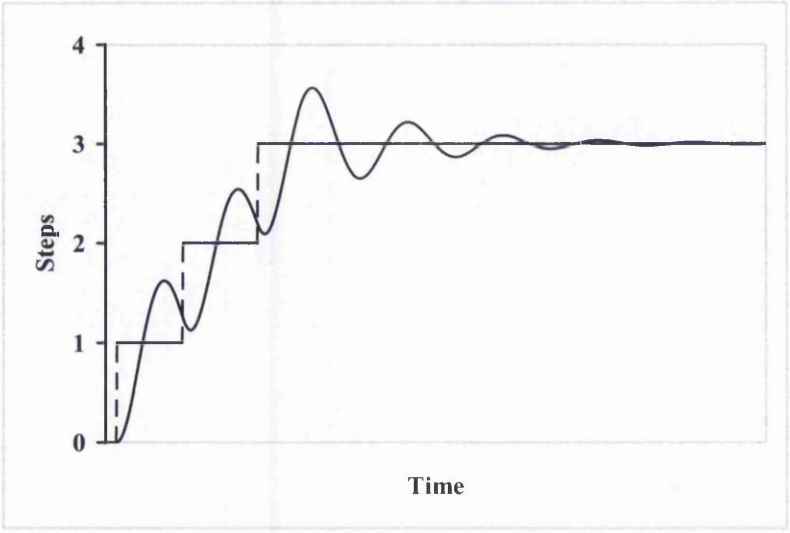


Figure 5-2: Multi-Step Operation of Stepper Motor for the System from Figure 5-1.

The multiple steps operation of the system in Figure 5-1 is illustrated in Figure 5-2. Damping is an indication of the rate of decay of a signal to its steady state value, related to settling time [3]. The possible situations for damped systems are underdamped, critically damped and overdamped system. The example system illustrated in Figure 5-1 is for an underdamped system. An underdamped system will overshoot its destination, then oscillate back and forth about its desired state. This can cause large inaccuracies and vibrations, which should be avoided. The degree of damping or underdamping is expressed as a damping factor which has value 1 for critically damped, greater than 1 for overdamped and between 0 and 1 for underdamped.

A system is critically damped when the response to a step change in desired position is achieved in the minimum possible time with no overshoot [3], as illustrated in Figure 5-3. On the other hand, an overdamped system, shown in Figure 5-4, will take a very long time to reach the desired position, asymptotically reaching the desired state without position overshoot. This will tend to put a higher burden on the driving motors.

In practice, there has to be a compromise between vibrations and settling time. Therefore, the practical damped process is normally close to the critical damped process value. An example of a practical damped stepper motor system is illustrated in Figure 5-5, which carries a damping factor of 0.7.

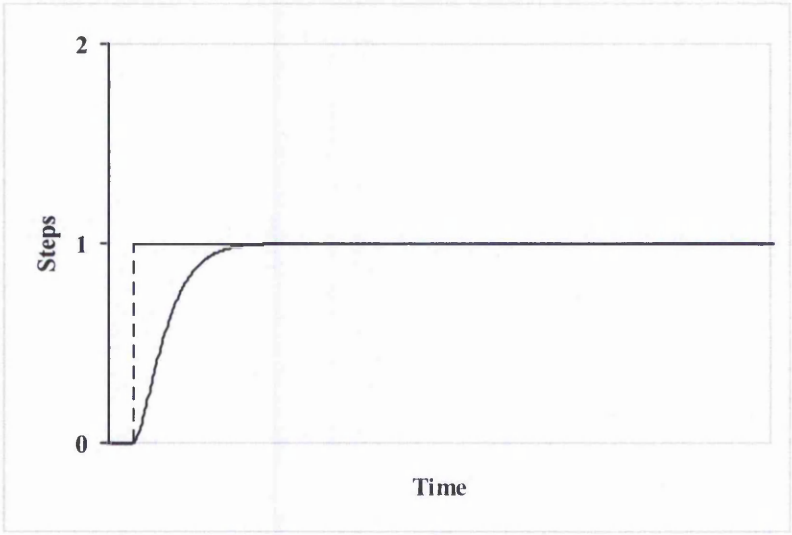


Figure 5-3: Typical Single Step Response for Critically Damped System (Damping Factor = 1).

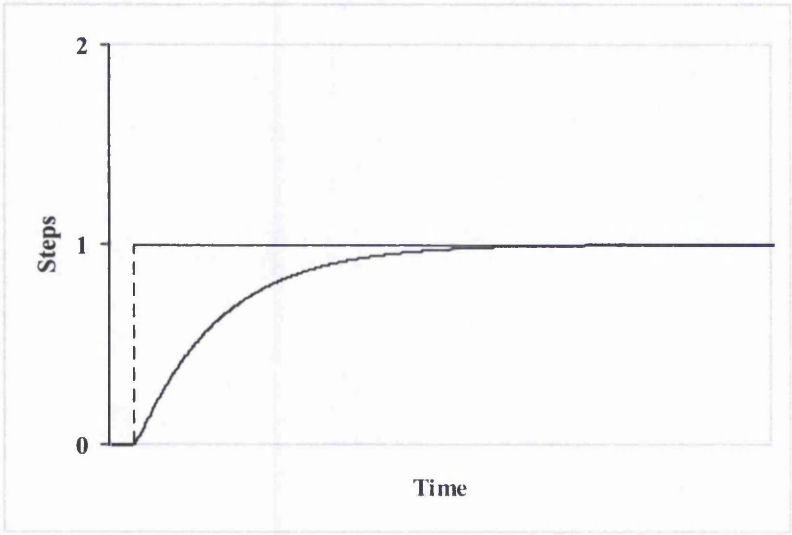


Figure 5-4: Typical Single Step Response for Overdamped System (Damping Factor = 2.5).

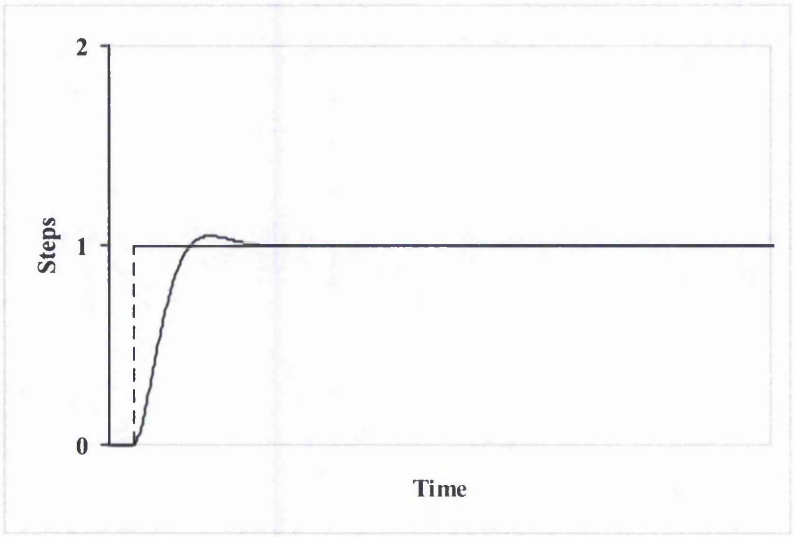


Figure 5-5: Practical Damped Stepper Motor System (Damping Factor = 0.7).

5.2 Simulation of Position

The simplest simulation is the Zero Order simulation where the motor is assumed to respond instantaneously to each command pulse. This simulation is impractical but it is useful to show the order in which the command pulses are sent. It also gives an idea of the path but without the detail.

There are two types of First Order simulation methods used, both of which have useful properties. The first type is the Varying Rate First Order simulation. With this simulation, the motion of every motor step varies linearly in time until the next step command pulse arrives. This simulation is suitable for simulating high speed motion because time between command pulses will be short for high speed interpolation. The motion appears smoother than with the Constant Rate First Order simulation.

For lower speed interpolation, the Constant Rate First Order simulation will be more appropriate. This simulation still assumes that motor step motion varies linearly with time until the step is completed. However, with this simulation, the time for this linear motion is constant for every motor step. The motion appears less smooth but it allows a step to be completed before the next one begins, as in the case for very slow speeds.

The behaviour of a stepper motor control system is similar to that of a mass which is located by a "magnetic spring" and can be modelled using a Second Order simulation. The Second Order simulation is closer to the real system than the previous are. To model this system more closely, higher order simulation might be needed.

5.2.1 Zero Order Simulation

The Zero Order simulation is the simplest simulation where the dynamic characteristic of the stepper motor is ignored. This simulation does not take into consideration the response time of the stepper motor. In other words, the stepper motor is assumed to move to the desired position instantaneously when a command pulse is received. The only possible directions of motion are parallel to one axis or diagonal (the diagonal move occurs when both axes have a pulse at the same time).

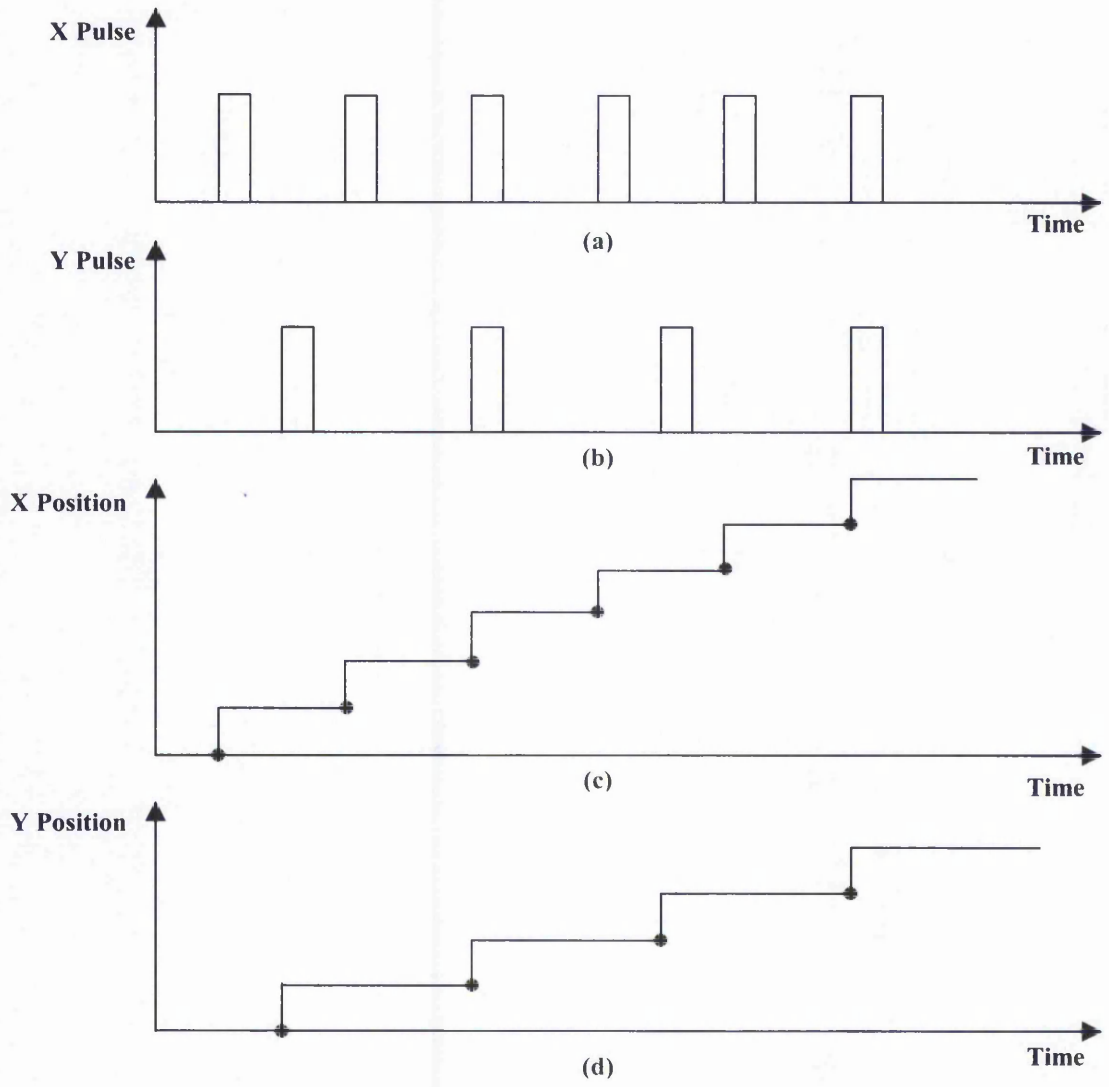


Figure 5-6: Example of Zero Order simulation: (a) X pulses; (b) Y Pulses; (c) Distance In X; (d) Distance in Y. This example is for the new linear interpolation for the line from (0,0) to (6,4).

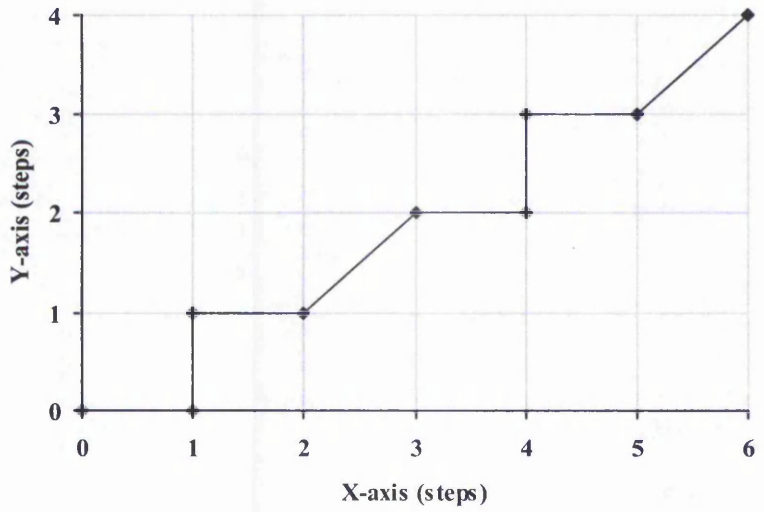


Figure 5-7: Example of Zero Order simulation – of the Path for the Example in Figure 5-6.

Figure 5-6 demonstrates how the Zero Order simulation simulates the motor motion for an individual axis when command pulses are received. It can be seen from the figure that the motor step is moved at the same instant the command pulse is received. The path generated from the previous example is illustrated in Figure 5-7.

5.2.2 Varying Rate First Order Simulation

The Varying Rate First Order simulation is a more realistic simulation compared to the Zero Order simulation. There will always be a response time before the stepper motor settles. In this simulation it is assumed that the motion is linear and the response time is equal to the time between the current command pulse and the next one. In other words, this simulation technique assumes that the stepper motor will only complete moving one stepper motor step when the next command pulse is received and that the motion between pulses varies linearly with time. The response time for the last command pulse is taken to be the same as the penultimate response time.

As in the case of the Zero Order simulation, the same example is shown in Figure 5-8 to illustrate the command pulses sent to the two axes and how these pulses will affect the

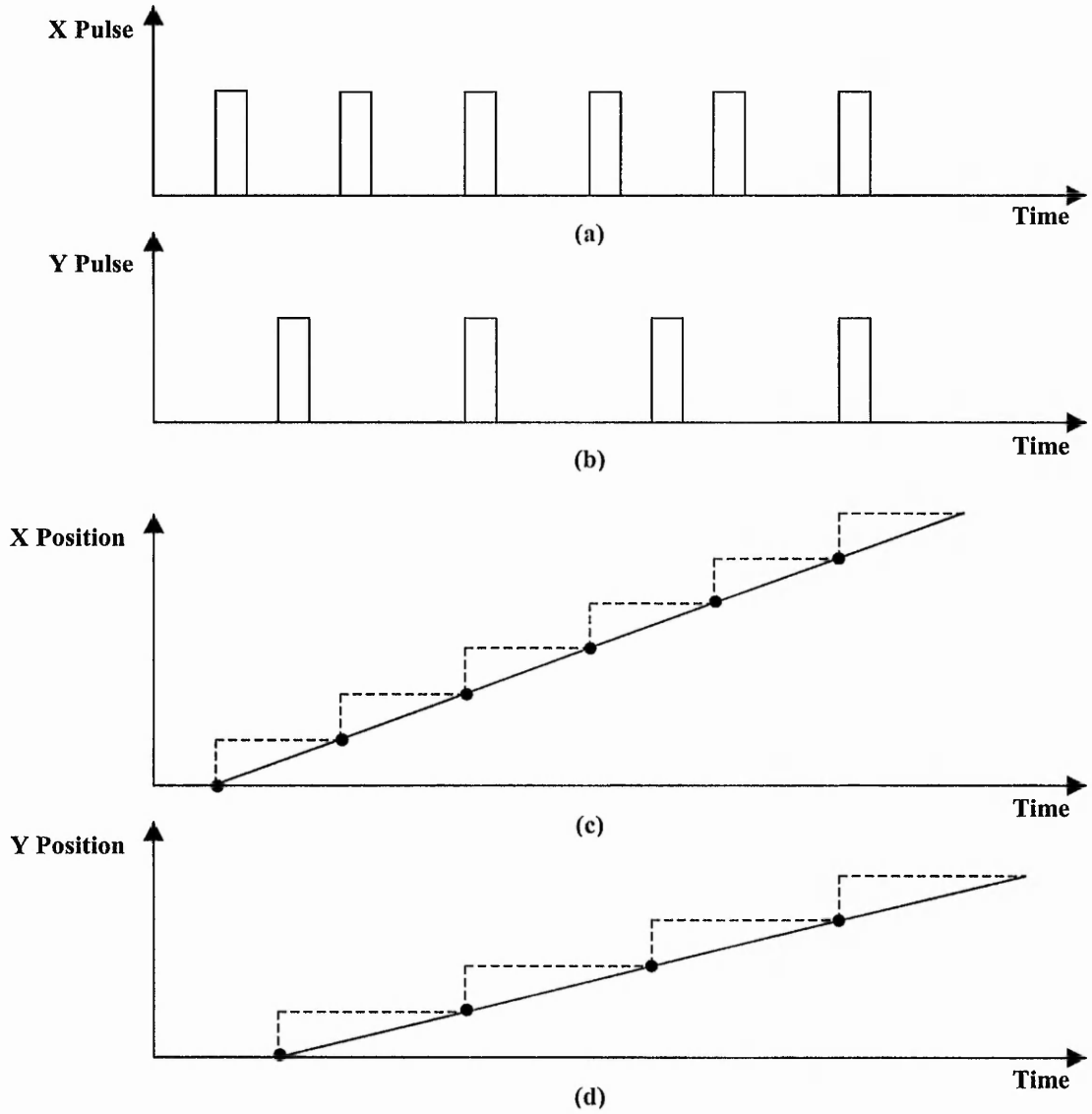


Figure 5-8: Example of Varying Rate First Order simulation: (a) X pulses; (b) Y Pulses; (c) Distance In X; (d) Distance in Y. This example is the same as for Figure 5-6.

axis positions. The X-axis motor motion varies linearly with time throughout the six command pulses because the time between pulses is constant for the X-axis in this example. The same happens to the Y-axis which consists of only four command pulses.

The plots in part (c) and (d) of Figure 5-8, position against time, are used to plot the simulated path. How this is done will be explained in detail in Section 5.2.5. The resultant path from this example is illustrated in Figure 5-9. From this figure, it can be clearly noticed that the path follows a smoother line in most parts of the interpolation.

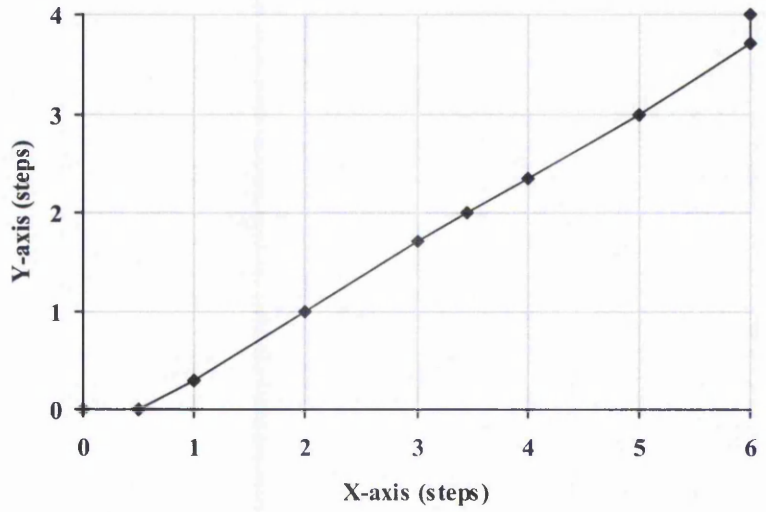


Figure 5-9: Example of Varying Rate First Order simulation – of the Path for the Example in Figure 5-8.

However, this simulation is likely to be less accurate for slow speeds. On the other hand, for higher speed, the Varying Rate First Order simulation may be able to show more closely the actual stepper motion, assuming it is smooth.

5.2.3 Constant Rate First Order Simulation

As discussed in the previous section, Varying Rate First Order simulation is more suitable for high speed interpolation. Therefore, a different simulation, more suitable for low speed interpolation, has been developed. The Constant Rate First Order Simulation assumes that the stepper motor has completed the movement before the next pulse is sent and that the time for the movement is always the same. Therefore an additional parameter is needed, which is the stepper motor response time. (During the response time the motion is assumed to be linear.)

Figure 5-10 has the plot of the motor motion for each individual axis using the Constant Rate First Order simulation. It should be noticed that at the instant of receiving a command pulse, the motor position changes linearly with time. The rate of this linear

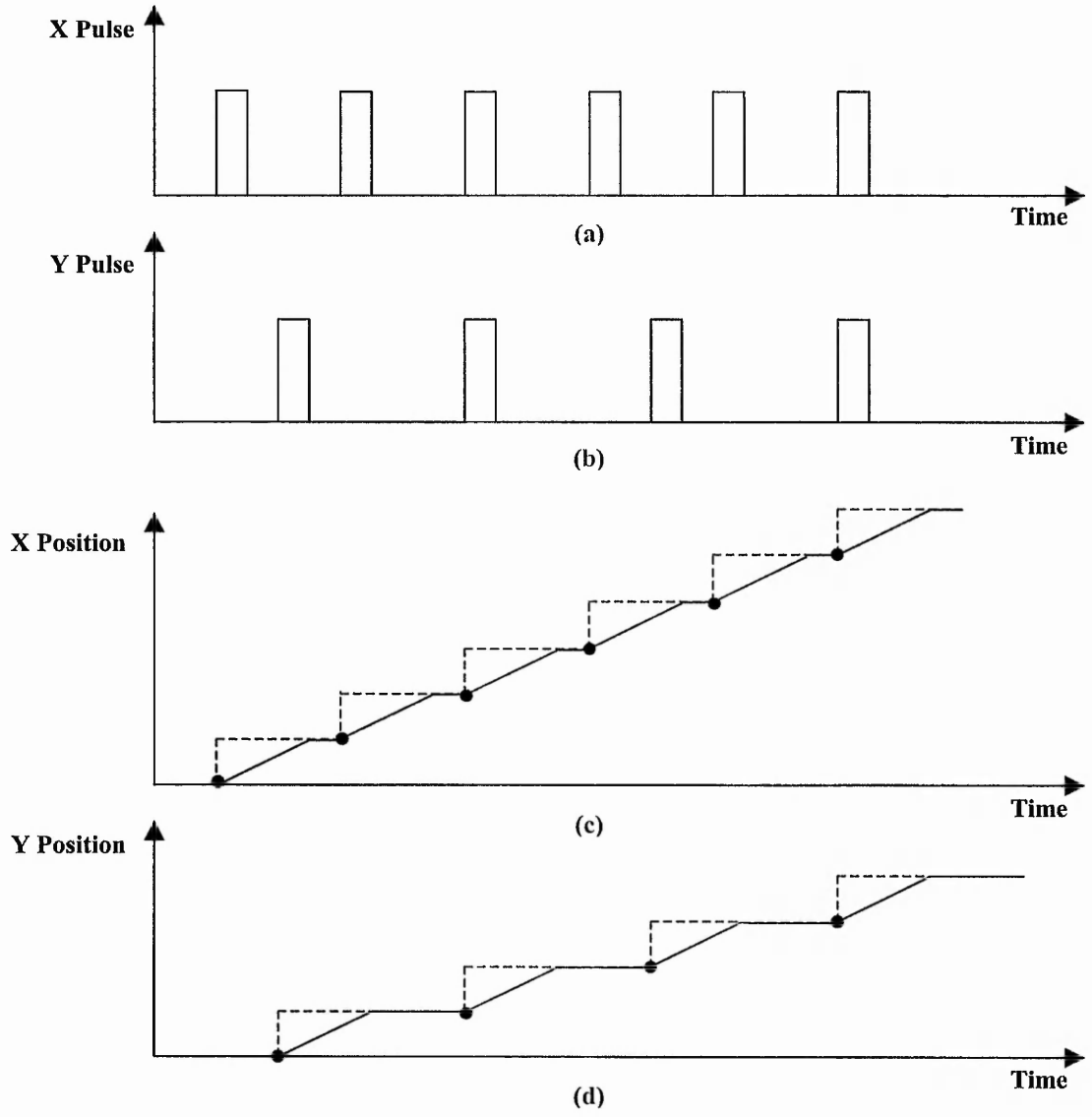


Figure 5-10: Example of Constant Rate First Order simulation: (a) X pulses; (b) Y Pulses; (c) Distance In X; (d) Distance in Y. This example is the same as for Figure 5-6.

motion is constant, in contrast to the Varying Rate First Order simulation where the rate of the linear motion depends on the time between pulses. In other words, the time it takes to settle in the next motor step position is the same for every motor step. This simulation is suitable for low speed interpolation when the motor will normally settle at the next motor step before the next command pulse is received. The plot of the simulated path for the same example is shown in Figure 5-11.

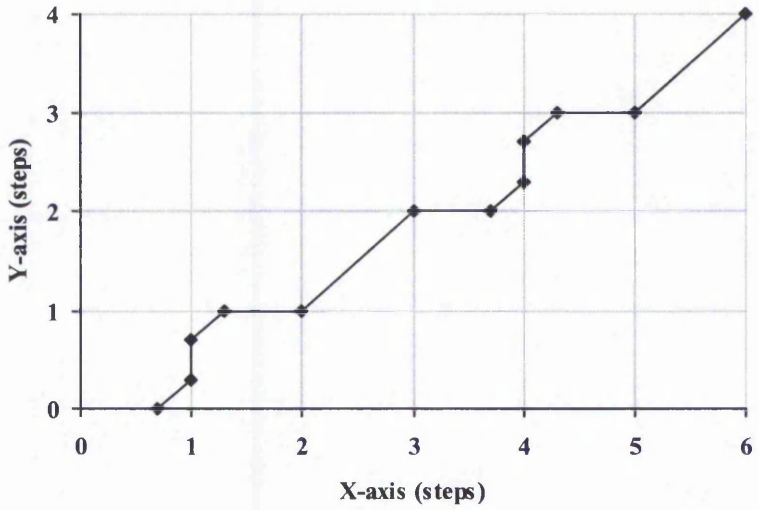


Figure 5-11: Example of Constant Rate First Order simulation – of the Path for the Example in Figure 5-10.

5.2.4 Second Order Simulation

When loss of synchronisation does not occur, the behaviour of a stepper motor control system is similar to that of a mass which is located by a "magnetic spring" as illustrated in Figure 5-12. Therefore, it is useful to use the Second Order simulation, because it is intended to model an ideal mass-spring system. An approximation is used to derive the algorithm used, so it may not model the mass-spring system exactly. In addition the stepper motor may not behave exactly like an ideal mass-spring system even when there

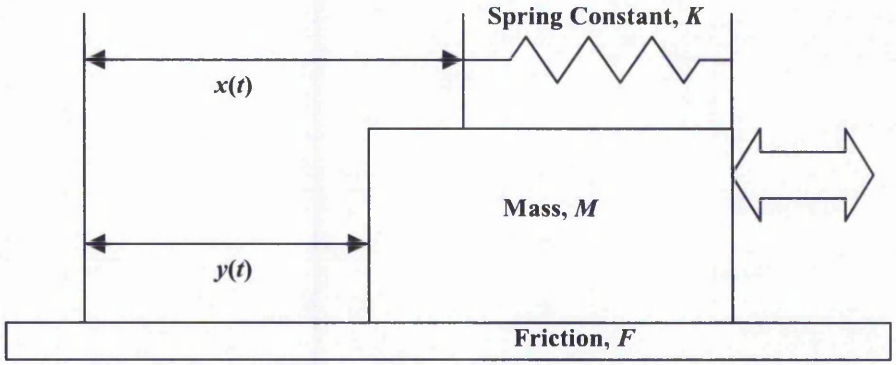


Figure 5-12: Mass-Spring Characteristics (adapted from [81]).

is no loss of synchronisation. Therefore careful testing would be needed before detailed conclusion could be drawn from this simulation alone.

The Second Order simulation is in some ways closer to a real system than the previous ones. The development of this simulation system has been done by Nicolaos [81]. From Figure 5-12, $x(t)$ is the input displacement while $y(t)$ is the output displacement. Analogously, $x(t)$ is the expected step position after every command pulse is received and $y(t)$ is the actual position moved by the stepper motor as a result of the input command pulses. This system can be represented by the following differential equation:

$$y(t) = M\frac{d^2x}{dt^2} + F\frac{dx}{dt} + Kx(t) \tag{5-1}$$

Following the work by Nicolaos [81] the stepper motor system can be represented by the following difference equation:

$$y_n = a_n x_n + a_{n-1}x_{n-1} + a_{n-2}x_{n-2} + b_{n-1}y_{n-1} + b_{n-2}y_{n-2} \tag{5-2}$$

where x_n is the input series and y_n is the output series, and:

$$\begin{aligned} a_0 &= \frac{\omega^2 T^2}{4 + 4\varsigma\omega T + \omega^2 T^2} \\ a_1 &= 2a_0 \\ a_2 &= a_0 \\ b_1 &= -2a_0\left(1 - 4\omega^{-2}T^{-2}\right) \\ b_2 &= -a_0\left(4\omega^{-2}T^{-2} - 4\varsigma\omega^{-1}T^{-1} + 1\right) \end{aligned} \tag{5-3}$$

ω is the natural frequency of the stepper motor while ς is the damping factor and T is the sample period. The recursive difference equation in (5-2) is used to simulate the output of the system for any input series. For all the simulations used in this work, ς has been set to 0.7 and ω is 628 rads/sec, equivalent to 100 Hz, because these represent typical values for a stepper motor. The value of T is chosen to be 0.1 ms so that enough points are available to plot the curve.

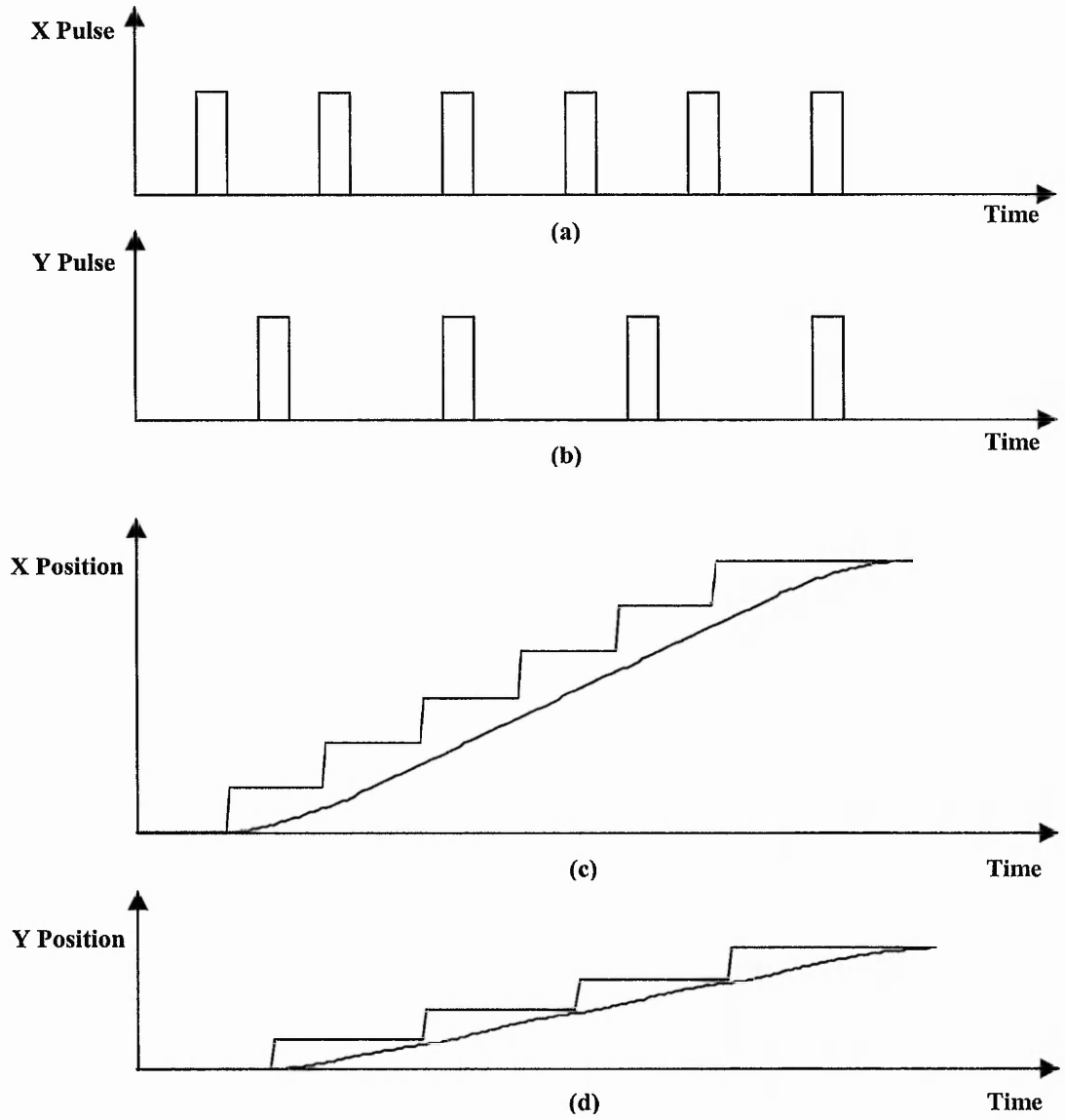


Figure 5-13: Example of Second Order simulation: (a) X pulses; (b) Y Pulses; (c) Distance in X; (d) Distance in Y. This example is the same as for Figure 5-6. (Damping Factor = 0.7; Speed = 500 steps/s).

Figure 5-13 demonstrates the Second Order simulation of the same example as before. This example uses a high-speed interpolation. In this example, the motor motion variations are very close to the Varying Linear First Order simulation (Figure 5-8). The simulated path is plotted in Figure 5-14. This plot demonstrates a smoother line compared to the Zero Order or the Constant Rate First Order simulations. The plot looks similar to the plot for Varying Linear First Order simulation in Figure 5-9.

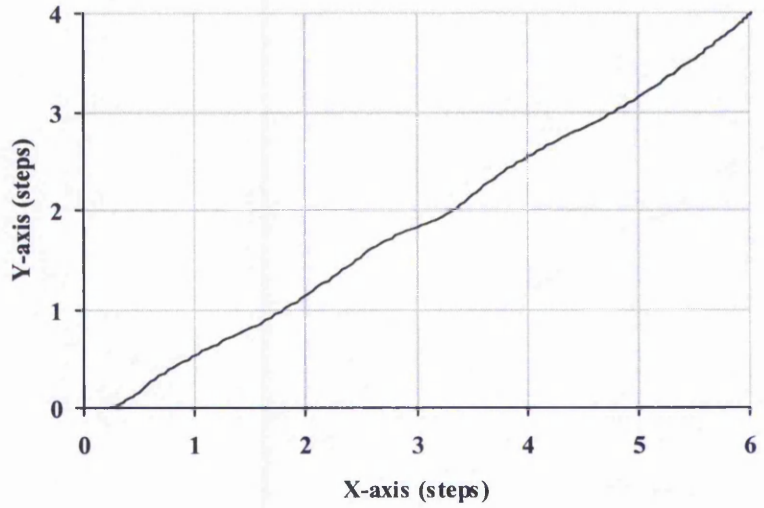


Figure 5-14: Example of Second Order simulation – of the Path for the Example in Figure 5-13

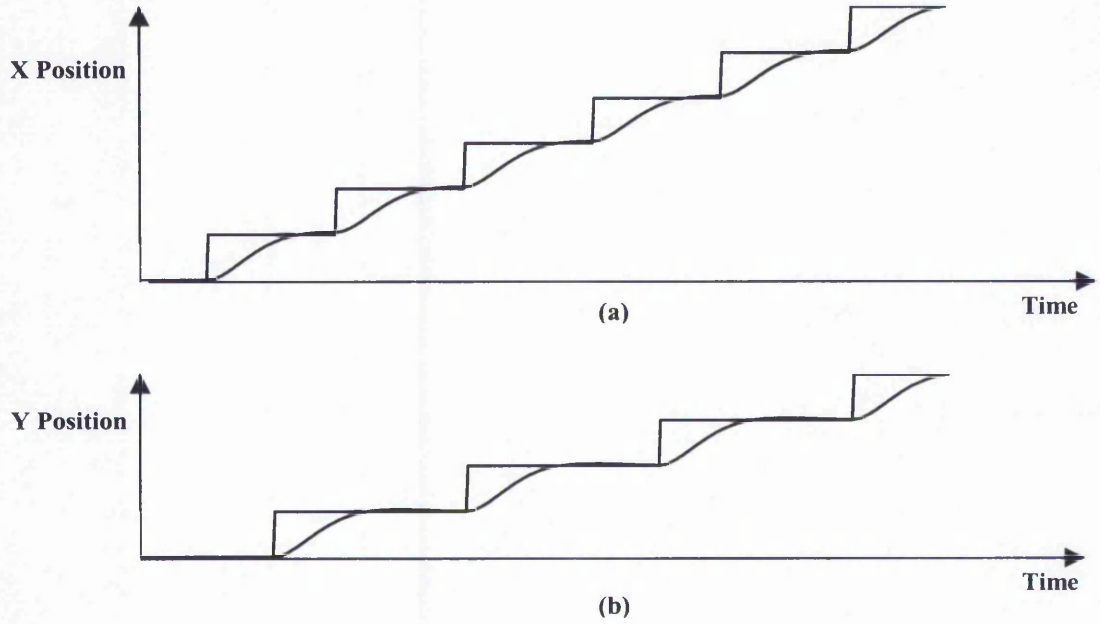


Figure 5-15: Example of Second Order Simulation (Low Speed): (a) Distance In X; (b) Distance In Y. This example is the same as for Figure 5-6, apart from the speed. (Damping Factor = 0.7; Speed = 167 steps/s).

Figure 5-15 uses the same parameters as in the example shown in Figure 5-13 but with low speed. The speed for this example is a third of the speed in the high-speed example. The motor motion variations from this low-speed Second Order simulation are very

similar to the plots in Figure 5-10. The plot of the simulated path is shown in Figure 5-16. It is quite similar to Figure 5-11 (Constant Rate First Order simulation) or even Figure 5-7 (Zero Order simulation).

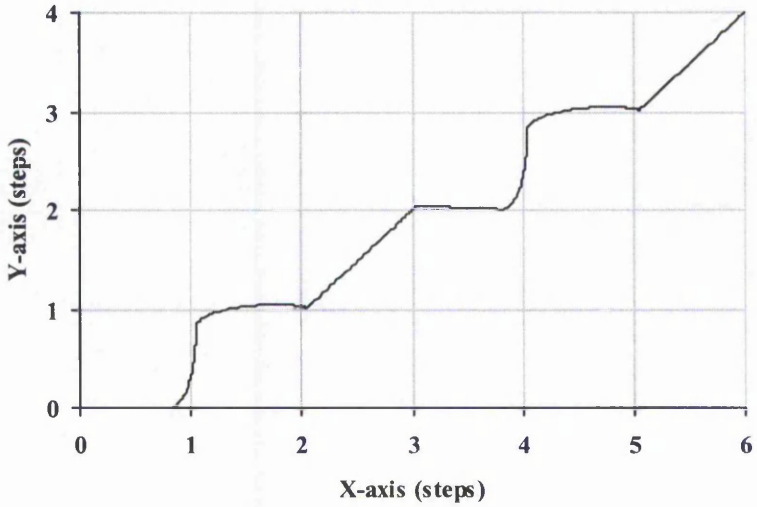


Figure 5-16: Example of Second Order simulation (Low Speed) – of the Path for the Example in Figure 5-15. It looks very similar to Constant Rate First Order simulation in Figure 5-11.

5.2.5 Calculation of the Simulated Path

For each method, the simulation results for both the X and Y-axis are combined to generate the simulated cutting path. Figure 5-17 to Figure 5-19 illustrate the results of the three Zero and First Order simulations (for the same inputs) for both axes and how an interpolated path is calculated from this information. The purpose of developing different simulation technique is to model more closely the expected result when used in a multi-axis stepper motor control system where the motor does not respond instantly to command pulses. The black circle in the individual axis motion indicates the point where one or both of the two axes changes speed according to the simulation. For each of these points (A,B,C etc), a point will be plotted on the simulated path plot. For instance, point A on the X-Y plane has the X-axis position at A_x and the Y-axis position

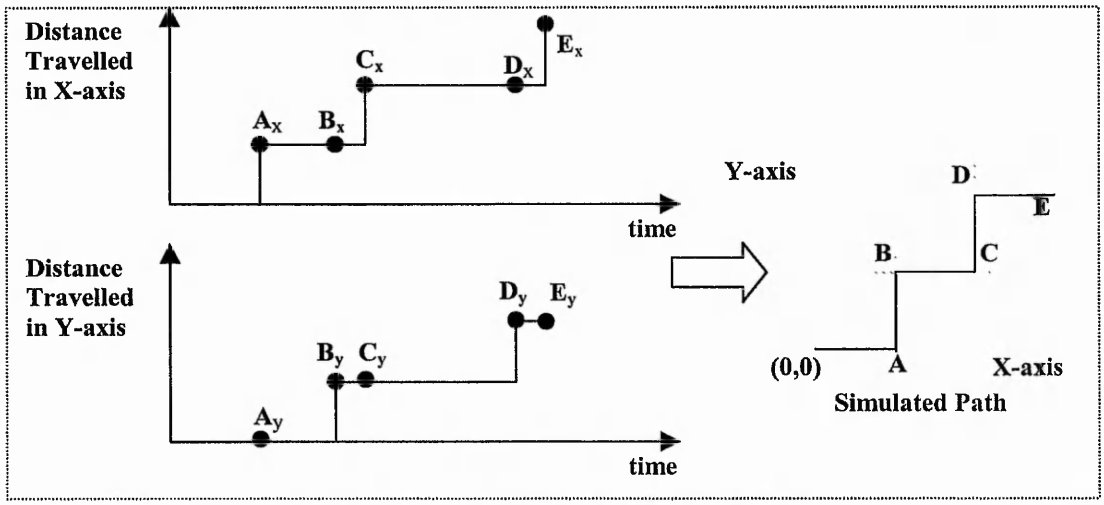


Figure 5-17: Zero Order Simulation.

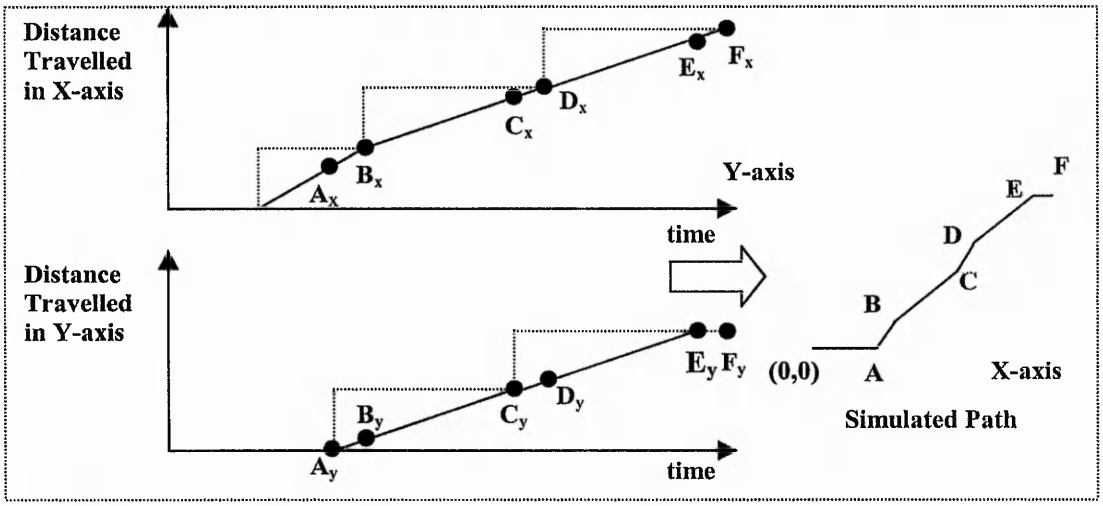


Figure 5-18: Varying Rate First Order Simulation for same input as in Figure 5-17.

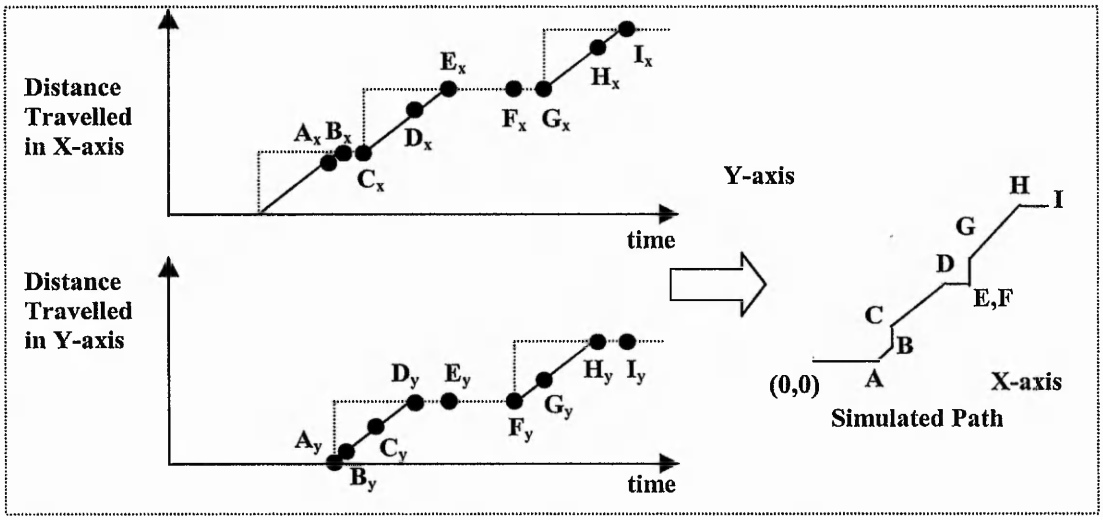


Figure 5-19: Constant Rate First Order Simulation for same input as in Figure 5-17.

at A_y. Since both axes change linearly between each pair of spots, the simulated path will be linear between them.

Another possible method to plot the simulated path is to have the position data calculated at very small time intervals. This will indeed generate much more data and can be avoided for Zero and First Order simulations. However, the Second Order simulation needs to use this approach because the motion is not linear. It uses the position data at small fixed time intervals to plot the simulated path.

5.2.6 Comparison of Path Generated by Different Simulation Algorithms

In general, the Varying Rate First Order simulation gives the smoothest simulation but is not realistic when pulse rate is low. On the other hand, the Zero Order and the Constant Rate First Order simulation are both jerky. The Second Order simulation looks more realistic compared to the rest. It exhibits the good characteristics of both the Varying Rate and Constant Rate First Order simulations. Figure 5-20 below illustrates how the Zero and First Order simulation discussed here will simulate the position of the stepper motor after three command pulses.

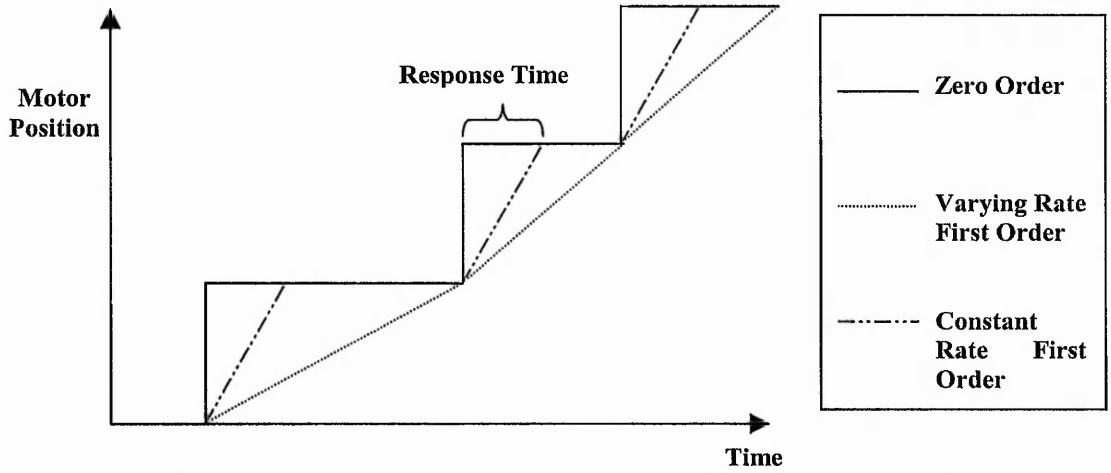


Figure 5-20: Step Response Using Zero and First Order Simulation.

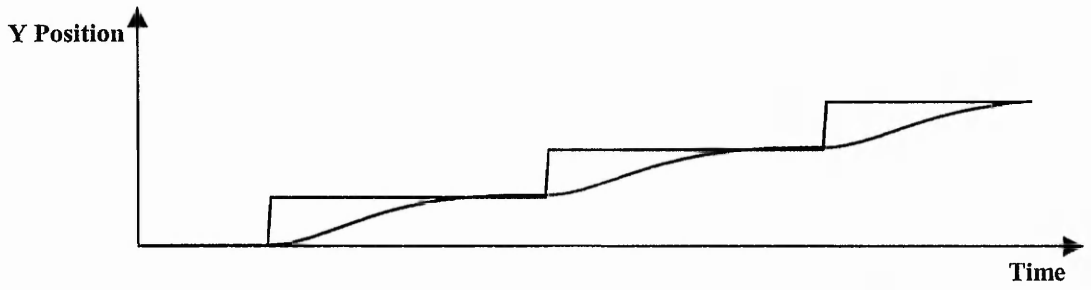


Figure 5-21: Step Response Using Second Order Simulation (Low Speed).

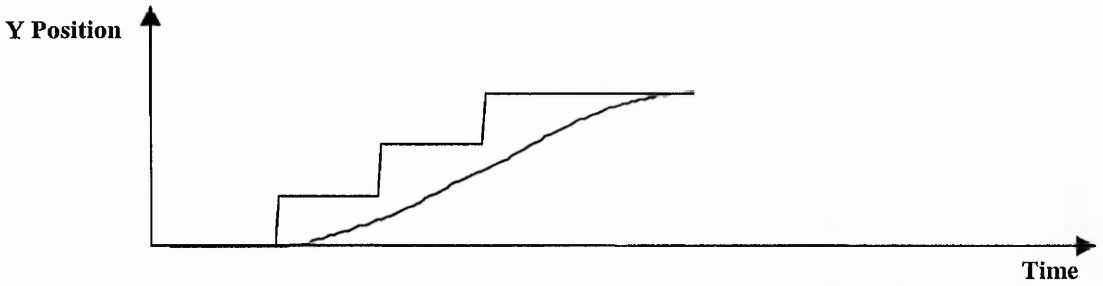


Figure 5-22: Step Response Using Second Order Simulation (High Speed).

