MOTION DESIGN FOR HIGH-SPEED MACHINES

ALI KIRECCI

A thesis submitted in partial fulfilment of the
requirements of Liverpool John Moores University
for the degree of Doctor of Philosophy

February 1993

Liverpool John Moores University
School of Engineering and Technology Management
Mechanisms and Machines Group

Table of Contents

ACKNOWLEDGEMENTS

Firstly, I wish to acknowledge the invaluable support and assistance of my primary supervisor Dr. Michael J. Gilmartin. He guided me academically throughout the whole period of my research and supervised the successful preparation of my thesis. Secondly Dr. George T. Rooney gave me invaluable advice in the initial stages of my work and has helped with my computer studies.

Professor John Rees Jones, the founder of the Mechanism and Machines Group of the Liverpool John Moores University and now a consultant for the Advanced Manufacturing Technology Group of Unilever Research gave me much time for discussion and advice on several aspects of my subject for which I must express my gratitude.

All current members and research workers of the Mechanisms and Machines Group contributed to the success of my research and I wish to acknowledge their help. Similarly I have received much technical help from members of the University workshop particularly Stephen Ebbrell in making the mechanical parts of the manipulator.

Assoc. Prof. Sedat Baysec of Gaziantep University, Mechanical Engineering Department, has given me much assistance and advice in the early stages and without his help this work would never have started.

My dear wife Makbule encouraged me throughout the research and I would like to take this opportunity to thank her.

ABSTRACT

MOTION DESIGN FOR HIGH-SPEED MACHINES

by

Ali Kirecci

The dynamic performance of a programmable manipulator depends on both the motion profile to be followed and the feedback control method used. To improve this performance the manipulator trajectory requires planning at an advanced level and an efficient control method has to be used.

The purpose of this study is to investigate high-level trajectory planning and trajectory tracing problems. It is shown that conventional trajectory planning methods where the motion curves are generated using standard mathematical functions are ineffective for general application especially when velocity and acceleration conditions are included. Polynomial functions are shown to be the most versatile for these applications but these can give curves with unexpected oscillations, commonly called meandering. In this study, a new method using polynomials is developed to overcome this disadvantage.

A general motion design computer program (MOTDES) is developed which enables the user to produce motion curves for general body motion in a plane. The program is fully interactive and operates within a graphics environment.

A planar manipulator is designed and constructed to investigate the practical problems of trajectory control particularly when operating at high speeds. Different trajectories are planned using MOTDES and implemented to the manipulator.

The precise tracing of a trajectory requires the use of advanced control methods such as adaptive control or learning. In learning control, the inputs of the current cycle are calculated using the experience of the previous cycle. The main advantage of learning control over adaptive control is its simplicity. It can be applied more easily in real time for high-speed systems. However, learning algorithms may cause saturation of the driving servo motors after a few learning cycles due to discontinuities being introduced into the command curve. To prevent this saturation problem a new approach involving the filtering of the input command is developed and tested.

CHAPTER 1
INTRODUCTION

1.1 Background to the Problems

The need to increase the rate of production, to provide for quick changes in the shape or form of a product in response to market demands and the increasing need for automation in industry require conventional machinery systems (cam driven and linkage mechanisms) to be more efficient and adaptable to changes. Generally, either linkages or cams are used to obtain non-uniform motion for the end-effector of a machine to perform the operation. End-effector is defined here as a special device that is attached to the output point of a mechanism to enable it to accomplish a specific task.

Conventional mechanisms are suitable for simple motions. They usually provide a "fixed" motion with only limited variations being possible by altering the link lengths. However, in some cases a group of machines works together for some processes, such as, "packaging" where usually a pick and place mechanism operates between independent machines. Similarly there are more complex tasks for example an assembly line on which, inspections, welding, spray painting, machining and other operations are needed. Such systems require synchronization of motion between the machines, therefore the positions and sometimes velocities of these machines must be proportional

to each other. Furthermore, the speed of the system may be required to be adjusted to change the rate of production. A system with conventional machines cannot satisfy these requirements easily.

Alternative systems to these conventional systems are servo motor driven systems (programmable systems). These are highly flexible and are able to follow a required trajectory with high precision and at high speeds. But, these systems bring their own problems. Trajectory planning (motion design) and trajectory tracing (trajectory control) are the major problems of these systems.

Trajectory planning: The activity of converting the description of a desired task to a trajectory by defining time sequences of configurations of the end-effector of a manipulator between start and final positions is referred to as trajectory planning or motion design. As the trajectory is executed, the end-effector traces a curve and changes its orientation. Normally, the trajectories of programmable systems in many industrial applications are planned manually or by using ineffective methods with the result that the machines are too slow to justify their use economically because of their improper trajectories [1.1],[1.2]. Their speed and hence their productivity are limited by the performance capabilities of their actuators. Increasing actuator size and power is not the best solution because of the increased cost and power consumption. A more successful approach is to design the trajectory at an advanced level in order to increase the speed of the system and perform a given task appropriately.

Trajectory tracing: When a desired trajectory is applied to a servo-system it responds in a characteristic fashion and follows the trajectory with an error. The physical features of the actuators and the gain setting of the controller and the smoothness of the required trajectory are the main parameters that determine the response of the system. However, precise tracing of a trajectory under different payloads require some effective control techniques.

The starting point of this study is based on these two problems of programmable systems. The design of an efficient program in order to plan trajectories at advanced level is the first objective. Secondly, an experimental prototype arrangement has been set up in order to understand the issues and gain experience concerning the practical problems of trajectory tracing with a programmable system. Also, an investigation of the importance of trajectory planning on trajectory tracing at high speed is made.

1.2 Comparison of conventional and programmable systems

Although programmable systems have great advantages they are not always the best choice because of their high cost and low efficiency of energy use. The selection of the system types depends on the conditions to be satisfied, one of the most important being the complexity of the motion required to be followed by the mechanism. Two types may be considered

(i) simple motion machines
(ii) complex motion machines.

In the first group the mechanisms perform a simple process. The only critical requirement is that the position of the mechanism be specified with respect to time. They are not needed to co-operate with any other machine except when acting as master machines. Conventional mechanisms can easily satisfy these requirements with the advantages of good dynamic response at high speeds. Servo driven systems have no advantages over these systems under these circumstances. Fig.1.1 shows examples of programmable, cam driven and linkage mechanisms.

The second group includes those mechanisms which depend on the motion of other machines and the position of the product. They are required to follow a trajectory to satisfy the co-ordination condition between the machines and to manipulate the product. Cam driven mechanisms cannot satisfy these requirements since one of the common features of all the standard cam motions is that they can generate only stop-go-stop type motions. Whatever

the required velocity and acceleration boundary conditions, cam motions can only produce zero velocity and zero acceleration at the ends of segments of a trajectory therefore synchronization between several machines can be extremely difficult to achieve with cam driven mechanisms. On the other hand, satisfying even position boundary conditions can be difficult for linkage mechanisms while the inclusion of velocity and acceleration constraints can make the design impossible. It is clear that programmable mechanisms have great advantages over more conventional mechanisms for the above case.

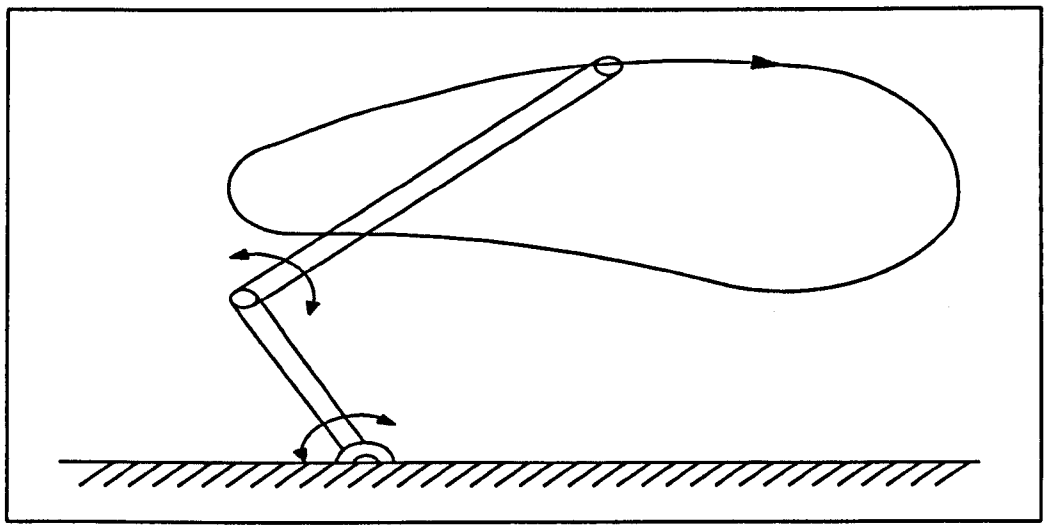
In some cases cam driven systems and programmable systems can both be suitable for an application. Detailed investigation and analysis are then necessary to make the best selection. Some of strategies to consider in making this choice are given below:

(i) Programmable systems are flexible; the path of the programmable machine can be automatically adjusted even while the system is operating. These systems are capable of producing a variety of products with virtually no time lost for change-over from one product to the next. There is minimum production time lost whilst programming the system and altering the set-up. In consequence, the system can be used to produce various combinations and schedules of product, instead of requiring that they be made in separate batches.

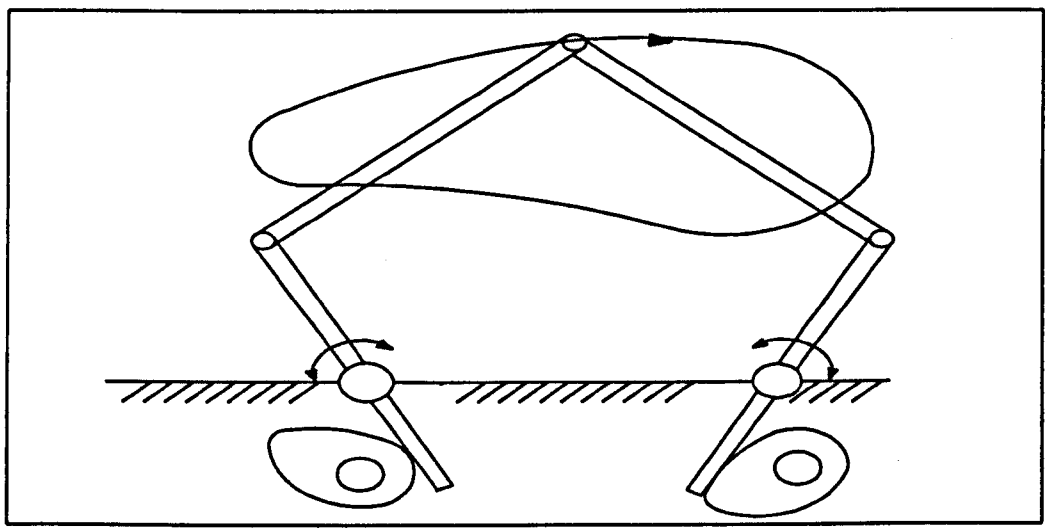
(ii) Conventional systems are cheaper and are more energy efficient [1.3].

(iii) Cam driven systems may produce inertia problems due to the masses of the cams and their necessary balancing masses. This will limit the speed of the machine in order to avoid excessive vibration, wear and shocks. On the other hand, servo-driven systems with the absence of such masses are capable of operating at higher speeds.

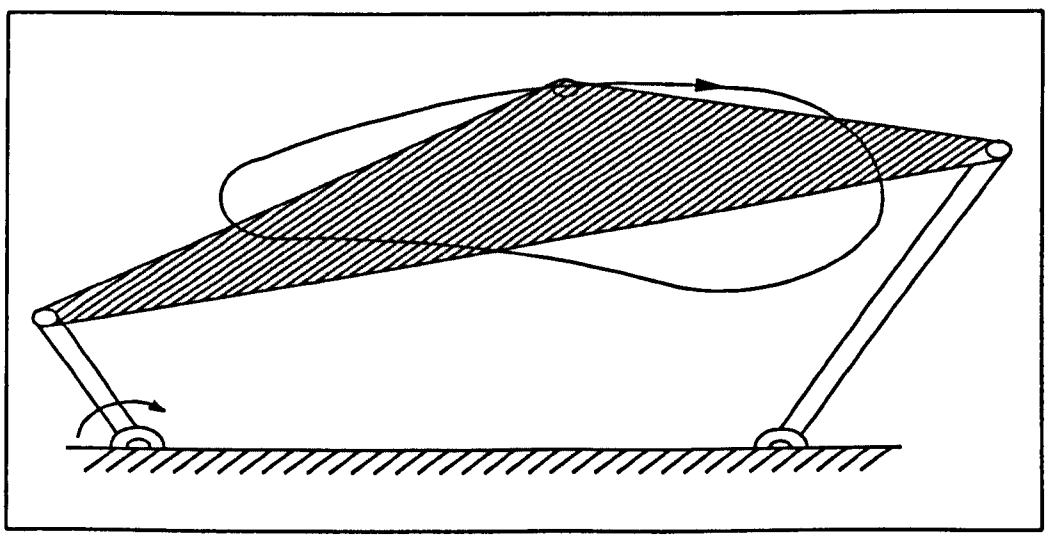
(iv) The precision of the executed trajectory depends on the cam surfaces for cam driven systems. Programmable systems can also achieve high precision resulting from the incorporation of high-resolution optical encoders.



(a) Programmable mechanism



(b) Cam–driven mechanism



(c) Linkage mechanism

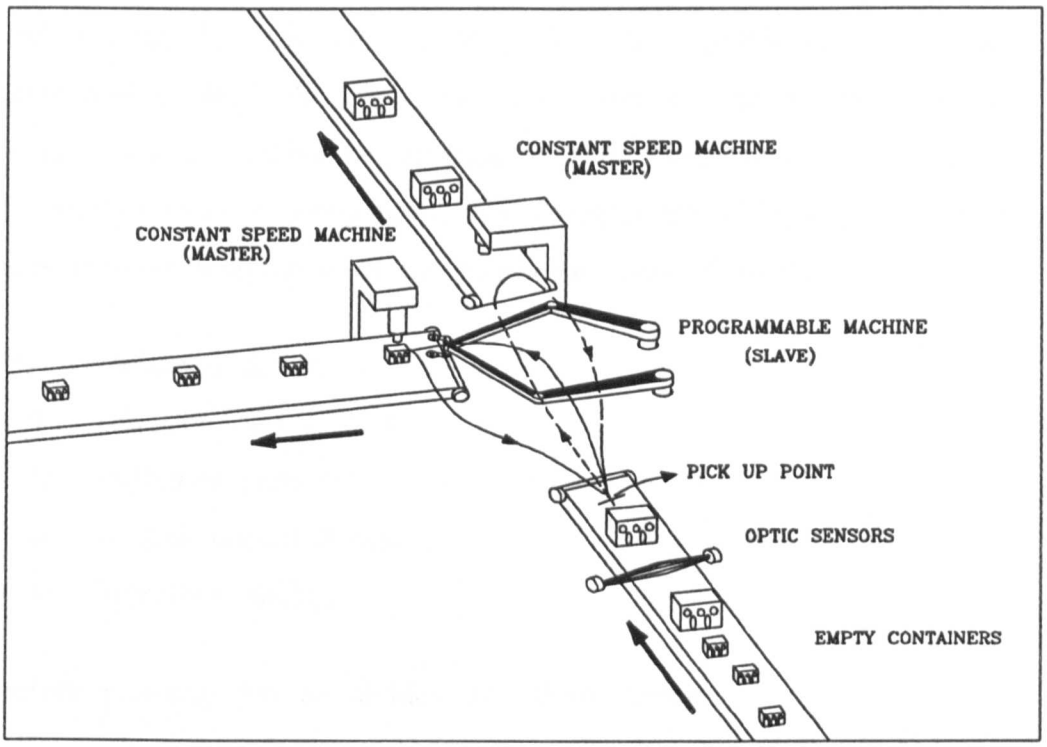
Fig.1.1. Different mechanisms to do the same job.

Fig.1.2 shows a production line which includes a programmable machine (slave) and two constant speed machines (masters) where it is required to transfer the products from a conveyor to the processing machines, but the main issue is the different distance between the products and their different sizes. In this typical example the programmable machine is required to:

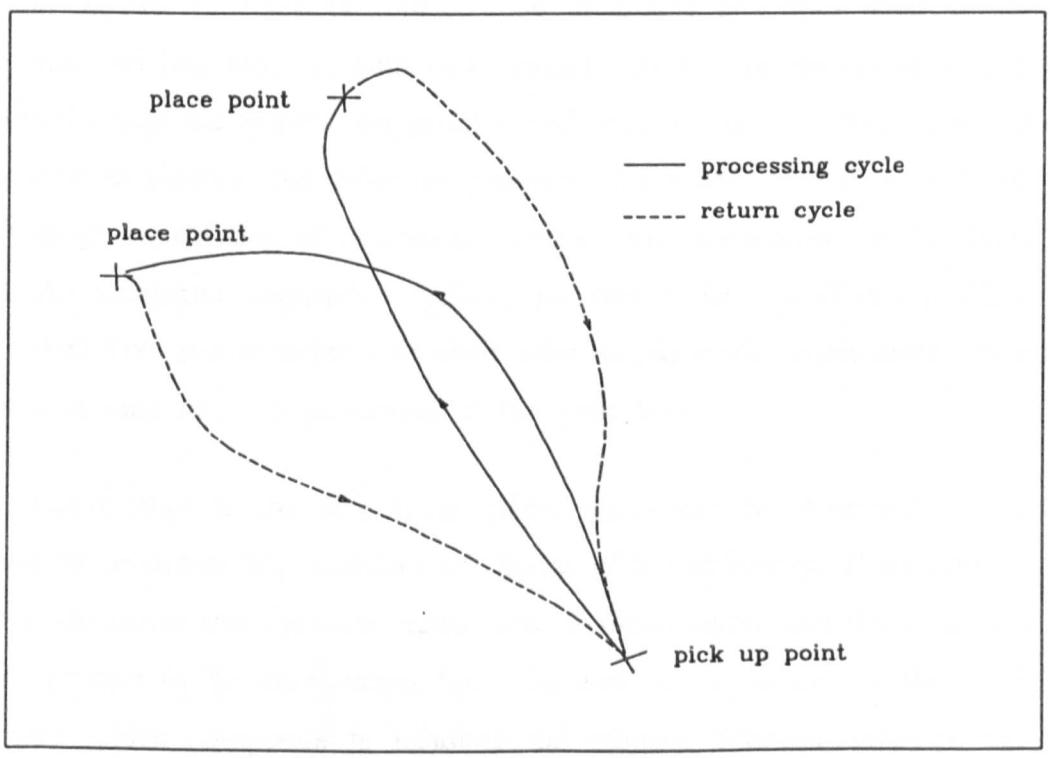
(i) determine the trajectory to be executed according to the size of the product and transfer it to the receiving machine
(ii) activate the processing machine in time for each process
(iii) adjust its speed for the return part of the trajectory to grasp the next product.

These conditions exhibit the importance of programmable systems and the need for trajectory planning. The trajectory of the mechanism must provide for co-ordination between all the machines while satisfying other requirements such as the need for smooth, continuous, and efficient motions.

The objective of this study is directed towards the trajectory planning problem especially for programmable, high-speed machines.



(a) A programmable system



(b) Two different paths for the above manipulator

Fig.1.2. A manipulator and its environment.

1.3 Programmable systems

The pace of progress in the development of high-speed machinery has increased interest in trajectory planning. However, problems of trajectory planning and its implementation remain difficult for the machine designer, particularly where machine performance is highly dependent on the reaction of the product to accelerating force. The development of high speed machine systems requires integration of the following areas of work:

(i) structural design
(ii) selection of actuators
(iii) trajectory planning
(iv) control system design
(v) trajectory tracing.

Trajectory planning can be divided into three stages:

The first stage is the decision for the level of the trajectory. Industrial manipulators have limited capability constraints since they can supply only a finite amount of force or torque. Also kinematic quantities such as acceleration and jerk must be kept to reasonable limits. The manipulation task defines the required end-effector position and orientations. Therefore, it would be logical to perform the trajectory planning at the end-effector level [1.4]. The output capabilities of actuators, however, are constrained at the joint level. An alternative approach is initially to convert the end-effector position and orientation requirements into their joint displacement equivalents. Path generation may then be performed at the joint level.

The second stage is the task specification which can be described as the process to determine the boundary conditions of the trajectory. These depend on the kinematic and dynamic restrictions of the machine and the tolerance of the product to the accelerating forces as well as the motion of the other machines which co-operate in handling the product. Position, velocity, ac-

celeration even jerk can be specified as boundary conditions at certain points to design a good trajectory. These particular points will be called "design points".

The third and the most important stage is the generation of motion curves. The ideal conditions for the performance of a manipulator can be achieved by a proper motion which is associated with a smooth, continuous and predictable curve [1.5], and [1.6]. Motion curves, in general, can be generated by means of time based mathematical functions. There are many mathematical functions which can be considered as candidates to design motion curves. These functions will be discussed in detail in Chapter 2. However, a few of them are summarised here.

Cam motion laws have been used traditionally to produce non-uniform motion. These functions are generally not suitable for designing general motions for the end-effector of a mechanism especially complex trajectories because of the difficulties previously mentioned. An example serves to clarify the situation. Assume that two identical robot arms follow trajectories to do the same job in the same period of time. The first trajectory is designed using cam motion laws and the other is designed by a suitable motion law. The definition of the suitable motion may be stated as follows. "It can be described as a motion which satisfies a given set of boundary conditions, producing smooth paths, which are continuous up to at least the second derivative (acceleration). These paths do not include any abrupt changes in their curves and importantly they exhibit low peak velocity". Now, the question is which robot arm has the better dynamic characteristics when they are performing these motions. If we examine the cam motion, it is continuous, smooth and gives the expected curve. However the main disadvantage is that the velocity and acceleration curves cannot be manipulated, thus, they have to be zero at the end of each segment. This situation causes higher velocity and acceleration values when compared to the other trajectory. High

velocity values and abrupt changes in acceleration create many problems in mechanical systems. There is therefore a need for flexible and powerful mathematical functions for general motion design.

Although many mathematical functions can be used to describe a trajectory, polynomial functions have the most suitable form since they lend themselves naturally to the solution of the types of problems involving arbitrary constraints [1.6]. The advantages of using polynomial functions for trajectory design include:

(i) they give continuous functions
(ii) they are explicit functions of time
(iii) they have a unique solution and strong sign-regularity
(iv) it is easy to specify boundary conditions for any derivative of the function
(v) they are easy to store, manipulate and evaluate on a digital computer.

In spite of these advantages, an important drawback with polynomials can make their direct application unsuitable for trajectory design. Polynomials can give curves which may oscillate between design points as their degree increases [1.7],[1.8] becoming more pronounced as the time interval increases [1.9]. This behaviour is known as **meandering** and it can be the cause of important problems in the machine. However the effects of this drawback can be moderated by means of a number of methods. A major method is to divide the whole motion into smaller segments and then to apply a number of lower order polynomial functions for these segments. Adding segments to one another to complete the trajectory produces a **segmented polynomial solution.** In order to have continuity from segment to segment the boundary conditions at the end of preceding segment can be accepted as boundary conditions at the beginning of the following segment. Unfortunately, in spite of dividing the motion into segments some of the polynomials will still give curves which do not lie within an acceptable

tolerance envelope. In this work a scheme has been developed to bring these curves within the specified tolerance envelope. The strategy is to use arbitrary powers (which may be real, integer or mixed in type) for the polynomial functions. The shape of the curve can then be adjusted without changing any boundary conditions. An arbitrary selection of the powers gives many different shapes for the trajectory between the boundary conditions, however, the best curve can be obtained by using optimization.

Another important group of functions which can be used in motion design are the rational functions. Rational functions are formed from quotients of two polynomials of different degrees. They have many of the advantages of polynomial functions, but unlike polynomials they do not produce meandering. However, they have other important drawbacks. Evaluation of this type of function may be quite difficult especially if there are derivative boundary conditions such as acceleration and jerk to satisfy. Another disadvantage is that the function may have poles for some values of the input variable at which the denominator of the function may become zero. The function is no longer satisfactory for the motion design because it tends to infinity at those points. The poles however may be shifted out of the relevant interval by changing the degree of the function.

If the trajectory has been planned at the end-effector level the trajectory must be transformed to the joint co-ordinates to determine the angular position of each driven axis. This can be done by using the geometry and link dimensions of the mechanism. Feasibility checks can be performed either during the inverse solution or they can be done when designing the motion. Whether the path is within the range of the mechanism or not and the maximum torque capacities of the driving motors are the main parameters to be checked.

To implement a motion in practice it is necessary to design an appropriate control system. A typical motion control system can be divided into three levels. The lowest level is the control of position which is performed by a

position control loop. The next level of control involves profile selection and sequence generation. The highest level includes the generation of the motion commands and the co-ordination of the motion with the overall process control. The motion controller closes the position loop around a servo motor by sensing its actual position with an incremental encoder. The motion position is decoded and compared with the command trajectory to form the position error. The error signal is processed by a stabilizing filter. The output of the filter is then amplified by the driver and then applied to the motor.

A programmable system usually follows the required trajectory with an error. The desired response can be obtained by a tuning algorithm. **Tuning** is the process of adjusting the gains in a control loop or adjusting the command trajectory to achieve the desired response.

1.4 Thesis structure

Chapter 2 includes an investigation of mathematical functions which can be used for motion design. There are two general methods for generating curves which satisfy a number of boundary conditions, curve fitting and interpolation. With curve fitting, the curves pass usually near the boundary conditions. With the interpolation method the curves must always pass through the boundary conditions. For motion design precise satisfaction of the boundary conditions is required so that the method of interpolation is appropriate.

Chapter 3 discusses motion design strategies and procedures for computer aided generation of curves. The choice of boundary conditions, the manipulation of the interpolation functions using dummy boundary conditions, the division of a path into smaller segments and a new method based on using arbitrary powers for polynomials are detailed in this chapter.

Chapter 4 outlines the software developed to produce motion curves for a mechanism, the end-effector of which moves over an X and Y plane and rotates about its own axis. The program includes many facilities to design

the optimum motion. The necessary boundary conditions must be specified to provide the main input for the software, while other interactivity is based on mouse click selection. The user can check the motion profiles and then may use modification menus to improve the characteristic shape of the motion. Motion curves for all three axes can be designed simultaneously. The program also includes a simulation option which shows how the product moves along the trajectory. The resultant trajectory can be stored as data to drive the servo motors or for other purposes.

Dynamic data structures have been used in writing the program, because, with a dynamic data structure one adds storage only as it is required and in the case of deletions the freed storage can be returned for re-use. The availability of dynamic data structures to model dynamic situations can save on storage and often allows a more structured approach to programming. If a static data structure, such as arrays, is used to represent the list of information, it must be big enough to contain all possible additions. This can lead to an initial over estimation of the array size and be wasteful of computer memory. Conversely underestimating the array size can fill the array size too soon and no more information can then be added.

Chapter 5 concerns the design of the experimental rig which is capable of following the required trajectory at high speed. A five-bar mechanism has been used as the optimum mechanism to follow planar trajectories. Carbon fibre linkages are used to minimize the inertia forces. The system is driven by two D.C. brushless servo motors.

Chapter 6 details the design of control software for the experimental rig. The host computer controls the whole system. The determination of the starting position of the mechanism, the inverse solution, feasibility checks and the loading of trajectories (five trajectories simultaneously) to the system are the essential elements of the program. The parameters of the servo control card and the speed of the system can be adjusted without interfacing

the system. Self tuning, monitoring the command and response curves as well as switching from one motion to another without change-over time are the other features of the software.

Chapter 7 discusses the concept of trajectory tuning to improve the response of the system. Depending on the parameters of the servo control system, there is usually an error between the command and response. The error can be improved by different methods such as:

(i) Determination of the optimum parameters at the required speed of the system.

(ii) Adjustment of the phase lag between the response and its command when this is appropriate.

(iii) The third alternative is to tune the command. This is a prediction method where the current command is modified by adding the error measured in the preceding cycle.

Several examples are examined on the experimental rig and the results are presented and compared graphically.

Chapter 8 contains the conclusions, discussion, and recommendations for further study.

CHAPTER 2
FUNCTIONS FOR MOTION DESIGN

2.1 Introduction

A problem associated with many engineering and statistical applications for which only certain data values are available is to find a smooth function whose curve passes through (not just near) the given boundary points. One important area for such applications is the problem of motion design. This concerns the determination of the path of the end effector of a mechanism. It may be defined as a function of time or it may be co-ordinated to the position of some other moving elements of the system. Values for position, velocity, acceleration and even jerk may be set as boundary conditions. The dynamic and kinematic constraints of the machine and the tolerance of the product to the manipulative forces as well as the minimum time to carry out a process effectively are the main features which determine the boundary conditions. The main interest is with the appearance of the curve, its shape or more precisely its Cartesian geometry, and how it may be constructed and computed.

In this chapter some interpolation methods are discussed and their formulation and characteristic features are illustrated diagrammatically; however their suitability for motion design will be discussed in chapter 3.

2.2 Interpolation Functions

A variety of mathematical forms is potentially suited to match a set of boundary conditions. Suitable forms which can be considered as candidates for motion design include:

(i) Exponential functions.
(ii) Logarithmic functions.
(iii) Hyperbolic functions.
(iv) Inverse hyperbolic functions.
(v) Trigonometric functions.
(vi) Inverse trigonometric functions.
(vii) Fourier series (Harmonic functions).
(viii) Standard cam functions.
(ix) Power form of polynomial functions.
(x) Newton interpolating function.
(xi) Lagrange interpolating function.
(xii) Hermite interpolating function.
(xiii) Rational functions.
(xiv) Cubic spline functions.
(xv) Quintic spline functions.

After detailed investigation of the motion laws listed above it is possible to form the conclusion that there are several strong inter-relationships between some of these functions. However, although the names of these functions are different their solution methods are similar and they also give similar curve characteristics but may be in different quadrants of the co-ordinate axes. The relationships between these functions are summarised as follows:

(v) Trigonometric functions and (vi) inverse trigonometric functions.

$$\mathrm{Sin}(y) = x \qquad \mathrm{Sin}^{-1}(x) = y \tag{2.1}$$

$$\mathrm{Cos}(y) = x \qquad \mathrm{Cos}^{-1}(x) = y \tag{2.2}$$

(vi) Inverse trigonometric and (iv) inverse hyperbolic functions.

$$\mathrm{Sin}^{-1}(iy) = i\mathrm{Sinh}^{-1}(x) \tag{2.3}$$

$$\mathrm{Cos}^{-1}(iy) = i\mathrm{Cosh}^{-1}(x) \tag{2.4}$$

Furthermore, Fourier series and cam motions are both compositions of trigonometric functions. Because of these similarities the above functions (iv), (v) and (vi) may be neglected. So we may re-classify the motion laws to be examined as follows:

(i) Exponential functions.
(ii) Logarithmic functions.
(iii) Hyperbolic functions.
(iv) Fourier series (Harmonic functions).
(v) Standard cam functions.
(vi) Power form of polynomial functions.
(vii) Newton interpolating function.
(viii) Lagrange interpolating function.
(ix) Hermite interpolating function.
(x) Rational functions.
(xi) Cubic spline functions.
(xii) Quintic spline functions.

2.3 Exponential Functions

There are many physical processes that are modelled by curves that exhibit exponential properties, either exponential growth or exponential decay. The problems and solution methods are described in [2.1]. They have stable curve characteristics, that means, the curves of the function and its derivatives are always smooth and predictable. The motion equations (displacement, velocity

and acceleration which correspond to the function, the first derivative and the second derivative of the function respectively) of the exponential function are:

$$f(x) = e^x \tag{2.5}$$

$$\dot{f}(x) = e^x \tag{2.6}$$

$$\ddot{f}(x) = e^x \tag{2.7}$$

The function can be modified to draw a certain path between two points. Such a path is known as a **segment**. The exponential function can be expressed as:

$$f(x) = a*e^{bx} \tag{2.8}$$

$$\dot{f}(x) = a*b*e^{bx} \tag{2.9}$$

$$\ddot{f}(x) = a*b^2*e^{bx} \tag{2.10}$$

Where a and b are constant coefficients and x is the input variable.

Fig.2.1. (Solid line) shows the basic curve characteristics of exponential functions. The values for derivative (velocity and acceleration) increase as x increases. The starting and final values of the velocity and acceleration curves for each segment depend on the values of a and b. Since each segment has different values for these quantities, discontinuities will be seen in the curves of velocity and acceleration between the segments (see Fig.2.2). The discontinuities in the curves make exponential functions unsuitable for producing acceptable motion curves. Another important drawback of these functions is that they will not pass through more than two arbitrarily specified boundary points.

2.4 Logarithmic Functions

The natural forms of the logarithmic function and its derivatives are:

$$f(x) = \ln(x) \tag{2.11}$$

$$\dot{f}(x) = \frac{1}{x} \tag{2.12}$$

$$\ddot{f}(x) = -\frac{1}{x^2} \tag{2.13}$$

The logarithmic function can be rearranged to draw a trajectory for a segment, such as:

$$f(x) = a + b * \ln(x) \tag{2.14}$$

$$\dot{f}(x) = \frac{b}{x} \tag{2.15}$$

$$\ddot{f}(x) = -\frac{b}{x^2} \tag{2.16}$$

The characteristic shapes of the curves of the logarithmic function and its derivatives are similar to the characteristic shapes of the exponential function. The curves of the derivatives of the function decrease inversely with the parameter x. The disadvantages of the function for motion design are similar to those for exponential functions. See Fig.2.1-2.2 (dashed line).

2.5 Hyperbolic Functions

The hyperbolic functions are combinations of the exponential functions e^x and e^{-x}. They have names which correspond with the names of trigonometric functions, which are:

(i) Hyperbolic sine functions.

(ii) Hyperbolic cosine functions.

(iii) Hyperbolic tangent functions.

(iv) Hyperbolic cosecant functions.

(v) Hyperbolic secant functions.

(vi) Hyperbolic cotangent functions.

In general, the hyperbolic functions have the same disadvantages as exponential functions from the point of view of the motion designer. As an example the hyperbolic sine function sinh(x) is examined here as the most basic function of this group. The motion equations of the functions are:

$$f(x) = \mathrm{Sinh}(x) = \frac{1}{2}(e^{x} - e^{-x}) \tag{2.17}$$

$$\dot{f}(x) = \mathrm{Cosh}(x) = \frac{1}{2}(e^{x} + e^{-x}) \tag{2.18}$$

$$\ddot{f}(x) = -\mathrm{Sinh}(x) = -\frac{1}{2}(e^{x} - e^{-x}) \tag{2.19}$$

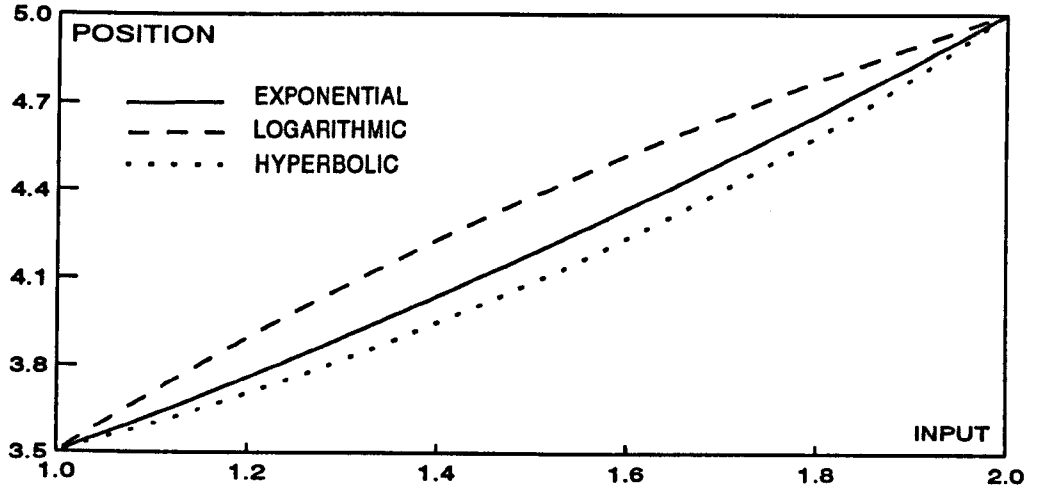
The function can be modified as before to fit the curves of a specified path by satisfying the required boundary conditions. Therefore the equations are converted into the form:

$$f(x) = a + b^{*}\,\mathrm{Sinh}(x) \tag{2.20}$$

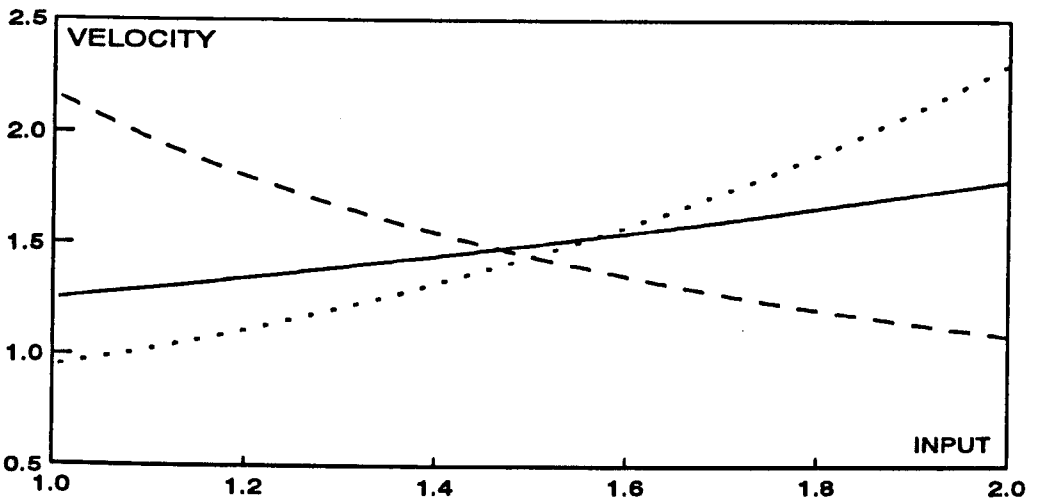
$$\dot{f}(x) = b^{*}\,\mathrm{Cosh}(x) \tag{2.21}$$

$$\ddot{f}(x) = -b^{*}\,\mathrm{Sinh}(x) \tag{2.22}$$

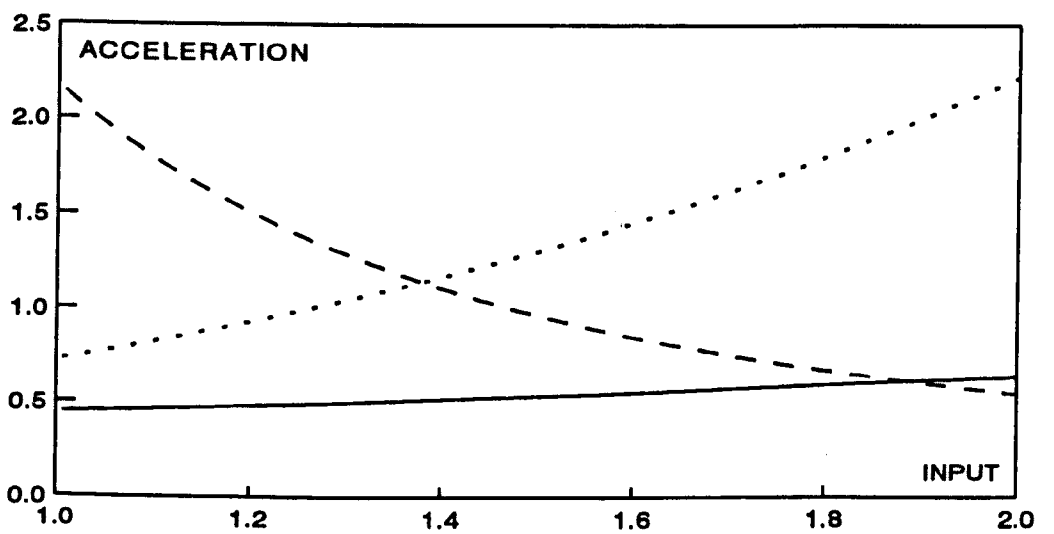
Fig.2.1 and Fig.2.2 (dotted lines) show the motion curves of a hyperbolic sine function. Usually the function draws an arc which is more circular than



a)position

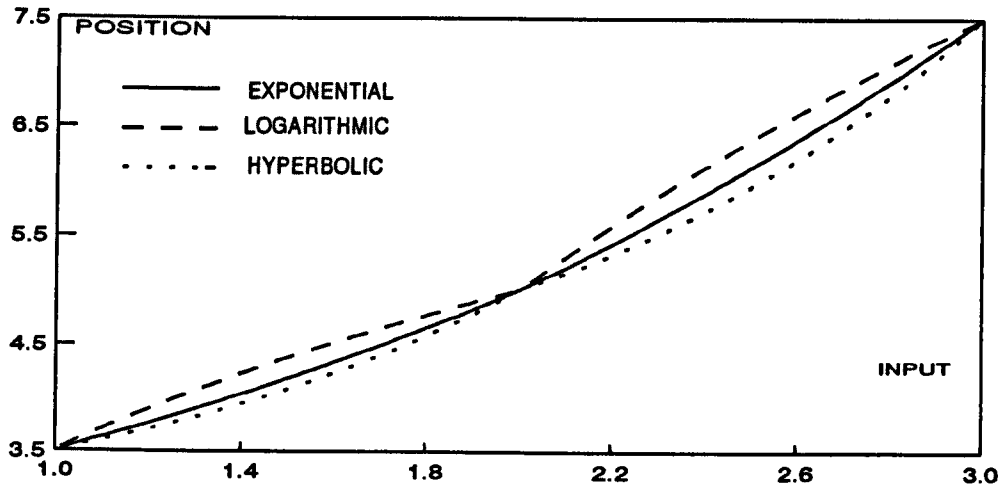


b)velocity

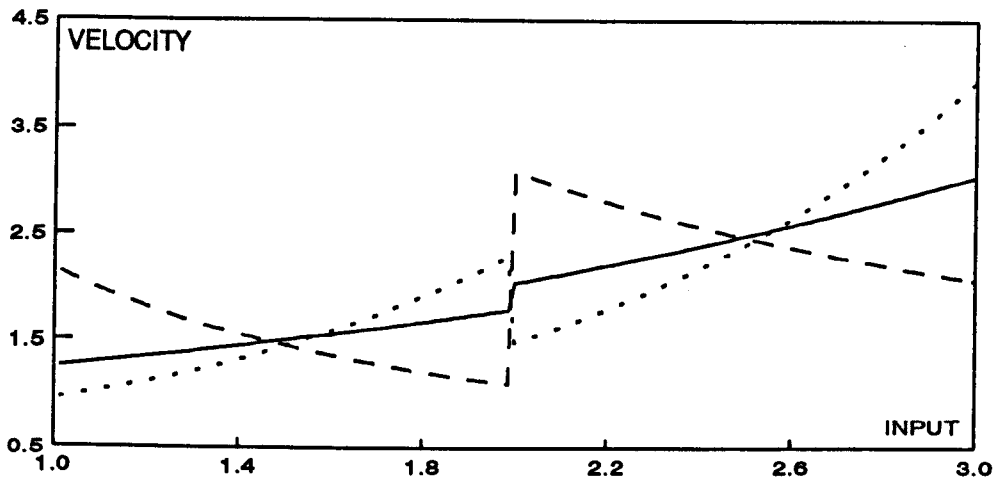


c) acceleration

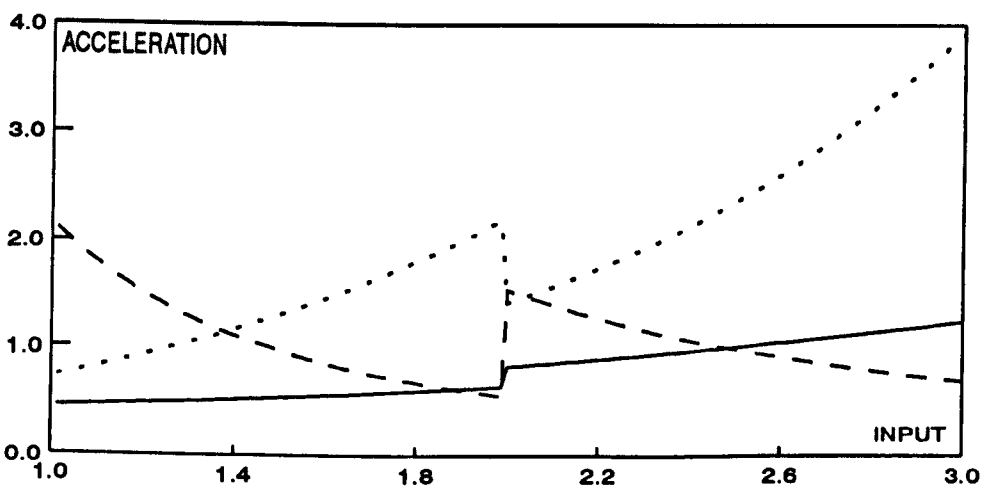
Fig.2.1 Curves of exponential, logarithmic and hyperbolic functions for one segment



a)position



b)velocity



c) acceleration

Fig.2.2 Curves of exponential, logarithmic and hyperbolic functions for two segments

the curves for exponential and logarithmic functions. The main disadvantage of the function is the pronounced discontinuity of velocity and acceleration curves between segments as seen in the figures.

2.6 Fourier Series

Fourier series arise from the task of representing a given periodic function $f(x)$ by a trigonometric series. The coefficients of the Fourier series are determined by the **Euler formulas** [2.2]. These give

$$f(x) = \frac{1}{2}a_0 + \sum_{n=1}^{\infty}(a_n \mathrm{Cos}(nx) + b_n \mathrm{Sin}(nx)) \qquad (n = 1, 2...) \qquad (2.23)$$

where the coefficients are determined by

$$a_0 = \frac{1}{\pi}\int_0^{2\pi} f(x)dx \qquad (2.24)$$

$$a_n = \frac{1}{\pi}\int_0^{2\pi} f(x)\,\mathrm{Cos}(nx)dt \qquad (2.25)$$

$$b_n = \frac{1}{\pi}\int_0^{2\pi} f(x)\,\mathrm{Sin}(nx)dx \qquad (2.26)$$

Obviously the infinite Fourier series are not applicable in practice so they must be truncated to permit programming for a computer. The truncated or finite Fourier series provides the basis for **harmonic analysis.**

2.6.1 Harmonic Analysis

The function given by a finite Fourier series can produce curves which are always continuous and pass through the data points (the data has to be periodic and equally spaced for harmonic analysis). The finite Fourier series

is written for $2n$ points in the standard interval (0 *to* 2π) and any given interval can be translated into the standard interval by stretching or shrinking the given arbitrary interval [2.3]. This standard interval then is divided into equal sub-intervals usually consisting of 6, 12, 24 or 60 terms.

2.6.2 Coefficients of Harmonic Analysis

Calculation of the coefficients of harmonic analysis is straight forward but requires a lot of time. Increasing the number of boundary conditions causes the computational time to increase by a factor equal to the square of the number of boundary conditions. If a finite Fourier series is written for $2n$ equally spaced boundary conditions over a standard interval (0 *to* 2π) such that:

$$\theta = (x - x_{initial})/(x_{final} - x_{initial})*2\Pi \tag{2.27}$$

then

$$f(\theta) = \frac{1}{2}a_0 + \left\{\sum_{k=1}^{n-1}(a_k \mathrm{Cos}(k\theta) + b_k \mathrm{Sin}(k\theta))\right\} + \frac{1}{2}a_n \mathrm{Cos}(n\theta) \tag{2.28}$$

where a_0, a_k and b_k are constant coefficients defined by

$$a_0 = \frac{1}{n}\sum_{i=0}^{2n-1} f(\theta)_i \tag{2.29}$$

$$a_k = \frac{1}{n}\sum_{i=0}^{2n-1} f(\theta)_i \mathrm{Cos}(i\theta) \tag{2.30}$$

$$b_k = \frac{1}{n}\sum_{i=0}^{2n-1} f(\theta)_i \mathrm{Sin}(i\theta) \tag{2.31}$$

2.6.3 Examples of Harmonic Analysis

The harmonic analysis approach makes possible the use of Fourier series for engineering applications, but with some restrictions. The number of boundary conditions for harmonic analysis is directly related the accuracy of the curves. Usually many boundary conditions are necessary to achieve good curves, especially if they are difficult curves. The interval between successive boundary conditions should be equal to achieve a good harmonic analysis.

The first example to be considered is a square wave curve defined with 12 boundary conditions, see Fig.2.3. The shape of the displacement curve is not however satisfactory. A much better result (dashed line) is obtained when the number of boundary conditions is increased to 40. This demonstrates the importance of the number of boundary conditions on the acceptability of the solution.

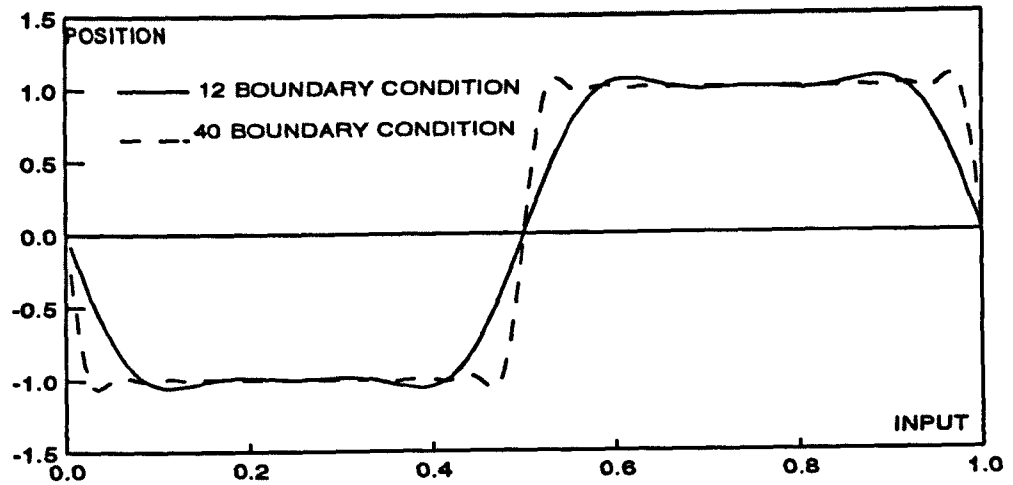
A further example is illustrated in Fig.2.4 and Fig.2.5 which show the application of harmonic analysis to the **Runge** function which is defined as

$$f(x) = \frac{1}{(1+x^2)} \tag{2.32}$$

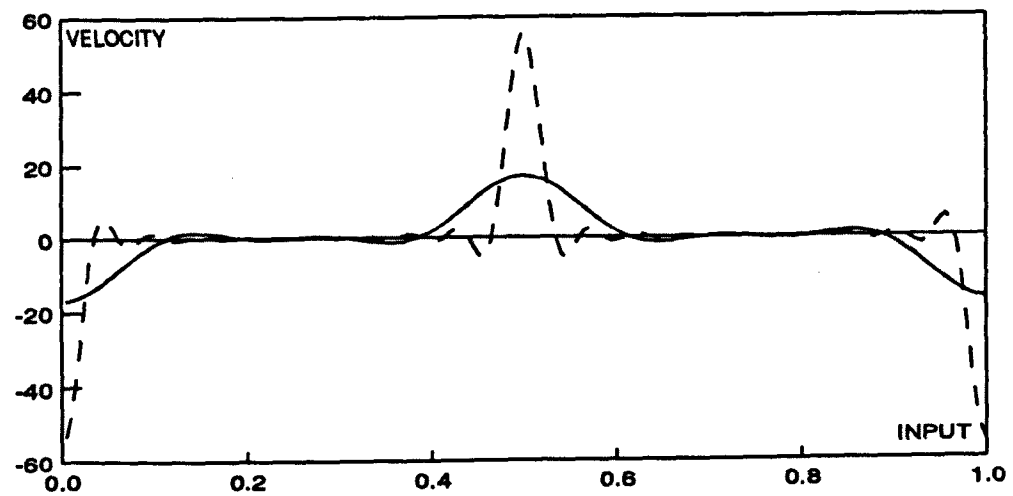
This function is often used to demonstrate the problem of oscillations when using interpolation functions [2.3],[2.4].

The first attempt with harmonic analysis is shown in the figure (dashed line) using 12 boundary conditions. The curves especially those for the derivatives, do not give a good match with the theoretical ones. In the second trial, 40 boundary conditions are defined and the result of the harmonic analysis is much more acceptable, with the displacement and velocity curves in particular matching the original curves very well.

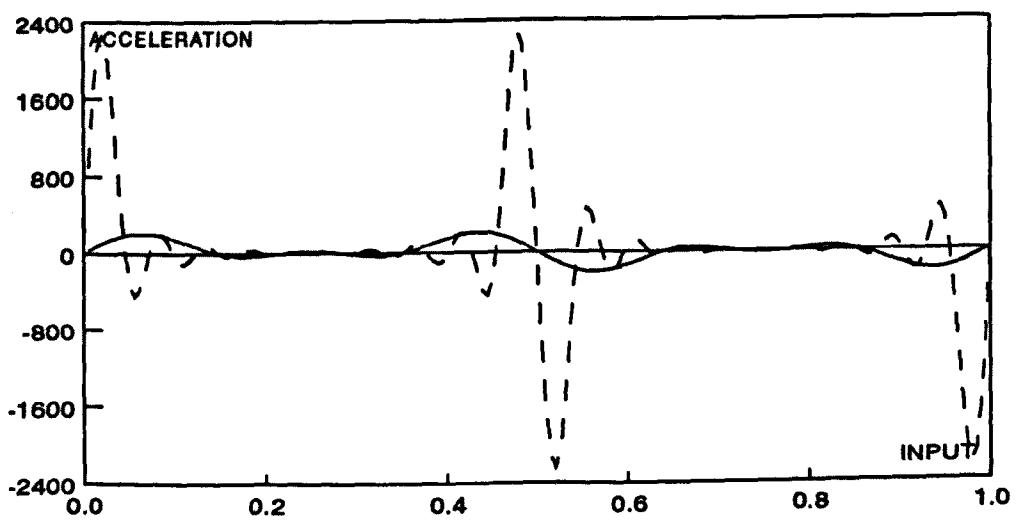
Harmonic analysis is used extensively because of some important advantages which are:



a) position

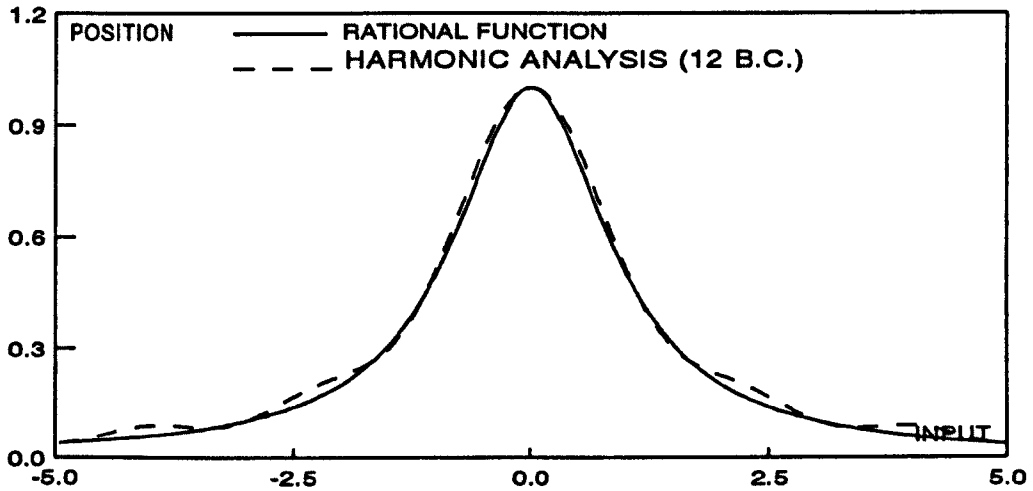


b) velocity

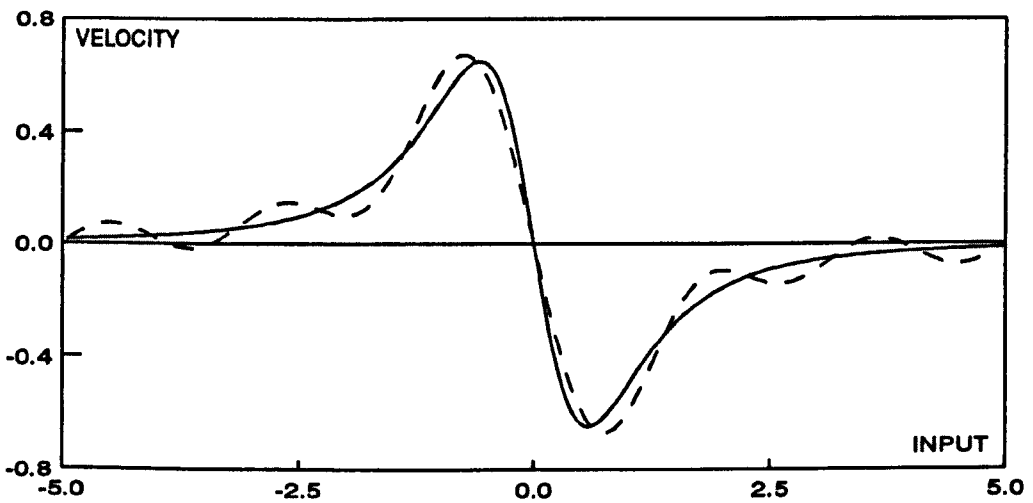


c) acceleration

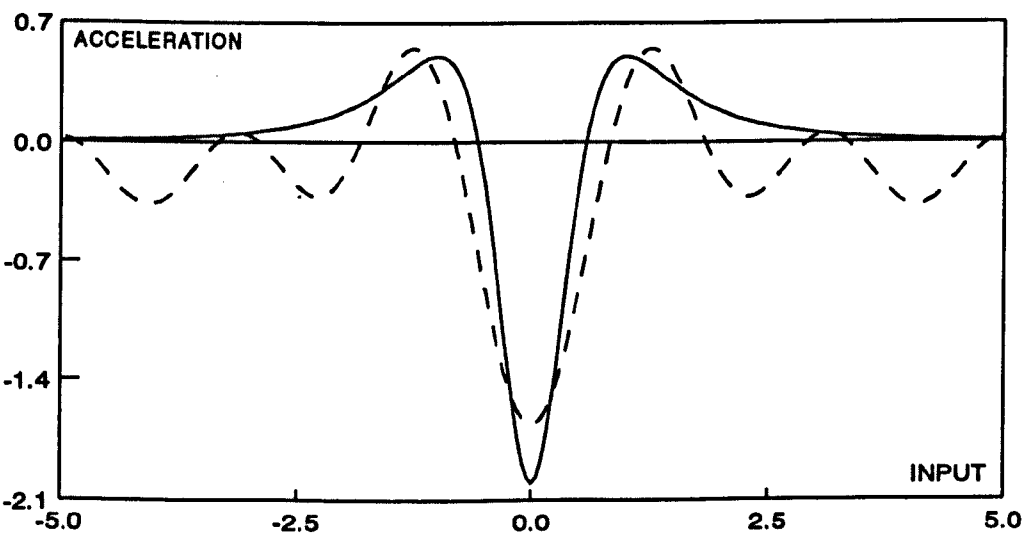
Fig.2.3 Harmonic analysis for a square wave shape



a) position

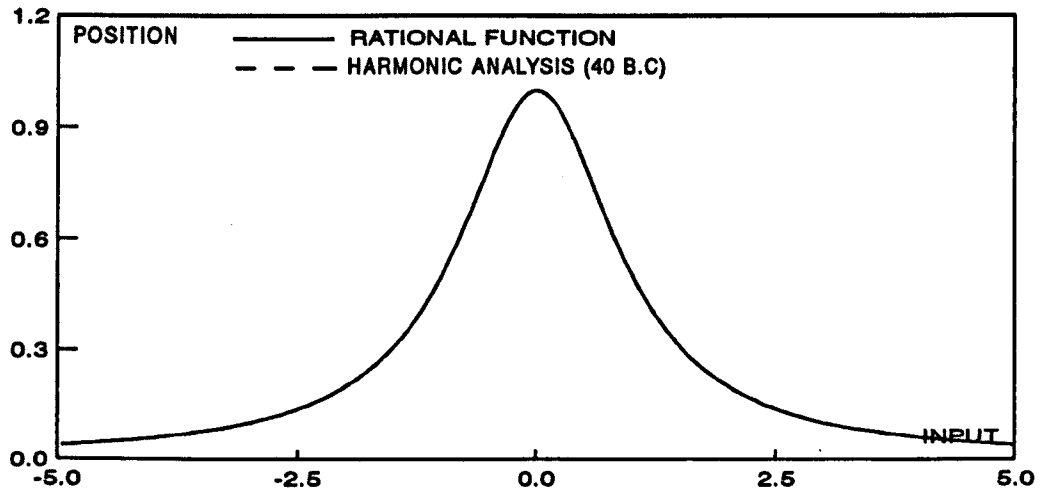


b) velocity

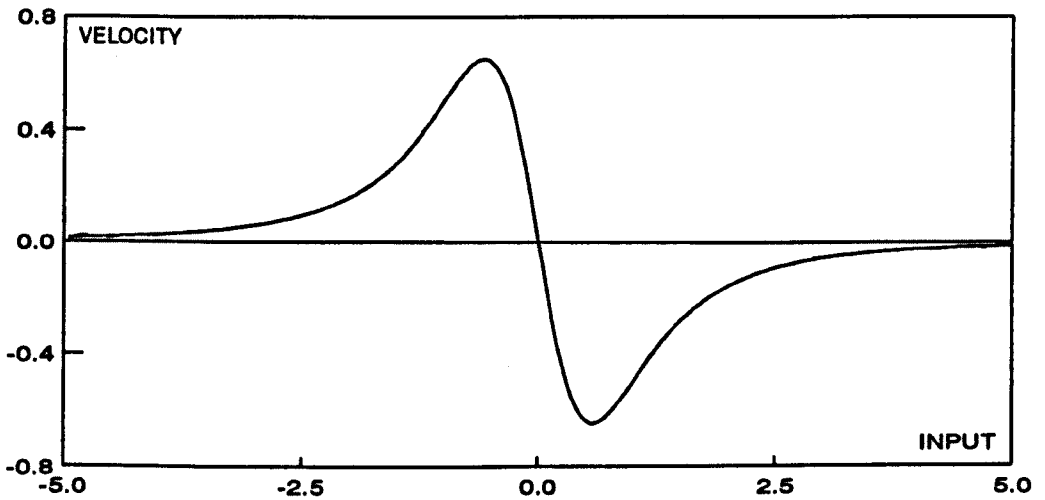


c) acceleration

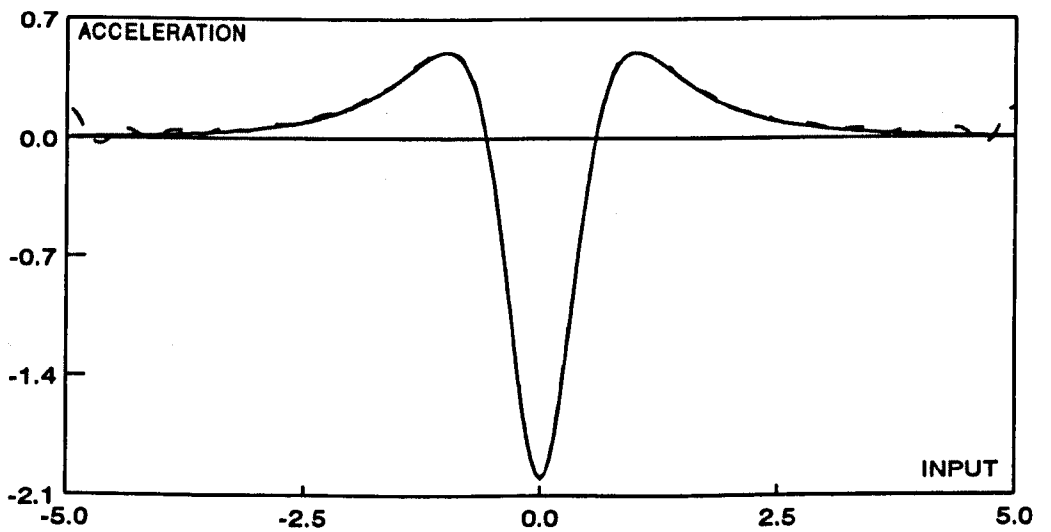
Fig.2.4 Harmonic analysis with 12 boundary condition for Runge function



a) position



b) velocity



c) acceleration

Fig.2.5 Harmonic analysis with 40 boundary condition for Runge function

(i) harmonic analysis produces continuous curves for all derivatives of the function

(ii) the determination of the constant coefficients is simple

(iii) the error is spread over the interval instead of being small near a single point

(iv) successive maximum errors oscillate in waves of approximately equal amplitude.

2.7 Standard Cam Motions

Mathematically determined laws are commonly used to define cam profiles. These have been found convenient from the stand-point of ease of layout reproduction and the control of motion characteristics. The aim of this section is to examine the suitability of some of the standard cam motions for general motion design. The basic cam motions can be classified into two main groups according to their dynamic characteristics, that is, whether they are suited for low-speed or high-speed applications.

2.7.1 Low-speed cam motions

Some basic cam motions have discontinuities in the acceleration curves especially at the beginning and at the ends of the curves giving rise to infinite jerk values at these points. Therefore such cam motions are not suitable for high-speed applications. These include:

2.7.1.1 Constant Velocity Motion (linear)

The equation describing the linear motion for a rise h with respect to θ is

$$d = h\frac{\theta}{\beta} \tag{2.33}$$

$$v = h\frac{\omega}{\beta} \tag{2.34}$$

$$a = 0 \tag{2.35}$$

where d = *displacement*

v = *velocity*

a = *acceleration*

ω = *angular velocity*

β = *total angular rotation*

2.7.1.2 Simple Harmonic Motion

Simple harmonic motion curves are widely used for cam design, since they give profiles which are simple to design and manufacture. Smoothness in velocity and acceleration during the stroke is the advantage inherent in this curve. However, the instantaneous changes in acceleration at the beginning and at the end of the stroke tend to cause vibration, noise and wear. Thus, if inertia loads are to be overcome by the follower, the resulting forces can cause severe stresses in the members. The motion equations are:

$$d = \left(\frac{h}{2}\right)\left(1 - \text{Cos}\left(\frac{\pi\theta}{\beta}\right)\right) \tag{2.36}$$

$$v = \left(\frac{h}{2}\right)\left(\frac{\pi\omega}{\beta}\right)\text{Sin}\left(\frac{\pi\theta}{\beta}\right) \tag{2.37}$$

$$a = \left(\frac{h}{2}\right)\left(\frac{\pi\omega}{\beta}\right)^2\text{Cos}\left(\frac{\pi\theta}{\beta}\right) \tag{2.38}$$

2.7.1.3 Double Harmonic Motion

The curve is formed from the difference of two harmonics. It is an un-symmetrical curve. The rate of change of acceleration at the beginning of the rise period is definite but it is infinite at the end of the period. The motion equations are:

$$d = \left(\frac{h}{2}\right)\left\{\left[1 - \mathrm{Cos}\left(\frac{\pi\theta}{\beta}\right)\right] - \left(\frac{1}{4}\right)\left[1 - \mathrm{Cos}\left(\frac{2\pi\theta}{\beta}\right)\right]\right\} \tag{2.39}$$

$$v = \left(\frac{h}{2}\right)\left(\frac{\pi\omega}{\beta}\right)\left\{\mathrm{Sin}\left(\frac{\pi\theta}{\beta}\right) - \left(\frac{1}{2}\right)\mathrm{Sin}\left(\frac{2\pi\theta}{\beta}\right)\right\} \tag{2.40}$$

$$a = \left(\frac{h}{2}\right)\left(\frac{\pi\omega}{\beta}\right)^2\left\{\mathrm{Cos}\left(\frac{\pi\theta}{\beta}\right) - \mathrm{Cos}\left(\frac{2\pi\theta}{\beta}\right)\right\} \tag{2.41}$$

2.7.1.4 Constant Acceleration Motion (bang-bang)

This motion law is most frequently used to design the motion of robot manipulators. It is often referred as bang-bang acceleration [1.5]. The rise motion is divided into two parts. In the first part of the period the follower moves with constant acceleration while in the second part the motion occurs with constant deceleration. The most important advantage of this motion curve is that for a given angle of the rotation and rise it produces the smallest possible acceleration. However, at both ends and mid-point of the curve the jerk tends to infinity. The equations of the follower motion to the cam rotation angle are:

$$0 < \theta < \left(\frac{\beta}{2}\right) \qquad \left(\frac{\beta}{2}\right) < \theta < \beta$$

$$d = 2h\left(\frac{\theta}{\beta}\right)^2 \qquad d = h - 2h\left(1 - \frac{\theta}{\beta}\right)^2 \tag{2.42}$$

$$v = 4h\omega\left(\frac{\theta}{\beta^2}\right) \qquad v = 4h\frac{\omega}{\beta}\left(1-\frac{\theta}{\beta}\right) \tag{2.43}$$

$$a = 4h\left(\frac{\omega}{\beta}\right)^2 \qquad a = -4h\left(\frac{\omega}{\beta}\right)^2 \tag{2.44}$$

2.7.1.5 Cubic or Constant Pulse #1 Motion

The cubic #1 curve is the combination of two third order curves. There is no abrupt change at the beginning or at the end of the curve, but infinite jerk at the mid-point. The equations are:

$$0 < \theta < \left(\frac{\beta}{2}\right) \qquad \left(\frac{\beta}{2}\right) < \theta < \beta$$

$$d = 4h\left(\frac{\theta}{\beta}\right)^3 \qquad d = h - 4h\left(1-\frac{\theta}{\beta}\right)^3 \tag{2.45}$$

$$v = 12h\frac{\omega}{\beta}\left(\frac{\theta}{\beta}\right)^2 \qquad v = 12h\frac{\omega}{\beta}\left(1-\frac{\theta}{\beta}\right)^2 \tag{2.46}$$

$$a = 24h\left(\frac{\omega}{\beta}\right)^2\left(\frac{\theta}{\beta}\right) \qquad a = -24h\left(\frac{\omega}{\beta}\right)^2\left(\frac{\theta}{\beta}\right) \tag{2.47}$$

2.7.1.6 Cubic or Constant Pulse #2 Motion

The acceleration curve of cubic #2 curve is continuous, but at both ends the jerk value becomes infinite. The characteristics of this curve are similar to the simple harmonic motion case. The equations are:

$$d = h\left(\frac{\theta}{\beta}\right)^2\left(3-\frac{2\theta}{\beta}\right) \tag{2.48}$$

$$v = 6h\left(\frac{\omega}{\beta}\right)\left(\frac{\theta}{\beta}\right)\left(1-\frac{\theta}{\beta}\right) \tag{2.49}$$

$$a = 6h\left(\frac{\omega}{\beta}\right)^2\left(1-\frac{2\theta}{\beta}\right) \tag{2.50}$$

2.7.2 Cam Motions Suitable for High-Speed Application

Some basic cam motions have continuous acceleration and finite jerk values over the length of the curve, therefore these cams can run at relatively high speeds. Examples are given below.

2.7.2.1 Cycloidal Motion

The displacement curve of the cycloidal motion is equivalent to the trace of a point on a circle rolling on a straight line without slipping. A cam with this profile has excellent dynamic characteristics of motion [2.5]. The maximum value of the acceleration of the follower for a given rise and time is somewhat higher than that of the simple harmonic motion curve. In spite of this, however, the cycloidal motion is used often as a basis for designing cams for high-speed machinery, see Fig.2.6. The equations of motion are:

$$d = \frac{h}{\pi}\left\{\pi\frac{\theta}{\beta}-\frac{1}{2}\mathrm{Sin}\left(\frac{2\pi\theta}{\beta}\right)\right\} \tag{2.51}$$

$$v = h\frac{\omega}{\beta}\left\{1-\mathrm{Cos}\left(\frac{2\pi\theta}{\beta}\right)\right\} \tag{2.52}$$

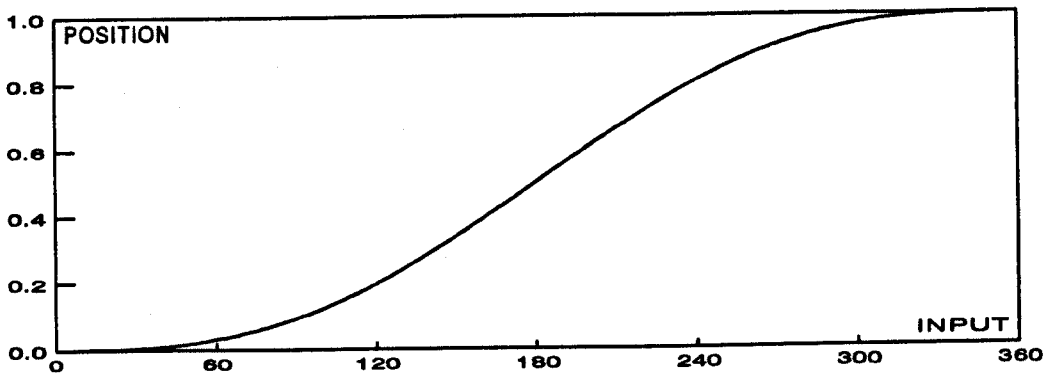
$$a = 2h\pi\left(\frac{\omega}{\beta}\right)^2\mathrm{Sin}\left(\frac{2\pi\theta}{\beta}\right) \tag{2.53}$$

Typically if $\frac{\omega}{\beta} = 1.$

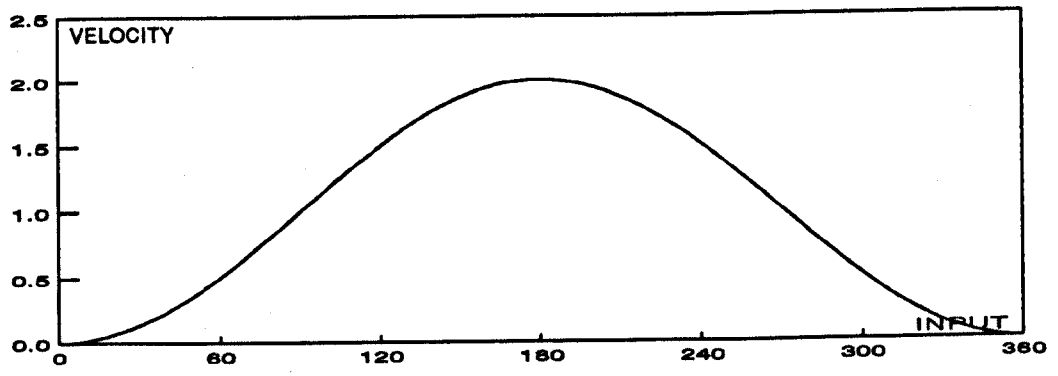
Peak acceleration $= 6.28$ *at* $\theta = 0.25\ \beta, \theta = 0.75\ \beta$

Initial jerk $= 39.47 at \theta = 0.0$

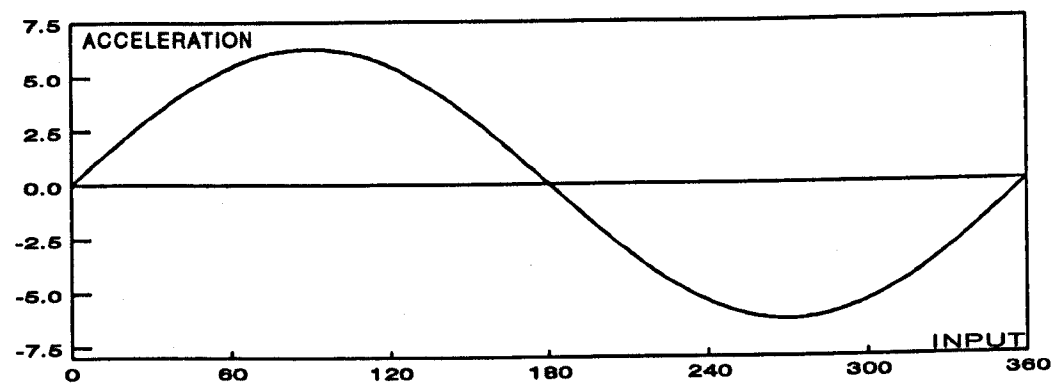
Cross $-$ *0verjerk* $= -39.47 at \theta = 0.5\beta$



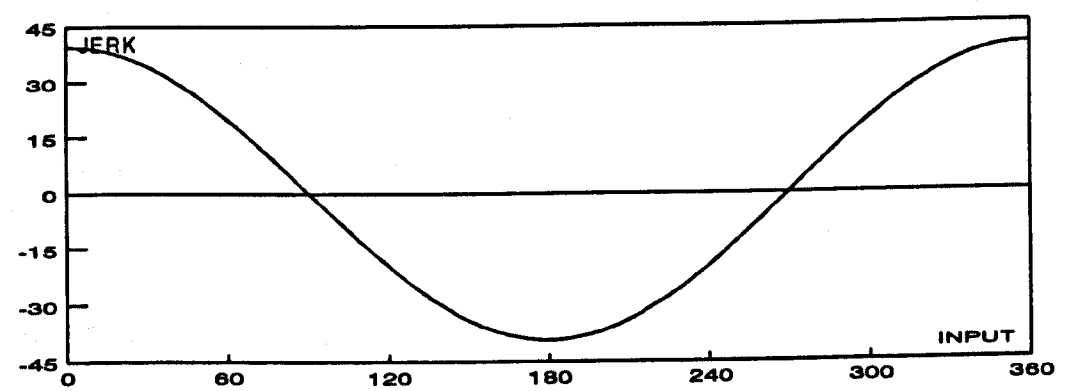
a) position



b) velocity



c) acceleration



d) jerk

Fig.2.6 Cycloidal motion curves

2.7.2.2 Modified trapezoidal motion

This motion is an improvement over the constant acceleration curve, see Fig.2.7. To avoid infinite jerk values at the ends and mid-point of this motion the rectangular shape of the acceleration is converted into a trapezoidal form. The equations of motion are:

$$A = \frac{8\pi}{2+\pi} \tag{2.54}$$

For $0 < \theta < 0.125\beta$

$$d = \left(\frac{Ah}{4\pi}\right)\left\{\frac{\theta}{\beta} - \frac{1}{4\pi}\mathrm{Sin}\left(\frac{4\pi\theta}{\beta}\right)\right\} \tag{2.55}$$

$$v = \left(\frac{Ah\omega}{4\pi\beta}\right)\left\{1 - \mathrm{Cos}\left(\frac{4\pi\theta}{\beta}\right)\right\} \tag{2.56}$$

$$a = Ah\left(\frac{\omega}{\beta}\right)^2 \mathrm{Sin}\left(\frac{4\pi\theta}{\beta}\right) \tag{2.57}$$

For $0.125\beta < \theta < 0.375\beta$

$$d = \left(\frac{Ah}{2}\right)\left(\frac{\theta}{\beta}\right)^2 + \left(\frac{A}{4}\right)\left(\frac{1}{\pi} - \frac{1}{2}\right)\left(\frac{\theta}{\beta}\right) + \left(\frac{A}{64}\right)\left(\frac{1}{2} - \frac{4}{\pi^2}\right) \tag{2.58}$$

$$v = Ah\omega\left(\frac{\theta}{\beta^2}\right) + \left(\frac{A\omega}{4\beta}\right)\left(\frac{1}{\pi} - \frac{1}{2}\right) \tag{2.59}$$

$$a = Ah\left(\frac{\omega}{\beta}\right)^2 \tag{2.60}$$

For $0.375\beta < \theta < 0.625\beta$

$$d = \left(-\frac{Ah}{16\pi^2}\right)\mathrm{Cos}\left(4\pi\left(\frac{\theta}{\beta} - 0.375\right)\right) + \left(\frac{A}{4}\right)\left(1 + \frac{1}{\pi}\right)\left(\frac{\theta}{\beta}\right) - \left(\frac{A}{16}\right) \tag{2.61}$$

$$v = \left(\frac{Ah\omega}{4\pi\beta}\right)\mathrm{Sin}\left(4\pi\left(\frac{\theta}{\beta} - 0.375\right)\right) + \left(\frac{A\omega}{4\beta}\right)\left(1 + \frac{1}{\pi}\right) \tag{2.62}$$

$$a = Ah\left(\frac{\omega}{\beta}\right)^2 \mathrm{Cos}\left(4\pi\left(\frac{\theta}{\beta} - 0.375\right)\right) \tag{2.63}$$

For $0.625\beta < \theta < 0.875\beta$

$$d = \left(-\frac{Ah}{2}\right)\left(\frac{\theta}{\beta}\right)^2 + \left(\frac{A}{8}\right)\left(7 + \frac{2}{\pi}\right)\left(\frac{\theta}{\beta}\right) - \left(\frac{33A}{128}\right) + \left(\frac{A}{16\pi^2}\right) \tag{2.64}$$

$$v = -Ah\omega\left(\frac{\theta}{\beta}\right) + \left(\frac{A}{8}\right)\left(7 + \frac{2}{\pi}\right)\left(\frac{\omega}{\beta}\right) \tag{2.65}$$

$$a = -Ah\frac{\omega^2}{\beta} \tag{2.66}$$

For $0.875\beta < \theta < 1.0\beta$

$$d = \left(\frac{Ah}{16\pi^2}\right)\mathrm{Cos}\left(4\pi\left(\frac{\theta}{\beta} - 0.875\right)\right) + \left(\frac{A}{4\pi}\right)\left(\frac{\theta}{\beta}\right) + \left(\frac{A}{8}\right) \tag{2.67}$$

$$v = \frac{-Ah\omega}{4\pi\beta}\mathrm{Sin}\left(4\pi\left(\frac{\theta}{\beta} - 0.875\right)\right) + \frac{A\omega}{4\pi\beta} \tag{2.68}$$

$$a = -Ah\left(\frac{\omega}{\beta}\right)^2 \mathrm{Cos}\left(4\pi\left(\frac{\theta}{\beta} - 0.875\right)\right) \tag{2.69}$$

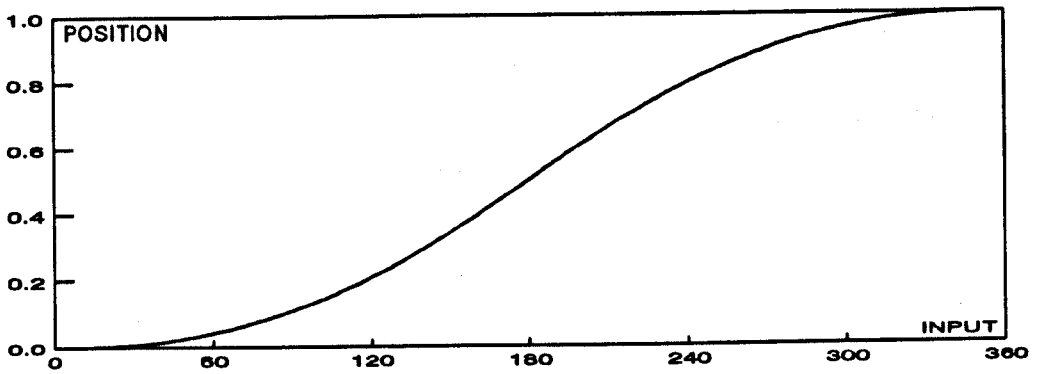
For this motion:

Peak acceleration = 4.88 *for* $0.125\beta < \theta < 0.375\beta$

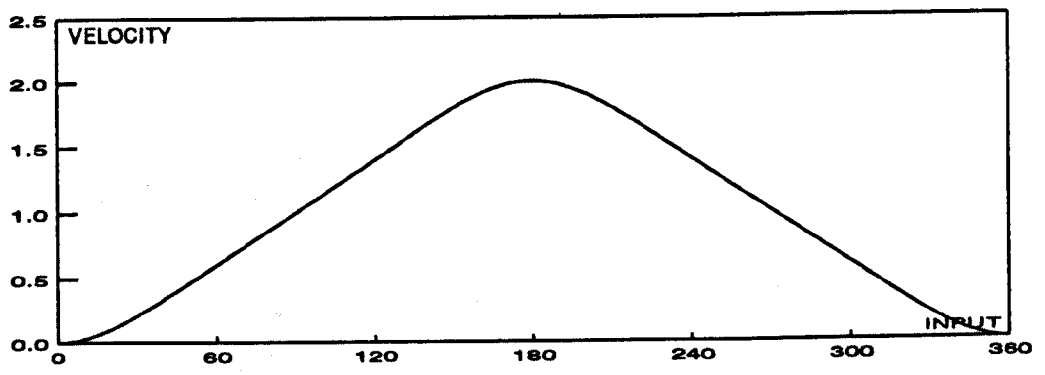
Peak acceleration = −4.88 *for* $0.625\beta < \theta < 0.875\beta$

Initial jerk = 76.42 *at* $\theta = 0.0$

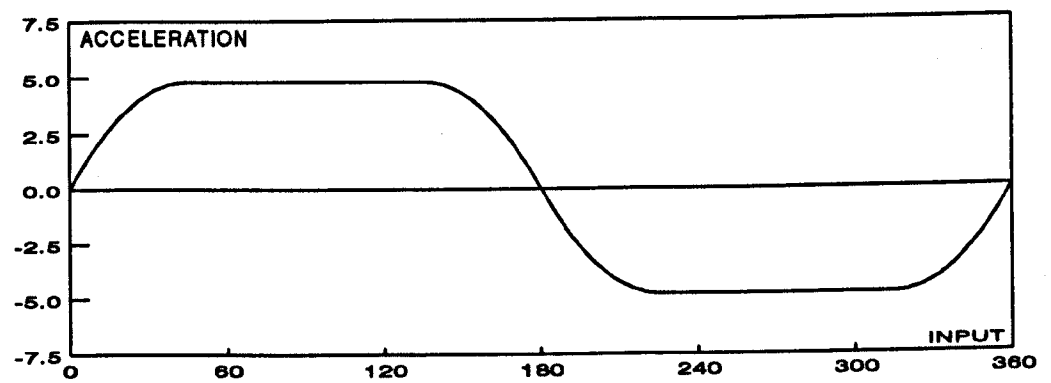
Cross − 0verjerk = −76.42 *at* $\theta = 0.5\beta$



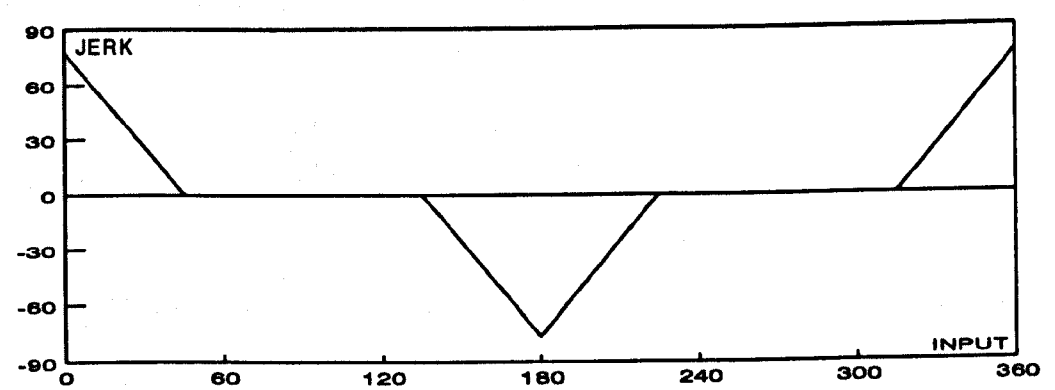
a) position



b) velocity



c) acceleration



d) jerk

Fig.2.7 Modified trapezoidal motion curves

2.7.2.3 Modified Sine Motion

This motion is similar to the modified trapezoidal motion. Three different curves are combined to design the modified sine motion. See Fig.2.8. The equations of the motion are given below.

$$A = \frac{\pi^2}{2\pi b + 2 - 8b} \tag{2.70}$$

and b is the angle of rotation of the cam at the peak acceleration.

For $0 < \theta < b$

$$d = \frac{2Ahb}{\pi}\left(\frac{\theta}{\beta} - \frac{2b}{\pi}\mathrm{Sin}\left(\frac{\pi\theta}{2b\beta}\right)\right) \tag{2.71}$$

$$v = \frac{2Ahb}{\pi}\left(\frac{\omega}{\beta}\right)\left(1 - \mathrm{Cos}\left(\frac{\pi\theta}{2b\beta}\right)\right) \tag{2.72}$$

$$a = Ah\left(\frac{\omega}{\beta}\right)^2 \mathrm{Sin}\left(\frac{\pi\theta}{2b\beta}\right) \tag{2.73}$$

For $b < \theta < 1.0 - b$

$$d = -Ah\left(\frac{1-2b}{\pi}\right)^2 \mathrm{Cos}\left\{\frac{\pi(\theta/\beta - b)}{(1-2b)}\right\} + \frac{2Ab\theta}{\pi\beta} + \frac{A}{\pi^2}(1-4b) \tag{2.74}$$

$$v = Ah\left(\frac{1-2b}{\pi}\right)\left(\frac{\omega}{\beta}\right)\mathrm{Sin}\left\{\frac{\pi(\theta/\beta - b)}{(1-2b)}\right\} + \frac{2Ab\omega}{\pi\beta} \tag{2.75}$$

$$a = Ah\left(\frac{\omega}{\beta}\right)^2 \mathrm{Cos}\left(\frac{\pi(\theta/\beta - b)}{1-2b}\right) \tag{2.76}$$

For $1.0 - b < \theta < 1$

$$d = Ah\left(\frac{2b}{\pi}\right)^2 \mathrm{Cos}\left(\frac{\pi(\theta/\beta - 1.0 + b)}{2b}\right) + \frac{2Ab\theta}{\pi\beta} + \frac{2A}{\pi^2}(1 - 4b) \tag{2.77}$$

$$v = -2Ahb\frac{\omega}{\pi\beta}\mathrm{Sin}\left(\frac{\pi(\theta/\beta - 1 + b)}{2b}\right) + \frac{2Ab\omega}{\pi\beta} \tag{2.78}$$

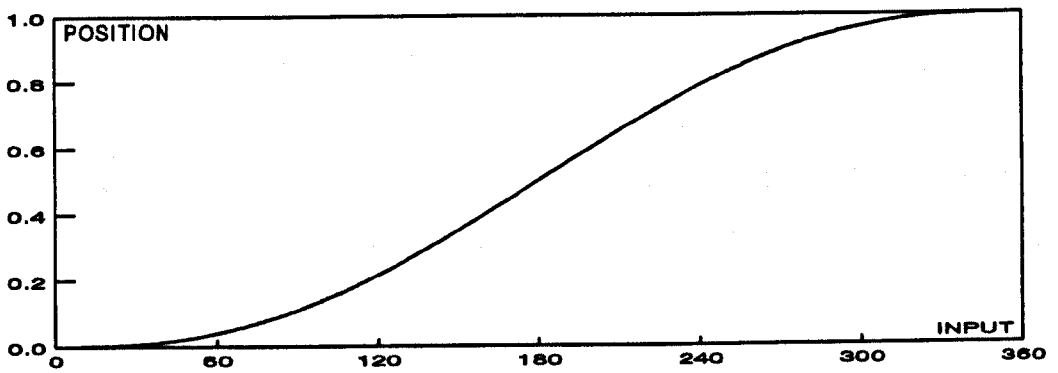
$$a = -Ah\left(\frac{\omega}{\beta}\right)^2 \mathrm{Cos}\left(\frac{\pi(\theta/\beta - 1 + b)}{2b}\right) \tag{2.79}$$

For this motion:

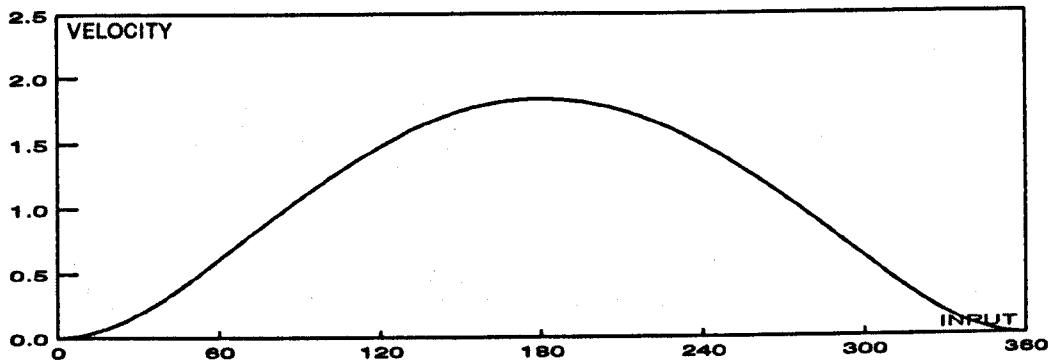
$$\textit{Peak acceleration} = \frac{\pi^2}{(2b\pi + 1 - ab)} \quad \textit{at} \quad \theta = b, \theta = 1.0 - b$$

$$\textit{Initial jerk} = \frac{\pi^3}{(4\pi b^2 + 4b - 16b^2)} \quad \textit{at} \quad \theta = 0.0$$

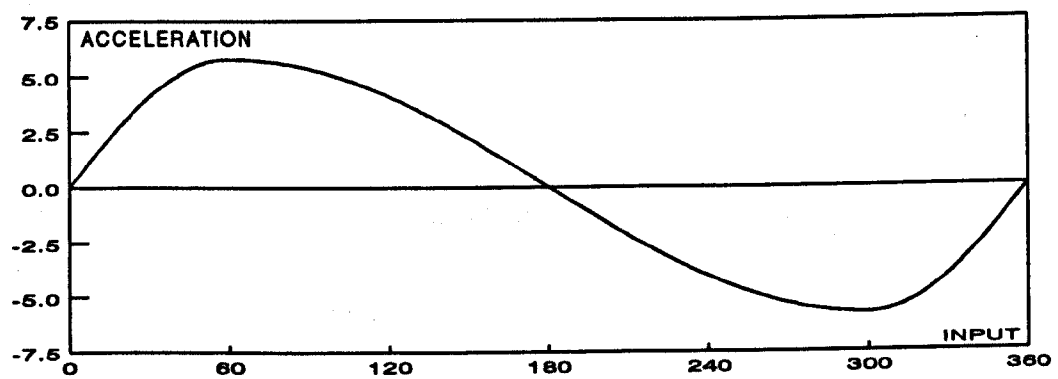
$$\textit{Cross} - \textit{0ver jerk} = -\frac{\pi^3}{(2 - 4b)(\pi b + 1 - 4b)}$$



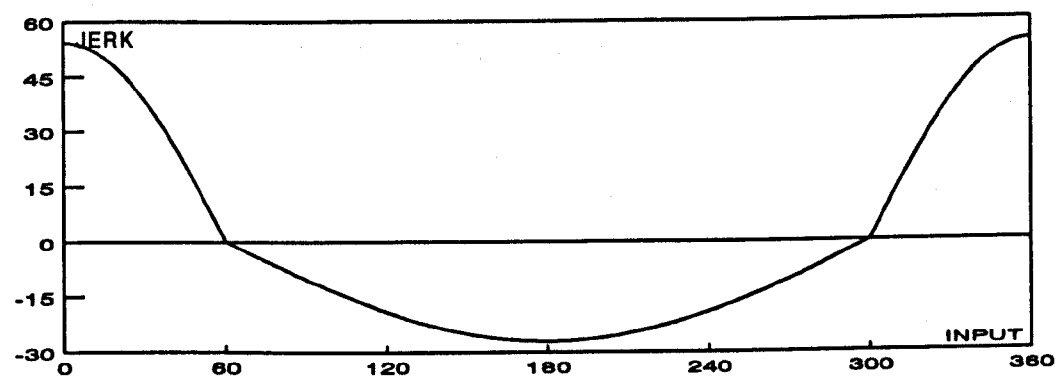
a) position



b) velocity



c) acceleration



d) jerk

Fig.2.8 Modified sine motion curves

2.7.2.4 Triple harmonic motion

The triple harmonic motion law is a general dwell-rise-dwell motion. This motion also has three small segments similar to the modified sine and modified trapezoidal motion. Cancelling the second and third harmonic converts the motion into the cycloidal form of motion. See Fig.2.9. The equations of motion are given below.

$$d = \left(h\frac{\theta}{\beta}\right) - \frac{h}{4\pi^2}\left\{A_1 \mathrm{Sin}\left(2\pi\frac{\theta}{\beta}\right) + \frac{A_2}{4}\mathrm{Sin}\left(4\pi\frac{\theta}{\beta}\right) + \frac{A_3}{9}\mathrm{Sin}\left(6\pi\frac{\theta}{\beta}\right)\right\} \tag{2.80}$$

$$v = h\frac{\omega}{\beta} - h\frac{\omega}{2\pi\beta}\left\{A_1 \mathrm{Cos}\left(2\pi\frac{\theta}{\beta}\right) + \left(\frac{A_2}{2}\right)\mathrm{Cos}\left(4\pi\frac{\theta}{\beta}\right) + \left(\frac{A_3}{3}\right)\mathrm{Cos}\left(6\pi\frac{\theta}{\beta}\right)\right\} \tag{2.81}$$

$$a = h\left(\frac{\omega}{\beta}\right)^2\left\{A_1 \mathrm{Sin}\left(2\pi\frac{\theta}{\beta}\right) + A_2 \mathrm{Sin}\left(4\pi\frac{\theta}{\beta}\right) + A_3 \mathrm{Sin}\left(6\pi\frac{\theta}{\beta}\right)\right\} \tag{2.82}$$

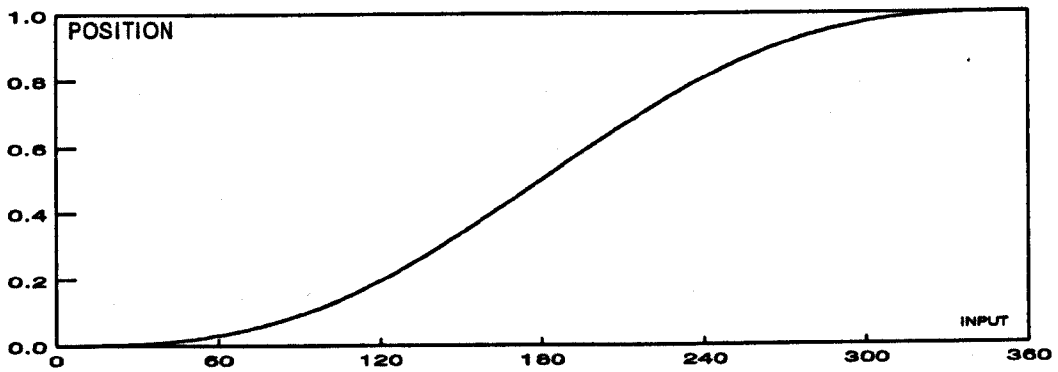
where

$$A_1 + \frac{A_2}{2} + \frac{A_3}{3} = 2\pi \tag{2.83}$$

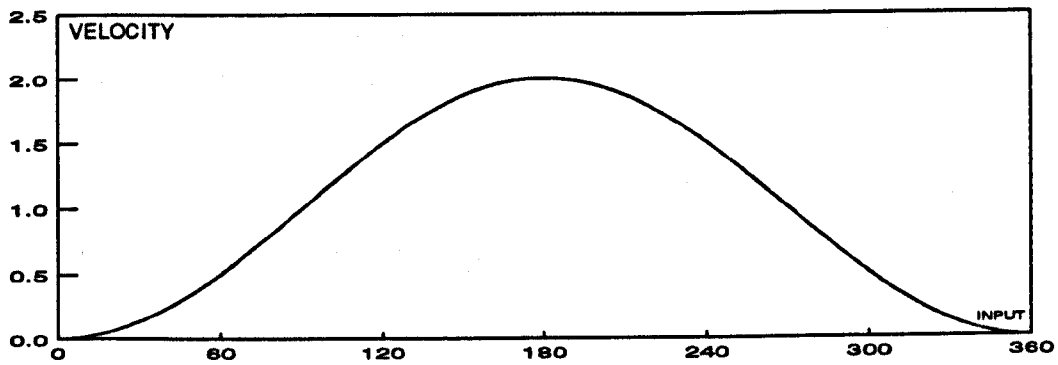
is the only constraint on the values of A_1, A_2 and A_3.

Initial $jerk = 2\pi(A_1 + 2A_2 + 3A_3)$ *at* $\theta = 0.0$

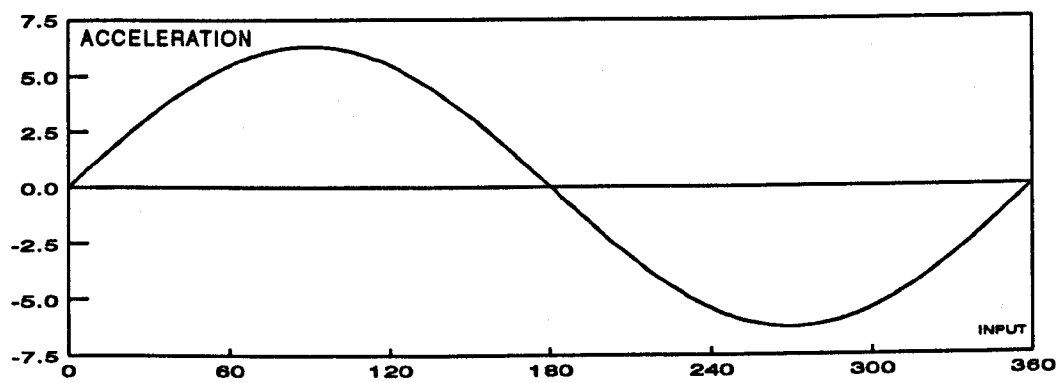
Cross - Over $jerk = 2\pi(-A_1 + 3A_2 - 3A_3)$ *at* $\theta = 0.5\beta$



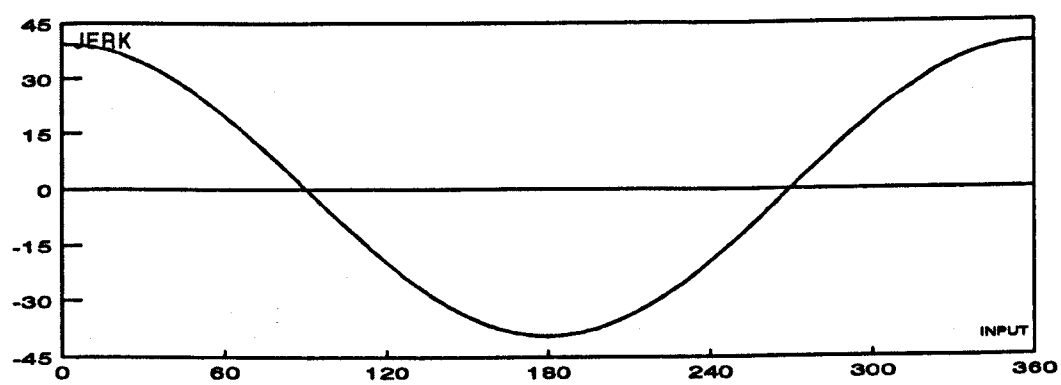
a) position



b) velocity



c) acceleration



d) jerk

Fig.2.9 Triple harmonic motion curves

2.8 Polynomial functions

One simple way of constructing a function $f(x)$ which ensures continuity of all its derivatives is to define it as the polynomial $p(x)$ of degree n which intersects $(n + 1)$ data points.

These characteristics make polynomials eminently suitable alternatives for the interpolation of a set of arbitrarily specified data points. However a significant drawback in their use is the tendency of the derived curve to oscillate between boundary conditions. In general the amount of oscillation increases with the degree of the function.

For two data points (x_0, y_0) and (x_1, y_1), the interpolation polynomial would be a straight line described by the function.

$$P(x) = a_0 + a_1 x \tag{2.84}$$

and we require

$$a_0 + a_1 x_0 = y_0 \qquad a_0 + a_1 x_1 = y_1 \tag{2.85}$$

which have a unique solution

$$a_0 = \frac{x_0 y_1 - x_1 y_0}{x_0 - x_1} \qquad a_1 = \frac{y_1 - y_0}{x_1 - x_0} \tag{2.86}$$

hence

$$P(x) = \frac{x_0 y_1 - x_1 y_0}{x_0 - x_1} + \frac{y_1 - y_0}{x_1 - x_0} x \tag{2.87}$$

and finally

$$P(x) = \frac{(x - x_1)}{(x_0 - x_1)} y_0 + \frac{(x - x_0)}{(x_1 - x_0)} y_1 \tag{2.88}$$

Similarly, three data pairs would be interpolated by a parabola. This process of polynomial interpolation can be extended to any number of data pairs.