The Second Order simulation is illustrated in Figure 5-21 and is seen to be very close to the Constant Rate First Order simulation for low speed. On the other hand, it is close to the Varying Rate First Order simulation for high speed as shown in Figure 5-22. All these simulations are useful to get an idea of how the motor will actually respond to the command pulses. Both the First and Second Order simulations are estimates of the real motion with the Second Order simulation may be a closer match to the actual response of a real motor.

5.3 Simulation of Speed

Speed simulation is used to simulate the speed of the individual axis. The technique employed here is to get the average speed between adjacent pulses. In other words, this technique assumes the response of the stepper motor to command pulses matches the response suggested using the Varying Rate First Order simulation. In reality, there may be some smoothing of the speed or it could be worse with vibrations. The following equation is used to calculate the speed between adjacent generated command pulses:

$$speed = \frac{step\_size}{timeOfPulse(n) - timeOfPulse(n-1)} \tag{5-4}$$

Using the path shown in Figure 5-18, a speed simulation can be plotted as shown in Figure 5-23.

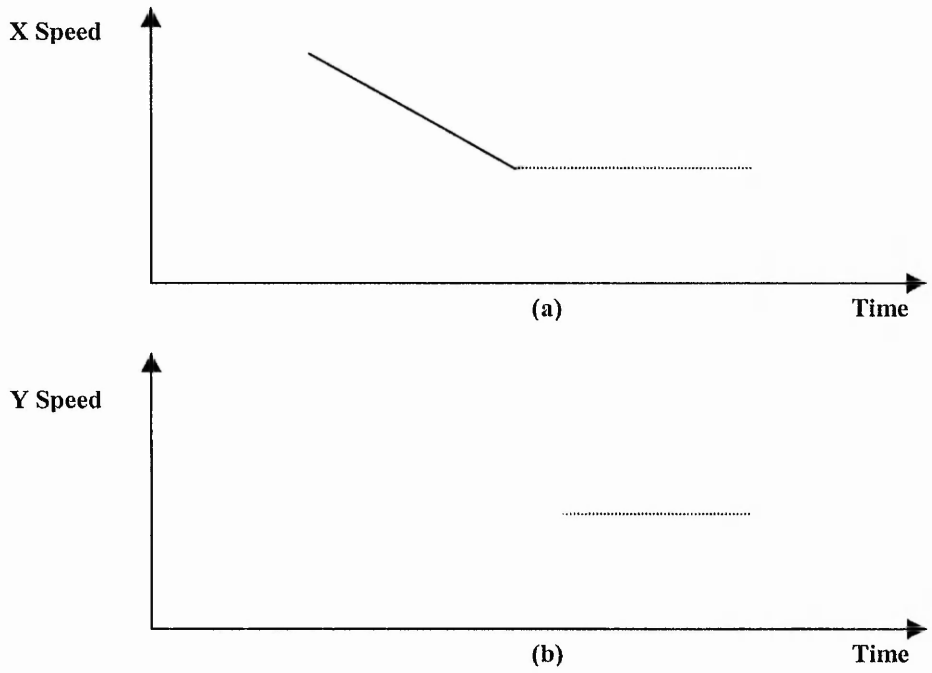
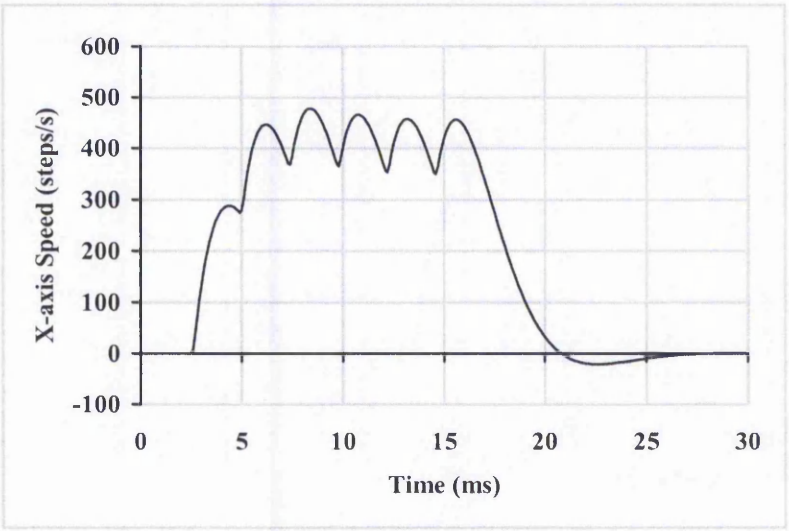
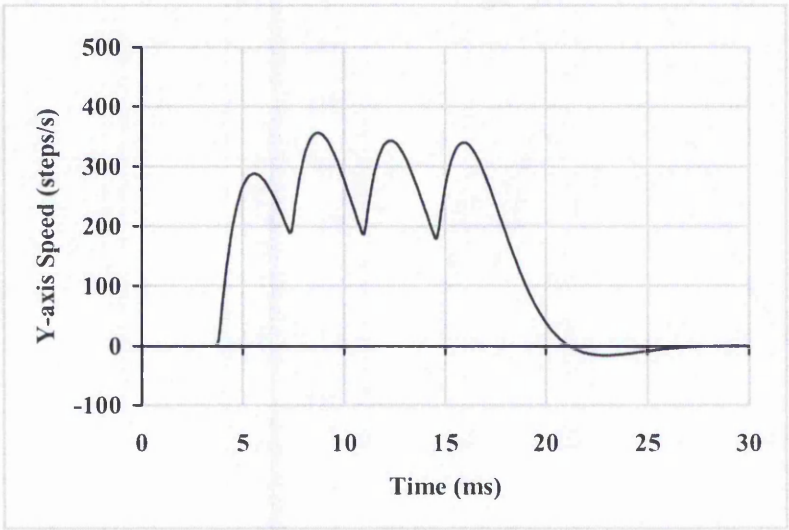


Figure 5-23: Example of Speed Simulation for the Example in Figure 5-18. (a) X-axis Speed; (b) Y-axis Speed, assuming that the pulse following continues at the same rate for each axis (when it reaches F on X-axis, E on Y-axis).

The speed simulation could make use of the Second Order simulated positions. However, this will present a rather oscillatory speed simulation in all cases, even when the pulses are at constant rate. In Figure 5-24, for high speed machining, the oscillation is not as bad as the low speed machining in Figure 5-25. From these Second Order simulations, it can be expected that, even when the pulse frequency is constant, there will be some variations in speed. These are unavoidable. What is more useful for this work is to identify variations in pulse frequency. Therefore, only the Varying Rate simulation has been considered for speed simulation for the evaluation.

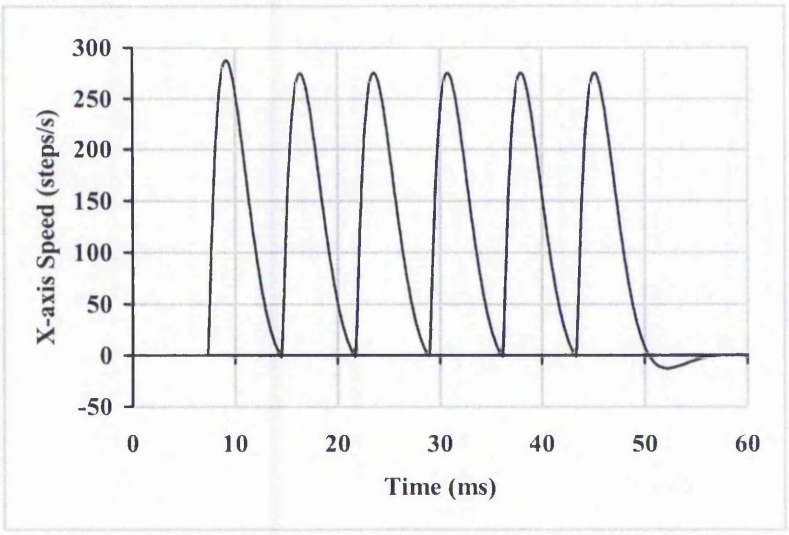


(a)

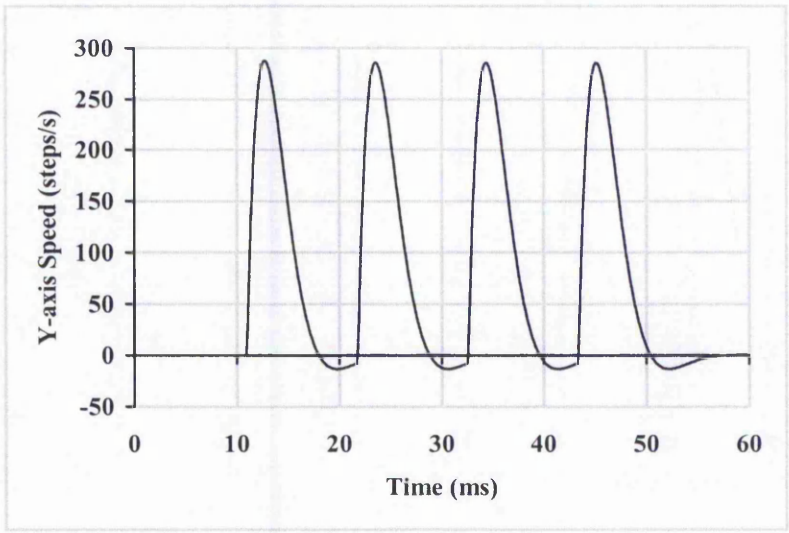


(b)

Figure 5-24: Second Order Speed for Example in Figure 5-13 (High Speed): (a) X Speed; (b) Y Speed.



(a)



(b)

Figure 5-25: Second Order Speed for Example in Figure 5-15 (Low Speed): (a) X Speed; (b) Y Speed.

6 Evaluation of The New Algorithms

Chapters 3 and 4 have introduced the new interpolation and acceleration algorithms and have included some simulation examples. It is important to compare the results from the new algorithms with previous algorithms. This chapter presents simulation results and an evaluation of the effectiveness of the algorithms developed. The two new interpolation algorithms are the Full Step and Half Step. The latter has been developed as an improvement of the former. In Section 6.2 both will be evaluated against previous interpolation algorithms, the Search-Step, Direct-Search and DDA algorithms. These algorithms have been explained in Section 2.3 and the software version of DDA has been used here.

The new acceleration algorithms have also been evaluated for their performance when used with the new interpolation algorithms. In fact, some of these simulations have been discussed in Section 4.4. Section 6.3 includes more simulation results for the new linear and parabolic acceleration algorithms. It is not possible to compare directly with earlier acceleration algorithms, because the acceleration algorithms have been designed for use with new interpolation algorithms. A protocol controller has been developed and simple tests have been performed using the same line and arc used for the simulation tests for acceleration. The results are presented in Section 6.4.

For the simulation results, two main criteria are chosen for evaluation:

- Position errors, and
- Speed variations (X and Y axis speed fluctuations)

The largest position error is calculated to give an idea of how accurate the required path is followed. On the other hand, the average error value is taken of the signed error to investigate how much the simulated path deviates on average from the required path. Since the effect of each command pulse is difficult to predict and depends on the motor used, four simulation methods have been used (as explained in Chapter 5).

A short line and small radius arc have been used for evaluation of interpolation. It is done at a relatively low speed because acceleration is not used. For acceleration algorithms, a longer line and larger radius arc are used so that there is time to accelerate up to the same speed.

Section 6.1 of this chapter explains the method for the calculation of position errors for both lines and arcs. The calculation for the speed simulation is also included in this section. The following section, Section 6.2, presents simulation results from the five different interpolation algorithms. The path is simulated using four different simulation methods, namely the Zero Order, Varying Rate First Order, Constant Rate First Order and Second Order simulation presented in Section 5.2. Section 6.3 concentrates on the simulation results for the linear and parabolic acceleration algorithms. The initial results from the practical implementation are presented in Section 6.4.

6.1 Methods Used for Evaluation

Section 6.1.1 describes how the position errors have been calculated for both lines and circular arcs. Section 6.1.2 explains how the axis speeds have been calculated from the list of pulse timings.

6.1.1 Calculation of Position Errors

Evaluation of position errors involves measuring how accurately the interpolated path can follow the desired path. For both linear and circular arc interpolation, the deviation of the simulated path is taken to be the shortest distance of each point on the simulated path from the desired line or arc. The method used in each case allows the shortest distance to be given a sign to indicate on which side of the line or arc it lies. This is useful for the error plots, because it gives greater information about the errors.

The shortest distance from a straight line is the perpendicular distance of the point from the line. Figure 6-1 illustrates how this distance is calculated. If the interpolated point is above the required path, the position error is assigned a positive value, while it is assigned a negative value if the interpolated point is below the required path. Thus the sign of the error value indicates which side of the line the point lies.

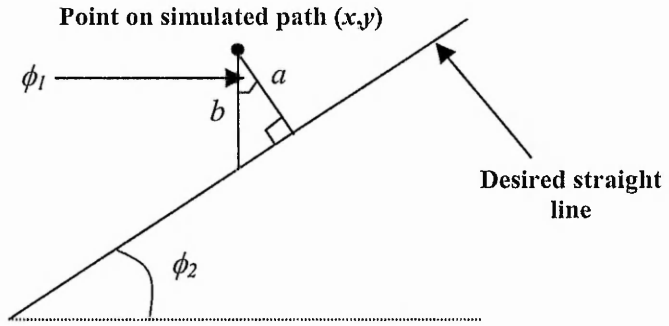


Figure 6-1: Calculation of Position Error for Line.

It can be shown that $\phi_1 = \phi_2$ and ϕ_2 can be calculated from the end point information as follows:

$$\phi_1 = \tan^{-1}\left(\frac{y\_end - y\_start}{x\_end - x\_start}\right) \tag{6-1}$$

Now, distance b is calculated as in (6-2).

$$b = y - \left(\frac{y\_end - y\_start}{x\_end - x\_start}\right)(x - x\_start) - y\_start \tag{6-2}$$

Using ϕ_1 and b from equations (6-1) and (6-2) the positional error is expressed as:

$$a = b \cos(\phi_1) \tag{6-3}$$

This gives positive and negative values, as required. In the case when $x\_end = x\_start$ (the line is parallel to the Y-axis), the error is calculated using the X-axis coordinate of the point and the line. The error is taken as positive to the right of the line and negative to the left.

For a circular arc, to find the shortest distance of a point from the arc, the closest point on the arc needs to be found. The closest point lies on the line from the point to the circle centre. Therefore the error can be calculated as the difference between the distance from the centre and the radius of the circle. The position error is calculated as in equation (6-4). A simulated point lying outside the circle will result in a positive error while for a point lying inside it will be negative.

$$\text{position error} = \sqrt{(x - x_{centre})^2 + (y - y_{centre})^2} - radius \tag{6-4}$$

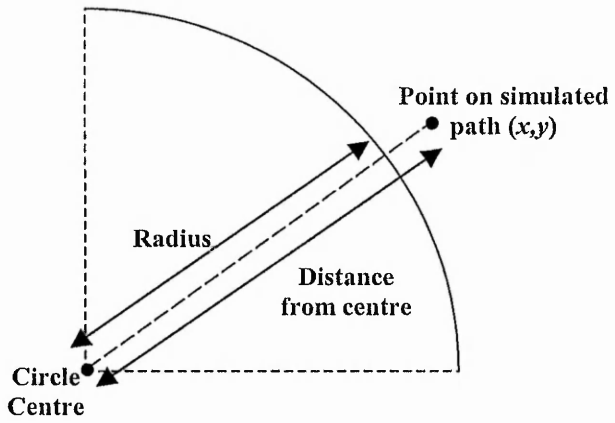


Figure 6-2: Calculation of Position Error for Circular Arc.

6.1.2 Calculation of Speed

Speed simulation is used to estimate the speed along the individual axis. The technique employed here is to find the time between adjacent pulses and use it to calculate the speed between the two pulses. Then this value is assigned to the second of the two pulses although it could equally well have been assigned to the first. In other words, this technique assumes the response of the stepper motor to command pulses matches the response using the Varying Rate First Order simulation. In reality, there may be some smoothing of the speed or the situation could be worse with large vibrations caused by sudden changes in speed. The following equation is used to calculate the speed between adjacent steps:

$$speed(n) = \frac{step\_size}{timeOfPulse(n) - timeOfPulse(n-1)} \qquad (6\text{-}5)$$

Because the speed is assigned to the second of the two pulses, there will be no speed value for the initial pulse.

6.2 Evaluation of the Interpolation Algorithms by Simulation

The evaluation of interpolation presented in this section does not consider acceleration and deceleration. The evaluation of the acceleration and deceleration algorithms is presented in Section 6.3. Therefore, only low speeds are used here. Five interpolation algorithms are discussed here. They are the three previous ones, Search-Step, Direct-Search, Digital Differential Analyser (DDA), and then the New Full Step algorithm and the New Half Step algorithm. To determine how closely the simulated path actually follows the required line or arc, the position error graphs have been plotted and are included in Appendix B. The position errors and speed variations are compared for the different interpolation algorithms. The results are summarised in Table 6-1 to Table 6-3 in Section 6.2.3.

For evaluation purposes, an example line having the following parameters have been used:

Start coordinates (steps): (0,0)
End coordinates (steps) : (30,20)

On the other hand, the circular arc example uses the following parameters:

Start coordinates (steps) : (20,0)
End coordinates (steps) : (0,20)
Centre coordinates (steps): (0,0)
Direction: Anticlockwise

A feedrate of 167 steps/s is used for the Constant Rate First Order simulation while the feedrates used for Second Order simulation are 167 and 500 steps/s. Two speeds are used to show the difference between very slow speed and a slightly higher speed. Only the plots for 500 steps/s are presented in this section for the Second Order Simulation. The largest errors and average errors for 167 steps/s are summarised in Table 6-2. The average errors use the signed error values, so that it can be seen whether the simulated path is centre on the required path. Typically with step size 0.01 mm, that means the required speed is 1.67 mm/s and 5 mm/s (0.1 m/min and 0.3 m/min).

6.2.1 Simulation Results for Linear Interpolation

Figure 6-3 to Figure 6-19 show the plots of the paths for the five interpolation algorithms using each of the simulation methods in turn. The corresponding position error plots are in Appendix B. The Zero Order simulated paths are shown in Figure 6-3 to Figure 6-7. For this example, the New Half Step algorithm coincides with the Search-Step algorithm. It can be seen that all the algorithms are able to follow the required line fairly closely. The Direct-Search interpolation appears to be closer to the required line than any of the other algorithms. This is because it allows diagonal both movements on both axes, whereas, in particular, the Search-Step allows a movement on only one axis at a time. The Search-Step interpolation works by looking at the position errors. If it has a positive error, it will try to move towards the negative error in the following command step.

With the Zero Order simulation, the largest position error is 0.55 for all interpolations, except the Direct-Search algorithm, which has 0.28. Because the path from the Direct-Search algorithm has many diagonal segments, it appears to represent the line more closely.

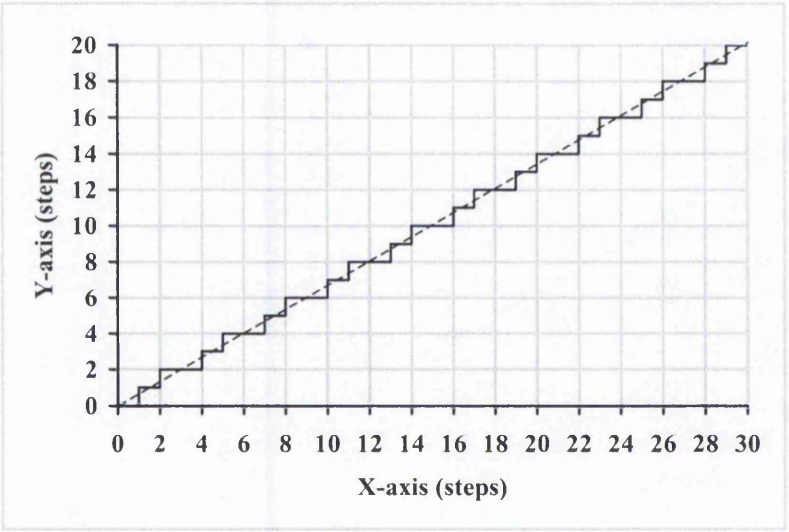


Figure 6-3: Plot of Search-Step Linear Interpolation (Zero and Constant Rate First Order Simulation). Largest Error = 0.55.

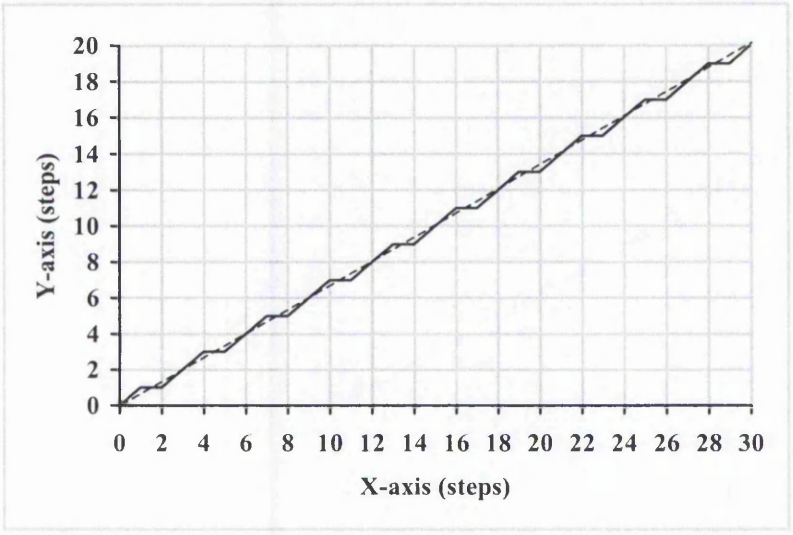


Figure 6-4: Plot of Direct-Search Linear Interpolation (Zero and Constant Rate First Order Simulation). Largest Error = 0.28.

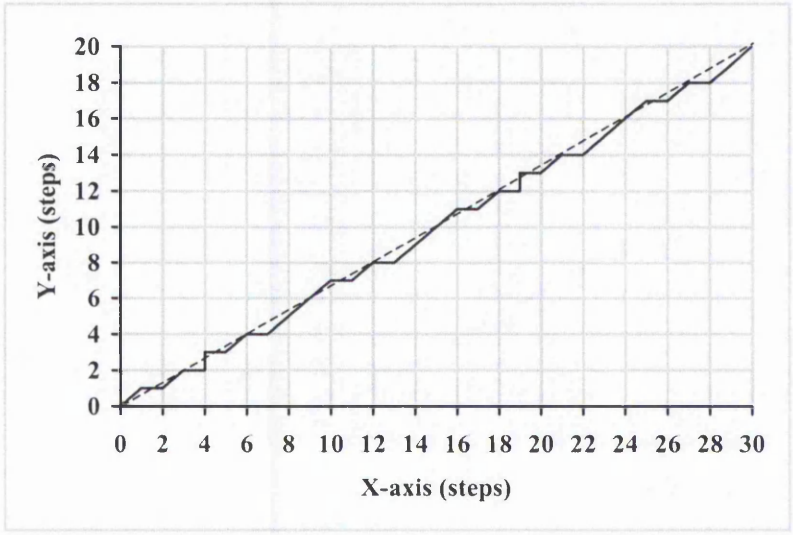


Figure 6-5: Plot of DDA Linear Interpolation (Zero and Constant Rate First Order Simulation). Largest Error = 0.55.

For the New Half Step algorithm, in this particular case, there are also no diagonal line segments. A diagonal line segment is caused when pulses in both the X and Y axes arrive at the same instant. However, the lack of diagonal lines does not necessary mean that the New Half Step algorithm will give larger errors in practice. The reason is because the time gap between the X and Y pulses can be very short but it will still produce a "step" and not a diagonal line with the Zero Order simulation. This is true

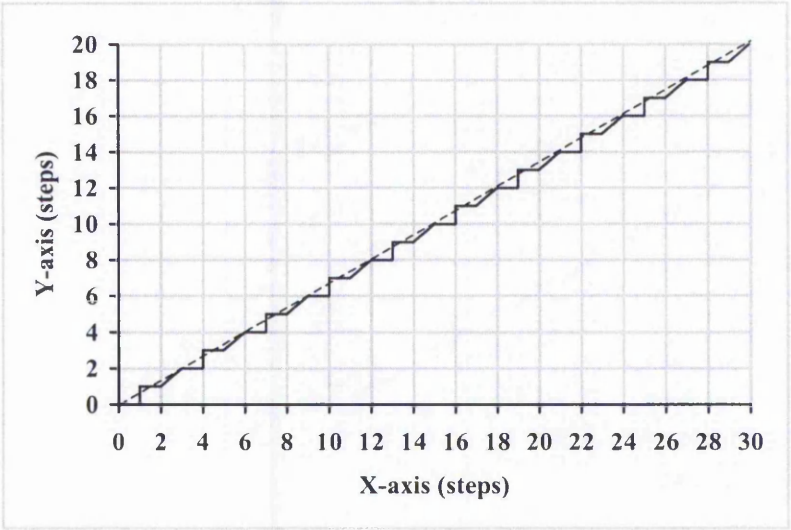


Figure 6-6: Plot of New Full Step Linear Interpolation (Zero Order Simulation). Largest Error = 0.55.

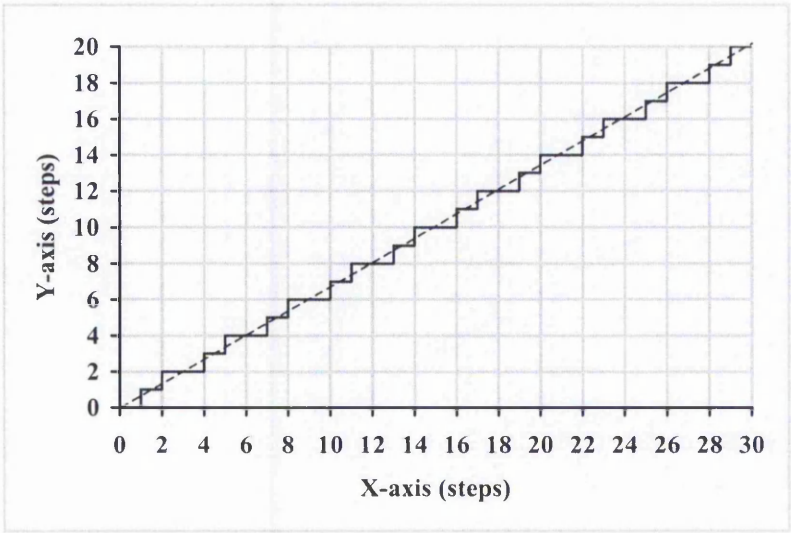


Figure 6-7: Plot of New Half Step Linear Interpolation (Zero Order Simulation). Largest Error = 0.55.

especially for the timing-based interpolation, because the pulses are not generated at fixed time intervals. Zero Order simulation is still useful because it gives a rough idea of the path and it shows the order in which the pulses are sent. However, it is the furthest from actual motion because it assumes each pulse produces instantaneous movement of one step. Therefore the detail of the path cannot be relied on.

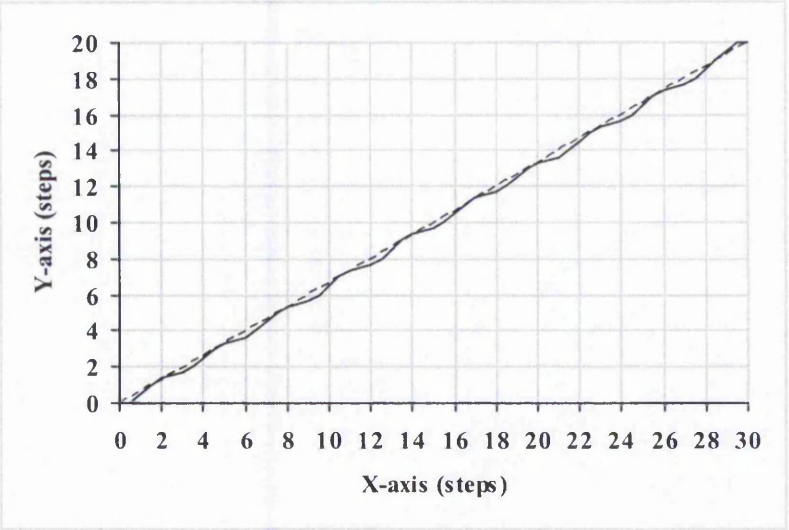


Figure 6-8: Plot of Search-Step Linear Interpolation (Varying Rate First Order Simulation). Largest Error = 0.28.

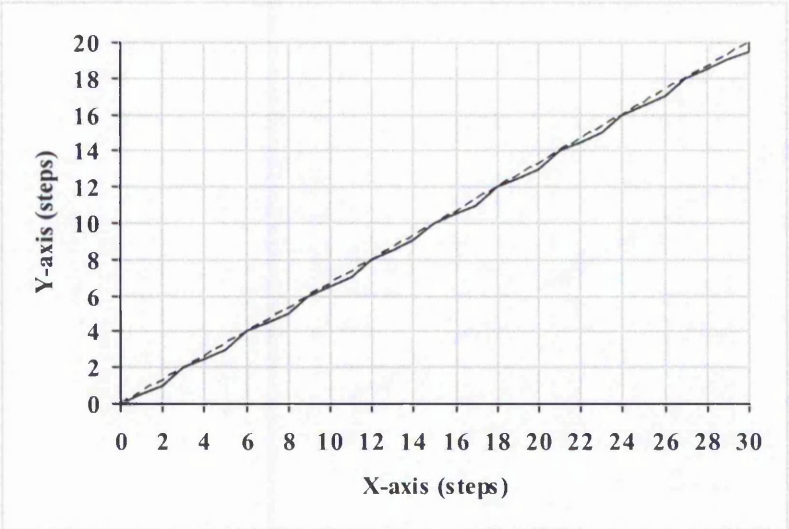


Figure 6-9: Plot of Direct-Search Linear Interpolation (Varying Rate First Order Simulation). Largest Error = 0.42.

The average error for the Search-Step, Direct-Search and New Half Step interpolation is zero, while for the DDA and New Full Step it has a negative value. This means that at most of the interpolation time, the generated path appears to be mostly below the required line for both DDA and the New Full Step interpolations. For the Zero Order simulation, the detail cannot be relied on, so the largest position error values are not very useful in practice.

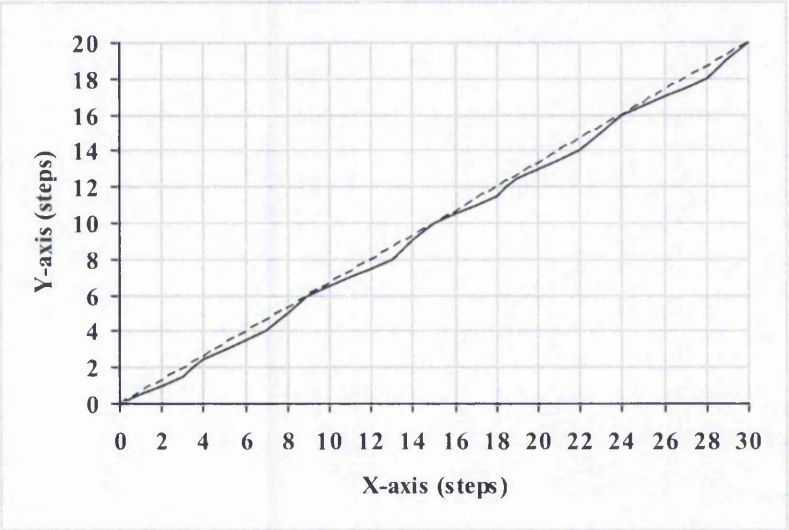


Figure 6-10: Plot of DDA Linear Interpolation (Varying Rate First Order Simulation). Largest Error = 0.55.

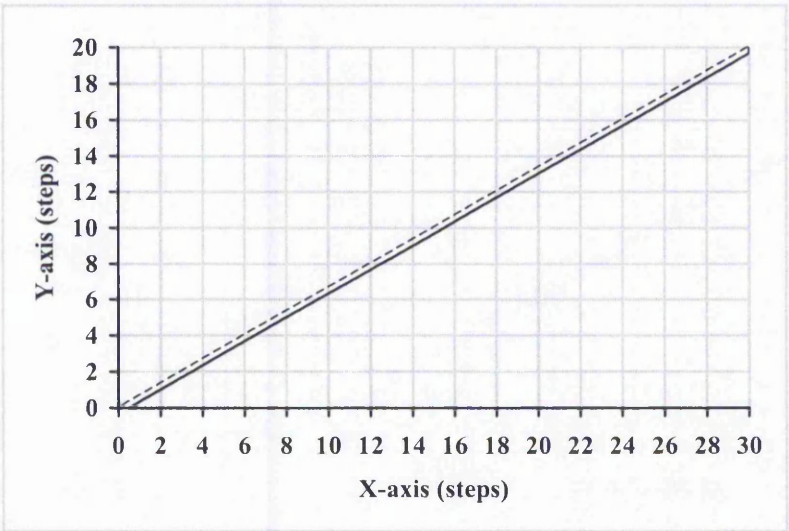


Figure 6-11: Plot of New Full Step Linear Interpolation (Varying Rate First Order Simulation). Largest Error = 0.28.

The higher order simulations produce more useful largest position error estimates. Figure 6-8 to Figure 6-12 show the same linear interpolation simulated using the Varying Rate First Order simulation. This simulation generates a smoother path than the Zero Order, and the New Half Step Interpolation is found to be closest to the required line. This simulation is expected to be closer to the real motion for higher speeds. The New Half Step interpolation has largest position error of 0.14, which is smaller than all the others.

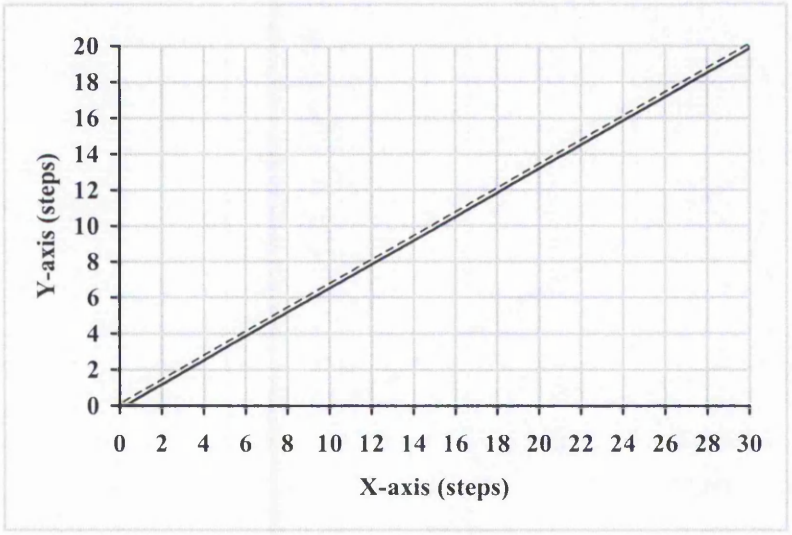


Figure 6-12: Plot of New Half Step Linear Interpolation (Varying Rate First Order Simulation). Largest Error = 0.14.

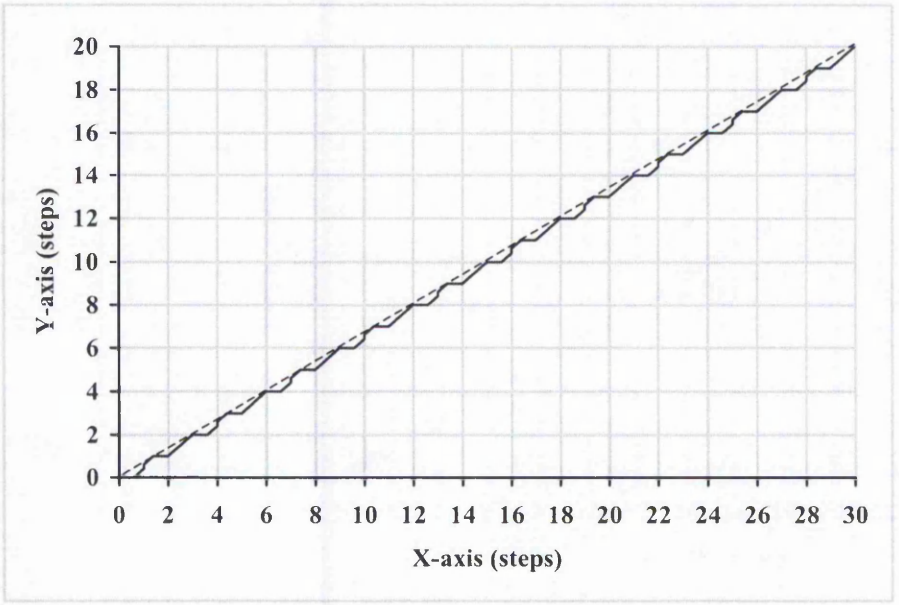


Figure 6-13: Plot of New Full Step Linear Interpolation (Constant Rate First Order Simulation). Response Time = 6ms. Largest Error = 0.33.

The Constant Rate First Order simulated path for both the new linear interpolation is shown in Figure 6-13 and Figure 6-14. The simulated paths using this simulation method for each of the other three interpolation algorithms is identical to the ones for the Zero Order simulation and are shown in Figure 6-3 to Figure 6-5. This simulation is

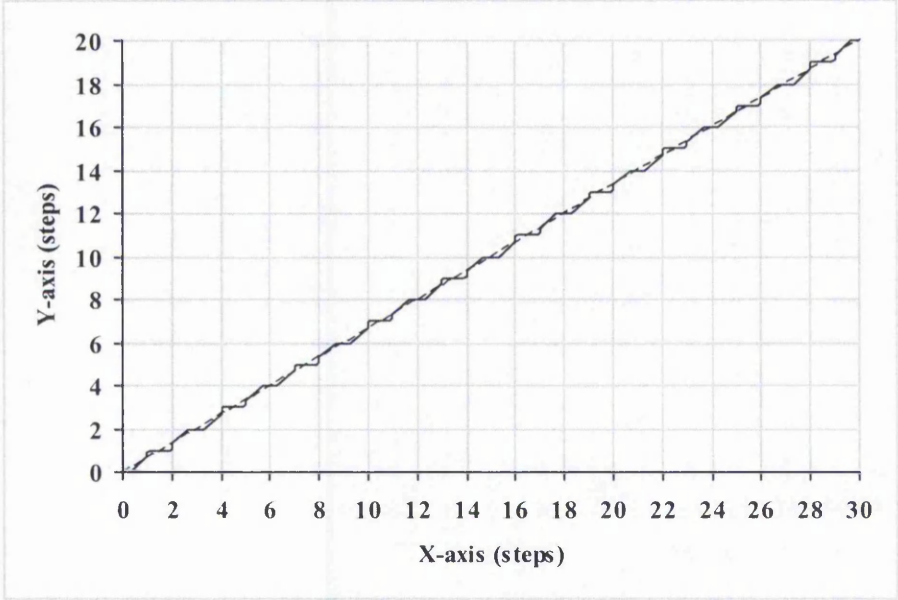


Figure 6-14: Plot of New Half Step Linear Interpolation (Constant Rate First Order Simulation). Response Time = 6ms. Largest Error = 0.22.

expected to be more realistic for lower speeds. If the response time coincides with the time interval between two command pulses, the simulated path will be same as for the Varying Rate First Order simulation. Again the new half step has largest position of 0.22, which is smaller than the others.

The Second Order simulation is expected to produce results that may be closer to the actual machining for a particular machine. Figure 6-15 to Figure 6-19 show the simulated path for the different interpolation algorithms. It can be noticed that the New Half Step Interpolation algorithm produce a path that is very close to the desired line. The Second Order simulation tends to produce a path that is closer to the Varying Rate First Order simulation for higher speeds but the simulated path tends to resemble the Constant Rate First Order at lower speeds, as explained in detail in Section 5.2.4. Lower speeds result in higher fluctuations in speed and whereas higher speeds cause smaller fluctuations in speed.

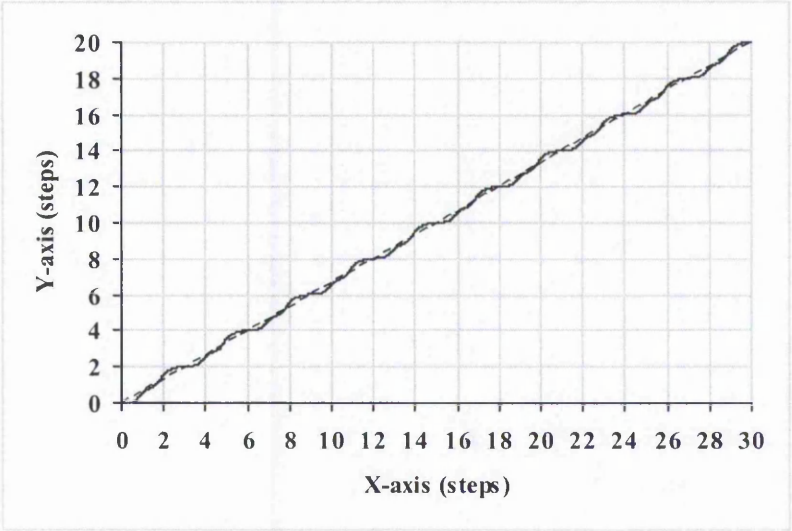


Figure 6-15: Plot of Search-Step Linear Interpolation (Second Order Simulation). The damping factor is 0.7. Largest Error = 0.28.

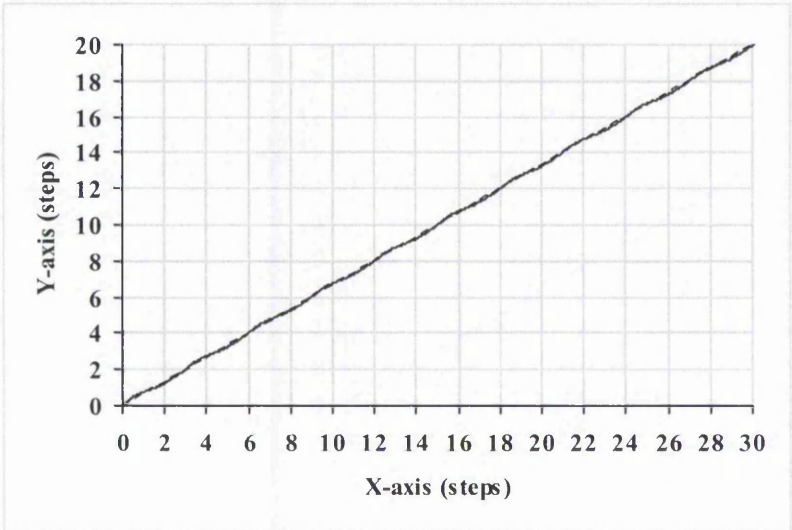


Figure 6-16: Plot of Direct-Search Linear Interpolation (Second Order Simulation). The damping factor is 0.7. Largest Error = 0.13.

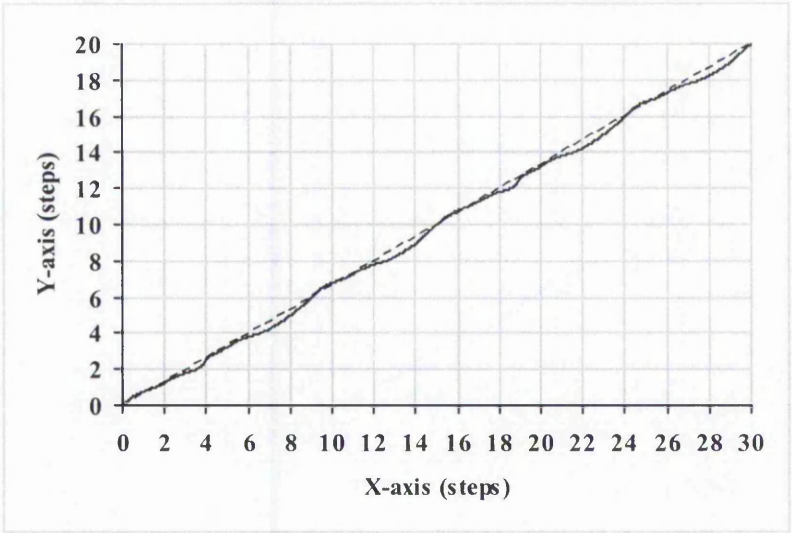


Figure 6-17: Plot of DDA Linear Interpolation (Second Order Simulation). The damping factor is 0.7. Largest Error = 0.38.

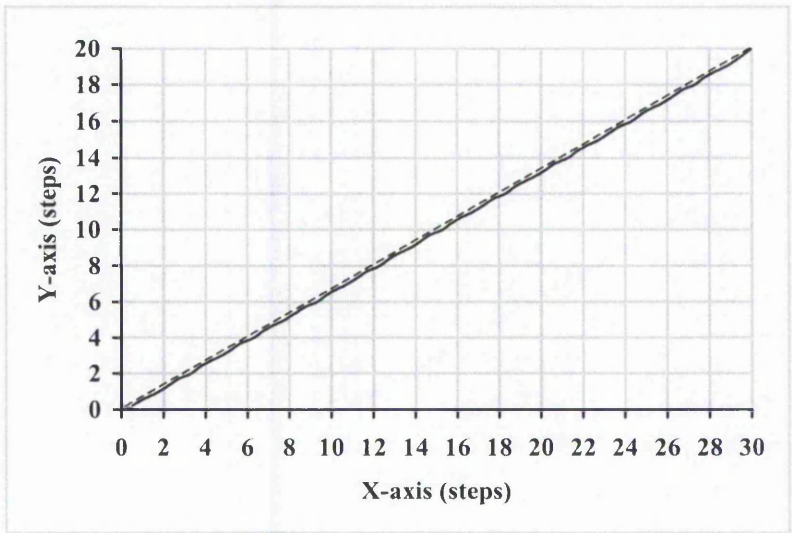


Figure 6-18: Plot of New Full Step Linear Interpolation (Second Order Simulation). The damping factor is 0.7. Largest Error = 0.19.

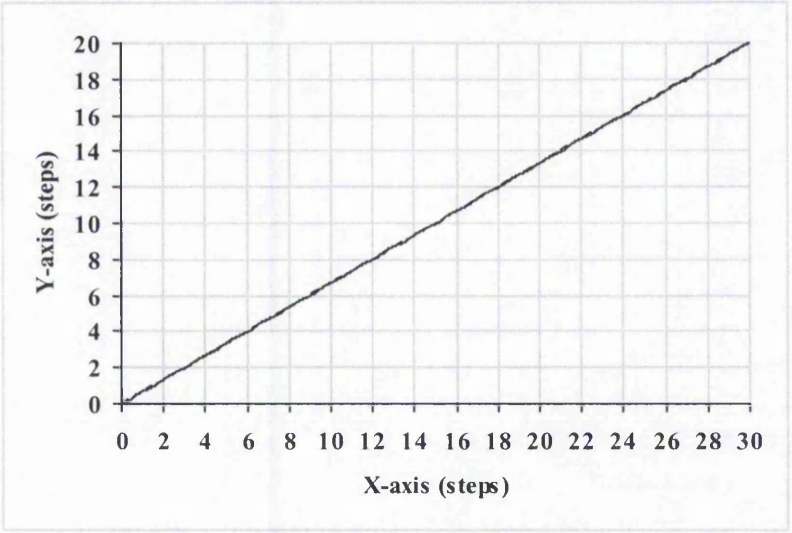
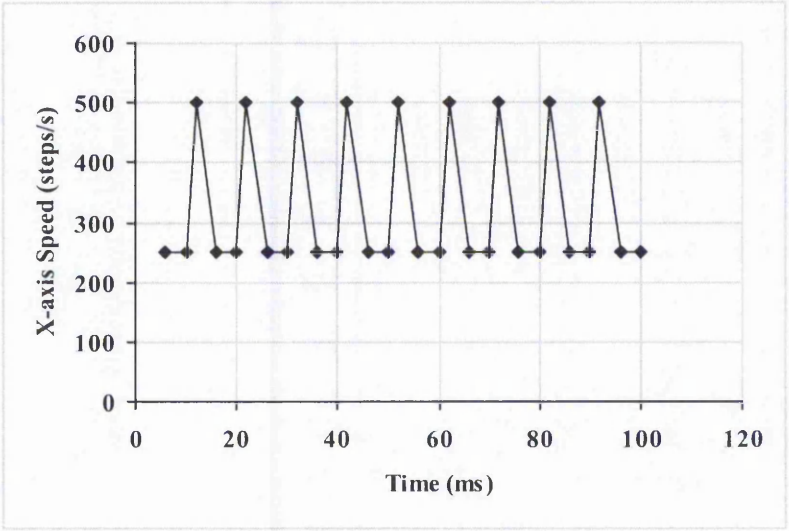


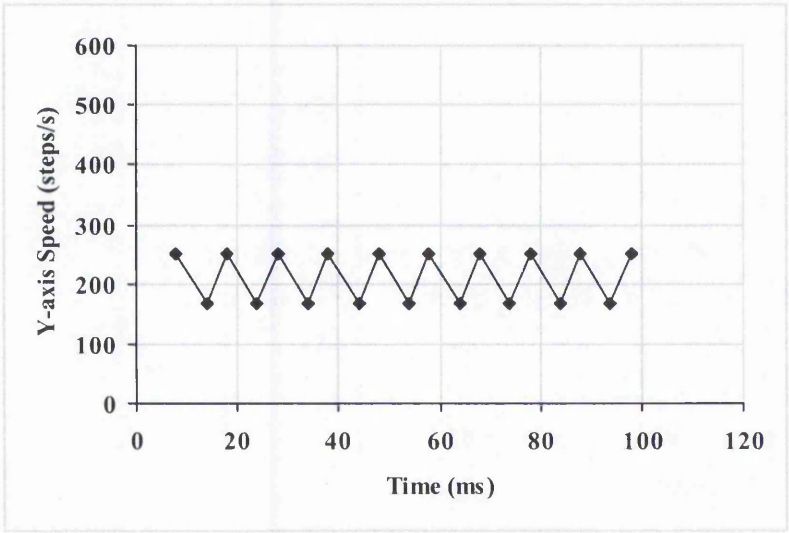
Figure 6-19: Plot of New Half Step Linear Interpolation (Second Order Simulation). The damping factor is 0.7. Largest Error = 0.06.

With Second Order simulation, the New Half-Step Linear interpolation can be seen to produce very small position errors, with the largest at only 0.06 of a step. It has been found that the largest position error for the New Half Step linear interpolation is smaller than for the other algorithms (apart from one case with Zero Order simulation). Therefore, from all the position errors simulation, it can be concluded that the New Half Step Linear interpolation produces the most desirable path, because it is expected to follow the required path very closely.

The speed variation for X and Y axes is shown in Figure 6-20 to Figure 6-24 for the different interpolation algorithms. Maintaining a constant speed is important to reduce the chance of vibrations, which is the main reason for developing the new algorithms. From the speed simulation results, only the two new linear interpolation algorithms are seen to produce a good constant speed for each individual axis. For the other algorithms, at least one of the individual axis speeds fluctuates between two speed levels. This is undesirable and increases the likelihood of machine vibrations. Although the X-axis speed in the Direct-Search algorithm is constant, fluctuation in speed is still exhibited in the Y-axis. Taken together with the results for position errors, these results show that the New Half Step algorithm produces the best behaviour compared to all the other algorithms.

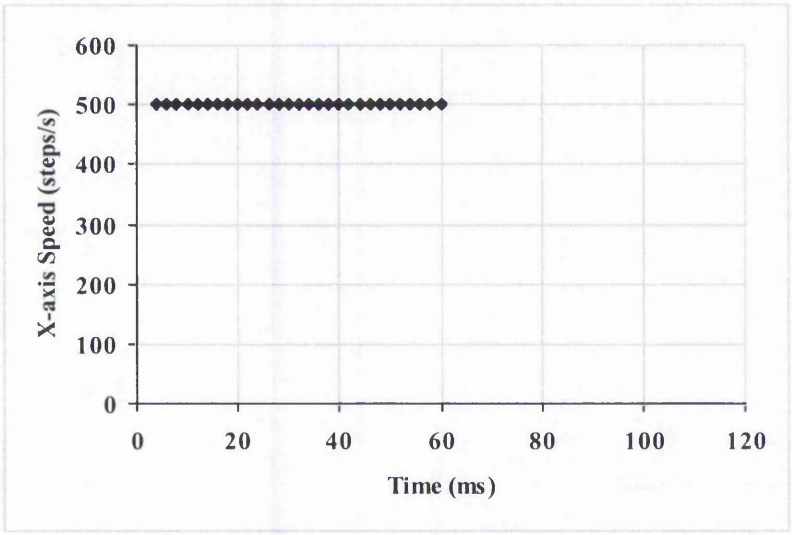


(a)

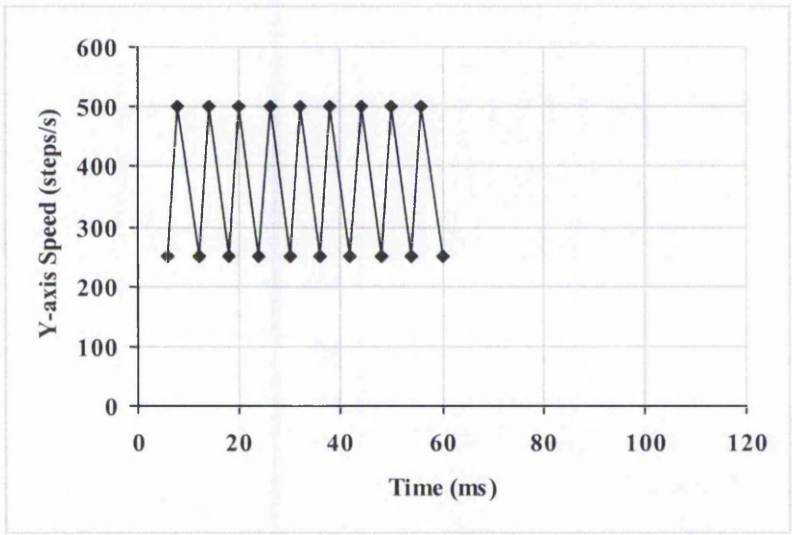


(b)

Figure 6-20: Speeds for Search-Step Linear Interpolation: (a) X-axis; (b) Y-axis.

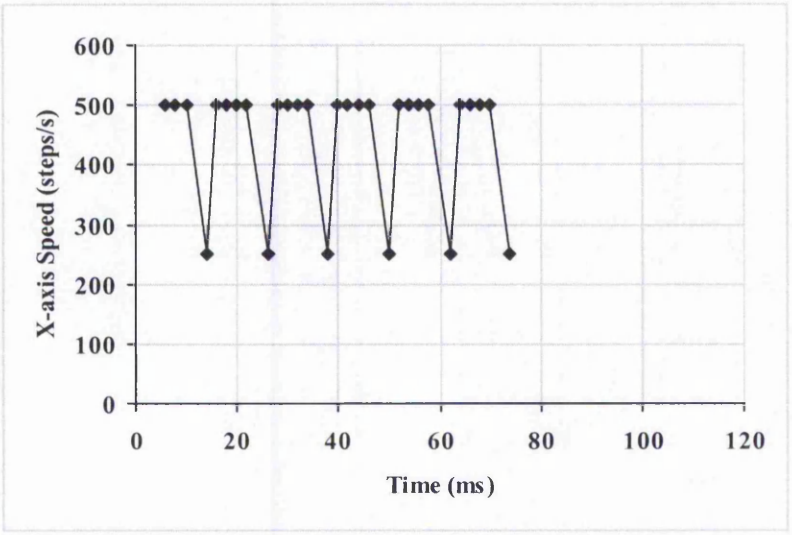


(a)

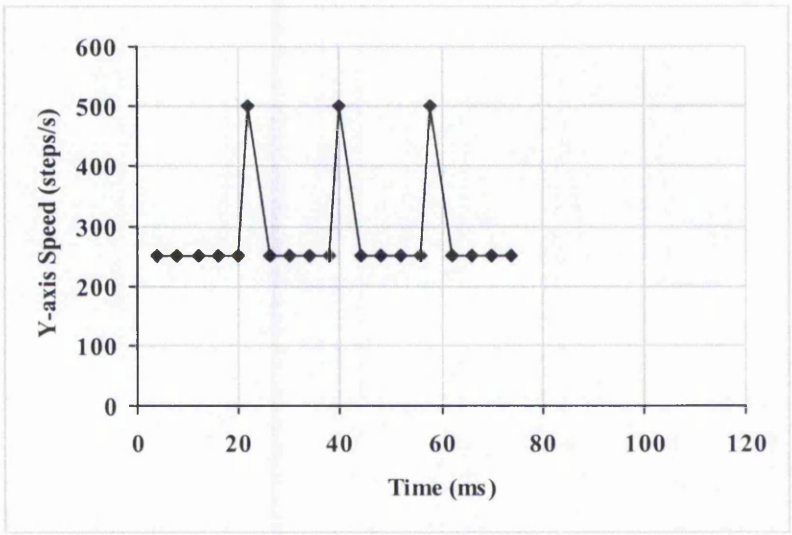


(b)

Figure 6-21: Speeds for Direct-Search Linear Interpolation: (a) X-axis; (b) Y-axis.

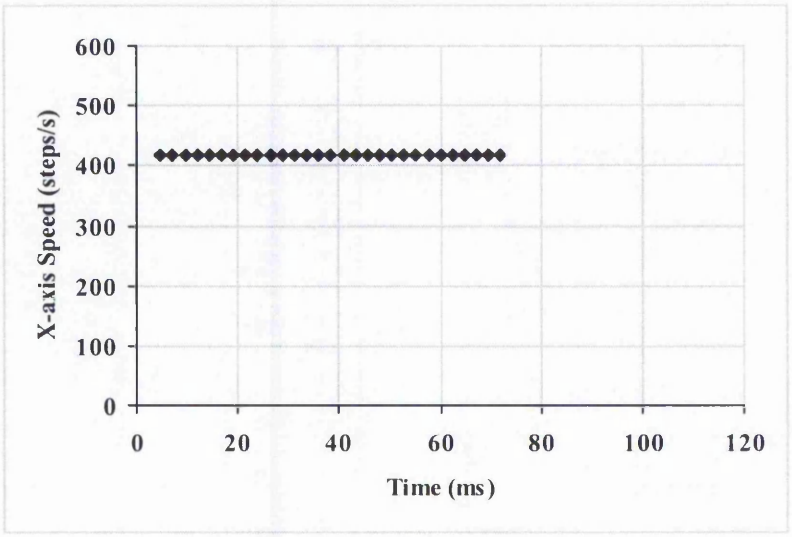


(a)

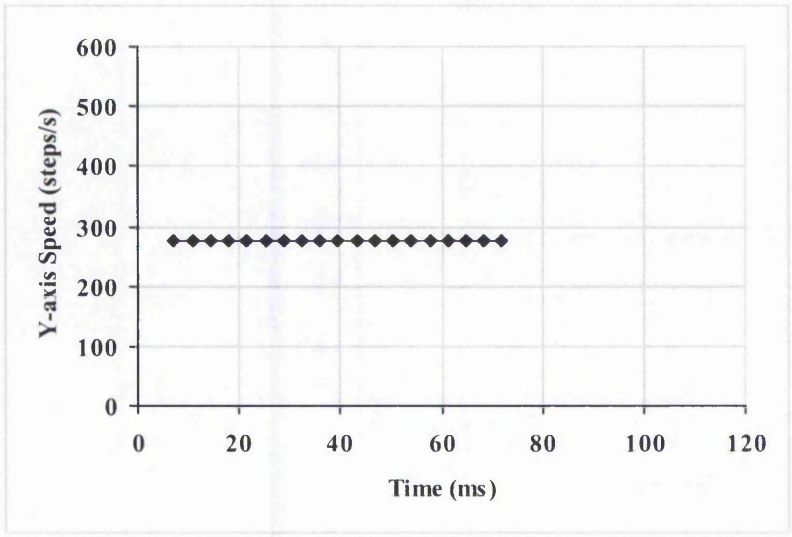


(b)

Figure 6-22: Speeds for DDA Linear Interpolation: (a) X-axis; (b) Y-axis.

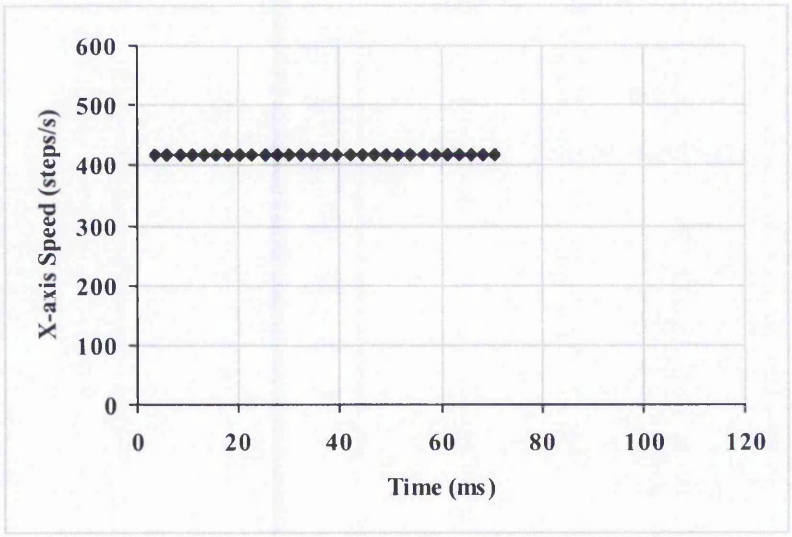


(a)

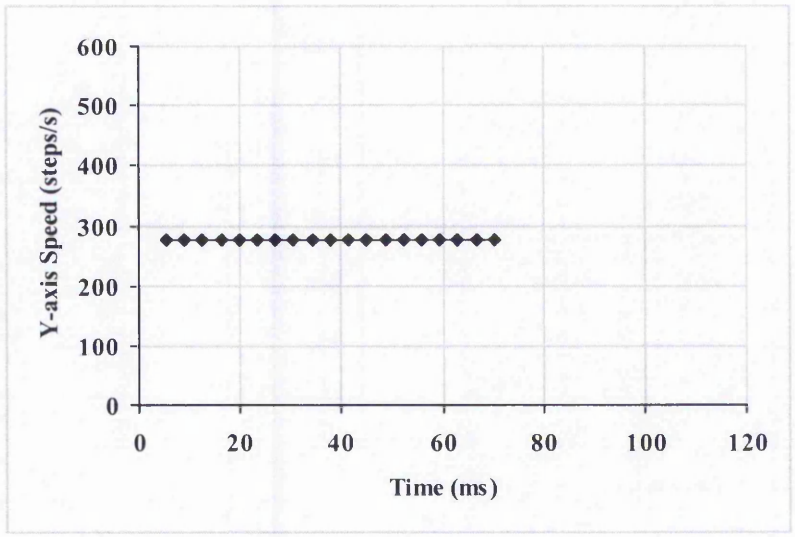


(b)

Figure 6-23: Speeds for New Full Step Linear Interpolation: (a) X-axis; (b) Y-axis.



(a)



(b)

Figure 6-24: Speeds for New Half Step Linear Interpolation: (a) X-axis; (b) Y-axis.

6.2.2 Simulation Results for Circular Arc Interpolation

As in the case of the linear example, the five interpolation algorithms have been compared. Firstly, the Zero Order simulated paths are shown in Figure 6-25 to Figure 6-29. As in the linear case, all error plots are shown in Appendix B. It can be seen that all the algorithms are able to follow the required arc closely. Again, the Search-Step interpolation works by looking at the position errors. If it has a positive error, it will try to move towards the negative error in the following command step. Negative errors correspond to inside of the circle while positive errors correspond to outside of the circle. This can cause a high position error, as can be seen at the beginning and end of the interpolation. As in the linear case, the Direct-Search algorithm appears to have smaller errors than the others.

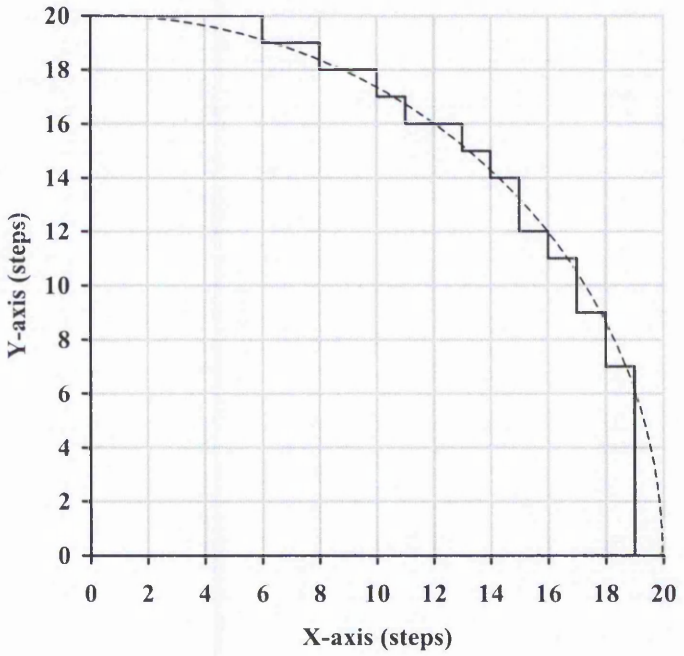


Figure 6-25: Plot of Search-Step Circular Arc Interpolation (Zero and Constant Rate First Order Simulation). Largest Error = 1.00.

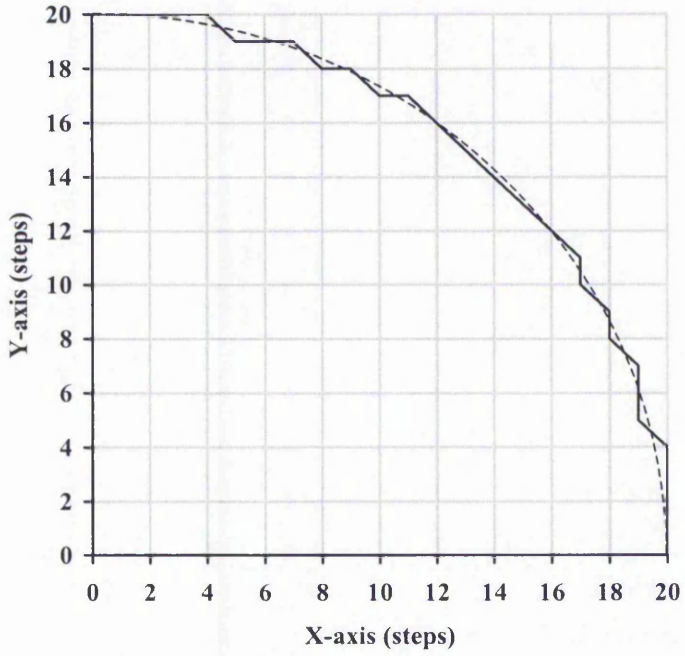


Figure 6-26: Plot of Direct-Search Circular Arc Interpolation (Zero and Constant Rate First Order Simulation). Largest Error = 0.40.

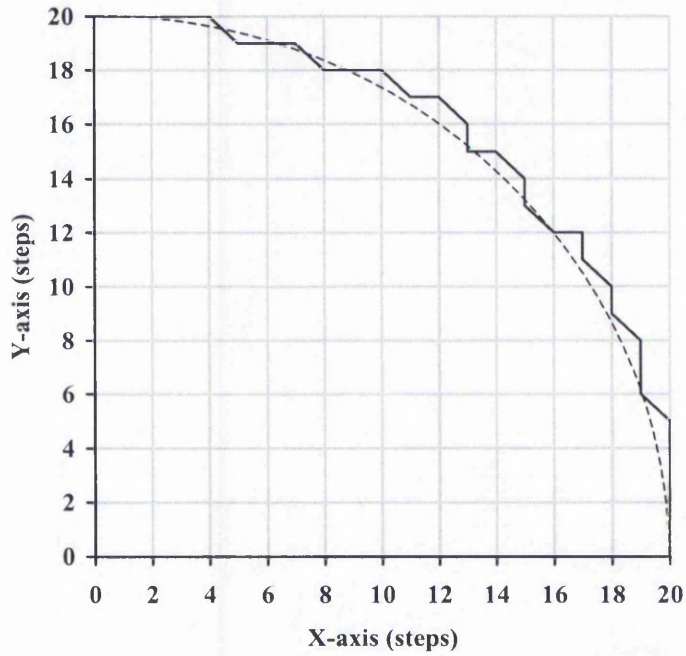


Figure 6-27: Plot of DDA Circular Arc Interpolation (Zero and Constant Rate First Order Simulation). Largest Error = 0.81.

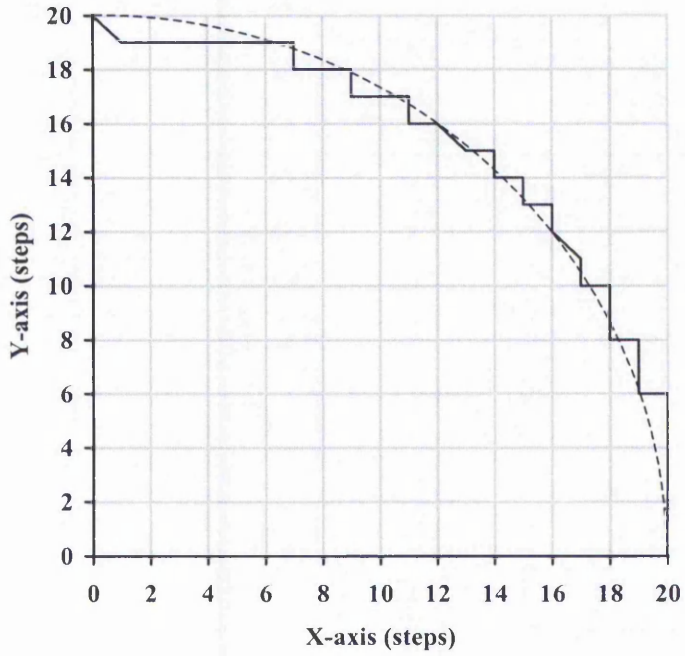


Figure 6-28: Plot of New Full Step Circular Arc Interpolation (Zero Order Simulation). Largest Error = 0.97.

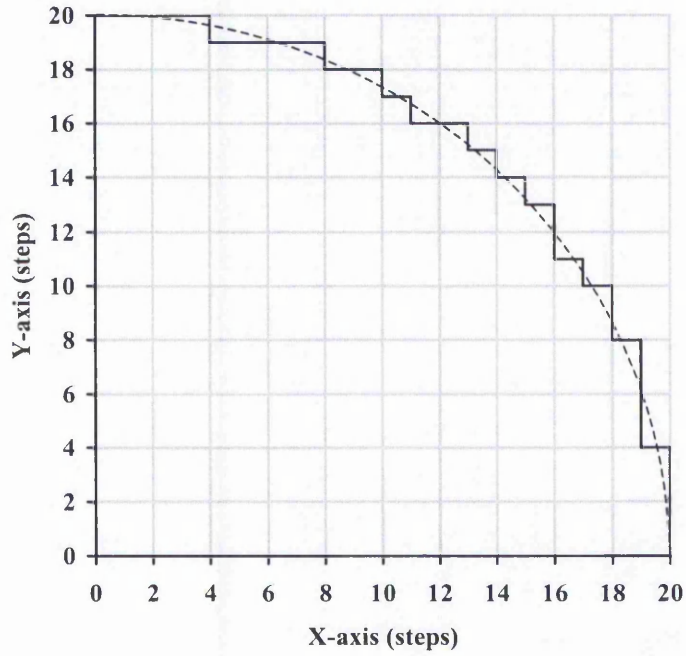


Figure 6-29: Plot of New Half Step Circular Arc Interpolation (Zero Order Simulation). Largest Error = 0.62.

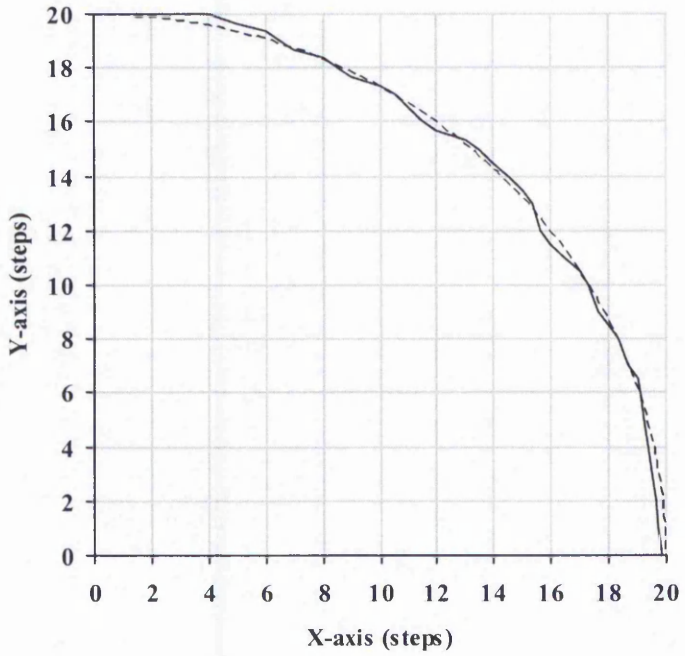


Figure 6-30: Plot of Search-Step Circular Arc Interpolation (Varying Rate First Order Simulation). Largest Error = 0.40.

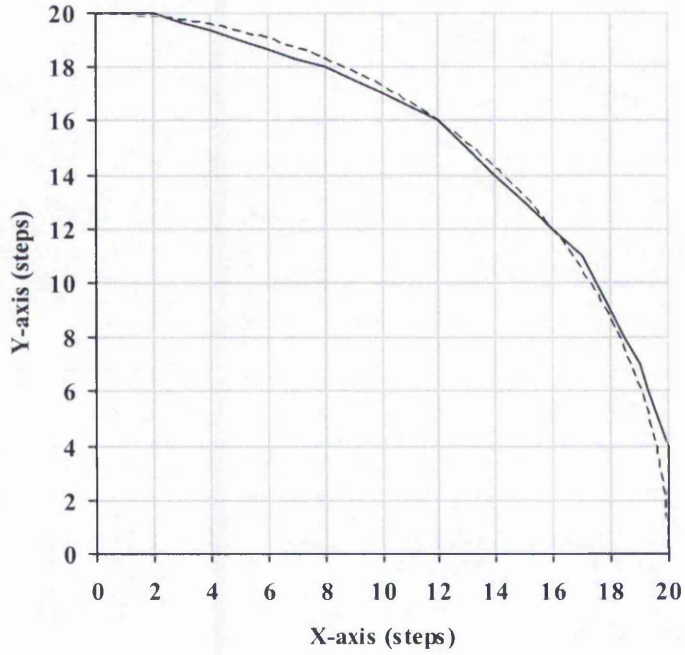


Figure 6-31: Plot of Direct-Search Circular Arc Interpolation (Varying Rate First Order Simulation). Largest Error = 0.40.

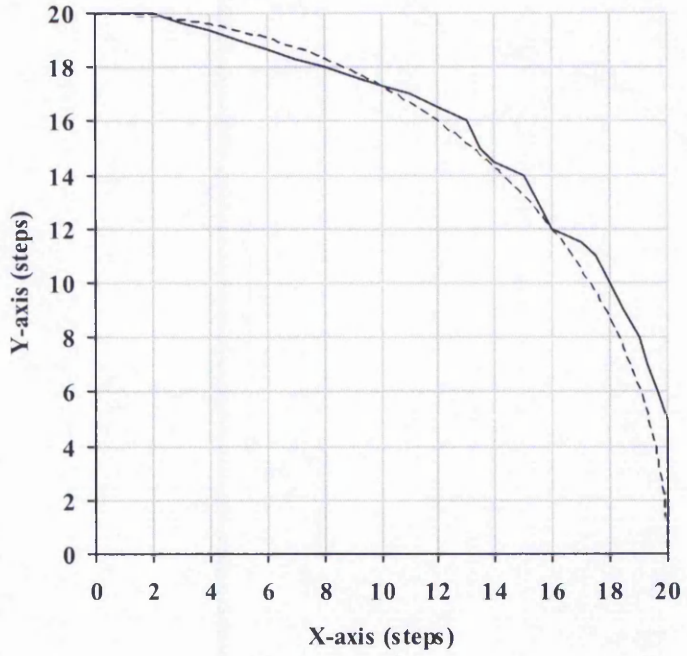


Figure 6-32: Plot of DDA Circular Arc Interpolation (Varying Rate First Order Simulation). Largest Error = 0.67.

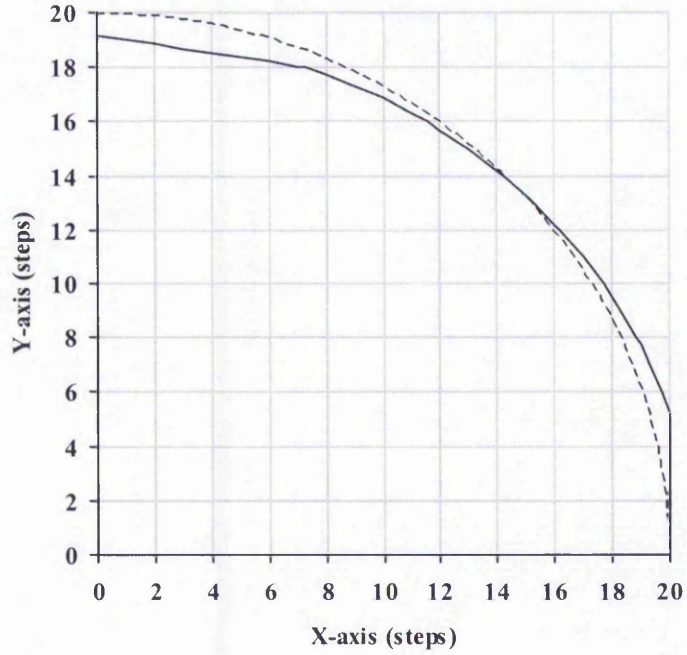


Figure 6-33: Plot of New Full Step Circular Arc Interpolation (Varying Rate First Order Simulation). Largest Error = 1.08.

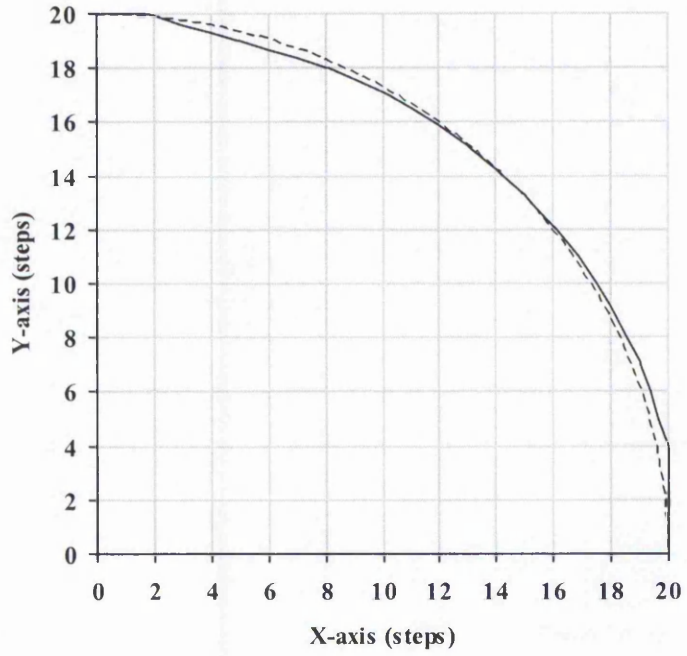


Figure 6-34: Plot of New Half Step Circular Arc Interpolation (Varying Rate First Order Simulation). Largest Error = 0.39.

The Varying Rate First Order simulation is shown in Figure 6-30 to Figure 6-34. Unfortunately, the largest position error for the New Full Step interpolation is the highest, caused by the end of the interpolation which has the final Y-axis step coming too late. However, with the used of the Half Step technique, this problem has been solved. In fact, the New Half-Step Circular Arc interpolation produces the smallest value for largest position error (0.39 step).

The Constant Rate First Order simulated paths for Search-Step, Direct-Search and DDA interpolation are same as the ones for the Zero Order simulated path and are shown in Figure 6-25 to Figure 6-27. Therefore, only the New Full and Half step interpolated path are presented here, Figure 6-35 and Figure 6-36. The largest position error for the New Half Step algorithm is 0.39 step, less than half a motor step and none of the other algorithms has a smaller value.

The Second Order simulated path for the circular arc is shown in Figure 6-37 to Figure 6-41. As in the case for the linear interpolation, the New Half Step interpolation has the

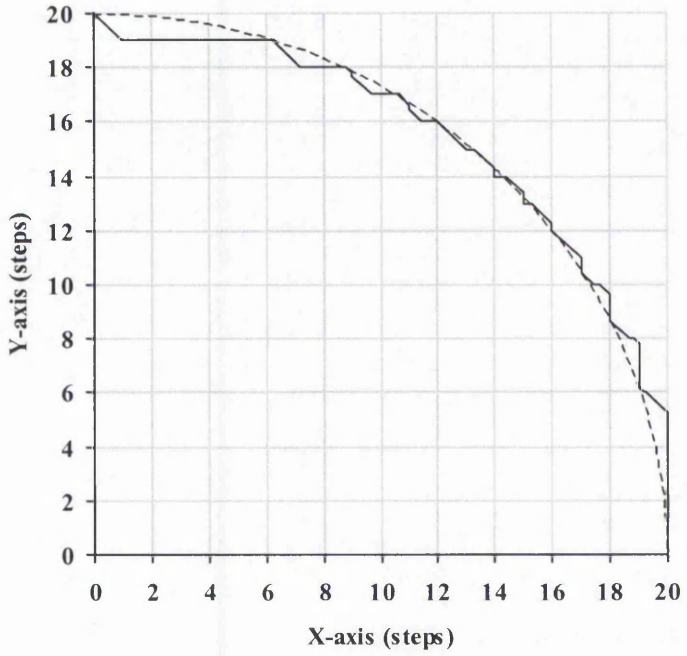


Figure 6-35: Plot of New Full Step Circular Arc Interpolation (Constant Rate First Order Simulation). The response time is 6ms. Largest Error = 0.97.

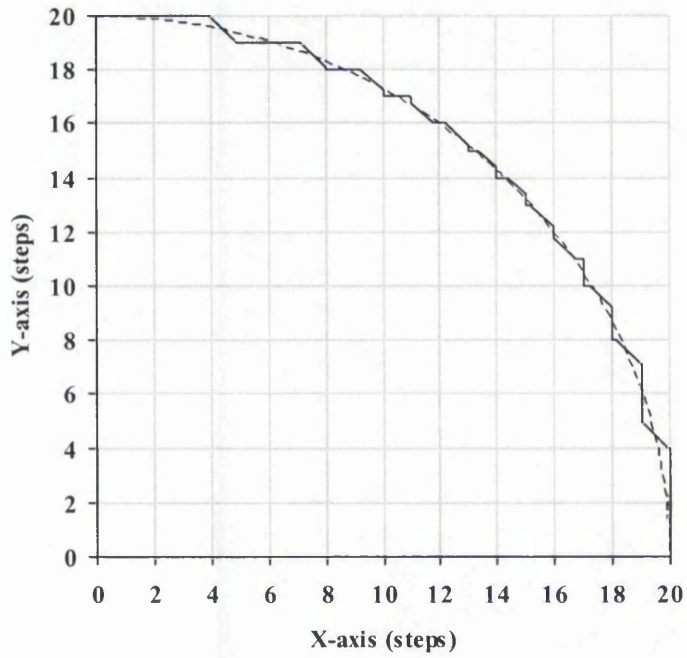


Figure 6-36: Plot of New Half Step Circular Arc Interpolation (Constant Rate First Order Simulation). The response time is 6ms. Largest Error = 0.39.

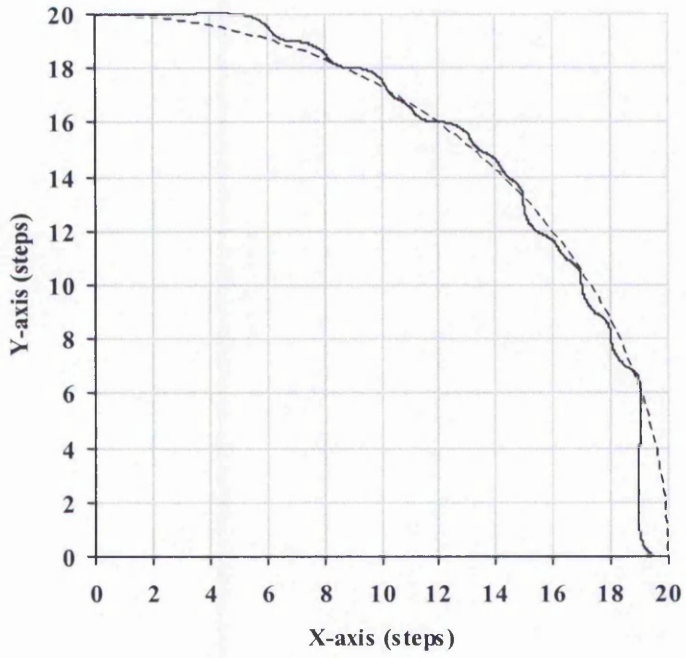


Figure 6-37: Plot of Search-Step Circular Arc Interpolation (Second Order Simulation). The damping factor is 0.7. Largest Error = 0.99.

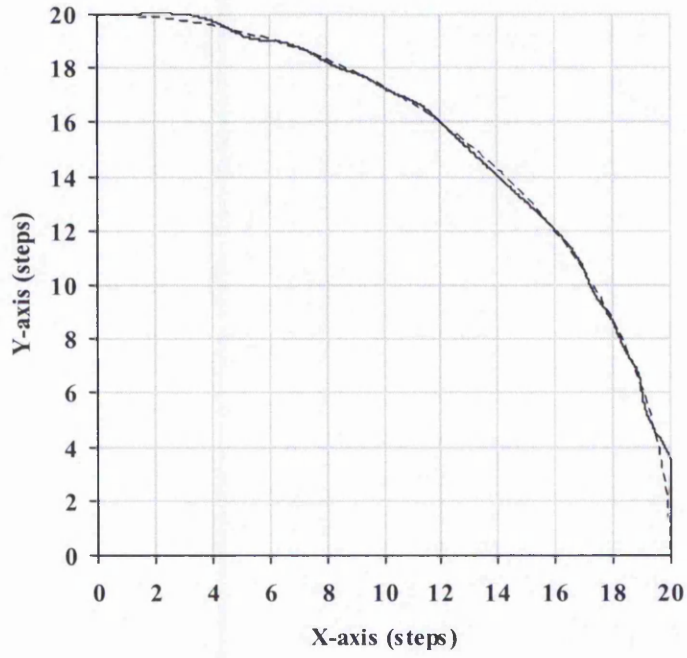


Figure 6-38: Plot of Direct-Search Circular Arc Interpolation (Second Order Simulation). The damping factor is 0.7. Largest Error = 0.29.

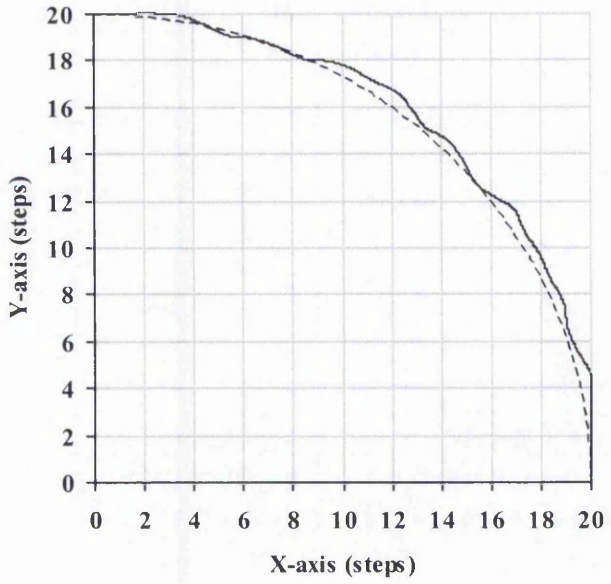


Figure 6-39: Plot of DDA Circular Arc Interpolation (Second Order Simulation). The damping factor is 0.7. Largest Error = 0.63.

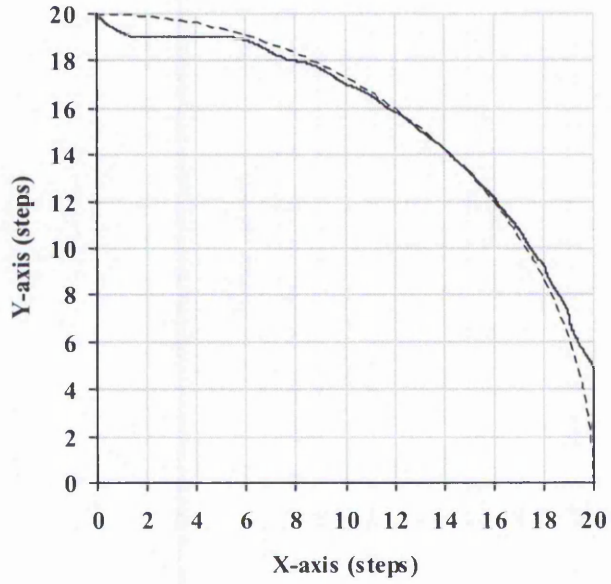


Figure 6-40: Plot of New Full Step Circular Arc Interpolation (Second Order Simulation). The damping factor is 0.7. Largest Error = 0.93.

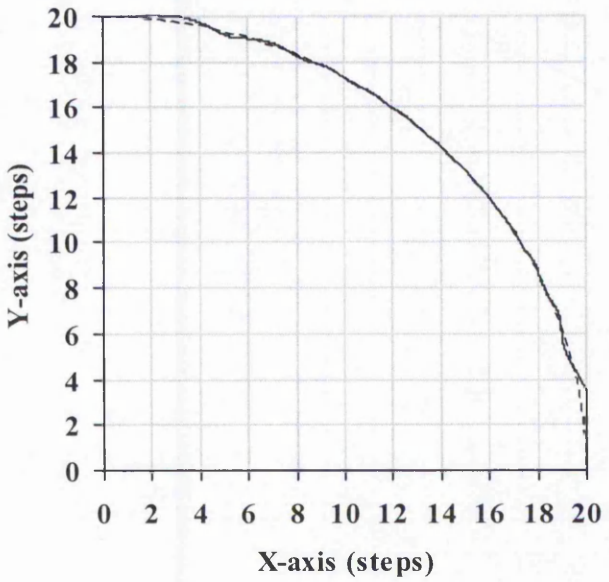
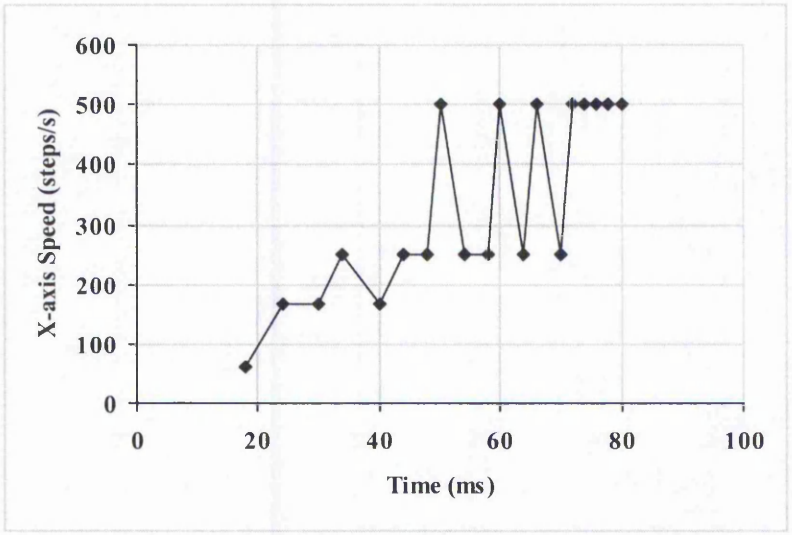


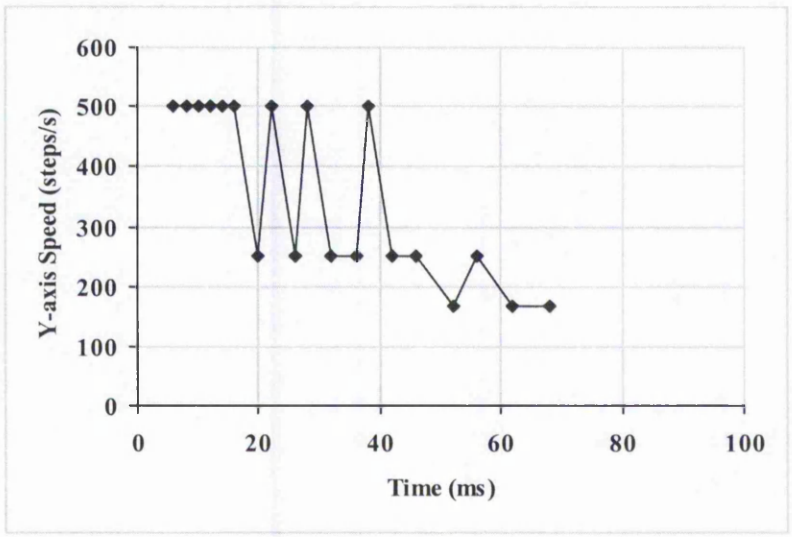
Figure 6-41: Plot of New Half Step Circular Arc Interpolation (Second Order Simulation). The damping factor is 0.7. Largest Error = 0.28.

smallest value for the largest position error for Second Order simulation, 0.28 step, which is less than half a motor step. Again it can be concluded that it produces the most desirable path.

The evaluation of the speed is performed by looking at the speed variation for each individual axis. The feedrate is 500 steps/s. Ideally, the X-axis speed should follow part of a sine wave while the Y-axis speed should follow part of a cosine wave. From the speed simulations, Figure 6-42 to Figure 6-46, it can be seen that only the new algorithms are able to produce a good smooth plot similar to the ideal shape of sine wave for X-axis or cosine wave for Y-axis. The speeds for the other algorithms fluctuate considerably around the ideal shape of the speed. The speeds for the Direct-Search algorithm are able to maintain a constant speed for part of the time. However, the sudden changes in speed between 500 to 250 steps/s is very undesirable, increasing the likelihood of machine vibrations. Thus, together with the results of the position errors, it can be seen that the New Half Step algorithm produces the best behaviour compared to all the other algorithms. Figure 6-46(c) shows the overall speed from the New Half Step algorithm. The highest variation is approximately 31 steps/s (6.2%).

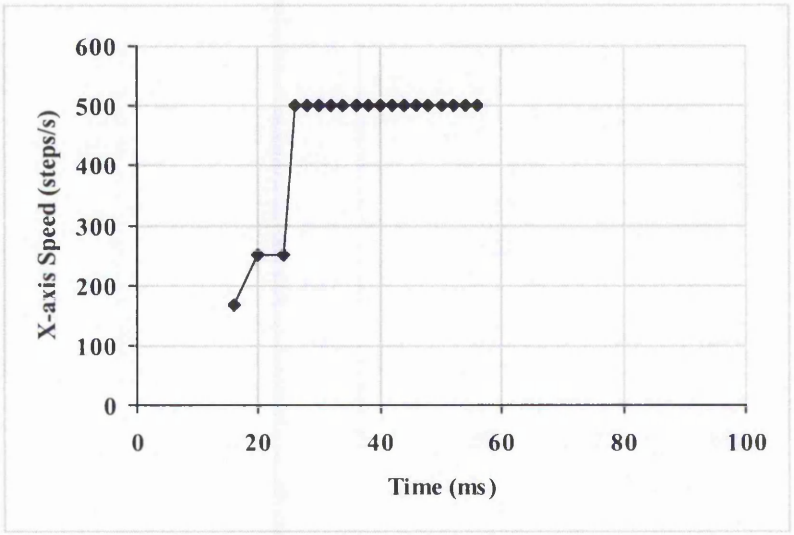


(a)

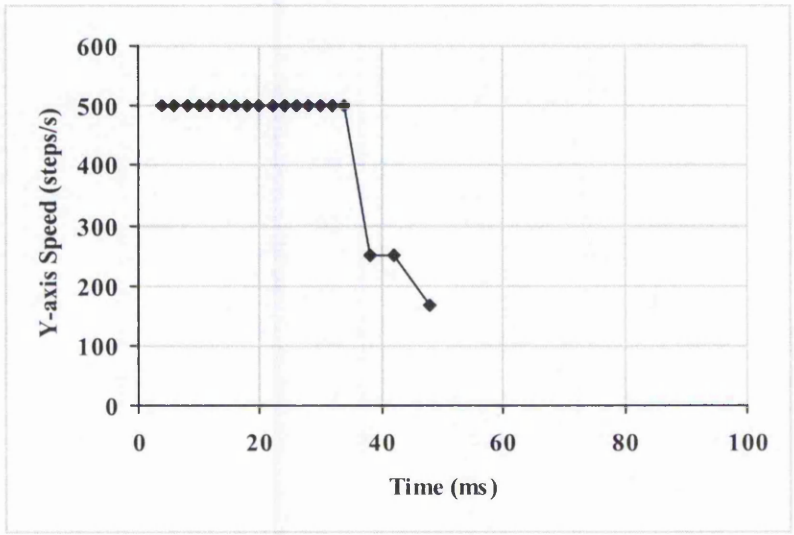


(b)

Figure 6-42: Speeds for Search-Step Circular Arc Interpolation: (a) X-axis; (b) Y-axis.

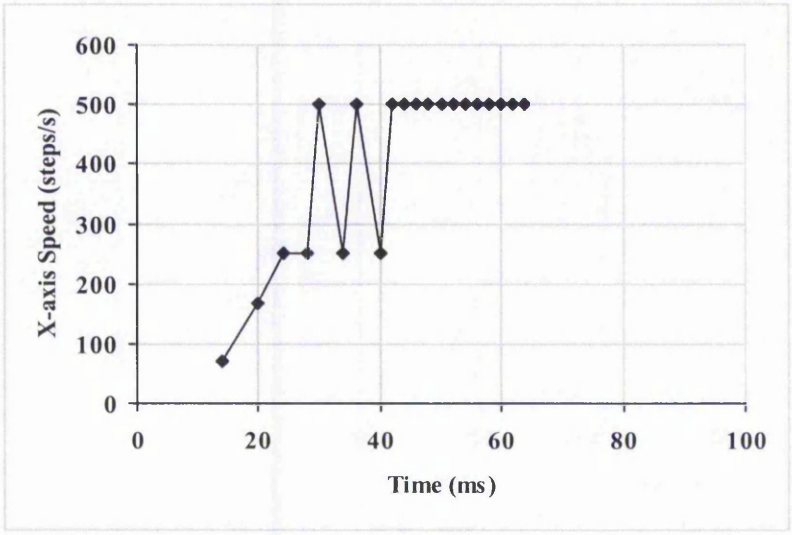


(a)

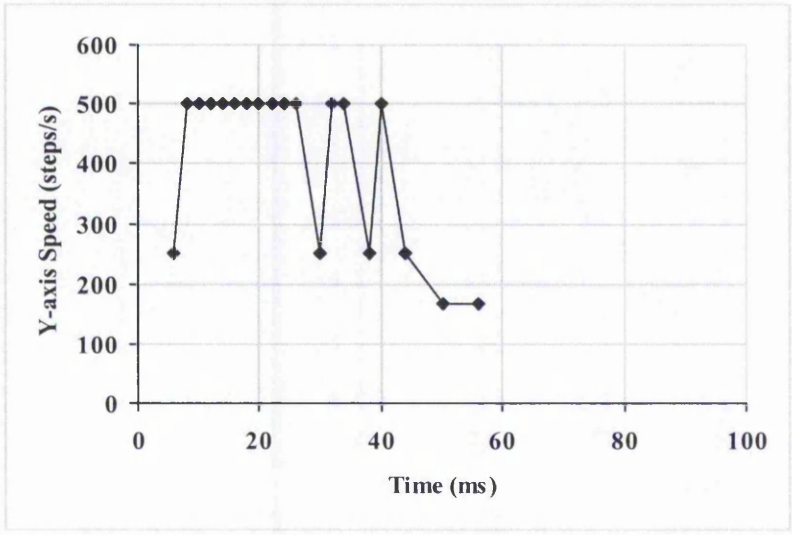


(b)

Figure 6-43: Speeds for DSM Circular Arc Interpolation: (a) X-axis; (b) Y-axis.

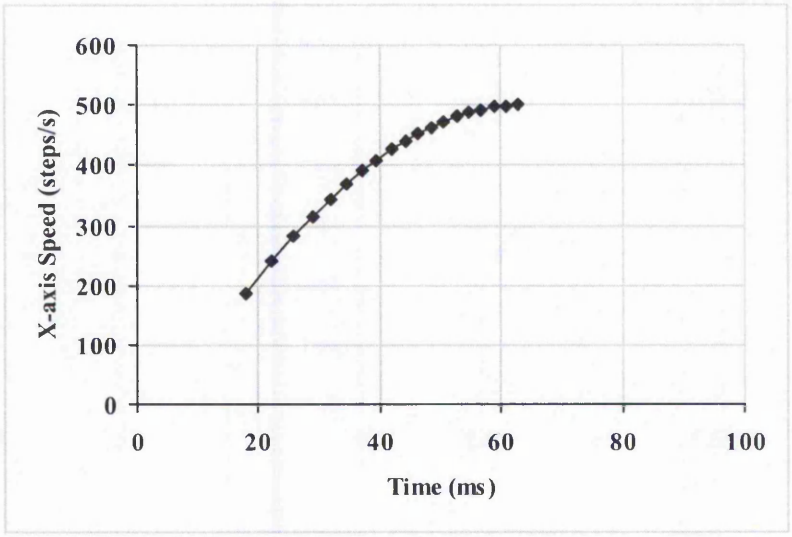


(a)

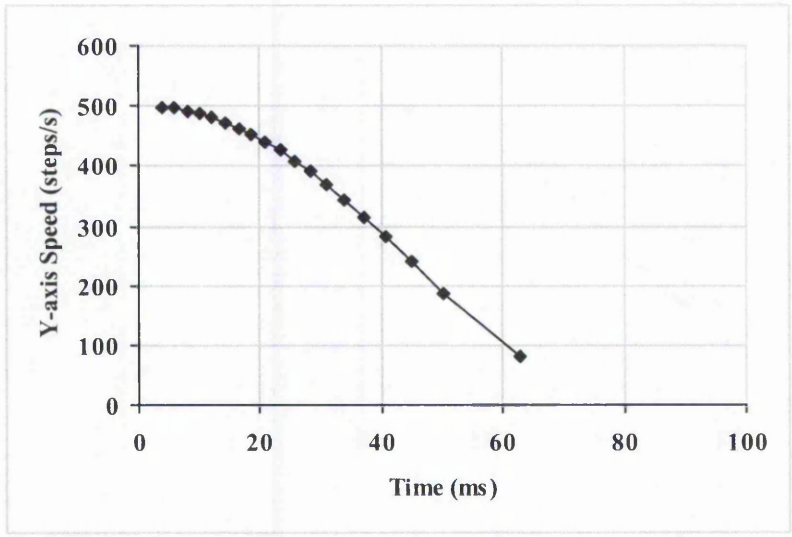


(b)

Figure 6-44: Speeds for DDA Circular Arc Interpolation: (a) X-axis; (b) Y-axis.