If a function $f(x)$ is defined as a polynomial of degree smaller or equal to n and $f(x)$ is expressed as a linear combination of $(n+1)$ terms, such as

$$P(x) = \sum_{i=0}^{n} C_i B_i(x) \tag{2.89}$$

where C_i are arbitrary coefficients independent of x and B_i are linearly independent functions of x, then the set of $(n+1)$ functions $B_i(x)$ for $(i = 0, 1...n)$ is said to form a basis for the polynomial. Several types of polynomial functions are examined in the following sections.

2.8.1 Power form

The most familiar basis would be the set $(1, x, x^2..x^n)$ of ascending powers of x and such a selection would lead to the usual method of writing down a polynomial in its power form, such that

$$P(x) = \sum_{i=0}^{n} C_i x^i \tag{2.90}$$

the open form of the function is

$$P(x) = C_0 + C_1 x_i + C_2 x_i^2 + C_3 x_i^3 + C_4 x_i^4 + \ldots + C_n x_i^n \tag{2.91}$$

This form of the polynomial function allows the specification of boundary conditions for the function and also its derivatives. This is an important requirement for motion design. The coefficients define the trajectory in a unique form with the number of known boundary conditions equalling the number of coefficients which are to be found. They can thus be used to define a motion of any complexity. Polynomials, which satisfy many boundary

conditions can however give rise to a large number of local maxima and minima or turning points because of their high degree. The power form of the polynomial function is examined in more detail in the next chapter.

2.8.2 Newton interpolating polynomial

One obvious and relatively quick way of arriving at a value for a polynomial function would result from a knowledge of the terms $(c_0, c_1x, c_2x^2, \ldots, c_nx^n)$. However, there is another similar formulation which is almost as quick to evaluate and whose coefficients are much more readily computed. This is Newton's form of the interpolating polynomial which can be written in the form:

$$P_n(x) = b_0 + b_1(x - x_0) + \ldots + b_n\{(x - x_0)(x - x_1)(x - x_2)\ldots(x - x_{n-1})\}$$

i.e.

$$P_n(x) = \sum_{j=0}^{n} b_j \prod_{k=0}^{j-1}(x - x_k) \tag{2.92}$$

Where the values $(b_0, b_1, \ldots, b_n)$ are numerical coefficients and the $(x_0, x_1, \ldots, x_n)$ are the data abscissae. The polynomial can be rearranged to reduce the computation time using nested multiplication. If we assume that the coefficients are known, then

$$P_n(x) = (\ldots((d_n(x - x_{n-1}) + d_{n-1}) \quad (x - x_{n-2}) + d_{n-2})\ldots(x - x_1) + d_1) + d_0 \tag{2.93}$$

The coefficients of the interpolation polynomial can be calculated by a divided difference table as shown below. For example if we have four data pairs to be interpolated $(0, 12), (1, 32), (2, 8)$ and $(3, 19)$.

TABLE 2.1 Newton's divided difference				
X_i	Y_i			
0	12=b0			
		(32-12)/(1-0)=20=b1		
1	32		(-24-20)/(2-0)=-22=b2	
		(8-32)/(2-1)=-24		(39.9/3)=13.1=b3
2	8		(11+24)/(3-1)=17.5	
		(19-8)/(3-2)=11		
3	19			

Derivation of an equation to calculate the coefficients for the interpolation is rather complicated but they can be easily calculated computationally. An algorithm is shown below to determine the coefficients of the interpolation. The second algorithm is for nested multiplication and this can be used for fast calculation of the coefficients of interpolation.

The Runge function (equation 2.32) is tested and Fig.2.10 is obtained, where the degree of the interpolation is 10.

Algorithm for calculation of coefficients.

```
For i=0 to n Do
Begin
b[i]=Y[i]
End

For j=1 to n Do
For i=n downto j Do
Begin
b[i]=(b[i]-b[i-1])/(X[i]-X[i-j])
End
```

Algorithm for nested multiplication.

For k=0 to n Do
Begin
Xbar[k]=X[0]+k*(X[n]-X[0])/n
sum=b[n]
For i=(n-1) downto 0 do
Begin
sum=sum*(Xbar[k]-X[i])+b[i]
End
Newton[k]=sum
End

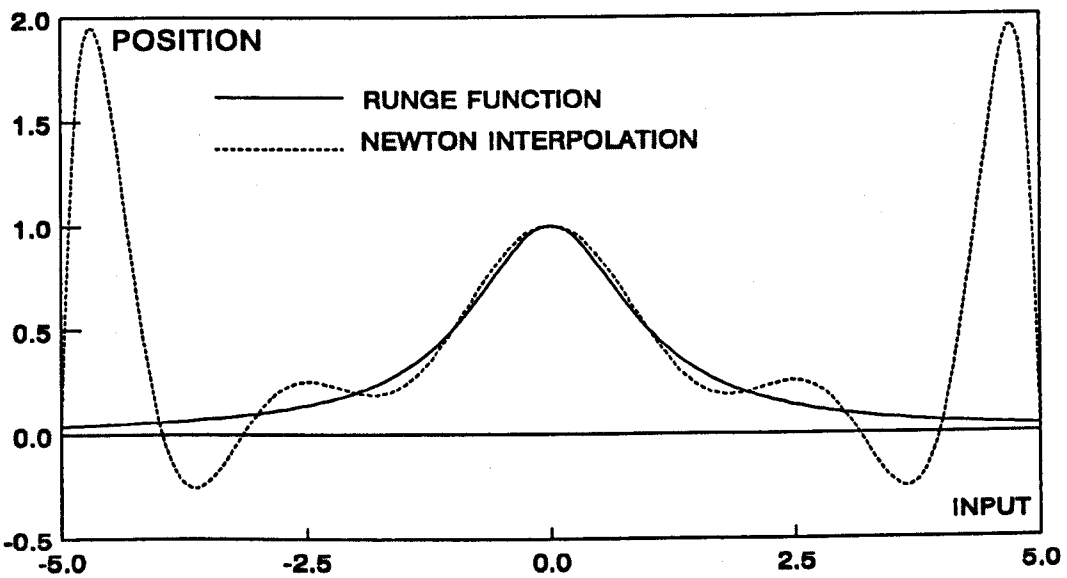


Fig.2.10. Newton interpolation for Runge function.

2.8.3 Lagrange interpolating polynomial

The interpolation polynomial can be written in a variety of forms, and among these the Newton form is probably the most convenient and efficient. However, the form known by the name of **Lagrange** can give a slightly better interpolation [2.6]. Suppose that we want to interpolate a set of arbitrary data points with fixed nodes $\{x_0, x_1..x_n\}$. We define a system of n special polynomials of degree n known as *Cardinal functions.* These are donated by $\{L_0, L_1..L_n\}$.

$$P(x) = \sum_{i=0}^{n} L_i(x) y_i \tag{2.94}$$

Recall equation (2.88) which is the simple form for a polynomial interpolation between two points.

$$P(x) = \frac{(x - x_1)}{(x_0 - x_1)} y_0 + \frac{(x - x_0)}{(x_1 - x_0)} y_1$$

Then

$$L_0(x) = \left(\frac{x - x_1}{x_0 - x_1} \right), \quad L_1(x) = \left(\frac{x - x_0}{x_1 - x_0} \right) \tag{2.95}$$

This result can be generalised to $(n + 1)$ data points

$$L_i(x) = \prod_{\substack{j=0 \\ j <> i}}^{n} \frac{(x - x_j)}{(x_i - x_j)} \tag{2.96}$$

see [2.7] for proof, where we have the property

$$L_i(x_j) = 0 \quad if \quad i <> j, \qquad L_i(x_i) = 1 \qquad if \quad i = j$$

The idea is to multiply each y_i by a polynomial that is 1 at x_i and 0 at the other n nodes and then take the sum of these $(n + 1)$ polynomials to get the unique interpolation polynomial of degree n. The resultant formula is called the Lagrange interpolation polynomial.

This formula indicates that $L_i(x)$ is the product of n linear factors:

$$L_i(x) = \left(\frac{x - x_0}{x_i - x_0} \right) \left(\frac{x - x_1}{x_i - x_1} \right) \cdots \left(\frac{x - x_{i-1}}{x_i - x_{i-1}} \right) \left(\frac{x - x_{i+1}}{x_i - x_{i+1}} \right) \cdots \left(\frac{x - x_n}{x_i - x_n} \right) \tag{2.97}$$

Finally the equation becomes

$$P(x) = \sum_{i=0}^{n} L_i(x) y_i = \sum_{i=0}^{n} \prod_{\substack{k=0 \\ k \neq i}}^{n} \frac{(x - x_k)}{(x_i - x_k)} y_i \qquad (2.98)$$

Fig.2.11 shows a Lagrange interpolation for the Runge function, where the interpolation function has the same conditions given for Fig.2.10. When the two curves are compared it can be seen that there is no significant difference. Both methods give similar solutions for this case.

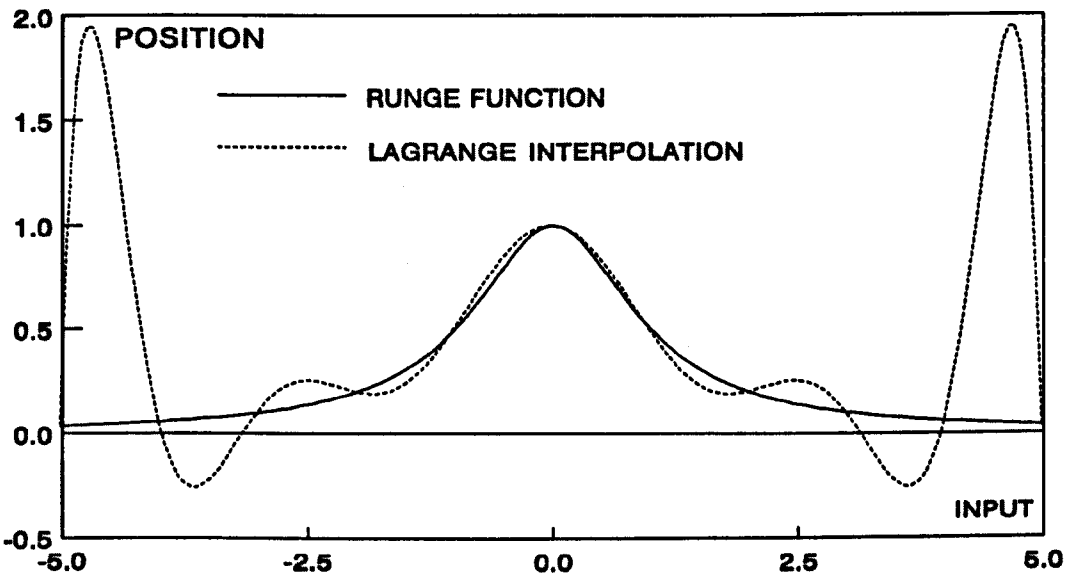


Fig.2.11 Lagrange interpolation for Runge function.

The Lagrange polynomials are not very practical for numerical work. One important disadvantage is that the coefficients of the function have to be recalculated for each data point, since their values change for each point. On the other, in the power form or Newton's form only one calculation of the coefficients is needed for the interpolation of all the values of $P(x)$. Therefore computation may become laborious in the Lagrange case.

A general algorithm is given below to calculate the Lagrange interpolation function.

Algorithm for Lagrange interpolation

```
For k=0 to n Do
Begin
Xbar[k]=X[0]+k*(X[n]-X[0])/n
sum=0.0
    For i=0 to n do
    Begin
    L[i]=1.0
        For j=0 to n do
        Begin
        If(j<>i) then L[i]=L[i]*(Xbar[k]-X[j])/(X[i]-X[j])
        End
    sum=sum+L[i]*Y[i]
    End
Lagrange[k]=sum
End
```

2.8.4 Hermite Interpolating polynomial

In certain situations the values of the first derivative $\dot{f}(x)$ may also be available in the data set to be interpolated. In these circumstances it is sometimes useful to consider a polynomial which considers not only the function values but also includes the first derivative values of $f(x)$ at appropriate points. Such a polynomial is the Hermite interpolation polynomial.

Let $X_i(i = 0, 1, 2...n)$ denote a set of $(n + 1)$ distinct points and suppose that y_i and $\dot{y}_i$ denote the numerical values $f(x)$ and $\dot{f}(x)$ at x_i A polynomial H of degree $(2n + 1)$ has a total $(2n + 2)$ coefficients, so one might expect to be able to choose these to satisfy the $(2n + 2)$ conditions $H(x_j) = y_j$ and $\dot{H}(x_j) = \dot{y}_j \{j = 0, 1...n\}$.

The interpolating polynomial $H(x)$ can be expressed in the form:

$$H(x) = \sum_{i=0}^{n} r_i(x)y_i + \sum_{i=0}^{n} s_i(x)\dot{y} \qquad (2.99)$$

where r_i and s_i $(i = 0, 1...n)$ are polynomials of degree at most $(2n + 1)$. Now the condition $H(x_j) = y_j$ is satisfied if

$$r_i(x_j) = \left\{ \begin{matrix} 0 & if \quad i <> j \\ 1 & if \quad i = j \end{matrix} \right\} \tag{2.100}$$

and $s_i(x_j) = 0$ for all i

so that

$$H(x_j) = \sum_{i=0}^{n} r_i(x_j) y_i + \sum_{i=0}^{n} s_i(x_j) \dot{y}_i = r_j(x_j) y_j = y_j \tag{2.101}$$

where $r_i(x)$ and $s_i(x)$ are determined as [2.8]

$$r_i(x) = \{1 - 2(x - x_i)\dot{L}_i(x_i)\} (L_i(x))^2 \tag{2.102}$$

and

$$s_i(x) = (x - x_i)(L_i(x))^2 \tag{2.103}$$

Hence the Hermite polynomial becomes

$$H(x) = \sum_{i=0}^{n} (1 - 2(x - x_i)\dot{L}_i(x_i))(L_i(x))^2 y_i \quad + \quad \sum_{i=0}^{n} (x - x_i)(L_i(x))^2 \dot{y}_i \tag{2.104}$$

since

$$L_i(x) = \prod_{\substack{j=0 \\ j <> i}}^{n} \frac{(x - x_j)}{(x_i - x_j)}$$

and its derivative would be

$$\dot{L}_i(x) = \sum_{\substack{j=0 \\ j <> i}}^{n} \left(\frac{1}{x_i - x_j} \right) \prod_{\substack{k=0 \\ k <> i \\ k <> j}}^{n} \left(\frac{x - x_k}{x_i - x_k} \right) \tag{2.105}$$

to obtain $\dot{L}_i(x_i)$ replace (x) by (x_i) in the above equation therefore we obtain

$$\prod_{\substack{k=0 \\ k<>i \\ k<>j}}^{n}\left(\frac{x_i - x_k}{x_i - x_k}\right) = 1 \tag{2.106}$$

and

$$\dot{L}_i(x_i) = \sum_{\substack{j=0 \\ j<>i}}^{n}\left(\frac{1}{x_i - x_j}\right) \tag{2.107}$$

Also, the condition $\dot{H}(x_j) = \dot{y}_j$ is satisfied if

$r_i(x_j) = 0$ for all i

$$s_i(x_j) = \left\{\begin{array}{l} 0 \quad if \quad i <> j \\ 1 \quad if \quad i = j \end{array}\right\} \tag{2.108}$$

$$\dot{H}(x_j) = \sum_{i=0}^{n} r_i(x_j)y_i + \sum_{i=0}^{n} s_i(x_j)\dot{y}_i = s_j(x_j)\dot{y}_j = \dot{y}_j \tag{2.109}$$

Fig.2.12 shows an example of the Hermite interpolation where it is used to draw the Runge curve. The interpolation uses 20 data points, ten of them are position values and the other ten are the first derivative values. The degree of the function is 19.

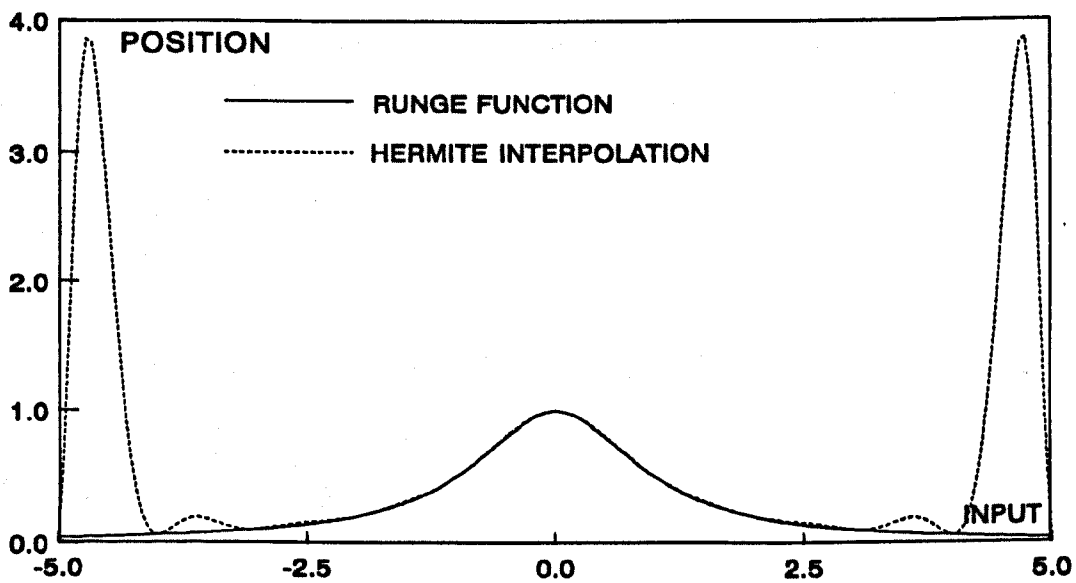


Fig.2.12 Hermite interpolation for Runge function.

The curve for the interpolation matches the required curve better at the middle points in comparison to the curves obtained using Newton and Lagrange interpolation, however, larger oscillations are evident at the two ends of the interval.

2.9 Rational interpolation

As candidates for the efficient representation of mathematical functions using easily computed expressions, rational functions consisting of quotients of two polynomials are often to be preferred to polynomials. Unlike polynomials, rational functions can still be used successfully to approximate singular or nearly singular functions, possibly over infinite intervals [2.7]. Also it has been found that, in general, rational interpolation can achieve a smaller maximum error for the same amount of computation than polynomial interpolation.

Rational interpolation can be denoted by

$$R(x) = \frac{P(x)}{Q(x)} = \frac{p_0 + p_1 x + p_2 x^2 + \ldots + p_u x^u}{q_0 + q_1 x + q_2 x^2 + \ldots + q_v x^v} \tag{2.110}$$

The first constant term in the denominator can be taken as unity without loss of generality, since we can always convert to this form by dividing numerator and denominator by q_0. The constant q_0 will generally not be zero, for in that case the function would be undefined at $(x = 0)$.

$$R(x) = \frac{P(x)}{Q(x)} = \frac{p_0 + p_1 x + p_2 x^2 + \ldots + p_u x^u}{1 + q_1 x + q_2 x^2 + \ldots + q_v x^v} \tag{2.111}$$

where u and v are the degrees of numerator and denominator, so that $R(x)$ satisfies for $(n + 1)$ different support abscissae $(x_0, x_1, x_2, \ldots, x_n)$ and given function values $(y_0, y_1, y_2, \ldots, y_n,)$ for the following conditions.

$$R(x_i) = y_i \qquad (i = 0, 1, 2, \ldots, n)$$

The number of interpolation conditions should coincide with the number of unknown coefficients, so we must have

$$u + v + 2 = n + 1 \tag{2.112}$$

conditions.

In Fig.2.13, the rational interpolation function is applied to the Runge example. The curve is obtained using 20 position data points. The result is quite acceptable. The function produces this difficult curve without any undesirable oscillation. This example demonstrates the difference between rational interpolation and other interpolations.

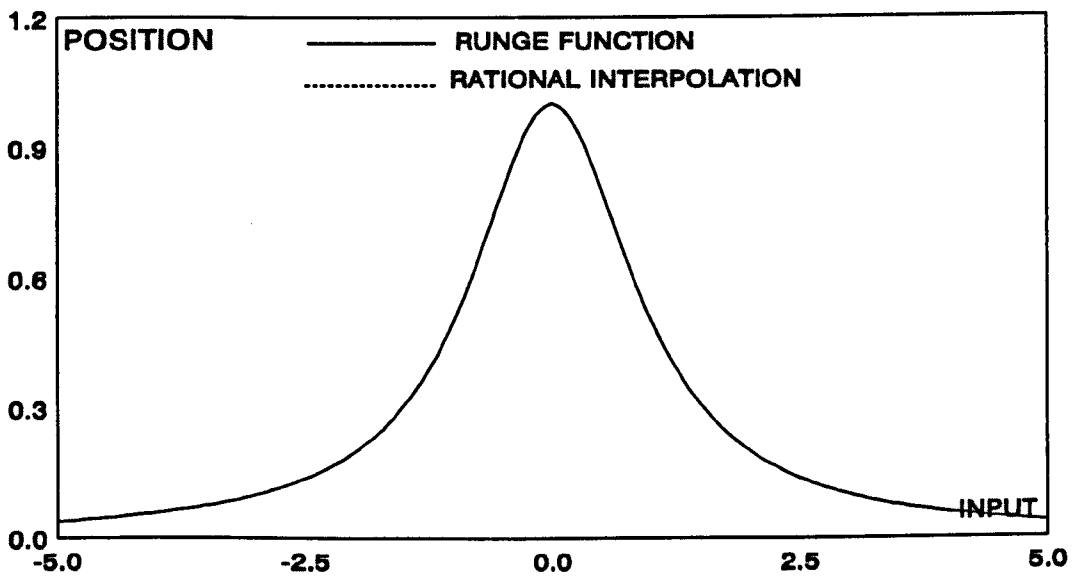


Fig.2.13. Rational interpolation for Runge function.

It is clear that rational interpolation possesses certain powerful advantages over polynomial interpolation since it produces curves which do not exhibit oscillations between the boundary conditions and it can absorb certain types of singularity.

There is, however, a need for rather greater caution in attempting to interpolate using rational functions. Some drawbacks in their use include:

(i) There are numerical difficulties which cannot be easily overcome. Rational functions are less convenient for certain analytical manipulations, such as differentiation and integration. In general, the interpolation function is also required to satisfy derivative values up to the third derivative for motion design applications. Differentiation of the rational functions may become very difficult to perform beyond the first derivative. Let $P(x)$ and $Q(x)$ be the functions for the numerator and denominator respectively. The rational function and its derivatives then would be:

$$R(x) = \frac{P(x)}{Q(x)} \tag{2.113}$$

$$\dot{R}(x) = \frac{\dot{P}(x)}{Q(x)} - \frac{P(x)\dot{Q}(x)}{Q(x)^2} \tag{2.114}$$

$$\ddot{R}(x) = \frac{-2\dot{P}(x)\dot{Q}(x)}{Q^2(x)} + \frac{2P(x)\dot{Q}^2(x)}{Q^3(x)} + \frac{\ddot{P}(x)}{Q(x)} - \frac{P(x)\ddot{Q}(x)}{Q^2(x)} \tag{2.115}$$

$$\dddot{R}(x) = \frac{6\dot{P}(x)\dot{Q}^2(x)}{Q^3(x)} - \frac{6P(x)\dot{Q}^3(x)}{Q^4(x)} - \frac{3\dot{Q}(x)\ddot{P}(x)}{Q^2(x)} - \frac{3\dot{P}(x)\ddot{Q}(x)}{Q^2(x)} + \frac{6P(x)\dot{Q}(x)\ddot{Q}(x)}{Q^3(x)} + \frac{\dddot{P}}{Q(x)} - \frac{P(x)\dddot{Q}(x)}{Q^2(x)} \tag{2.116}$$

putting the open forms of $P(x)$ and $Q(x)$ into the equations makes the solution much more difficult. For example, consider a rational function which is required to satisfy eight boundary conditions so the number of terms in the rational function must be equal to eight. $P(x)$ is selected as a four degree polynomial, that means it has five terms, and $Q(x)$ is a two degree of polynomial function. The rational function and its derivatives including first second and third are given below.

$$R(x) = \frac{p_0 + p_1x + p_2x^2 + p_3x^3 + p_4x^4}{q_0 + q_1x + q_2x^2} \tag{2.117}$$

$$\dot{R}(x) = \frac{p_1 + 2p_2x + 3p_3x^2 + 4p_4x^3}{q_0 + q_1x + q_2x^2} - \frac{(q_1 + 2q_2x)\ \ (p_0 + p_1x + p_2x^2 + p_3x^3 + p_4x^4)}{(q_0 + q_1x + q_2x^2)^2} \quad (2.118)$$

$$\ddot{R}(x) = \frac{2p_2 + 6p_3x + 12p_4x^2}{q_0 + q_1x + q_2x^2} - \frac{2(q_1 + 2q_2x)\ \ (p_1 + 2p_2x + 3p_3x^2 + 4p_4x^3)}{(q_0 + q_1x + q_2x^2)^2}$$

$$+ \frac{2(q_1 + 2q_2x)^2(p_0 + p_1x + p_2x^2 + p_3x^3 + p_4x^4)}{(q_0 + q_1x + q_2x^2)^3}$$

$$- \frac{(2q_2)(p_0 + p_1x + p_2x^2 + p_3x^3 + p_4x^4)}{(q_0 + q_1x + q_2x^2)^2} \quad (2.119)$$

$$\dddot{R}(x) = -\frac{3(q_1 + 2q_2x)(2p_2 + 6p_3x + 12p_4x^2)}{(q_0 + q_1x + q_2x^2)^2} + \frac{(6p_3 + 24p_4x)}{q_0 + q_1x + q_2x^2}$$

$$+ \frac{6(q_1 + 2q_2x)^2(p_1 + 2p_2x + 3p_3x^2 + 4p_4x^3)}{(q_0 + q_1x + q_2x^2)^3} - \frac{6q_2(p_1 + 2p_2x + 3p_3x^2 + 4p_4x^3)}{(q_0 + q_1x + q_2x^2)^2}$$

$$- \frac{6(q_1 + 2q_2x)^3(p_0 + p_1x + p_2x^2 + p_3x^3 + p_4x^4)}{(q_0 + q_1x + q_2x^2)^4}$$

$$+ \frac{12q_2(q_1 + 2q_2x)(p_0 + p_1x + p_2x^2 + p_3x^3 + p_4x^4)}{(q_0 + q_1x + q_2x^2)^3} \quad (2.120)$$

The above equations show that the manipulation of rational functions for a general solution is quite difficult. However, a computer program has been developed (see Appendix A) for rational interpolations. This can be used to examine the features of the function such as oscillations and smoothness for different sets of data. The program can be used to interpolate any set of data but for position boundary conditions only.

(ii) In contrast to polynomial interpolation it cannot be shown that a rational interpolation function $R_{(x)}$ always exists that solves the interpolation problem for an arbitrary set of given data [2.7]. For example, consider the case where an interpolation is required between two points given by the boundary conditions shown in Table 2.2.

Let $u = 1$ and $v = 2$

$$R(x) = \frac{P(x)}{Q(x)} = \frac{p_0 + p_1 x}{1 + q_1 x + q_2 x^2} \tag{2.121}$$

putting the position boundary conditions into this equation gives

$p_0 = 0$ and $p_1 = 1 + q_1 + q_2$

The other coefficients can be determined by taking the derivatives of the function and making them equal to the boundary conditions shown in the second column of Table 2.2.

Differentiating (2.121)

$$\dot{R}(x) = \frac{\dot{P}(x)}{Q(x)} - \frac{P(x)\dot{Q}(x)}{Q^2(x)}$$

$$= \frac{p_1}{1 + q_1 x + q_2 x^2} - \frac{(p_0 + p_1 x)(q_1 + 2q_2 x)}{(1 + q_1 x + q_2 x^2)^2} \tag{2.122}$$

and putting the values of p_0 , p_1 and derivative boundary conditions into the above equation we can obtain

$q_1 = -2$ and $q_2 = 1$

then the interpolation function can be obtained in terms of calculated coefficients as shown below.

$$R(x) = \frac{0}{1 - 2x + x^2} = 0 \quad (except \quad for \quad x = 1) \tag{2.123}$$

TABLE 2.2. Constraints for rational interpolation

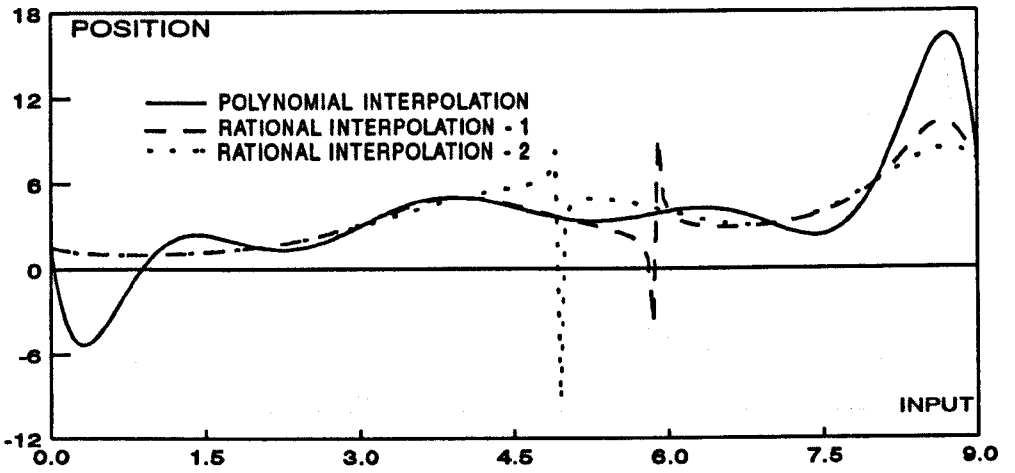
F(x)	F(x)'	F(x)''	F(x)'''	x
0.00	0.00	--	--	0.0
1.00	0.00	--	--	1.0

(iii) Rational functions may have poles for some values of x_i which make the denominator of the function zero. The function tends to infinity for these values and the resulting interpolation will in general no longer be satisfactory for the user. However, the only constraint is $(u + v + 2 = n + 1)$ to determine the powers of numerator and the denominator so the user may be able to shift these poles out of the required interval by changing these powers. Some examples are shown in Fig.2.14-15. These are also used to compare the amount of oscillations obtained with polynomial interpolation for the same set of data.

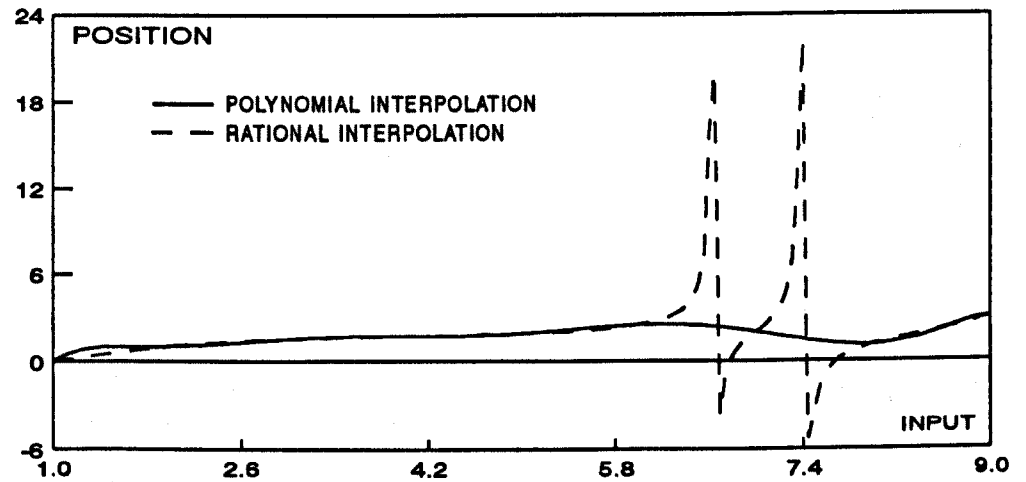
The first example (see Fig.2.14(a)) shows the polynomial curve (solid) and rational curve (dashed). In comparison, the rational curve is smoother than the polynomial curve but it displays two poles causing discontinuity in the curve. The dotted line is another rational function for the same data with different powers. Despite using different powers for the function the pole still remains. Changing the powers has simply altered the position of the pole as can be seen.

The second example shows a successful attempt to shift the poles out of the desired region. In the Fig.2.14(b) it is seen that the rational curve (shown dashed) has two poles. These poles are shifted by using different powers for the function with the result shown in the Fig.2.14(c). The solid line is obtained with polynomial interpolation. The rational curve is slightly better than the polynomial curve.

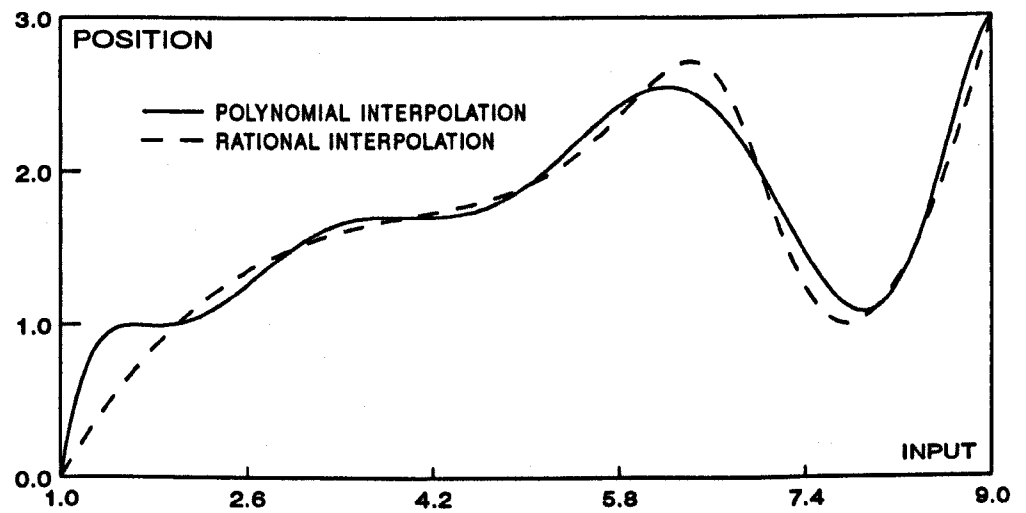
Another example is shown in the Fig.2.15(a) where a polynomial curve (shown solid) has a large oscillation while the rational curve has a pole. However, the pole is removed by means of changing the powers of the function but there is nothing to be done to improve the oscillation of the polynomial curve (see Fig.2.15(b)). The same rational curve is shown with a larger scale in Fig.2.15(c) to show how smooth and efficient such curves can be.



a) example - 1

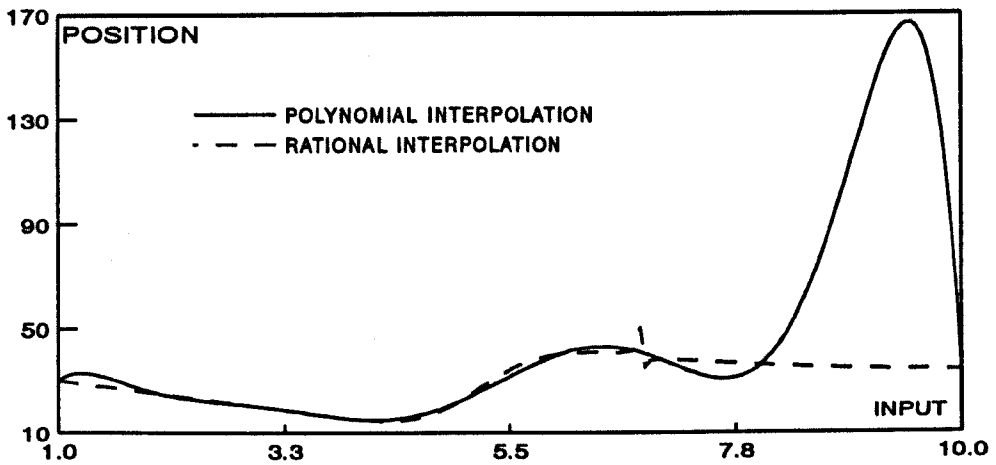


b) example - 2

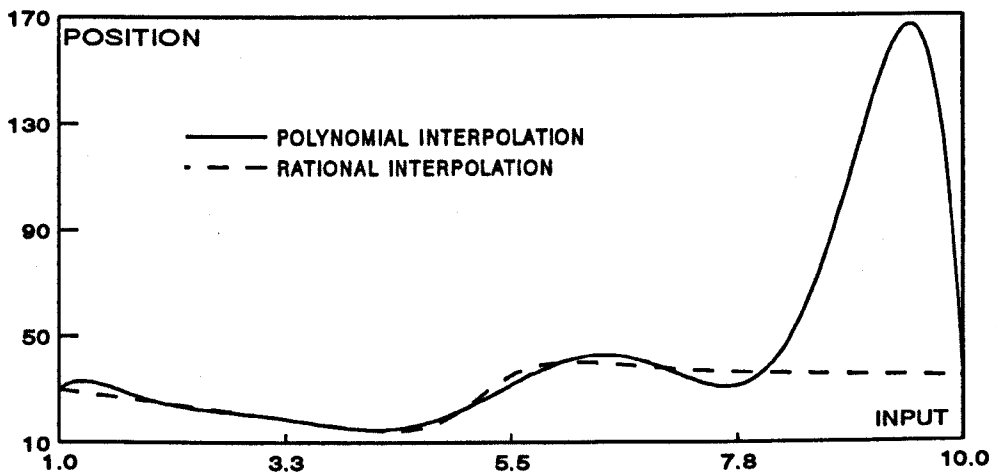


c) example -3

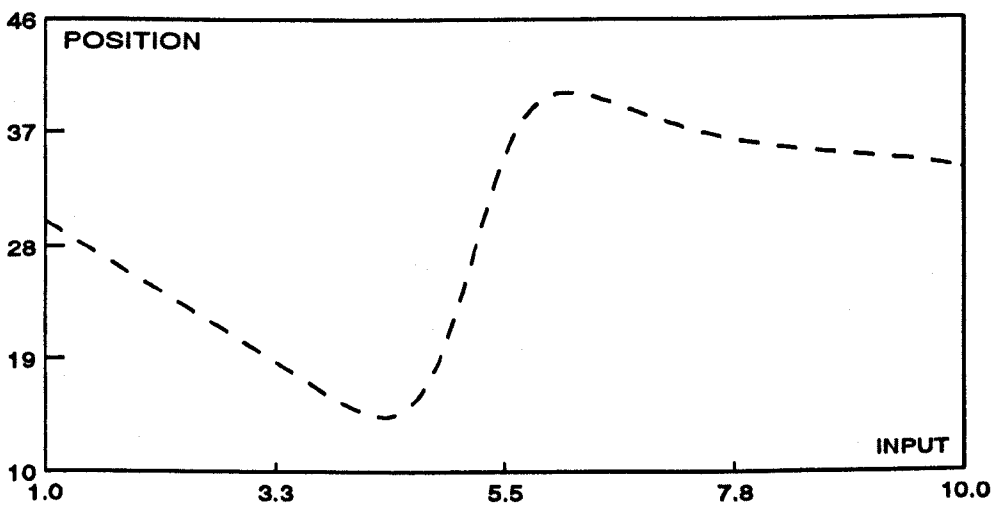
Fig.2.14 Rational and polynomial interpolations



a) example - 4n



b) example - 5



c) example -6

Fig.2.15 Rational and polynomial interpolations

In conclusion, in spite of the previously mentioned difficulties, rational functions are more accurate and yield better results, compared with polynomial interpolation in many cases.

2.10 Spline functions

One approach to avoid oscillations with high degree polynomial interpolation is to divide the curve into a series of intervals. Different polynomial functions can then be developed for each interval. The scheme is known as piecewise polynomial interpolation. There are, however, significant disadvantages associated with the piecewise approach because for piecewise interpolation only continuity of position can be guaranteed.

Piecewise polynomials are continuous in the function while their first derivative, second derivative and even higher derivative values can be matched at the ends of each adjoining interval. Piecewise polynomials possessing these properties are known as splines. Although there are many spline types such as B-splines, G-splines, quadratic splines, periodic splines etc, the cubic and quintic forms have been covered in the study.

2.10.1 Cubic splines

The generation of interpolation spline curves is a useful and powerful tool in computer-aided design. Although the cubic spline has many sophisticated mathematical properties [1.9] and [2.8], the curves sometimes display undesirable oscillations. Several methods have been developed to control the shape of the cubic interpolation such as those in [2.9]-[2.16]. Cubic splines are smooth and continuous in position, slope and curvature. Their curves are much more predictable in comparison to curves of polynomial interpolation because of their low degree. The term "spline" is derived from the name of a device traditionally used by draftsmen consisting of a flexible strip of

steel used to form a smooth curve through a set of data points. Since a steel strip is subject to the laws of elastic deflection its shape is given by a cubic polynomial between the data points.

A procedure to generate a series of cubic polynomials that pass through a set of $(1,2...n)$ data points and have continuity of slope and curvature is described below. Equations for an interval which lies between two points (x_i, y_i) and (x_{i+1}, y_{i+1}) are:

$$y = a_i(x - x_i)^3 + b_i(x - x_i)^2 + c_i(x - x_i) + d_i \qquad (i = 0, 1, 2, 3...n) \qquad (2.124)$$

taking the first and second derivative of this equation gives

$$\dot{y} = 3a_i(x - x_i)^2 + 2b_i(x - x_i) + c_i \qquad (2.125)$$

$$\ddot{y} = 6a_i(x - x_i) + 2b_i \qquad (2.126)$$

Equating the function to the two data points and substituting h_i for $(x_{i+1} - x_i)$ gives

$$y_i = d_i \qquad (2.127)$$

$$y_{i+1} = a_i h_i^3 + b_i h_i^2 + c_i h_i + d_i \qquad (2.128)$$

$$\dot{y}_i = c_i \qquad (2.129)$$

$$\dot{y}_{i+1} = 3a_i h_i^2 + 2b_i h_i + c_i \qquad (2.130)$$

$$\ddot{y}_i = 2b_i \qquad (2.131)$$

$$\ddot{y}_{i+1} = 6a_i h_i + 2b_i \qquad (2.132)$$

In order to satisfy the interpolation and provide continuity for the first and second derivatives at the ends of the interval, it is most convenient to

express the coefficients a_i, b_i, c_i and d_i by means of the given function values $(y_i,\ y_{i+1})$ and the unknown second derivatives $(\ddot{y}_i,\ \ddot{y}_{i+1})$ at the two ends of the interval $(x_i - x_{i+1})$, so that

$$a_i = \frac{1}{6h_i}(\ddot{y}_{i+1} - \ddot{y}_i) \tag{2.133}$$

$$b_i = \frac{1}{2}\ddot{y}_i \tag{2.134}$$

$$c_i = \frac{1}{h_i}(y_{i+1} - y_i) - \frac{1}{6}h_i(\ddot{y}_{i+1} + 2\ddot{y}_i) \tag{2.135}$$

$$d_i = y_i \tag{2.136}$$

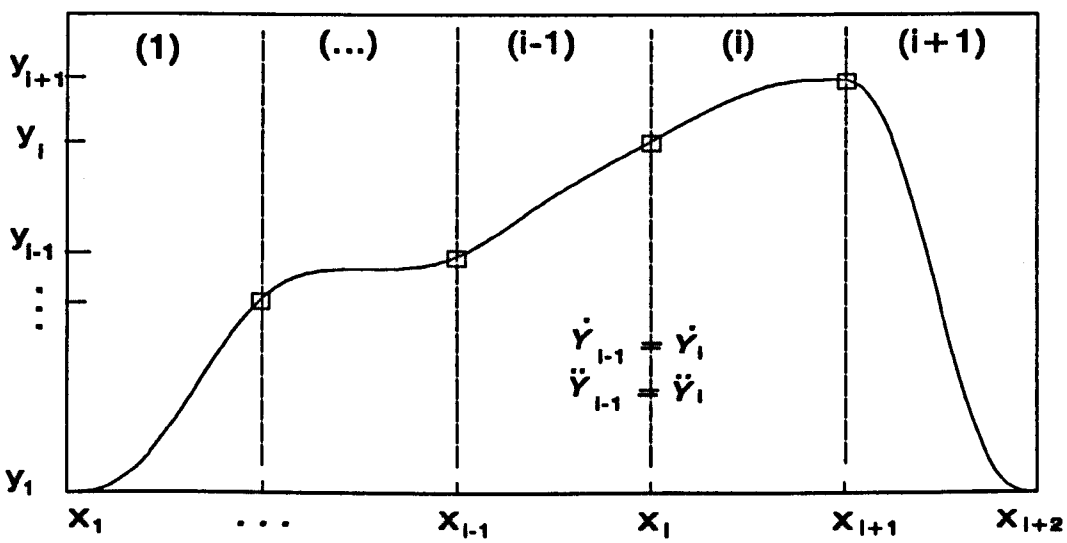


Fig.2.16. Construction of the cubic spline.

We can invoke the condition that the slope of the two functions which join at $(x_i,\ y_i)$ must be equal, see Fig.2.16. The slope of the function for the i^{th} interval at the left end is

$$\dot{y}_i = c_i \tag{2.137}$$

In the previous interval from x_{i-1} *to* x_i the slope at its right end will be

$$\dot{y}_i = 3a_{i-1}h_{i-1}^2 + 2b_{i-1}h_{i-1} + c_{i-1} \tag{2.138}$$

Equating equations (2.137) and (2.138) and substituting for a_i, b_i, c_i and d_i gives

$$\dot{y}_i = \frac{y_{i+1} - y_i}{h_i} - \frac{2h_i\ddot{y}_i + h_i\ddot{y}_{i+1}}{6}$$

$$= 3\frac{(\ddot{y}_i - \ddot{y}_{i-1})}{6h_{i-1}}h_{i-1}^2 + 2\frac{(\ddot{y}_{i-1})}{2}h_{i-1} + \frac{y_i - y_{i-1}}{h_{i-1}} - \frac{2h_{i-1}\ddot{y}_{i-1} + h_{i-1}\ddot{y}_i}{6} \qquad (2.139)$$

Simplifying the equation we obtain

$$h_{i-1}\ddot{y}_{i-1} + (2h_{i-1} + 2h_i)\ddot{y}_i + h_i\ddot{y}_{i+1} = 6\left(\frac{y_{i+1} - y_i}{h_i} - \frac{y_i - y_{i-1}}{h_{i-1}}\right) \qquad (2.140)$$

Eq.2.140 applies at each interval point from ($i = 2$ *to* $i = n - 1$). This gives ($n - 2$) equations and "n" unknown values of $\ddot{y}_i$. We can get two additional equations for the end points of the whole curve involving $\ddot{y}_1$ and $\ddot{y}_n$. Arbitrary values may be specified but for a natural spline they should equal zero.

Some examples of interpolation using cubic splines are given below with their data. Dashed and solid lines represent cubic splines and polynomial interpolations respectively. The curves of cubic spline are smoother than the curves of polynomial interpolation in each example.

It is possible to construct splines other than cubic, such as quadratic (with continuity for the function and the first derivative values only) and quintic spline (with continuity of derivatives up to and including those of fourth degree).

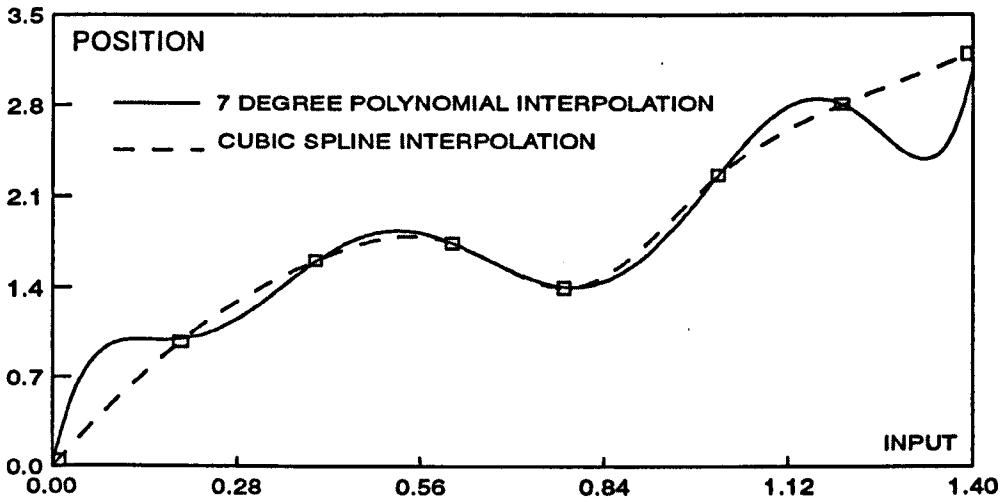


Fig.2.17 Comparison of cubic spline with polynomial interpolation.

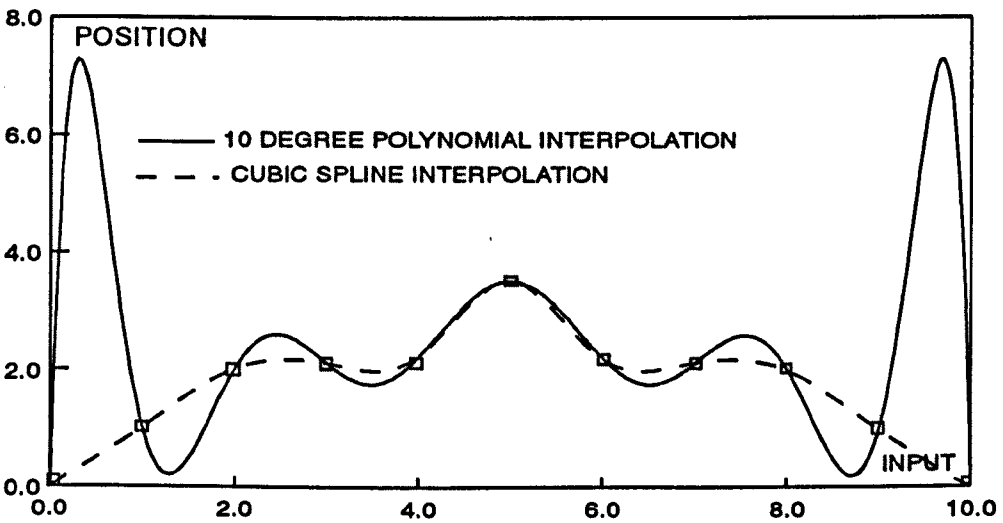


Fig.2.18 Comparison of cubic spline with polynomial interpolation.

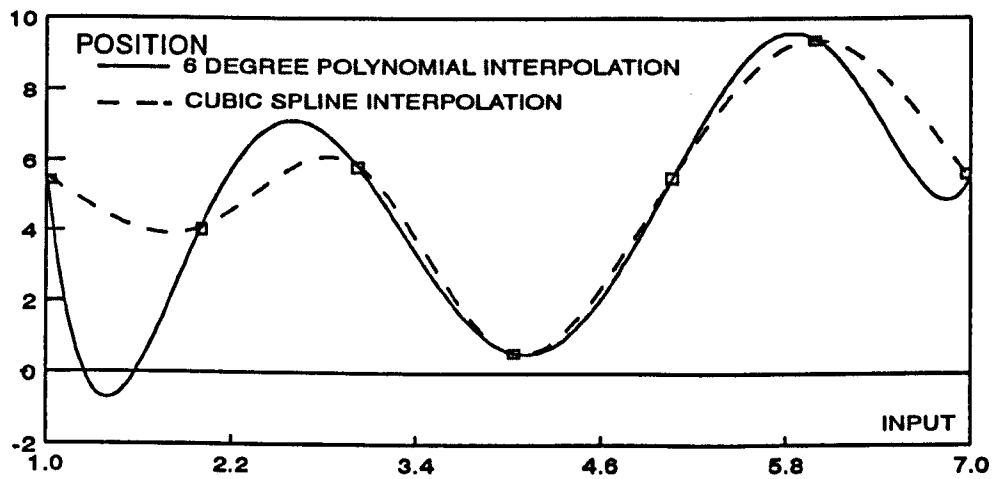


Fig.2.19 Comparison of cubic spline with polynomial interpolation.