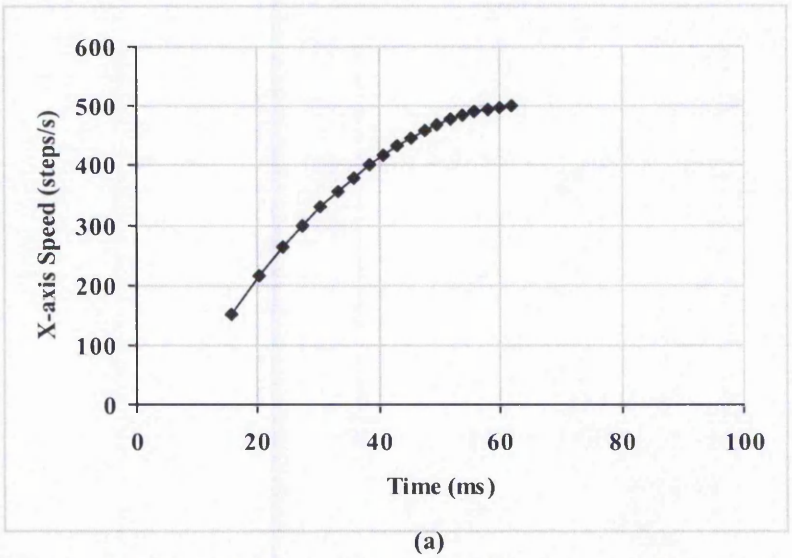


(a)

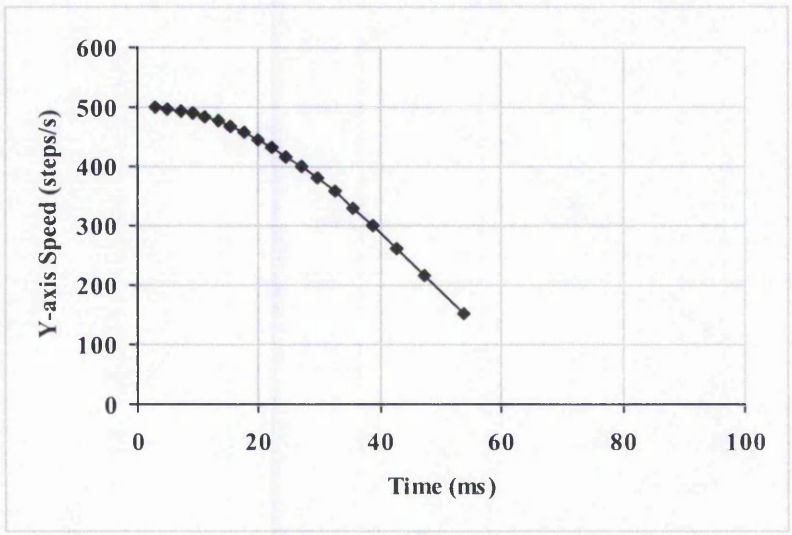


(b)

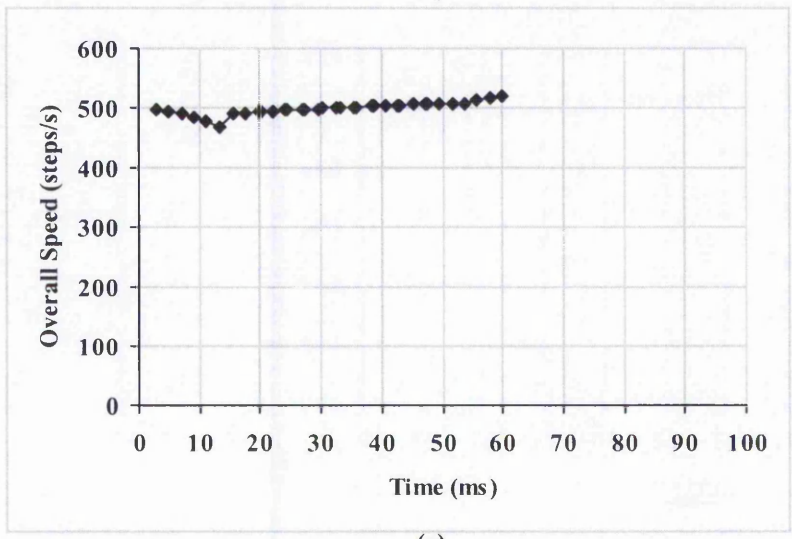
Figure 6-45: Speeds for New Full Step Circular Arc Interpolation: (a) X-axis; (b) Y-axis.



(a)



(b)



(c)

Figure 6-46: Speeds for New Half Step Circular Arc Interpolation: (a) X-axis; (b) Y-axis; (c) Overall Speed.

6.2.3 Discussion

The existence of speed fluctuations or abrupt changes in speed with all the previous algorithms is a big disadvantage because they are likely to result in more errors during actual machining by causing machine vibrations. The new algorithms are greatly superior when dealing with speed, because there are no sudden changes in speed. When there are changes in speed, in the case of the arc, the speed always changes gradually. This is important to minimise the likelihood of any machine vibrations.

Table 6-1 to Table 6-3 summarise the values obtained for the largest position errors and the average of the signed errors for the different interpolation algorithms under simulation. Two different speeds have been chosen: 0.1 m/min is very low speed and 0.3 m/min is a higher speed. Higher speeds cannot be simulated with Second Order simulation unless acceleration is used. The average error is taken over the signed error values (where positive in one side of the curve and negative the other). The Zero and Varying Rate First Order simulation produce the same result regardless of the speed, while Constant Rate First Order simulation is more appropriate for lower speed (0.1 m/min) because it assumes that the motor step is completed before the next command pulse.

It has been explained in Section 6.2.1 that Zero Order simulation is not reliable for detailed values because it is only useful to give a rough idea of the path and not the actual motion of the motors. Therefore the Zero Order results have not been used for detailed evaluation.

From these tables, it can be seen that the New Half Step interpolation algorithm is able to follow the desired path very closely. In a few cases, the Search-Step or the Direct Search interpolation algorithm is slightly closer on average to the desired path. However, the New Half Step algorithm always has the smallest value for the largest error.

For the Second Order simulation from both the largest and average position errors, the New Half Step linear interpolation performs the best. For circular arc interpolation at

Table 6-1: Position Errors for Different Interpolation Algorithms (regardless of the speed).

Simulation Type	Interpolation Algorithm	Line		Circular Arc	
		Largest Error	Average Error	Largest Error	Average Error
Zero Order	Search-Step	0.55	0.00	1.00	-0.09
	Direct-Search	0.28	0.00	0.40	0.01
	DDA	0.55	-0.14	0.81	0.24
	New Full Step	0.55	-0.14	0.97	-0.03
	New Half Step	**0.55**	**0.00**	**0.62**	**0.06**
Varying Rate First Order	Search-Step	0.28	-0.12	0.40	-0.02
	Direct-Search	0.42	-0.14	0.40	-0.01
	DDA	0.55	-0.27	0.67	0.20
	New Full Step	0.28	-0.28	1.08	-0.14
	New Half Step	**0.14**	**-0.14**	**0.39**	**0.00**

Table 6-2: Position Errors for Different Interpolation Algorithms (at 0.1 m/min).

Simulation Type	Interpolation Algorithm	Line		Circular Arc	
		Largest Error	Average Error	Largest Error	Average Error
Constant Rate First Order	Search-Step	0.55	0.00	1.00	-0.09
	Direct-Search	0.28	0.00	0.40	0.01
	DDA	0.55	-0.14	0.81	0.24
	New Full Step	0.33	-0.14	0.97	-0.06
	New Half Step	**0.22**	**0.00**	**0.39**	**0.03**
Second Order	Search-Step	0.58	0.00	1.05	-0.08
	Direct-Search	0.30	0.00	0.43	0.00
	DDA	0.57	-0.13	0.83	0.22
	New Full Step	0.46	-0.14	0.98	-0.07
	New Half Step	**0.28**	**0.00**	**0.42**	**0.02**

Table 6-3: Position Errors for Different Interpolation Algorithms (at 0.3 m/min).

Simulation Type	Interpolation Algorithm	Line		Circular Arc	
		Largest Error	Average Error	Largest Error	Average Error
Second Order	Search-Step	0.28	0.00	0.99	-0.08
	Direct-Search	0.13	0.00	0.29	0.00
	DDA	0.38	-0.13	0.63	0.19
	New Full Step	0.19	-0.13	0.93	-0.07
	New Half Step	**0.06**	**0.00**	**0.28**	**0.02**

0.3 m/min, the New Half Step is also the best in terms of largest position error. At 0.1 m/min, the New Half Step interpolation is also best when the largest position error is considered. However, the average position error is slightly larger (by 0.02) than for the Direct-Search interpolation. The plots of position error throughout the two interpolations can be seen in Appendix B, Figure B-31 and Figure B-34. From there, it can be seen that the position error for the New Half Step algorithm is very close to zero most of the time. On the other hand, for the Direct-Search algorithm, higher fluctuations are identified over more of the path. Because of the equal fluctuations above and below the required path, the average error has turned out to be zero. Even in the case of First Order simulation, the largest position error is always smallest for the New Half Step interpolation and the average error is worse in just 2 cases by at most 0.02 step. Therefore, it is expected that the path generated by New Half Step linear interpolation will be closest to the desired path when performing the actual machining.

Based on the speed simulation, the new algorithms have by far the best performance in terms of smoothness of motion. From the First and Second Order simulation results, the Half Step algorithms can be seen to improve on the Full Step one because of the reduction in the position errors. Therefore, the Half Step algorithm is expected to give smoother motion than all the previous interpolation algorithms and there is no significant loss of accuracy.

All the simulations where speed was involved have been at low speeds, because otherwise acceleration is needed. Therefore, one acceleration algorithm has been used to investigate the effect of motion at constant high speed. Figure 6-47 shows the Second Order simulation of New Half Step interpolation of a line at high speed (4.8 m/min). The line shown in this figure is a small segment of a longer line from the part where it is travelling at constant speed. The linear acceleration algorithm has been used to accelerate up to this speed. The largest error when the motion along this line is at constant speed is approximately 0.007 step. For a circular arc of radius 2000 steps, travelling at the constant speed of 4.8 m/min (with preceding acceleration algorithm), it is found that the largest error is also 0.009 step. This error has been calculated only for the part of the arc from angle 31.79° to 60° because it was inside the region of constant speed. These results suggest that the errors are lower at high speeds.

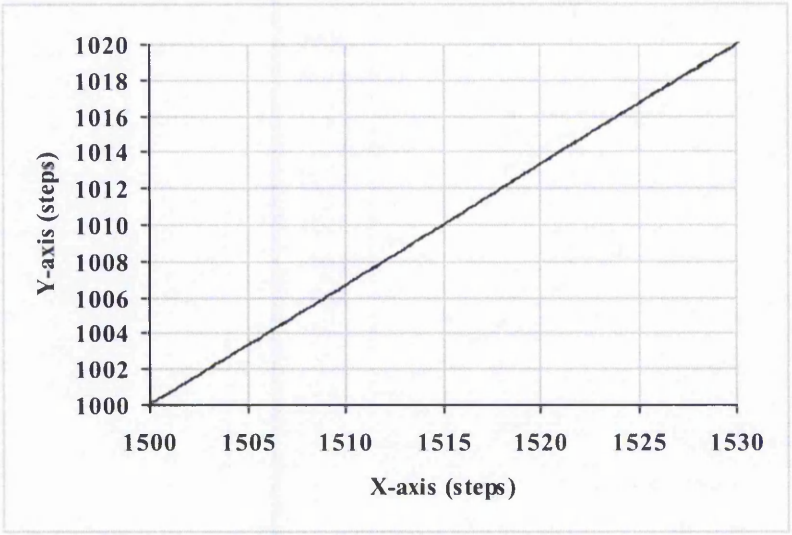


Figure 6-47: Plot of a High Speed Interpolation of a Line (New Half Step) at the Constant Phase of the Motion (Second Order simulation). Acceleration was completed well before this stage was reached.

Table 6-4: Table of Position Errors for Line at Speed 0.3 m/min.

Line Angle	0°	0.57°	33.69°	45°
Largest Error for Full Step	0	0.98	0.19	0
Largest Error for Half Step	0	0.52	0.06	0

The effect of direction of the motion has also been investigated. From the error plots for the circular arc, which are included in Appendix B. it has been found that higher errors are at the beginning and end of interpolation. This could be because the speed in one axis is much higher than the other, and not because of the acceleration. Therefore, a test has been performed on the interpolation of a line almost parallel to the X-axis, from (0,0) to (100,1) at 0.3 m/min (at an angle of 0.57°). It is found that the largest error for Second Order simulated path is 0.98 step for New Full Step interpolation and it is 0.52 step for New Half Step interpolation. These values can be seen to be considerably higher than the errors for the line (0,0) to (30,20) in Table 6-3. Table 6-4 summarises the errors that occur at different speeds. In simulation the errors for 0° are 0 because all the motion is always along the X-axis. For 45°, the two axes will behave identically so

again the error is 0. When the angle gets close to 0° but is greater than 0°, the error is thought to be largest because of the low speed on the Y-axis. In the last example, the Y-axis speed is very low at 0.003 m/min, much slower than the Y-axis speed in the example in Figure 5-15.

6.3 Evaluation of The New Acceleration Algorithms by Simulation

To enable the new interpolation algorithms to be used for high speed machining, new acceleration algorithms have been developed. The details of these algorithms are outlined in Sections 4.2 and 4.3. They are the linear and parabolic acceleration, with the parabolic acceleration appearing to have much advantage. A brief evaluation of these algorithms is included in Section 4.4. This section presents further simulation results to evaluate the two acceleration algorithms. From Section 6.2, it has been found that the New Half Step interpolation algorithms produce an improved interpolation when compared to the other interpolations. Therefore, it has been used for the evaluation of the new acceleration algorithms.

As in the case of the interpolation algorithms, the position errors help to determine whether the generated path is able to follow the required path closely, while the speed variations highlight the possible increased chance of vibrations. The position error plots are in Appendix C.

As for interpolation, one line and one arc have been used as examples to evaluate the acceleration algorithms. This time the line and arc are both much longer (compared to the examples used for the evaluation on the interpolation in Section 6.1) to allow time for acceleration and deceleration. This line is in the same direction as before and the arc is again anticlockwise in the first quadrant but with a much longer radius. The parameters used for line is as follows:

$X_{start} = 0$
$Y_{start} = 0$
X_{end} = 3000 steps (0.03m)
Y_{end} = 2000 steps (0.02m)
Feedrate = 8000 steps/s (4.8 m/min or 80 mm/s)
Starting speed = 200 steps/s (2 mm/s)

On the other hand, the parameters for circular arc example is as follows:

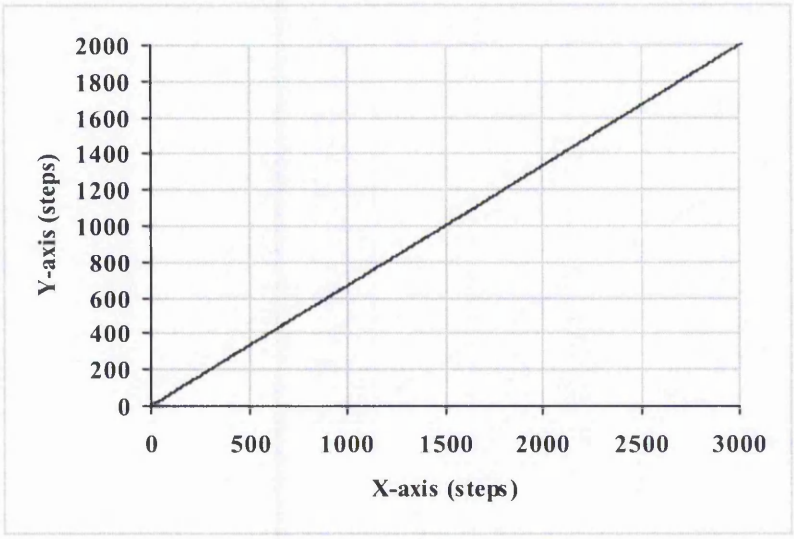
X_{start} = 2000 steps (0.02m)
Y_{start} = 0
X_{end} = 0
Y_{end} = 2000 steps (0.02m)
Centre coordinates = (0,0)
Feedrate = 8000 steps/s (4.8 m/min or 80 mm/s)
Starting speed = 200 steps/s (2 mm/s)

Typically with step size 0.01 mm, that means required speed is 4.8 m/min.

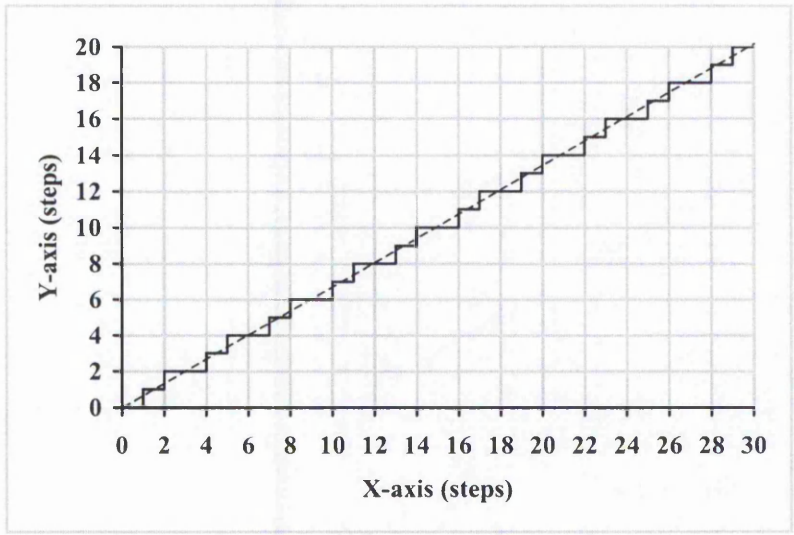
6.3.1 Simulation Results for Linear Acceleration

The first evaluation is a comparison of the position errors presented by the new linear acceleration algorithms for the line. The plots are shown in this section but detailed error plots are in Appendix C. This will in fact determine how closely the desired path is followed. The acceleration used is 35,000 steps/s^2 (0.35 m/s^2), which is a reasonable value without causing problems to the machining. The Zero Order simulation is shown in Figure 6-48(a) and Figure 6-48(b) shows a small segment of the former graph. Figure 6-48b is exactly the same as the one shown in Figure 6-7 for linear interpolation without acceleration. This demonstrates that the ratio between the X and Y movements are still be maintained in this section and the same is true throughout the interpolation. An identical path is simulated when using the same interpolation without acceleration.

The same linear example using the Varying Rate First Order simulation is shown in Figure 6-49. Figure 6-50 is the Constant Rate First Order simulated path for the linear interpolation with linear acceleration. Figure 6-50(b) is the starting segment of the interpolation where the speed is very low, so the simulated path is similar to the Zero Order simulation in Figure 6-48.



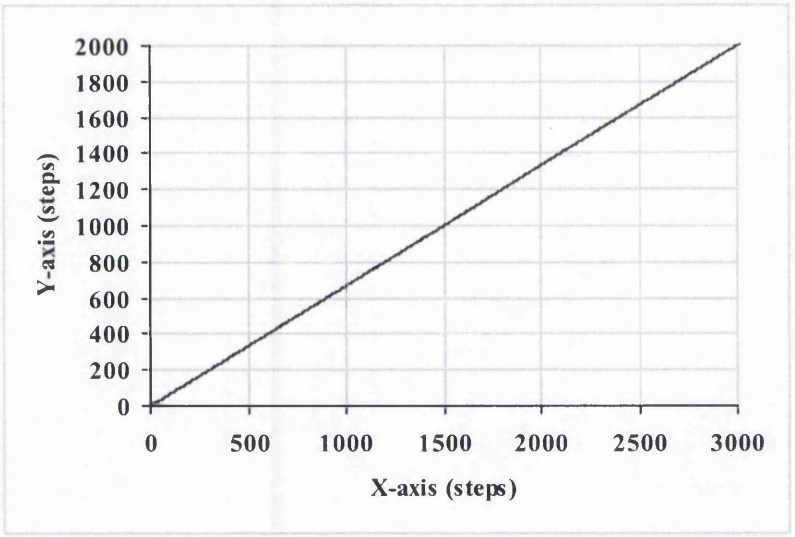
(a)



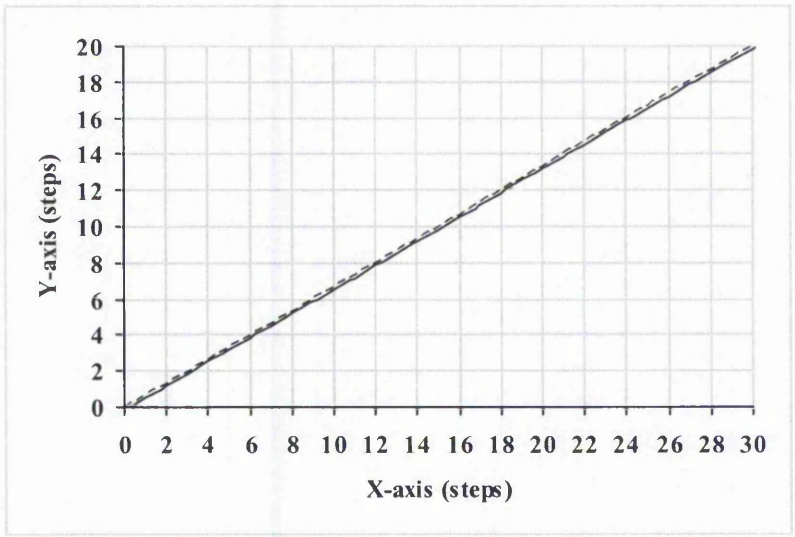
(b)

Figure 6-48: Plot of New Half Step Linear Interpolation with Linear Acceleration (Zero Order Simulation). Largest Error = 0.55 (a) Full graph; (b) Detail of path for 30 steps in X-axis.

The Second Order simulated path is shown in Figure 6-51. The slower speed at the start of the interpolation can be seen to have small position errors. The simulated path becomes much smoother after the second X-axis step. This is because the path is smoother at higher speeds.

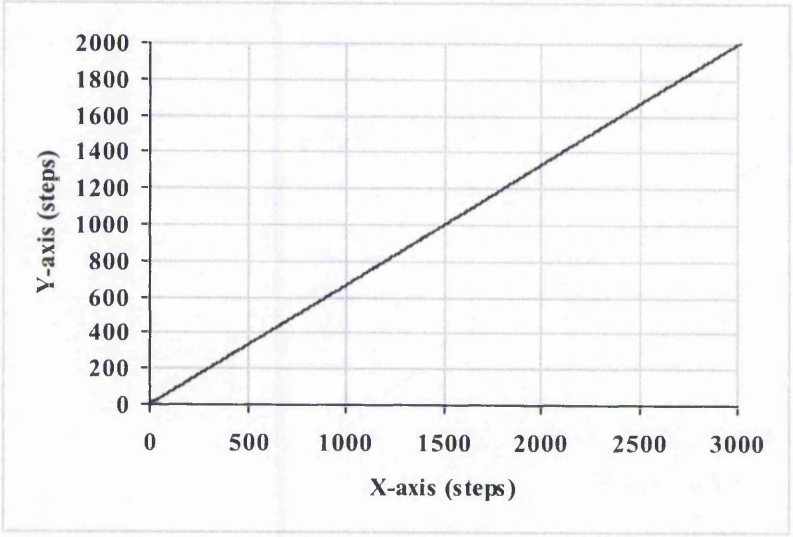


(a)

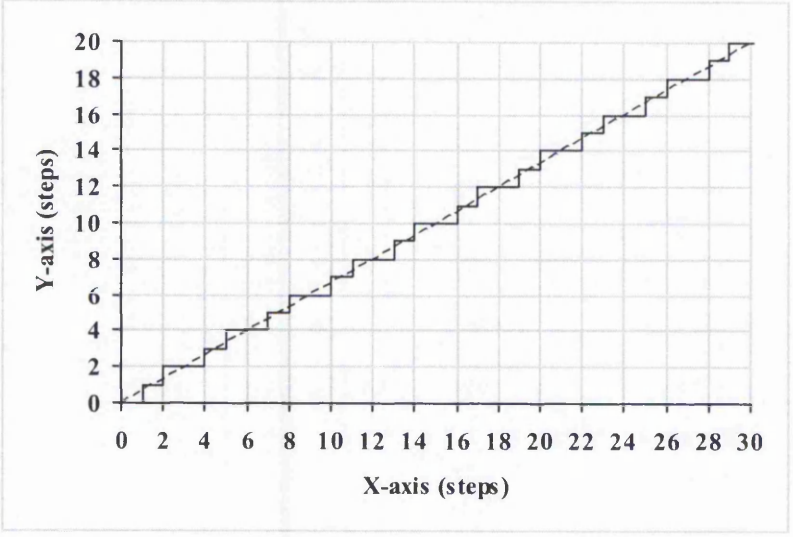


(b)

Figure 6-49: Plot of New Half Step Linear Interpolation with Linear Acceleration (Varying Rate First Order Simulation). Largest Error = 0.18 (a) Full graph; (b) Detail of path for 30 steps in X-axis.

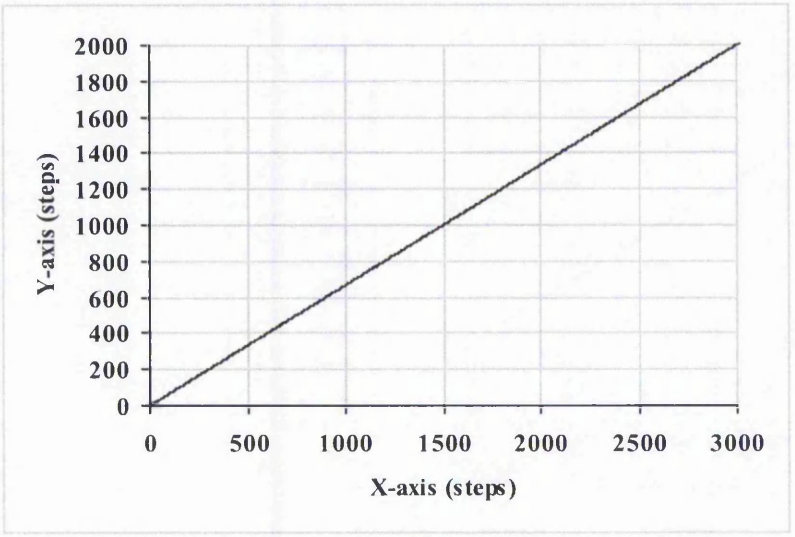


(a)

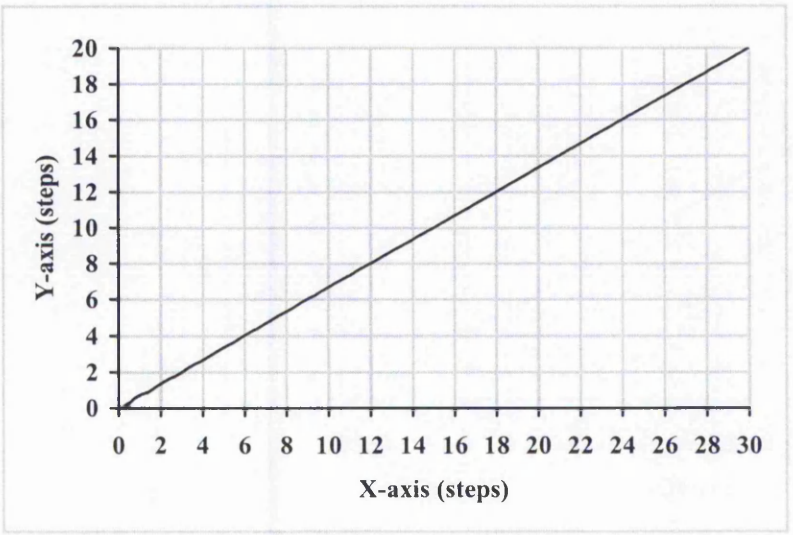


(b)

Figure 6-50: Plot of New Half Step Linear Interpolation with Linear Acceleration (Constant Rate First Order Simulation; Response time = 0.1 ms). Largest Error = 0.55 (a) Full graph; (b) Detail of path for 30 steps in X-axis.



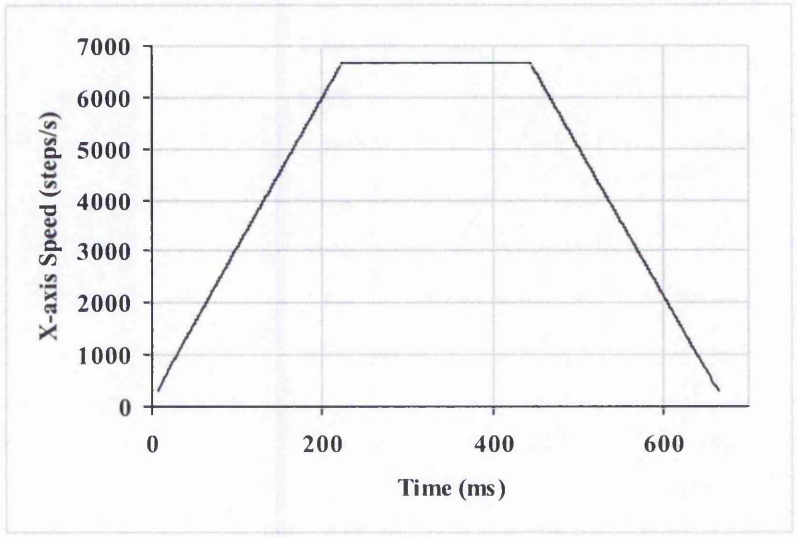
(a)



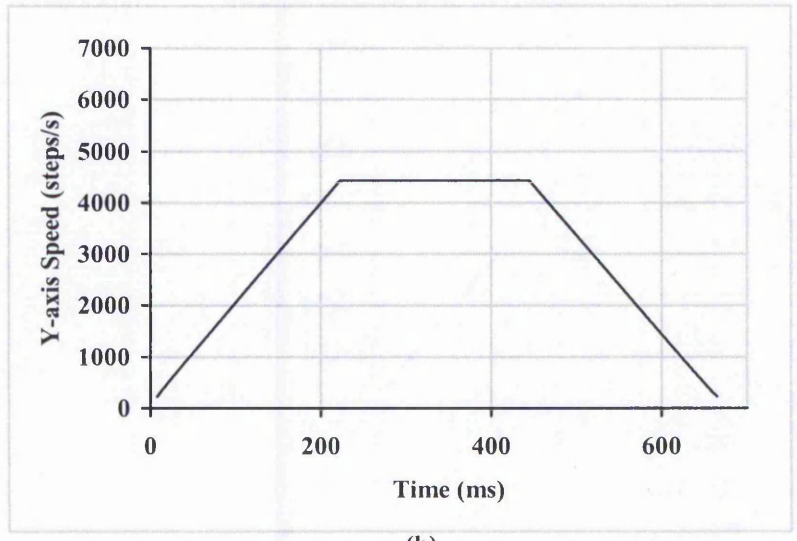
(b)

Figure 6-51: Plot of New Half Step Linear Interpolation with Linear Acceleration (Second Order Simulation). Largest Error = 0.11 (a) Full graph; (b) Detail of path for 30 steps in X-axis.

The speed plots for the line are shown in Figure 6-52. The new half step linear interpolation will produce a constant speed on both axes. With the inclusion of the linear acceleration, the speed increases linearly from the starting speed until it reaches



(a)



(b)

Figure 6-52: Simulation of Speed with Linear Acceleration/Deceleration for Linear Interpolation for (a) X-axis; (b) Y-axis. (X-axis = 3000 steps, Y-axis = 2000 steps. Acceleration = 35000 pulses/s^2. Maximum Resultant Speed = 8000 steps/s. Acceleration stops at 220 ms and deceleration starts at 445 ms)

the required speed (at 220 ms). At a particular instant (445 ms), the speed starts to decrease linearly to the stopping speed (which is taken to be same as the starting speed).

The new linear acceleration can also be employed with the circular arc interpolation. This is explained in Section 4.2.1. The following figures (Figure 6-53 to Figure 6-56)

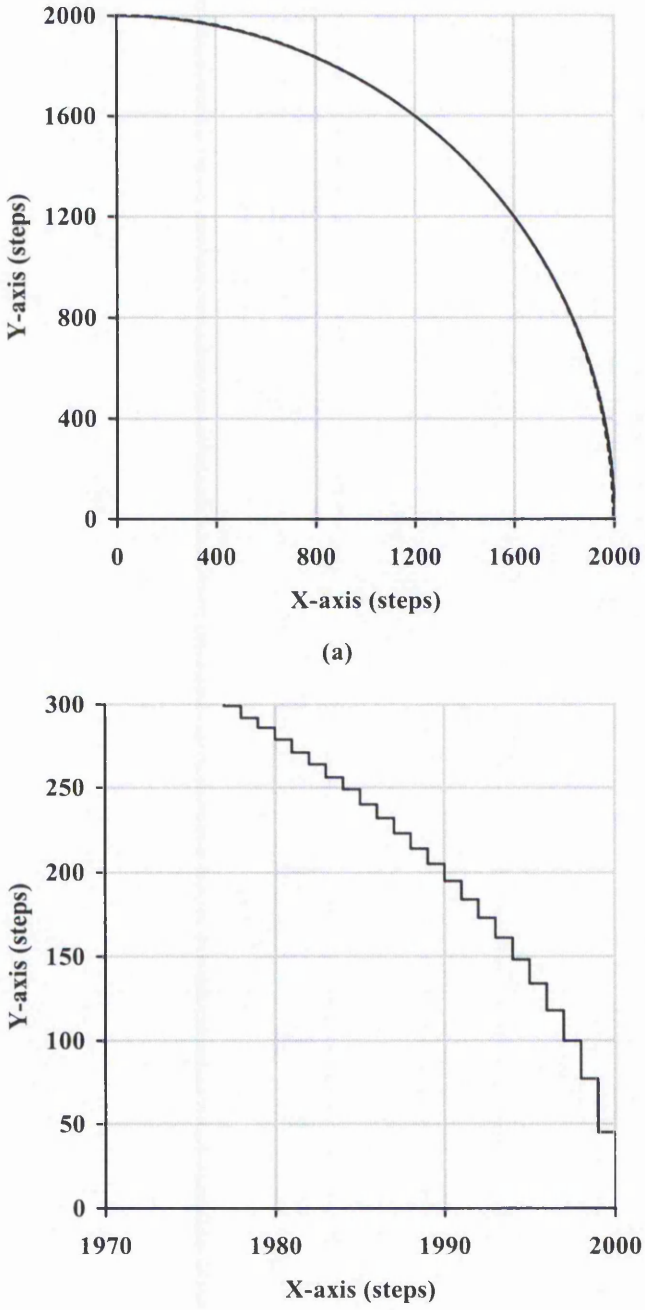
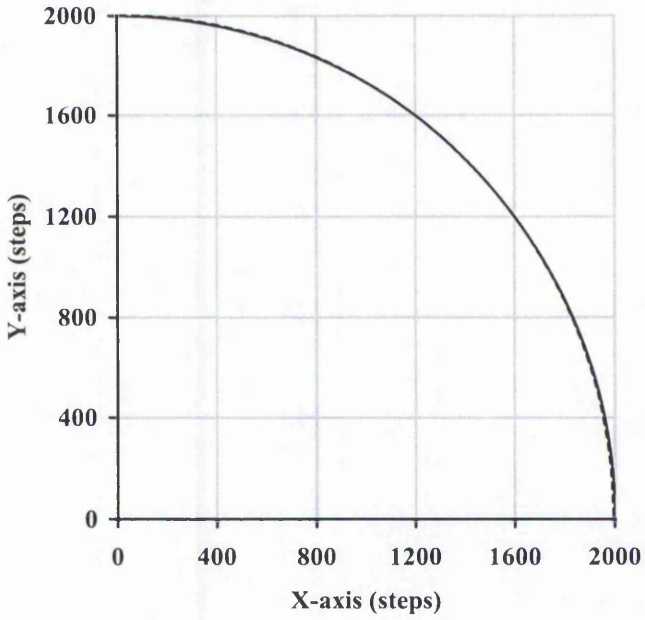
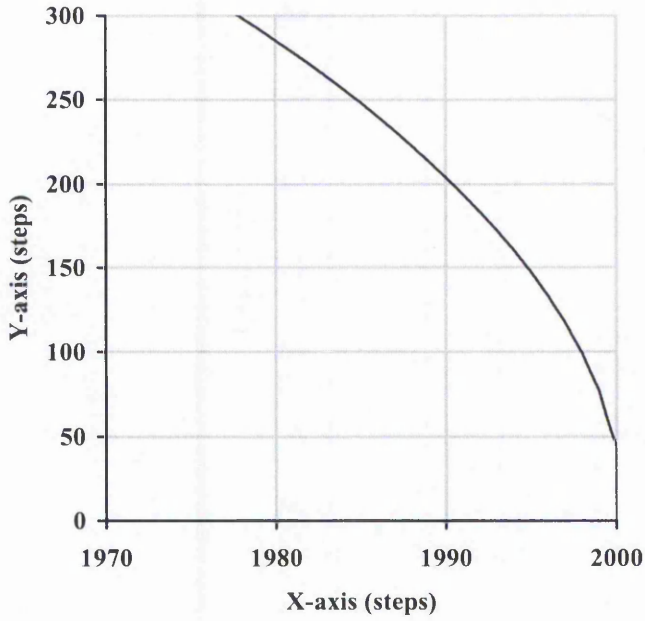


Figure 6-53: Plot of New Half Step Circular Arc Interpolation with Linear Acceleration (Zero Order Simulation). Largest Error = 0.71 (a) Full graph; (b) Detail of path for 300 steps in Y-axis. (Note: X-axis scale is 100 times the Y-axis scale).

show the simulated paths produced when employing the linear acceleration for the arc. In each case the detail has an X-axis scale 100 times the Y-axis scale. Again the Second Order plot becomes smoother as the speed increases. A smooth path is followed after approximately nine X-axis steps.

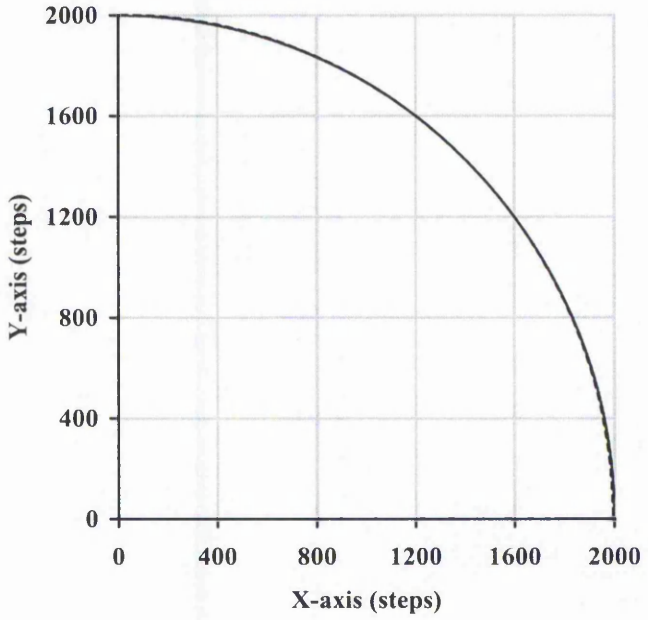


(a)

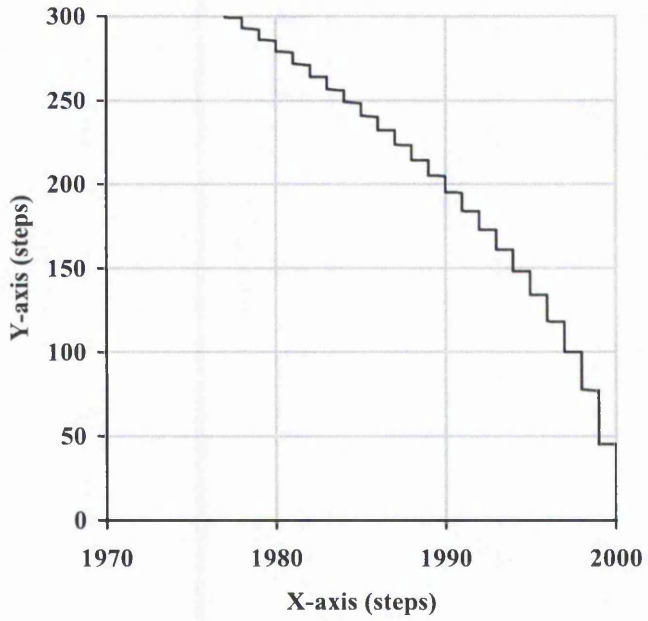


(b)

Figure 6-54: Plot of New Half Step Circular Arc Interpolation with Linear Acceleration (Varying Rate First Order Simulation). Largest Error = 0.59 (a) Full graph; (b) Detail of path for 300 steps in Y-axis. (Note: X-axis scale is 100 times the Y-axis scale).

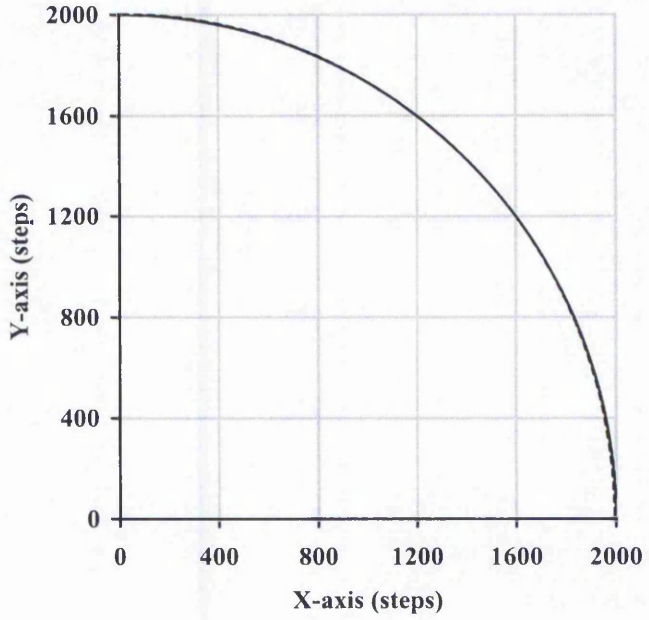


(a)

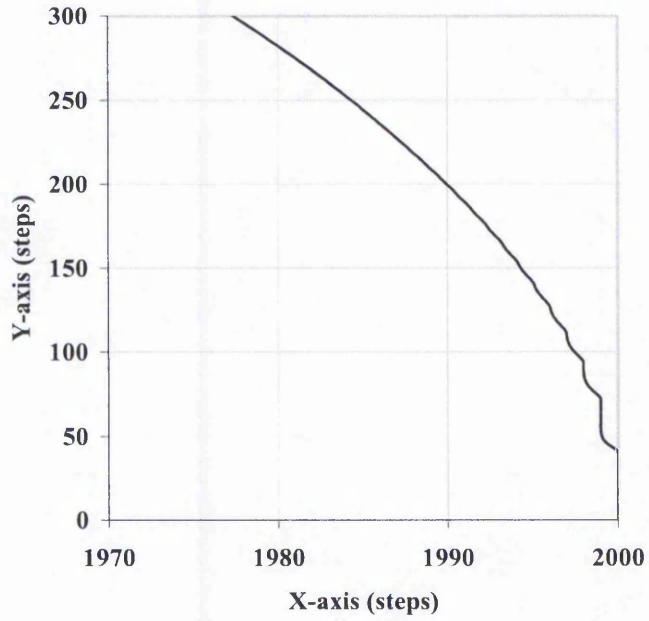


(b)

Figure 6-55: Plot of New Half Step Circular Arc Interpolation with Linear Acceleration (Constant Rate First Order Simulation). Largest Error = 0.51 (a) Full graph; (b) Detail of path for 300 steps in Y-axis. (Note: X-axis scale is 100 times the Y-axis scale).

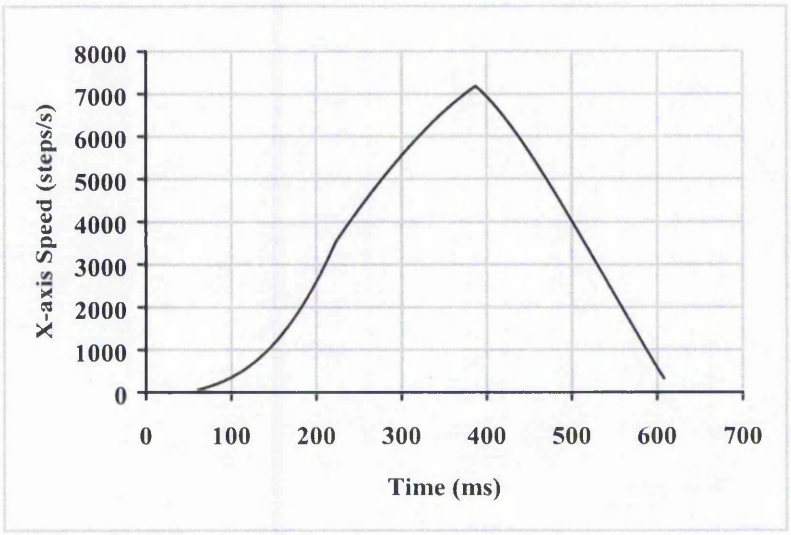


(a)

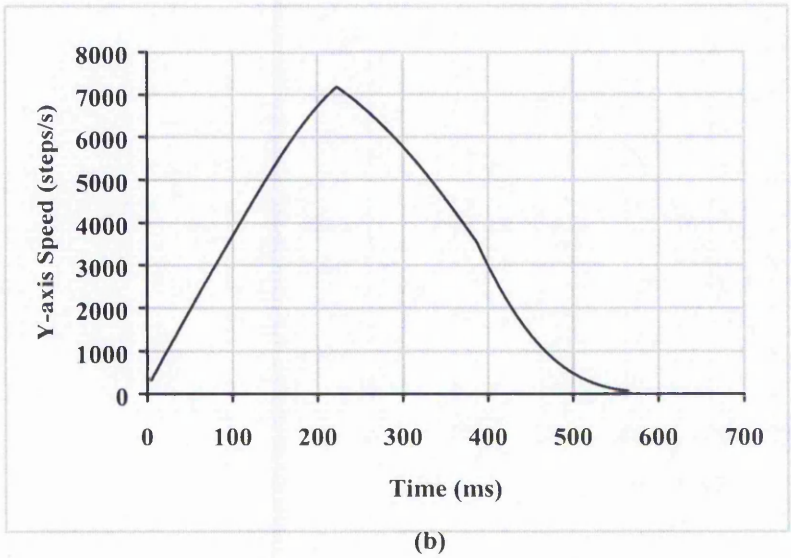


(b)

Figure 6-56: Plot of New Half Step Circular Arc Interpolation with Linear Acceleration (Second Order Simulation). Largest Error = 0.42 (a) Full graph; (b) Detail of path for 300 steps in Y-axis. (Note: X-axis scale is 100 times the Y-axis scale).



(a)



(b)

Figure 6-57: Simulation of Speed with Linear Acceleration and Deceleration for Circular Arc Interpolation for (a) X-axis; (b) Y-axis. (Arc = Anticlockwise 1st Quadrant with radius of 2000 steps. Acceleration = 35000 pulses/s^2. Maximum Resultant Speed = 8000 steps/s. Acceleration stops at 220 ms and deceleration starts at 390 ms).

The speed simulation for the circular arc interpolation with linear acceleration is shown in Figure 6-57. The acceleration and deceleration of the overall curve is linear. However, when the speed for each individual axis is examined, it does not seem to be linear because it follows a cosine/sine curve even without acceleration. The middle part of each axis speed is still part of either a sine or cosine wave (between approximately 220 and 390 ms). Since it is not obvious from the individual axis speed whether the overall speed is maintained, an overall speed graph is plotted in Figure 6-58. The highest speed variation in the constant phase is 2 steps/s or 0.025%.

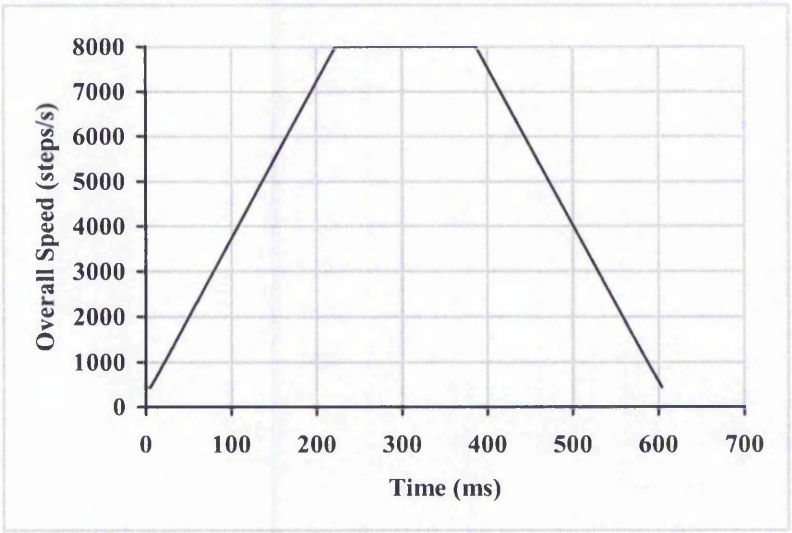
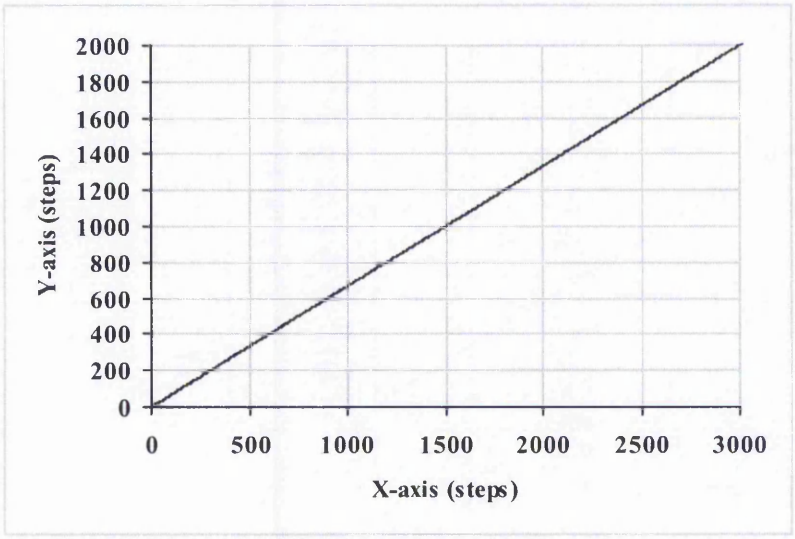


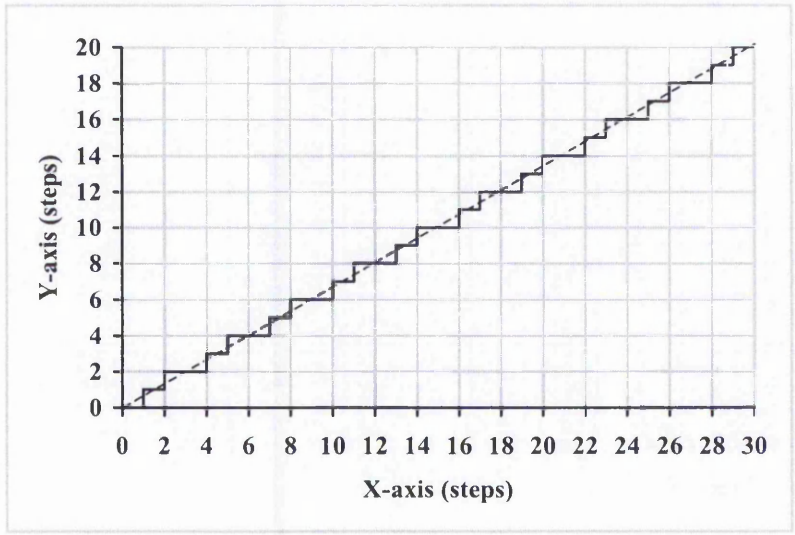
Figure 6-58: Overall Speed of Linear Acceleration on Circular Arc Interpolation.

6.3.2 Simulation Results for Parabolic Acceleration

This section covers the evaluation of the new parabolic acceleration algorithm. Again the error plots are shown in Appendix C. The highest acceleration rate is 104,000 steps/s^2 (1.04 m/s^2 or 0.15 ms acceleration time). Firstly, the simulated path is presented for Zero Order simulation in Figure 6-59. The simulated path is identical to the one generated by the linear acceleration. In fact, when simulated using the Zero Order simulation, the same path should be produced regardless of the type of acceleration or even without acceleration. The same path simulated with the Varying Rate First Order simulation is shown in Figure 6-60. Figure 6-61 demonstrates the Constant Rate First Order simulated path. Figure 6-62 shows the Second Order simulation. The different



(a)

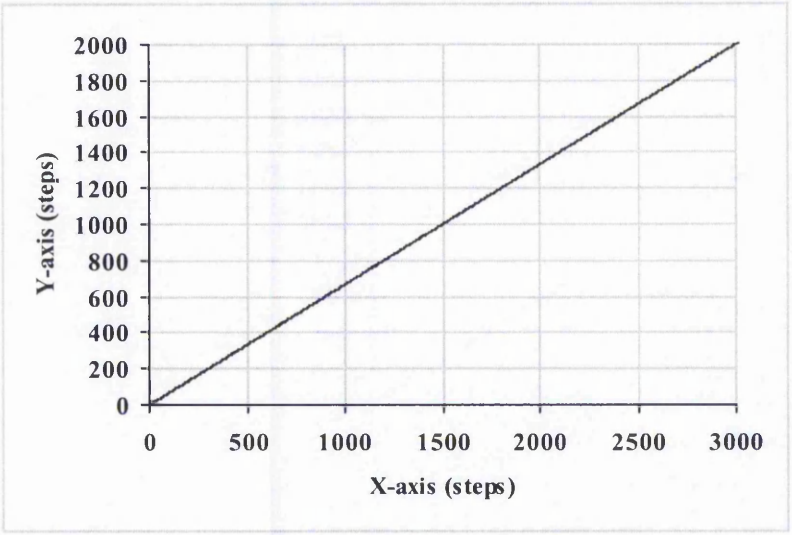


(b)

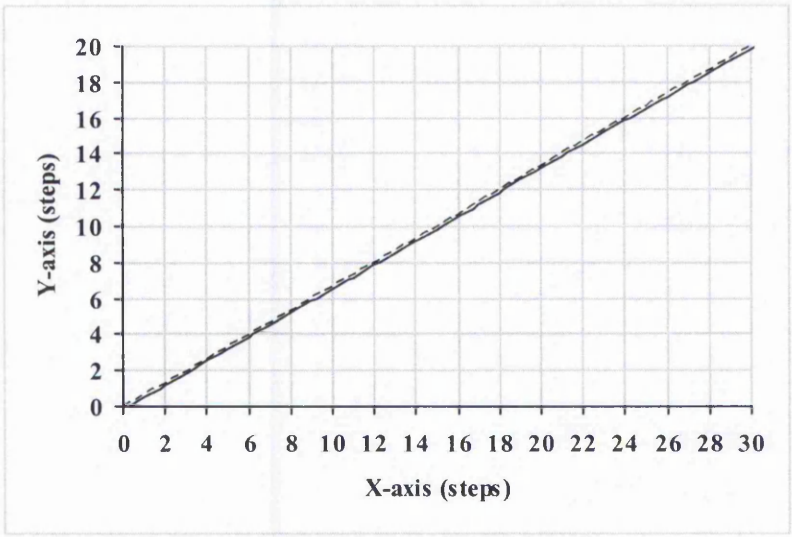
Figure 6-59: Plot of New Half Step Linear Interpolation with Parabolic Acceleration (Zero Order Simulation). Largest Error = 0.55 (a) Full graph; (b) Detail of path for 30 steps in X-axis.

simulated paths for the parabolic acceleration are fairly similar to the ones for the linear acceleration and the largest errors and the average errors are almost the same.

For first or second order simulation, although not identical, the simulated path is very similar when employing either the linear or parabolic acceleration. As in the linear case,



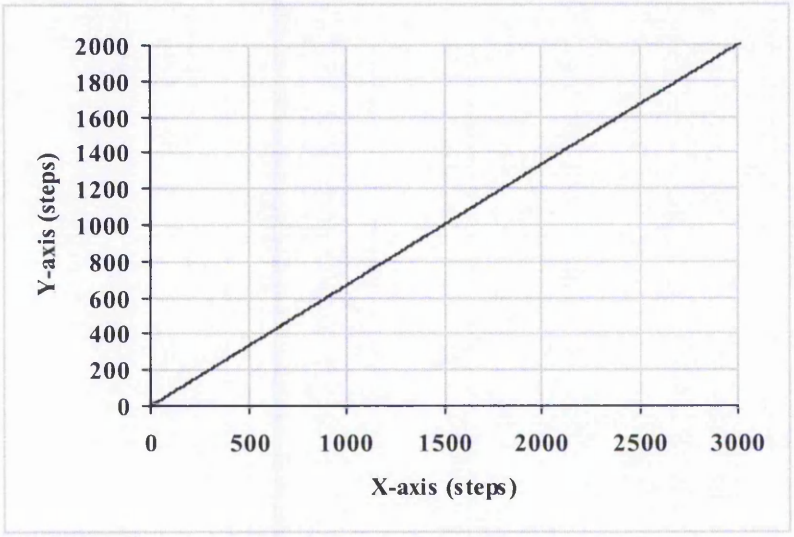
(a)



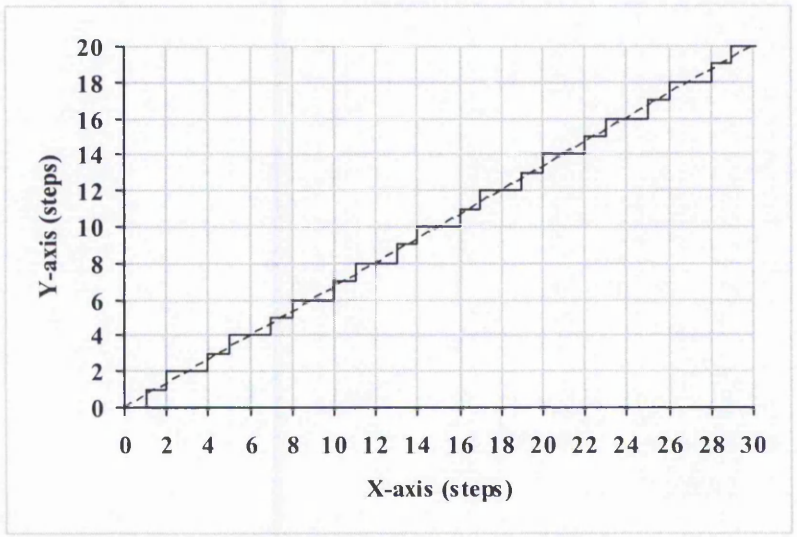
(b)

Figure 6-60: Plot of New Half Step Linear Interpolation with Parabolic Acceleration (Varying Rate First Order Simulation). Largest Error = 0.19 (a) Full graph; (b) Detail of path for 30 steps in X-axis.

the start segment of the Constant Rate First Order simulated path, Figure 6-61(b), is identical to the start segment of the Zero Order simulated path, Figure 6-59(b). This is because the speed in this segment is very low. For the Second Order simulated path (Figure 6-62), the smoothing effect is greater than for linear acceleration because the acceleration rate is higher. The error reduces to a very small value after one step.



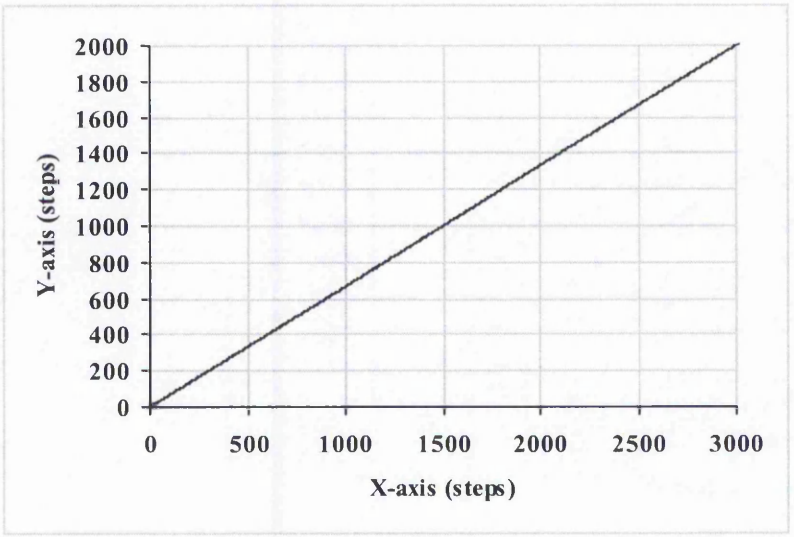
(a)



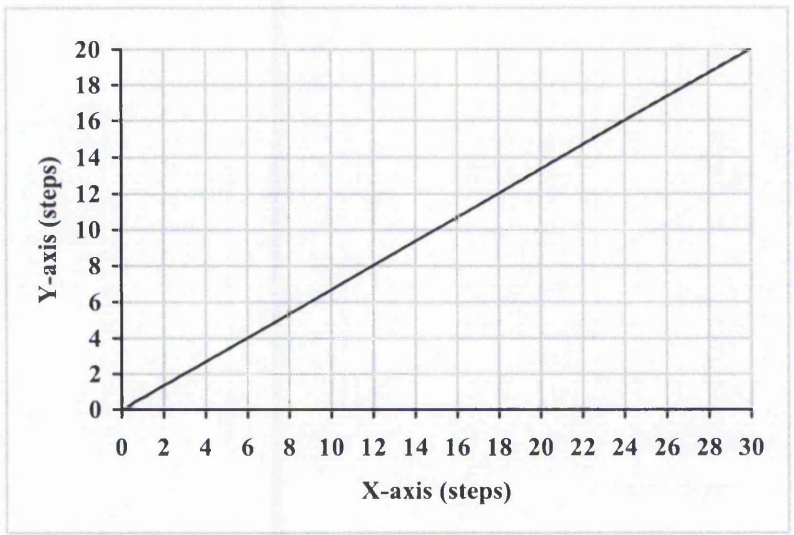
(b)

Figure 6-61: Plot of New Half Step Linear Interpolation with Parabolic Acceleration (Constant Rate First Order Simulation). Largest Error = 0.55 (a) Full graph; (b) Detail of path for 30 steps in X-axis.

The speed simulation for the linear interpolation with parabolic acceleration is shown in Figure 6-63. As expected, the acceleration and deceleration phases of the graph resembles the shape of a parabola. Therefore the overall speed will also have the same shape but scaled up.

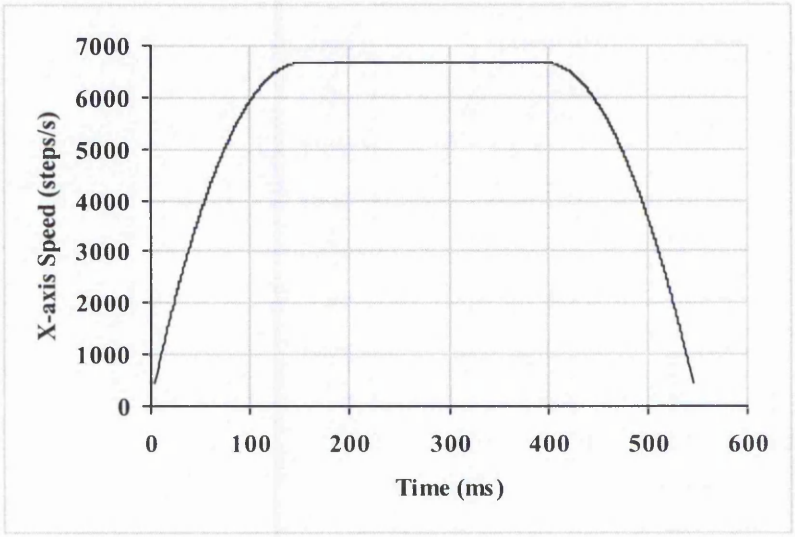


(a)

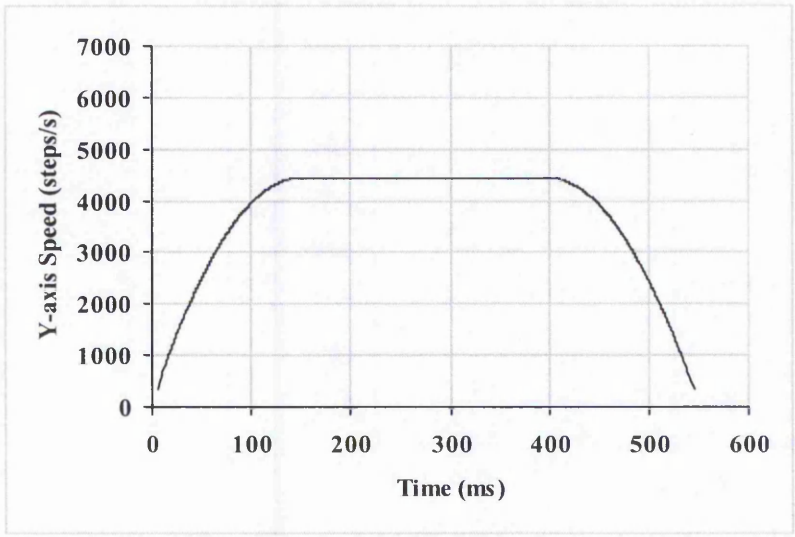


(b)

Figure 6-62: Plot of New Half Step Linear Interpolation with Parabolic Acceleration (Second Order Simulation). Largest Error = 0.07 (a) Full graph; (b) Detail of path for 30 steps in X-axis.



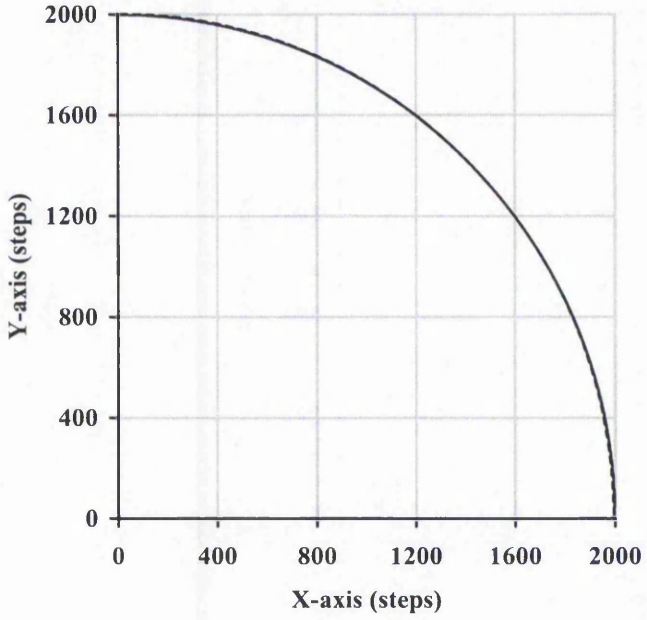
(a)



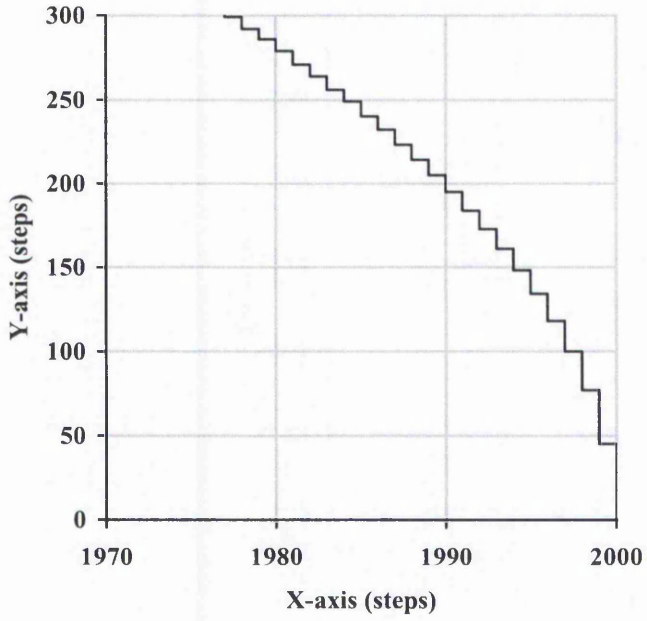
(b)

Figure 6-63: Simulation of Speed with Parabolic Acceleration/Deceleration for Linear Interpolation for (a) X-axis; and (b) Y-axis. (X-axis = 3000 steps, Y-axis = 2000 steps. Maximum Resultant Speed = 8000 steps/s. Maximum acceleration = 104,000 steps/s^2. Acceleration ends at 150 ms, Deceleration starts at approximately 395 ms).

The parabolic acceleration has also been implemented with the circular arc interpolation. Again, the New Half Step Circular Arc interpolation is used for evaluation. The Zero Order simulated path is shown in Figure 6-64 and is identical to the arc for linear acceleration. On the other hand, Figure 6-65 shows the Varying Rate First Order simulated path, which looks very similar to the linear case.

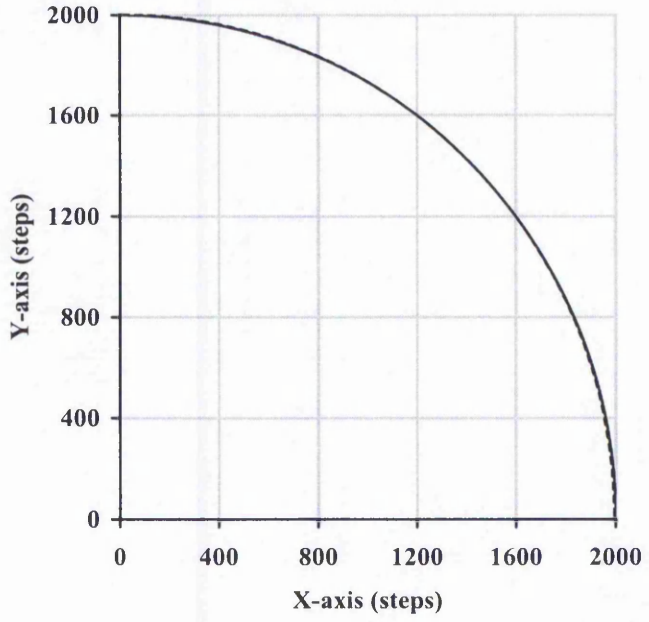


(a)

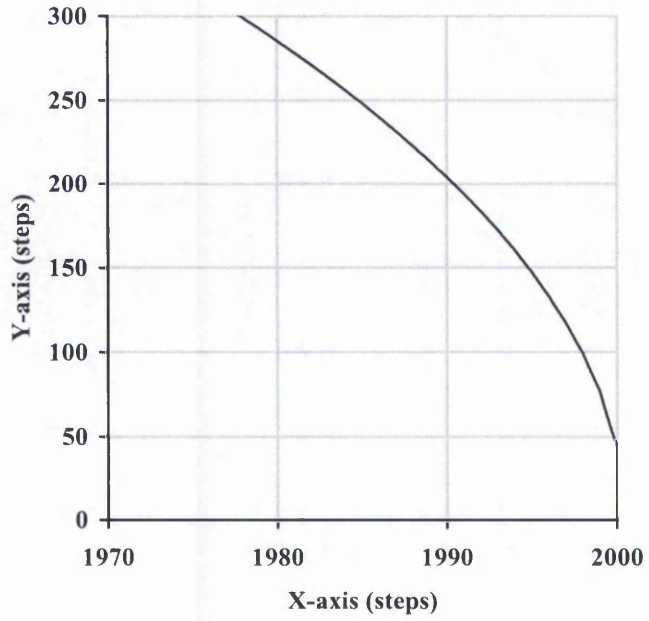


(b)

Figure 6-64: Plot of New Half Step Circular Arc Interpolation with Parabolic Acceleration (Zero Order Simulation). Largest Error = 0.71 (a) Full graph; (b) Detail of path for 300 steps in Y-axis. (Note: X-axis scale is 100 times the Y-axis scale).

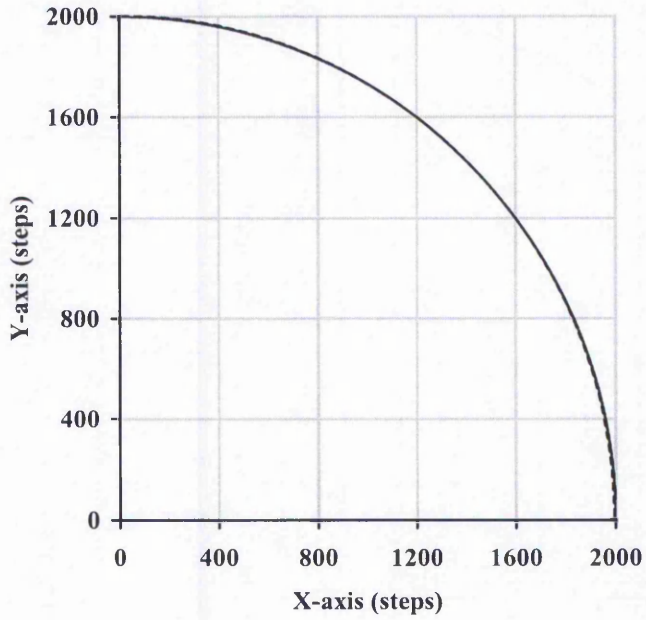


(a)

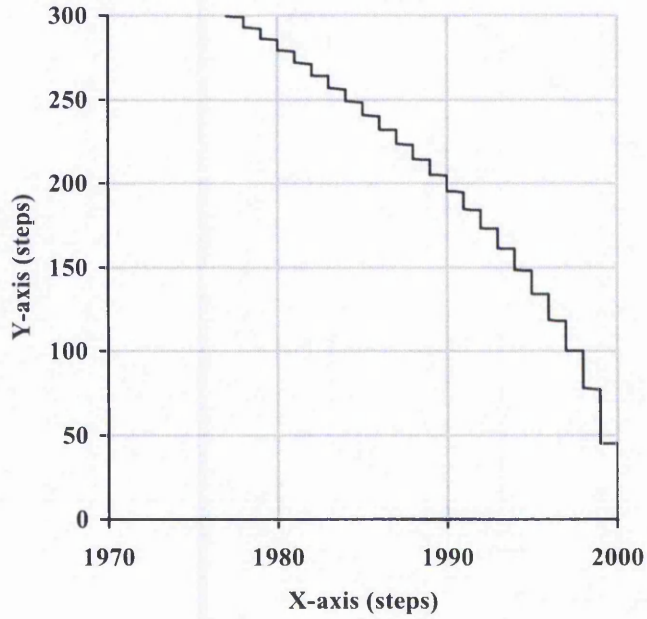


(b)

Figure 6-65: Plot of New Half Step Circular Arc Interpolation with Parabolic Acceleration (Varying Rate First Order Simulation). Largest Error = 0.58 (a) Full graph; (b) Detail of path for 300 steps in Y-axis. (Note: X-axis scale is 100 times the Y-axis scale).



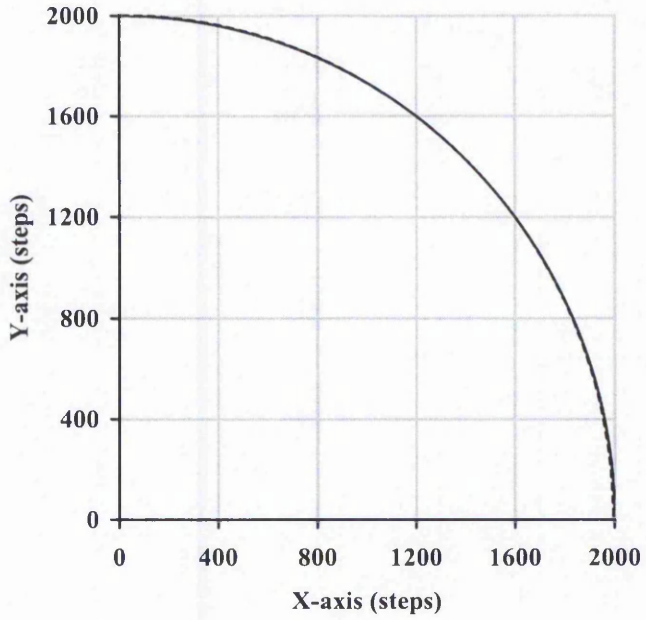
(a)



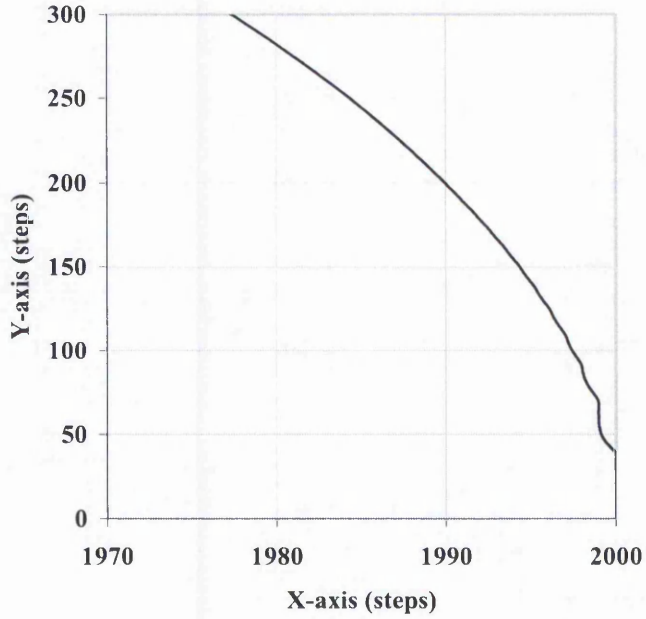
(b)

Figure 6-66: Plot of New Half Step Circular Arc Interpolation with Parabolic Acceleration (Constant Rate First Order Simulation). Largest Error = 0.50 (a) Full graph; (b) Detail of path for 300 steps in Y-axis. (Note: X-axis scale is 100 times the Y-axis scale).

Figure 6-66 shows the simulated path for Constant Rate First Order simulation and the Second Order simulated path is shown in Figure 6-67. Again, it can be seen that the



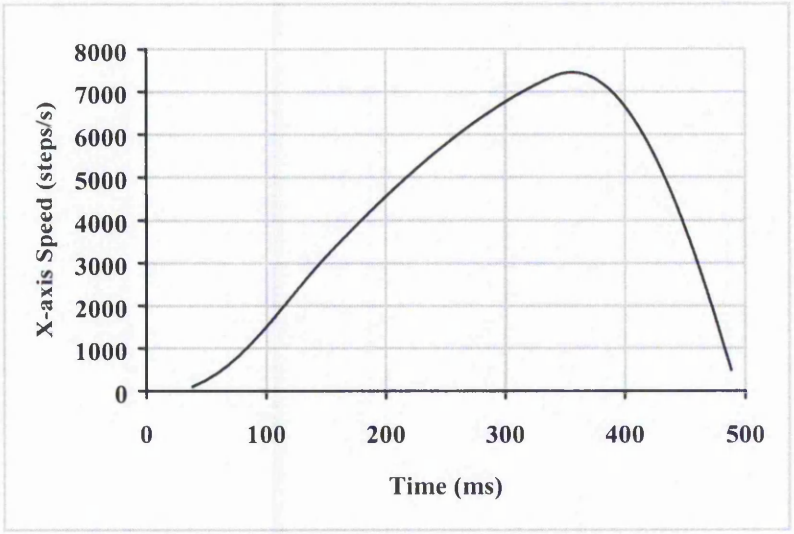
(a)



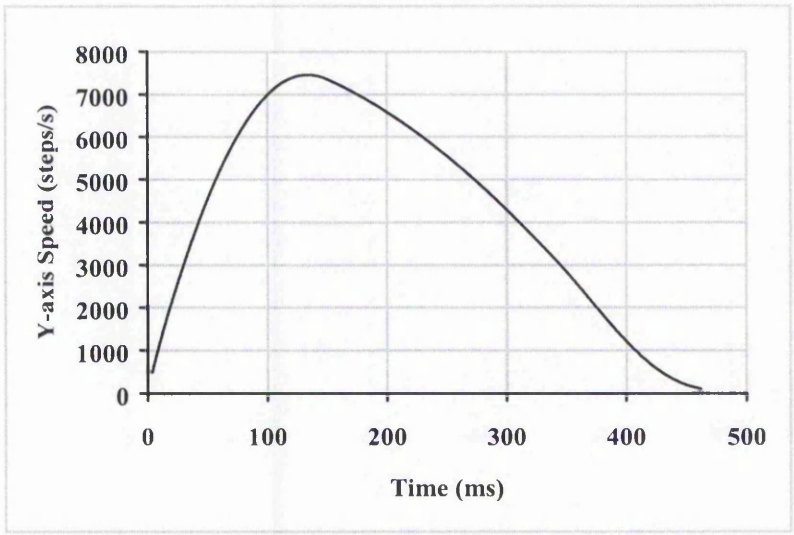
(b)

Figure 6-67: Plot of New Half Step Circular Arc Interpolation with Parabolic Acceleration (Second Order Simulation). Largest Error = 0.37 (a) Full graph; (b) Detail of path for 300 steps in Y-axis. (Note: X-axis scale is 100 times the Y-axis scale).

simulated paths for the parabolic acceleration for circular arc are similar to the paths for linear acceleration. Again, the plot becomes smoother more rapidly than in the case of linear acceleration. It takes about six X-axis steps to maintain at a smooth path.



(a)



(b)

Figure 6-68: Simulation of Speed with Parabolic Acceleration and Deceleration for Circular Interpolation for (a) X-axis; and (b) Y-axis ((Arc = Anticlockwise 1st Quadrant with radius of 2000 steps. Maximum Resultant Speed = 8000 steps/s. Maximum acceleration = 104,000 steps/s^2. Acceleration ends at 150 ms, Deceleration starts at approximately 340 ms).

Figure 6-68 demonstrates the changes of speed throughout the interpolation process when employing parabolic acceleration. Without the acceleration, the X-axis speed follows part of a sine wave and the Y-axis part of a cosine wave. The middle part of each plot, when the resultant speed is constant, is still part of the sine or cosine wave between approximately (150 and 340 ms). For the X-axis, a rather gentle increase in speed is noticed at the beginning while the drop in speed during deceleration is steeper. The reverse is true for the Y-axis speed. The overall speed is illustrated in Figure 6-69 and shows the required behaviour of parabolic acceleration, constant speed and then parabolic deceleration. The highest speed variation in the constant phase is 2 steps/s or 0.025%.

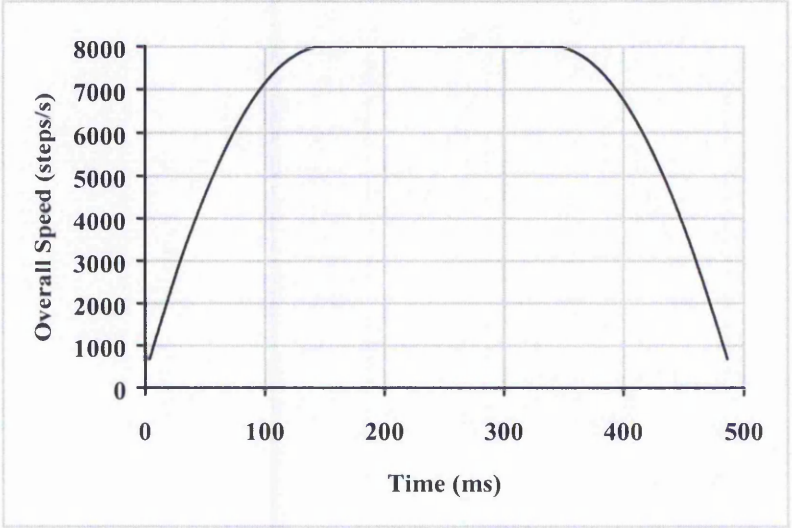


Figure 6-69: Overall Speed of Parabolic Acceleration for Circular Arc Interpolation. The speed is constant from 150 ms to 340 ms.

6.3.3 Discussion

Table 6-5 summarises the largest and average position errors for linear and parabolic acceleration with new half step interpolation. The average error is taken over the signed error where positive in one side of the curve and negative the other.

From this table, it can be seen that the largest position errors and average position errors for the linear and parabolic acceleration are very similar. The position errors throughout

the machining are plotted and shown in Appendix C. They do look similar for both linear and parabolic acceleration. Errors are largest at beginning and end of machining, except with the Zero Order simulation when no difference is expected. The large errors occur when the motion is at low speed and when it is under acceleration. For the circular arc, the errors are high for a longer time at the beginning and end of the motion, because when the motion is almost parallel to one axis, the speed on the other axis is extremely low.

Table 6-5: Largest and Average Position Errors for linear and parabolic acceleration on new half step interpolation.

Simulation Type	Acceleration Algorithm	Line		Circular Arc	
		Largest Error	Average Error	Largest Error	Average Error
Zero Order	Linear	0.55	0.00	0.71	0.00
	Parabolic	0.55	0.00	0.71	0.00
VR First Order	Linear	0.18	-0.14	0.59	0.00
	Parabolic	0.19	-0.14	0.58	0.00
CR First Order	Linear	0.55	0.00	0.51	0.00
	Parabolic	0.55	0.00	0.50	0.00
Second Order	Linear	0.11	0.00	0.42	0.01
	Parabolic	0.07	0.00	0.37	0.01

The main noticeable difference between these two acceleration algorithms is the machining time. The machining time for linear acceleration for line is 665 ms and 610 ms for circular arc, whereas for parabolic acceleration, they are 545 ms and 490 ms, respectively. Thus the times are reduced by 120 ms for the line and arc. The saving in time depends mainly on the relative values of two values, firstly the constant acceleration rate for linear acceleration and secondly the maximum acceleration used at the beginning (and end) of the motion with parabolic acceleration. This depends on the machine used. If the two values are equal, then parabolic acceleration takes longer but normally it is possible to achieve considerably higher acceleration at low speeds (see Section 4.1), whereas the constant acceleration is limited by the maximum acceleration at the required speed. In practice a combination of the two methods could be used to achieve the best time.

Table 6-6 and Table 6-7 summarise the acceleration rate, acceleration time and total machining time when employing different interpolation algorithms on a line and circular arc, respectively. In both cases, the total time for the parabolic acceleration is 120 ms shorter than for the linear acceleration.

Table 6-6: Interpolation for the Line from (0,0) to (3000,2000) at 4.8 m/min using the two different acceleration algorithms.

Acceleration	Acceleration rate (m/s^2)	Acceleration Time (ms)	Total Time (ms)
Linear	0.35	220	665
Parabolic	1.04	150	545

Table 6-7: Interpolation for the Circular Arc from (2000,0) to (0,2000) at 4.8 m/min using the two different acceleration algorithms.

Acceleration	Acceleration rate (m/s^2)	Acceleration Time (ms)	Total Time (ms)
Linear	0.35	220	610
Parabolic	1.04	150	490

In Section 6.4, the results from the practical implementation of the algorithms show that considerably higher accelerations could be achieved with the motors used.

6.4 Initial Tests of the Practical Implementation

A protocol controller has been developed and simple tests of the practical implementation of the new algorithms have been performed on a prototype CNC machine using the same line but an arc of double the radius, 4000 steps, as in Section 6.3. The controller consists of the 'C6711 Digital Signal Processor (DSP), which runs at a maximum speed of 150 MHz. The command pulses are sent out via the onboard timers. The linear and parabolic accelerations have been tested for four different values of maximum speed. The initial testing has not included detailed monitoring using shaft encoders. Instead, the testing has involved determining the highest acceleration rate that the machine is able to achieve for each planned speed without appearing to lose synchronisation. Since no encoders are used, this is only judged by examining the path followed and by repeating it several times. The results for interpolation of a line using linear and parabolic acceleration are summarised in Table 6-8 and Table 6-9 respectively.

For both acceleration algorithms, it can be seen that the achievable acceleration decreases as the planned speed increases. Generally, the total time decreases as the planned speed goes up. However, for the linear case, when the planned speed is 7.5 m/min, the acceleration needs to be so low that the total time is actually longer. If the line was longer, then this would not be the case, because it would spend longer at the planned speed. In all cases, the parabolic acceleration achieves a shorter time than the linear acceleration for the same planned speed.

Table 6-8: Linear Acceleration for Interpolation of the Line from (0,0) to (3000,2000) showing highest acceleration rate achieved for each planned speed.

Planned Speed (m/min)	Highest Acceleration rate (m/s^2)	Acceleration Time (ms)	Total Time (ms)
4.5	2.5	29	509
5.5	2.0	45	435
6.5	1.5	71	400
7.5	0.8	154	437

Table 6-9: Parabolic Acceleration for Interpolation of the Line from (0,0) to (3000,2000) showing highest initial acceleration rate achieved for each planned speed.

Planned Speed (m/min)	Highest Initial Acceleration (m/s^2)	Acceleration Time (ms)	Total Time (ms)
4.5	7.3	20	493
5.5	5.1	35	415
6.5	3.9	55	368
7.5	2.6	95	350

From Table 6-8, the highest acceleration achieved for a given speed for the X-axis can be determined. It is assumed that the X-axis is the limiting factor because it has to move faster. To demonstrate a rough idea of the relationship between the individual axis speed and the achievable acceleration for that speed, a plot of X-axis acceleration against speed is plotted in Figure 6-70. In each case, the limiting factor must be the acceleration at the highest speed used, because a higher acceleration can be achieved for the lower speed values. The shape of this plot resembles somewhat the shape of the curve for torque against speed for a stepper motor (see Figure 2-21). Since acceleration is proportional to torque, the general shape for the two curves would be expected to be similar.

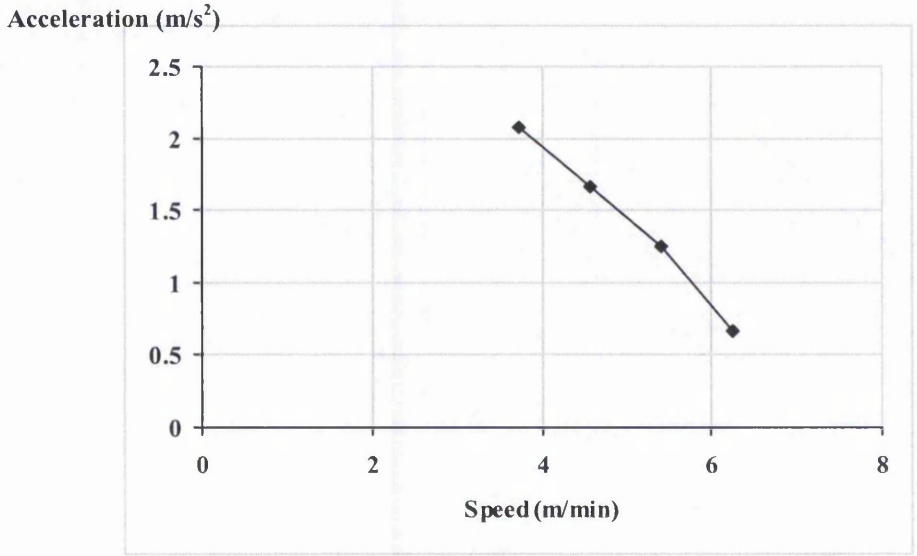


Figure 6-70: Plot of Achievable Acceleration against Speed for the X-axis motor (based on the Results from Table 6-8).

The results for a circular arc with the two methods of acceleration are summarised in Table 6-10 and Table 6-11. A longer arc is used (when compared to the simulation examples) to allow enough time for acceleration up to speed of 6.5 m/min. As with the line, acceleration has to be reduced for higher speeds and the time for machining will decrease, provided the arc is long enough. Even to machine an arc at constant speed requires the motor on an individual axis to accelerate and decelerate to achieve a sine wave (see Figure 2-1). Therefore the acceleration for an individual axis cannot be deduced from the tables and could sometimes be higher than the resultant acceleration. This would explain why the resultant acceleration has to be lower than for the line.

Table 6-10: Linear Acceleration with Interpolation of the Circular Arc from (4000,0) to (0,4000).

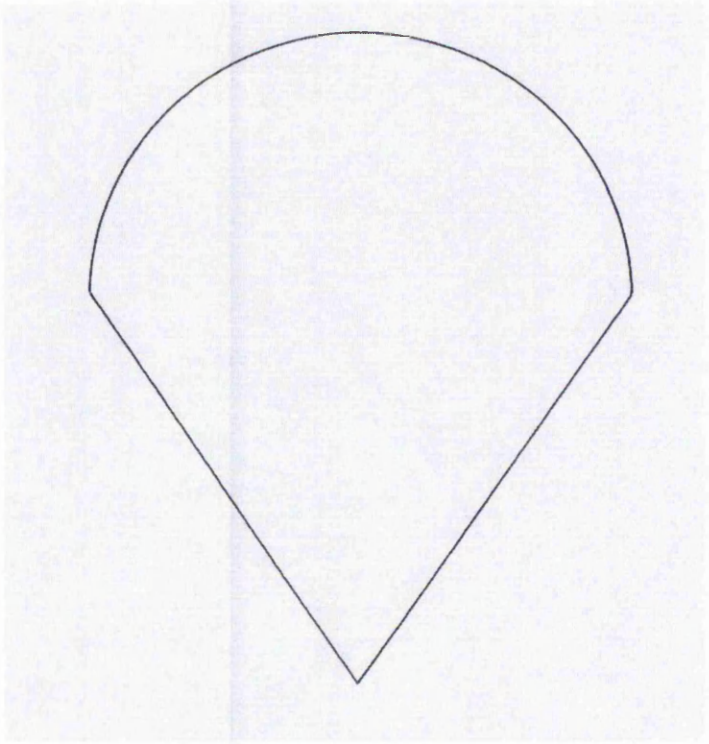
Speed (m/min)	Acceleration (m/s^2)	Acceleration Time (ms)	Total Time (ms)
4.5	2.0	37	872
5.5	0.5	179	858
6.5	0.3	354	925

Table 6-11: Parabolic Acceleration with Interpolation of the Circular Arc from (4000,0) to (0,4000).

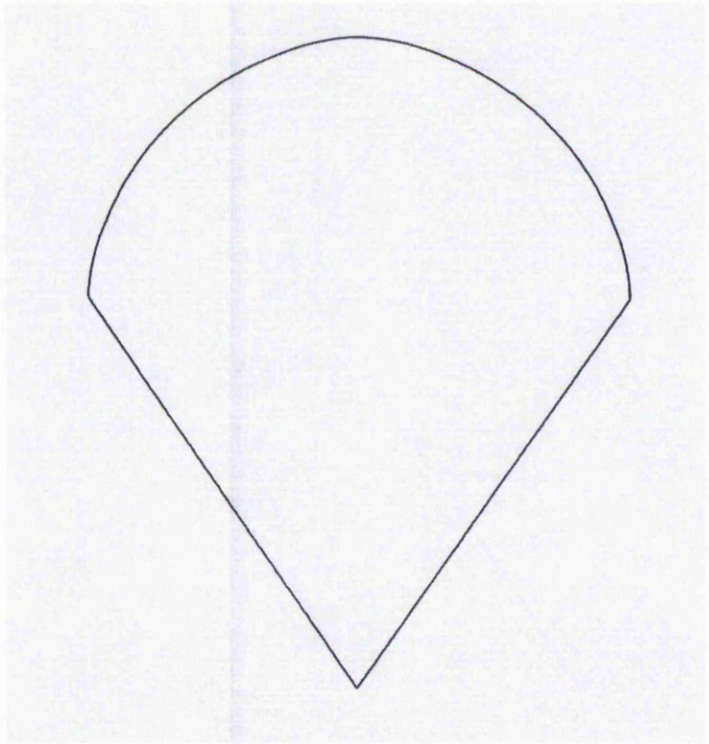
Speed (m/min)	Highest (Initial) Acceleration (m/s^2)	Acceleration Time (ms)	Total Time (ms)
4.5	4.9	30	856
5.5	1.4	130	768
6.5	0.7	300	774

Although the circular arc appears to be able to be machined at the speeds shown in these tables, it has been found that the shape deviates noticeably from the required one for the higher speeds. This was found when a complete shape was used for the tests. It consists of two 90° arcs and two lines, shown in Figure 6-71. Each arc and line is machined by accelerating from rest and then decelerating back to rest. Examples of the path produced by the prototype CNC machine are shown for linear acceleration in Figure 6-71 and for parabolic acceleration in Figure 6-72. Part (a) of each of these figures are running at a lower speed for the circular arc, which seems to be able to follow the arc correctly.

However, for higher speed in part (b), the middle of each arc seems slightly flattened although it appears to be still able to maintain the correct number of steps in X and Y axes. Rough measurements, with a ruler, of the distance in X and Y axes show correct overall dimensions but the radial position of the middle of each arc from its centre has been found to be about 39 mm instead of 40 mm. This would explain the visual effect and requires further, more detailed, investigation.

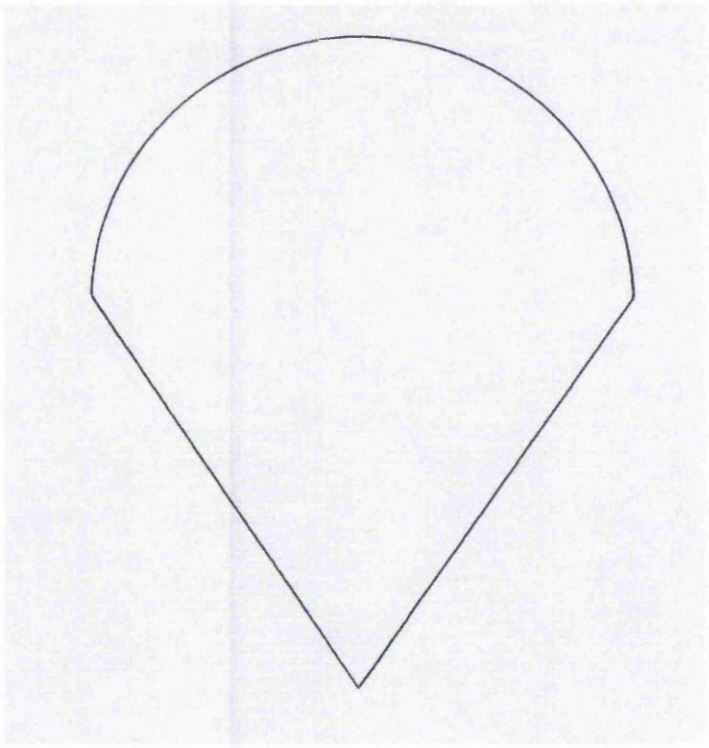


(a)

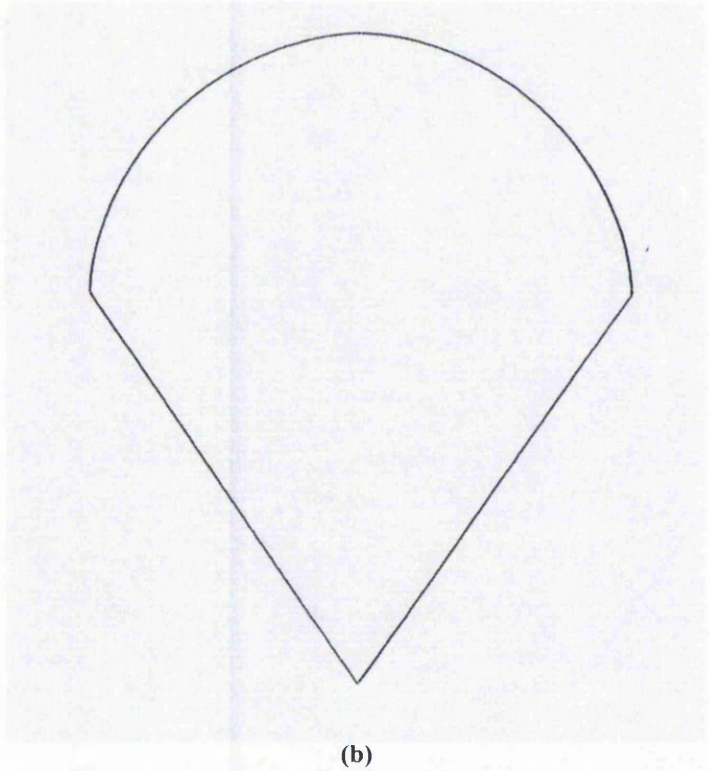


(b)

Figure 6-71: Scanned Image of the Plotted Path Using Linear Acceleration running at 7.5 m/min for the two lines (acceleration = 0.5 m/s^2) and the rate for the two arcs are: (a) 1.5 m/min (acceleration = 5.0 m/s^2; (b) 6.5 m/min (acceleration = 0.4 m/s^2). The shape was plotted four times for the plot in part (b).



(a)



(b)

Figure 6-72: Scanned Image of the Plotted Path Using Parabolic Acceleration running at 7.5 m/min for the two lines (acceleration time = 95 ms) and the rate for the two arcs are: (a) 1.5 m/min (acceleration time = 4 ms); (b) 6.5 m/min (acceleration time = 300 ms).

6.5 Summary

The main aim of this chapter has been to evaluate the new interpolation and acceleration algorithms by demonstrating the simulated path, position errors and speed variation when employing different techniques. Initial results from the practical implementation of the algorithms have also been included.

For evaluation of interpolation, five interpolation algorithms have been used. Three of them are the existing interpolation algorithms. The other two algorithms are the New Full Step and Half Step interpolation algorithms, where the latter is an improvement to the former. Both algorithms have been developed by the Author to produce very smooth motion to maintain a constant speed or changing gradually as required, without any abrupt changes in speed. This is important to minimise the likelihood of any vibrations.

Based on the simulations, the New Half Step interpolation algorithms are a significant improvement on previous algorithms. For smoothness of motion, both the Full and Half Step interpolation algorithms have been shown to be greatly superior to the previous algorithms. When position errors are considered, the New Half Step algorithms are a considerable improvement over the Full Step ones. Generally, based on the simulations, the New Half Step algorithms are expected to have errors no worse and often much smaller than for the previous algorithms. The results from interpolation of a line almost parallel to the X-axis show larger errors. This confirms the practical experience that problems can occur when motion is at an angle very close to one axis, so that one axis is much slower than the other.

Again, based on the simulations, the new acceleration algorithms are able to achieve very smooth motion. The constant phase of the overall speed profile will be similar to the one without acceleration. The speed variations in the acceleration and deceleration phases follow smoothly the required linear or parabolic acceleration. The position errors are somewhat larger than for the interpolation algorithms at the beginning and end of the motion, mainly because of the very low speeds, especially when combined with acceleration and deceleration.

Parabolic acceleration can reduce overall time compared to linear acceleration by exploiting the motor's ability to accelerate faster at slow speeds. The new linear and parabolic acceleration algorithms have been developed for use with the New Half Step interpolation algorithms. However, they can also be used with any interpolation algorithm, provided that it produces pulse timings at constant speed interpolation, which can then be passed to the acceleration algorithm.

An initial evaluation of the practical implementation of the new algorithms has been undertaken on a CNC machine. The results are promising and broadly in agreement with the simulation results.

7 Conclusions and Future Work

7.1 Conclusions

This research has investigated problems with a stepper motor control system caused by interpolation and acceleration algorithms. During multi-axis machining (when interpolation and acceleration planning algorithms are interfaced to the stepper motors), there are three main identified problems. They are vibrations on an individual axis, positional errors and varying resultant speed. These errors are mostly caused by the interpolation and acceleration planning algorithms used and machine vibrations, some of which may result from the algorithms. The work has focused on the development of interpolation algorithms which minimise errors, reduce unnecessary fluctuations in speed on an individual axis and maintain constant overall speed whenever possible.