TABLE 2.3. Data for spline interpolation (Fig.2.17)				
F(x)	$F(x)'$	$F(x)''$	$F(x)'''$	x
0.00	3.0	--	--	0.0
1.00	--	--	--	0.2
1.60	--	--	--	0.4
1.75	--	--	--	0.6
1.40	--	--	--	0.8
2.20	--	--	--	1.0
2.80	--	--	--	1.2
3.20	2.0	--	--	1.4

TABLE 2.4. Data for spline interpolation (Fig.2.18)				
F(x)	$F(x)'$	$F(x)''$	$F(x)'''$	x
0.00	--	0.0	--	0.0
1.00	--	--	--	1.0
2.00	--	--	--	2.0
2.10	--	--	--	3.0
2.20	--	--	--	4.0
3.50	--	--	--	5.0
2.20	--	--	--	6.0
2.10	--	--	--	7.0
2.00	--	--	--	8.0
1.00	--	--	--	9.0
0.00	--	0.0	--	10.0

TABLE 2.5. Data for spline interpolation (Fig.2.19)				
F(x)	F(x)'	F(x)''	F(x)'''	x
5.541	--	0.0	--	1.0
4.074	--	--	--	2.0
5.900	--	--	--	3.0
0.611	--	--	--	4.0
5.000	--	--	--	5.0
9.388	--	--	--	6.0
5.541	--	0.0	--	7.0

2.10.2 Quintic splines

Quintic splines are extensions of cubic splines. In addition to the condition of continuity of the first and second derivative of the function, the third and the fourth derivatives are also made continuous. The function can be easily manipulated to produce distinct shapes of curves by using the advantages of free boundary conditions at the ends of the curve. Additionally sacrificing the continuity at some points for higher derivatives allows the user to specify some additional derivative boundary conditions at the join positions. Because of these advantages, the quintic spline has been used for kinematic design, especially in cam design by MacCarthy [2.18] and [2.19]. It is shown that standard cam laws can be approximated accurately with a small number of points and appropriate boundary conditions.

Using the notation developed earlier for the cubic spline, a scheme to generate a series of quintic polynomials that pass through $(0, 1, 2...n)$ data points and have continuity up to the fourth derivative is given below.

A quintic function and its derivatives which lie between the points (x_i, y_i) and (x_{i+1}, y_{i+1}) are:

$$y_i = a_i h_i^5 + b_i h_i^4 + c_i h_i^3 + d_i h_i^2 + e_i h_i + f_i \tag{2.141}$$

$$\dot{y}_i = 5a_i h_i^4 + 4b_i h_i^3 + 3c_i h_i^2 + 2d_i h_i + e_i \tag{2.142}$$

$$\ddot{y}_i = 20a_i h_i^3 + 12b_i h_i^2 + 6c_i h_i + 2d_i \tag{2.143}$$

$$\dddot{y}_i = 60a_i h_i^2 + 24b_i h_i + 6c_i \tag{2.144}$$

$$\ddddot{y}_i = 120a_i h_i + 24b_i \tag{2.145}$$

Each function for each interval has six unknown and one known variables and furthermore we have the following properties for each point between the end points:

$$\dot{y}_{i-1(e)} = \dot{y}_{i(s)} \qquad (i = 2, \ldots n-1) \tag{2.146}$$

$$\ddot{y}_{i-1(e)} = \ddot{y}_{i(s)} \qquad (i = 2, \ldots n-1) \tag{2.147}$$

$$\dddot{y}_{i-1(e)} = \dddot{y}_{i(s)} \qquad (i = 2, \ldots n-1) \tag{2.148}$$

$$\ddddot{y}_{i-1(e)} = \ddddot{y}_{i(s)} \qquad (i = 2, \ldots n-1) \tag{2.149}$$

where i=segment number, e=end of a segment and s=start of a segment.

Then there are $6(n-1)$ unknown coefficients and $6(n-2)+2$ equations therefore four end conditions are necessary for the solution. These end conditions can be stipulated at both ends of the interpolation curve in terms of any derivative (up to the fourth derivative) of the function.

The main advantage of the quintic spline over the cubic spline is that the quintic spline is more flexible and it tolerates more modifications. Another important advantage is that the motion curves of the quintic spline are continuous at least up to the second derivative through the curve, whereas the continuity of the first and second derivative together can not be guaranteed at the end points of the curves formed from cubic splines. Quintic splines will be compared with the other laws at the end of the next chapter.

CHAPTER 3
POWER FORM OF POLYNOMIAL FUNCTIONS AND MOTION DESIGN STRATEGIES

3.1 Introduction

In this chapter the power form of the polynomial interpolation is discussed in detail. Although this form has many advantages, it is shown to suffer from the major disadvantage that meandering may occur. Some techniques to prevent meandering in the motion curves are introduced. One method to improve interpolation curves is to use dummy boundary conditions, another is to divide the curve into segments. A novel method is to change the powers of the function artificially thereby gaining the freedom to obtain different shapes of curve between the end points of segments without changing the boundary conditions. This enables the user to manipulate the function in order to achieve a definite shape for a motion curve.

3.2 Power form of polynomial functions

The power form of polynomial interpolation is the most suitable form for motion design of all the interpolation functions. This is because with this form it is feasible to specify arbitrary boundary conditions which may include position, velocity, acceleration and even higher derivatives. Manipulation of the function is easy and furthermore it can produce continuous curves up to the required derivative. These advantages make the power form of polynomials particularly appropriate for motion design.

3.2.1 Calculation of boundary conditions

In general, the determination of the coefficients of a polynomial equation which defines a motion, even with included precision points or dummy variables other than the initial and final points, can be dealt with using matrix formulation. The matrices encompass the imposed boundary conditions, the corresponding times for those boundary conditions and the unknown coefficients. The notation for the matrix form is

$$A = T.C \tag{3.1}$$

Where **T** is the matrix containing the known time variables, **A** is the vector containing the boundary conditions and **C** is the vector for the unknown coefficients of the polynomials. The solution for **C** lies with the inverse of the vector **T**. The multiplication of the above equation by the inverse of **T** matrix gives

$$C = T^{-1}.A \tag{3.2}$$

Consider the case where a trajectory is to be designed using a number of design points each one of which is defined at a specified time along the trajectory. Every design point is further specified by a number of boundary conditions which may or may not include values for displacement, velocity, acceleration and jerk. The degree of the polynomial required for the motion is n with $(n + 1)$ coefficients requiring evaluation.

The input variable for motion equations is time which we shall denote as t. Position, velocity, acceleration and jerk equations will be represented by d, v, a and j respectively.

The form of the polynomial function at any design point along the trajectory will conform to the following:

Displacement equation is

$$d_i = C_0 + C_1 t_i + C_2 t_i^2 + C_3 t_i^3 + C_4 t_i^4 + \ldots + C_n t_i^n \tag{3.3}$$

where i is the index number of the boundary condition.

Differentiation of the displacement gives the velocity,

$$v_i = 0 + C_1 + 2C_2 t_i + 3C_3 t_i^2 + 4C_4 t_i^3 + \ldots + nC_n t_i^{n-1} \tag{3.4}$$

Differentiation of the velocity gives the acceleration

$$a_i = 0 + 0 + 2C_2 + 6C_3 t_i + 12C_4 t_i^2 + \ldots + n(n-1)C_n t_i^{n-2} \tag{3.5}$$

The jerk is obtained by differentiating the acceleration

$$j_i = 0 + 0 + 0 + 6C_3 + 24C_4 t_i + \ldots + n(n-1)(n-2)C_n t_i^{n-3} \tag{3.6}$$

The equations are completed for each boundary condition and then assembled into the matrix form.

$$A = T * C$$

$$\begin{pmatrix} d_1 \\ \vdots \\ d_i \\ v_{i+1} \\ \vdots \\ v_j \\ a_{j+1} \\ \vdots \\ a_k \\ j_{k+1} \\ \vdots \\ j_{n+1} \end{pmatrix} = \begin{pmatrix} 1 & t_1 & t_1^2 & t_1^3 & t_1^4 & \cdots & t_1^n \\ \vdots & \vdots & \vdots & \vdots & \vdots & & \vdots \\ 1 & t_i & t_i^2 & t_i^3 & t_i^4 & \cdots & t_i^n \\ 0 & 1 & 2t_{i+1} & 3t_{i+1}^2 & 4t_{i+1}^3 & \cdots & nt_{i+1}^{n-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots & & \vdots \\ 0 & 1 & 2t_j 2 & 3t_j^2 & 3t_j^3 & \cdots & nt_j^{n-1} \\ 0 & 0 & 2 & 6t_{j+1} & 12t_{j+1}^2 & \cdots & n(n-1)t_{j+1}^{n-2} \\ \vdots & \vdots & \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 2 & 6t_k & 12t_k^2 & \cdots & n(n-1)t_k^{n-2} \\ 0 & 0 & 0 & 6 & 24t_{k+1} & \cdots & n(n-1)(n-2)t_{k+1}^{n-3} \\ \vdots & \vdots & \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & 6 & 24t_{n+1} & \cdots & n(n-1)(n-2)t_{n+1}^{n-3} \end{pmatrix} * \begin{pmatrix} C_0 \\ \vdots \\ C_i \\ C_{i+1} \\ \vdots \\ C_j \\ C_{j+1} \\ \vdots \\ C_k \\ C_{k+1} \\ \vdots \\ C_n \end{pmatrix}$$

Although the advantages of using polynomial functions for motion design have been mentioned before, they have an important drawback which is known as **'meandering'**. This means the interpolated curve produces oscillations between the boundary conditions as illustrated in Fig.3.1.

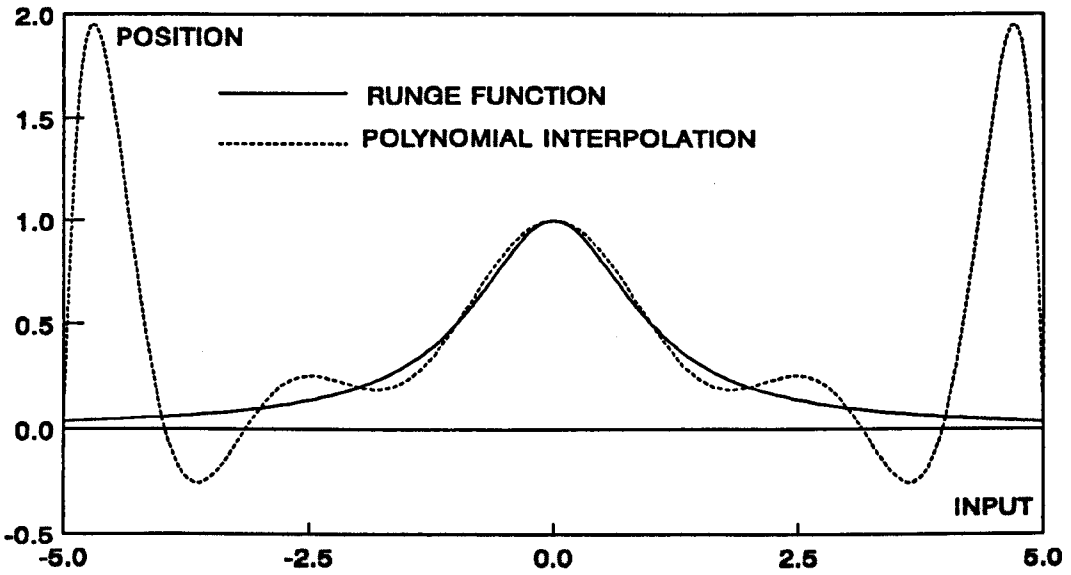


Fig.3.1. An example showing oscillation of the power form of a polynomial.

In the figure the dashed line is obtained using a ten-degree polynomial to interpolate for eleven evenly-spaced data points which satisfy the Runge function. The curve fails to fit the function at any points other than the data points. The oscillations are seen to be particularly bad at the ends of the interval.

3.3 Causes of meandering

Several factors influence the tendency of polynomials to produce oscillations between design points when used for motion design. It has already been mentioned (see Chapter 1) that the amount of oscillation increases as the degree of the interpolating polynomial is increased while equal spacing between the data points can cause larger oscillations.

Another factor, probably the most important, is the shape of the curve to be produced. If the data points are such that the function is forced to follow a curve which has one or more sharp turning points, then most probably

undesirable oscillations will occur, especially if the function has high degree. In general, conditions which affect the occurrence of oscillations when interpolating using polynomial include:

(i) the shape of the required path

(ii) the degree of the polynomial function

(iii) values of the boundary conditions, especially derivative values

(iv) time intervals between design points

(v) round off and truncation errors.

Each one of these contributes to the tendency for a polynomial function to oscillate or meander. The presence of a sharp turning point in the required curve is illustrated with the Runge function shown in Fig.3.1. The interpolation curve shows extreme oscillations at each end of the curve.

Often, the power form of polynomial interpolation may not produce acceptable motion curves for every case. However, the flexibility of polynomials makes it possible to derive new forms which can be employed to obtain improved curves. Some methods to achieve this are discussed below:

3.4 Dummy variable methods

The effect of derivative constraint values at the ends of a segment on the interpolation is illustrated in Fig.3.2. When no derivative constraints are specified, interpolation will give a straight line between the two design points. When values for the first derivatives of the function are specified, the slope of the curve at these points is constrained. The inclusion of some dummy constraints can be used to manipulate the shape of the function between the data points. Generally the worst oscillations occur at the beginning and at the end of the curve, therefore these additional constraints need to be

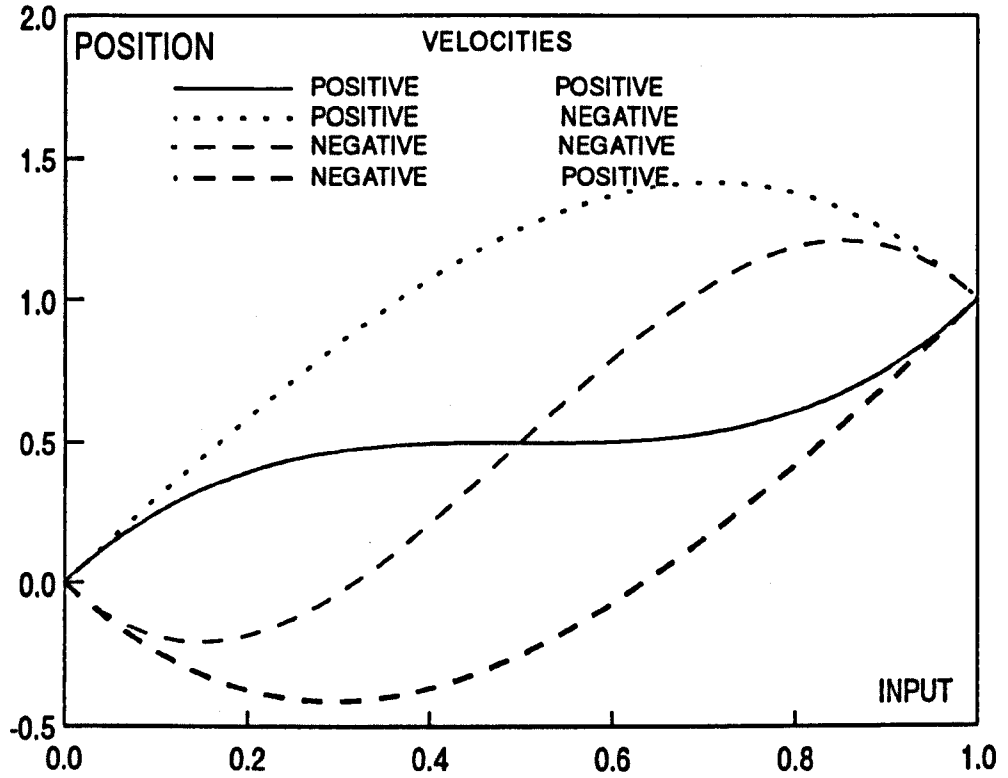


Fig.3.2 The effect of derivative constraints on the interpolation curve

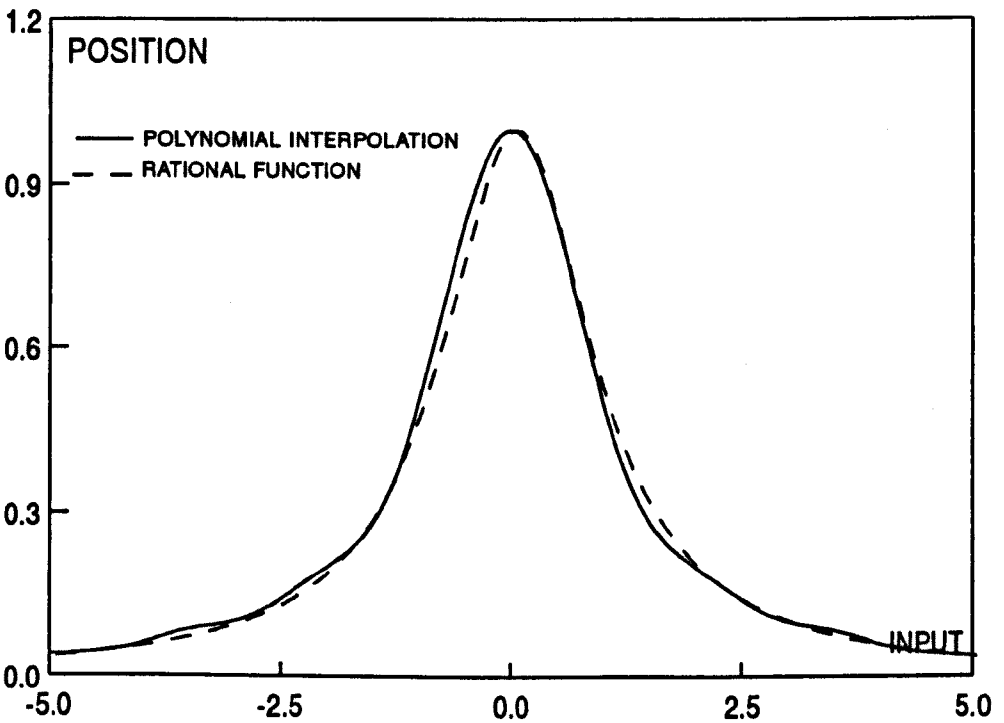


Fig.3.3 Rational function and high degree polynomial interpolation

concentrated near those points. Fig.3.3 includes the same constraints of Fig.3.1 with the addition of ten dummy constraints specified for the first, second and third derivatives as shown in Table 3.1.

TABLE 3.1. Constraints for Fig.3.3

pos.	vel.	acc.	jerk	input
0.0384	0.00	0.00	0.00	-5.00
0.0588	0.05	0.07	--	-4.00
0.1000	--	--	--	-3.00
0.2000	--	--	--	-2.00
0.5000	--	--	--	-1.00
1.0000	--	--	--	0.00
0.5000	--	--	--	1.0
0.2000	--	--	--	2.00
0.1000	--	--	--	3.00
0.0588	-0.05	0.07	--	4.00
0.0384	0.00	0.00	0.00	5.00

Although the degree of resultant interpolation is 20 the resultant curve gives a much better match to the required curve with only negligible amount of oscillation present. Although beneficial in this case, generally the arbitrary specification of additional constraints will not always reduce the amount of oscillation. In some situations the amount of oscillation may be made worse. To obtain a good curve requires some degree of trial and error. However the technique may be useful where the interpolation may be represented by a single function.

3.5 Segmented polynomial

This method has been used to control the profile of motion curves in [3.1] and [3.2]. As previously mentioned the presence of turning points in the interpolation curve can be an important cause of meandering. To overcome this the curve can be split up into parts at these points. In this way the amount of meandering along the curve can be better controlled. Segmented

polynomial interpolation also requires polynomials of lower degree than would be the case if interpolation was carried out over the complete interval. This is also beneficial since the lower degree polynomial interpolations give better curves. Fig.3.4 shows an example where the path is divided into two parts at the turning point seen at the mid-point of the interval. Polynomial interpolation is applied independently for each part and the total curve obtained by adding the second part to the end of the first. The result is very effective. In spite of the high degree (10 degree) of the polynomial used, the interpolation curve matches the original curve perfectly.

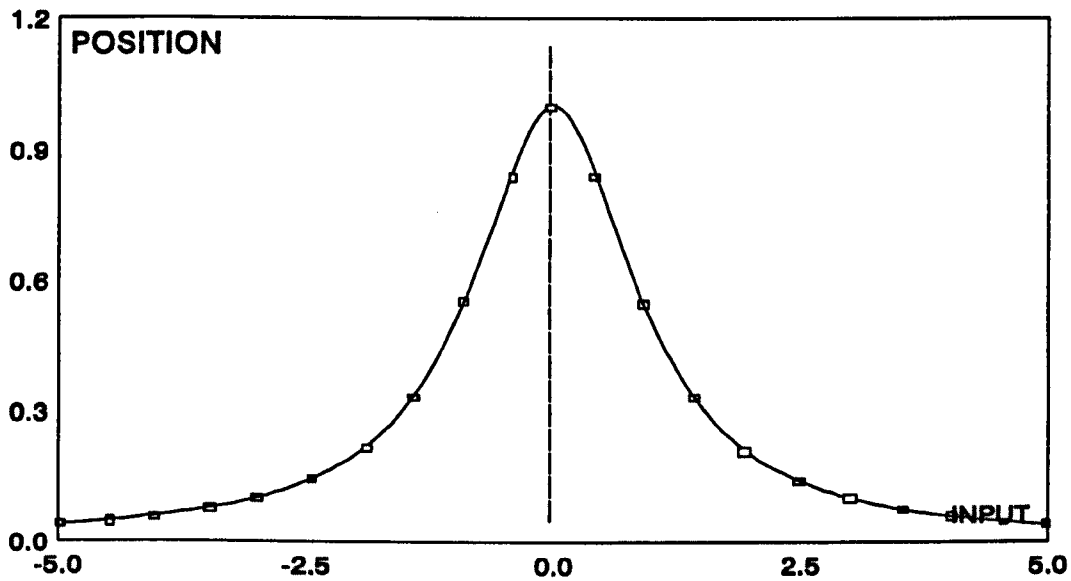


Fig.3.4. Division a curve into segments.

Although dividing a curve into segments at the turning point gave a good result for the above example, in other cases it may not be sufficient. A better and more practical approach is to divide the curve into intervals between the design points and interpolate the segments with piece-wise polynomials. To ensure continuity from segment to segment up to the required degree, the boundary conditions at the end of the preceding segment can be accepted as initial conditions for the next segment. This approach is known as the segmented polynomial method. An example showing the procedure of segmenting a path is seen in Fig.3.5 where the continuity of the motion curves is required up to the level of acceleration.

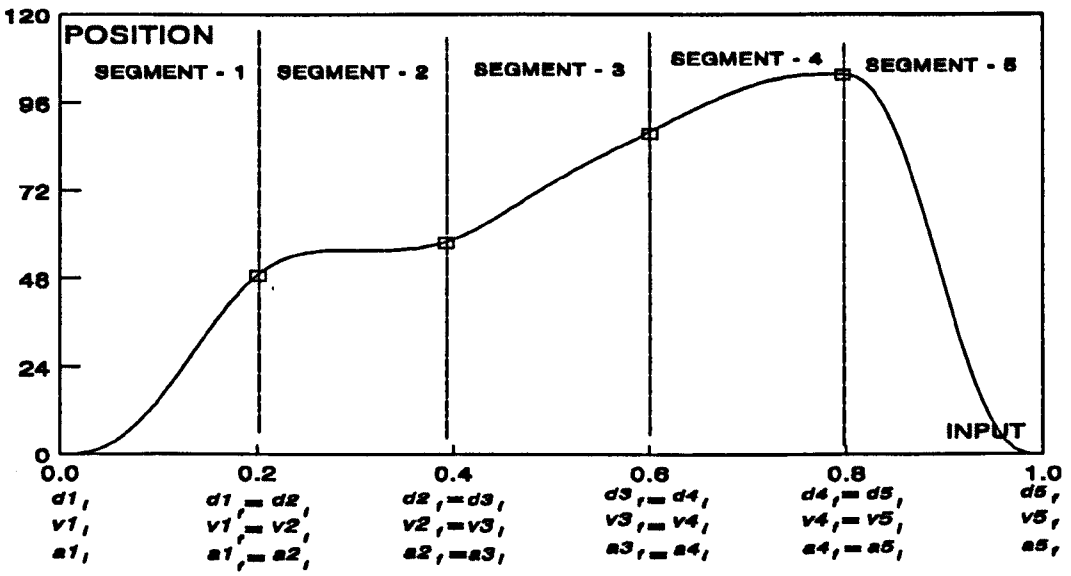


Fig.3.5. Procedure for segmented interpolation.

The segmented polynomial method is a powerful tool for motion design, it clearly prevents extreme oscillations occurring in the interpolated curves such as that shown in Fig.3.1. However in some situations the trajectory may still suffer from meandering due to severe velocity and acceleration constraints being present (see Fig.3.7) In these cases further division of segments is not possible because a segment must include at least two design points. Since polynomial functions give unique solutions there is nothing to be done when the boundary conditions at the design points are fixed. Another issue is that although the interpolation curve may not include any obvious oscillations it may lie outside a specified tolerance envelope.

To decide whether the interpolated curve is either acceptable or not it must be judged against any limit values which may be dictated by the requirements for the system. However there are some basic design limits that the trajectory should satisfy if it is be considered reasonable. These limits can be represented by rectangular shapes between successive design points as shown in Fig.3.6 where each rectangular shape is called the maximum tolerance envelope for a segment.

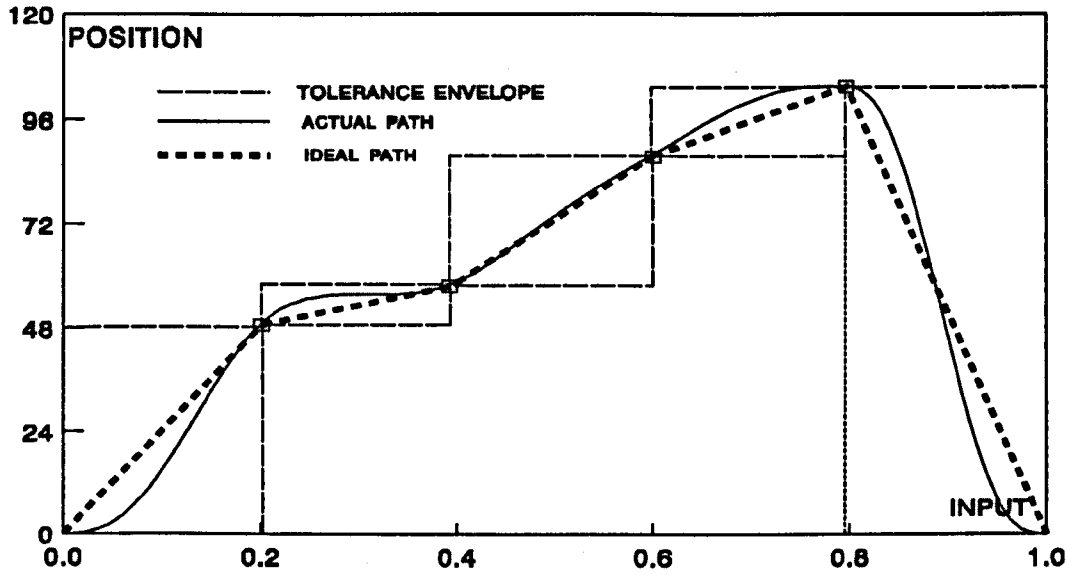
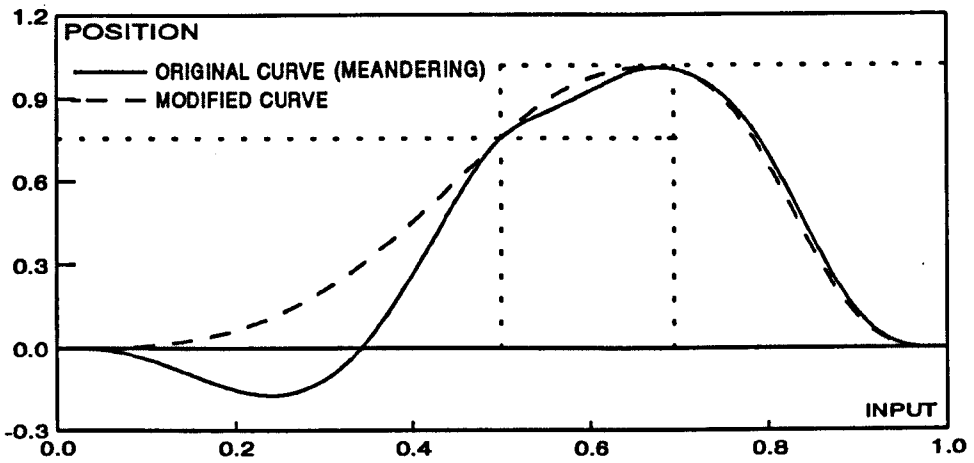


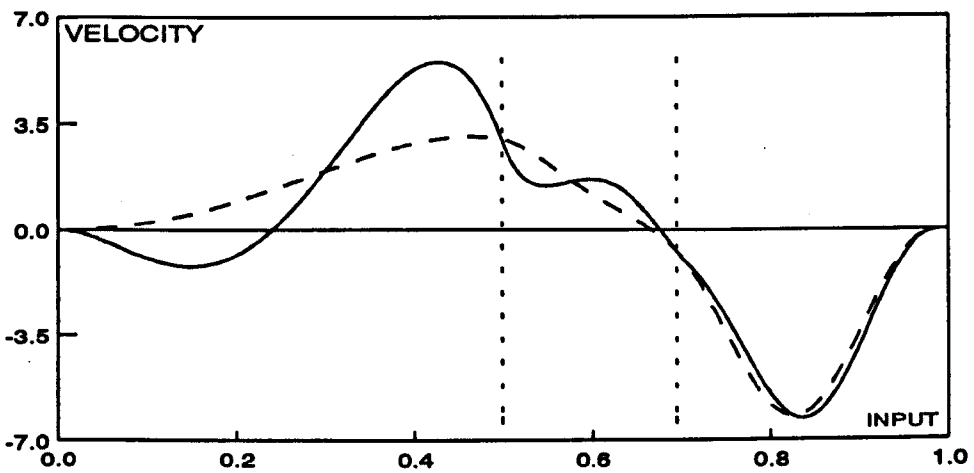
Fig.3.6. Determination of tolerance envelope.

The rectangles define the limits for meandering in each segment and violation of these limits may result in some unacceptable disturbing action by the system. For example, consider a trajectory for the slider of a slider-crank mechanism which crosses the tolerance envelope at the beginning of the trajectory. See Fig.3.7 (solid line) and 3.8. The mechanism starts its motion but curiously, because there is no constraint such as negative velocity or acceleration at the first design point (as seen in Table.3.2), it moves backwards for a distance and returns to the starting point then completes the cycle. A part of the cycle time is wasted for this unnecessary motion and since there is less time to achieve the actual motion the mechanism has to move faster to complete the loop in time.

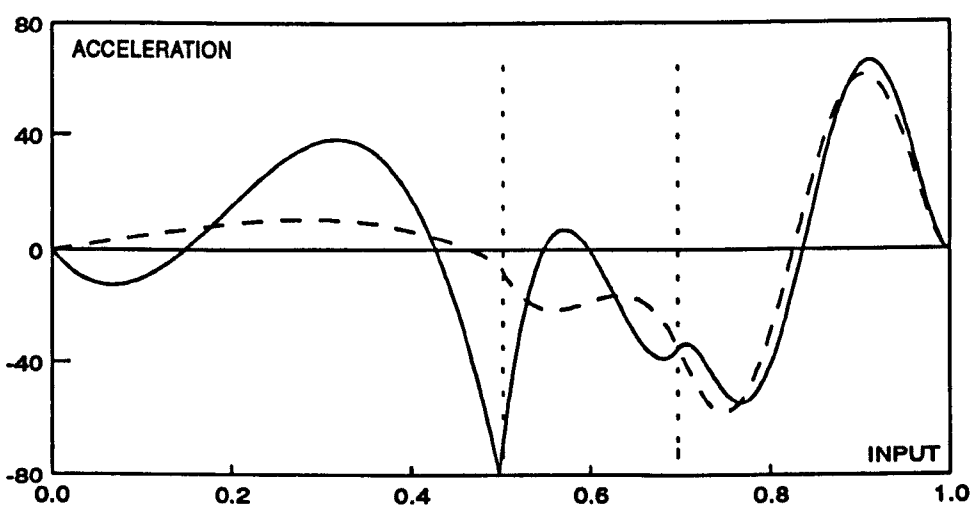
The cause of the meandering in this example is the poorly chosen acceleration constraint at the second design point. The expected trend of the velocity between these two design points is positive because an increasing positive displacement is required with the final velocity higher than the initial value. Allowing a large negative acceleration constraint at the second point forms a contradiction since it forces the velocity curve to adopt a negative slope at that point, see Fig.3.7-b (solid line).



a) position



b) velocity



c) acceleration

Fig.3.7 Motion curves of a slider crank mechanism with and without meandering

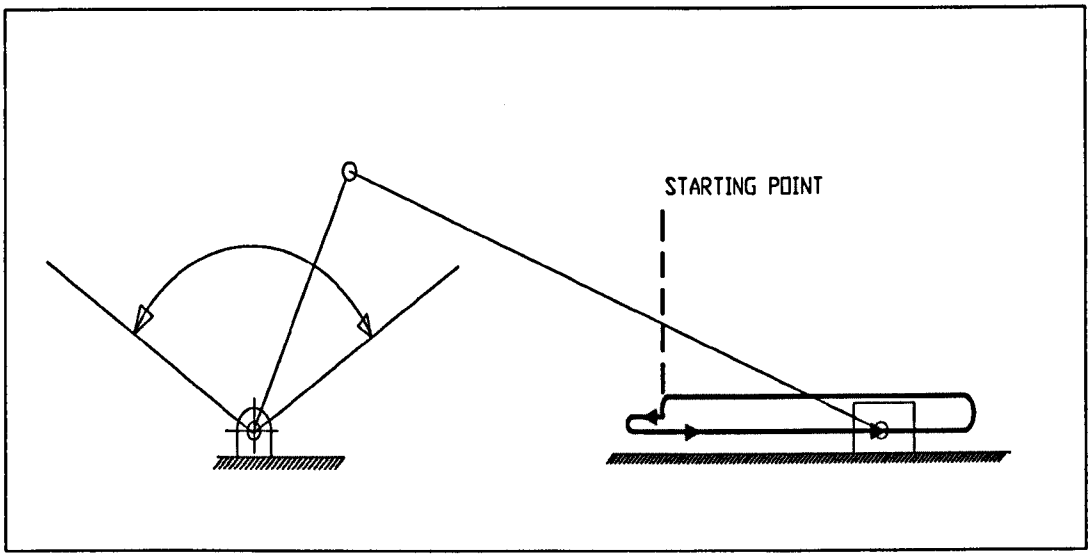


Fig.3.8. Meandering problem for a slider crank mechanism driven by a servo motor.

TABLE 3.2. Constraints for Fig.3.7

pos.	vel.	acc.	jerk	input
0.00	0.00	0.00	--	0.00
0.85	3.00	-75.0	--	0.50
1.00	0.00	-30	--	0.70
0.00	0.00	0.00	--	1.00

That means that the velocity curve has to reach a bigger value at an intermediate point in the segment than the velocity value defined at the end of the segment. Another curve is shown in the figure (dashed line) which is obtained when a zero acceleration constraint is imposed at the second design point. In this case the curve shows no undesirable meandering. Specifying extreme acceleration values sometimes creates a supplementary velocity area and this area must be balanced somewhere in the segment to satisfy the position boundary conditions. These excessive areas are indicated in Fig.3.7-b. This situation explains the reason for the negative displacement at the beginning of the segment. From this example, it may be concluded that the presence of oscillations in the curves obtained using the segmented method are dependent on the values of the boundary conditions especially those for velocity or acceleration.

3.6 Polynomials with arbitrary powers

In order to overcome the drawbacks of motion design using the segmented polynomial method, a new technique has been proposed. It allows the shape of a curve to be improved without changing any given boundary constraints. The method is not only appropriate for modifying poor trajectories which suffer from meandering but it can also be used to refine and improve normal curves. This technique is based on the fact that changing the powers of the polynomial function changes the path between the design points. In this application, the number of coefficients is equal to the number of boundary conditions as for ordinary polynomial functions, however, the powers can be arranged arbitrarily.

From equation (3.3) the power form of a polynomial function has the form:

$$d(t) = C_0 + C_1 t^1 + C_2 t^2 + \ldots + C_n t^n$$

It is clear that the powers of the function systematically increase for each term and it seems that these are the only parameters which can alter the characteristic shape of the curve while still satisfying the required boundary conditions. The form of a polynomial function with only two position constraints is:

$$d(t) = C_0 + C_1 t^1 \tag{3.7}$$

The shape of the curve for the above function is a straight line between the two end points. However, this shape can be altered without changing the boundary conditions. For example, increasing the power by one will turn the function into the form of

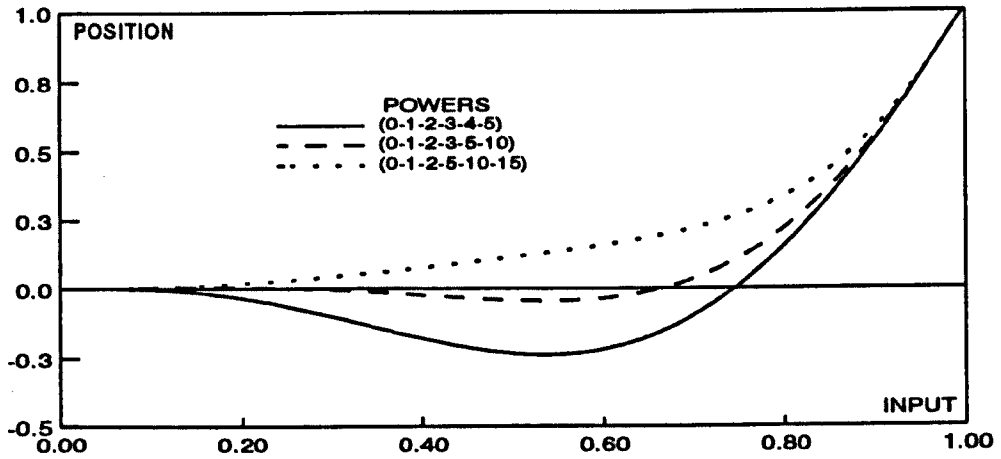
$$d(t) = C_0 + C_1 t^2 \tag{3.8}$$

Here the function is of second degree and it satisfies the constraints by following another path. Similarly third and fourth degree functions can be derived which will also satisfy the same boundary conditions. This means that polynomial functions can satisfy a set of boundary conditions with arbitrarily chosen powers. The availability of computers with high processing speeds makes it possible to scan rapidly many interpolations produced by using arbitrary powers and to select the most suitable function which gives the best curve. Some techniques to derive better polynomial interpolation curves are discussed below.

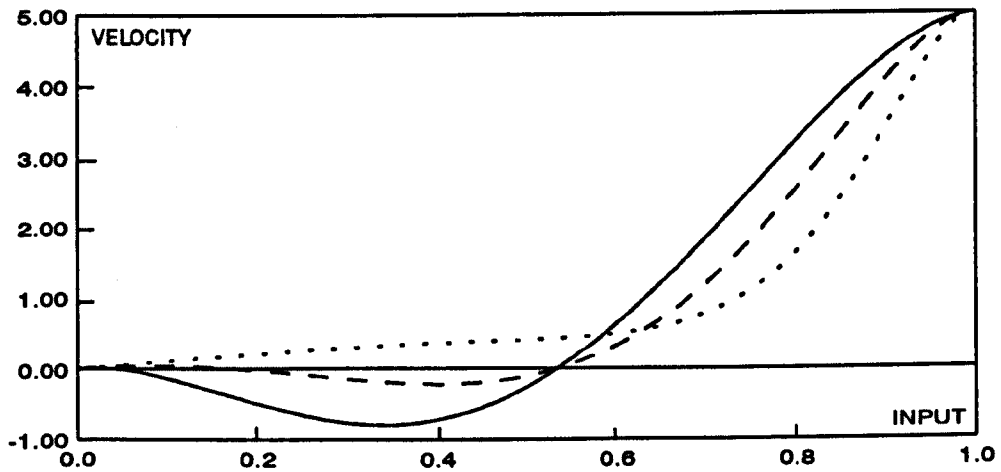
3.6.1 Raising the Powers

The effect on curve interpolation using polynomials with raised powers was investigated by Dudley [3.3] in the context of cam design for automotive applications. Integer powers only were considered. Similar work in cam design was later carried out by Stoddart [3.4]-[3.6], Tesar and Mathew [3.7].

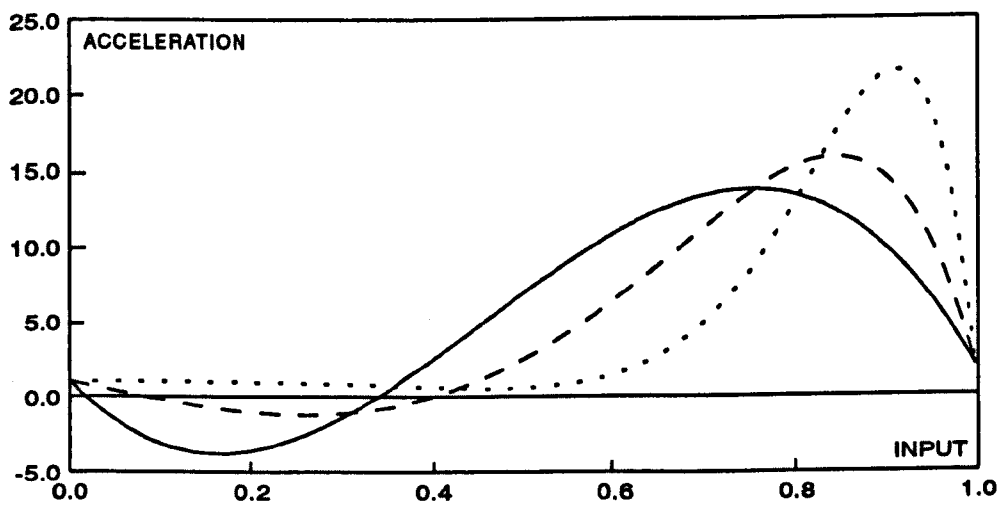
An examination of numerous curves generated using computer-aided analysis has shown that the effect of raising the powers of the polynomial produces a constant slope in the curve of the second derivative at the start of the segment for a period. The length of this period increases as the powers increase. An interpolation using a five-degree polynomial for the data in Table 3.3 is illustrated in Fig.3.9 where the curve finishes its trajectory by crossing the required tolerance zone. The second curve is obtained (dashed line) by increasing the powers of the last two terms in the function, it is better than the original curve but still meandering is present. However the trajectory is much improved when the power of the last three terms are increased. Observation of the curve for the acceleration shows that it remains flat for much of the interval while ending with a pronounced peak. The result achieved is perfectly acceptable if that peak is allowable by the designer.



a) position



b) velocity



c) acceleration

Fig.3.9 Polynomial interpolation with raised powers

TABLE 3.3. Constraints for Fig.3.9				
pos.	vel.	acc.	jerk	input
0.00	0.00	1.00	--	0.0
1.00	5.00	2.0	--	1.0

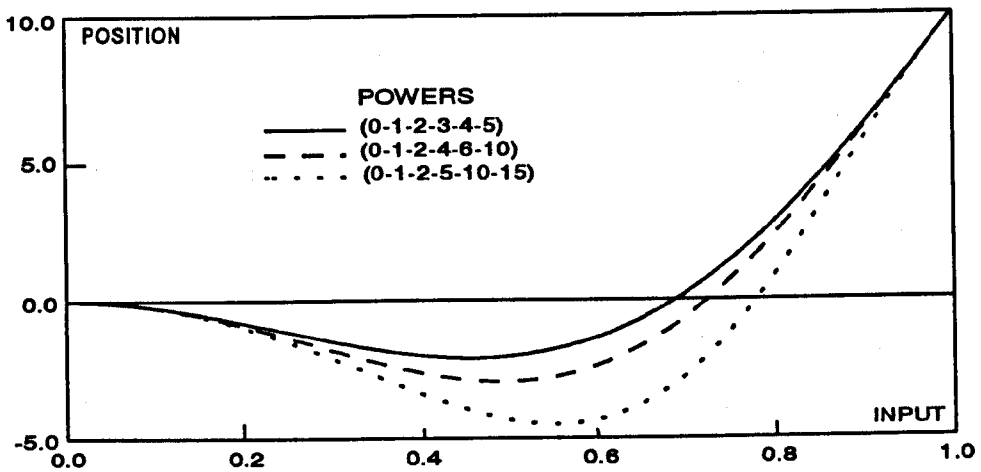
The improvements obtained in trajectories by means of raising the powers depend on the initial values of the first and second derivatives of the segment. As previously mentioned increasing the powers artificially produces an acceleration with an initial constant slope for a period. Because of this, there are some cases where the method can not improve the trajectory. In fact, it may worsen the degree of oscillation in the curve. For example, Fig.3.10 shows a segment which has a positive displacement with negative velocity and acceleration at the beginning of the segment. The data for this example is given in Table 3.4. Raising the power of the polynomial lengthens the period for which the acceleration remains negative. The result is to increase the amount of oscillation in the trajectory.

TABLE 3.4. Constraints for Fig.3.10				
pos.	vel.	acc.	jerk	input
0.00	0.00	-40.0	--	0.0
10.0	40.00	0.00	--	1.0

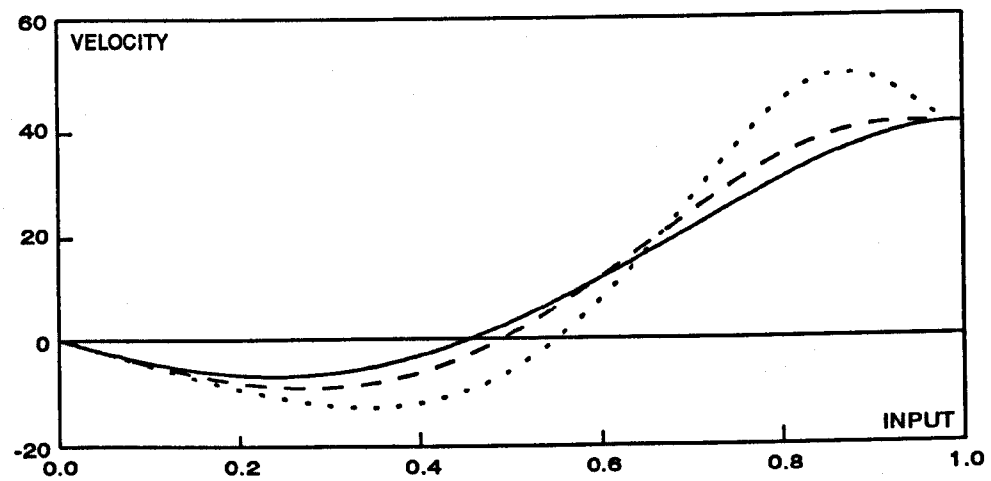
3.6.2 Reducing the powers

The effect on the interpolation when powers of the polynomial are reduced is also of considerable interest. Since the powers are reduced, their values must be real. The modified polynomial with the reduced powers may be written as

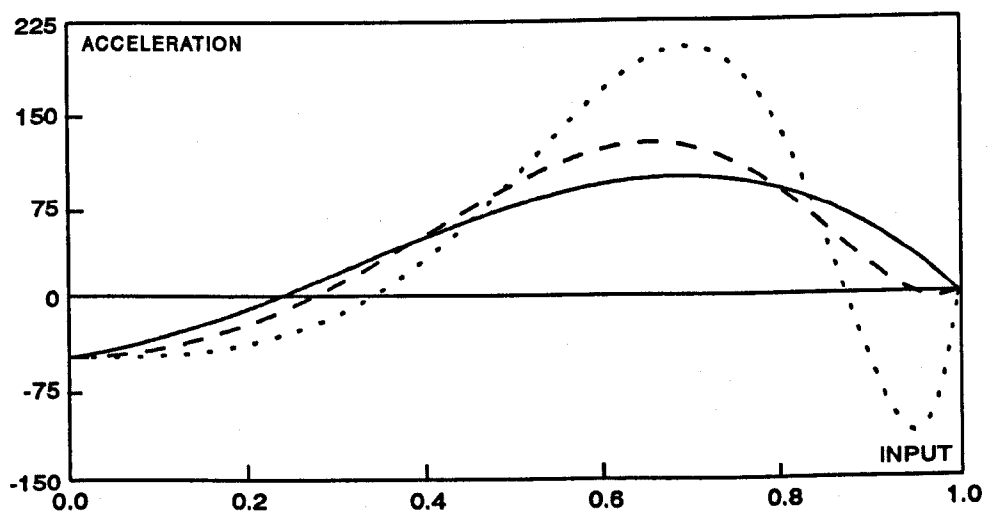
$$d(t) = \sum_{i=0}^{j} C_i t^i + \sum_{i=j+1}^{n} C_i t^{q_i} \qquad (3.9)$$



a) position



b) velocity



c) acceleration

Fig.3.10.Increasing the meandering due to improper selection of the powers

where the function includes two sub-functions. The first sub-function must satisfy the boundary conditions which are specified at the beginning of the segment. The powers in this function may not be changed in order to preserve the values for the function and its derivatives when the parameter t is equal to zero. Also in order to obtain a smooth interpolation curve through the segments the continuity from segment to segment must be assured. Reducing the powers of polynomial functions too much causes derivative curves which may include step changes at the segment joining points especially in the acceleration curves. Such an example is shown in Fig.3.11(dotted line). To meet these conditions, the minimum allowable power which may be obtained by reduction should satisfy

$$q > m + 0.5 \tag{3.10}$$

Where

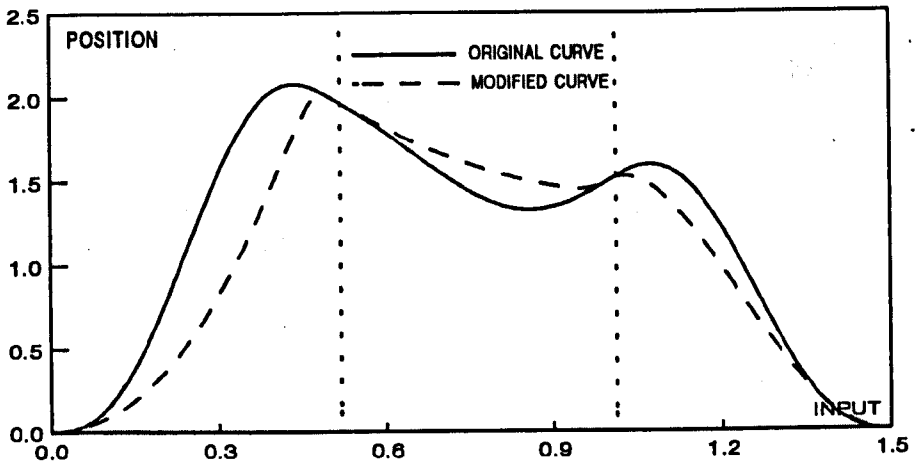
q = minimum allowable power

m = degree of derivative up to which continuity of the function is required between segments

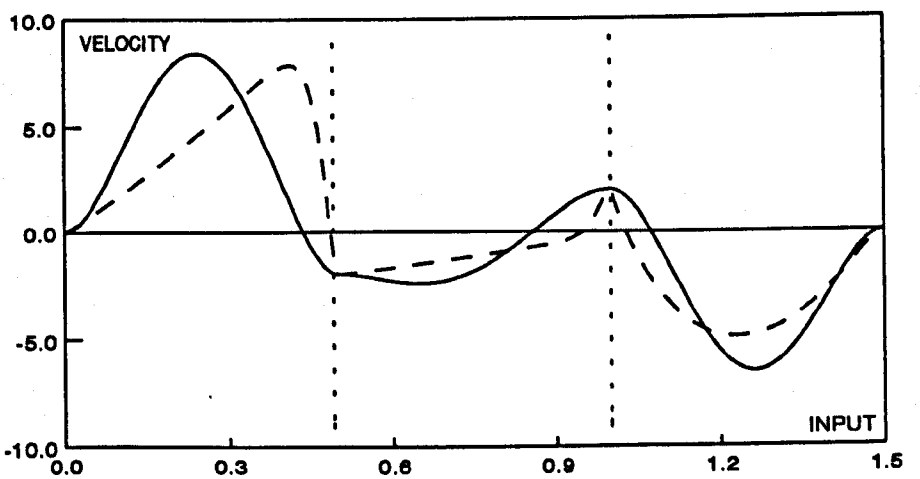
For example, if continuity is required up to the second derivative of the function, the minimum power will be q=2.5.

Although this condition is empirical, it is supported by successful results which have been obtained for many examples. More accurate results can be obtained by selecting the powers of the function interactively or using an optimization method.