New interpolation algorithms have been developed which provide a novel approach of calculating the timing for every individual pulse generated. Thus, there will be ultimate control of the pulses so that a smooth stream of pulses can be sent out, reducing the likelihood of machine vibrations. Such calculation-intensive algorithms are made possible with the use of the Digital Signal Processor (DSP). A parameter has been used to synchronise the motion of the different stepper motors and the most appropriate parameter has been found to be distance along the curve. Both linear and circular arc interpolation algorithms have been successfully developed and evaluated against three of the most common previous interpolation algorithms. The simulation results for the Search-Step, DSM and DDA algorithms were compared with the results for the new algorithms. There are two different types of interpolation algorithms, the Full Step algorithm and the Half Step algorithm, where the latter are an improvement on the former. Both these algorithms are able to produce much smoother sequences of pulses than the previous algorithms under simulation. The New Half Step interpolation algorithms have errors mainly better than the previous algorithms and they show significant reduction in positional errors over the Full Step algorithm. In addition, the likelihood of X-axis and Y-axis speed fluctuations has been greatly diminished, thus

reducing the chance of vibrations. The overall speed is close to constant during interpolation.

In practice, the stepper motors are unable to meet the high speed demands instantly. Instead, the motor needs to start instantly at the (low) pull-in speed and then accelerate to the required speed. To achieve acceleration with the newly developed interpolation algorithms, new acceleration algorithms have been developed. Both linear and parabolic acceleration algorithms have been developed, based on previous acceleration algorithms. Using an appropriately chosen parameter for the parabolic acceleration, it is possible for the machining time to be shorter than for the linear counterpart. This is done by exploiting the fact that stepper motors are capable of high acceleration at low speeds but lower acceleration as the speed increases.

The evaluation of the new interpolation and acceleration algorithms has included simulation of the motion generated using four different partial simulation methods. Two main criteria have been chosen for evaluation, position errors and speed variation. The Zero Order simulation was useful for the general path but not reliable for detail. The First Order and Second Order simulations all had useful aspects and broadly agree that the Half Step algorithm is the best.

Simulation examples for the new acceleration algorithms with the new interpolation algorithms have been evaluated. The two new acceleration algorithms, linear and parabolic acceleration, have been shown to be expected to allow the desired path to be followed very closely. The speed simulations show that the speed on an individual axis is able to maintain gradual changes in speed as required for both geometry and acceleration. Overall speed is close to constant except during acceleration and deceleration when it increases or decreases gradually.

Based on the simulation results, all the objectives have been achieved in the case of lines and circular arcs. The practical implementation has undergone initial simple testing and the results are promising. It is expected that the new approach can be extended to more complex curves, such as splines. From the experiments with different speeds, it is very likely that the new algorithms will allow improved high speed

interpolation. However, further investigations are required using instrumentation with encoders to measure the errors more accurately.

7.2 Future Work

The new interpolation and acceleration algorithms have been developed and successfully evaluated on a simulation platform. The new algorithms are able to improve on the Search-Step, DSM and DDA algorithms for linear and circular-arc interpolation.

The testing of the practical implementation should be extended to include more precise measurements. The actual response of the motors can then be evaluated more thoroughly using motion encoders mounted on the motor shafts. From the readings of these motion encoders, the actual motion of the motor can be deduced to calculate the position errors. The massive calculation capability of the DSP enables it to perform the required calculation for both new linear and circular arc interpolation as well as acceleration. A comparison of the two new acceleration algorithms should be carried out, including whether vibrations are caused by the sudden change at the end of the linear acceleration.

As described in previous chapters, one of the challenges in the field of motion control is to be able to machine complex paths. Different interpolation methods for stepper motors are available to machine such curves. However, most of these methods still suffer from failure to maintain constant speed along the desired cutting path. In addition, most of them involve approximating of the complex path initially with short straight lines and circular arcs and ultimately with short facets. Therefore, the new approach to interpolation should be extended to interpolation of more complex curves, such as splines. One problem in designing such interpolation is to deal with parametrisation. The new line and arc interpolation algorithms use distance as the parameter. However, Farin [61] has shown that this is not possible with a polynomial spline unless the degree is 1 (ie. a straight line). The closest match to distance parametrisation for splines is the chord length parametrisation and distance along the curve can be calculated

numerically. Again it must be ensured that there are no abrupt changes of speed in any individual axis. In other words, the axis speed must only change gradually. This will require further investigation, in order to generate a smooth stream of pulses.

Previously, for both stepper motors and servomotors, the interpolation has taken place at predefined distance or time intervals. The end of such an interval might not coincide with a whole X or Y-axis step. The new interpolation algorithms consider the different axes separately, and not bound by the distance or time intervals. Because of the open loop architecture of the investigated system, it is feasible to make sure that the pulses are generated individually. In other words, each pulse is sent at a separate time. To extend this idea to a closed loop servomotor system, it is worth investigating the possibility of sending a block of X or Y pulses at a time. However, the new idea is that, the block of X and Y pulses are not synchronised at the same distance or time intervals but they should be dealt with individually.

The deceleration used throughout this research is symmetrical to the acceleration. However, practical experience shows that a motor can be made to decelerate faster when compared to the acceleration. This is because friction slows down acceleration but helps with deceleration. Therefore, further investigation on the deceleration should be performed, by either having an inverse parabolic acceleration or having a shorter deceleration time. It may also be possible to improve acceleration by using a different acceleration algorithm designed for each motor.

References

[1] Jon Bodsworth
The Egypt Archieve
Web: http://www.egyptarchive.co.uk/

[2] Frank Nanfara, Tony Uccelo, Derek Murphy,
"The CNC Workshop: A Multimedia Introduction to Computer Numerical Control",
Addison-Wesley, 1999

[3] Motion Control System,
Catalog 8000-3/USA,
Compumotor, Parker Automation

[4] Steiger W.,
"Smooth Continuous Path Motion Generation for Stepping Motor Driven Manipulators",
PhD Thesis, Department of Mathematics and Computing,
The Nottingham Trent University,
Nottingham, England, 1994

[5] Stout A.J.,
"Instrumentation Techniques and Improved Control of Stepper Motor Driven Machinery",
PhD Thesis, Department of Computing and Mathematics,
The Nottingham Trent University,
Nottingham, England, 1999

[6] Jon Stenerson, Kelly Curran,
"Computer Numerical Control Operation and Programming",
Prentice Hall, 1996

[7] Jonathan Lin S.C.,
"Computer Numerical Control: From Programming to Networking",
Delmar Publishers Inc.

[8] Steiger W., Sherkat N., Thomas P.D.,
"Smooth Continuous Path Motion Generation for Stepping Motors",
Proceedings of 3rd IFAC/IFIP Workshop on Algorithms and Architectures for Real Time Control, Ostend,
pp 543-547, 31 May – 1 June 1995

[9] Bailey S.J.
"Servomotor and Stepper: Key Elements of Motion Control Design",
Control Engineering,
pp 55 – 59, Feb 1986

[10] Bailey S.J.,
"Steppers and Servomotors Offer Designers Rich Motion Control Option",
Control Engineering,
pp 81 – 85, May 1988

[11] Renton D., Elbestawi M.A.,
"High Speed Servo Control of Multi-Axis Machine Tools",
International Journal of Machine Tools and Manufacture,
March 2000, Vol. 40(3), pp 539 - 559

[12] Pacer Systems Ltd
Gauntley Street,
Nottingham NG7 5HF
Web: http://www.pacersys.co.uk/

[13] Axiomatic Technology Ltd
Graphic House, Noel Street, Kimberley,
Nottingham NG16 2NE
Web: http://www.axitech.co.uk/

[14] Koelsch J.R.,
"Use the Beam for Better Cutting",
Manufacturing Engineering,
pp 51 – 55, Jan 1991

[15] Koren Y.,
"Control of Machine Tools",
Journal of Manufacturing Science and Engineering,
Vol. 119, pp 749 – 755,
November 1997

[16] Lim F.S., Wong Y.S., Rahman M.,
"Design and Development of a PC-Based CNC System Using LSI chips",
Journal of the Institution of Engineers,
Vol. 30, No. 1, pp 63-69,
Jan/Feb 1990

[17] Stout A.J., Orton P.A., Thomas P.D.,
"Stepper Motor Control System Analysis and Design",
Proceedings of the Circuits, Systems and Computers International Conference,
Hellenic Naval Academy,
Piraeus, Greece, 1996

[18] Rong-Shine Lin,
"Real-time Surface Interpolator for 3-D Parametric Surface Machining on 3 Axis Machine Tools,"
International Journal of Machine Tools and Manufacture,
Vol. 40, No. 10, pp 1513 – 1526, 2000

[19] Steiger W., Sherkat N.,
"Smooth Continuous Path Motion Through Variable Pulse Control",
ASME Joint European Conference on Engineering Systems, Design and Analysis, London, UK, 1994.

[20] Stout A.J., Orton P.A., Thomas P.D.,
"Improving Continuous Path Motion Using Stepper Motors",
Proceedings of the 10th European Simulation Symposium & Exhibition,
The Nottingham Trent University, Nottingham, UK, 1998

[21] Koren Y.,
"Computer Control of Manufacturing Systems",
Mc-Graw-Hill International Student Edition, 1983.

[22] Mansor W.,
"Microcontroller-Based Intelligent System for Motion Control",
IEEE Region 10 International Conference on Electrical and Electronic Technology,
Vol. 1 and 2, pp 439 – 441, 2001

[23] Kenjo Takashi, Sugawara Akira,
"Stepping Motors and their Microprocessor Controls",
Oxford University Press, 1994

[24] Bell R., Lowth A.C., Shelley R.B.,
"The Application of Stepping Motors to Machine Tools",
Brighton: Machining Publishing, 1970

[25] Pei X., Zheng J.J., Ruan J.,
"Study of High Speed Positioning System Actuated by Stepper Motor",
Advances In Abrasive Processes,
pp 469 – 474, Jun 02 – 06, 2001

[26] Gao HY, Cheng SK, Sun L, Kang EL,
"Maximum Torque/Current Control of 2-Phase Hybrid Stepping Motor",
IEEE IEMDC '03: IEEE International Electric Machines and Drives Conference,
Vols 1 – 3, pp 1781 – 1786, 2003

[27] Carrica D, Funes MA, Gonzalez SA,
"Novel Stepper Motor Controller Based on FPGA Hardware Implementation",
IEEE-ASME Transactions on Mechatronics,
Vol. 8, Part 1, pp 120 – 124, March 2003

[28] Wale J.D., Pollock C.,
"Hybrid Stepping Motors & Drives",
Power Engineering Journal,
Vol. 15, Part 1, pp 5 – 12, 2001

[29] Acarnley P.P.,
"Stepping Motors: A Guide to Modern Theory and Practice",
IEE Control Engineering Series 19, Second Edition, Peter Peregrinus Ltd, 1984.

[30] Belanger P.R.,
"Estimation of Angular Velocity and Acceleration from Shaft Encoder Measurements"
IEEE International Conference on Robotics and Automation,
May 1992

[31] Adams KG, Vanreenen M,
"A Low-Cost Stepper Motor Positioning System with Minor Closed-Loop Control"
International Journal of Advanced Manufacturing Technology,
Vol 10, No. 3, pp 191 – 197, 1995

[32] Chai OH, Wong YS, Poo AN,
"An Interpolation Scheme for Tool-Radius Compensated Parabolic Paths for CNC",
IIE Transactions,
Vol. 28, Part 1, 11 – 17 Jan 1996

[33] James V. Valentino, Joseph Goldenberg,
"Introduction to Computer Numerical Control – Second Edition",
Prentice Hall, 2000.

[34] El-Mounayri H, Kishawy H, Tandon V,
"Optimized CNC End-Milling: A Practical Approach",
International Journal of Computer Integrated Manufacturing,
Vol 15, Part 5, pp 453 – 470, Sep – Oct 2002

[35] Chih-Ching Lo,
"A New Approach to CNC Tool Path Generation",
Computer Aided Design,
Vol. 30, No. 8, pp 649 – 655, 1998

[36] Prasad B.S.V., Vidyasagar M.V.V., Gururaja G.U.,
"Design and Development of Control Software for a PC Based DNC Controller",
Computers In Industry,
Vol. 12, pp 329 – 334, 1989

[37] Kim D.I.,
"Study on Interpolation Algorithms of CNC Machine Tools",
IEEE Industry Applications Conference 1995,
Vol. 3, pp 1930 – 1937, 1995

[38] Fleisig R.V., Spence A.D.,
"A Constant Feed and Reduced Angular Acceleration Interpolation Algorithm for Multi-Axis Machining",
Computer Aided Design,
Vol. 33, pp 1 – 15, 2001

[39] Glenn A. Graham,
"Encyclopedia of Industrial Automation",
Long Scientific & Technical, 1988, ISBN 0-582-03566-X

[40] Papaioannou Spiros G.,
"Interpolation Algorithms for Numerical Control",
Computers In Industry, 1979, No. 1, pp 27-40

[41] Koren Y., Masory O.,
"Reference-Pulse Circular Interpolators for CNC Systems",
Transactions of ASME Journal of Engineering for Industry,
Vol. 103, No. 1, pp 131 – 136, 1981

[42] Koren Y., Masory O.,
"Reference-Word Circular Interpolators for CNC Systems",
Transactions of ASME Journal of Engineering for Industry,
Vol. 104, pp 400 – 405, 1982

[43] Lim F.S., Wong Y.S., Rahman M.,
"Circular Interpolators for Numerical Control: A Comparison of the Modified DDA Techniques and an LSI Interpolator",
Computers In Industry,
Vol. 18, No. 1, pp 41 – 52, 1992

[44] Chai O.H., Wong Y.S., Poo A.N.,
"A DDA Parabolic Interpolator for Computer Numerical Control of Machine Tools",
Mechatronics,
Vol. 4, No. 7, pp 673 – 692, 1994

[45] Bollinger JG, Duffie NA,
"Computer Control of Machines and Processes",
Addison-Wesley Publishing Company,
1988

[46] Angelov A.S., Mihailov A.D., Nachev G.N.,
"Simple Algorithm for Curvilinear Interpolation",
Proceedings of 11th Annual Meeting and Technical Conference Numerical Control Society,
pp 338 – 348, March 31 – April 3, 1974

[47] Goldberg Ken, Goldberg Melvin,
"X-Y Interpolation Algorithms",
Robotics Age,
5(3), pp 22 – 25, May 1983

[48] Pak H.A., Daneshmend L.K.,
"Microprocessor Control of Multiaxis Continuous Path Motion",
Microprocessors and Microsystems,
Vol. 6, No. 10, pp 519 – 527, December 1982

[49] Massory O., Koren Y.,
"The Direct-Search Method In CNC Interpolators",
ASME Winter Annual Meeting (Production Engineering Division),
Francisco, Calf., pp 2 – 8, 1978

[50] LHO J-J., NA S-J.,
"A Study on an Improved Direct-Search Interpolation Method for Two Axis CNC Thermal Cutting Systems",
Proceedings Institution Mechanical Engineers,
Vol. 206, No. B1, pp 67 – 76, 1992

[51] Bresenham Jack,
"Algorithms for Computer Numerical Control of A Digital Plotter",
IBM Systems Journal, 1965, Vol. 4, No. 1, pp 25-30

[52] Liu XW, Cheng K,
"Three-Dimensional Extension of Bresenham's Algorithm and Its Application in Straight-Line Interpolation",
Proceedings of the Institution of Mechanical Engineers Part B- Journal of Engineering Manufacture,
Vol. 216, Part 3, pp 459 – 463, 2002

[53] Yang DCH, Kong T,
"Parametric Interpolator versus Linear Interpolator for precision CNC Machining",
Computer Aided Design 1994, Vol. 26(3), pp 225-233

[54] Kiritsis D.,
"High Precision Interpolation Algorithm for 3D Parametric Curve Generation",
Computer Aided Design,
Vol. 26(11), pp 850 - 856

[55] Chou J-J., Yang DCH,
"Command Generation for Three-Axis CNC Machining",
Proceedings of Advances in Manufacturing System Engineering, ASME Winter Annual Meeting, San Fransisco, CA, USA, pp 29-37, 1989

[56] Yeh S.S. and Hsu P.L.,
"The Speed-Controlled Interpolator for Machining Parametric Curves",
Computer Aided Design, 30 April 1999, Vol. 31, No. 5, pp 349-357

[57] Huang JT, Yang DCH,
"A Generalised Interpolator for Command Generation of Parametric Curves In Computer-Controlled Machines",
ASME 1992 Japan-USA Symp. Flexible Automation, Vol. 1, pp 393-399

[58] Shpitalni M, Koren Y, Lo CC,
"Real-Time Curve Interpolators",
Computer Aided Design 1994, Vol. 26, pp 832-838

[59] Lo CC,
"Feedback Interpolators for CNC Machine Tools",
ASME Journal of Manufacturing Science and Engineering 1997, 119, pp 589-592

[60] Poliakoff J.F.,
"The Digital Representation of Two-Dimensional Cutter Paths",
PhD Thesis, Department of Computing,
The Nottingham Trent University, September 1992

[61] Gerald Farin,
"Curves and Surfaces for Computer Aided Geometric Design: A Practical Guide",
Boston: London: Academic, 1993

[62] Bartels R.H., Beatty J.C., Barsky B.A.,
"An Introduction to Splines for Use in Computer Graphics and Geometric Modelling",
Morgan Kaufmann, 1987

[63] Ali I., Bedi S.,
"NC Controller for Surface Machining",
International Journal of Advanced Manufacturing Technology,
Vol. 7, pp 150 – 158, 1992

[64] Ishida J,
"The General B-Spline Interpolation Method and Its Application to the Modification of Curves and Surfaces",
Computer Aided Design,
Vol. 29, Part 11, pp 779 – 790, Nov 1997

[65] Bedi S., Ali I. and Quan N.,
"Advanced Interpolation Techniques for NC Machines",
Journal of Engineering for Industry, August 1993, Vol. 115, pp 329-336

[66] Bedi S. and Quan N.,
"Spline Interpolation Technique for NC Machines",
Computers In Industry, 1992, Vol. 18, No. 3, pp 307-313

[67] Jeon J.W.,
"A Generalized Approach for the Acceleration and Deceleration of CNC Machine Tools",
Proceedings of the 1996 IEEE IECON – 22nd International Conference on Industrial Electronics, Control, and Instrumentation,
Vol. 1-3, pp 1283 – 1288, 1996

[68] Dong-IL Kim, Jae Wook Jeon, Sungkwun Kim,
"Software Acceleration/Deceleration Methods for Industrial Robots and CNC Machine Tools",
Mechatronics, Vol. 4, No. 1, pp 37 – 53, 1994

[69] Chin TC and Mital DP,
"A 2-axis Stepper Motor Motion Control System",
1991 International Conference on Industrial Electronics, Control and Instrumentation",
Oct 28 – Nov 01, 1991, IECON 91, Vol 1-3, pp 397-401

[70] Mital D.P., Chai C.T., Myint T.,
"A Precision Stepper Motor Controller for Robotic Applications"
IEEE Industry Applications Society Annual Meeting,
Pts 1 – 2, pp 686 – 691, 1989

[71] Lee T., Chin T.C., Mital D.P.,
"A 2-Axis Stepper Motor Motion Control System with Adaptive Capability",
Proceedings of the 1992 International Conference on Industrial Electronics, Control, Instrumentation and Automation,
Vol. 1 – 3, pp 442 – 446, 1992

[72] Palmin S. and Shlain V.,
"Stepper Motor Controller with Parabolic Velocity Profile Allows Maximum Torque",
Control Engineering, Feb 1986

[73] Kim D., Song J., Kim S.,
"Dependence Of Machining Accuracy On Acceleration/Deceleration And Interpolation Methods In CNC Machine Tools",
IEEE – IAS Annual Meeting, pp 1898 – 1905, Denver, USA, 1994

[74] Sherkat Nasser,
"High Speed Processing for Real-Time Control of Multiple Axes Continuous Path Manipulators",
PhD Thesis, Department of Computing, Nottingham Polytechnic, 1989

[75] Sherkat N. and Thomas P.D.,
"High Speed Processing for Real-Time Control of Multiple Axes Continuous Path Manipulators", Conference on Real-Time Systems: Theory and Application, York, UK, Sept. 1989

[76] Kim Dong-IL, Kim Jin-Il, Kim Sungkwun,
"Design of Digital Signal Processor System for CNC Systems",
International Conference on Industrial Electronics, Control and Instrumentation,
pp 1861 – 1866, 1991

[77] Huang S.J., Shieh M.H.,
"Application of DSP Controller on X-Y Table Servo Control",
Vol. 16, pp 205 – 211, 2000

[78] Discussion from Dr Poliakoff J.

[79] R. Butler, E. Kerr,
"An Introduction to Numerical Methods",
Pitman Publishing

[80] John H. Matthews,
"Numerical Methods",
Prentice-Hall International Editions

[81] Nicolaos Christodoulou,
"The Programmable Graphical Simulation of Stepping Motors Driven CNC 2-Axes Machine",
MSc Thesis, Department of Computing,
The Nottingham Trent University,
2000

Appendix A: Newton-Raphson

This appendix explains how the Newton-Raphson method has been used to calculate the times required for the parabolic acceleration algorithm. The aim is to find a solution for $f(t) = 0$. Given the value of $t = t_k$ at the end of the kth iteration, t_{k+1} is defined as [79][80]:

$$t_{k+1} = t_k - \frac{f(t_k)}{f'(t_k)} \tag{A-1}$$

$f(t_k)$ represents the value of the function t_k, and $f'(t_k)$ is the derivative (slope) at t_k, which represents $df(t)/dt$.

The iteration represented by equation (A-1) is repeated until a close enough approximation is reached. This is determined by a parameter, ε, which is a value very close to zero. The following checking is made to determine if a close enough root is found.

$$|f(t_k)| \leq \varepsilon \tag{A-2}$$

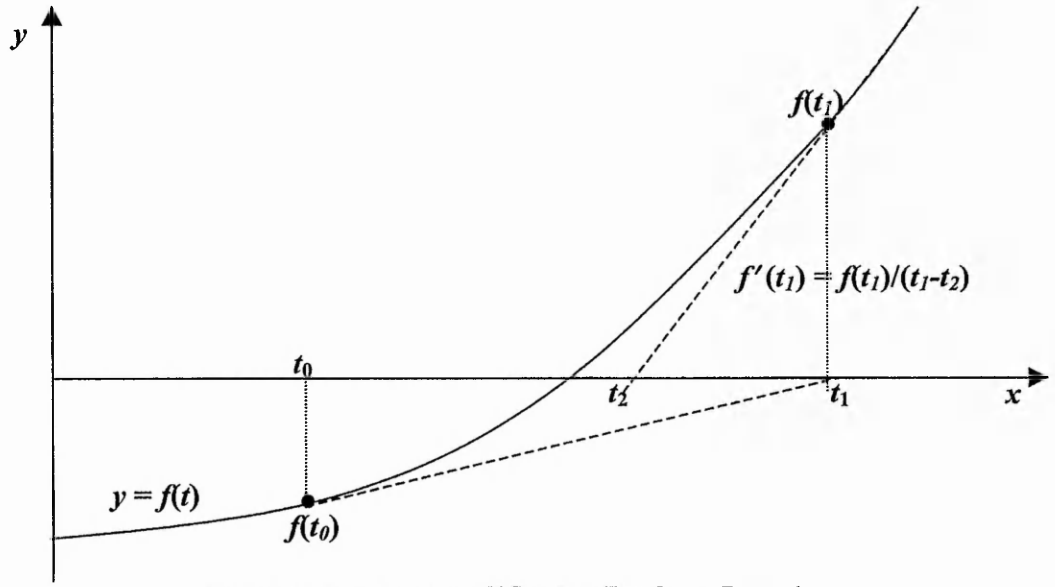


Figure A-1: Example of Newton-Raphson Iteration.

A graphical representation of the Newton-Raphson method will be helpful. An example of Newton-Raphson method is illustrated in Figure A-1. An initial guess of the root is assumed to be t_0. A tangent to the curve at $(t_0, f(t_0))$ is constructed. It is extrapolated until it intersects the t axis to get t_1. This point of intersection is taken as the new approximation to the root.

The equation of the tangent at t_0 is

$$y - f(t_0) = m(t - t_0) \tag{A-3}$$

where

$$m = f'(t_0) \tag{A-4}$$

Substituting equation (A-4) into (A-3) yields:

$$y - f(t_0) = f'(t_0)(t - t_0) \tag{A-5}$$

Since the tangent intersects the t axis at y=0 and this value gives the next approximation of the root. Thus,

$$t_1 = t_0 - \frac{f(t_0)}{f'(t_0)} \tag{A-6}$$

The tangent at $(t_1, f(t_1))$ cuts the t axis at t_2 which is a better approximation of the root than t_1 or t_0. The same process is repeated to generate a sequence of iterations (t_3, t_4, …, t_k) until the convergence is obtained whenever possible.

$$t_k \to r \text{ as } k \to \infty \tag{A-7}$$

Based on the technique of Newton-Raphson iteration, the root of equation (4-14) can be found. At each pulse, the distance to be travelled is calculated and included in equation

(4-14) using Newton-Raphson with a new function each time. Each new function is defined as follows:

$$f(t) = s(t) - s_x(n_x) \tag{A-8}$$

and t_n can be found by solving:

$$f(t) = 0; \tag{A-9}$$

$s_x(n_x)$ is the expected distance travelled (when the nth x pulse is to be sent and it is reached) at time $t(n)$. The new function, $f(t)$, represents the difference between the distance required and the distance travelled at time t. Therefore, at $t_k(n)$, we have $f(t_k(n))$ is close to zero. The explanation above is clearer with a graphical representation of a particular example. In these graphs, the following parameters are employed:

$$T = 0.15 \text{ s}$$
$$v_0 = 2 \text{ ms}$$
$$v_m = 80 \text{ mm/s}$$

Figure A-2 illustrates the speed graph generated from equation (4-9). Therefore the constant phase of the speed profile has the value 0 seconds in this diagram. Normally, there would be a constant phase between acceleration and deceleration. In this example, the parabolic speed reaches the required speed at the predefined time, which is 0.15s and then decreases again parabolically.

Integration of equation (4-9) will result in the distance travelled relative to the pulse timing and is defined as in equation (4-15). The graphical representation of the function $s(t)$ is shown in Figure A-3. This graph can be analysed in two segments. The first segment is the graph before 0.15s, which is the acceleration phase, and the second segment is the one after 0.15s, which is the deceleration phase.

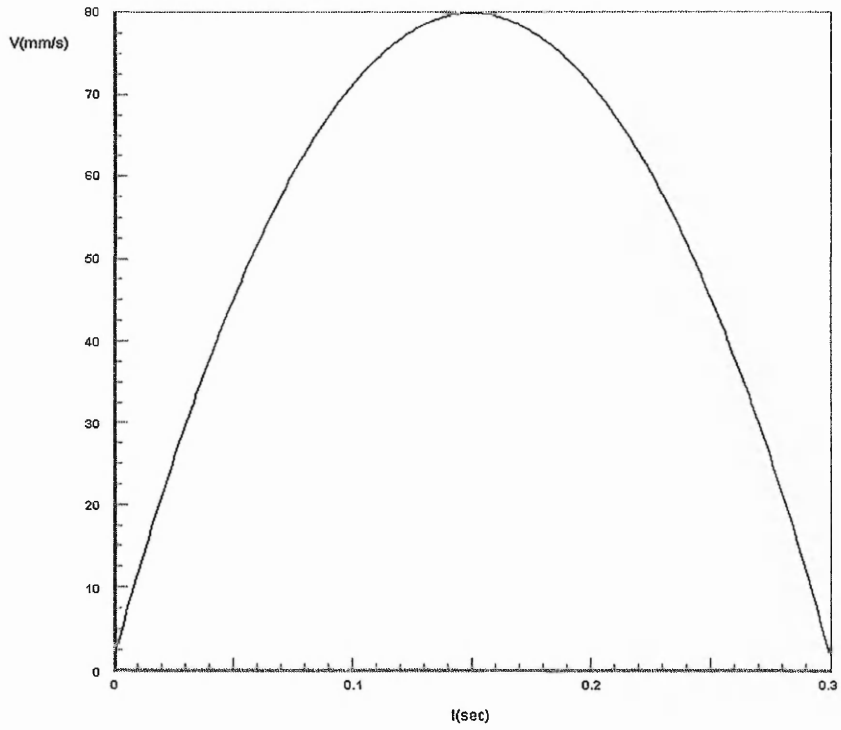


Figure A-2: Example of Parabolic Acceleration/Deceleration where there is No Constant Speed Phase.

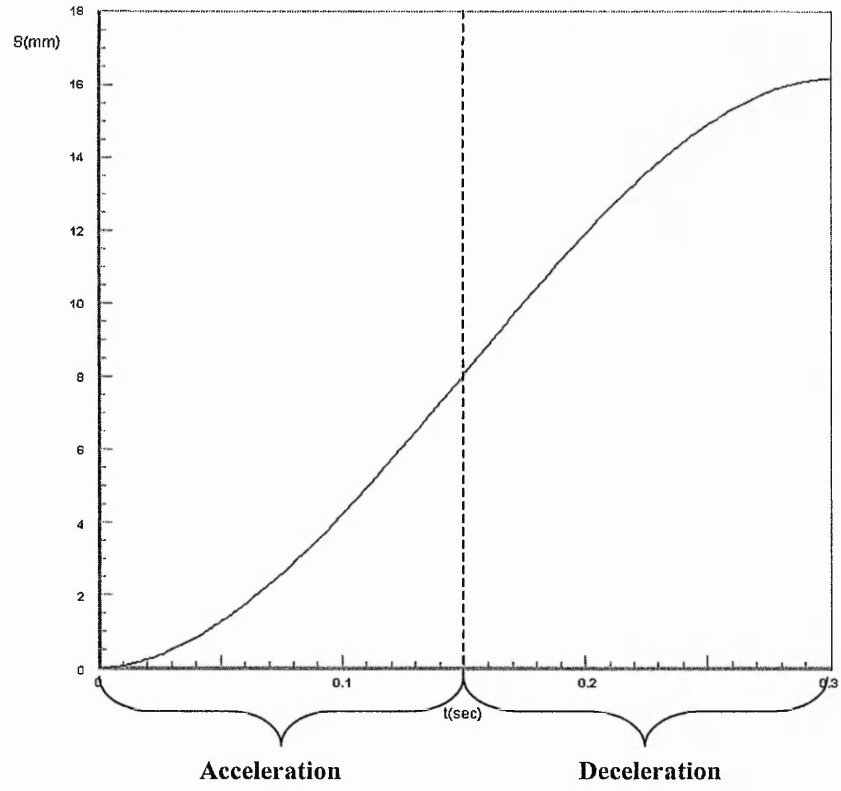


Figure A-3: Distance Travelled During Acceleration/Deceleration from Figure 4-10.

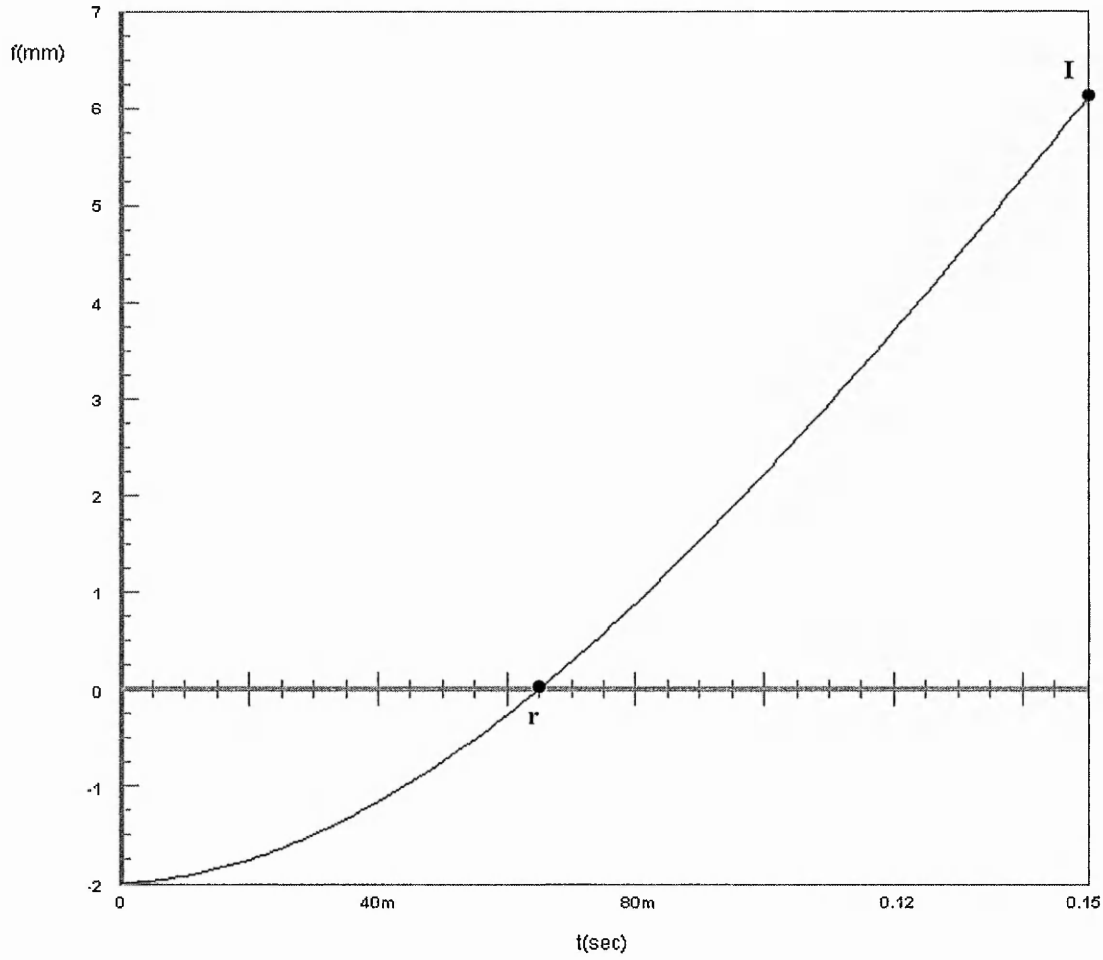


Figure A-4: Example of Function Used in Newton-Raphson for Acceleration in the Case when $s(n) = 2$ mm.

The acceleration phase is considered first. To find the pulse timing when a certain distance has been travelled, the graph of $f(t)$ is the graph of $s(t)$ shifted down by that distance so that the graph cuts the t axis at the required pulse timing. As an example, to find the pulse timing when the machining has passed through a distance of 2 mm, the graph of $f(t)$ is the graph of $s(t)$ shifted down by 2 mm, as illustrated in Figure A-4. The new function, $f(t)$, is defined as in equation (A-8).

To start the Newton-Raphson iteration, an approximation to the root has to be determined. The total time (T) for the acceleration phase has been chosen as the initial estimate of the root for all pulse timings. In this example, it is 0.15s. It is found that the

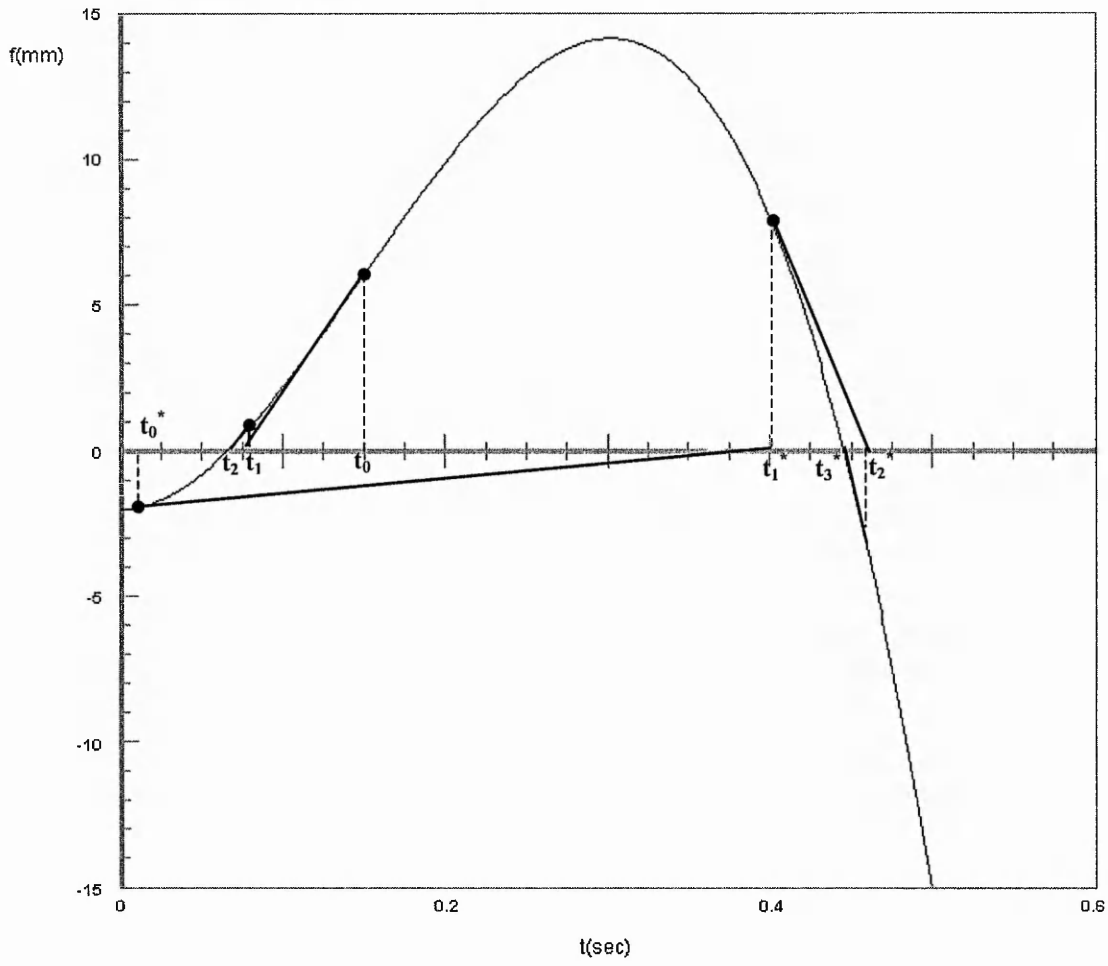


Figure A-5: Example of how the Newton-Raphson Iteration with Two Different Initial Estimates of the Root can give different results.

root converges quickly and the error is very small after a few iterations. In this example, the pulse timing for the acceleration has to be between 0 to 0.15s. Similarly, the pulse timing for the deceleration phase has to be between 0.15 to 0.3s. From the curve in Figure A-5, we can tell that the highest curvature occurs at the start of the acceleration and the end of the deceleration phase. The least curvature happens at the end of the acceleration phase, which is 0.15s in this example. Therefore, if this is chosen to be the initial estimate of the root, it will always ensure that the intermediate timings will not be out of the 0 to 0.15s range.

During the acceleration phase, the gradient of the curve (between the root and the inflection point, I), $f'(t)$, is increasing, so tangent will be below curve and will hit axis to the right of the root, r.

The example in Figure A-5 demonstrates two chosen initial estimate for the root, t_0 and t_0^*. Using t_0 (inflection point = 0.15s) as the estimate enables the required root to be found after two iterations. On the other hand, with the t_0^* as the initial estimate, the next estimate obtained from the Newton-Raphson iteration will result in a root estimate that is beyond 0.3s. Thus, the root found is not the root that is required. This example has been exaggerated to demonstrate the problems that may occur if an improper initial estimate is used for the root.

The deceleration phase can be performed in a similar way. However, to determine the pulse timing in the deceleration phase, a little more work has to be done because the timing generated from the Newton-Raphson iteration uses a time of zero for the constant phase.

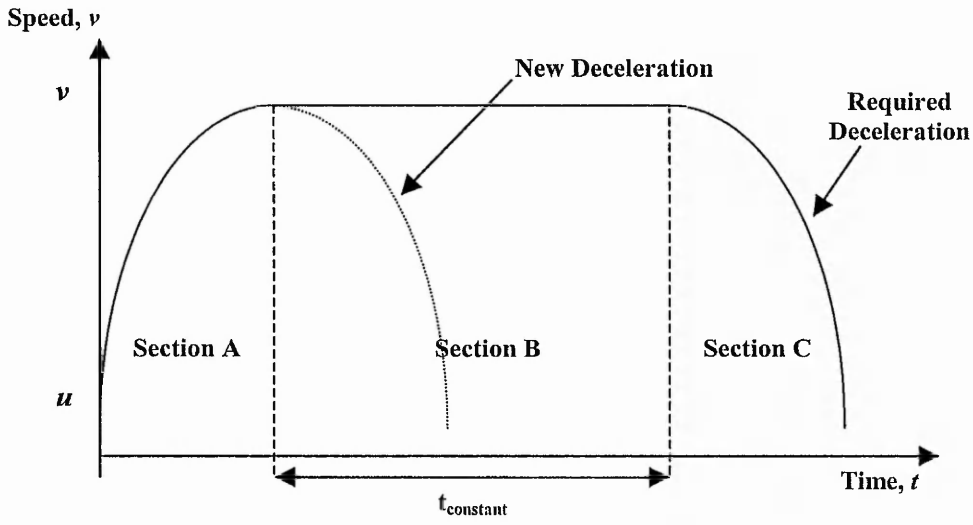


Figure A-6: Parabolic Deceleration Adjustment.

To find the pulse timing at the deceleration phase, the first step is to assume that the constant phase is of length zero, $t_{constant} = 0$. In other words, the required acceleration curve is shifted to the left in the time domain so that the start of the deceleration coincides with the end of the acceleration phase. Since the pulse timing is calculated

using distance as the parameter, the shifting of the required deceleration curve will involve altering the distance. To do this, the distance moved in the constant phase has to be subtracted from the actual distance, which effectively removes the area in Section B from Figure A-6. After shifting the deceleration phase, the pulse timing can be calculated as in the acceleration phase using the Newton-Raphson iterations. However, this timing value will have to have $t_{constant}$ added to it to obtain the required pulse timing, due to the shifting performed earlier.

As in the case of the acceleration phase, the deceleration phase is illustrated with a simple example. Assuming that the required distance is 15mm after shifting, the deceleration graph, Figure A-7, will have to be shifted down by 15mm to assure that the pulse timing for distance of 15mm crosses the *t* axis. It should be noted that this is not the actual distance travelled.

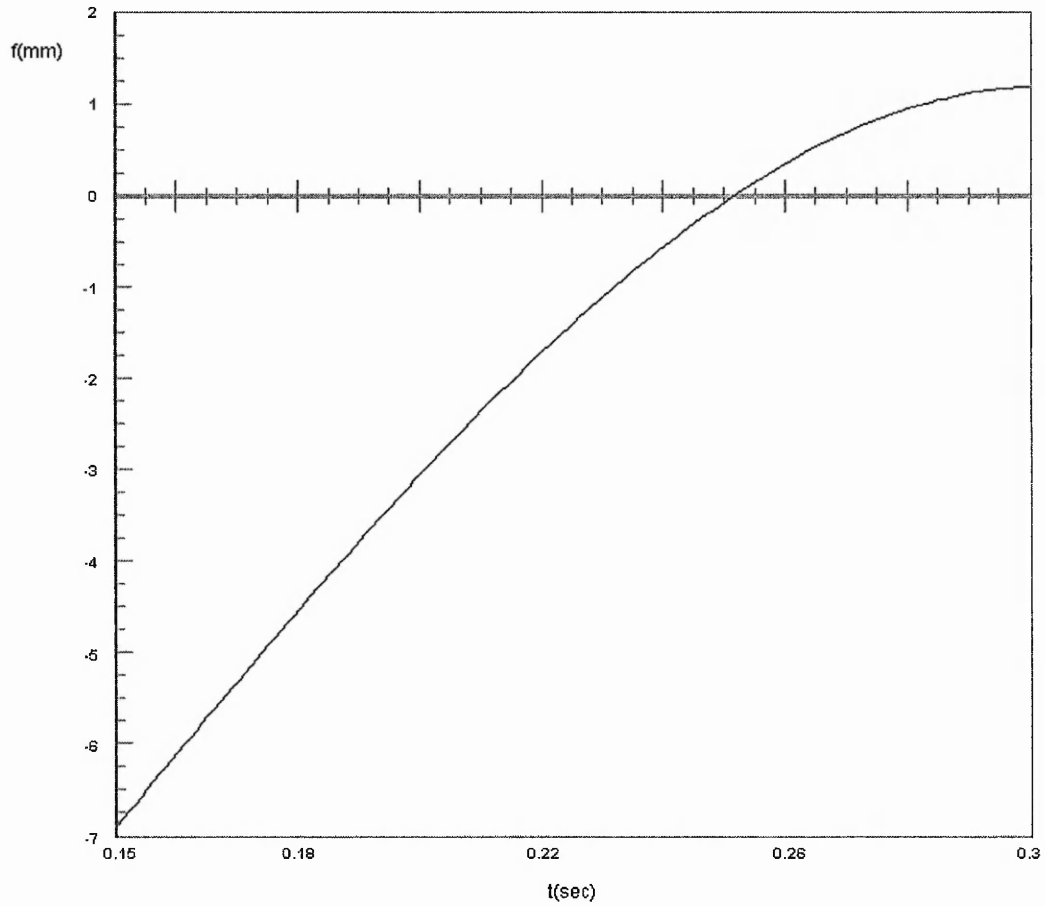


Figure A-7: Function Used In Newton-Raphson for Deceleration in the Case when s(n)=15 mm with Time Starting at 0.15s.

Appendix B: Position Error Plots for Interpolation

Appendix B includes figures showing the position error plots for the five interpolation algorithms, using different simulation methods, under evaluation. Section B.1 shows the position error plots for the line while Section B.2 is for error plots of the circular arc interpolation. The plots in this appendix correspond to the line and circular arc example in Section 6.2.

For the line, Figures B-1 to B-17 correspond to Figures 6-3 to 6-19. For the circular arc, Figures B-18 to B-34 correspond to Figures 6-25 to 6-41.

B.1 Straight Line

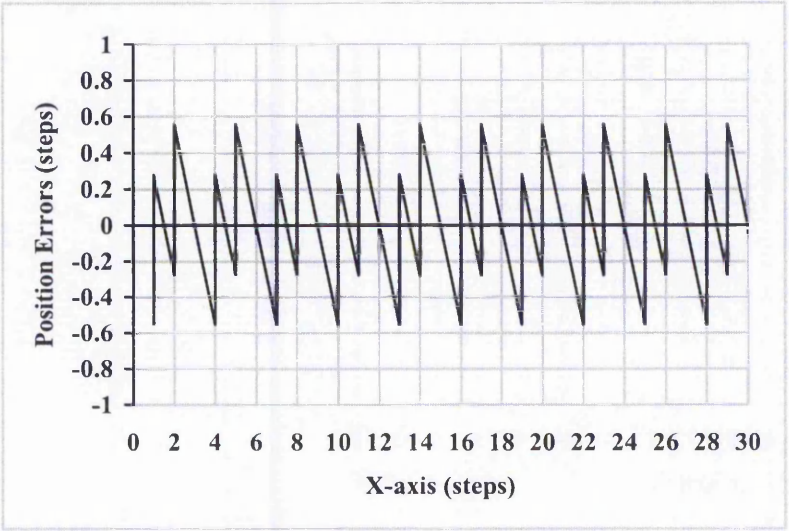


Figure B-1: Error Plots of Search-Step Linear Interpolation (Zero and Constant Rate First Order Simulation) shown in Figure 6.3. The largest position error here is 0.55.

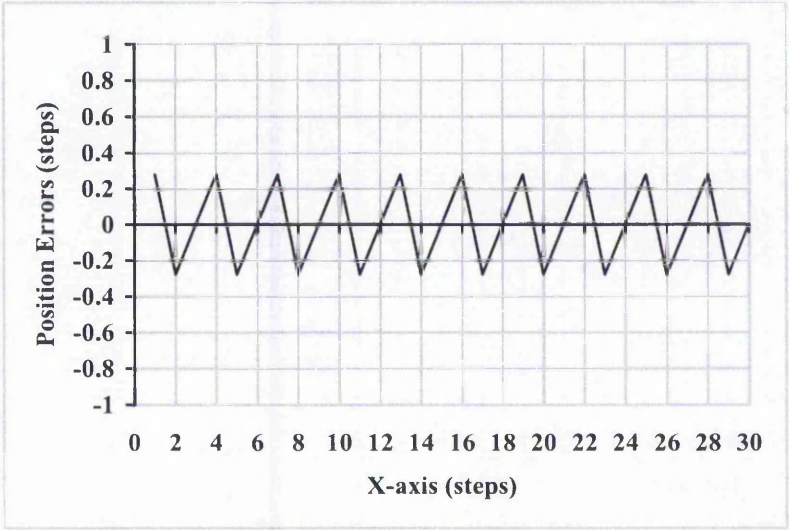


Figure B-2: Error Plots of Direct-Search Linear Interpolation (Zero and Constant Rate First Order Simulation) shown in Figure 6.4. The largest position error here is 0.28.

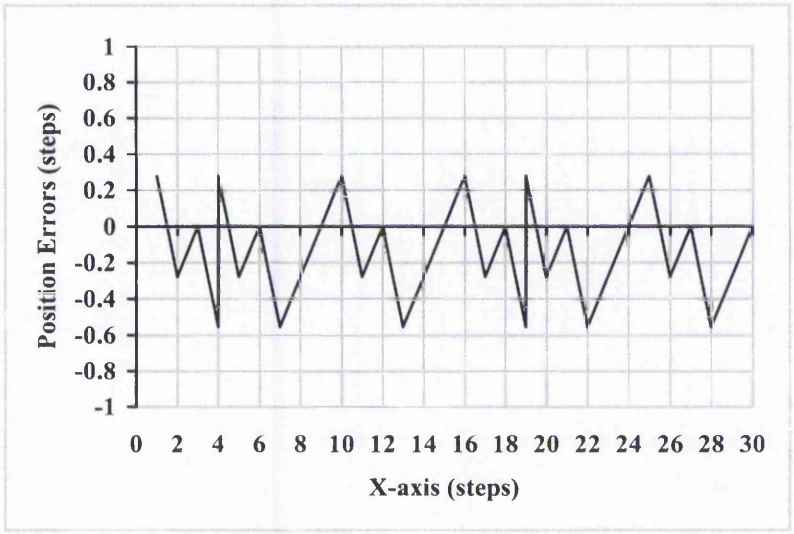


Figure B-3: Error Plots of DDA Linear Interpolation (Zero and Constant Rate First Order Simulation) shown in Figure 6.5. The largest position error here is 0.55.

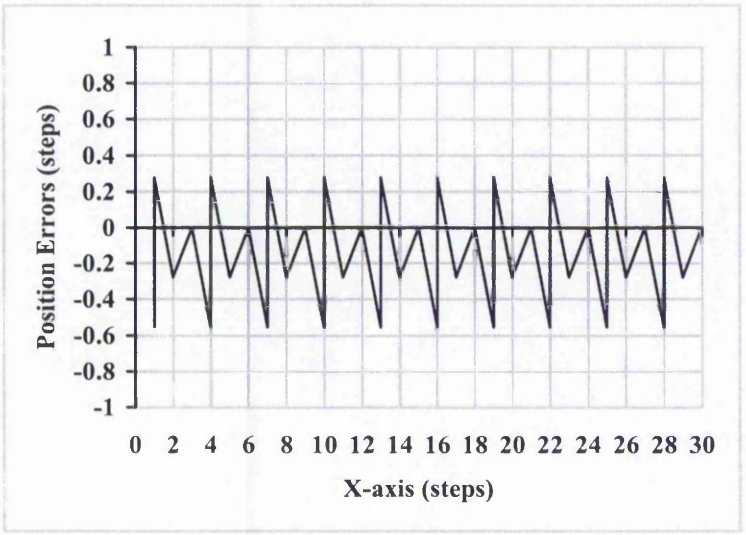


Figure B-4: Error Plots of New Full Step Linear Interpolation (Zero Order Simulation) shown in Figure 6.6. The largest position error here is 0.55.