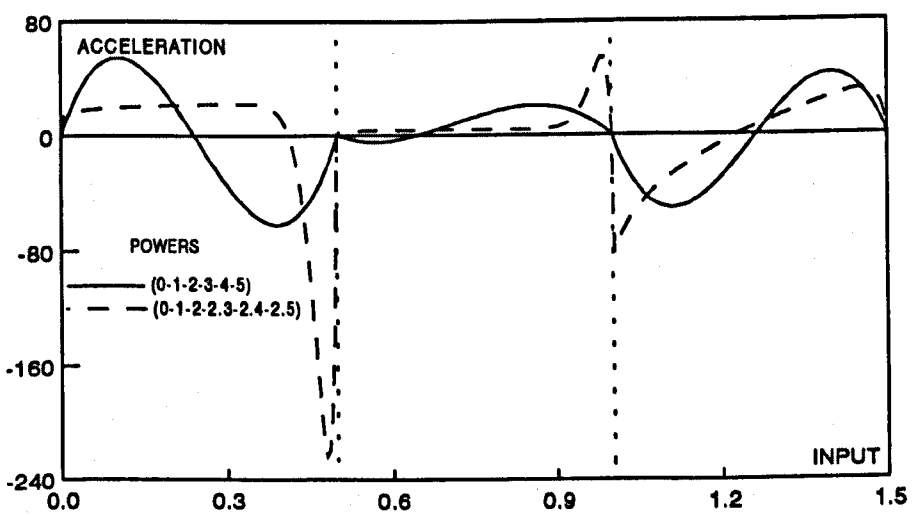
The effect of raising and reducing powers for the interpolation which satisfies the conditions listed in Table 3.6 is shown in Fig.3.12. Here the solid line is the original curve which goes out of the required tolerance zone. The first attempt to modify the curve is to increase the powers of the function but this approach has increased the amount of the oscillation. However, a much better curve is obtained when the powers are reduced (dotted line).



a) position

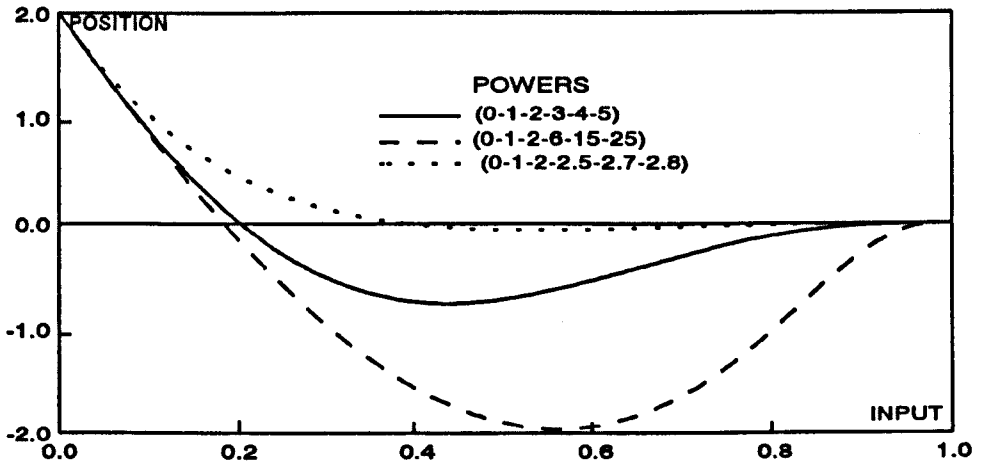


b) velocity

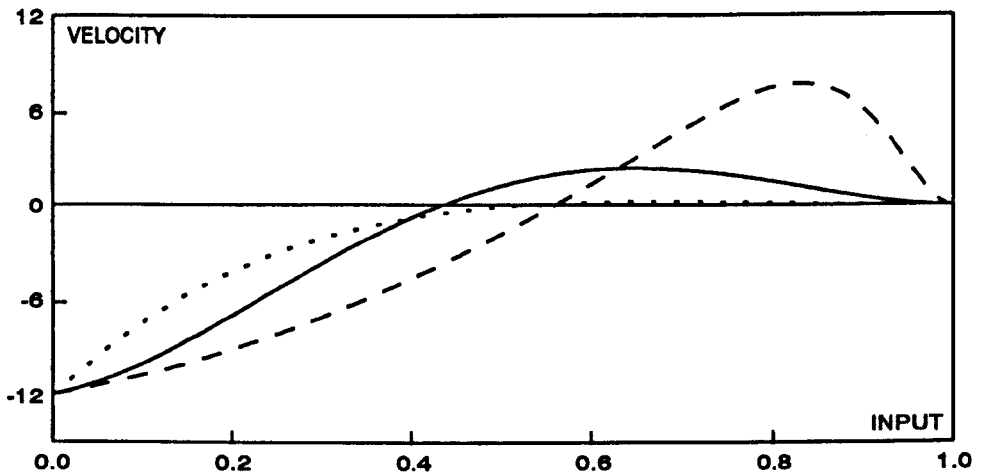


c) acceleration

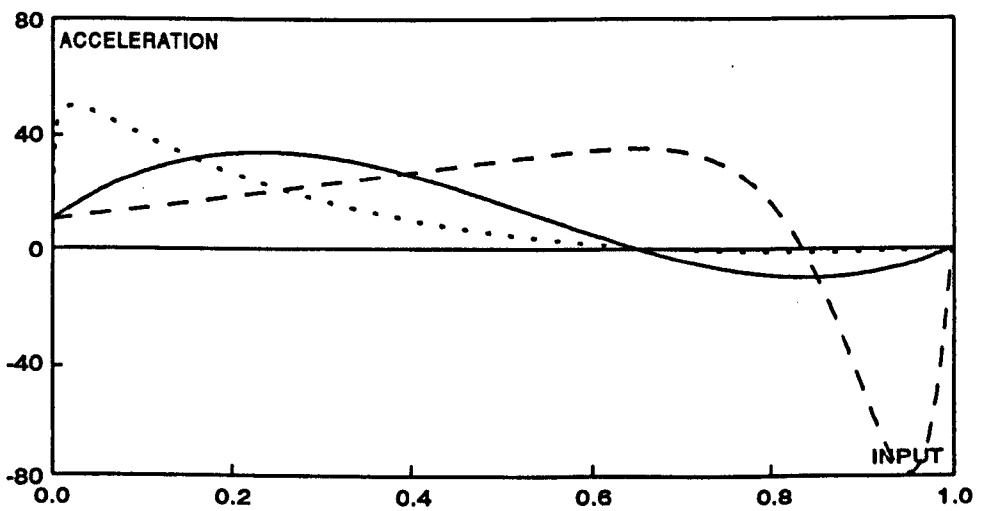
Fig.3.11 Discontinuous acceleration due to too low degree polynomial interpolation



a) position



b) velocity



c) acceleration

Fig.3.12 Motion curves of a segment with arbitrary power polynomial

It can be seen that raising the powers forces the main undulations towards the end of the segment in the acceleration curve. Reducing the powers is seen to reverse this effect, that is, force the main undulation to the start of the curve. The better interpolation curve is obtained using reduced powers in the above case.

TABLE 3.5. Constraints for Fig.3.11

pos.	vel.	acc.	jerk	input
0.00	0.00	0.00	--	0.0
2.00	-2.00	0.00	--	0.0
1.50	2.00	0.00	--	1.0
0.00	0.00	0.00	--	1.0

TABLE 3.6. Constraints for Fig.3.12

pos.	vel.	acc.	jerk	input
2.00	-12.00	10.00	--	0.0
0.00	0.00	0.00	--	1.0

3.6.3 Combination of Reduced and Raised Powers

In general it is found that oscillation in polynomials can be better controlled if the undulations in the derivative curves are pushed towards the two ends of the segment with the minimum amount of disturbance between. To achieve this result a combination of reduced and raised powers are used and the function is modified as shown below.

$$d(t) = \sum_{i=0}^{j} C_i t^i + \sum_{i=j+1}^{k} C_i t^{q_i} + \sum_{i=k+1}^{n} C_i t^{m_i} \qquad (3.11)$$

where

i= non-changeable powers

q_i= reduced powers

m_i= increased powers.

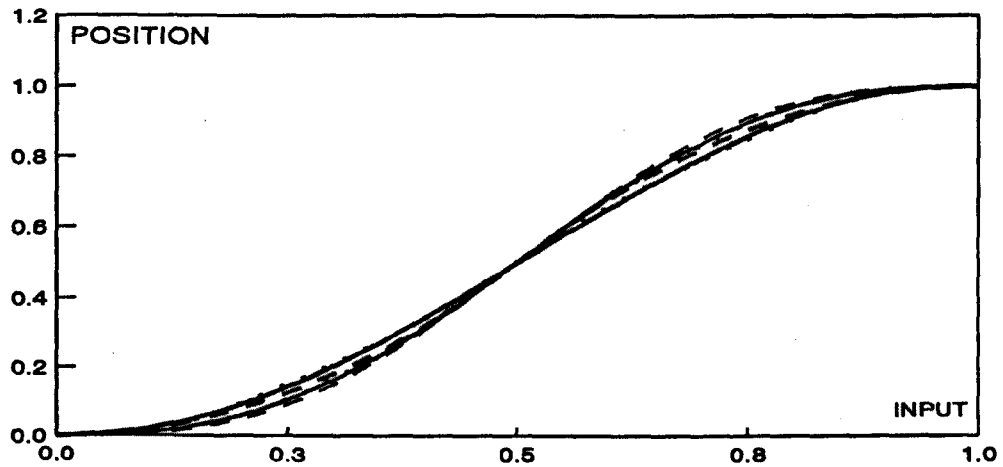
The peak undulation is moved towards the beginning of the derivative curve when reduced powers are used and towards the end with increased powers. Therefore optimum motion curves can be achieved by putting limits on the powers which may be changed. The allowable peaks in the velocity and acceleration curves can be used to set these limits. The procedure has been computerised and the interpolation achieved for a systematic selection of powers.

A comparison of some motion laws is presented in Fig.3.13. These curves show that arbitrary power polynomials functions can produce remarkably efficient motion curves. In this example motion laws are tested for a unit displacement with zero velocity and acceleration at the ends of the segment. If we compare the acceleration curves the arbitrary power polynomial law gives an average acceleration curve. Bang-bang and trapezoidal motion curves have slightly lower peak acceleration values. However, if we compare the velocity curves, the arbitrary power polynomial motion has the lowest peak velocity value. In this example the powers used for polynomial function are (0,1,2,2.52,2.54,39).

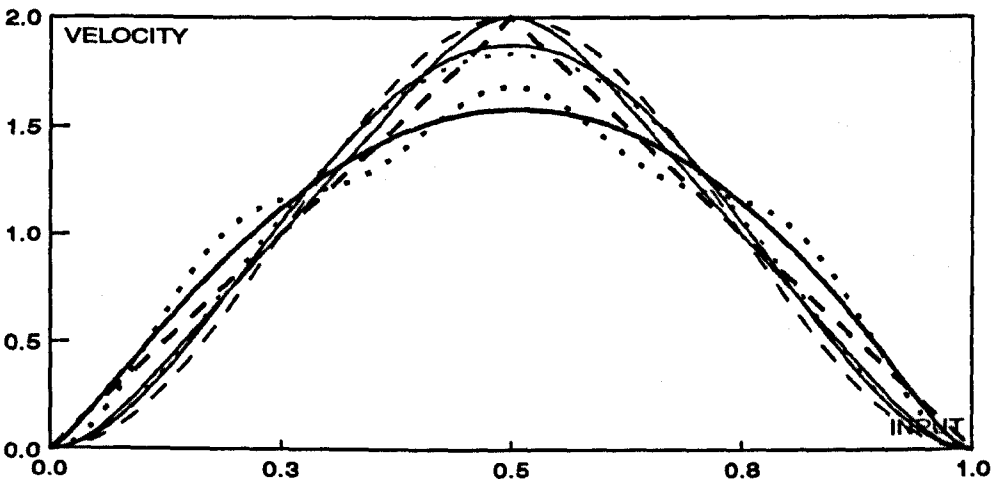
Fig.3.14 shows an example of polynomial interpolation for a four-segment curve. The values of the constraints are listed in Tables 3.7. The solid curve shows normal polynomial interpolation. Considerable improvements is seen when the polynomial interpolation is modified using arbitrary powers.

In the example all the segments are modified and the powers for these segments are:

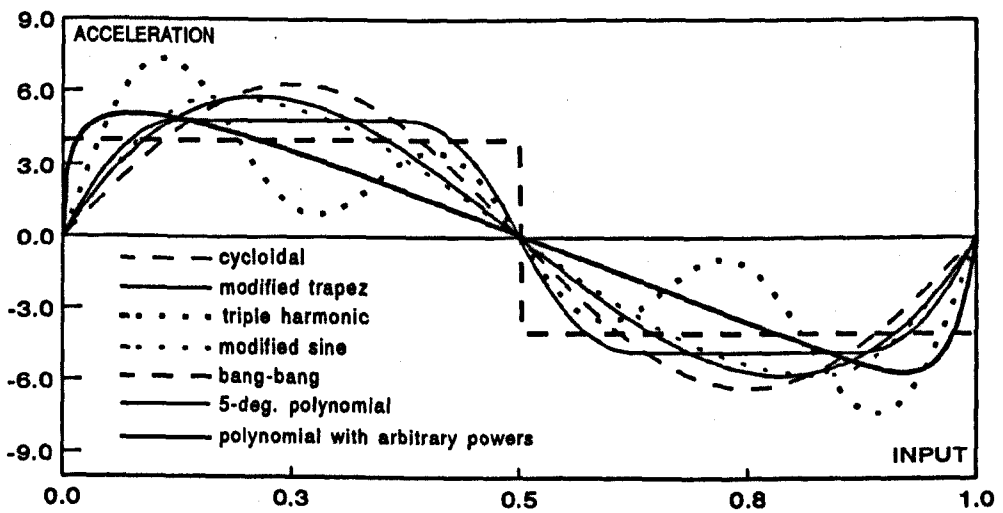
Segment-1=(0,1,2,2.52,2.56)
Segment-2=(0,1,2,2.56,2.72)
Segment-3=(0,1,2,2.52,2.56,26).



a) position

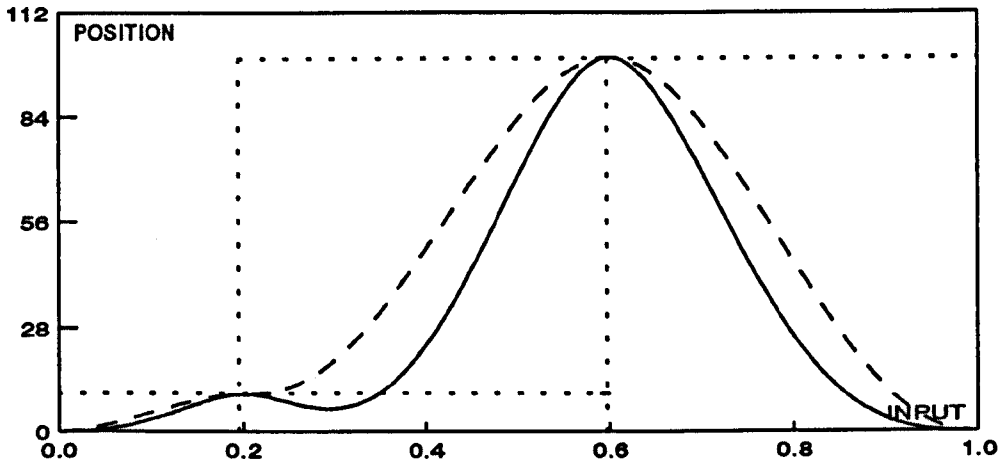


b) velocity

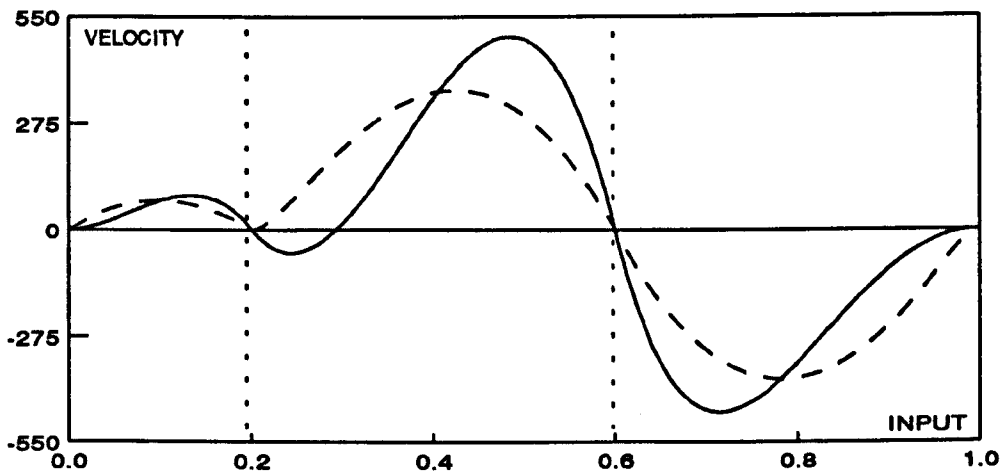


c) acceleration

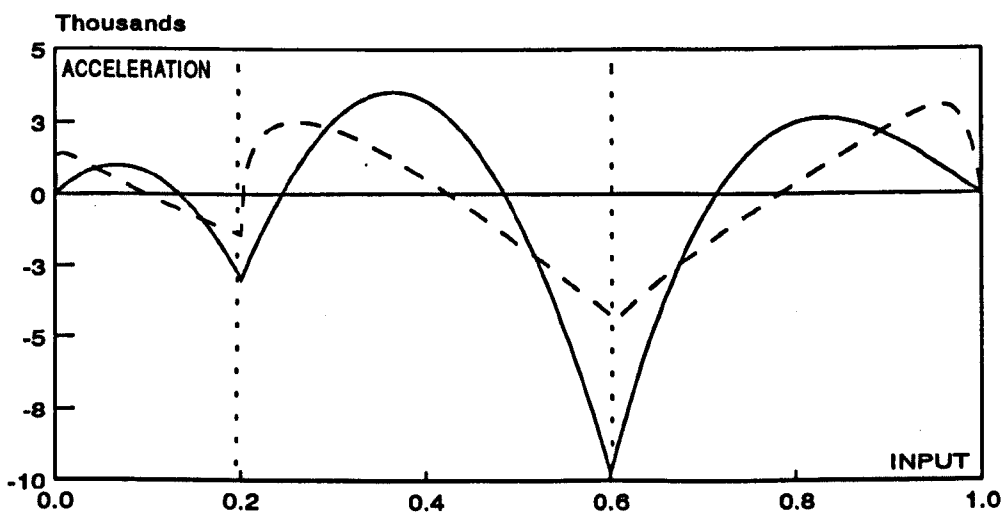
Fig.3.13.Comparison of motion laws



a) position



b) velocity



c) acceleration

Fig.3.14 Curves of trajectory by arbitrary degree polynomial interpolation

TABLE 3.7. Constraints for Fig.3.14				
pos.	vel.	acc.	jerk	input
0.0	0.0	0.0	--	0.0
10.0	0.0	no/match	--	0.2
100.0	0.0	no/match	--	0.6
0.0	0.0	0.0	--	1.0

3.7 Motion law selection for high-speed motion generation

The aim is to develop a general trajectory model which can be applicable to any kind of motion design problem including X and Y axes. Therefore the principal features of the mathematical form should be:

(i) The mathematical form must be an explicit function of time. This is because in many cases the velocity and acceleration curves are also needed to be defined specifically. These curves can be obtained by differentiating the function with respect to time.

(ii) The function must be continuous up to second order (acceleration). Any discontinuity in the acceleration curve provokes an infinite value for the jerk (third derivative) at the same point. Such conditions provide a source of disturbances for a system, especially when operating at high cyclic frequencies. Vibration, noise and shock are the main results.

(iii) Specification of arbitrary boundary conditions for derivatives of the function should be both possible and simple in order to achieve the required motion curves. One important complication is that the motion of a mechanism may depend on the motion of other mechanisms which operate together for the processing of a product. Therefore there must be synchronization between the motion of these machines which requires specification of position, velocity and acceleration boundary conditions. However, many of the mathematical functions such as Fourier series, spline functions, logarithmic and exponential functions are not able to satisfy arbitrary required boundary conditions.

(iv) The solution of the function coefficients should be simple.

(v) The behaviour of the interpolation function must be predictable. Many of these can give curves which pass through the boundary points and may be continuous up to acceleration, however, most interpolation functions can produce extremely large oscillations between the design points depending on the required shape of the trajectory.

The exponential, logarithmic and hyperbolic functions have important drawbacks when used to produce motion curves. Although these functions may precisely satisfy position boundary conditions, the control of velocity or acceleration curves is not possible with the result that they usually produce discontinuous velocity and acceleration curves between the segments. (Seen in Fig.2.2.)

The harmonic analysis method sometimes produces quite efficient interpolation curves, however, the method has some important drawbacks from the point of view of the motion designer. These include:

(i) Specification of boundary conditions is possible only for the function or one of its derivatives. This means that manipulation of all motion curves together (displacement, velocity, and acceleration) is not possible.

(ii) It may be necessary to define many boundary conditions to achieve a good interpolation curve for some difficult motion profiles.

(iii) The calculations are straight forward but require longer time then the other methods.

(iv) The formula in harmonic analysis is in terms of sine and cosine functions. Since a large number of sines and cosines are used, and since the arguments of these sines and cosines may have inherent errors, the total error may build up considerably when many boundary conditions are included.

Rational functions are very effective mathematical forms for interpolation of an arbitrary data set. Unfortunately, the function is not appropriate for differentiation. Another important disadvantages is that the function may produce a discontinuity in the trajectory as discussed before (see section 2.9). Because of these difficulties rational functions are not recommended for motion design.

Principally, quintic splines and segmented polynomials give similar solutions. Because, in many cases, controlling velocity, acceleration as well as position is necessary to obtain a specific trajectory therefore quintic splines produce continuous curves up to the acceleration under these circumstances as does the five-degree segmented-polynomial method.

Cubic spline functions are used traditionally to design curves. They mostly produce satisfactory curves, but, the setting of values for the derivatives of the function is not allowed. However they can be used for some cases where only position constraints are important.

In practice, there is no particular interpolation method which can be applied to motion design problems without the need for any modification. In other words no mathematical function can satisfy all the required conditions. The use of polynomials however offers advantages over other mathematical functions. They can, in principle, satisfy all the requirements of motion design, but they can suffer from the important drawback for difficult trajectories known as meandering. However it has been shown that this phenomena can be prevented by means of some novel modification methods.

The selected motion laws which have been considered as suitable forms for general high-speed motion design software are listed below.

1) Polynomial functions

Despite many forms of polynomial function the power form is found to be the most suitable form for motion design. The disadvantages of this function can be modified by following methods:

(i) segmented method

(ii) using arbitrary powers

2) Cubic spline function

3) Standard cam motions

(i) cycloidal motion

(ii) modified sine motion

(iii) modified trapezoidal motion

(iv) triple harmonic motion

(v) dwell motion

(vi) constant speed motion

(vii) constant acceleration motion.

CHAPTER 4
TRAJECTORY PLANNING

4.1 Introduction

The activity of converting the description of a desired motion to a trajectory by defining time sequences of configurations of the end-effector of a manipulator between start and final positions, feasibility checks and adaptation of the motion before implementation is referred to as motion design or trajectory planning. As the trajectory is executed, the tip of the end-effector traces a curve and changes its orientation. The net effect is a translation and a rotation of the end-effector. The curve traced by the end-effector is called a **path**, whereas a **trajectory** is a path with time constraints, that is, it includes velocity and acceleration as well as position and orientation along the path.

For many industrial applications, present computer controlled manipulators are too slow to justify their use economically because of their improper trajectories [4.1],[4.2]. Their speed and hence their productivity are limited by the performance capabilities of their actuators. Increasing actuator size and power is not the best solution because of the increased cost and power consumption. A more successful approach is to design the trajectory at an advanced level in order to increase the speed of the system and perform a given task appropriately.

Computer-controlled manipulator schemes can be divided into two levels in order to simplify the problem. The upper level is trajectory planning and the lower level is path tracking or path control [4.3]-[4.5]. There are different approaches for trajectory planning some of which are given below, however, trajectory tracing is presented in Chapter-7.

Traditionally, the trajectories of robotic manipulators are planned manually [4.6]. These paths are normally composed of simple elements, such as straight lines and circular arcs. Typically, the path is first constructed with a number of straight line segments and then the corner of each segment is smoothed with a circular arc. Such planning generally fails to select the best trajectory since no account is taken of the effect of the highly non-linear characteristics of the motion of the manipulators.

Another common method is to design a trajectory with constant acceleration and constant deceleration segments often known as the bang-bang method [1.5] and [4.7]. The values of these accelerations and decelerations are generally selected to avoid actuator saturation during the motion. However, designing a trajectory using constant speed and constant acceleration/deceleration produces discontinuous acceleration and results in infinite jerk values at the connection points of the segments. Such conditions produce large disturbances for a system especially at high speeds.

Another general method is to plan a trajectory using spline functions. The main advantages and disadvantages of this method have been discussed in section 2.10.1.

A completely different method is to design the motion by means of computer graphics. This approach would allow the user to plan manipulator strategies while being able to watch the motion of the manipulator and its surroundings in a graphical environment. Optimization of the trajectory by invoking different options in the menu and accessing a data base provides an extension of such planning. An application of this method might be the example shown in Fig.4.1 where the requirement is to plan the trajectory of the end-effector

of a robotic manipulator in co-ordination with the motion of another machine. This is a typical example of a "pick and place" mechanism. Here, the manipulator loads empty containers to the filling machine by following the designed trajectory.

In this chapter, some basic concepts relating to computer controlled systems are introduced.

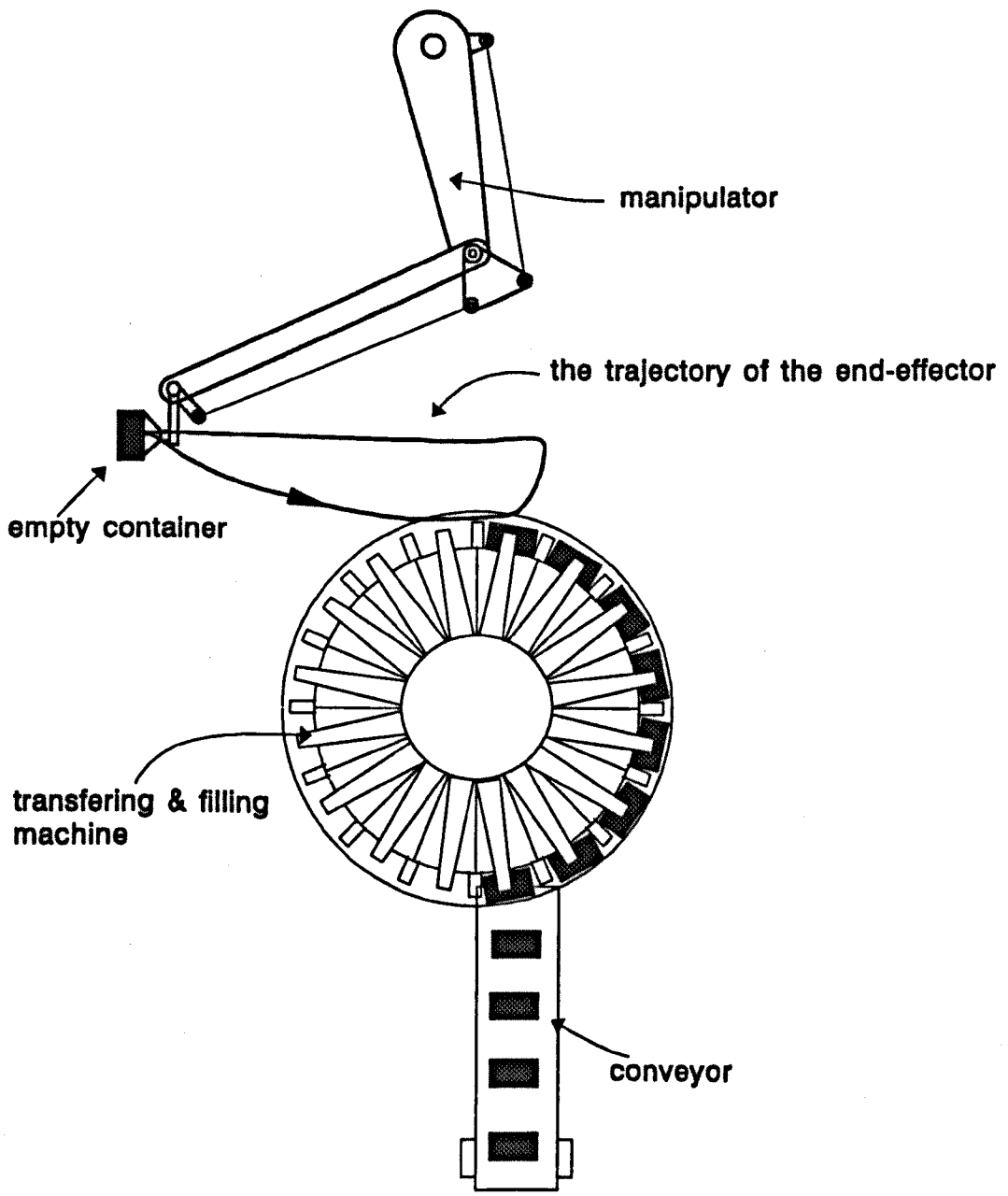


Fig.4.1.simulation of a robotic arm and its environment on computer.

4.2 Basis of manipulator motion planning and control

A basic problem in a programmable system is planning motions to solve some previously specified task, and then controlling the system as it executes the commands necessary to achieve these motions.

The path of a manipulator for repetitive tasks can be split into two distinct sections according to the importance of an activity.

(i) Process section: the process section is the part of the path in which execution of a process takes place such as cutting, welding, painting etc. However in some cases a manipulator may perform multiple tasks. For example, consider a path on which the end-effector performs the following processes for a product; grasping, insertion, bending and welding operations. Each task corresponds to a part of the path. And each part can be described by a start position, final position and corresponding orientations of the end-effector. Additionally, the velocity and acceleration values during the grasping, and insertion phase, the welding and bending speeds, and finally the time intervals for each operation comprise the main constraints to be imposed on the system.

(ii) Move and return section: The constraints for the processing sections may be exclusive, that means they may not be allowed to change. However, the trajectory may include some segments in which the manipulator moves from one point to another without doing any process. Each one of these usually corresponds to a segment which lies between two independent process segments. Also, there is another non-process segment (which is necessary) when the end-effector is returned from the final point in the processing chain to the initial point in order to start a new cycle. These segments provide some periods of the cycle which are largely unconstrained and therefore offer the most suitable sections for design in order to achieve optimum trajectories. The user is therefore normally free to specify arbitrary boundary conditions for these segments to fulfil the task.

Usually, the information at the start and end points of a process is sufficient to construct the segments of a trajectory. However, sometimes it is necessary to specify the motion in much more detail than simply giving the starting and final configurations for a segment. There may be a need, for example, to prevent a possible collision or to obtain a specific shape for a path. One way to include more detail in path description is to give a sequence of intermediate points between the initial and final positions. Although referred to as "points" they may include specifications for positions, orientations and further information such as velocity and acceleration. These intermediate points together with the initial and final points of a segment will be called **design points.**

Manipulator motions are generally partitioned into two categories: free motion and compliant motion [4.8]. For a trajectory planner, it is a relatively simple matter to construct a system which will move the manipulator to any position during a required time interval within the range of the manipulator. Any manipulator task which can be adequately expressed as a sequence of positions could be completed using this positioning system. However, the control of compliant motion refers to the manipulator motion in the presence of end-effector contact with external surfaces. In this case the motion cannot be adequately expressed by a sequence of positions. The contact of the end-effector with external surfaces necessitates another mode of control, one which takes into account the force acting on the manipulator. Numerous manipulator tasks involve compliant motion. Examples include placing an object on a table, opening a door, grasping an object, driving a screw, inspecting a surface by feeling it, installing a fuse or light bulb.

All compliant motion tasks are easily performed by humans, but they are generally very difficult for robot manipulators. This deficiency is one of the factors currently limiting the application of manipulators. Because of this, the most common applications of manipulators today are for pick-and-place operations, spot welding and spray painting processes which do not require sophisticated compliant motion capabilities.

There are two different approaches for trajectory planning: off-line and on-line planning. Off-line planning is carried out outside the manipulator environment and makes use of extensive software usually including computer graphics. The development of trajectories take place without access to the manipulator itself and without recourse to real time operation.

Off-line planning offers some potential benefits. A simulation of the manipulator can be displayed in the graphic environment, and the whole of the planning can be carried out within the computer environment. Thus, many of the problems peculiar to the manipulator programming tend to diminish. Problems such as, for example, unexpected oscillations in the trajectory (meandering) or excessive torque requirements in the planned trajectory which may cause saturation of the actuators.

Normally, off-line planning systems serve explicitly, which means that every action taken in the system must be performed by the user. However, explicit planning can be extended to high-level planning the so-called task level planning. In this system, the user may simply state goals such as "transfer the products", "insert the bolt" and even "assemble the machine". This extension is accomplished by providing automated solutions to various sub-tasks as these solutions become available, and letting the programmer use them to explore various options in the simulated environment [4.9].

On-line trajectory planning refers to the determination of the history of a motion by means of on-board sensory equipment and then the generation and execution of the motion in real time. That means, for a specific task, the constraints for the trajectory including constraints for obstacles along the path and constraints for co-ordination with other machines will be determined be means of sensory equipment; the optimum trajectory will be generated and this trajectory will be executed in a real time. Consider a robotic manipulator which assembles a machine in an assembly batch. Such an intelligent machine would have an outstanding impact in industry. Such planning requires highly sensitive sensory equipment, high-speed computers

and artificial intelligence capabilities. Although all these can be supported with today's technology, nevertheless, such planning cannot be easily undertaken in practice because of the inherent dynamic complexities associated with its implementation [4.10], high cost, limited application area and difficulties with the trajectory tracing.

However, some studies on on-line trajectory planning have been done by [4.5] and [4.11]

4.3 Kinematics and dynamics of manipulators

An understanding of the kinematics, statics and dynamics of a manipulator is essential for successful planning and control of a system. Kinematics is the relationship between position, velocity and acceleration among the links of a manipulator without regard to the forces influencing the motion. Describing a path in Cartesian or joint space and producing motion curves by means of interpolation functions lies in the area of kinematics. Also, transformation of some parameters from joint co-ordinate systems to Cartesian co-ordinate systems is known as direct kinematic analysis whereas transformation in the opposite direction is known as inverse kinematic analysis. Both transformations require a knowledge of the relationship between the manipulator geometry and that of its environment.

Statics concerns the relationships of the forces and torques acting between the manipulator and its environment and also between the links. In statics, an equilibrium analysis of the manipulator system in a rest configuration is performed by equating all external forces and torques acting on the links to zero. External forces and torques arise from gravity and from environmental interactions usually through the end-effector. Static analysis is particularly important in the study of compliant motion. The laws of statics can be appropriately generalized to situations where the manipulator is not at rest by means of Newton's Second Law, which may be used to reduce a dynamic state to a static one through the formulation of the force of inertia. This

principle states that the acceleration of a body generates a force of inertia which can be considered together with other forces acting on a body. Thus we may use equilibrium analysis even though the system is accelerating.

In order to move a manipulator along a trajectory, torques must be provided by motors at the joints. The problem of computing the values of the torques to apply to the joints to achieve a desired trajectory is the problem of dynamic analysis.

4.4 Selection of the co-ordinate system

Currently, two different co-ordinate systems have been used for trajectory planning. Joint co-ordinate system planning refers to planning a trajectory at the joint level in terms of each actuator's θ, $\dot{\theta}$ and $\ddot{\theta}$ values. This co-ordinate system has been used by [4.7] and [4.12]-[4.14]. However, Cartesian co-ordinate system planning refers to trajectory planning at the end-effector level in terms of the end-effector's position, velocity and acceleration this system has been used by [1.1], [1.4], [4.4] and [4.15].

4.4.1 Joint Co-ordinate system planning

This is a method of trajectory generation in which the motion curves are expressed in terms of functions of the joint angles. Usually a path is described using Cartesian co-ordinates to specify the desired positions and orientation. These co-ordinates can then be converted into the joint co-ordinate system by means of inverse kinematics provided the necessary transformations are known. Then, a smooth function may be found for each joint. The time required for each segment should be the same for each joint so that all joints will reach the desired design point at the same time, thus resulting in the Cartesian position of each design point being attained. Other than specifying the same duration for each joint, determination of the desired joint angle function for a particular joint does not depend on the functions for the other joints.

The main advantage in using joint co-ordinates systems is the ability to relate these co-ordinates directly to the actuator requirements. Although both systems require transformation of some information from one system to another, the transformation from joint co-ordinates to Cartesian co-ordinates is easy and straight forward whereas transformation in the reverse direction is more difficult. Moreover, using Cartesian co-ordinate systems may be quite expensive in respect of computational effort if the trajectory planning is required in real time. This requires the trajectory to be converted from Cartesian co-ordinates into joints angles at the run time. Therefore, trajectory planning in joint co-ordinate systems is more desirable for on-line trajectory planners because of the above difficulties when using Cartesian co-ordinate systems.

4.4.2 Cartesian co-ordinate system planning

Path shapes may be described within a Cartesian co-ordinate system in terms of functions which compute the Cartesian motion of the end-effector as functions of time. The trajectory can be planned directly from the user's definition without regard to the geometry of the manipulator and therefore no need for inverse kinematics. With this system it is relatively easy to examine the paths to be designed.

Working within a Cartesian co-ordinate system seems more attractive for off-line trajectory planning, because, the path is more conveniently described in Cartesian co-ordinates and analysing the motion of the end-effector is easy. Since all planning takes place without reference to the manipulator, the user can plan the trajectory in much more detail. It is possible to try different options to improve the trajectory and graphic simulation can be used to help judge such improvements.

In this study the Cartesian co-ordinate system is selected as the most appropriate co-ordinate system. This is because a high-level trajectory design is required and this needs the interaction of the designer with the computer to take some decisions and make suitable modifications.

4.5 Path description

It may be assumed that once we have defined a path from a start position to a destination, all the manipulator has to do is follow it. However, this will ignore the many constraints which have to be considered before a path becomes a trajectory.

Controlling the trajectories of high-speed manipulators requires a knowledge of the complete system dynamics. Admissible velocities and accelerations/decelerations along a prescribed path depend on all the forces acting along that path as well as on the constraints given by path geometry and by restrictions of joint torques or forces [4.16].

Thus, a trajectory planning requires an initial planning stage to determine these constraints. The constraints can be classified into sub-groups as follows:

(i) kinematic constraints
(ii) constraints on the actuators of the system
(iii) constraints on the product.

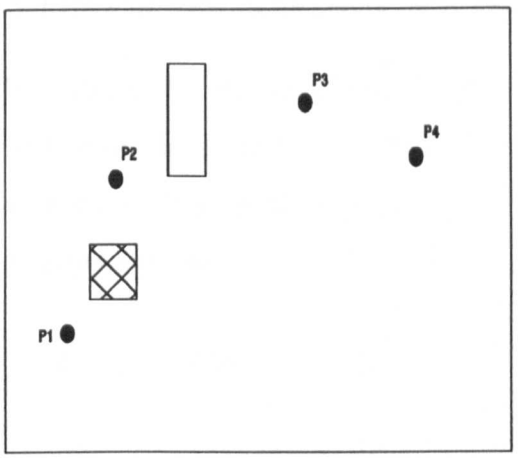
Knowing the constraints, they can be converted into suitable parameters to be used as inputs for the motion design. They may comprise values for time, position, velocity and acceleration.

4.5.1 Determination of kinematic constraints

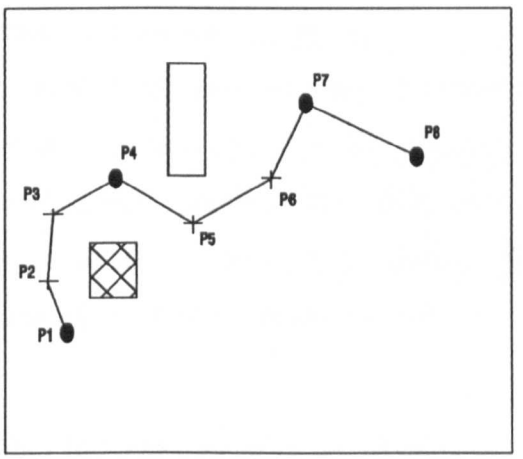
The path can be simplified to facilitate the description of the motion of the end-effector. For example, the designer may simply specify a number of important locations for the processing segments ($P1, P2, P3, P4$) together with

any obstacles which lie in the plane (Fig.4.2(a)). This is the main part of the path as far as the processes are concerned. In addition, there may be some intermediate points between these which may be specified to prevent a possible collision between the manipulator and other elements while the system is under operation. In this case, the path has to pass through these intermediate points as well. In Fig.4.2-(b) each straight line represents a segment of the path. For cyclical motion the end-effector has to move to the initial point again so the path can be formed as a closed loop as shown in Fig.4.2-(c). Now the designer can determine the time values for these segments (see Fig.4.2-(d)). The time values for processing segments are usually dependent on external factors such as for example, the speed of any co-operating machine, or the limits on the processing speeds or velocities or acceleration values which affect the product to be transferred. The time values for move and return segments are usually dependent on the capability of the manipulator. If the cycle time is fixed then the user can easily calculate these values. But, if the aim is to achieve a time-optimum trajectory (to drive the system at its highest speed) the time values for these segments can be calculated by comparing the actuators acceleration capacities with the generated acceleration curve in the computer environment.

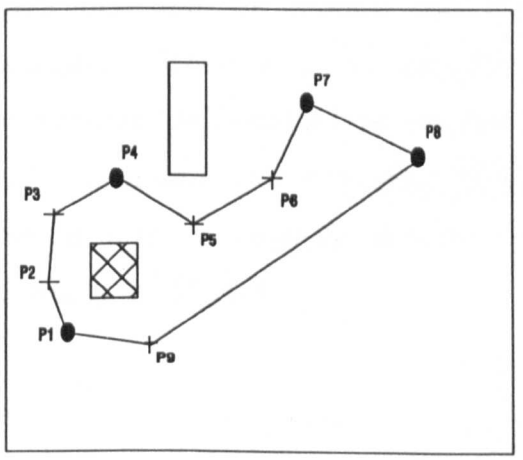
The velocity and acceleration values are important for all design points through the path. They must be continuous at the intermediate points of segments to obtain a smooth motion. However, they may be particularly important at the start and end points of each segment if the motion of the end-effector is required to match the motion of another mechanism or it is required to pick up or place a product at these points. Therefore their velocity and acceleration values must be specifically defined to achieve a co-ordination between the machines and also to ensure the safety of the products as well as machines. These values may be shown on the path with their directions. See Fig.4.2(e-f).



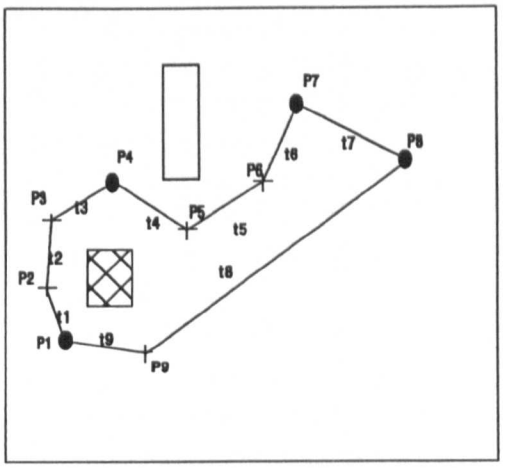
(a) Define positions and obstacles.



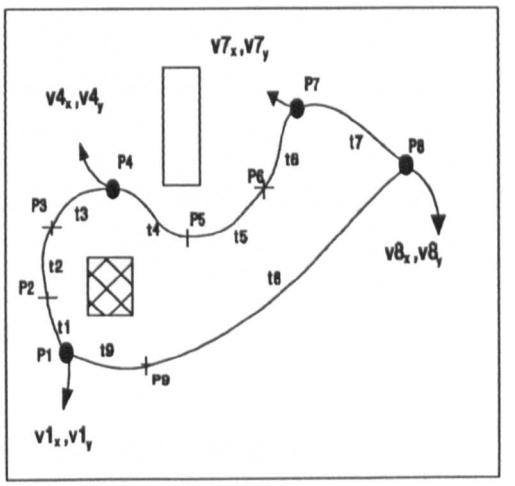
(b) Determine intermediate points.



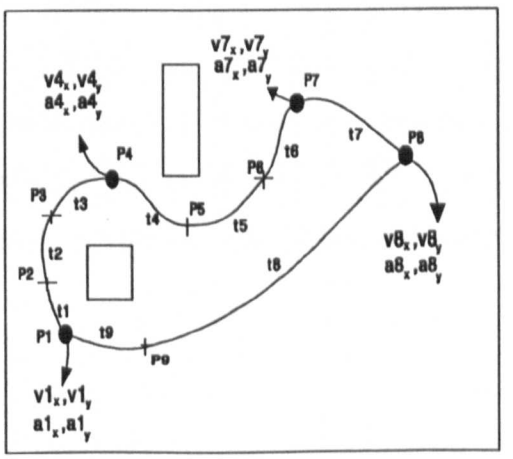
(c) Complete the cycle by return segment



(d) Determine time values for segments



(e) Specify velocities



(f) Specify accelerations

Fig.4.2 Description of a trajectory on Cartesian co-ordinate system.

4.5.2 Determination of system actuator capacities

The motion of the end-effector over certain incremental distances within a given time determines the required torque from the motors as a function of the motion. The relationship is given by Newton's Second Law which can be expressed as;

$$T_m - T_r = (J_m + J_p)\ddot{\Theta} \tag{4.1}$$

Where T_m is the torque delivered by motors and T_r is the total resistive torque due to friction, viscosity, gravity and resistance forces. Jm is the motor inertia and J_p is the parts inertia of the moving elements expressed at joint level and includes the inertia of the product which is transported by the machine and $\ddot{\Theta}$ is the angular acceleration of the actuator. The angular accelerations required for each axis can be transferred to the end-effector point as an acceleration to check whether the designed motion exceeds the capability of the system or not.

4.5.3 Determination of permissible acceleration for the product

In some cases the end-effector may transport a product from a source to a destination and the product may be sensitive to accelerating forces (for example, a liquid in a can container). Therefore the maximum permissible acceleration and deceleration of the product are also important parameters to be calculated. The maximum allowable acceleration and deceleration zones due to actuator capacity and the product sensitivity are illustrated schematically in Fig.4.3.

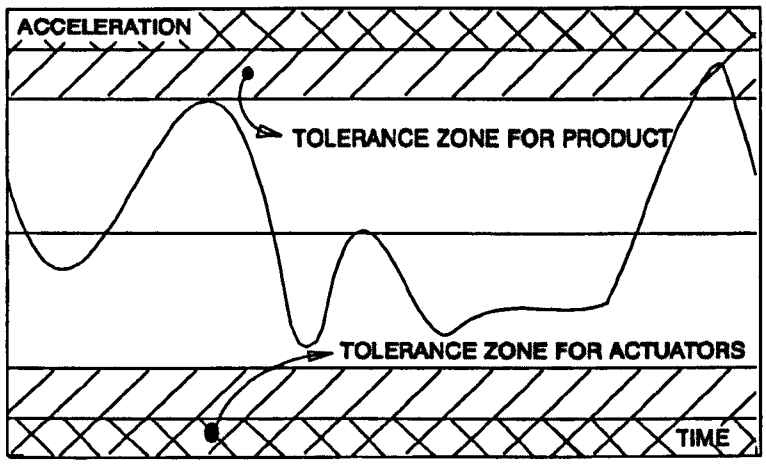


Fig.4.3 Permissible acceleration zone.

4.6 Generation of motion curves

The trajectory of a manipulator is usually determined by means of a time-based mathematical function. Where the function is subject to all necessary conditions such as desired positions, velocities, accelerations and even jerks. In addition the permissible torque capacities of the actuators and the reaction of the product to the accelerating and decelerating forces are the other parameters to be satisfied.

These details are sufficient to start a computer-aided motion design. The motion curves of the end-effector for each axis (**X** and **Y**) can be examined independently after specifying the inputs and selecting the motion laws for segments. The generation of motion curves in a graphical environment is the subject of the next chapter.

4.7 Optimization

There are many different functions including arbitrary power polynomial functions which can be used to satisfy the constraints for a given trajectory planning. Some form of selection is needed to identify those plans which are acceptable. An acceptable trajectory can be described as a trajectory which satisfies the required constraints with minimum disturbances expressed by, for example, low peak velocities and low jerk values. In order to satisfy such conditions additional steps are needed to improve the initial design.

These steps include examining the motion curves to see whether they satisfy certain parameters. The parameters and their priority of importance may change according to the operating circumstances. Most commonly, they comprise in order to importance:

(i) position tolerance envelope
(ii) permissible acceleration and deceleration
(iii) smoothness
(iv) peak velocities.

Position tolerance envelope: The position curve of each segment has to lie within a specified tolerance envelope to eliminate a possible collision of the end-effector with any other objects in the work space or for other purposes. However, sometimes the designer may not specify any tolerance envelopes. In these cases the curve of each segment can be checked to lie within a maximum tolerance envelope which is formed by the rectangular zone between the end points of each segment. The maximum tolerance envelope is described in detail in section 3.5.

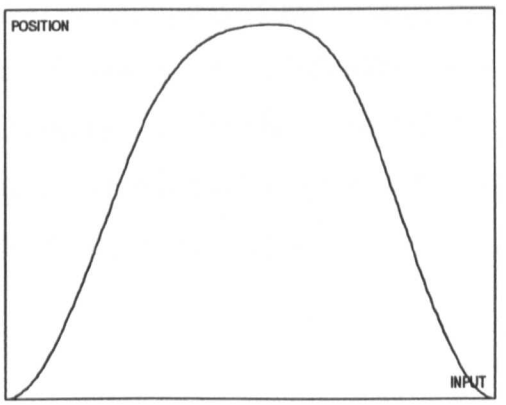
Permissible acceleration and deceleration: The dynamic constraints of a system are also important as well its kinematic constraints. However, putting these values into a function as inputs may not be possible in some cases. For example, designers usually prefer a five degree polynomial interpolation to construct motion curves which correspond to position, velocity and acceleration constraints at the ends of a segment. An acceleration curve corresponding to a five degree polynomial segment follows an oscillatory path (see Fig.4.3). Therefore to include actuator acceleration capacities (see section.4.5.2 for determination of actuator acceleration capacity) and permissible acceleration values of the product as additional acceleration inputs in the interpolation function is not practical. This is because the resulting acceleration curve for the motion will most probably violate these limiting values between the end points of a segment due to its oscillatory characteristic.

A more convenient method is to calculate the permissible acceleration and deceleration values at the end-effector point by transforming the actuators acceleration capacities and to check the acceleration curve against these limits. The designer can modify the curve if it is not in the required zone by altering boundary conditions, changing the time interval or using arbitrary powers for the polynomial function.

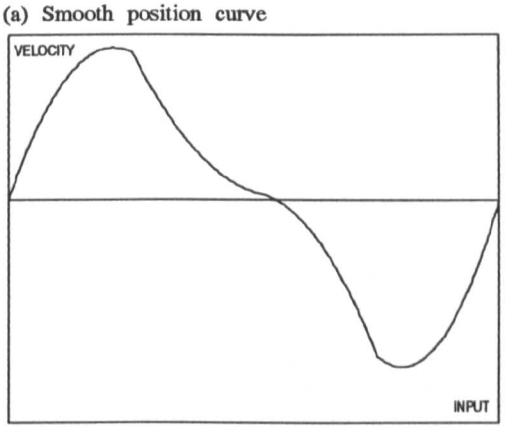
Smoothness: Generally, a trajectory is required to be as smooth as possible to reduce vibration, noise and wear effects. The degree of smoothness will depend on the degree of the continuity of the motion curves. For example, consider a trajectory which has a continuous velocity curve only whereas a second trajectory gives continuous velocity and acceleration curves (Fig.4.4). For this example, we can say that the second trajectory is smoother that the first. This implies that smoothness is directly associated with the continuity of the curves. The question is which derivative should the motion curves of a trajectory be continuous up to. Since-high degree continuity requires a high-degree polynomial interpolation the continuity of motion curves should considered within reasonable limits. Usually continuous velocity and acceleration conditions which corresponds to a five degree polynomial provide effective trajectories. In some cases, however, continuity of the jerk curve may also be compulsory especially at high speeds.

Designing a trajectory using five or seven-degree polynomial segments needs six or eight boundary conditions respectively. Some of these may be geometric constraints on the system which are fixed and therefore cannot be changed. However, others may be arbitrary boundary conditions which are specified for the continuity of motion curves.

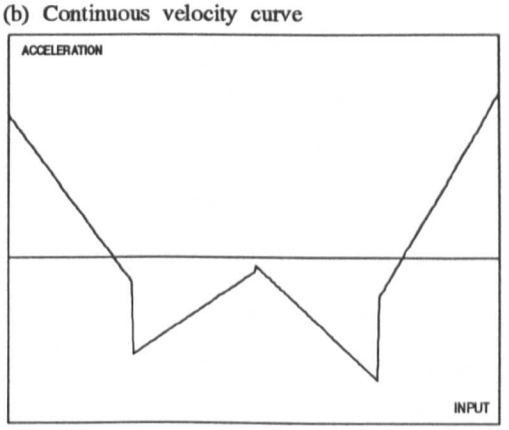
Maximum velocity: The peak velocity value in the curve is one of the most important parameters which determines an optimum trajectory. A trajectory should be smooth and satisfy the required conditions with the peak velocities in the curve as low as possible. A high velocity may not be desirable when transporting a product while it also may cause instability of



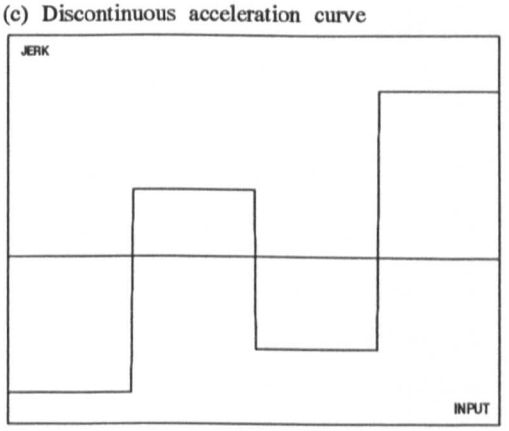
(a) Smooth position curve



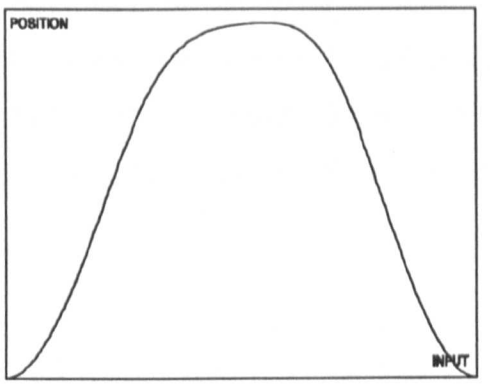
(b) Continuous velocity curve



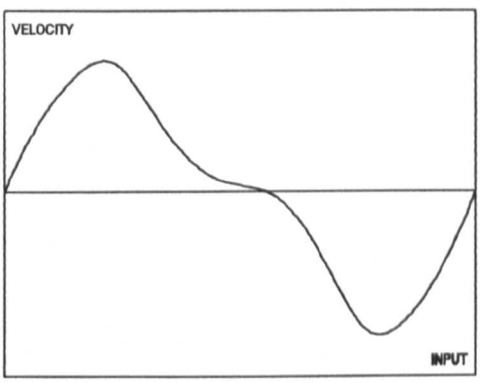
(c) Discontinuous acceleration curve



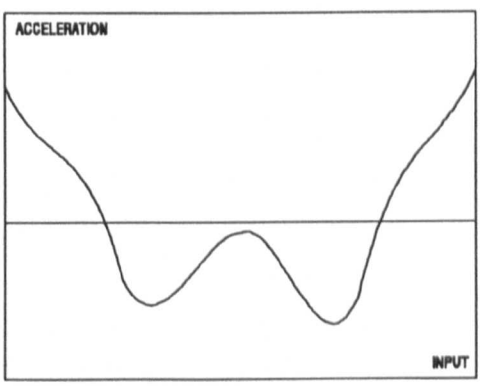
(d) jerk is infinite at the ends of segments



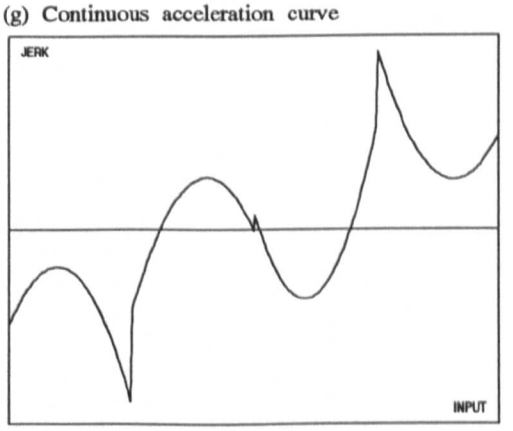
(e) Smooth position curve



(f) Smooth velocity curve



(g) Continuous acceleration curve



(h) Discontinuous jerk curve

Fig.4.4. Relationship between continuity and smoothness.

the system particularly at high speeds. The ability to use dummy boundary conditions and the availability of different motion laws, especially polynomial interpolations together with the new arbitrary powers method, present a wide range of potential solutions. The designer may achieve an optimum solution by trying these options.

CHAPTER 5
GENERATION OF MOTION CURVES

5.1 Introduction

A new approach to manipulator motion planning is to generate motion curves in a computer-graphics environment and simulate the motion of the end-effector in an off-line manner rather than planning the trajectory in real time. Such an approach will allow the user to analyse the motion of the end-effector without needing to access the machine system. Therefore, the user can modify any undesirable behaviour of the trajectory due to the interpolation function or any other reason while in the computer-graphic environment.

In order to fulfil this task, a motion design program called **MOTDES** is developed. The program is capable of producing motion curves for the end effector of a mechanism which can perform a body motion in a plane, that is, the end effector can move in the (X-Y) plane and rotate about an axis perpendicular to the plane. Several motion types have been included to provide a wide degree of flexibility for the user. The program includes many facilities to achieve a desired motion. The necessary boundary conditions comprise the main input for the software; other interactivity is based on mouse click selection. The user can examine the motion curves in a graphical environment. If they are not appropriate then he may use modification menus to improve the characteristic of the motion. Motion curves for all three axes

(X,Y,Θ) can be designed simultaneously. The program also includes a simulation option which shows how the product moves along the trajectory. The resultant trajectory can be saved for later purposes.

5.2 The software (MOTDES)

The motion design program is written in the Pascal programming language under the GEM (Graphics Environment Management) for AT compatible PC machines. GEM is selected because of its graphics input and output capabilities and the efficient interface provided between the user and computer. The program can produce motion for single or multidegree planar mechanisms. Three dimensional motions are not included. The program consist of three basic stages:

(i) input module for trajectory building
(ii) interactive module in order to achieve the required trajectory
(iii) simulation of the motion of the end-effector.

The program is designed to provide an efficient interface between the user and computer such that it is possible to design a motion just by using the mouse-click option only without needing any communication through the keyboard. In trajectory building, the user communicates with the program in order to specify the boundary conditions. The motion curves appear on the screen when the first stage is finished. Smooth interactivity, easy to use and analysis capabilities of the program allow the user to improve the trajectory by trying different options until a satisfactory trajectory is obtained. This may be achieved by changing boundary conditions, selecting different motion types or other options. The motion of the end-effector can be simulated in a Cartesian co-ordinate system at any time after finishing the draft planning.

5.2.1 Trajectory building

Some information must be supplied to the program by the user in order to build a trajectory. On first executing the program, the curves of a default

trajectory appear which includes four segments for each dimension (X,Y,Θ). Five-degree polynomial interpolations have been used to produce the motion curves for each of these segments. The user can easily delete or insert additional segments to the default trajectory if necessary while any boundary conditions may be changed in order to obtain a new trajectory. However it should be remembered that different motion types require different inputs. For example, the user must specify position, velocity and acceleration boundary conditions for a polynomial function at the ends of the segments in order to produce a continuous acceleration curve. On the other hand, cam motions require only the specification of positions. Cubic splines are suitable for non-segmented solutions and velocity or acceleration values can be specified at the ends of the curve but not in between. Therefore the user first should select the motion type for each segments and then detail the inputs.

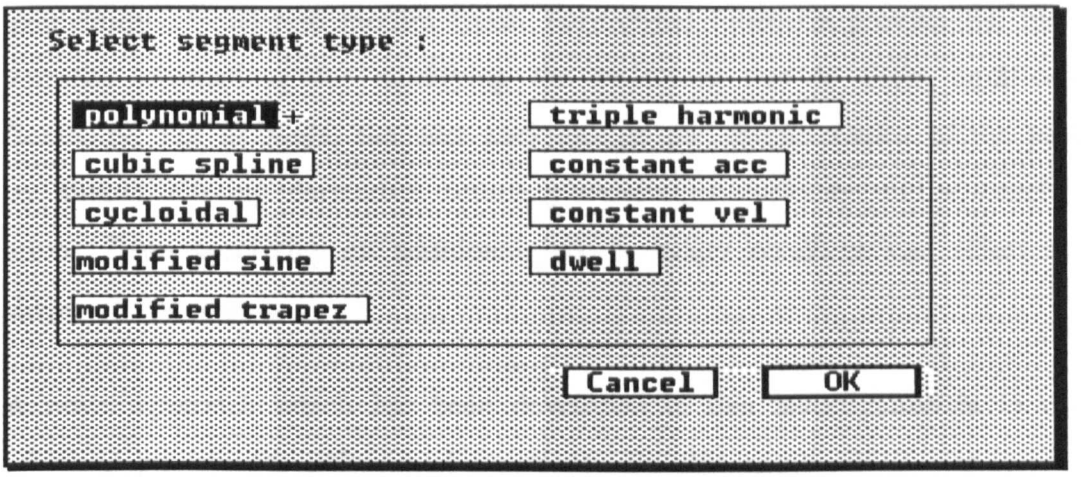
The available motion types used in the program and some input menus for them are shown in Fig.5.1(a-e)

5.2.2 Interactive trajectory generation

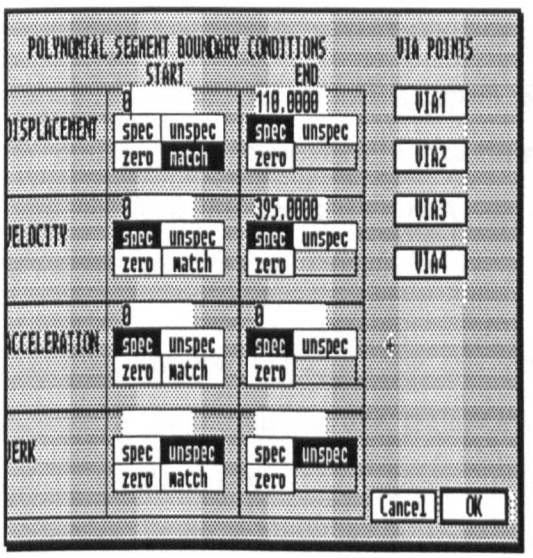
The selection of motion types for each segment independently and the specification of the inputs corresponding to the boundary conditions forms the first stage in generating a trajectory. After this, the user can examine the resulting motion curves to see whether the trajectory is acceptable or not. The most important factors to be checked for may be listed as follows:

(i) degree of meandering
(ii) smoothness
(iii) peak velocity values
(iv) peak acceleration values.

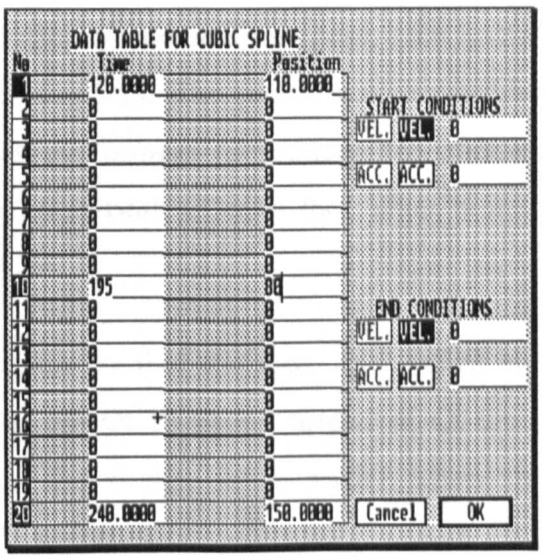
The significance of these factors has been described earlier in section 4.5. If the trajectory is not acceptable the user can modify it by making use of other options available in the program. The most important option is the



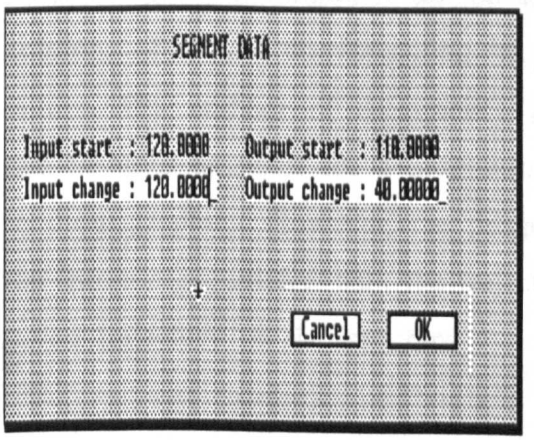
a) motion types



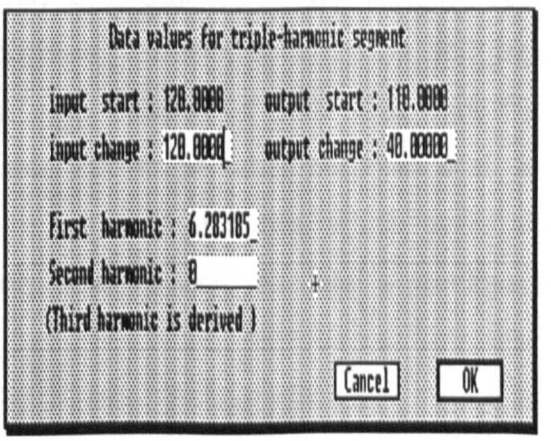
b) input menu for polynomial segment



c) input menu for cubic spline segment



d) input menu for cycloidal segment



e) input menu for triple harmonic segment

Fig.5.1(a-e). Motion types and their input menus.

availability of different motion types. Polynomial functions provide the most versatile alternatives for modifying the trajectory due to their significant flexibility. For example, the user may change the characteristics of the path by simply altering derivative boundary conditions (velocity and/or acceleration). Even specifying an additional jerk boundary condition can effect the shape of the curves.

A novel approach which has been developed in this investigation is to use arbitrary powers for polynomial functions. By this means, it is possible to alter the powers to produce thousands of polynomial curves for the same boundary conditions of a segment. It is observed that arbitrary powers can provide curves which have lower peak velocity and/or lower peak acceleration values than the other motion types. See Fig.3.13. The program also provides an option that automatically determines the powers to produce the shortest path between the end points assuring a lower peak velocity than other motion laws for a segment.

Another important facility of the program is that the user can adjust the boundary conditions for the trajectory in the (X-Y) plane instead of (X-time) or (Y-time) plane. This option makes the design much easier when the objective is to achieve a difficult path shape.

After designing a motion it can be saved as a data file which includes four columns of a data corresponding to position, velocity, acceleration and jerk values. The number of data points in each column is 300. This data file can be used as input for a programmable system or other purposes such as to draw graphics. However, the program has another option which saves all the information related with the trajectory thus the user can load a previously saved trajectory to the program.

5.2.3 Simulation of the motion of end-effector

The path can be displayed to examine the motion of the end-effector. An arrow is used to represent the position and angle, made with respect to the

horizontal axis, of the end-effector along the path. The values of the co-ordinates (X,Y,Θ) can also be displayed while the arrow moves along the trajectory (see Fig.5.12). The simulation option gives an indication to the user how the end-effector accelerates or decelerates along the path. In addition, the corresponding velocity, acceleration and jerk curves can also be displayed.

5.3 Data structure

Normally, it is assumed that all items of data are kept within some kind of array which is dimensioned before the program is executed. The maximum amount of memory that would be needed for arrays should be decided when writing a program. If the program is run for a small set of samples, then much of the space will never be used. If the program is run for a large set of samples, then the allocated space may not be large enough, even when the computer memory itself is not fully used, simply because the original bounds on the arrays were too small.

Even if the arrays are carefully declared large enough to use up all the available memory, the program can still encounter overflow, since one array may reach its limit while a great deal of unused space remains for others. Since different runs of the program may cause different lists to grow or shrink, it may be impossible to tell before the program actually executes which lists will overflow.

To avoid the above difficulties a dynamic data structure is used to write the program. With such a data structure we add storage only as it is required and in the case of deletion we can return unwanted storage for re-use. There are different types of data structures but mainly they can be divided into two groups, list structures and tree structures.

5.3.1 List structure

A simple list structure consists of a set of nodes linked together by a special pointer at the head of each node. The data fields of each node contain some sort of information and the end of the list is signalled by a node containing a nil pointer as shown in Fig.5.2.

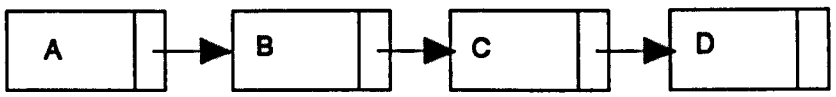


Fig.5.2. A simple list structure.

During the execution of the program a new node can be inserted into the list or a node can be deleted from the list as shown in Fig.5.3-4.

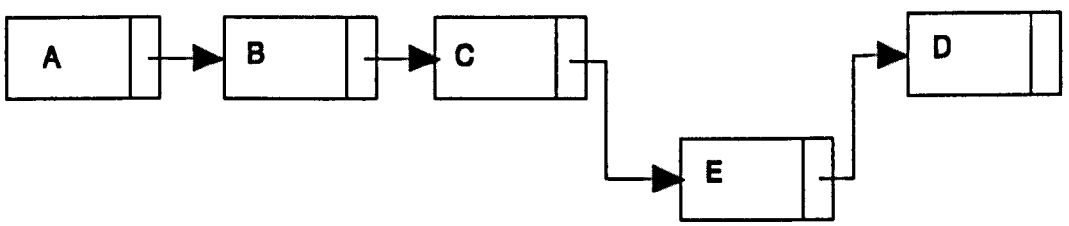


Fig.5.3. Insertion of a node to a list.

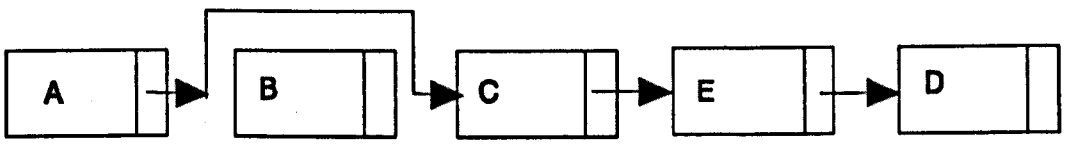


Fig.5.4. Deleting a node from the list.

More complex structures can be created by adding lists to each other. Such structures are known as linked lists. For example the structure of the MOTDES program is based on linked lists as is shown in Fig.5.5. The main list includes three nodes which hold information about axes (X,Y,Θ) of a trajectory. Each axis has a corresponding list of nodes which contain information about the segments associated with the axis. Furthermore, each segment node is formed from another sub-list of nodes in which information at the design points such as position, velocity, acceleration and others is stored.

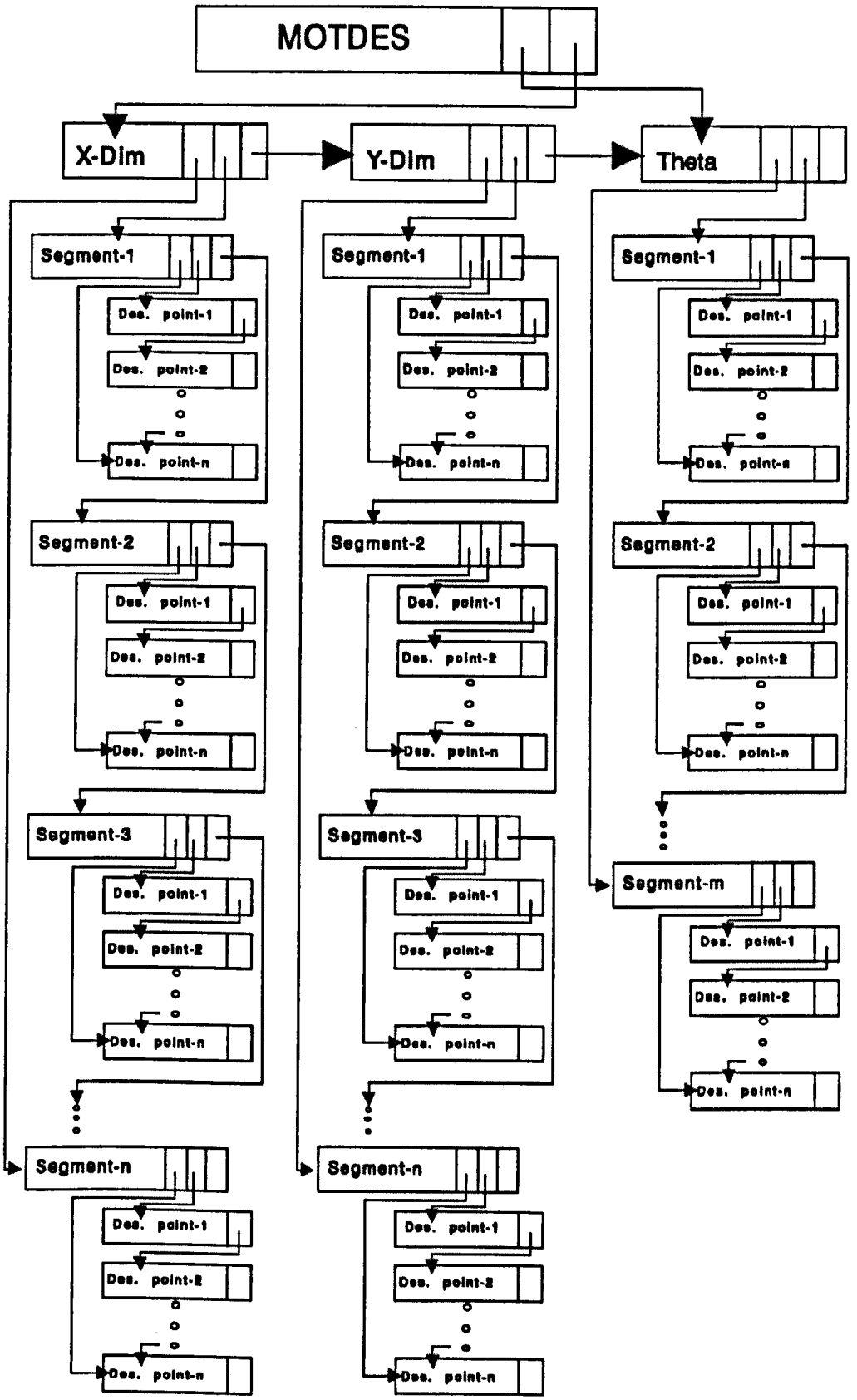


Fig.5.5. Data structure for MOTDES.

5.3.2 Tree structure

Trees are non-linear data structures used in computer programming. There are several ways to represent tree structures, but the most understandable way is to sketch them graphically.

We can describe a tree structure as a finite set of one or more nodes, one of them is a special node called the root and the remaining nodes are separated into disjointed sets, where each of these sets is called a sub-tree of the root.

A tree structure is one in which items of data are related by branches. If, in a structure, the maximum number of branches from any one node is always two, such a tree is called an ordered or binary tree. Trees that have more than two branches are called general trees or descendant trees.

Their shapes resemble upside down trees. A tree structure is one in which items of data are related by branches. Commonly trees are classified into two different groups, *ancestor* and *descendant*, according to their objectives. In ancestor trees, each root is considered as the child of nodes of its sub-trees. For example, the tree in Fig.5.6 shows the ancestors of A. B and C are the parents of A. B's parents are D and E who are also grandparents of A.

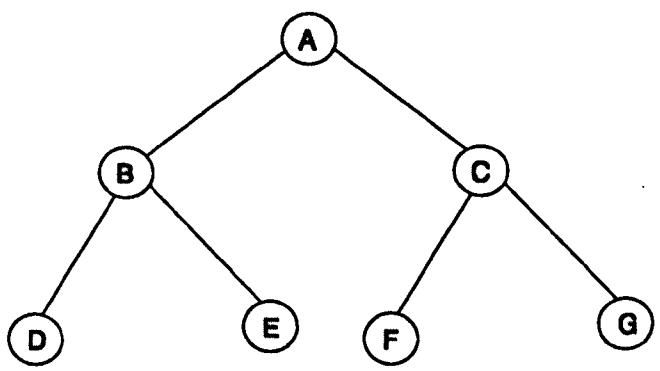


Fig.5.6. An ancestor tree.

In descendant trees, each node is considered as the parent of the root nodes of its sub-trees. As an example A has two children B and C. D, E, and F are the children of B, whereas, C has only one child G as shown in Fig.5.7.

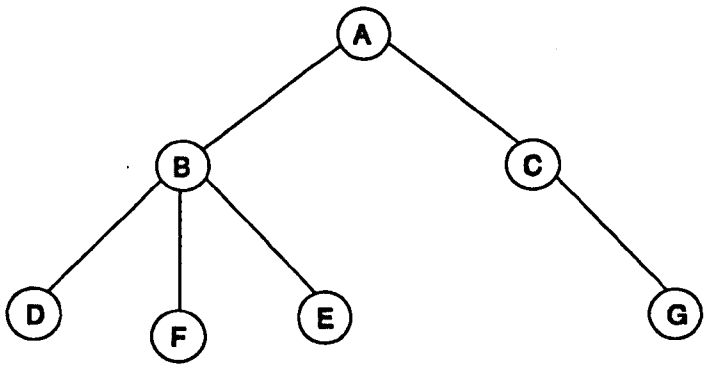


Fig.5.7. An descendant tree.

Some simple expression of binary trees are shown in Fig.5.8 and 5.9.

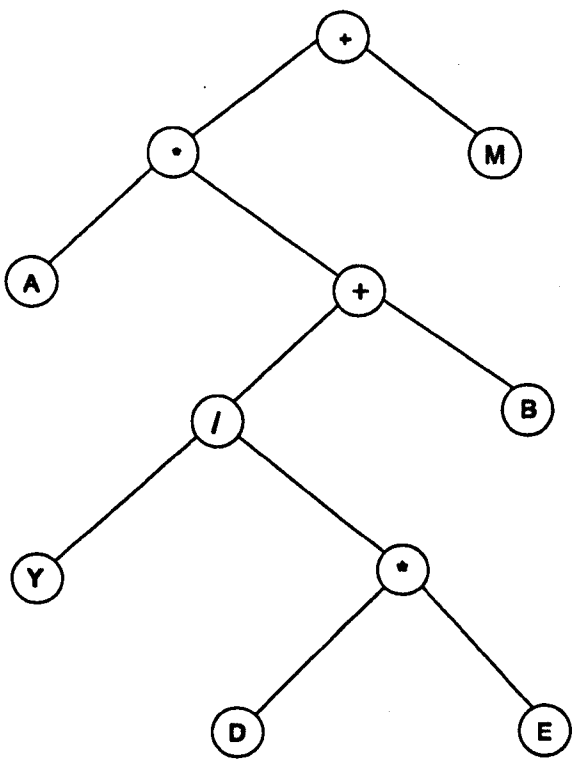


Fig.5.8. The expression tree for { $A\left(\frac{Y}{D*E}+B\right)+M$ }.

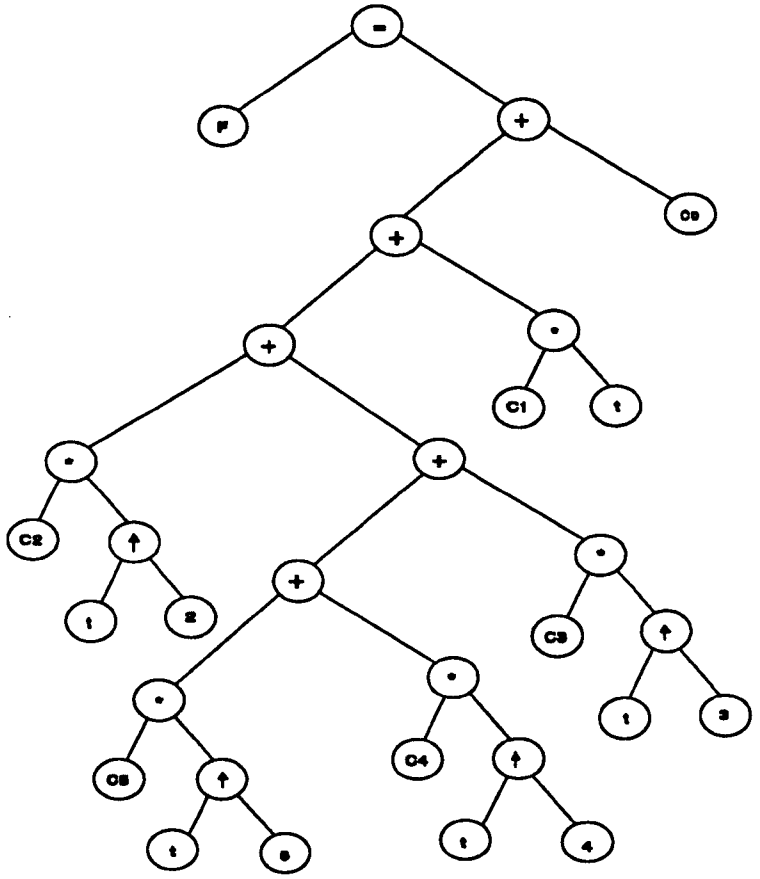


Fig.5.9. The expression for a five degree polynomial function using binary trees. (exponentials are denoted by ↑).

More information about data structures can be found in [5.1] and [5.2]

5.4 Generation of example trajectories

Different trajectories are presented in this section to explain the problems of motion design and solution procedures for them. The other purpose is to introduce **MOTDES** using figures which are directly taken from the program.

Example-1: The first example is taken from a project currently under development [5.4]. The problem is to insert a fuse element into a fuse cartridge. Bending and welding of the ends of the fuse element are follow-on processes to be performed by a manipulator. The path for the processes is shown in Fig.5.10. The points shown on the path indicate the positions of the design points. The structure of the insertion mechanism which comprises the manipulator, an X-Y table and welding tools is given Fig.5.11. This complex process is required to be carried out in 0.666 seconds (90 rpm). The path includes eight segments. The data for this process is given in Table 5.1 where all the units are in millimetres and seconds. However, all the boundary conditions shown in the table are not compulsory. Some of them are specified by the user in order to obtain smooth and continuous curves.

The path for the process that is designed using the program is shown in Fig.5.12. The motion curves for each co-ordinate axis with respect to time are given in Fig.5.13. In the first segment, the manipulator grips the product from the stock point (**P1**) and inserts it into the fuse cartridge. After the insertion, the bending process for the both sides of the fuse element take place at point (**P2**). As seen from Fig.5.13, the bending and welding operations are done in dwell segments. The cartridge is also manipulated by a computer controlled X-Y table in order to provide co-ordinated motion (see Fig.5.11). The manipulator and X-Y table move the fuse element and cartridge to another point (**P3**) for the welding operation of the right side of the fuse. The next point (**P4**) on the path is for the welding operation of the left side. By finishing the last welding operation, the manipulator starts to withdraw the insertion tool from the inside of the cartridge (**P5**). Then, the manipulator and X-Y table moves independently in order to start another cycle. In the task specification the point (**P6**) is required on the line which is between point (**P1**) and (**P2**). Since it is not functional its position on the line is changed in order obtain smoother motion.

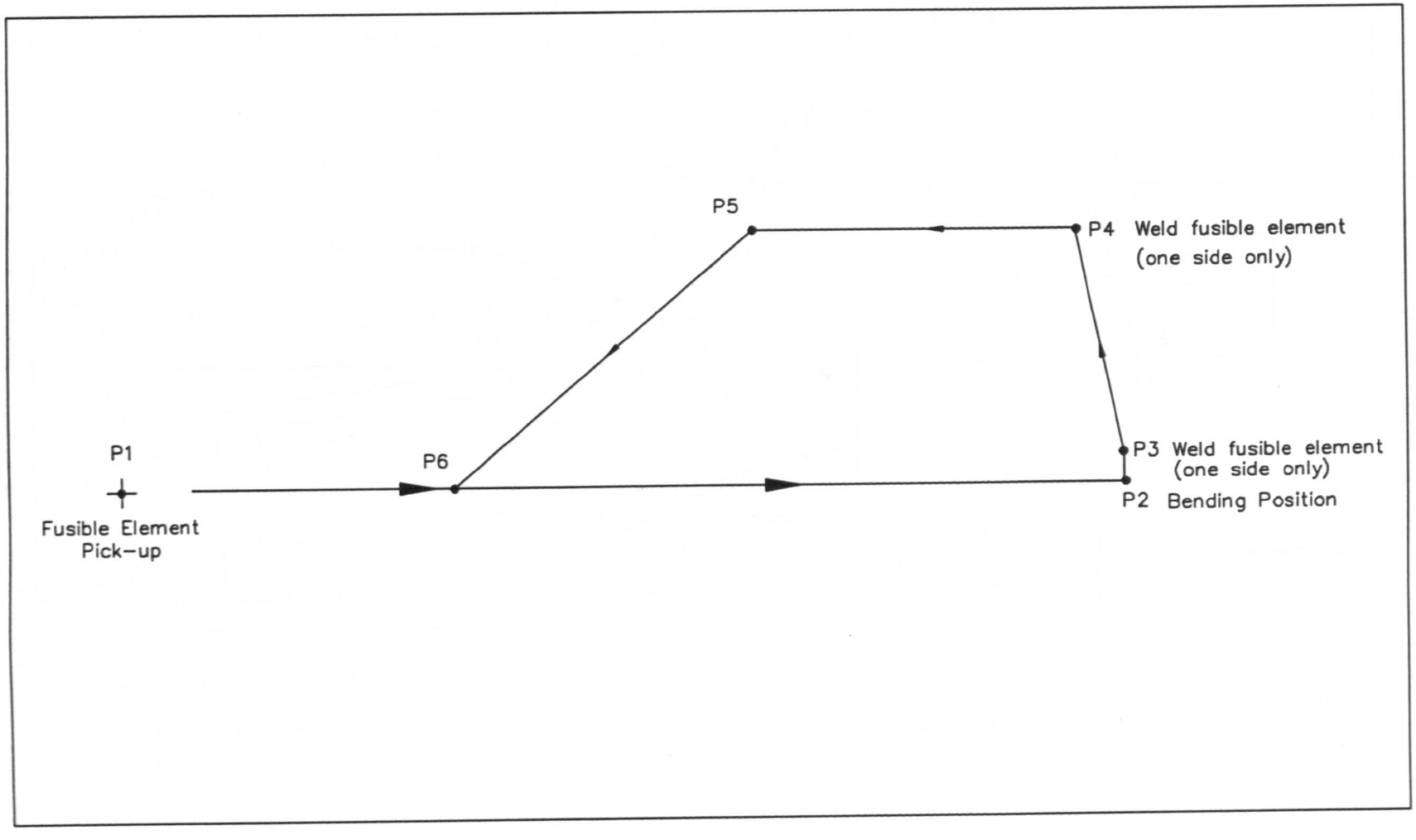


Fig.5.10 Trajectory for insertion tool to bend and weld fusible elements

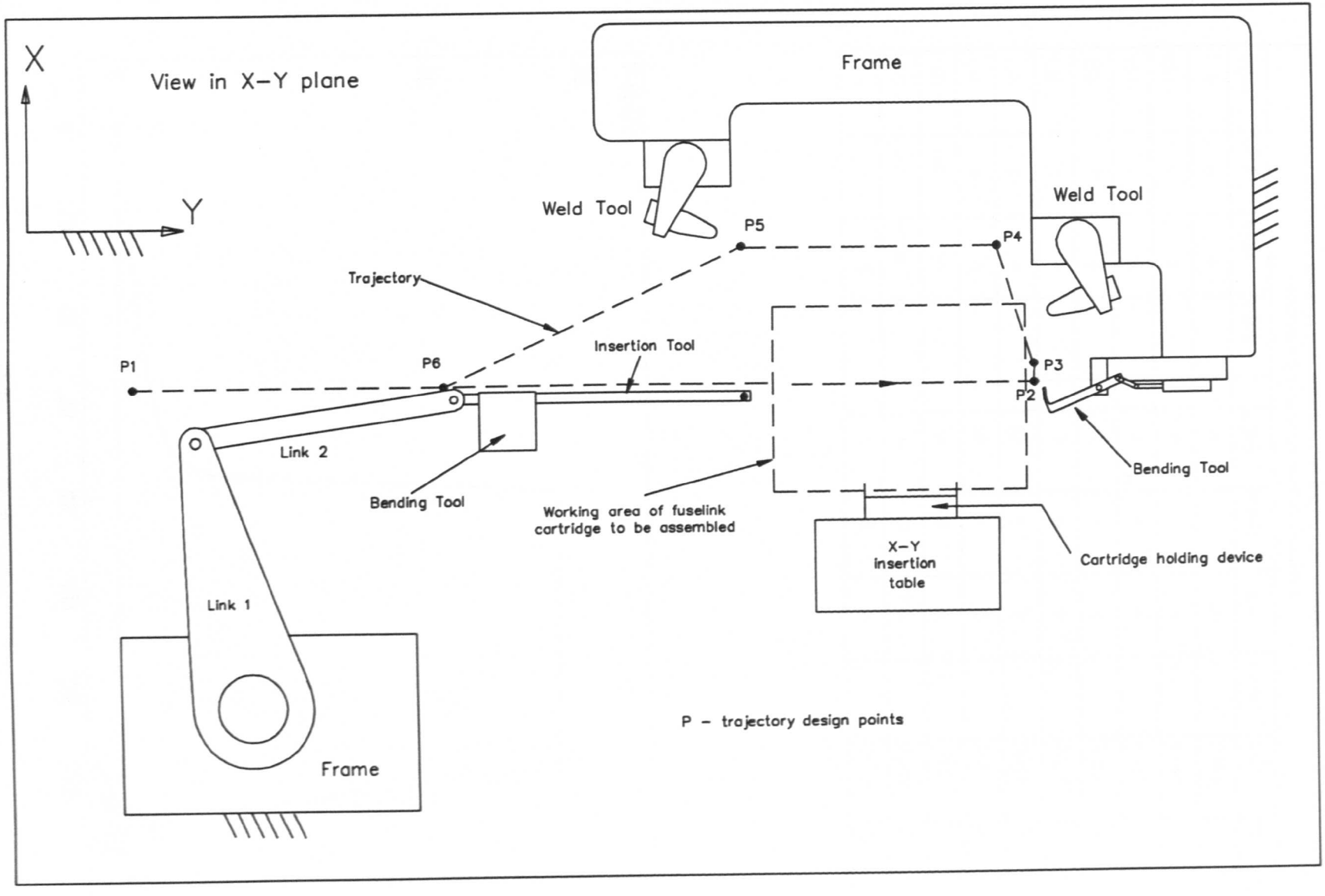


Fig.5.11 Insertion mechanism and design points for the trajectory

TABLE 5.1. Constraints used for example 1										
x-pos.	x-vel.	x-acc.	x-jerk		y-pos.	y-vel.	y-acc.	y-jerk	time	no
0	0	0	--		0	0	0	--	0.000	1
150	0	0	--		0	0	0	--	0.129	2
150	0	0	--		0	0	0	--	0.215	3
150	0	0	--		2	0	0	--	0.258	4
150	0	0	--		2	0	0	--	0.344	5
138	0	0	--		28	0	0	--	0.408	6
138	0	0	--		28	0	0	--	0.494	7
82	-116	3920	--		28	0	0	--	0.537	8
75	-43	-1020	--		6	-62	760	--	0.580	9
0	0	0	--		0	0	0	--	0.666	10

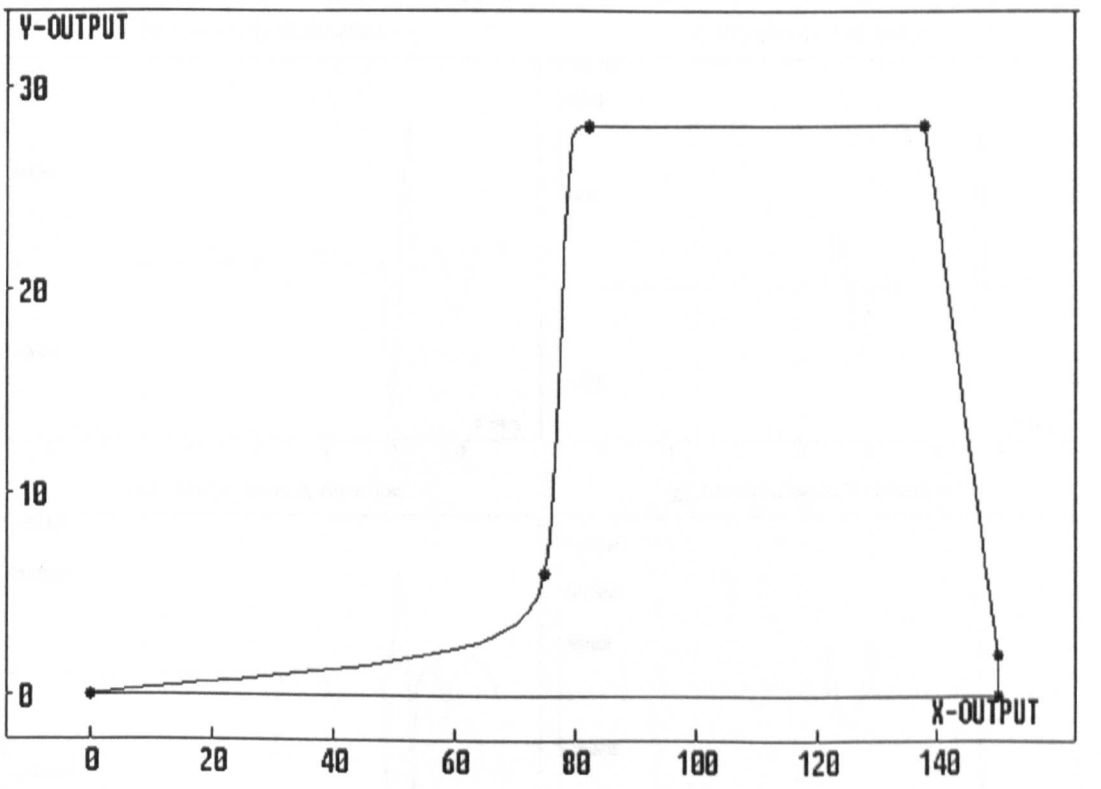
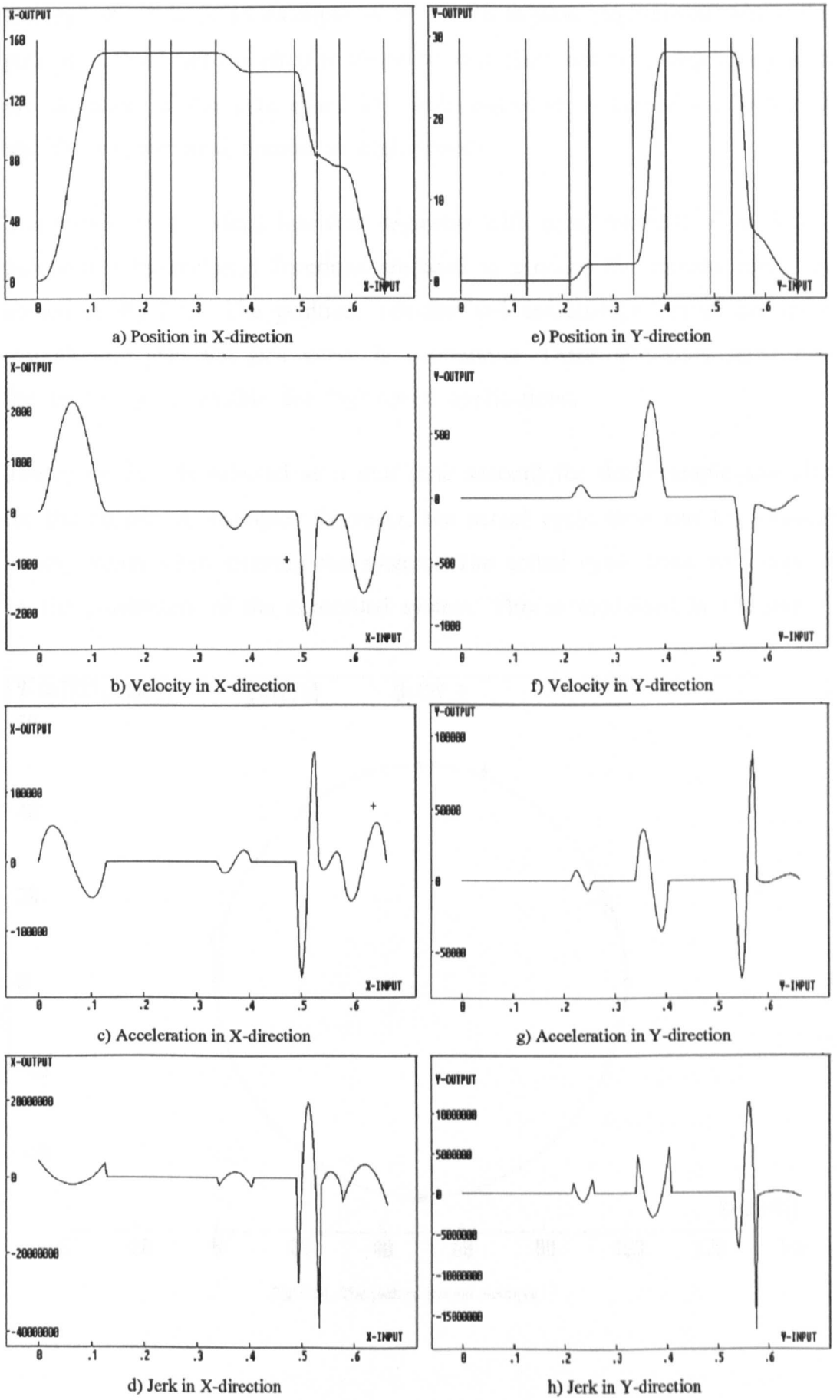


Fig.5.12. The path of the first example.



a) Position in X-direction

e) Position in Y-direction

b) Velocity in X-direction

f) Velocity in Y-direction

c) Acceleration in X-direction

g) Acceleration in Y-direction

d) Jerk in X-direction

h) Jerk in Y-direction

Fig.5.13. Motion curves of the first example.

Example-2: This is an example of a smooth motion requirement where the path is defined with a circular shape so that there are no sharp changes in the direction of the path (Fig.5.14). This trajectory is specially selected to test the experimental system at high speeds.

The trajectory is divided into four segments with equal intervals. Five degree polynomial interpolation functions are used to produce the motion curves as shown in Fig.5.15. The position, velocity and acceleration curves are quite smooth and also the jerk curve is continuous. These situations show that the trajectory is feasible for high-speed applications.

The cycle time is selected as a unit (one second) for this example and also for the remaining examples, however, the actual cycle time can be adjusted to any value when running the system. The actual cycle time will depend on the parameters of the controlled system. This is explained in Chapter 7.

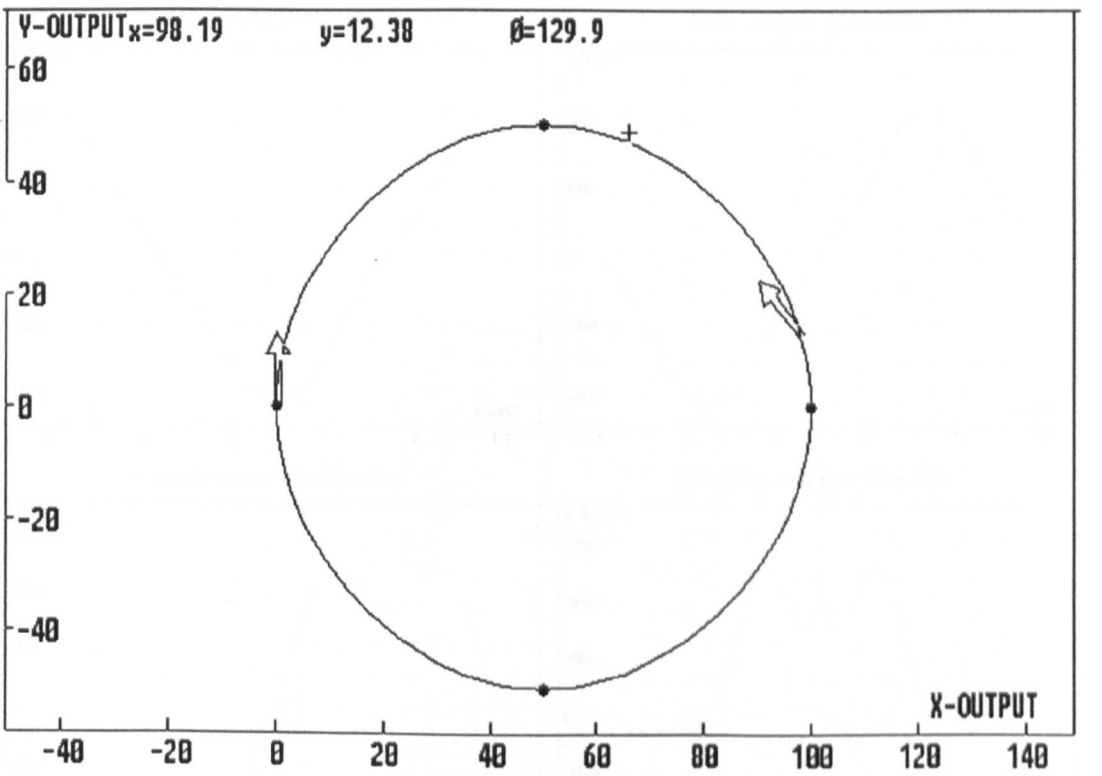
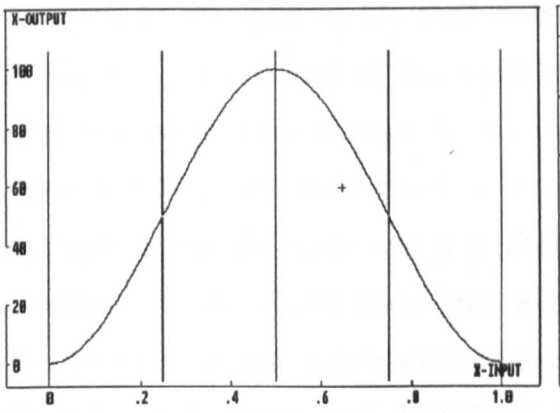
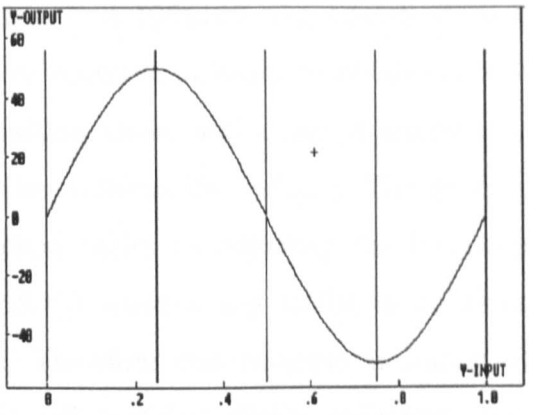


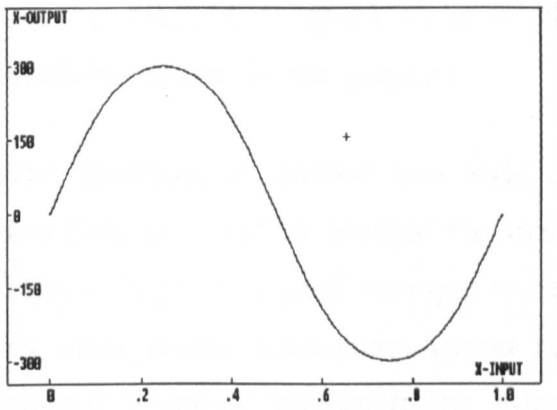
Fig.5.14. The path of second example.