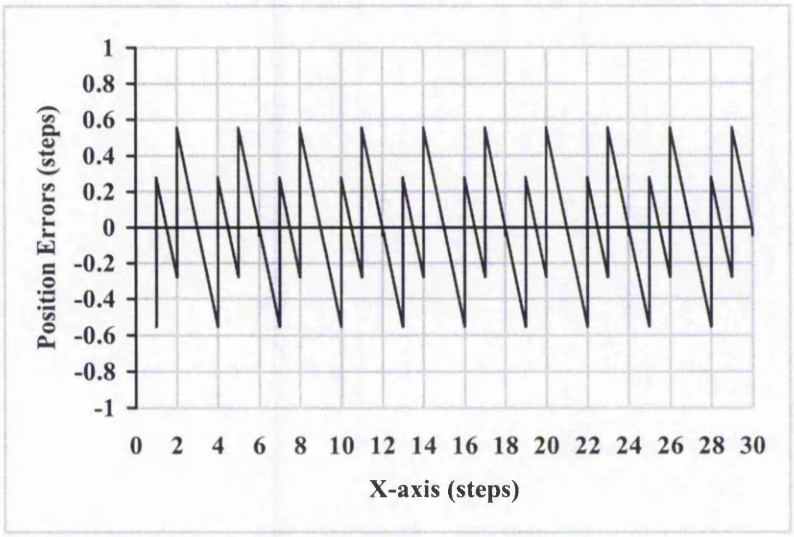


Figure B-5: Error Plots of New Half Step Linear Interpolation (Zero Order Simulation) shown in Figure 6.7. The largest position error here is 0.55.

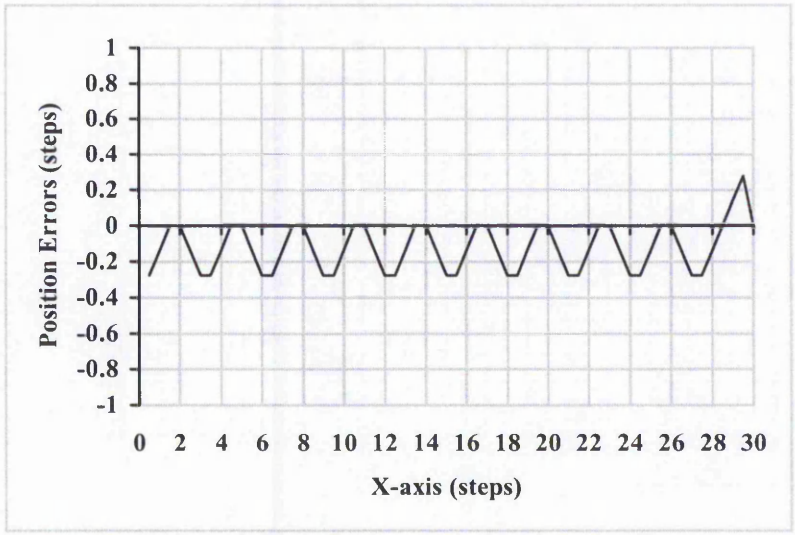


Figure B-6: Error Plots of Search-Step Linear Interpolation (Varying Rate First Order Simulation) shown in Figure 6.8. The largest position error here is 0.28.

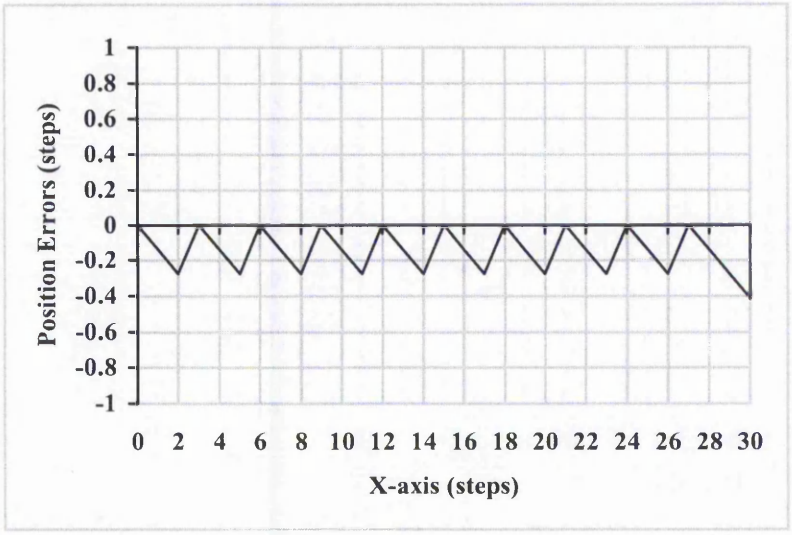


Figure B-7: Error Plots of Direct-Search Linear Interpolation (Varying Rate First Order Simulation) shown in Figure 6.9. The largest position error here is 0.42.

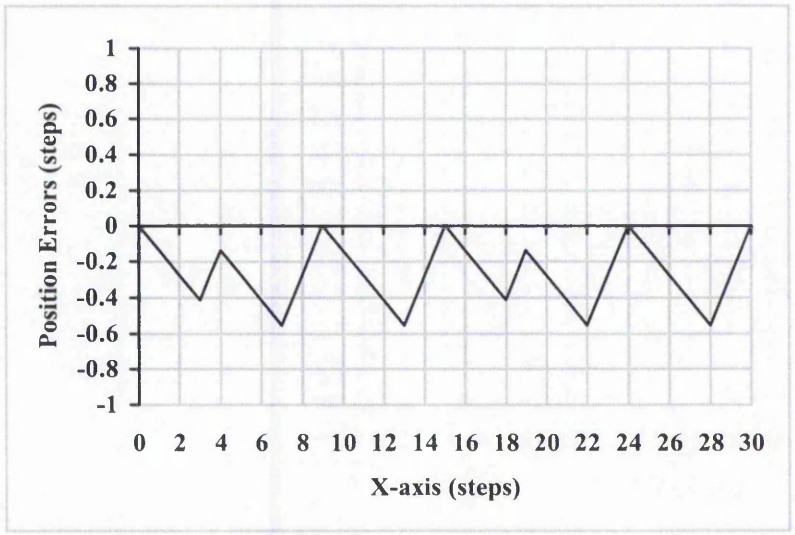


Figure B-8: Error Plots of DDA Linear Interpolation (Varying Rate First Order Simulation) shown in Figure 6.10. The largest position error here is 0.55.

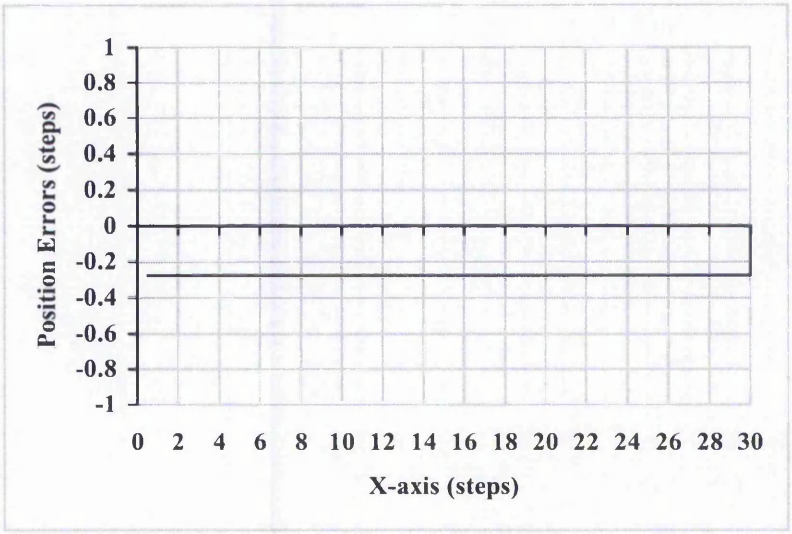


Figure B-9: Error Plots of New Full Step Linear Interpolation (Varying Rate First Order Simulation) shown in Figure 6.11. The largest position error here is 0.28.

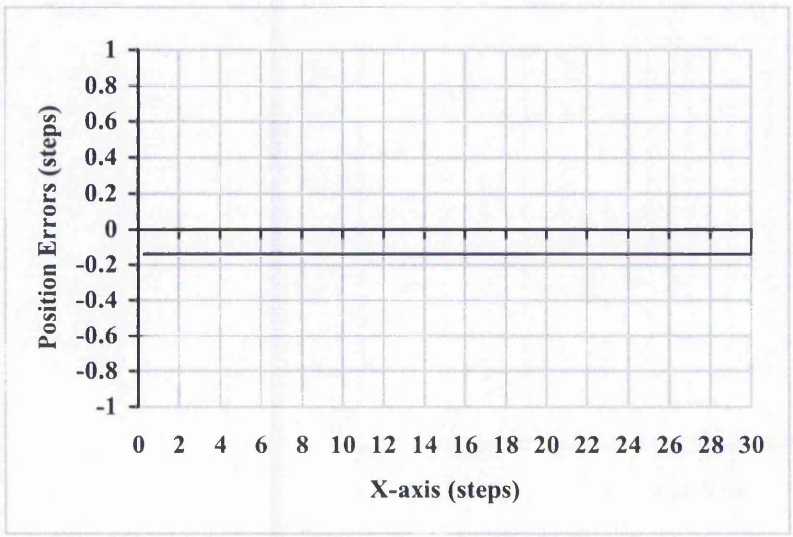


Figure B-10: Error Plots of New Half Step Linear Interpolation (Varying Rate First Order Simulation) shown in Figure 6.12. The largest position error here is 0.14.

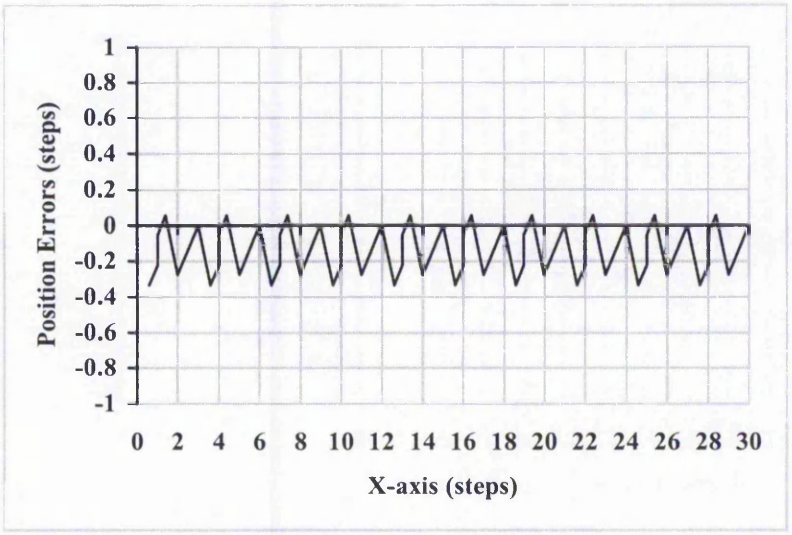


Figure B-11: Error Plots of New Full Step Linear Interpolation (Constant Rate First Order Simulation) shown in Figure 6.13. The largest position error here is 0.33.

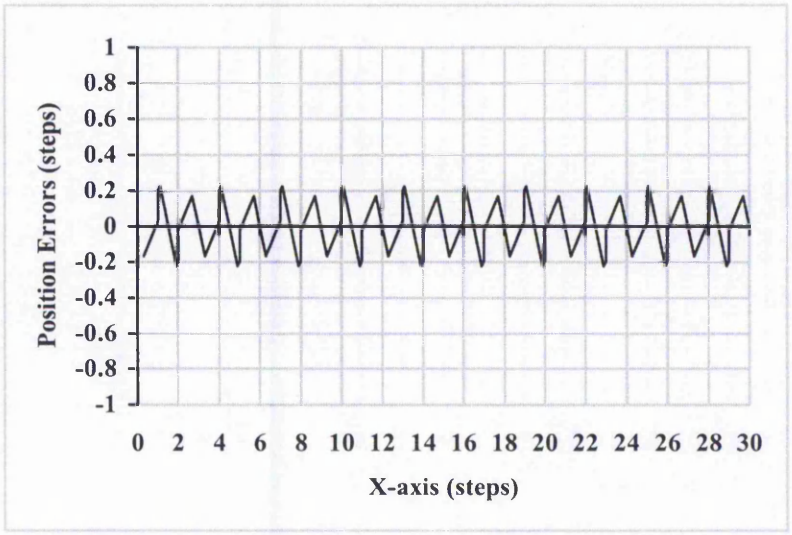


Figure B-12: Error Plots of New Half Step Linear Interpolation (Constant Rate First Order Simulation) shown in Figure 6.14. The largest position error here is 0.22.

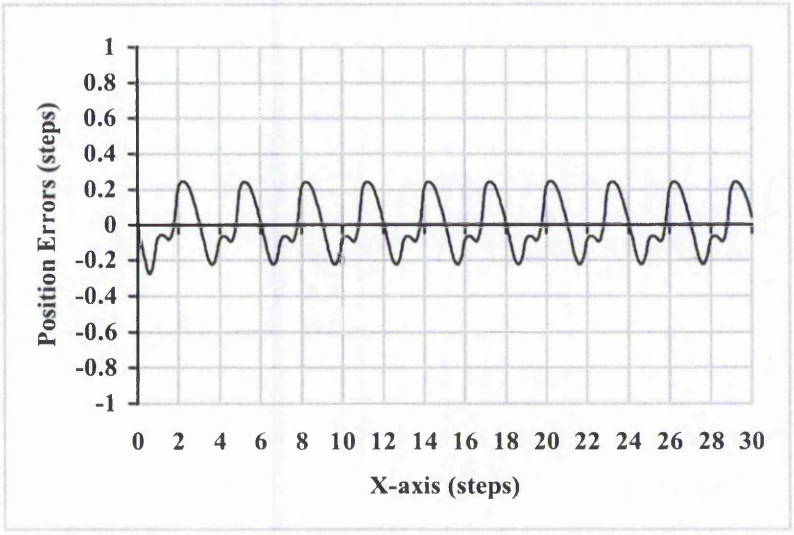


Figure B-13: Error Plots of Search-Step Linear Interpolation (Second Order Simulation) shown in Figure 6.15. The largest position error here is 0.28.

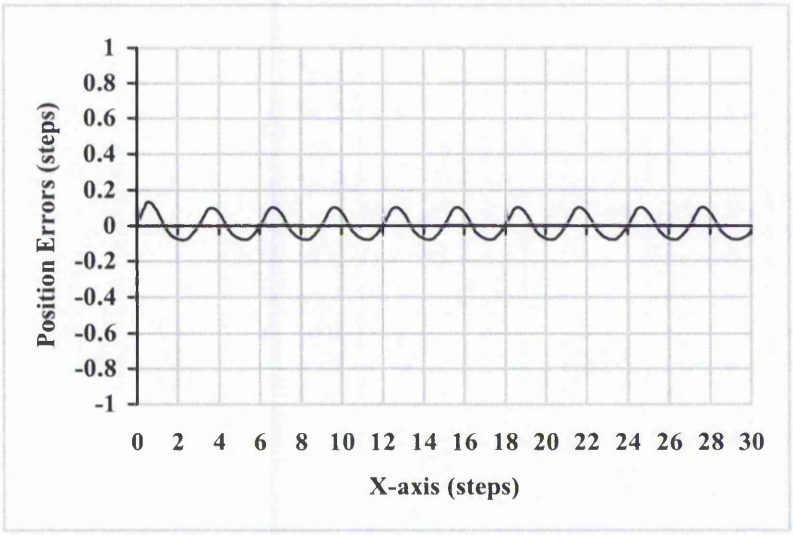


Figure B-14: Error Plots of Direct-Search Linear Interpolation (Second Order Simulation) shown in Figure 6.16. The largest position error here is 0.13.

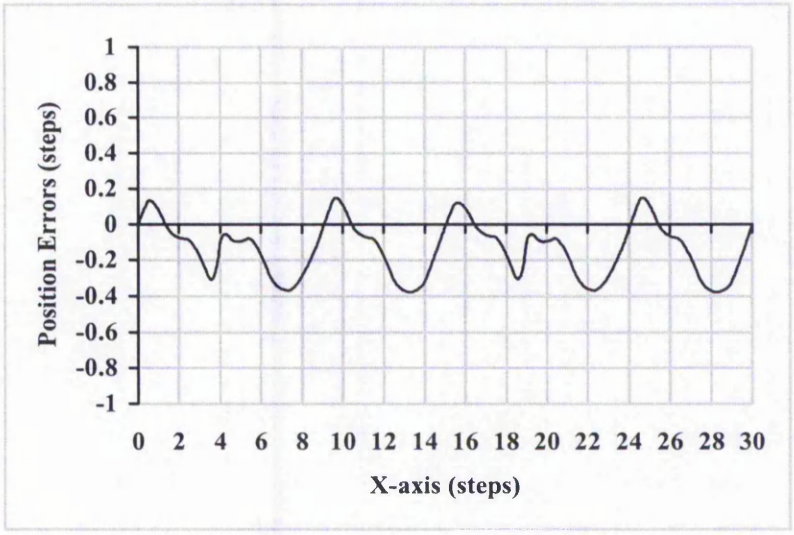


Figure B-15: Error Plots of DDA Linear Interpolation (Second Order Simulation) shown in Figure 6.17. The largest position error here is 0.38.

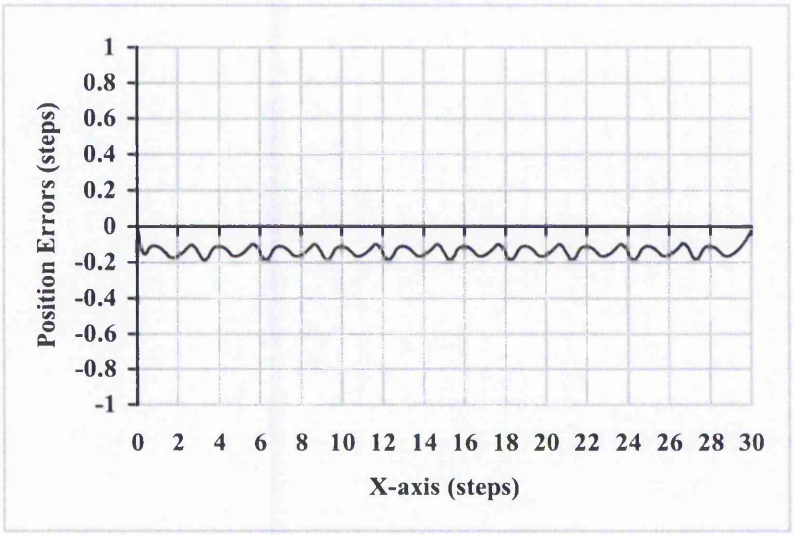


Figure B-16: Error Plots of New Full Step Linear Interpolation (Second Order Simulation) shown in Figure 6.18. The largest position error here is 0.19.

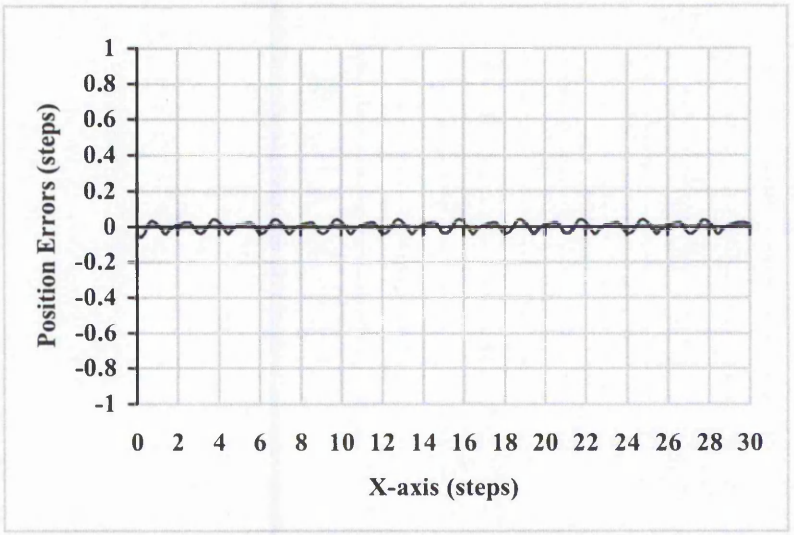


Figure B-17: Error Plots of New Half Step Linear Interpolation (Second Order Simulation) shown in Figure 6.19. The largest position error here is 0.06.

B.2 Circular Arc

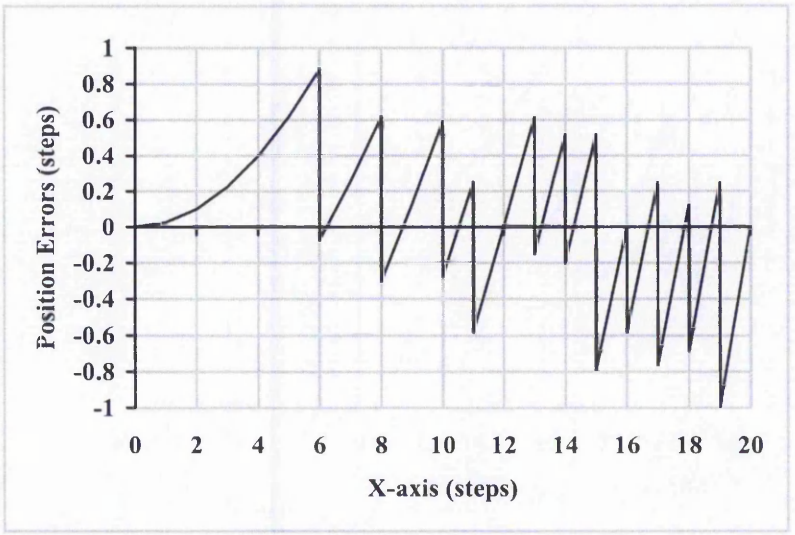


Figure B-18: Error Plots of Search-Step Circular Arc Interpolation (Zero Order Simulation) shown in Figure 6.25. The largest position error here is 1.00.

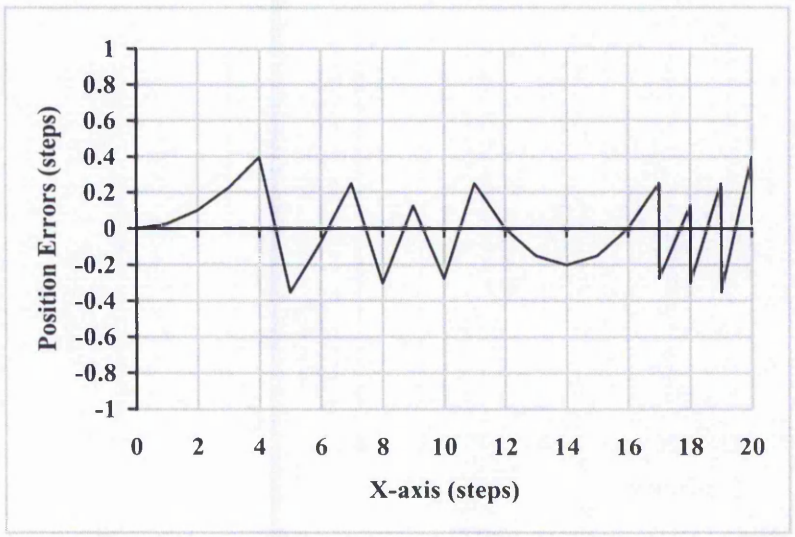


Figure B-19: Error Plots of Direct-Search Circular Arc Interpolation (Zero Order Simulation) shown in Figure 6.26. The largest position error here is 0.40.

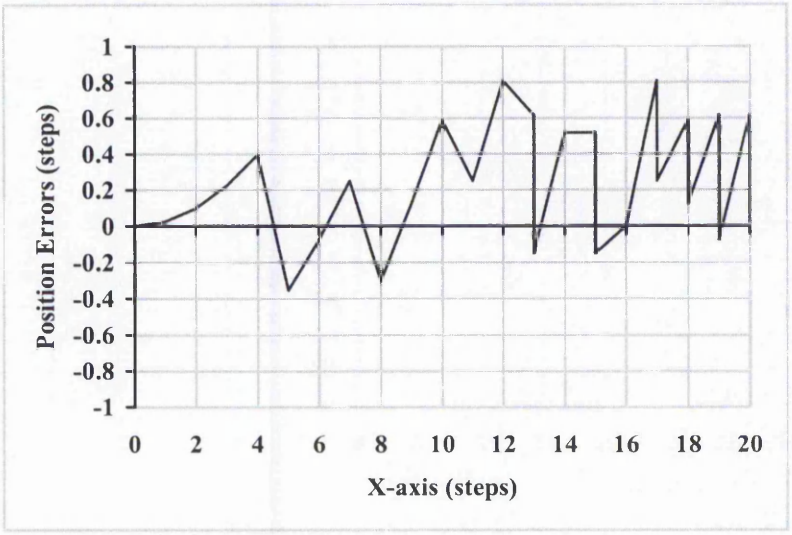


Figure B-20: Error Plots of DDA Circular Arc Interpolation (Zero Order Simulation) shown in Figure 6.27. The largest position error here is 0.81.

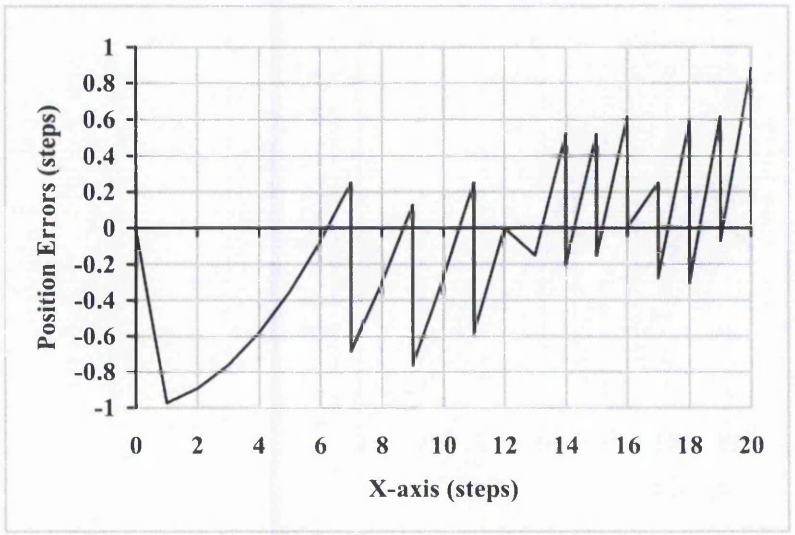


Figure B-21: Error Plots of New Full Step Circular Arc Interpolation (Zero Order Simulation) shown in Figure 6.28. The largest position error here is 0.97.

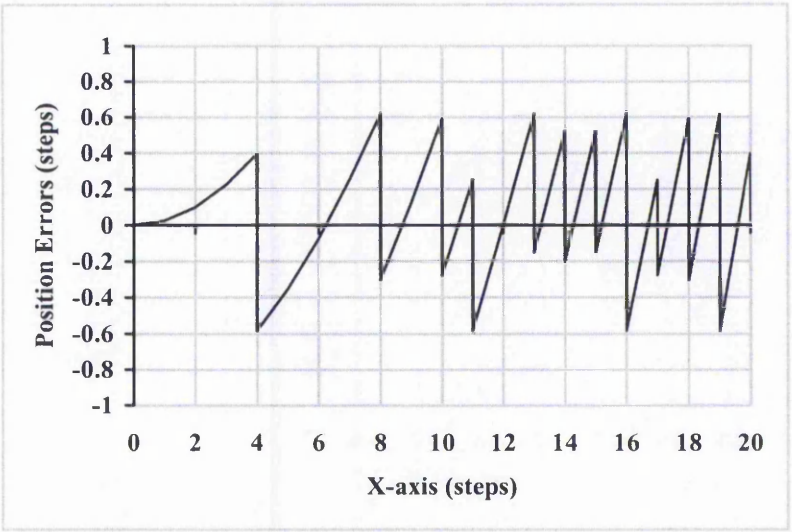


Figure B-22: Error Plots of New Half Step Circular Arc Interpolation (Zero Order Simulation) shown in Figure 6.29. The largest position error here is 0.62.

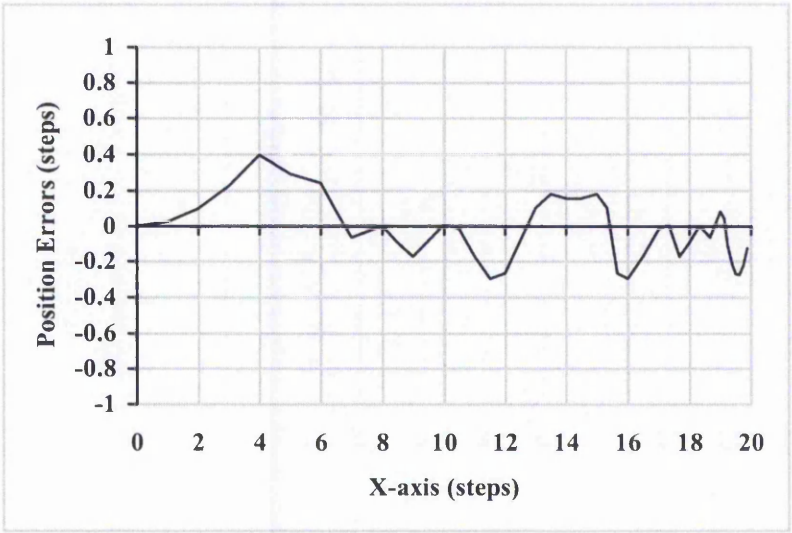


Figure B-23: Error Plots of Search-Step Circular Arc Interpolation (Varying Rate First Order Simulation) shown in Figure 6.30. The largest position error here is 0.40.

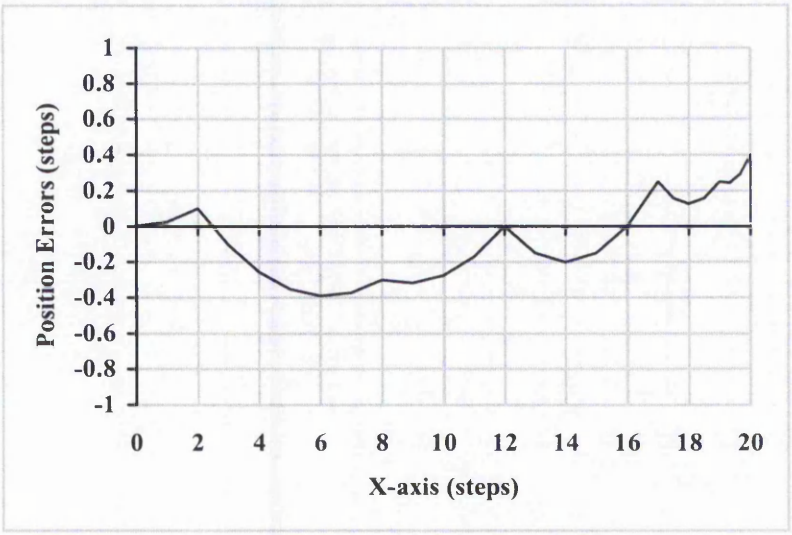


Figure B-24: Error Plots of Direct-Search Circular Arc Interpolation (Varying Rate First Order Simulation) shown in Figure 6.31. The largest position error here is 0.40.

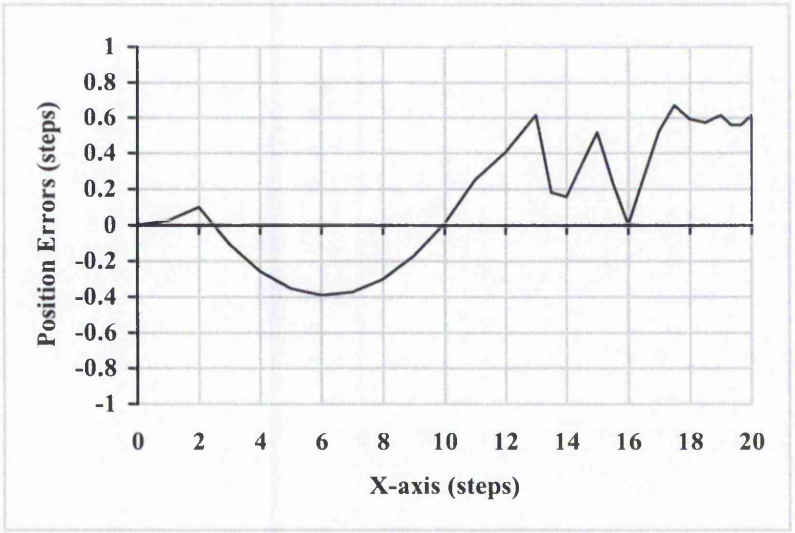


Figure B-25: Error Plots of DDA Circular Arc Interpolation (Varying Rate First Order Simulation) shown in Figure 6.32. The largest position error here is 0.67.

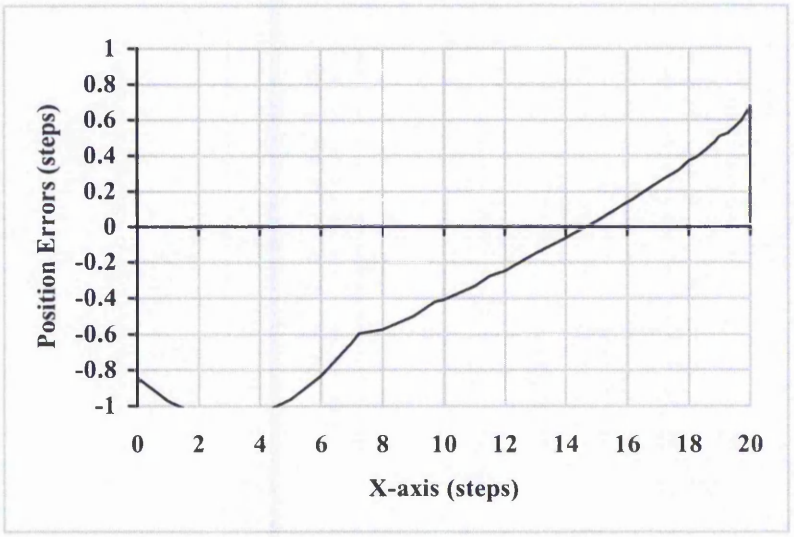


Figure B-26: Error Plots of New Full Step Circular Arc Interpolation (Varying Rate First Order Simulation) shown in Figure 6.33. The largest position error here is 1.08.

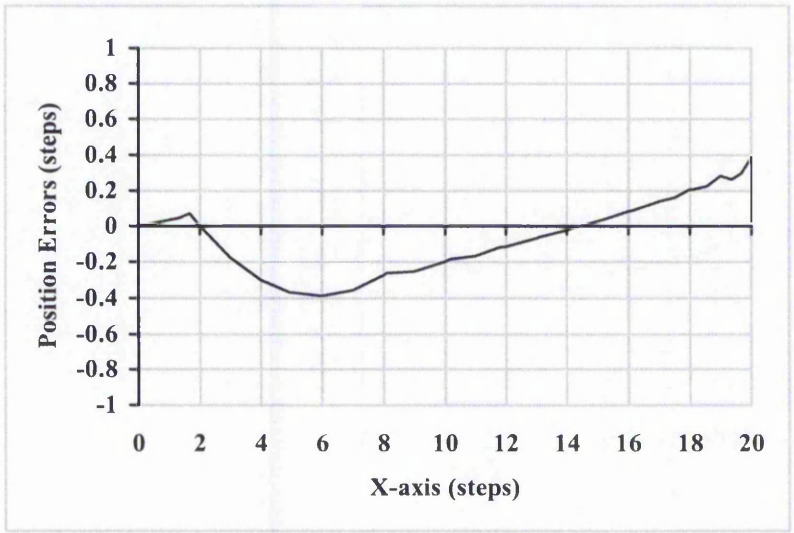


Figure B-27: Error Plots of New Half Step Circular Arc Interpolation (Varying Rate First Order Simulation) shown in Figure 6.34. The largest position error here is 0.39.

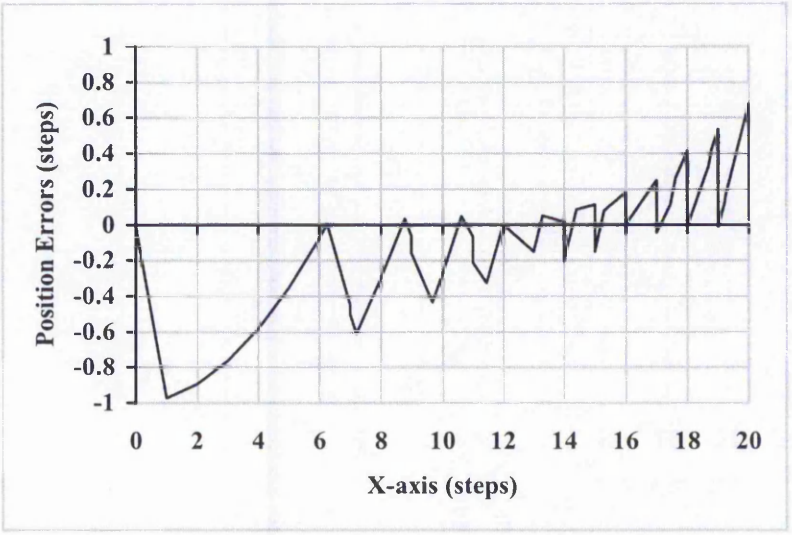


Figure B-28: Error Plots of New Full Step Circular Arc Interpolation (Constant Rate First Order Simulation) shown in Figure 6.35. The largest position error here is 0.97.

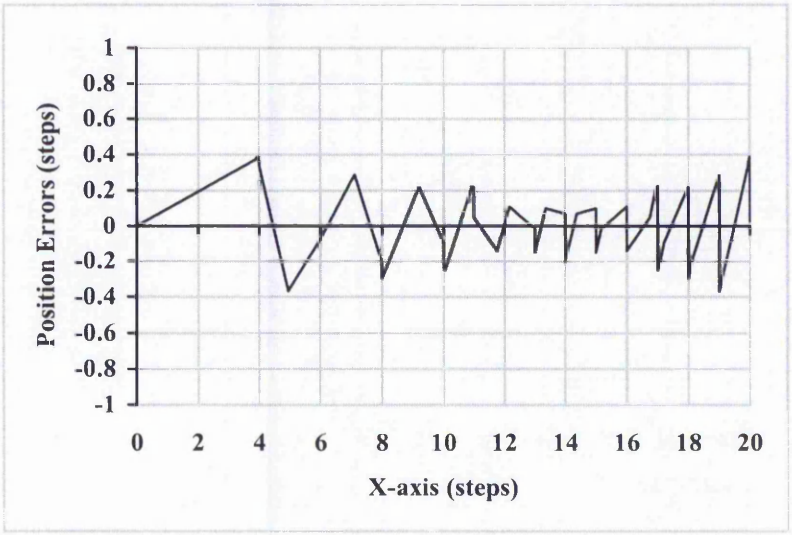


Figure B-29: Error Plots of New Half Step Circular Arc Interpolation (Constant Rate First Order Simulation) shown in Figure 6.36. The largest position error here is 0.39.

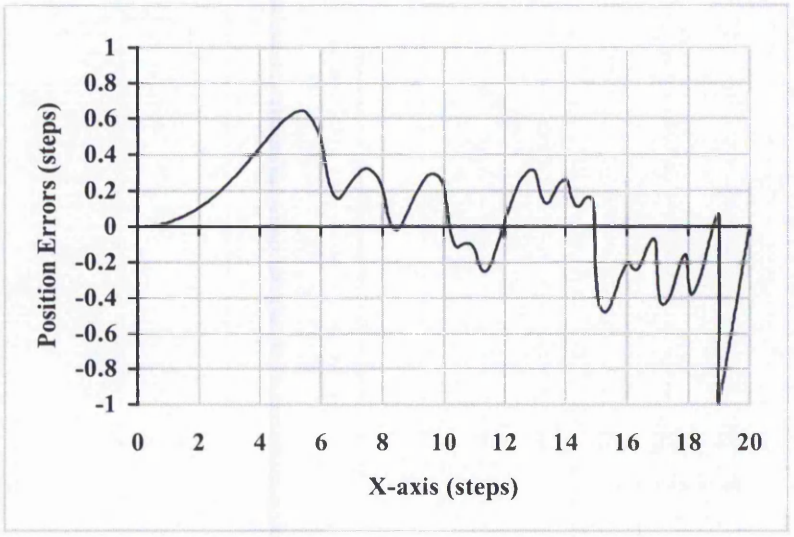


Figure B-30: Error Plots of Search-Step Circular Arc Interpolation (Second Order Simulation) shown in Figure 6.37. The largest position error here is 0.99.

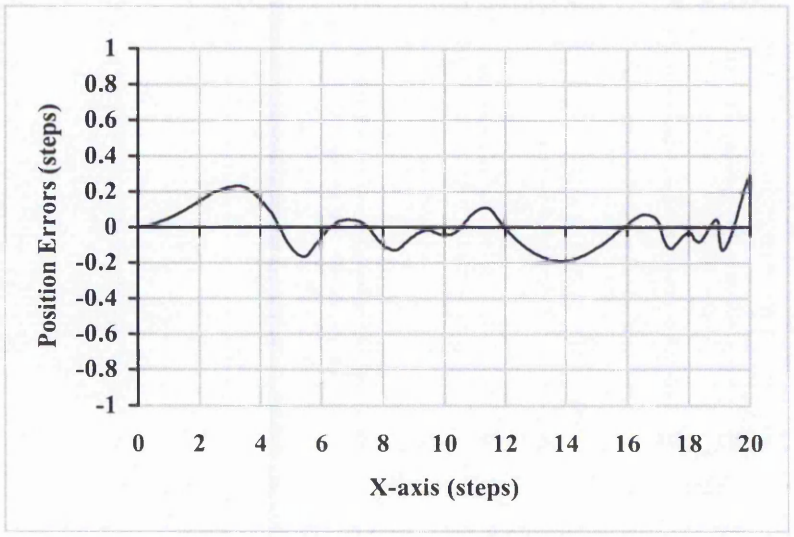


Figure B-31: Error Plots of Direct-Search Circular Arc Interpolation (Second Order Simulation) shown in Figure 6.38. The largest position error here is 0.29.

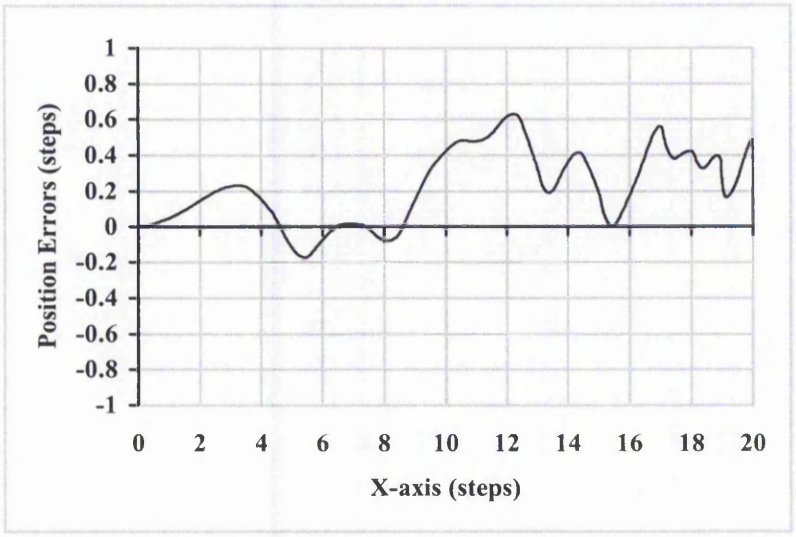


Figure B-32: Error Plots of DDA Circular Arc Interpolation (Second Order Simulation) shown in Figure 6.39. The largest position error here is 0.63.

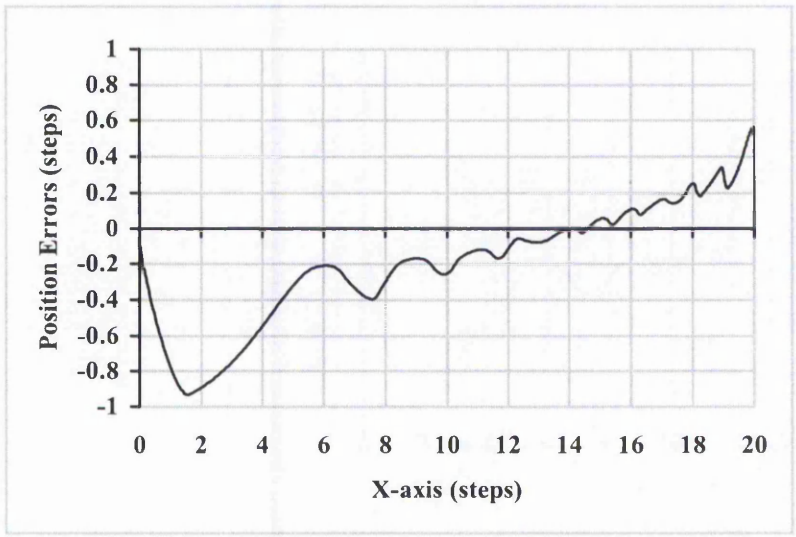


Figure B-33: Error Plots of New Full Step Circular Arc Interpolation (Second Order Simulation) shown in Figure 6.40. The largest position error here is 0.93.

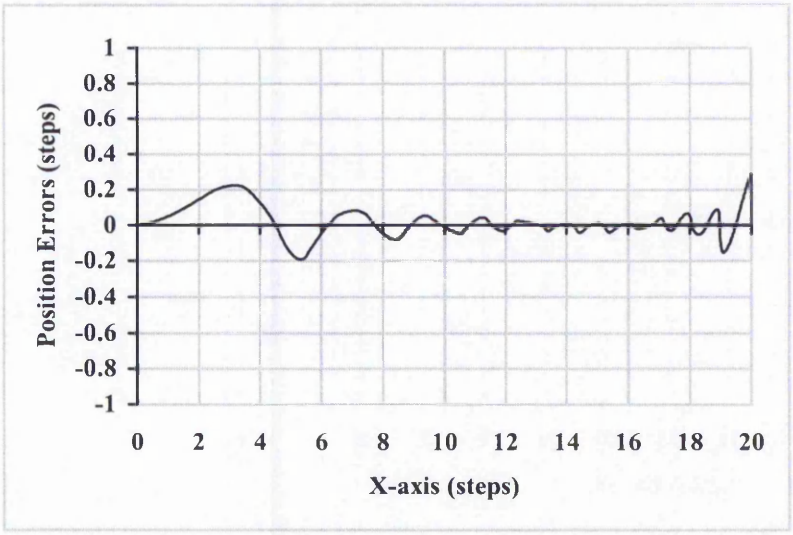


Figure B-34: Error Plots of New Half Step Circular Arc Interpolation (Second Order Simulation) shown in Figure 6.41. The largest position error here is 0.28.

Appendix C: Position Error Plots for Acceleration

Section C1 shows the position errors plot for the line under acceleration and Section C2 shows the plots for the circular arc. The figures are shown in pairs, the first using linear acceleration and the second using parabolic acceleration for comparison.

Acceleration stops at x=761 steps and deceleration starts at x=2241 steps for linear acceleration on line. For parabolic acceleration, acceleration stops at x=676 steps and deceleration starts at x=2326 steps.

For linear acceleration on the circular arc, acceleration stops at x=1794 steps and deceleration starts at x=881 steps while for parabolic acceleration, acceleration stops at x=1837 steps and deceleration starts at x=788 steps.

For the line, Figures C-1, C-3, C-5 and C-7 correspond to Figures 6-48 to 6-51 while Figures C-2, C-4, C-6 and C-8 correspond to Figures 6-59 to 6-62. For the arc, Figure C-9, C-11, C-13 and C-15 correspond to Figures 6-53 to 6-56 while Figures C-10, C-12, C-14 and C-16 correspond to Figures 6-64 to 6-67.

C.1 Straight Line

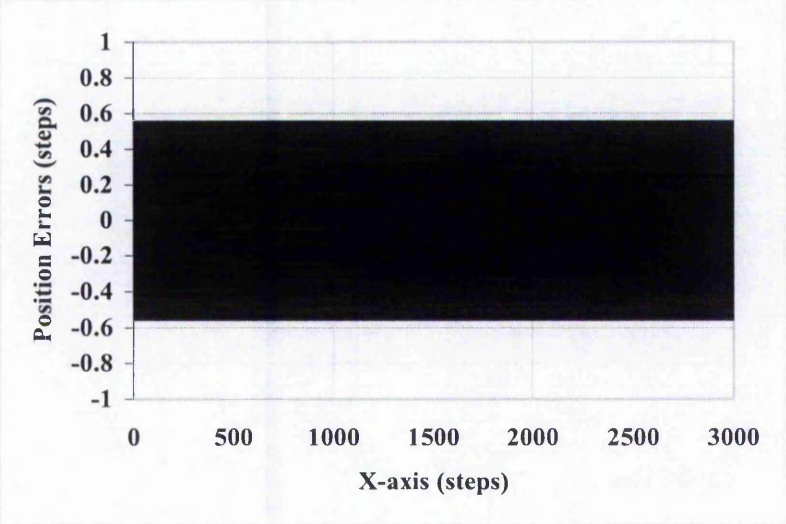


Figure C-1: Error Plots of New Half Step Linear Interpolation with Linear Acceleration (Zero Order Simulation) shown in Figure 6.48. The largest position error here is 0.55.

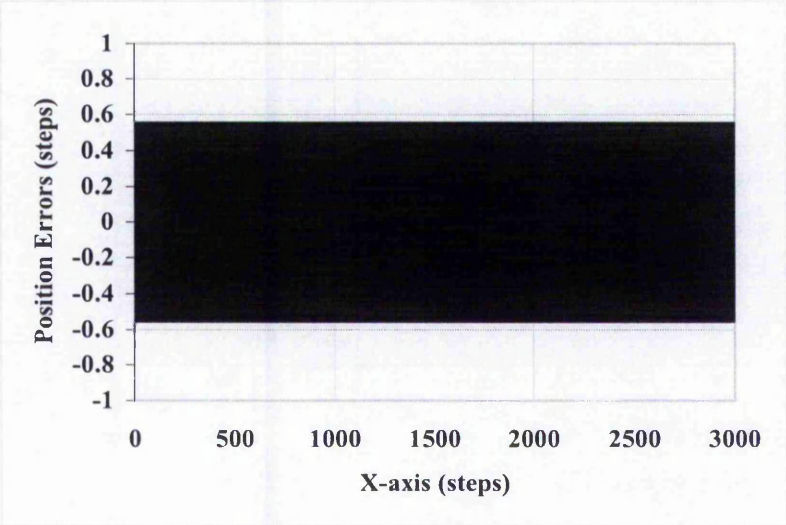


Figure C-2: Error Plots of New Half Step Linear Interpolation with Parabolic Acceleration (Zero Order Simulation) shown in Figure 6.59. The largest position error here is 0.55.