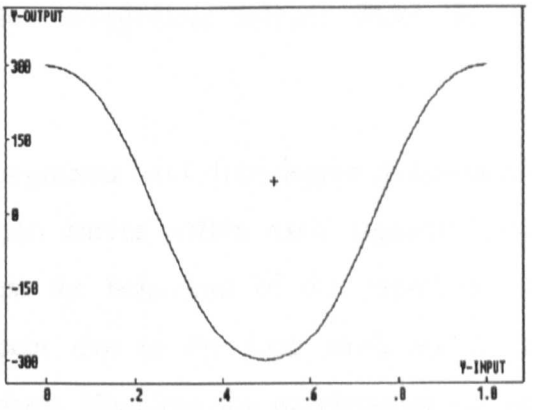
a) Position in X-direction



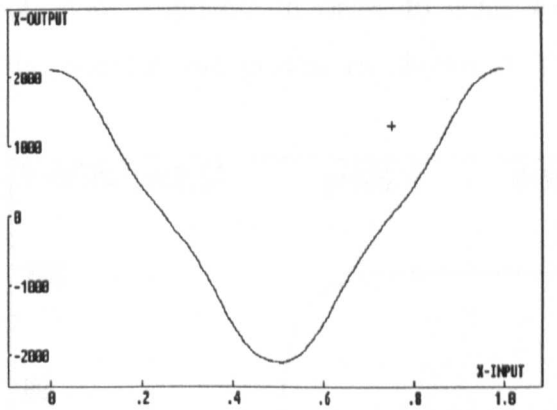
e) Position in Y-direction



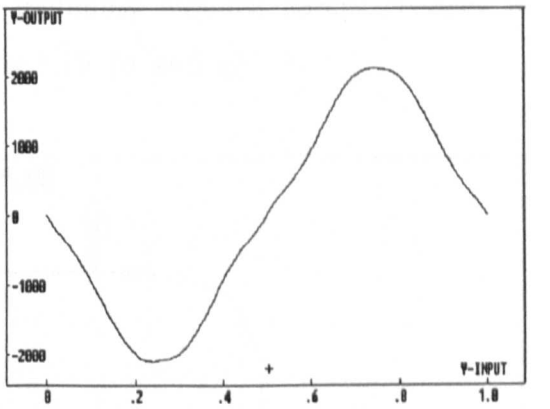
b) Velocity in X-direction



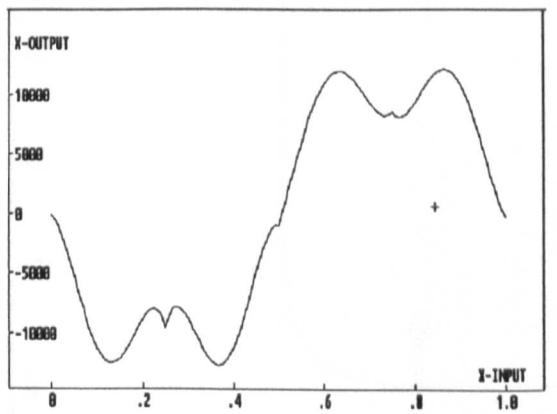
f) Velocity in Y-direction



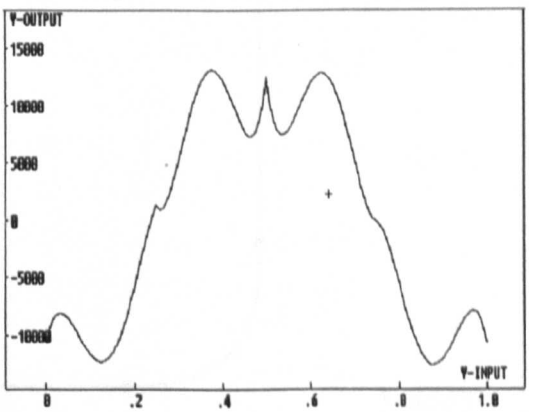
c) Acceleration in X-direction



g) Acceleration in Y-direction



d) Jerk in X-direction



h) Jerk in Y-direction

Fig.5.15. Motion curves of the second example.

Example-3: The path of the third example is a 100x100 mm square as seen in Fig.5.16. The corners of the square are rounded in order to obtain smooth motion curves (sharp changes in the position curve will cause discontinuous velocity curves and corresponding infinite acceleration values). The path is a simple geometric shape and it is obtained easily by adjusting the boundary conditions in the (**X-Y**) plane (see Fig.5.17) without any initial calculations for the velocity and acceleration values. Therefore this example is important since it demonstrates how **MOTDES** is efficient for simple trajectory generation. Fig.5.18 is given to show the enlargement feature which is an available option in the program.

The trajectory is divided into eight segments and five-degree polynomial functions are used to produce the motion curves within each segment. The square shape is a good example to test the behaviour of the experimental rig since severe acceleration curves occur due to the four dwell and four circular segments included in the trajectory. Blending the acceleration values of these segments in order to achieve continuity requires abrupt changes in the acceleration curves as shown in Fig.5.19 (c and g)

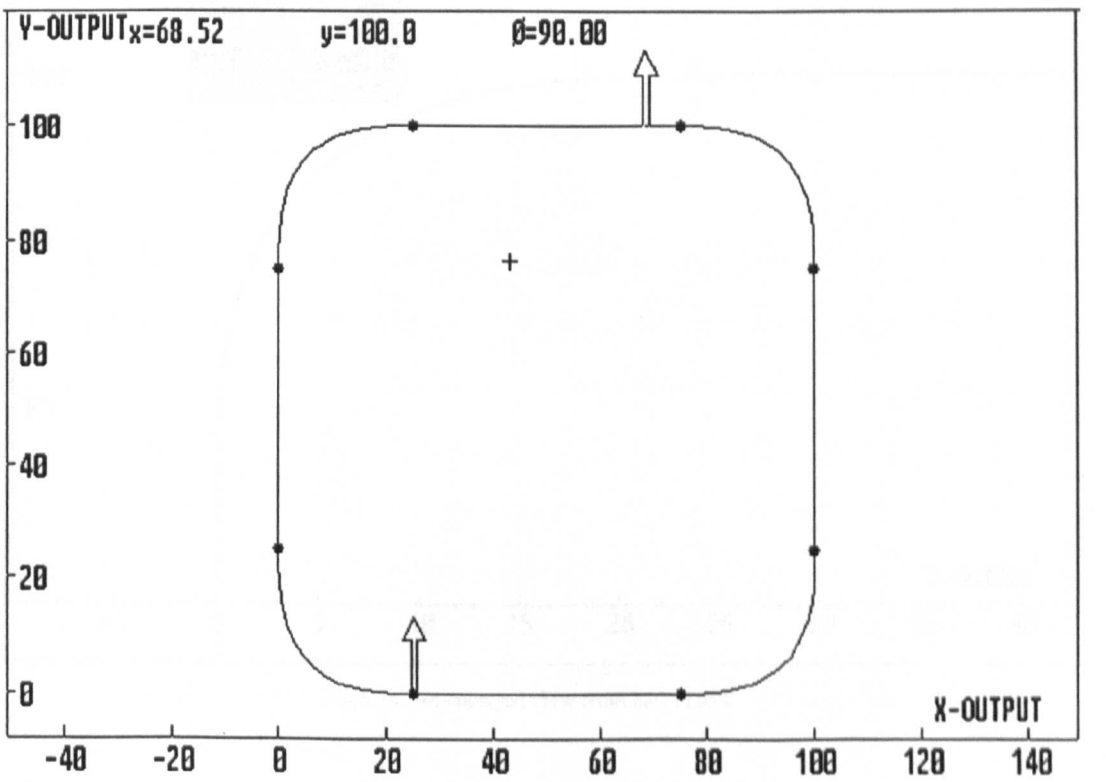


Fig.5.16. The path of the third example.

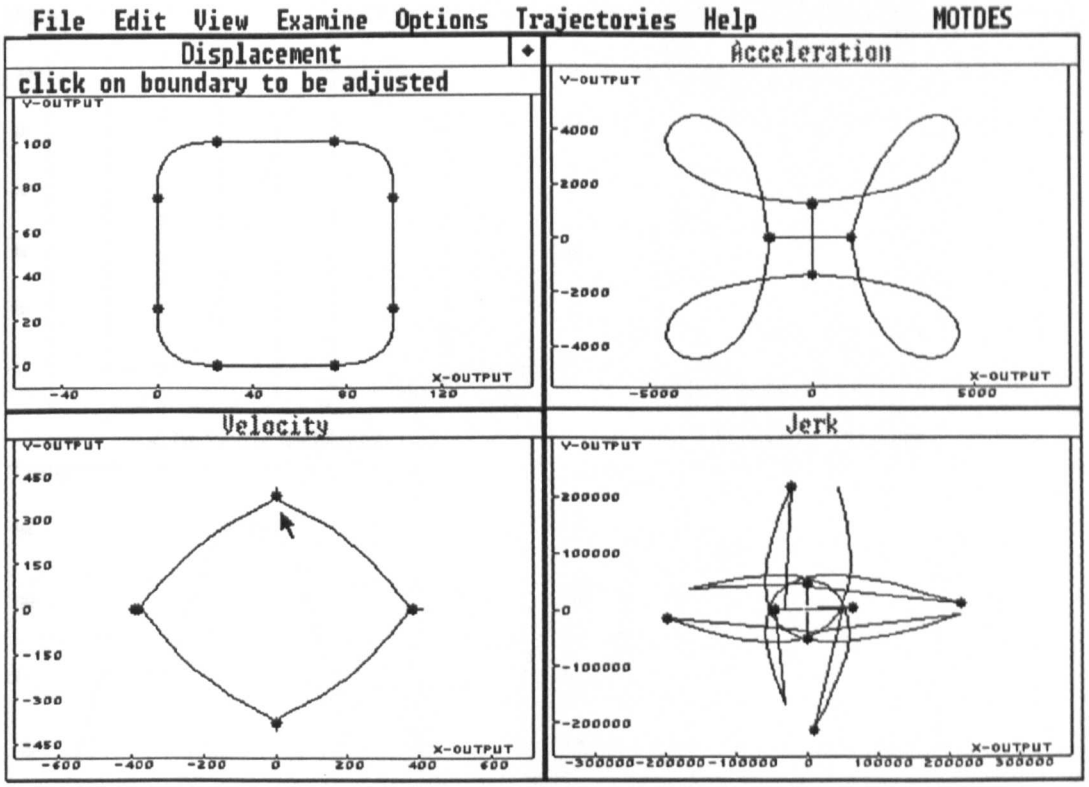


Fig.5.17. A view from MOTDES.

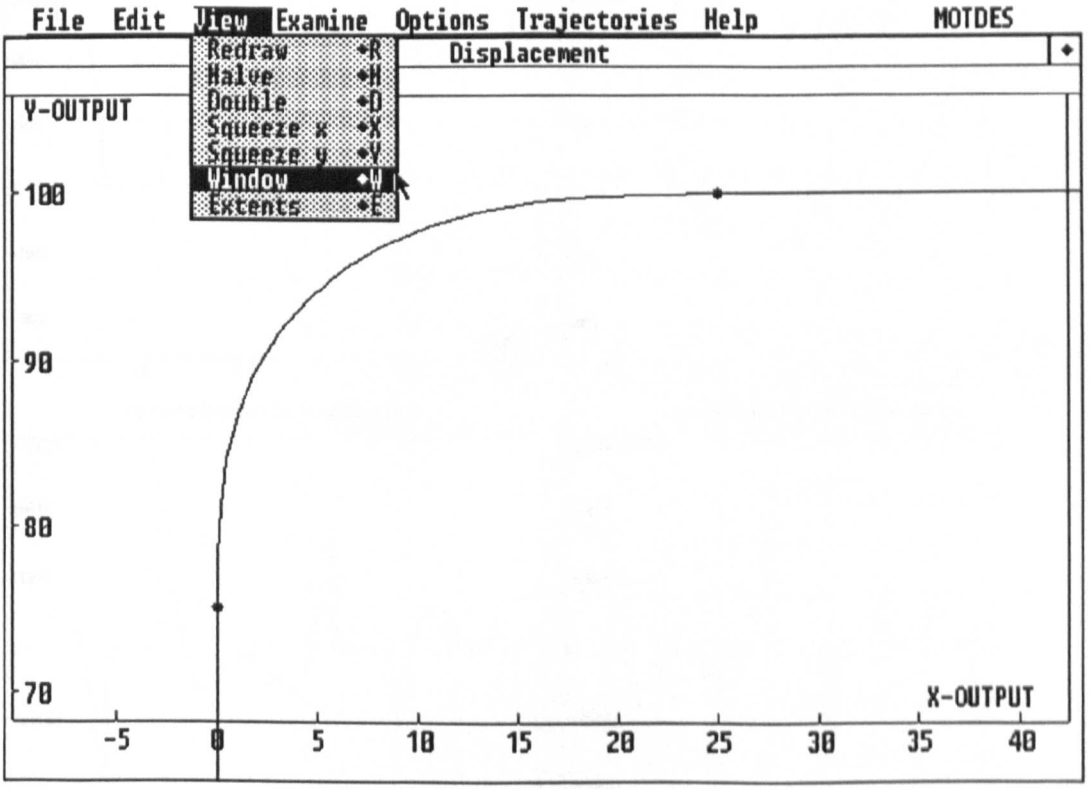
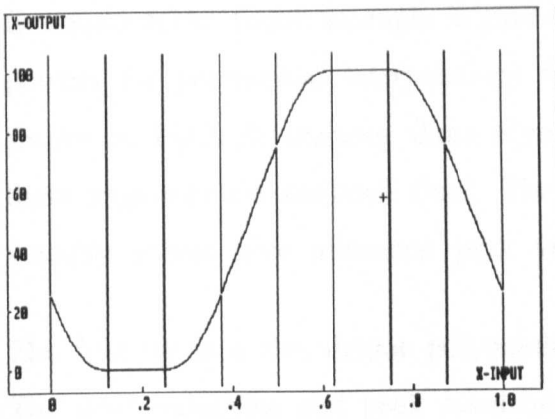
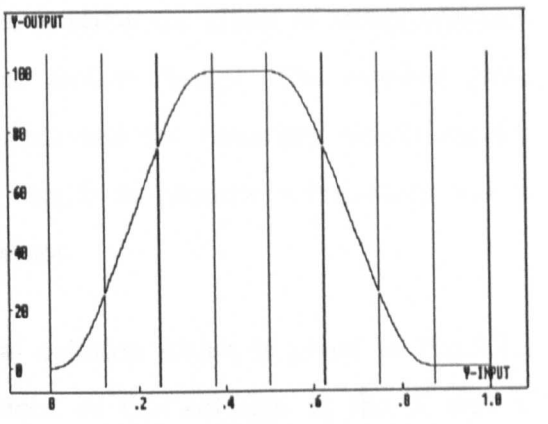


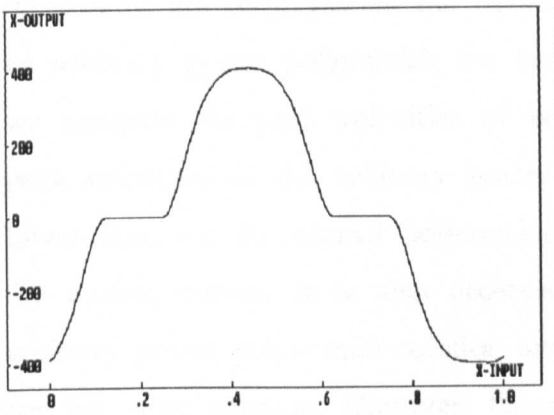
Fig.5.18. An enlarged view from MOTDES.



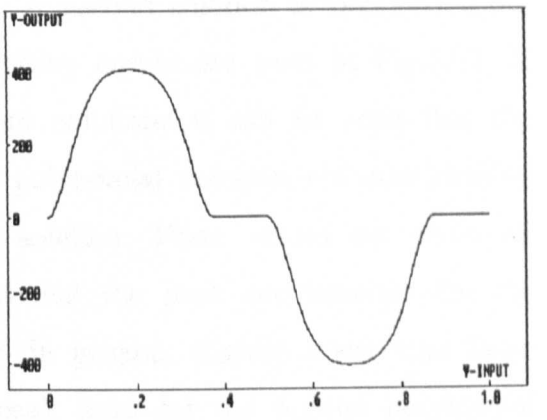
a) Position in X-direction



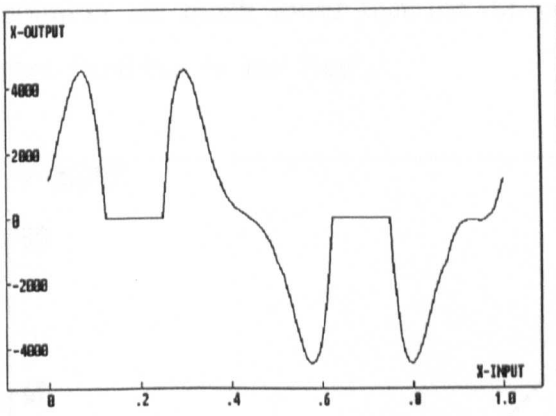
e) Position in Y-direction



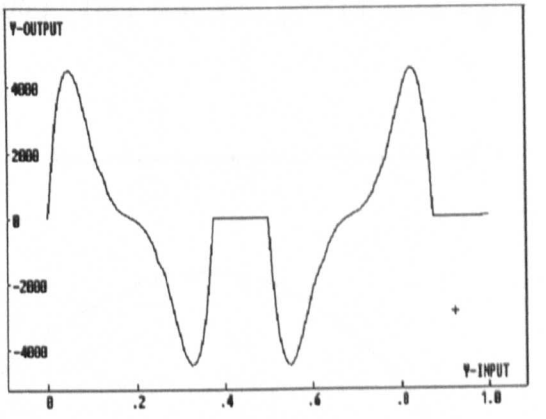
b) Velocity in X-direction



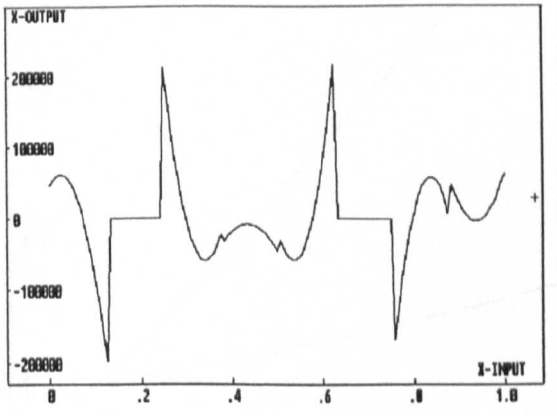
f) Velocity in Y-direction



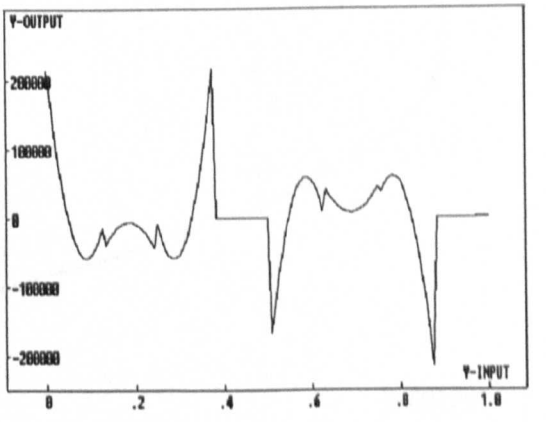
c) Acceleration in X-direction



g) Acceleration in Y-direction



d) Jerk in X-direction



h) Jerk in Y-direction

Fig.5.19. Motion curves of the third example.

Example-4: The fourth example is chosen to show the effect of using arbitrary powers for polynomial interpolations for motion design. The required path, shown in Fig.5.20, includes three segments and the boundary conditions for these segments are assumed fixed. The aim is to produce a trajectory whose velocity curves give minimum peak values.

The first try is a five degree polynomial solution which is given in Fig.5.21. The peak velocities and peak accelerations of this solution in the **X** and **Y** dimensions are displayed on the figure. A second solution is obtained using an arbitrary power polynomial; the resulting curves are seen in Fig.5.22. If we compare the peak velocities of both solutions it can be seen that the peak velocities of the arbitrary power polynomial solution are considerably lower than for the normal polynomial solution. These values are given on the motion curves. It is also observed that the peak accelerations for the arbitrary power polynomial solution are ,in general, slightly lower than these for the other solution. However, the peak jerks for the normal polynomial solution are much lower that for the other. This outcome proves once again that "nothing is for free".

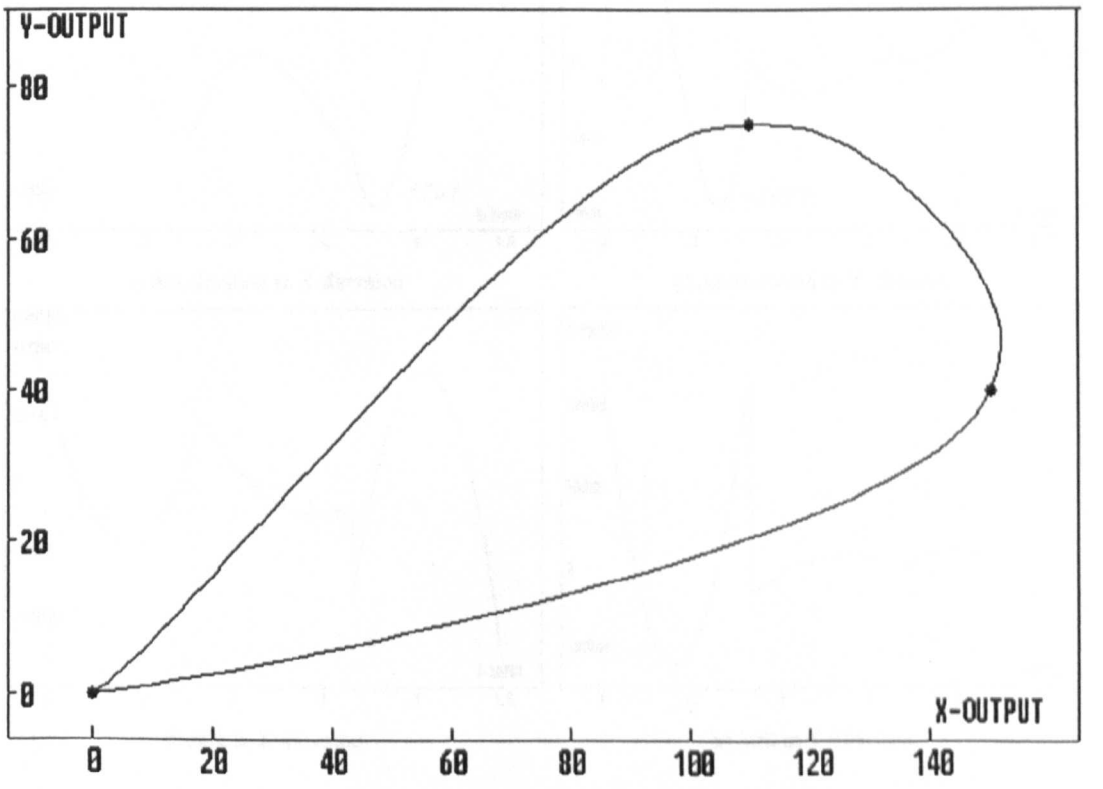
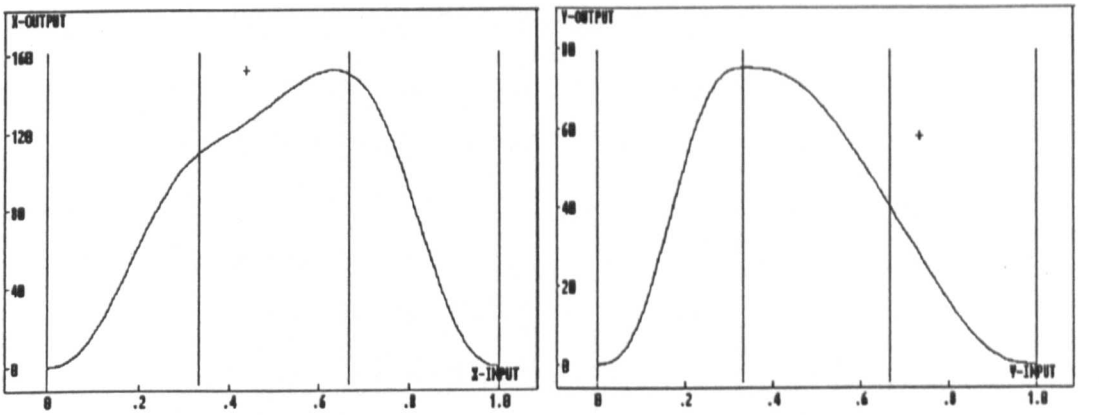
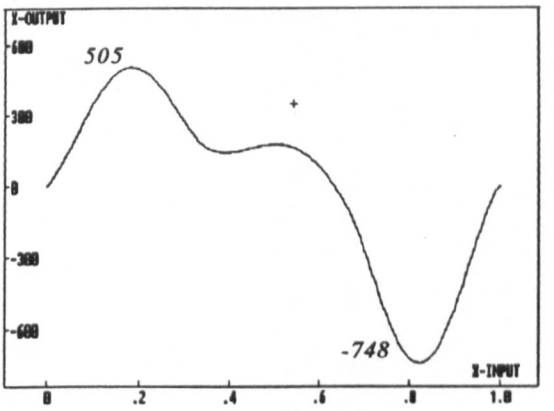


Fig.5.20. The path of the fourth example.

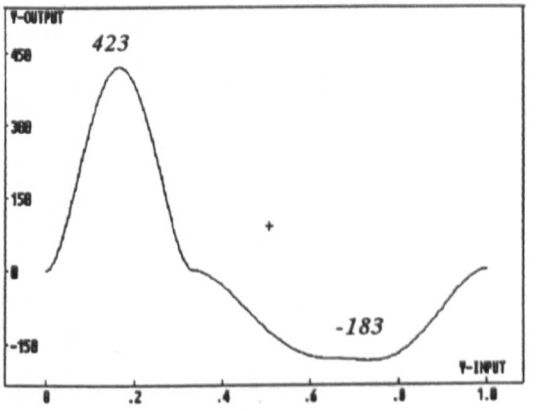


a) Position in X-direction

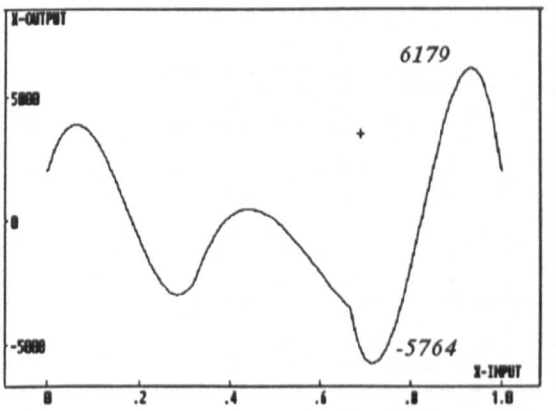
e) Position in Y-direction



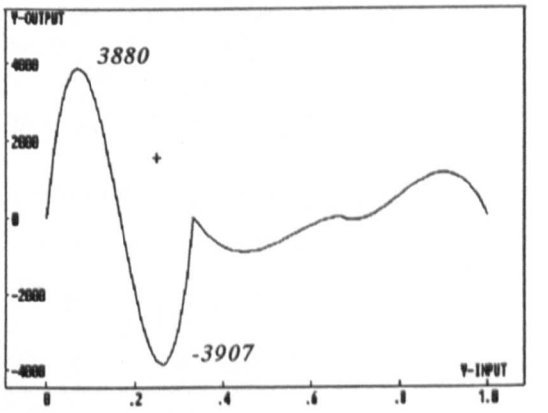
b) Velocity in X-direction



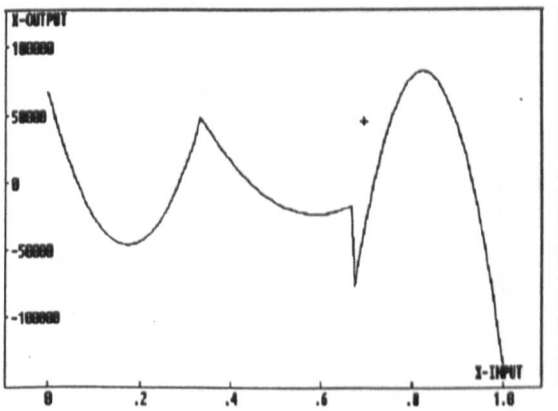
f) Velocity in Y-direction



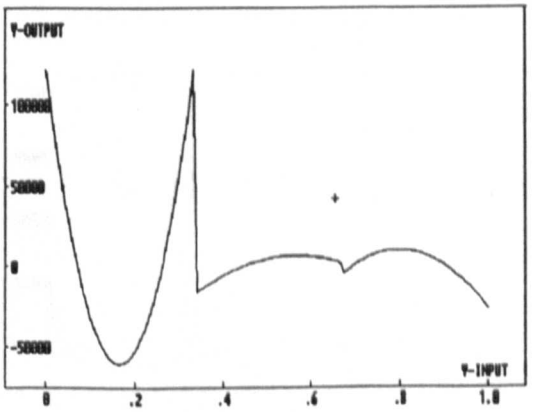
c) Acceleration in X-direction



g) Acceleration in Y-direction

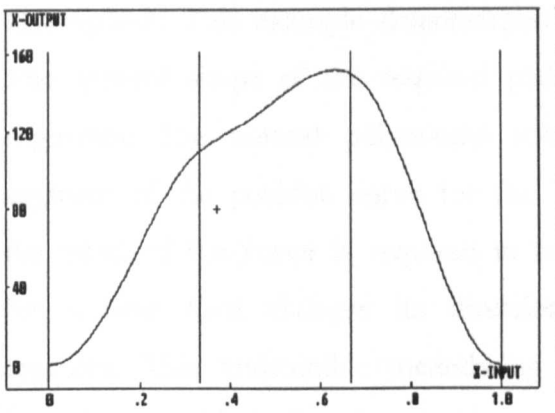


d) Jerk in X-direction

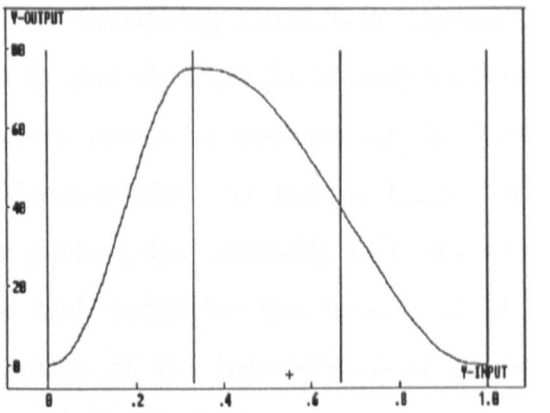


h) Jerk in Y-direction

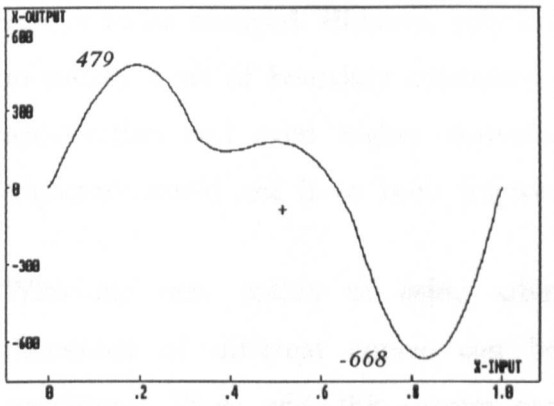
Fig.5.21. Motion curves of the fourth example by five-degree polynomial .



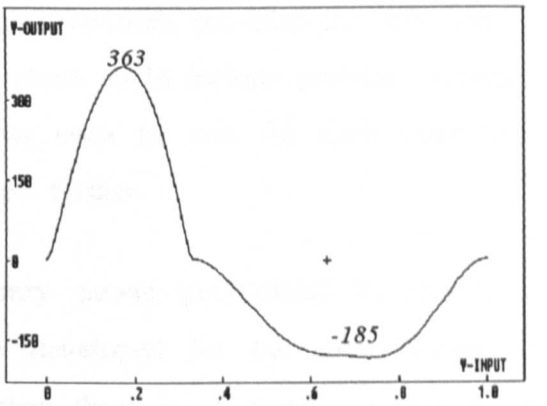
a) Position in X-direction



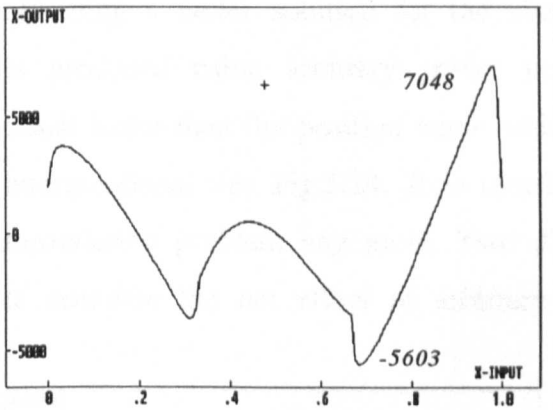
e) Position in Y-direction



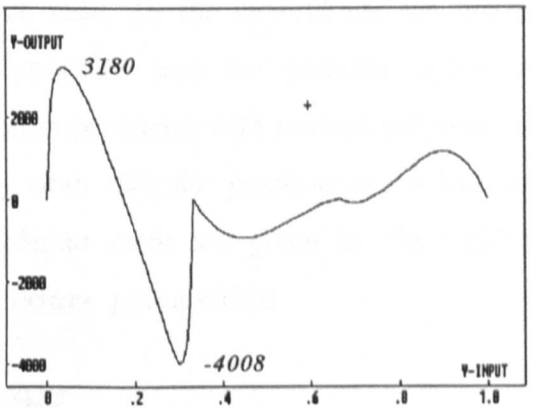
b) Velocity in X-direction



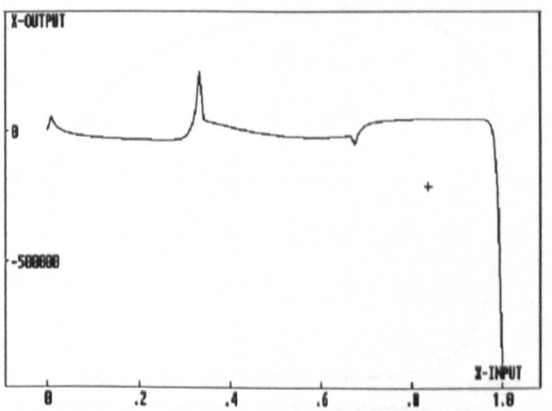
f) Velocity in Y-direction



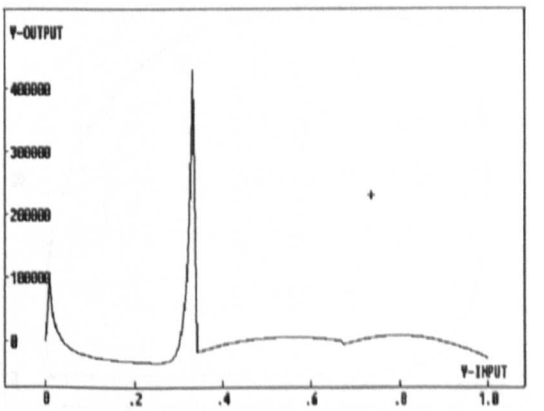
c) Acceleration in X-direction



g) Acceleration in Y-direction



d) Jerk in X-direction



h) Jerk in Y-direction

Fig.5.22. Motion curves of the fourth example by arbitrary power polynomial.

Example-5: This example demonstrates the meandering effect in a trajectory. The general shape of the required path is seen in Fig.5.23 it includes four segments. The normal polynomial solution creates a problem in the first segment of the position curve for the X-axis motion. As seen in Fig.5.25(a) the trend of the curve is required to be positive but contrarily it is negative for a time then changes its direction and completes the motion of the segment. This undesirable meandering action of the trajectory could be an important problem for the designer especially if the boundary conditions cannot to be changed. Hitherto, polynomial functions provided the only means to satisfy a set of boundary conditions which could include position, velocity acceleration and even higher derivatives such as jerk. In such cases the trajectory could not have been improved further.

With the new option of using arbitrary power polynomial interpolation, thousands of different curves can be developed for the same boundary conditions. Thus, with this greater choice, there is an improved chance of obtaining a better solution for the above case. In the second try the curve is produced using arbitrary power polynomials and the position curve is much better than the position curve which is produced with normal polynomial interpolations. See Fig.5.24. It is clearly seen that the position curve has no meandering problem any more. Two different paths are given in Fig.5.25-26 to compare the net effect of arbitrary power polynomials.

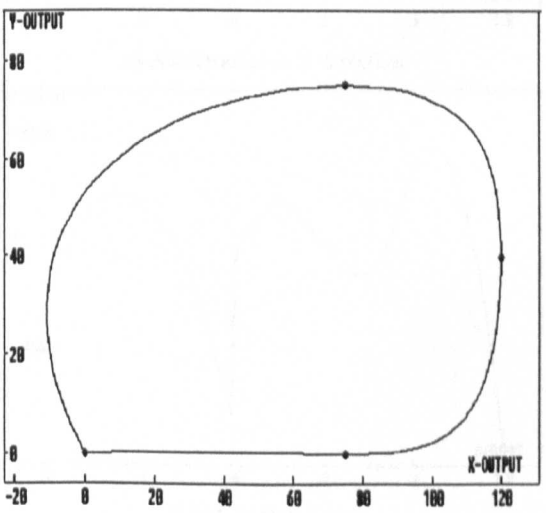


Fig.5.23. Path includes meandering.

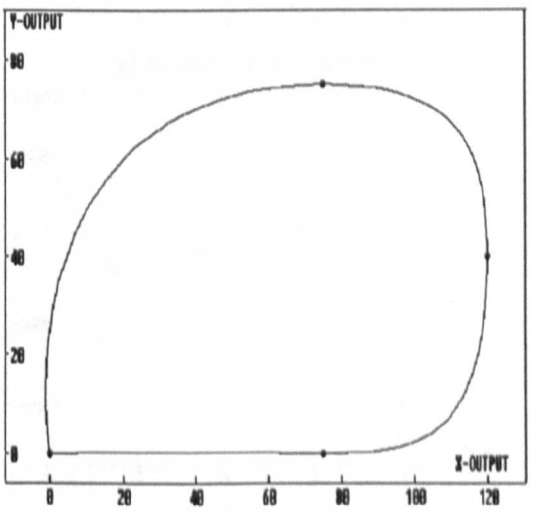
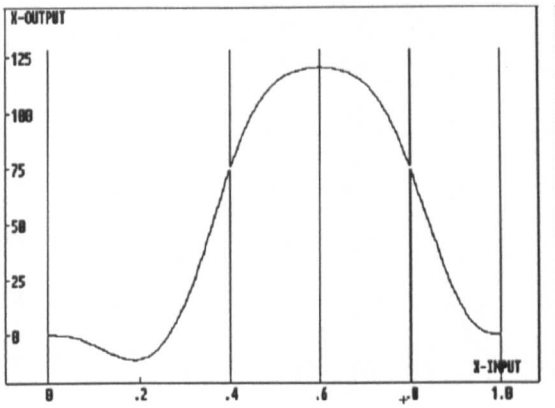
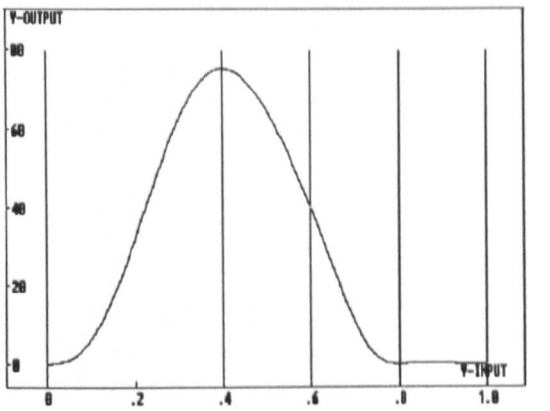


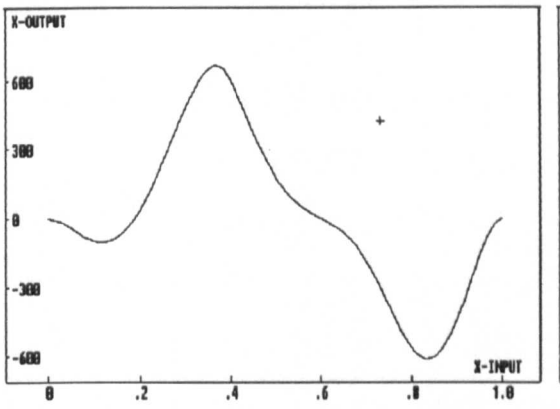
Fig.5.24. Path doesn't include meandering.



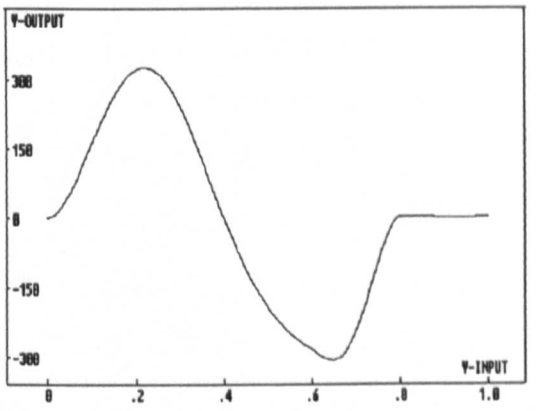
a) Position in X-direction



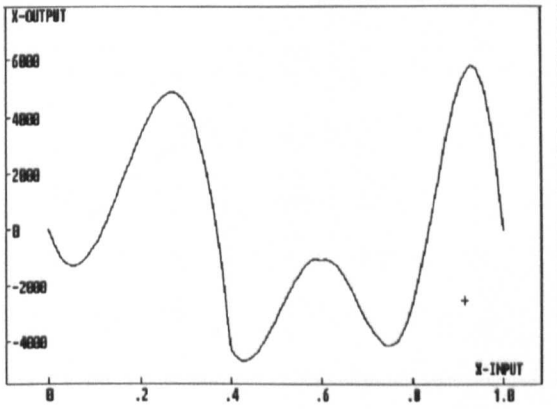
e) Position in Y-direction



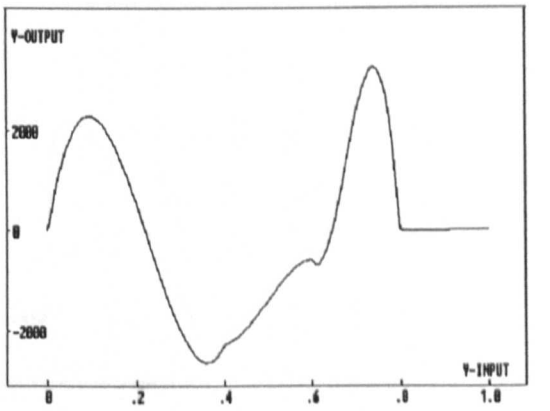
b) Velocity in X-direction



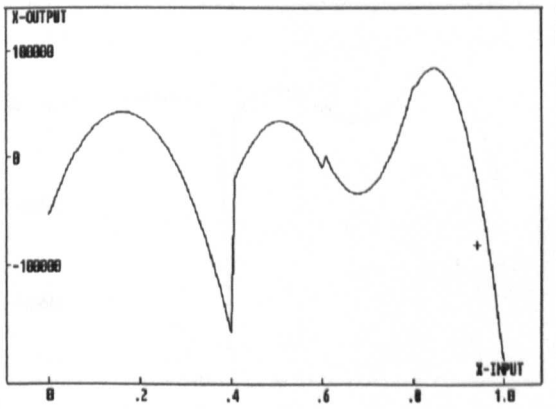
f) Velocity in Y-direction



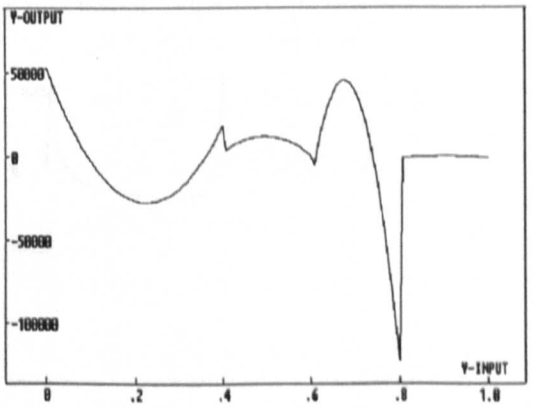
c) Acceleration in X-direction



g) Acceleration in Y-direction

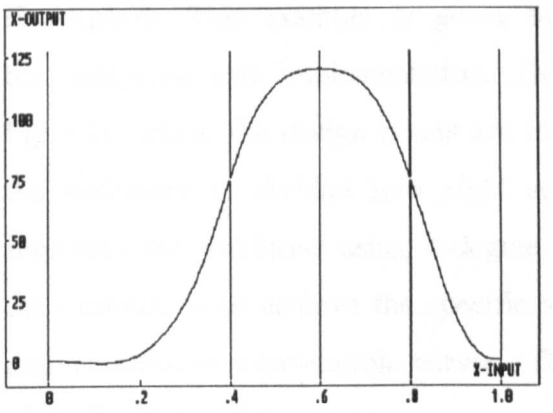


d) Jerk in X-direction

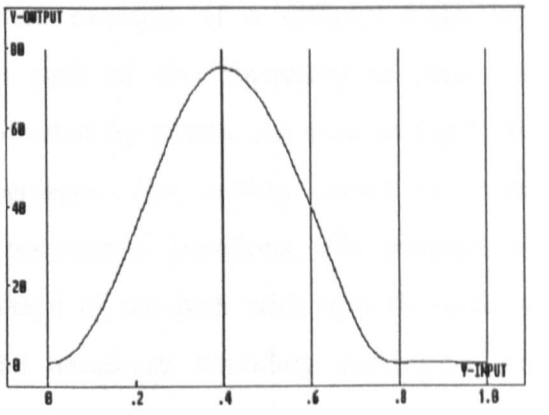


h) Jerk in Y-direction

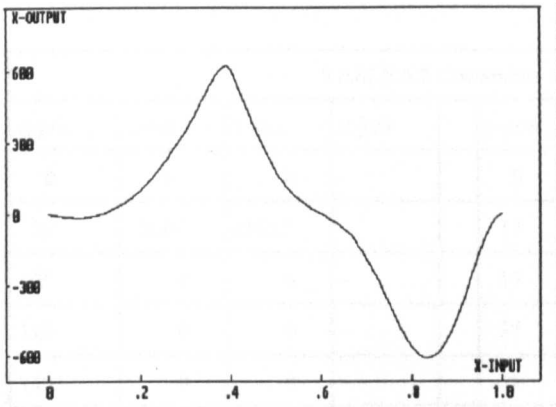
Fig.5.25. Motion curves of the fifth example by normal polynomial.



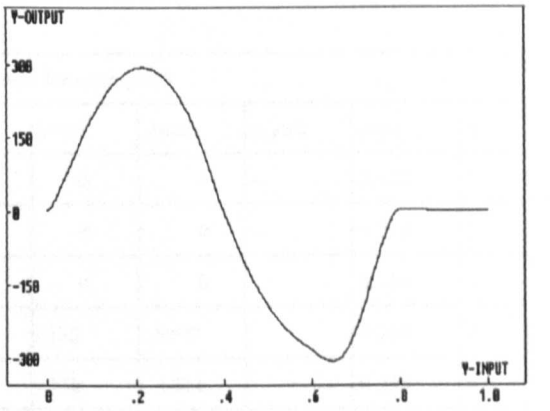
a) Position in X-direction



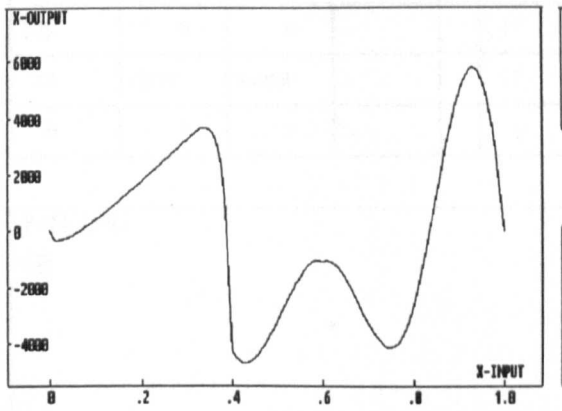
e) Position in Y-direction



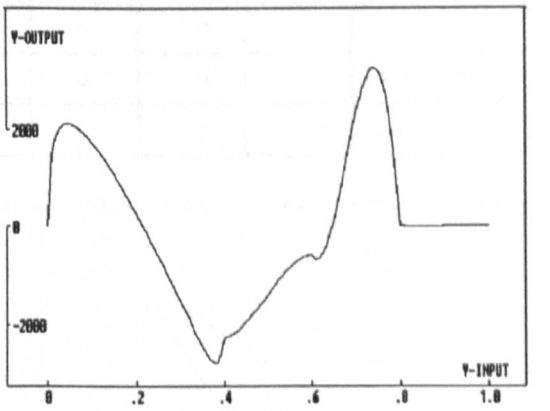
b) Velocity in X-direction



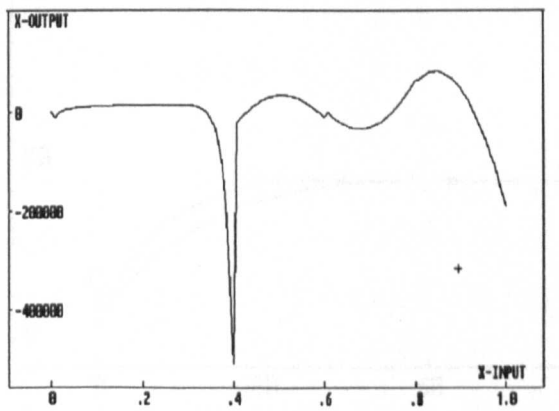
f) Velocity in Y-direction



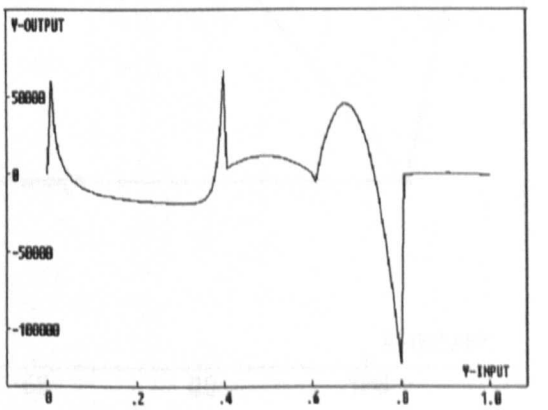
c) Acceleration in X-direction



g) Acceleration in Y-direction



d) Jerk in X-direction



h) Jerk in Y-direction

Fig.5.26. Motion curves of the fifth example with arbitrary power polynomial.

Example-6: This example is given as an example of a difficult trajectory for designing and implementation. The path of this trajectory is given in Fig.5.27, where the design points are indicated by points. As seen in Fig.5.28, the trajectory is divided into eight segments. The motion curves of these segments are produced using 5-degree polynomial functions. The purpose of the example is to achieve the specific shape of the path with smooth velocity and continuous acceleration curves. The necessary boundary conditions are given in Table 5.2.