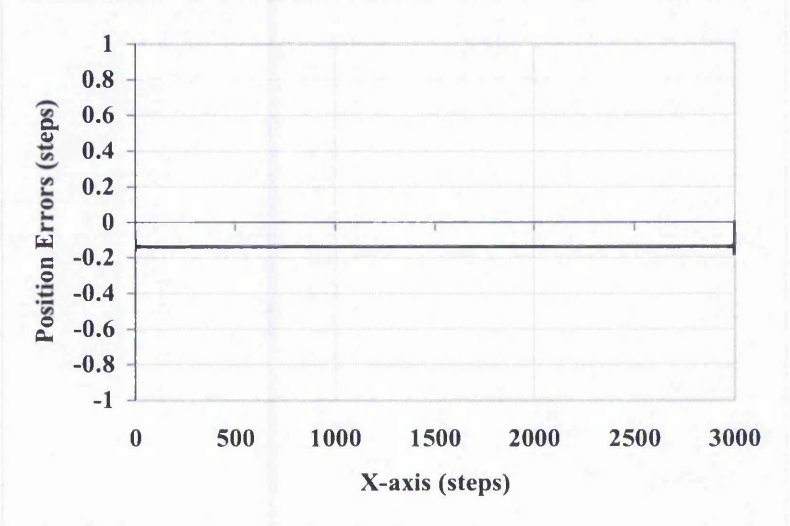


Figure C-3: Error Plots of New Half Step Linear Interpolation with Linear Acceleration (Varying Rate First Order Simulation) shown in Figure 6.49. The largest position error here is 0.18.

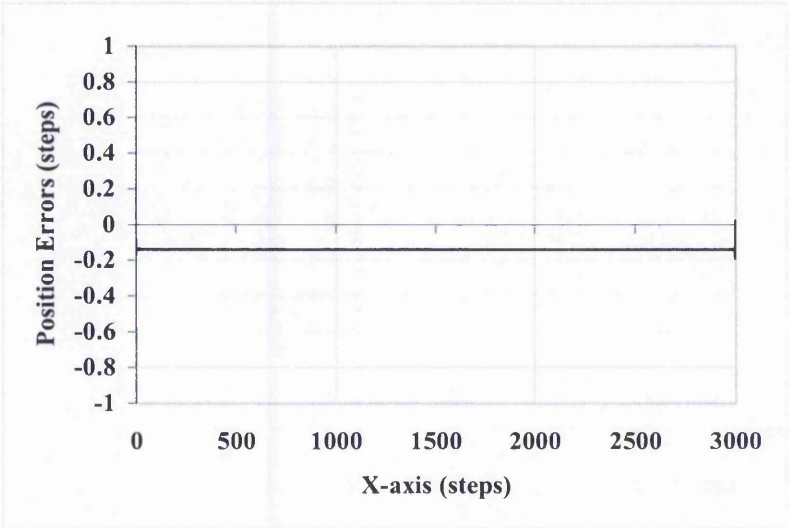


Figure C-4: Error Plots of New Half Step Linear Interpolation with Parabolic Acceleration (Varying Rate First Order Simulation) shown in Figure 6.60. The largest position error here is 0.19.

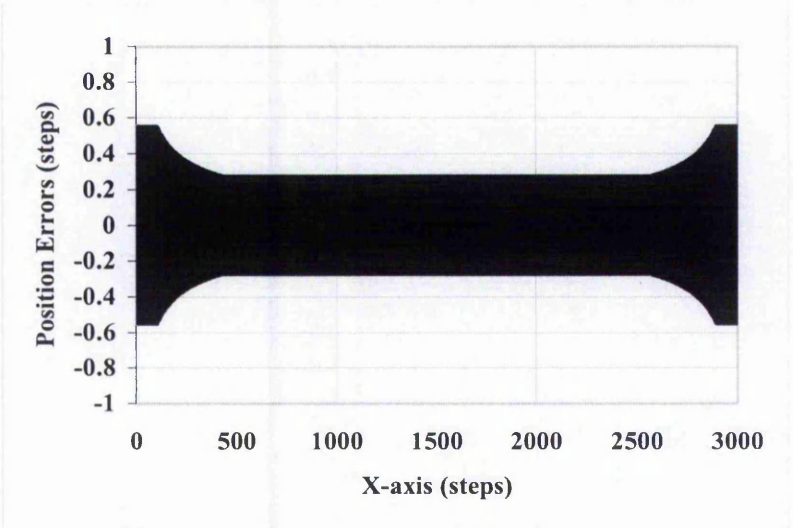


Figure C-5: Error Plots of New Half Step Linear Interpolation with Linear Acceleration (Constant Rate First Order Simulation) shown in Figure 6.50. The largest position error here is 0.55.

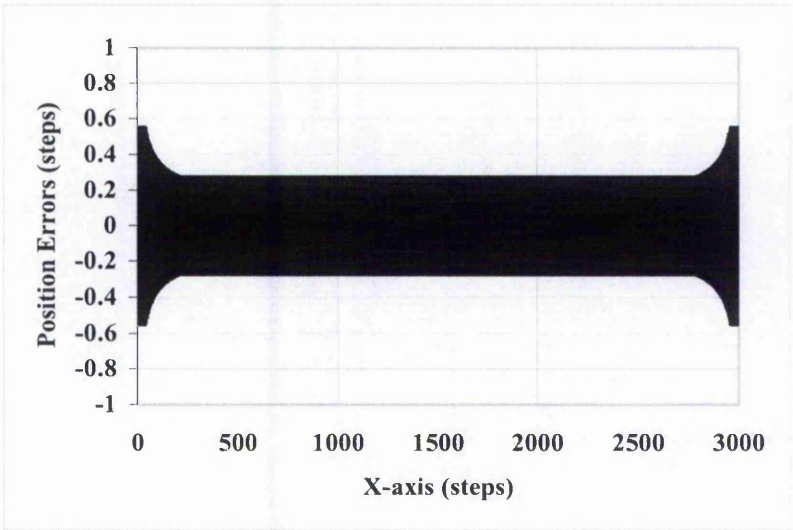


Figure C-6: Error Plots of New Half Step Linear Interpolation with Parabolic Acceleration (Constant Rate First Order Simulation) shown in Figure 6.61. The largest position error here is 0.55.

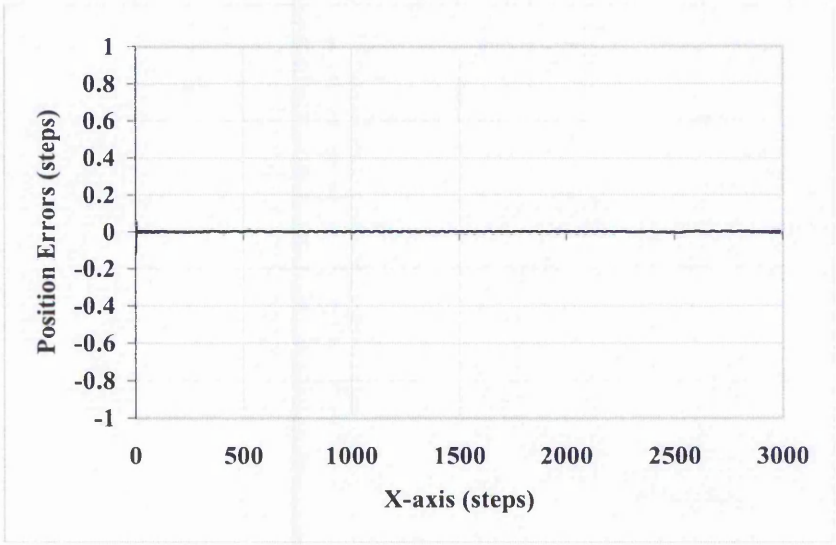


Figure C-7: Error Plots of New Half Step Linear Interpolation with Linear Acceleration (Second Order Simulation) shown in Figure 6.51. The largest position error here is 0.11.

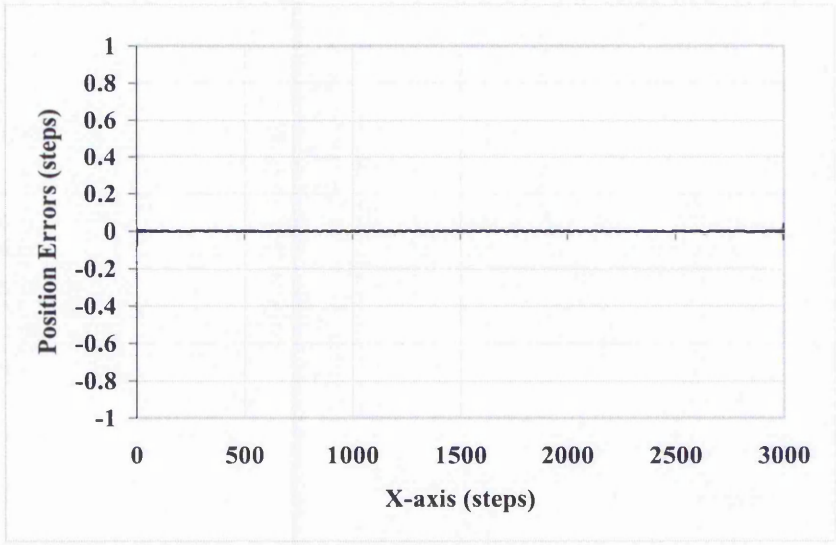


Figure C-8: Error Plots of New Half Step Linear Interpolation with Parabolic Acceleration (Second Order Simulation) shown in Figure 6.62. The largest position error here is 0.07.

C.2 Circular Arc

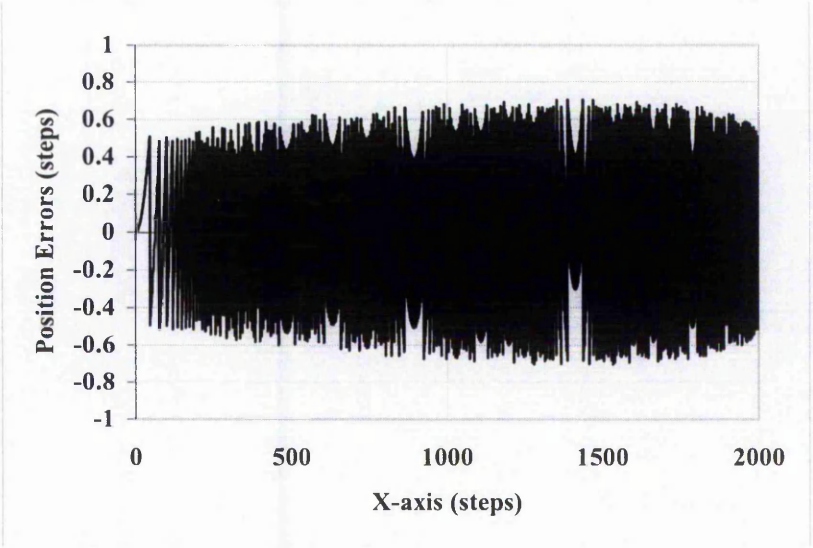


Figure C-9: Error Plots of New Half Step Circular Arc Interpolation with Linear Acceleration (Zero Order Simulation) shown in Figure 6.53. The largest position error here is 0.71.

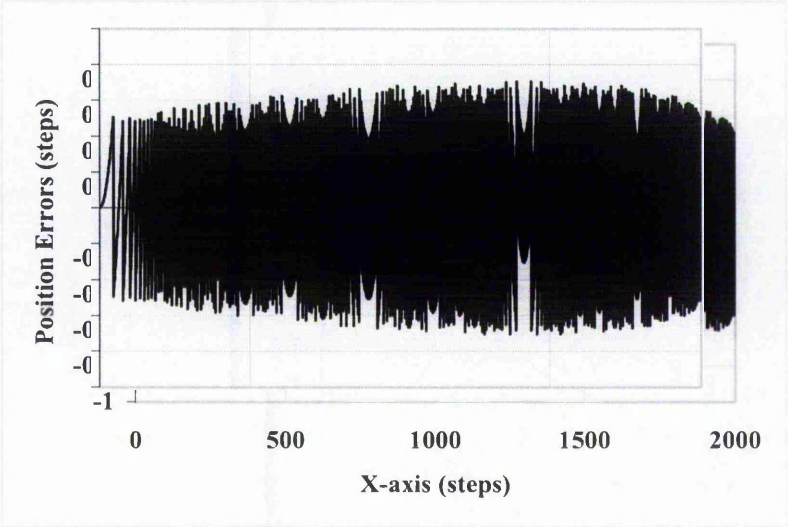


Figure C-10: Error Plots of New Half Step Circular Arc Interpolation with Parabolic Acceleration (Zero Order Simulation) shown in Figure 6.64. The largest position error here is 0.71.

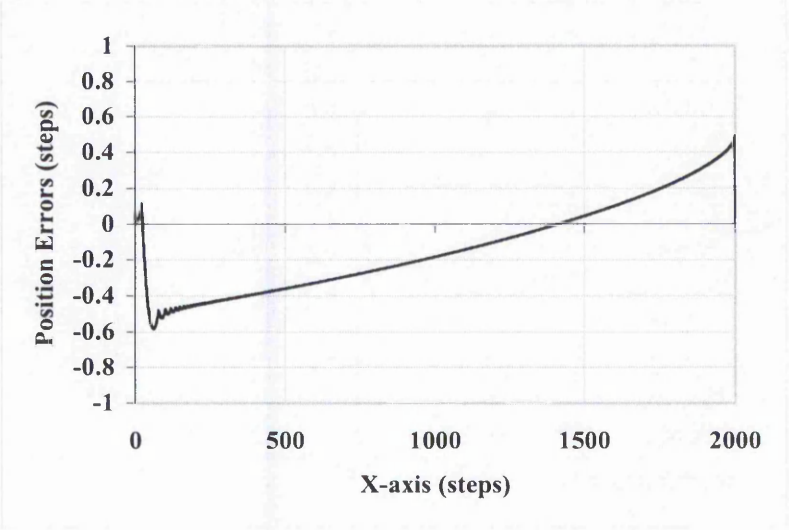


Figure C-11: Error Plots of New Half Step Circular Arc Interpolation with Linear Acceleration (Varying Rate First Order Simulation) shown in Figure 6.54. The largest position error here is 0.59.

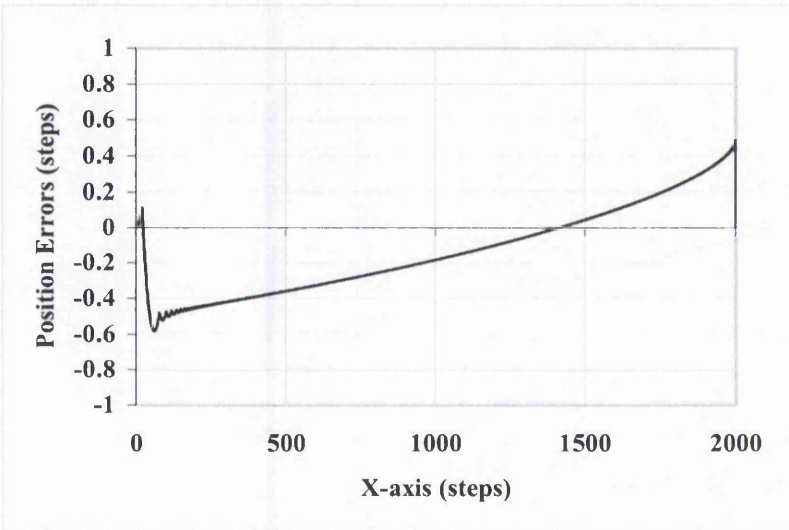


Figure C-12: Error Plots of New Half Step Circular Arc Interpolation with Parabolic Acceleration (Varying Rate First Order Simulation) shown in Figure 6.65. The largest position error here is 0.58.

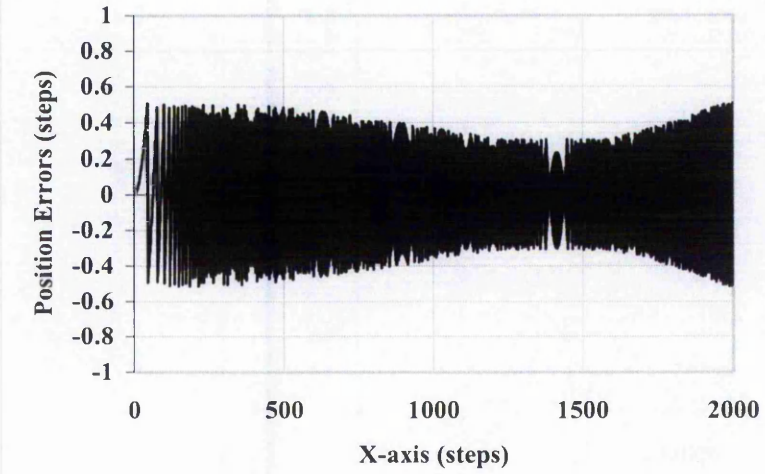


Figure C-13: Error Plots of New Half Step Circular Arc Interpolation with Linear Acceleration (Constant Rate First Order Simulation) shown in Figure 6.55. The largest position error here is 0.51.

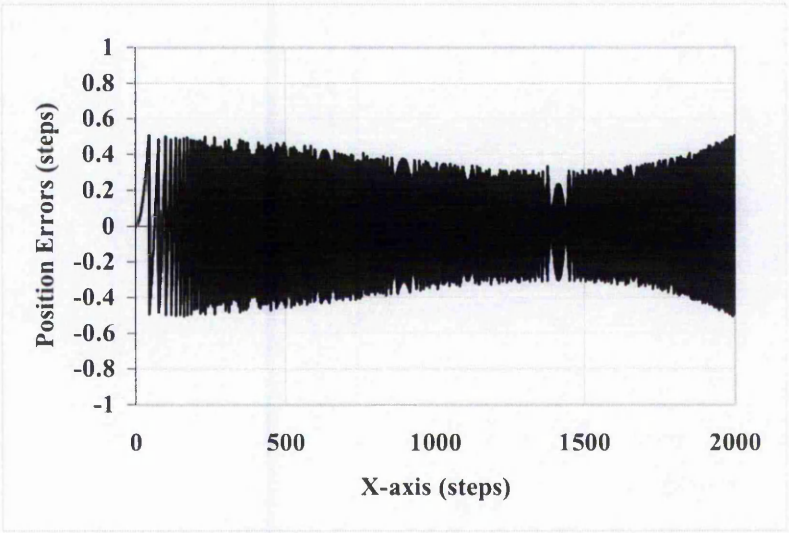


Figure C-14: Error Plots of New Half Step Circular Arc Interpolation with Parabolic Acceleration (Constant Rate First Order Simulation) shown in Figure 6.66. The largest position error here is 0.50.

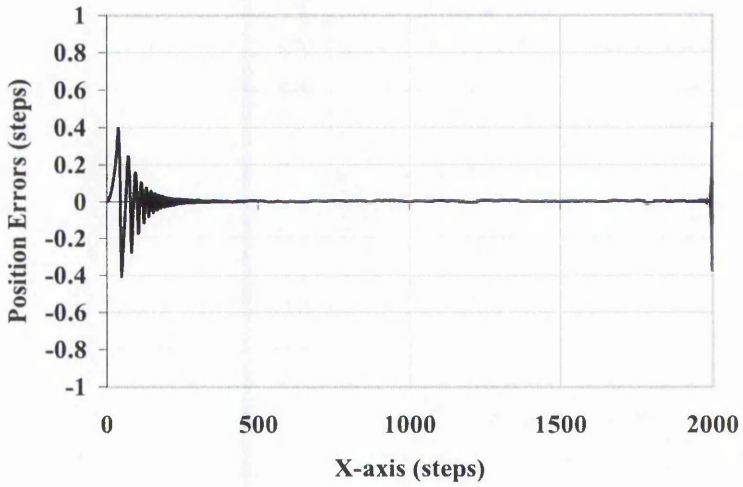


Figure C-15: Error Plots of New Half Step Circular Arc Interpolation with Linear Acceleration (Second Order Simulation) shown in Figure 6.56. The largest position error here is 0.42.

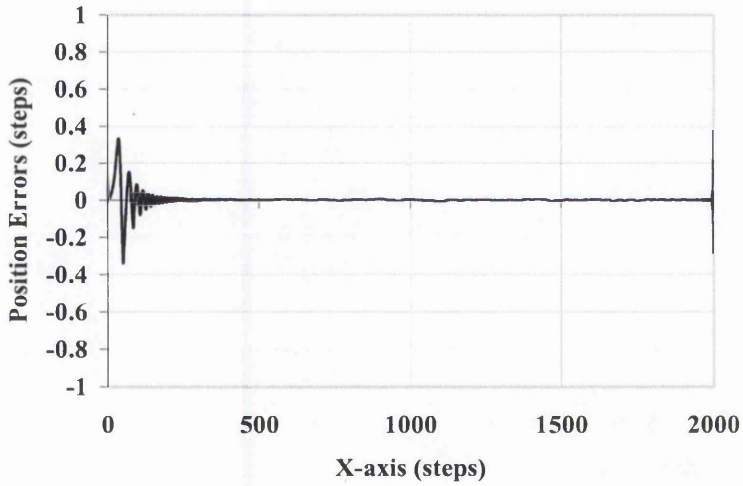


Figure C-16: Error Plots of New Half Step Circular Arc Interpolation with Parabolic Acceleration (Second Order Simulation) shown in Figure 6.67. The largest position error here is 0.37.

Appendix D: Published Papers

The Author of this thesis has attended two conferences and published the following papers:

- "Parameter-Based Algorithms for Interpolation with Stepper Motors",
  Yuan K. Chow, Janet F. Poliakoff, Peter D. Thomas,
  EPSRC 'PREP 2002' Conference for Postgraduate Researchers in Electronics, Photonics, Communications and Software

- "Interpolation and Acceleration Algorithms for Stepper Motors – A Parametric Approach",
  Yuan K. Chow, Janet F. Poliakoff, Peter D. Thomas,
  8th IEEE International Conference on Methods and Models in Automation and Robotics,
  2-5 September 2002,
  Szczecin, Poland

PARAMETER-BASED ALGORITHMS FOR INTERPOLATION WITH STEPPER MOTORS

Yuan K. Chow, Janet F. Poliakoff, Peter D. Thomas
Department of Computing and Mathematics, The Nottingham Trent University, Nottingham

Key words to describe the work: Computer Numerical Control (CNC), Interpolation, Stepper motor, Parametric curve, Smooth motion

Key Results: Algorithms for smooth path following have been developed for stepper motor controlled machining. These algorithms maintain the desired speed while avoiding possible jerks on individual axes, thus reducing both vibrations and deviation from the required path.

How does the work advance the state-of-the-art?: Parameter-based synchronisation enables individual axes to be synchronised without interpolating particular intermediate points. The high-speed calculations needed are available with recent developments in digital signal processors.

Motivation (problems addressed): Most existing stepper motor control systems suffer from jerky motion because of the interpolation and acceleration algorithms used. Jerky motion on individual axes can often cause machine vibrations. Another problem of these control systems is with the positional errors in the path, which are a result of the interpolation algorithms used. These positional errors can then be exacerbated by any machine vibrations. A path is normally required to be machined at a predefined constant speed. However, this is not easily achievable because the resultant speed of the machine tends to vary with the path direction.

Introduction

Smooth continuous path motion has always been of major importance in machine tool control for continuous motion of the machine tool (e.g. cutter) around a particular shape. Not only the actual path but also the speed of the machine tool needs to be controlled [1]. Unfortunately, smooth motion along the path is not easily achievable because of the interpolation and acceleration algorithms used. Interpolation is used to generate the path between two defined points by sending pulses to the stepper motors. Acceleration algorithms enable the motion to begin and end smoothly without stalling [2].

Existing interpolation algorithms

In many machining systems for complex paths, the path is first approximated by a lower degree curve (e.g. line and circular arc) and interpolation is performed on the latter. The interpolation algorithms normally used are the Digital Differential Analyser (DDA) and Search-Step. The DDA algorithm is able to follow the curve with, on average, the desired speed but the generated path deviates from the required path [3]. The Search-Step algorithm generates interpolation steps according to the direction and the geometry of the required path, resulting in higher accuracy path following [4]. However, the resultant speed varies considerably depending on the direction of the path. Furthermore, the speed in individual axes with both methods is likely to be jerky.

New interpolation algorithms

New line and arc interpolation algorithms have been developed to take all the above problems into consideration. The new algorithms generate pulses on each axis, allowing the machine to follow the required path according to the geometry of the path, while keeping the motion at the required speed. This is achieved by using a parameter to synchronise individual axes without interpolating particular intermediate points. For both lines and arcs, distance along the curve has been chosen as the parameter. In addition, a novel approach of calculating the timing for every individual pulse is employed. These timings are generated with respect to the geometry of the desired path.

Evaluation of algorithms

A simulation platform has been developed with two types of simulation of the positional errors of the machine in 2D, which we have named the Varying Linear and Constant Linear. Axis speed is simulated using the time intervals between pulses. These simulations give an indication of the results from a real machine.

Results

The DDA and Search-Step algorithms have been used as a comparison to evaluate the performance of the new algorithms. As an example of the evaluation, one line and one arc were chosen to interpolate at speed 0.08m/s and step size 0.01mm. The maximum positional errors obtained were calculated in each case. The line was from (0,0) to (0.3mm,0.2mm). The circular arc was interpolated from (0.2mm,0) to (0,0.2mm) with the arc centre at (0,0).

Table 1. Maximum positional errors

Simulation type	Interpolation algorithm	Maximum positional errors (μm)	
		Linear	Arc
Varying Linear	DDA	2.80	8.81
	Search-Step	2.77	3.96
	New	1.39	3.89
Constant Linear	DDA	5.55	12.13
	Search-Step	5.55	10.00
	New	2.77	3.96

In both cases, the new algorithms have reduced the maximum positional errors, as can be seen in Table 1. In theory, the speed on the X-axis for a line or arc would be constant or part of a sine wave, respectively. For both linear and circular arc interpolation, the motion of each axis were found to be jerky when the DDA and Search-Step methods are employed, as shown in Figs 1 and 2 for the case of circular arc. On the other hand, for the new linear interpolation, the speed on each axis is constant. For the new circular arc interpolation (Fig 3), the speed on the X-axis increases smoothly. Therefore, in both cases, it is less likely that any undesirable vibrations of the machine will be caused.

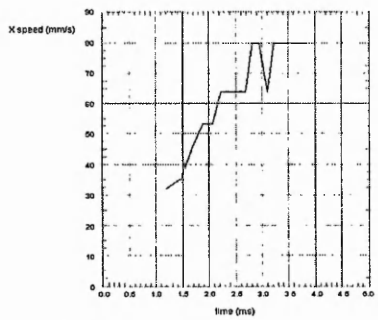


Fig 1. Speed on X-axis for DDA arc

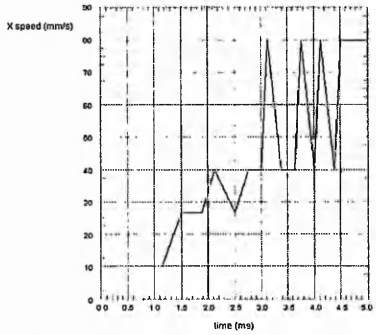


Fig 2. Speed on X-axis for Search-Step arc

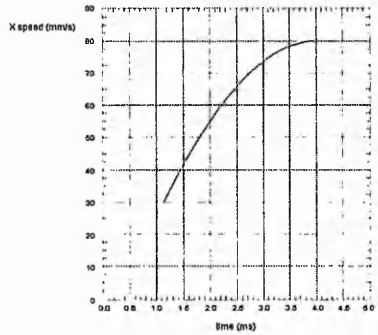


Fig 3. Speed on X-axis for new arc

Conclusions

New interpolation algorithms have been developed which employ a novel approach of calculating the timing for every individual pulse using a parameter to synchronise the different axes. These timings are generated with respect to the geometry of the desired path, thus reducing the likelihood of sudden jumps in motor speed. In addition, the new algorithms are able to follow the desired shape more closely. Further work involves the development of acceleration algorithms to make possible high-speed machining.

References

[1] Koren Y., "Computer Control of Manufacturing Systems", Mc-Graw-Hill International Student Edition, 1983, pp 11-12

[2] Kim D., Song J., Kim S., "Dependence of Machining Accuracy on Acceleration/Deceleration and Interpolation Methods in CNC Machine Tools", IEEE-IAS Annual Meeting, USA, 1994, pp 1898-1905

[3] Lim F., Wong Y., Rahman M., "Circular Interpolators for numerical control: A comparison of the modified DDA 11techniques and an LSI interpolator", Computers In Industry, 1992, Vol. 18, No. 1, pp 41-52

[4] Papaioannou G., "Interpolation Algorithms for Numerical Control", Computers In Industry, 1979, No. 1, pp 27-40

MMAR 2002 2 – 5 September 2002, Szczecin, Poland
8th IEEE International Conference on Methods and Models in Automation and Robotics

INTERPOLATION AND ACCELERATION ALGORITHMS FOR STEPPER MOTORS – A PARAMETRIC APPROACH

YUAN K. CHOW†, JANET F. POLIAKOFF‡, PETER D. THOMAS*

The Nottingham Trent University, Department of Computing and Mathematics, Nottingham, NG1 4BU, UK,
†yuan.chow@ntu.ac.uk, ‡janet.poliakoff@ntu.ac.uk, *peter.thomas@ntu.ac.uk

Abstract. In many applications of CNC machining, precise path following is an important requirement. Moreover, this machining has to be done smoothly and at a constant surface speed. Most machining systems use closed servomotor architecture but stepper motors are an inexpensive alternative. This paper presents our improved interpolation and acceleration algorithms for use with such an open-loop stepper motor system. The new interpolation algorithms are based on the use of a parameter derived from the path geometry and generate pulses with appropriate timings for the stepper motors. The new algorithms have been evaluated against existing linear and circular arc interpolation algorithms. A new parabolic acceleration technique has also been developed for use with the new interpolation algorithms.

Key Words. Computer Numerical Control (CNC), interpolation, stepper motor, smooth motion, parabolic acceleration.

1. INTRODUCTION

Precise path following and smooth continuous path motion are of major importance in machine tool control for CNC machines. Not only the actual path but also the surface speed of the machine tool needs to be controlled as it moves round a given shape [1]. The movement of the machine head along the different axes can be controlled by stepper motors or servomotors. In both cases the motion is controlled by sending appropriate control pulses to the motors but in servomotors systems feedback allows the control signals to be adjusted according to the current position of the end effector. Stepper motors, however, normally operate without feedback and this paper describes algorithms aimed at improving the motion of such stepper motor-driven machines. The machining of the required shape is achieved by adjusting the speed of the stepper motors on the different axes. Interpolation involves generating the path by sending pulses to the motors. Acceleration algorithms enable the motion to begin and end smoothly without stalling, which is needed for high-speed machining.

Stepper motors are popular because of the simplicity of their interface requirements and low costs. However, they are less suitable for very high-speed machining because of their dynamic behaviour using existing algorithms. When used within multi-axis continuous path control systems, stepper motors have three disadvantages: jerky motion, errors in position and varying surface speed. Previous work at Nottingham Trent University has addressed the problems by introducing a variable pulse algorithm with a look back / look ahead feature [2]. This helps to smooth and overcome the deficiencies of the motion generated by the existing interpolation algorithms. Further study of machine performance has led to the introduction of a filter process, which improves path smoothness at the expense of some degradation of interpolator accuracy [3]. Thus, both techniques involve smoothing the trains of pulses after they have been generated. This paper addresses the problem from a more fundamental viewpoint, because it involves generating pulse timings which are smooth initially.

2. PREVIOUS INTERPOLATION ALGORITMS

In many machining systems for complex paths, the path is first approximated by a lower degree curve, such as line and circular arc, and interpolation is performed on this curve. Previous interpolation algorithms include the Digital Differential Analyser (DDA) and the Search-Step algorithm. The principle operation of the DDA interpolator is the performing of digital integration on velocity to obtain the position reached at fixed intervals of time (Δt) [4]. This is expressed by:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{v}_k \Delta t \qquad (1)$$

where $\mathbf{x}_i$ is the position at the start of the i^{th} interval and $\mathbf{v}_i$ is the corresponding velocity at the start of that interval. Thus, over the interval Δt motion is generated in the direction of the tangent to the curve at the start of the interval. This results in some deviation from the required path. Since the velocity is involved in the interpolation algorithm, the path can be machined at the required speed.

The Search-Step method relies on the local geometry of the curve to generate each step. This method uses curves defined in implicit form, where each point on the curve satisfies an equation of the form $f(x,y)=0$. Points lying off the curve have $f(x,y)\neq 0$ and the value of $f(x,y)$ is proportional to the error in position [5]. The Search-Step algorithm finds the next interpolated point by choosing the one closest to the desired curve, i.e. the one for which $f(x,y)$ is closest to 0. Unlike the DDA, Search-Step does not control the surface speed fully during interpolation, because the resultant speed can vary, depending whether the motion is on just one axis or more than one. Therefore, for interpolation of a 2D circular arc Search-Step, there can be a speed variation up to a factor of $\sqrt{2}$.

3. INVESTIGATED PROBLEMS

As explained above, there are three main problems with the previous interpolation algorithms: jerky motion, error in position and varying surface speed. Jerky motion is caused by vibrations, which often occur when the time between pulses to a stepper motor varies erratically and such small effects may then be exacerbated by the resonant frequencies of the system. Errors in position (measured by the distance between the actual path and the desired path) can be caused by jerky motion but can also result from approximation of curves by facets. Variations in actual surface speed from the required speed (or feed rate) can result from interpolation algorithms and are likely to be exacerbated when vibrations occur.

4. THE NEW INTERPOLATION ALGORITHMS FOR STEPPER MOTORS

New linear and circular arc interpolation algorithms have been developed which reduce the above problems. On a particular axis the pulses are generated with timings which allow the machine to follow the required path according to the geometry of the path, while moving at the required speed. Unlike the previous methods, we have used a parameter-based method to synchronise the individual axes. For both lines and arcs, distance along the curve has been found to be the most appropriate parameter, because distance can easily be related to the required speed. These pulse timings are calculated using the path geometry and the required surface speed. The idea is to imagine a point travelling along the curve at the required speed and send a pulse to the motor for a given axis every time a step in that direction is completed. For 2D linear interpolation, a straight linear can be expressed in a parametric form as

$$\begin{aligned} x(s) &= x_{start} + s\cos\theta\ , \\ y(s) &= y_{start} + s\sin\theta\ , \end{aligned} \qquad (2)$$

where s is the distance along the line, (x_{start}, y_{start}) is the start point and θ the (constant) angle between the line and the x-axis. If L is the length of one motor step, then the distance along the curve of the n^{th} step on each of the axes is given by:

$$s_x(n) = \frac{nL}{\cos\theta} \text{ and } s_y(n) = \frac{nL}{\sin\theta}. \qquad (3)$$

From these equations the corresponding timings of the n^{th} pulses on each of the axes for machining at constant surface speed V are given by:

$$\begin{aligned} t_x(n) &= \frac{s_x(n)}{V} = \frac{nL}{V\cos\theta}, \text{ and} \\ t_y(n) &= \frac{s_y(n)}{V} = \frac{nL}{V\sin\theta} \end{aligned} \qquad (4)$$

A circular arc in the first quadrant of radius R, centre (x_{centre}, y_{centre}) and start angle α_{start} is expressed in a parametric form as:

$$\begin{aligned} x(s) &= x_{centre} + R\cos\left(\alpha_{start} \pm \frac{s}{R}\right), \\ y(s) &= y_{centre} + R\sin\left(\alpha_{start} \pm \frac{s}{R}\right) \end{aligned} \qquad (5)$$

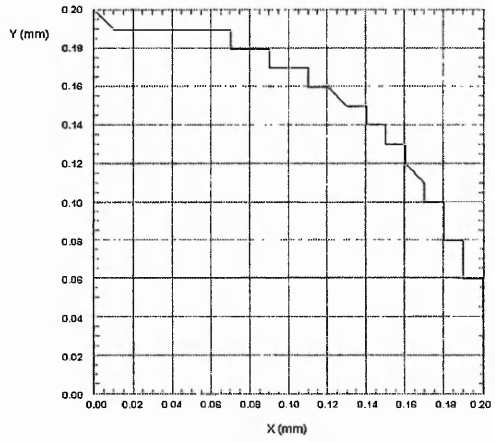
(with the plus sign for anticlockwise and the minus sign for clockwise). This time we have for the n^{th} steps on each of the axes:

$$s_x(n) = \pm R\left[\cos^{-1}\left(\cos\alpha_{start} \mp \frac{nL}{R}\right) - \alpha_{start}\right],$$

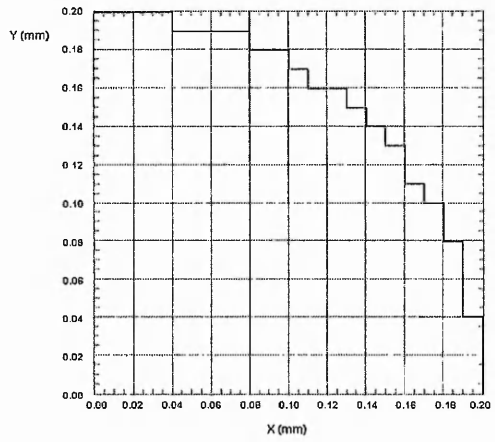
$$s_y(n) = \pm R\left[\sin^{-1}\left(\sin\alpha_{start} \pm \frac{nL}{R}\right) - \alpha_{start}\right] \qquad (6)$$

As in the case for linear interpolation, the pulse timings can then be calculated for motion at constant surface speed. The expressions used above for circular arc interpolation are only suitable for first quadrant circular arc interpolation; for the other three quadrants, the expression is similar.

During the simulation (see next section) of the new circular arc interpolation for a quarter of a circle, it was found that the error in position can be close to one complete step. This error can be reduced, however, if the pulse timing is adjusted so that the pulse is sent once the half way point between two steps has been reached, rather than when a whole step is completed. This "half step" technique can be seen to improve on the circular interpolation as shown in Fig. 1. In addition, if the end position is not a complete step, the maximum error at the end will be within ± 0.5 of a step.



(a)



(b)

Fig. 1. Instantaneous Step Simulation of (a) the new arc interpolation and (b) the new arc interpolation incorporating the Half Step technique

5. SIMULATION OF INTERPOLATION ALGORITHMS

The new interpolation algorithms have been compared with the DDA and Search-Step algorithms using simple simulation. Two main areas have been chosen for evaluation, (1) errors in position, (2) speed on the x and y-axes, and three simulation methods have been used. None of these is very realistic but, between them, they allow us to compare the different algorithms.

The first method, Instantaneous Step Simulation, does not take into consideration the response time of the stepper motor; the stepper motor is assumed to move to the desired position instantaneously as soon as a command pulse is received. Thus, this simulation is very far from the motion of a real machine but it does allow the order of pulse timings on the two axes to be seen, as shown in Fig. 1.

Smoother motion is obtained by using the second method, Varying Linear Simulation, which assumes that the motion between pulses varies linearly with time and that the stepper motor will have completed moving one step only when the next command pulse is received. This is also not entirely realistic but simulates to some extent the smoothing effect of the motor and drive circuits.

The third method, Constant Linear Simulation, assumes that: (i) the motion after each pulse is at fixed (constant) speed until the step has been completed before remaining stationary until the next pulse; (ii) the stepper motor has completed the movement before the next pulse. Therefore, an additional parameter is needed for this fixed constant speed, which simulates the stepper motor response.

For simulation of axis speed we have used the Varying Linear method, i.e. the speed between two pulses is assumed to be inversely proportional to the time between the two pulses. A real motor, on the one hand, will allow some smoothing but, on the other hand, will often suffer from vibrations, making the situation worse.

As an example of the simulation one line and one arc were interpolated at surface speed 0.08m/s with step size 0.01mm. The line was from (0,0) to (0.3,0.2) and the arc was from (0.2,0) to (0,0.2) with the arc centre at (0,0) (all in mm.). For all four interpolation methods, in all three simulations, the new half-step algorithms have the smallest value for the maximum error in position, as can be seen from Table 1.

Ideally, the speeds on both axes for a line would be constant, while for an arc they would each be part of a sine wave. For both linear and circular arc interpolation, we have found that the motion of each axis is jerky when the DDA and Search-Step methods are employed. On the other hand, for the

new interpolation algorithms, the speeds on the two axes were close to the ideal, i.e. constant or changing smoothly, respectively. Fig. 2 shows the results of the *x*-axis speed simulation for a circular arc for three of the circular arc interpolation algorithms. For the new half step circular arc interpolation (Fig. 2(c)), the speed on the *x*-axis increases smoothly. Therefore with the new algorithms it is less likely that there will be any undesirable vibrations of the machine.

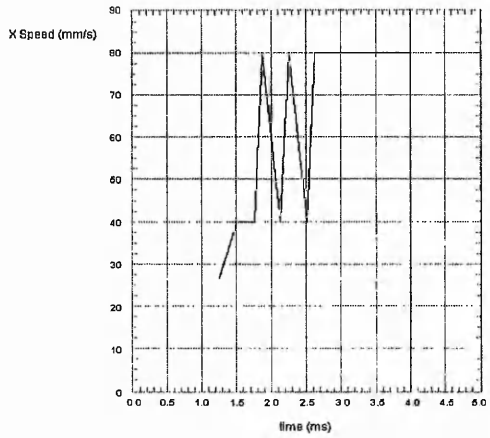
Table 1. Maximum error in position for the four interpolation algorithms. (Step size = 0.01mm.)

Simulation type	Interpolation Algorithm	Maximum error in position (μm)	
		Line	Arc
Instantaneous Step	Search-Step	5.55	10.00
	DDA	5.55	8.09
	New	5.55	9.74
	New (Half)	**5.55**	**6.16**
Varying Linear	Search-Step	2.77	3.96
	DDA	5.55	6.70
	New	2.77	10.76
	New (Half)	**1.39**	**3.89**
Constant Linear	Search-Step	5.55	10.00
	DDA	5.55	8.09
	New	4.17	9.74
	New (Half)	**2.77**	**3.96**

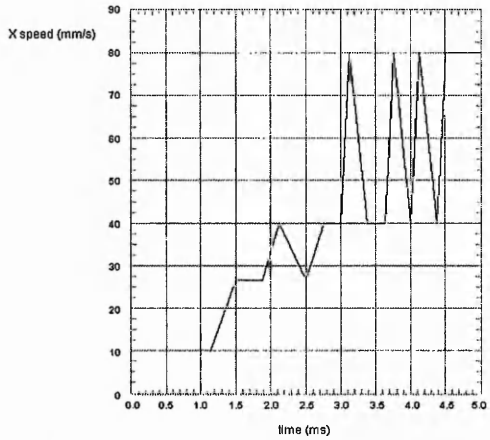
6. PREVIOUS STEPPER MOTOR ACCELERATION ALGORITHMS

Two common types of acceleration technique used for high-speed machining are linear and parabolic acceleration, in which the speed changes either linearly or parabolically with time. The minimum speed of the motor depends on the rotor and load inertia [6] and this will also be the speed at which the motor can start moving from rest.

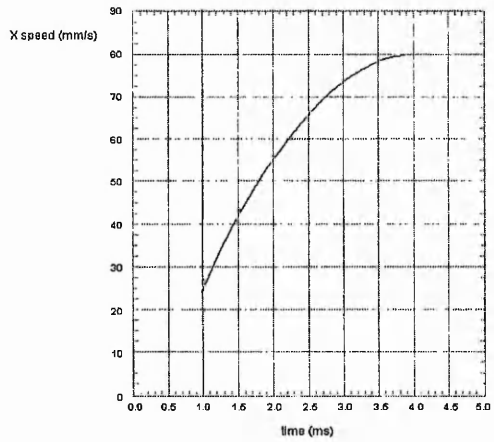
Linear acceleration results in slow acceleration and much of the available torque is not utilised [6]. A parabolic acceleration algorithm allows a higher acceleration at low motor speed combined with a lower rate at high speed. With this method, much more of the available motor torque can be utilised and the stepper motors can therefore be used at higher speeds. Moreover, Dong-Il Kim, et al. [7] have shown that machining accuracy is improved with parabolic acceleration in comparison with linear acceleration. This is likely to be because any linear acceleration algorithm involves sharp discontinuities in the acceleration, which tend to cause increased vibration and overshoot.



(a)



(b)



(c)

Fig. 2. Speed on the x-axis for interpolation of an arc with (a) DDA, (b) Search-Step and (c) the new half step algorithm.

7. THE NEW PARABOLIC ACCELERATION ALGORITHM FOR STEPPER MOTORS

A new parabolic acceleration algorithm has been developed for use with the new interpolation algorithms. The output is pulse timings which allow the stepper motors to accelerate or decelerate smoothly. The equations for speed during the acceleration and deceleration phases were developed by Palmin et al. [6] as follows:

$$V = pt^2 + qt + V_0 \tag{7}$$

where $p = \dfrac{V_0 - V_m}{T^2}$ and $q = -2pT$

V_0 is the minimum speed, V_m is the maximum speed and T is the time taken during acceleration or deceleration. Palmin et al. tried to calculate the timing t_n for every command pulse but used an approximation by assuming that

$$V_n = V_{n-1} + a_{n-1}(t_n - t_{n-1}) \tag{8}$$

where V_n is the speed at time t_n and a_n is the acceleration at time t_n.

In order to maintain the desired shape during path following, the new parabolic acceleration and deceleration needs to be applied to the surface speed and not the speed for an individual axis. From equation 7, the distance travelled, s, can be calculated as follows:

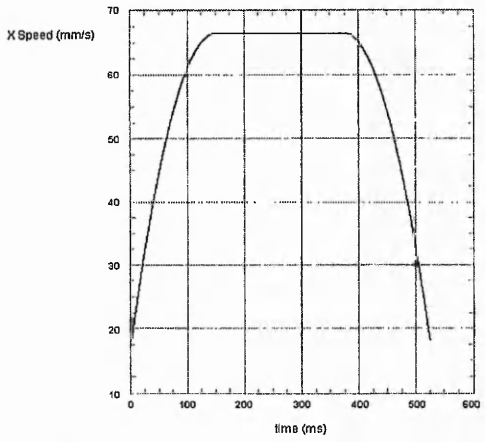
$$s = \int V dt = \frac{pt^3}{3} + \frac{qt^2}{2} + V_0 t \tag{9}$$

The distance, s, for every axis step movement is obtained by interpolation from equations 3 and 6. To determine each pulse timing, the cubic equation 9 needs to be solved, which we have implemented using the Newton-Raphson iteration.

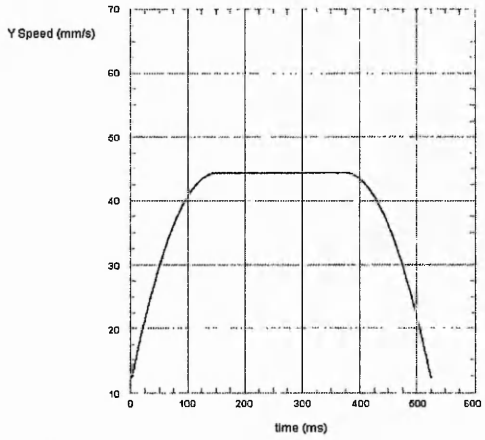
8. SIMULATION OF ACCELERATION AND DECELERATION

For the simulation of acceleration and deceleration for a straight line and an arc we have chosen desired surface speed 0.08 m/sec, with step size 0.01 mm, as before, and both acceleration time and deceleration time set to 0.15 sec. Longer paths were chosen this time to allow for acceleration up to the full speed followed by deceleration.

Fig. 3 shows the speeds on the two axes for the straight line from (0,0) to (30,20) (in mm), which are the same apart from a constant factor. The arc was from (20,0) to (0,20) with centre (0,0) (in mm) and Fig. 4 shows the speeds on two axes in this case.



(a)



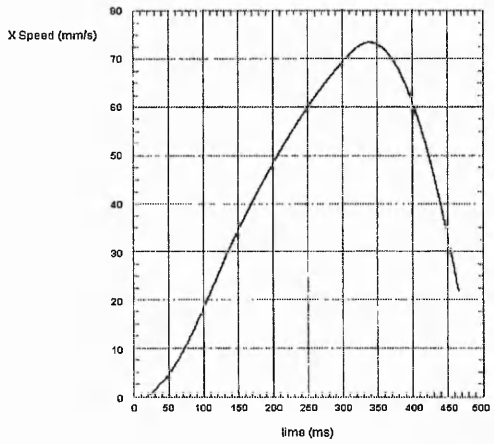
(b)

Fig. 3. Parabolic Acceleration and Deceleration for linear interpolation for (a) x-axis and (b) y-axis.

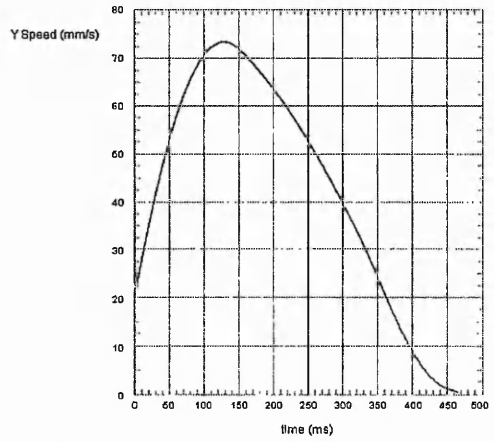
With the parabolic acceleration algorithm, the machine is still able to follow the required path. The maximum errors in position resulting from the new parabolic and interpolation algorithms are shown in Table 2 below. It can be seen that new linear and circular interpolation with half-step technique are able to generate a path that matches the required one within an error of less than one motor step.

Table 2. Maximum errors in position for the new interpolation and acceleration algorithms. (Step size = 0.01mm.)

Simulation type	Full/Half Step Interpolation	Maximum error in position (μm)	
		Line	Arc
Instantaneous Step	Full	8.63	10.00
	Half	**5.86**	**7.31**
Varying Linear	Full	2.90	13.00
	Half	**1.48**	**5.75**
Constant Linear	Full	5.55	10.4
	Half	**5.55**	**5.33**



(a)



(b)

Fig. 4. Parabolic Acceleration and Deceleration for circular arc interpolation for (a) x-axis and (b) y-axis.

In these examples the simulations shown have assumed that the time for deceleration is the same as the time for acceleration. However, there is no need for them to be the same and in practice deceleration time can almost certainly be shorter.

9. DISCUSSION

The parabolic acceleration algorithm is able to utilise much more of the available torque from the stepper motor than the linear acceleration algorithm. However, further investigation is needed into how well this algorithm utilises the torque. Measurement of the system can identify the maximum acceleration that is possible at different speeds. This may lead to the development of improved acceleration algorithms.

10. CONCLUSIONS

New interpolation algorithms have been developed which employ a novel approach of calculating the timing for every individual pulse using the distance along the curve as a parameter to synchronise the different axes. The pulse timings are generated with respect to the geometry of the desired path, thus reducing the likelihood of vibrations and sudden jumps in motor speed. In addition, the new algorithms are expected to follow the desired shape more closely. To enable the developed interpolation algorithms to be used for high-speed machining, a parabolic acceleration algorithm has also been developed. The simulation results show that the new interpolation and acceleration algorithms are able to follow the required path closely. Further investigation is needed to evaluate the improvements achieved in practice.

11. REFERENCES

1. Koren Y., "Computer Control of Manufacturing Systems", McGraw-Hill International Student Edition, 1983
2. Steiger W., Sherkat N., Thomas P., "A Stepping Motor Control Algorithm For Smooth Continuous Path Motion", Proceedings of the IEEE Int. Conf. On Control '94, University of Warwick, UK, Vol.1, No. 389, pp 816-821, 1994
3. Stout A., Thomas P.D., Orton P.A., "Improving Continuous Path Motion Using Stepper Motors", Proceedings Of The 10th European Simulation Symposium & Exhibition, The Nottingham Trent University, Nottingham, UK, 1998
4. Papaioannou S., "Interpolation Algorithms For Numerical Control", Computers In Industry, No.1, pp 27-40, 1979
5. Massory O., Koren Y., "The Direct-Search Method In CNC Interpolators", ASME Winter Annual Meeting, San Francisco, Calif., pp 2 – 8, 1978
6. Palmin S., Shlain V., "Stepper Motor Controller With Parabolic Velocity Profile Allows Maximum Torque", Control Engineering, Feb 1986
7. Kim D., Song J., Kim S., "Dependence Of Machining Accuracy On Acceleration / Deceleration And Interpolation Methods In CNC Machine Tools", IEEE – IAS Annual Meeting, pp 1898 – 1905, Denver, USA, 1994