TABLE 5.2 Constraints used for example 6

x-pos.	x-vel.	x-acc.	x-jerk		y-pos.	y-vel.	y-acc.	y-jerk	time	no
0	0	0	--		0	0	6	--	0.000	1
30	1080	21617	--		17	0	0	--	0.070	2
105	0	0	--		17	0	0	--	0.150	3
110	0	0	--		37	-125	-8600	--	0.220	4
110	0	0	--		35	-130	6021	--	0.235	5
76	-117	0	--		55	0	0	--	0.315	6
105	0	0	--		17	0	0	--	0.410	7
43	-2050	34958	--		17	0	0	--	0.452	8
0	0	0	--		0	0	6	--	0.500	9

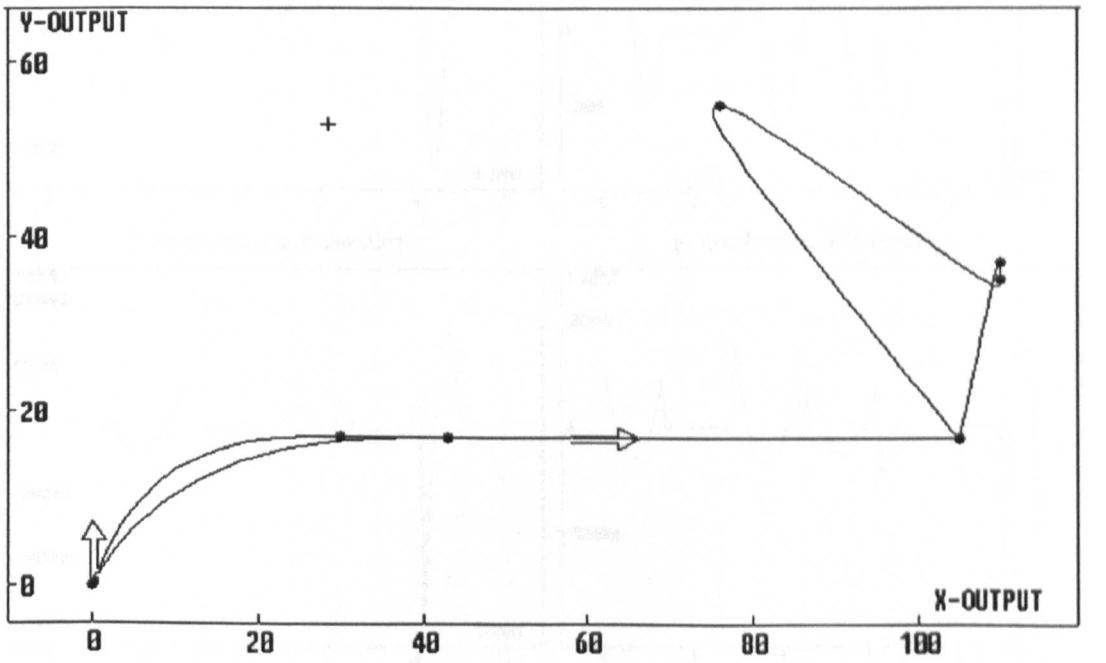
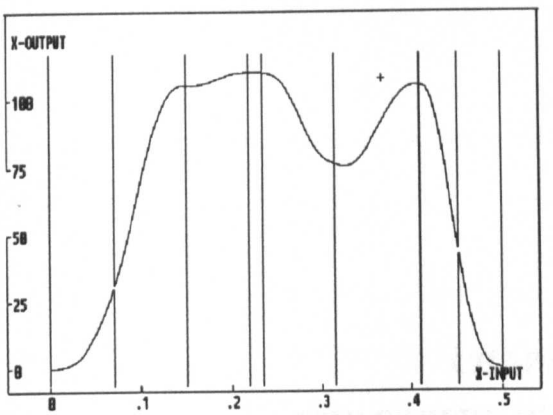
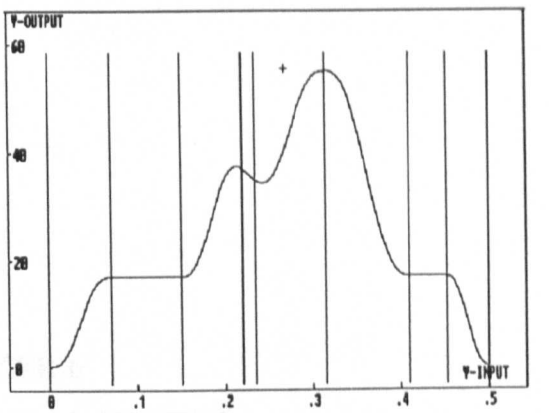


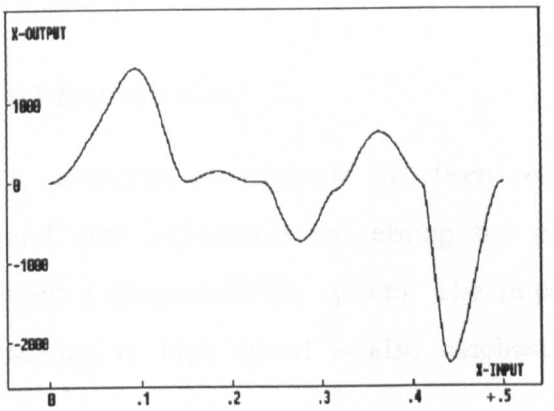
Fig.5.27. The path of the sixth example.



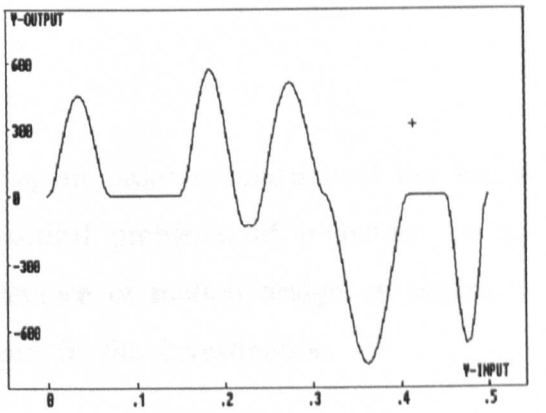
a) Position in X-direction



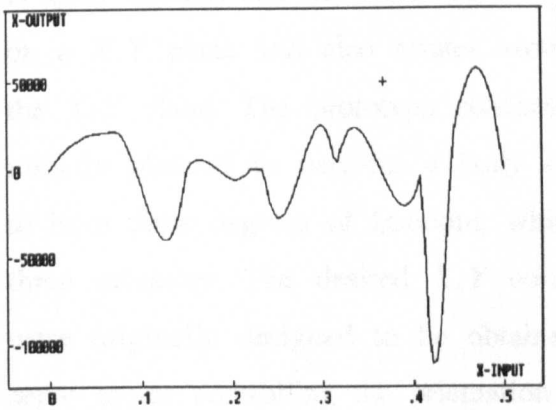
e) Position in Y-direction



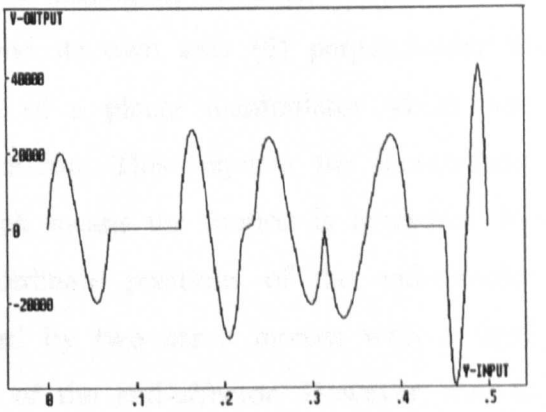
b) Velocity in X-direction



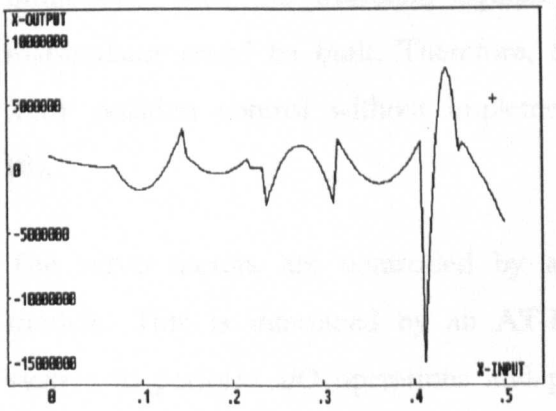
f) Velocity in Y-direction



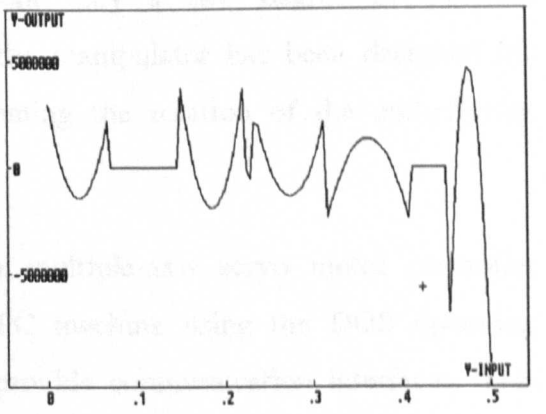
c) Acceleration in X-direction



g) Acceleration in Y-direction



d) Jerk in X-direction



h) Jerk in Y-direction

Fig.5.28. Motion curves of the sixth example.

CHAPTER 6
THE EXPERIMENTAL SYSTEM

6.1 Introduction

A prototype arrangement has been set up in order to understand the issues and gain experience concerning the practical problems of trajectory tracing with a programmable system. The importance of motion design on trajectory tracing at high speed is also emphasized in the investigation.

A body motion can be described as the motion of an object which moves on a X, Y plane and also rotates around its own axis (θ) perpendicular to the X, Y plane. The prototype consists of a planar manipulator which was initially planned to perform a body motion. This requires the manipulator to have three degrees of freedom, which means the motion is controlled by three actuators. The desired X, Y co-ordinate positions of the end-effector were originally designed to be obtained by two servo motors with a third servo motor controlling the orientation of the end-effector. However, due to limitations with the available equipment only a two degree of freedom manipulator could be built. Therefore, the manipulator has been designed for $X\&Y$ position control without implementing the rotation of the end-effector (θ).

The servo motors are controlled by a multiple-axis servo motor controller module. This is interfaced by an AT-PC machine using the DOS operating system to perform I/O operations and provide communication interfaces. This kind of control system is known as a bus-based control system.

The system is a direct drive system which means the motors are coupled directly to the load without the use of belts or gears. This enables high speeds to be achieved and avoids the problems of backlash if geared drives are used.

The outline of a programmable system is discussed in this chapter. The selection of a suitable mechanism for the system, its detailed design, the control circuit used and the kinematic analysis employed form the main subjects.

6.2 Selection of the mechanism

There are numerous kinematic chains which are capable of producing planar mechanisms whose output provides body motion. Udoakang [6.1] identified some of these chains for three degree of freedom mechanisms and formalised criteria to determine the optimum planar mechanism.

A selection of some of the many configurations for a planar mechanism which provides body motion is shown in Fig.6.1. Two of these are particularly suited for prototype development (Fig.6.1(a) and (b)) because of the following reasons:

The two mechanisms are efficient because they include simple chains of links connected by revolute joints. The links can be built from light materials in order to reduce their moment of inertia to a minimum. In general, programmable systems do not have the capacity to regenerate energy because of the non-uniform motion of the actuators. The higher the inertia of the mechanism, the larger the size of actuator required to produce the necessary input energy.

The other mechanisms require linear inputs which means rotational motion has to be converted to linear motion using auxiliary components such as screws or hydraulic elements which increase the cost and inertia of the system.

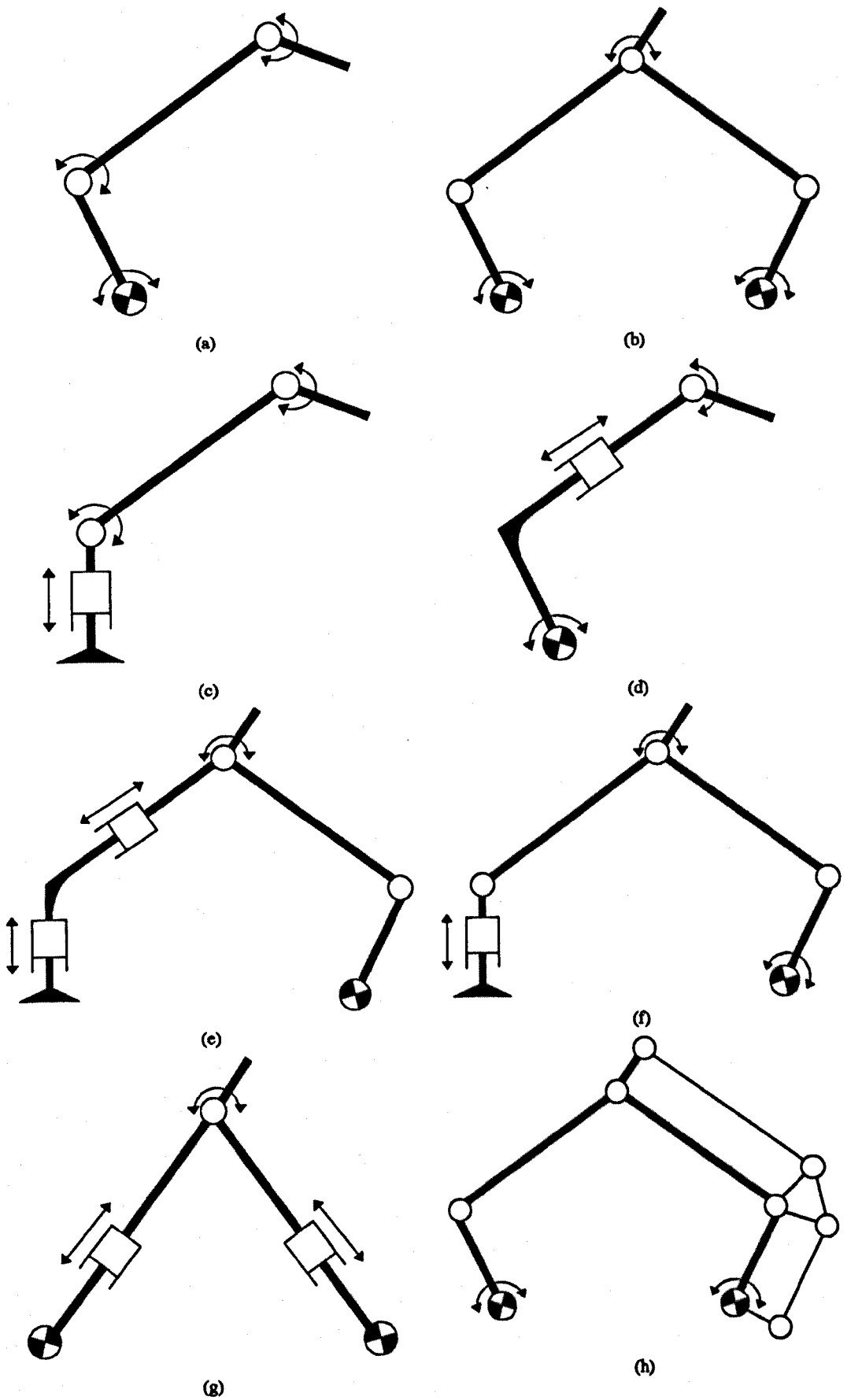


Fig.6.1. Some mechanisms to perform a body motion in a plane.

If we compare the two mechanisms with each other, it can be seen that both of them may have advantages and disadvantages. For example, the five-bar mechanism has the advantage of a stiffer structure because it is pivoted to the ground at two points, whereas, the other mechanism has the advantages of a larger work-space due to its open chain form. In our case, the criteria to select the desirable mechanism are formalized as follows:

(i) the end-effector should be able to move along any X-Y path within the specified work space

(ii) the mechanism has to be stiff to prevent possible oscillations at higher speeds

(iii) the total moment of inertia of the moving elements should be as low as possible because of the following reasons:
(a) to reduce the torque capacities and hence the motor sizes.
(b) to eliminate lower natural frequencies

(iv) the mechanism should be straight forward to manufacture.

After consideration of the above criteria the five-bar mechanism was selected as the most appropriate mechanism for the manipulator.

6.3 Detailed design

A prototype has been designed and built to implement different planar motions in the context of a computer controlled system. An isometric drawing of the manipulator is given in Fig.6.2 where A and B indicate the axes of the servo motor drives and the end-effector point is indicated by C. The distance between the axes is 310 mm. Both cranks and both coupler links have the same lengths which are 100 and 272 mm respectively. The links are made of hollow carbon-fibre material to reduce the mass moment of inertia of the manipulator. This material is very light by comparison with steel or aluminium while still providing quite strong structures for the links. The dimensions of

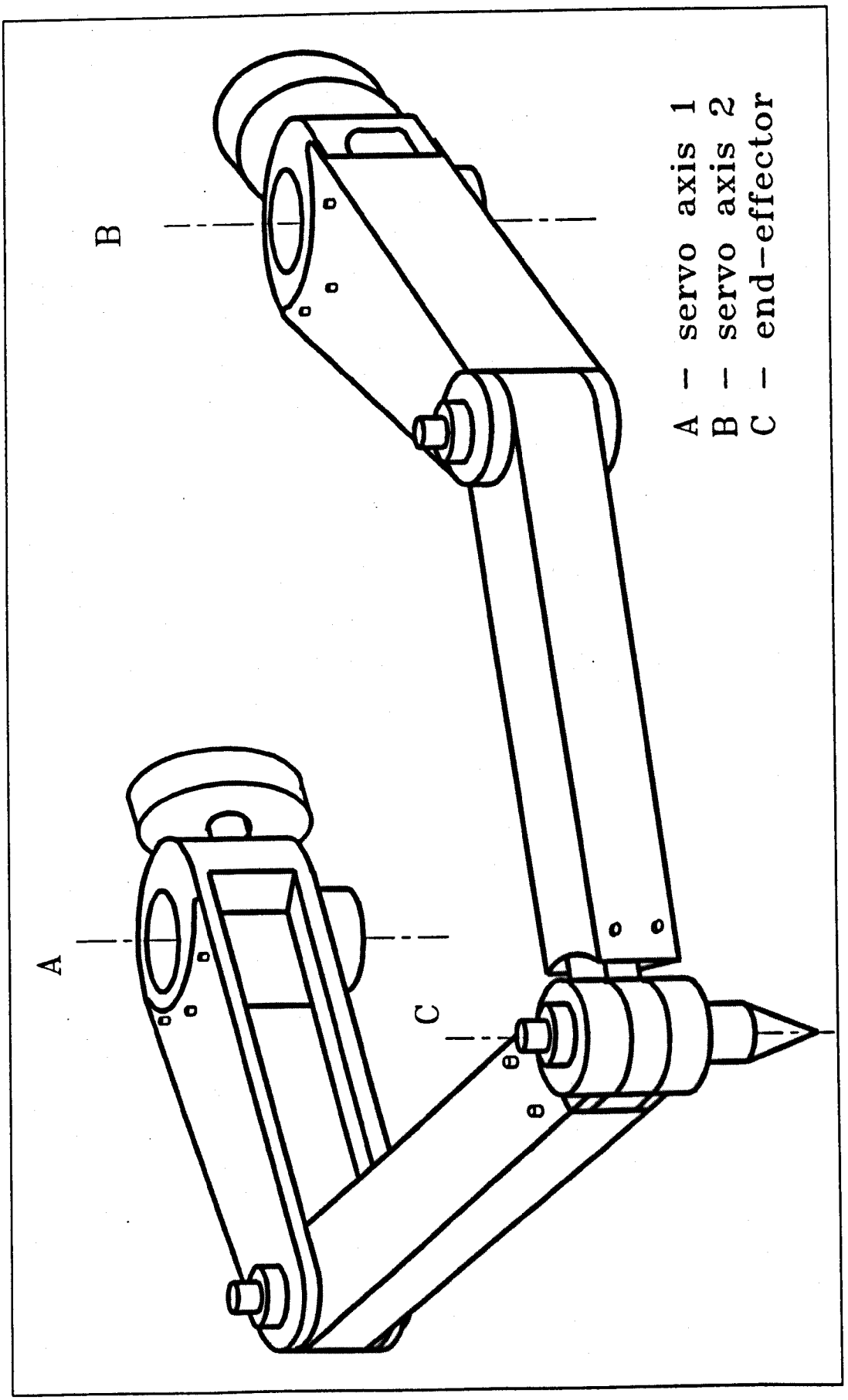


Fig.6.2 Isometric drawing of the arrangement

the links, width, depth and thickness, are large enough to prevent any oscillation at the end-effector point. The technical drawings together with the names of, and the material used for, the parts of the manipulator and photographs of the arrangement are given in Appendix A.

Two aluminium bushes are used to fix the cranks to the shafts of the motors. One end of each coupler link is pivoted to the two cranks using journal bearings. The details of the pivots can be seen in the assembly drawing (see Appendix A, Fig.1). The free ends of the coupler links are pivoted to each other using a double lap joint in order to close the manipulator. To obtain the trace of the manipulator a pen holder is placed at the centre of this pivot as shown in Fig.6.2.

The coupler links were initially pivoted to the cranks as shown below (see Fig.6.3). This configuration of the pivots can prevent crashing of the manipulator, in the case of failure of the control software, by allowing full rotation of the cranks to take place.

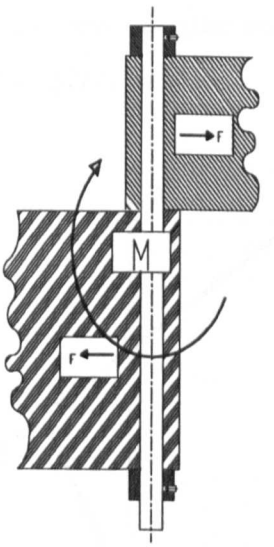


Fig.6.3. An unbalanced pivot

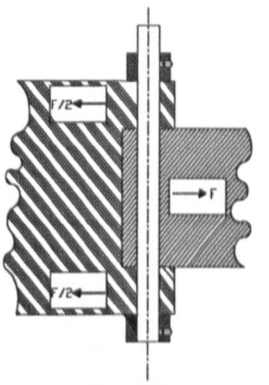


Fig.6.4. A balanced pivot

However, such a configuration creates unbalanced moments on the manipulator which can result in vibrations particularly at higher speeds. In order to maximize the speed of the system the configuration of the pivots was later rearranged to the balanced form shown in Fig.6.4.

6.4 Balancing of the system

Unbalanced masses on a mechanism create shaking forces and moments on the drive bearings during the movement of the mechanism. The resulting vibration will lead to an inaccurate tracing of the path of the end-effector. For a proper control of the system, the shaking forces and moments on the frames should be eliminated.

In general, a five-bar mechanism may be balanced as follows:

(i) First of all, the coupler links can be balanced by bringing their centres of gravity to the crank-coupler joint centres by adding counterbalance weights.

(ii) Similarly, the cranks can be balanced by bringing the centre of the gravity of the cranks (including the weights of the coupler links at the joints) to the pivot centres.

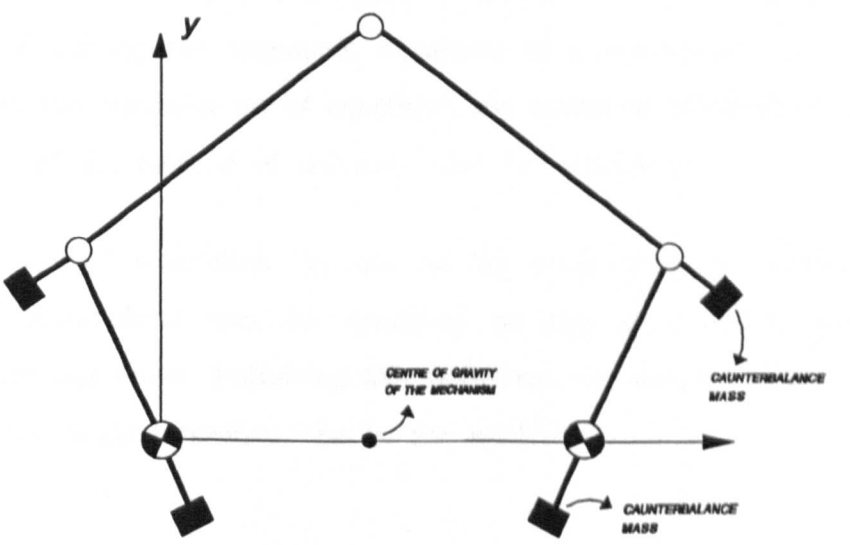


Fig.6.5. Balancing of a five-bar mechanism

In others word, the mechanism can be balanced by bringing the centre of gravity of the entire mechanism to the mid-point of the centre line joining the two frame pivots as shown in Fig.6.5. (The centre of the gravity must be at the same point for every position of the joint angles)

These two steps assure the balancing of the mechanism in the X-Y plane, however, the mechanism may have unbalanced couples in the X-Z plane or Y-Z plane. In order to prevent unbalanced couples acting on the joints of the mechanism the link ends must be interleaved so that the mass centres of each link lies on its longitudinal axis and the bearings are placed at equal distances about the coupler axes (see Fig.6.4).

The geometry of the carbon fibre links and the configuration of the joints between the coupler links and the cranks do not allow counterbalance weights for the couplers to be attached to these links. Therefore the manipulator is partially balanced by attaching weights to the cranks only. This situation may disturb the stability of the manipulator, but the very light material used for the construction of the links may compensate for this imbalance.

6.5 Kinematic analysis

The joint angles of the manipulator are required to be found for given numerical values of the end-effector coordinates in a Cartesian plane. The problem of solving the kinematic equations of a manipulator is nonlinear. Because of the nonlinear set of equations, the existence of solutions, multiple solutions and the method of solution must be considered.

The existence of a solution depends on the work-space. A work-space for a planar manipulator can be described as the area within which the end-effector can move. Following this definition, we can say that a solution exists if the desired positions lie in the work-space.

Another possible problem in solving kinematic equations is the existence of multiple solutions. Fig.6.6 show four possible modes for the same mechanism which satisfy the same desired point in a Cartesian plane but each with different joint angles.

Multiple solutions may cause problems because the system has to be able to choose one of them. Relevant criteria may vary according to the situation but a very reasonable choice would be the closest solution to the previous joint angles. Another method would be to determine all the possible modes of the manipulator and to choose one of them to define the working area for the manipulator. The motion of the manipulator can then be restricted to lie within this working area of the selected mode thereby precluding other multiple solutions.

To achieve this end, the boundaries of the mode can be specified by corresponding maximum and minimum values for the joint angles. As seen from Fig.6.6, the permissible joint angles vary with respect to the position of the end-effector, however, the important criterion is that the cranks and their coupler links must not pass through a collinear position for any set of joint angles. More specifically, (see Fig.6.7), the angles (ABP) and (PCD) must be less than or equal to 180 degrees.

In this application, the mode of the manipulator is selected as that shown in Fig.6.7 because the manipulator with this configuration normally has better transmission angles that those for the other modes.

Mathematical relationship: The arrangement of the five-bar manipulator is shown in Fig.6.7. The angular positions of both cranks ($\theta 1$, $\theta 2$) can be calculated in terms of the position of the end-effector **P** and the link lengths of the manipulator ($a0$, $a1$, $a2$, $a3$ $a4$). The manipulator is divided into two parts to simplify the analysis and each part will be handled independently. The crank $a1$ and coupler link $a2$ and the point P determine the first half of the manipulator. Similarly the other half is determined by the position of P, the second coupler link $a3$ and the second crank $a4$.

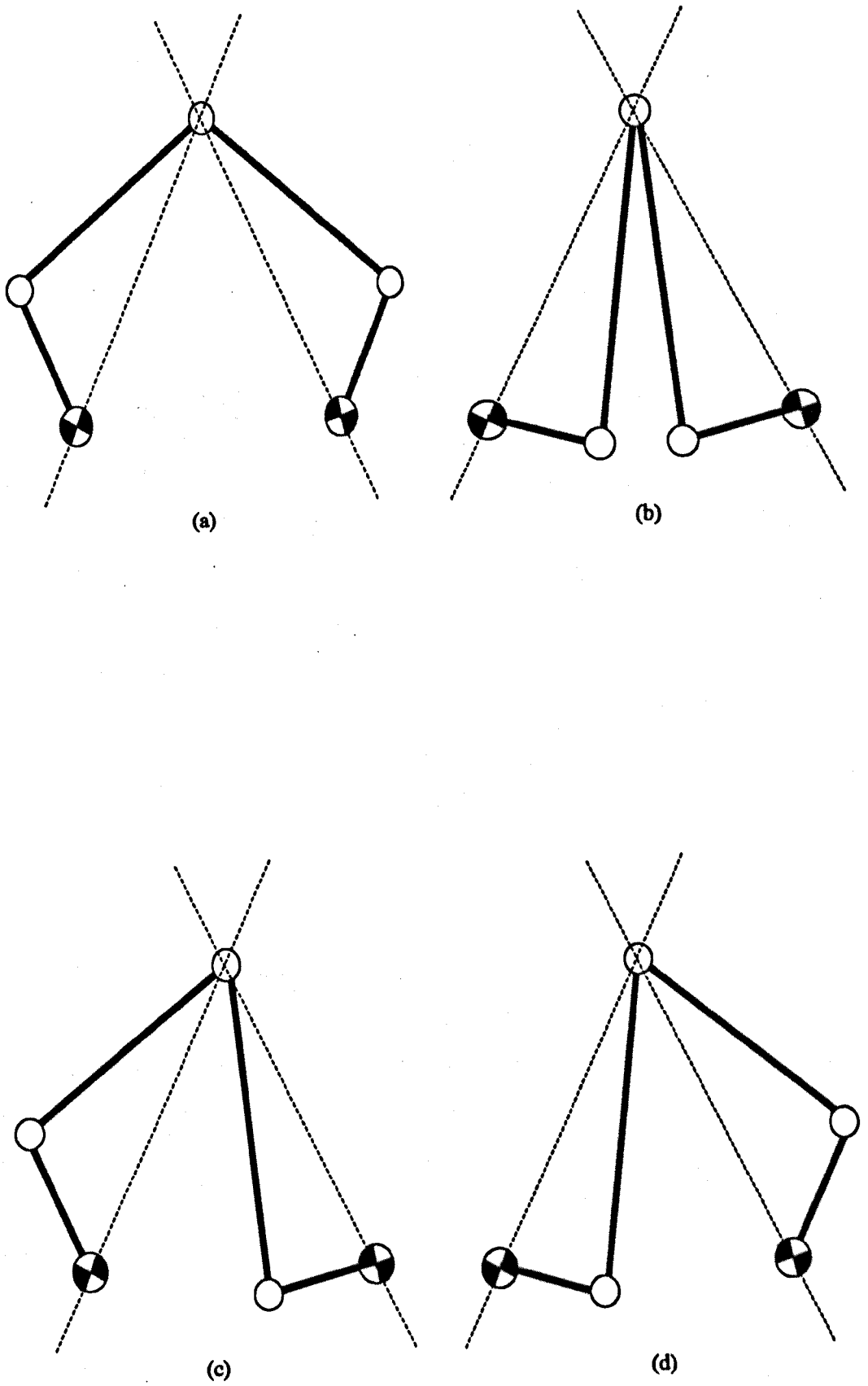


Fig.6.6. Four possible configurations for a five-bar manipulator

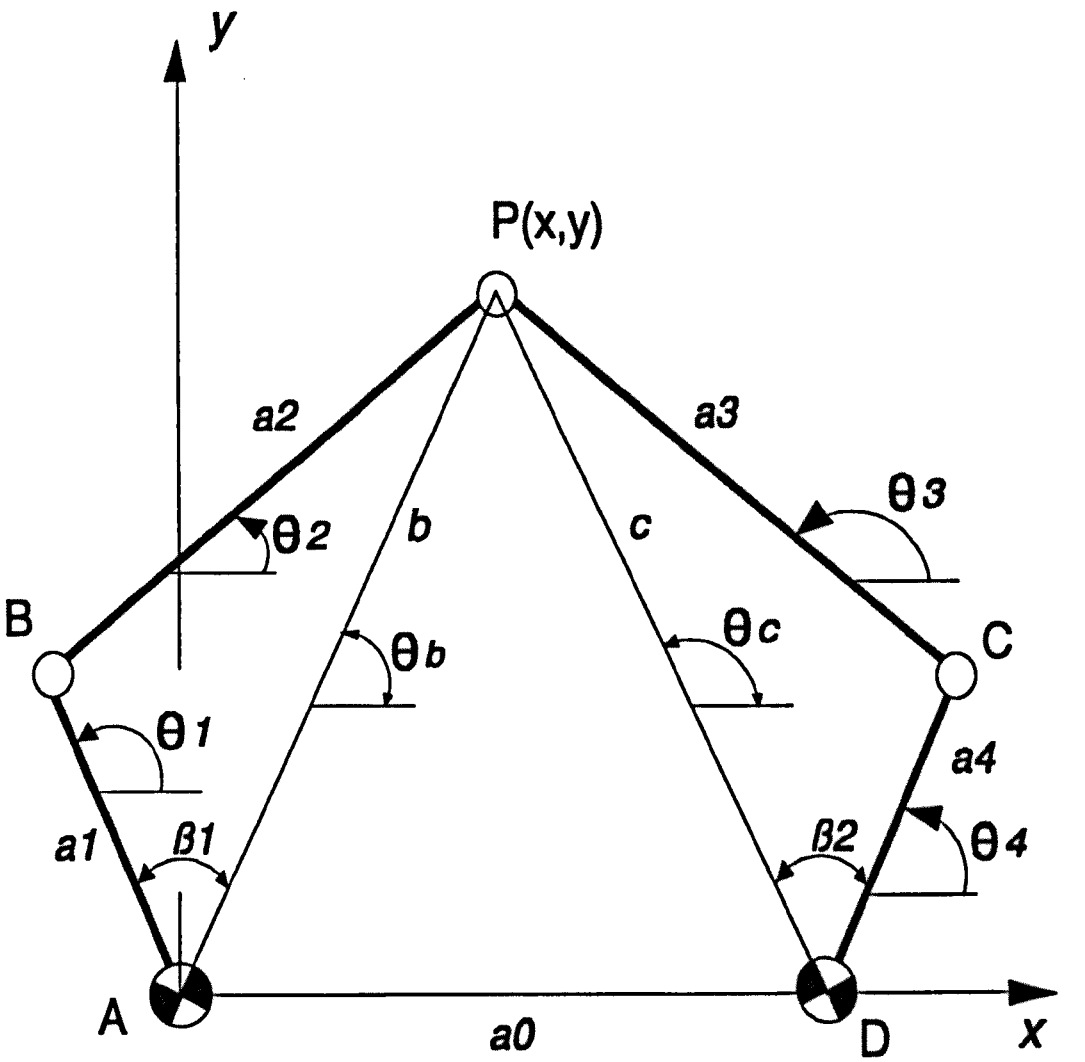


Fig.6.7. Detailed five-bar manipulator

There is no general algorithm which may be employed to obtain an inverse solution for a manipulator, however, the angular position of the cranks in this case can be calculated using geometric relationships of the links of the manipulator. The evaluation of equations for the joints angles for a five-bar manipulator with reference to Fig.6.7 is given below:

$$b = (x^2 + y^2)^{1/2} \tag{6.1}$$

$$\theta b = \tan^{-1}\left(\frac{y}{x}\right) \tag{6.2}$$

From the Cosine theorem applied to triangle ABP

$$a2^2 = a1^2 + b^2 - 2*a1*b*\text{Cos}(\beta 1) \tag{6.3}$$

$$\beta 1 = \text{Cos}^{-1}\left(\frac{a2^2 - a1^2 - b^2}{-2*a1*b}\right) = \text{Cos}^{-1}\left(\frac{b^2 + a1^2 - a2^2}{2*a1*b}\right) \tag{6.4}$$

$$\theta 1 = \beta 1 + \theta b \tag{6.5}$$

$\theta 1$ is valid if angle $(ABP) \leq 180$ degree.

Where $\theta 1$ is the angular position of the first joint.

Using a similar relationship we can calculate the angular position of the second joint.

$$c = ((a0 - x)^2 + y^2)^{1/2} \tag{6.6}$$

$$\theta c = 180 - \tan^{-1}\left(\frac{Py}{a0 - Px}\right) \tag{6.7}$$

Considering the triangle (DCP);

$$a3^2 = c^2 + a4^2 - 2*c*a4*\text{Cos}(\beta 2) \tag{6.8}$$

$$\beta 2 = \text{Cos}^{-1}\left(\frac{a3^2 - c^2 - a4^2}{-2*c*a4}\right) = \text{Cos}^{-1}\left(\frac{c^2 + a4^2 - a3^2}{2*c*a4}\right) \tag{6.9}$$

$$\theta 4 = \theta c - \beta 2 \tag{6.10}$$

$\theta 4$ is valid if angle $(DCP) \leq 180$ degree.

Where $\theta 4$ is the angular position of the second joint.

6.6 Control circuit

A typical motion control system can be represented in block diagram form as shown in Fig.6.8. In such systems, the control algorithm can be divided into two levels. The first level of control involves profile selection, the generation of the motion commands and the co-ordination of the motion with overall process control. The next level is the trajectory control. The motion controller closes the position loop around the servo motors by sensing the positions of the motor shafts with incremental encoders. The positions are decoded and compared with the command positions, the difference forming the position errors. The error signals are processed by a stabilizing filter. The outputs of the filter are then amplified by the servo drivers before they are applied to the motors.

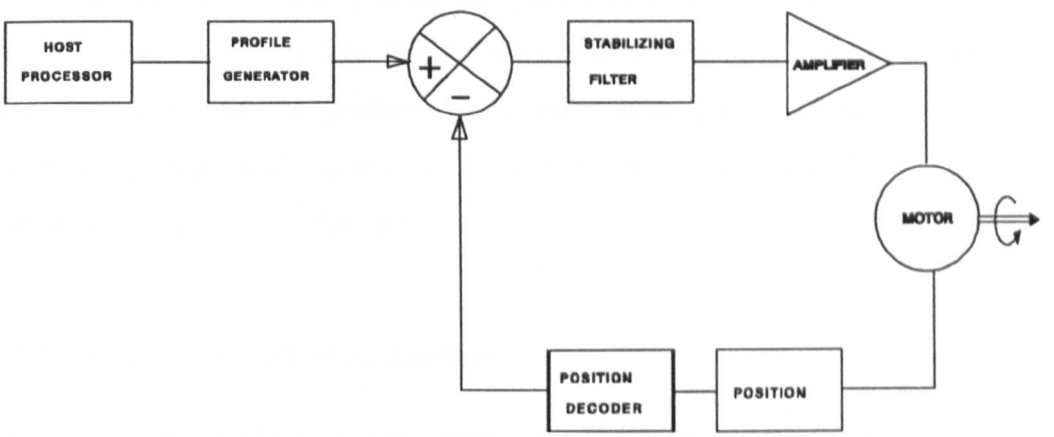


Fig.6.8. Block diagram for a motion control system

The control circuit of the arrangement used in this investigation consists of the following components:

(i) a PC-AT compatible 80386 16-bit 20 Mhz computer (MITAC)

(ii) Two 'Electro Craft S-series' permanent magnet synchronous motors which have integral mounted optical encoders

(iii) two servo digital drivers (BRU-500) whose control circuitry utilizes a 16-bit microprocessor

(iv) a servo motor controller module (Warwick Computer Designs).

In the following sections, some of basic components which constitute the hardware of a general servo system are described. The main part of the system is the controller which converts the digital information for each point of a trajectory into an analog signal. These signals are in the form of voltages which are sent to the servo drives as pulses which are transmitted at regular intervals in order to control them. The job of the servo drive is to draw electrical energy from the mains (at constant voltage and frequency) and supply electrical energy to each motor at whatever voltage and frequency is necessary to achieve a desired motor output. A servo motor is actually a dc-motor but with the inertia of the rotating parts minimized to provide for rapid changes in acceleration. The feedback element is the other important part of a servo system and this is usually attached to the servo motor. The controller can read the position or velocity from the feedback element and modify the command signal to prevent an error between desired position and actual position or the rotor.

6.6.1 Servo motor controller module

The controller module is the most functional part of a servo system. It converts a desired set of data, usually the positions or velocities of a trajectory, to analog signals mostly in the range of (-10,+10) volts. The outputs from the controller module are used as command signals for the servo motors after amplification by the amplifier.

Modern controller modules generally interface with a computer. They can be inserted into slots available on most computer expansion boards. There are numerous controller modules available commercially most employing

proportional-derivative (PD) or proportinal-integral-derivative (PID) control methods. In the following sections the controller module used for the experimental system is described.

The controller card is a general purpose motor controller capable of controlling up to three axes. It provides position and velocity control for dc, dc-brushless or stepper motors. The system block diagram and internal block diagram of the controller card are given in Fig.6.9 and Fig.6.10 respectively. The controller receives the input command from the host processor and position feedback from an incremental encoder with quadrature output. The controller then compares the desired position to the actual position and computes compensated motor commands using a programmable digital filter which continues trying to achieve the desired position until a new command is received from the host processor.

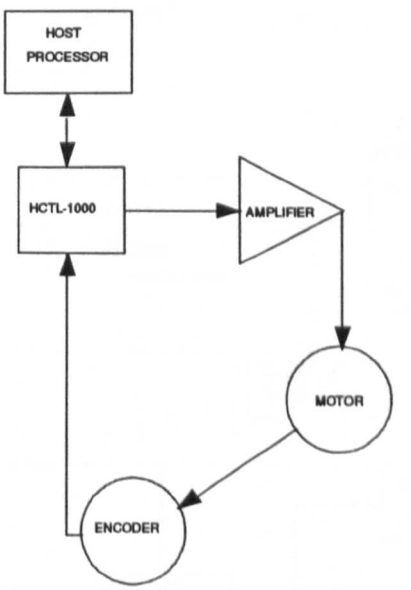


Fig.6.9. System block diagram of HCTL-1000 (Warwick board).

a) I/O interface

The controller card assumes that the host processor is a bank of 8-bit registers and the values of these registers can be changed by the control software at any time. The data in these registers controls the operation of

the card. The host processor communicates to the controller over a bidirectional multiplexed 8-bit data bus. Four I/O control lines execute the data transfer.

It is important that the host computer does not attempt to perform too many I/O operations in a single sampling period of the controller. Each I/O operation interrupts the execution of the controller card's internal code for one clock cycle. However, for every unit increase in the sample time register above the minimum limit the user may perform 16 additional I/O operations per sample time.

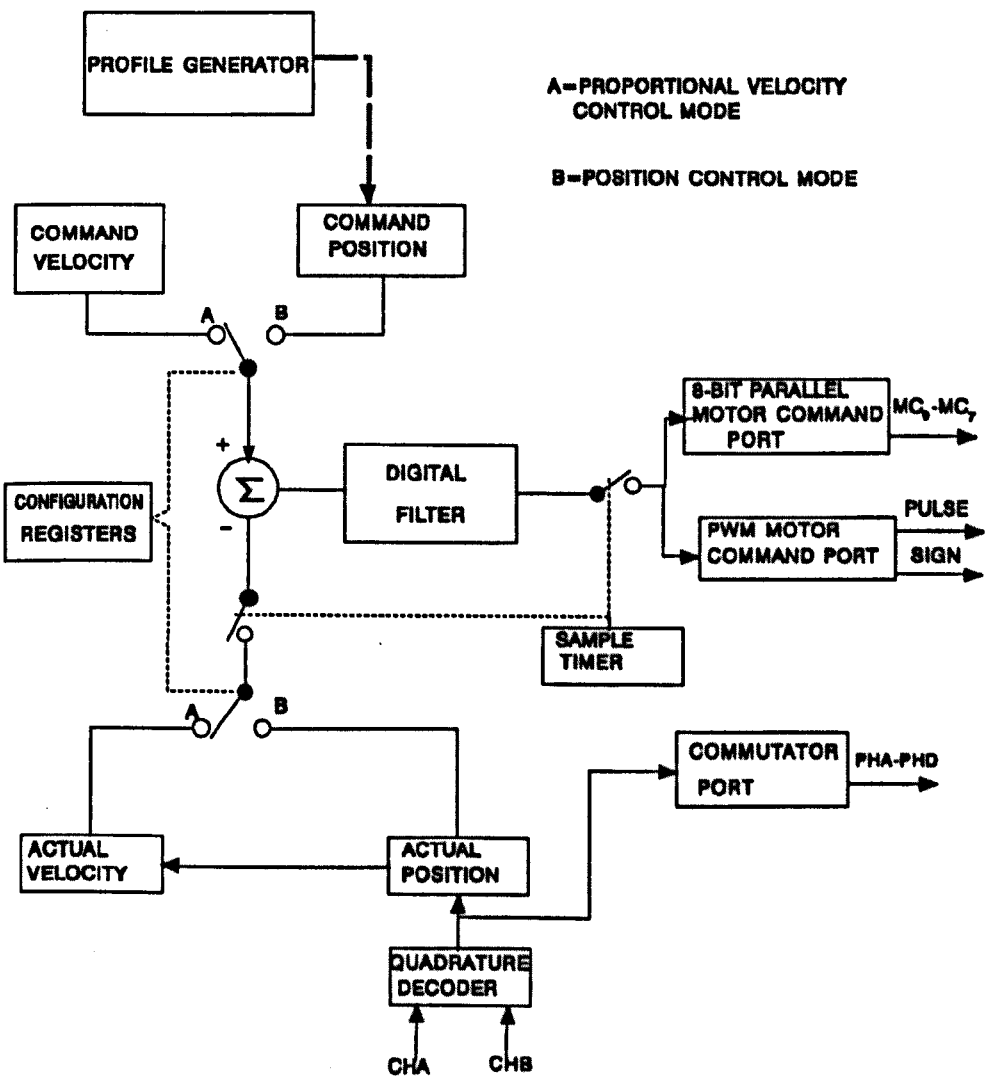


Fig.6.10. HCTL-1000 controller's block diagram.

b) Encoder interface

The controller card accepts TTL compatible outputs from 2-channel incremental encoders. Channels A and B are internally decoded into quadrature counts which increment or decrement the 24-bit position counter. For example, a 2000-count encoder is decoded into 8000-quadrature counts per revolution. The position counter will be incremented when channel B leads channel A. The controller card employs an internal 3-bit state delay filter to remove any noise spikes from the encoder inputs to remain stable for three consecutive clock raising edges for an encoder pulse to be considered valid by the controller's actual position counter. The user should therefore generally avoid creating encoder pulses of less than 3 clock cycles.

c) Digital filter

The controller card is not a PID controller, however, it uses a digital filter $D(z)$ to compensate for closed loop system stability. The compensation $D(z)$ has the form:

$$D(z) = \frac{K^*\left(z - \frac{A}{256}\right)}{4^*\left(z + \frac{B}{256}\right)}$$

where

z = the digital domain operator
K = digital filter gain
A = digital filter zero
B = digital filter pole

The compensation is a first-order lead filter which in combination with sample timer T affects the dynamic step response and stability of the control system. The sample timer T determines the rate at which the control algorithm is executed. All parameters A, B, K and T are 8-bit scalars that can be changed by the user at any time. The implementation of the digital filter in the time domain is according to the rule below:

$$MC_n = \left(\frac{K}{4}\right)(X_n) - \left[\left(\frac{A}{256}\right)\left(\frac{K}{4}\right)(X_{n-1}) + \left(\frac{B}{256}\right)(MC_{n-1})\right]$$

where

n	= current sample time
$n-1$	= previous sample time
MC_n	= Motor command output at n
MC_{n-1}	= Motor command output at $n-1$
X_n	= Actual command position at n
X_{n-1}	= Actual command position at $n-1$

d) Sampling period

The content of this register sets the sampling period of the controller. The sampling period is:

$$t = 16(T+1)\left(\frac{1}{frequency\ \ of\ \ the\ \ external\ \ clock}\right)$$

where T=register value

There are two options for the frequency of the external clock. Its jumpers can be set either to 1-MHz or 2-MHz. The sample timer has a limit on the minimum allowable sample time depending on the control mode being executed. The limits are given below:

(i)	Position	Control (T=7)
(ii)	Proportional	Control (T=7)
(iii)	Trapezoidal Profile	Control (T=15)
(iv)	Integral Control	Mode (T=15)

The maximum value of T is (256). With a 2-Mhz clock, the sample time can vary from 64 micro seconds to 2048 micro seconds. With 1-Mhz clock the sample time can vary 128 micro seconds to 4096 micro seconds.

Digital closed-loop systems with slow sampling times have lower stability and lower bandwidth than similar systems with faster sampling times. To keep the system stability and bandwidth as high as possible the controller should be programmed with the fastest sampling time possible.

6.6.2 Dc-motors

There are many different types of dc-motors which may be used to provide motion for a programmable system, however, most of the basic types are described below.

a) stepper motors

Stepper motors can be controlled by a microprocessor or programmable controller. Their unique feature is that the output shaft rotates in a series of discrete angular intervals or steps, one step being taken each time a command pulse is received. When a definite number of pulses has been supplied the shaft will have turned through a known angle and this makes the motor ideally suited for open-loop position control.

Stepper motors have the benefits of low cost, simplicity in construction and high reliability. They are simple to drive and control since they do not have feedback components and therefore work with an open-loop configuration. They can provide high torque at low speeds (4-5 times the continuous torque of a brush servo motor of the same size). There are different types of stepper motors but the most widely used type is the permanent magnet motor.

The main drawbacks of stepper motors include resonance effects and relatively long settling times, rough performance at low speed and positional errors as a result of the open-loop system. The resolution of the rotor angle is normally in the range 1.8-90 degrees which is low in comparison to the resolution of servo motor encoders (1-0.0001 degrees).

b) Dc-servo motors

The structures of servo motors are similar to conventional dc-motors, however, servo motors are more suitable for use in applications which require rapid acceleration and deceleration. Servo motors are usually driven under the control of dc-drives that provide varying voltages for the motors, thereby allowing the motor position and speed to be controlled. Servo systems comprise closed-loop systems since they incorporate feedback components. In order to achieve exact position and speed control the feedback information can be used to tune the gains of the system.

c) Dc-brushless servo motors

These type of servo motors behave in a similar way to dc-brush servo motors, but they have additional features. In the brushless motor, the construction of the iron cored motor is turned inside out, so that the rotor becomes a permanent magnet and the stator becomes a wound iron core. The other advantages of the brushless motor is the elimination of conventional commutator which is source of wear and frequent maintenance. Additionally, they usually incorporate an integral high resolution position sensor. They also give high torque-to-inertia ratios and hence provide high acceleration capabilities.

d) Hybrid servo motors

These motors have the advantages of both servo and stepper motors. They work in a closed-loop system since they have feedback components and can produce high torques because they are stepper motors. Also, hybrid motors have other advantages of stepper motors, for example, they are reliable, cheap and easy to control. Hybrid servo motors can be driven in precisely the same fashion as servo motors but with low resolution due to the step angles involved.

6.6.3 Servo drives

Amplifiers for both brush and brushless servo motors are either analog or digital. Analog drives have been used for many years but digital drives are a relatively recent innovation.

In the traditional analog drive, the desired motor position or velocity is represented by an analog input voltage usually in the range of $\pm$ 10 volts.

Full forward velocity is represented by +10 volts, and full reverse by -10 volts. Intermediate voltages represent velocities in proportion to the voltages. Various adjustments needed to tune an analog drive are usually made by means of potentiometers.

The digital controlled drives are the alternative to the analog drives. These drives work in a similar way to analog drives, but they are easier to use and can be tuned by sending data from a terminal or computer.

Servo drives may operate with different modes of control and each control mode has a different methodology. The control modes and their purposes are described briefly below:

a) Current control

The purpose of the current controller is to make the actual motor current follow the current reference signal, therefore, the motor current will be always under control.

b) Torque control

For some applications the motor may be required to operate under specific torque values regardless of speed and position. For example, in a wire drawing operation the diameter of the wire is inversely proportional to the applied torque developed by the driving motor. A high torque value is

necessary for small diameter wires, whereas large diameter wires require a low torque value. However, the important requirement is to keep the torque value constant in order to maintain a uniform wire diameter.

c) Speed control

The system operates under the speed feedback provided by a tachogenerator. In this mode of control the difference between the actual and desired speed is amplified, for example, if the actual motor speed is less than the desired speed, the speed amplifier will demand current in proportion to the speed error, and the motor will therefore accelerate in an attempt to minimize the speed error.

6.6.4 Feedback sensors

Servo motor driven systems use different types of feedback sensors to close the loop and determine the joint position or velocity. These feedback sensors can be classified into groups according to the type of output signal.

a) Positional feedback sensors

There are three primary positional feedback devices which are the potentiometer, the resolver and the optical encoder. The potentiometer is a variable resistor that provides an output voltage proportional to the angular position of the shaft. The resolver uses magnetic coupling between transformers to measure rotation. Optical encoders comprise absolute and incremental types. In both types a beam of light is interrupted by radial slots on a rotating opaque disc to determine position.

b) Velocity feedback sensors

The tachometer is a device whose output voltage is proportional to motor angular velocity. This can be used in a feedback loop to provide velocity control.

CHAPTER 7
TRAJECTORY TRACING

7.1 Introduction

The second stage of the control scheme for programmable systems is *trajectory tracing* for which some control techniques are applied in order to obtain acceptable results from a system. When a desired signal is applied to a servo-system it responds in a characteristic fashion. The physical features of the actuators and the gain setting of the controller are the main parameters that determine the response of the system. The optimum gain values can be determined by developing a mathematical model of the system or by applying an appropriate experimental method. Controllers with fixed gain values are effective for many conventional processes such as pick and place tasks using slow speed manipulators. However, there are several cases where precise tracing of a fast trajectory under different payloads requires more advanced control techniques to ensure stability of the process where fixed parameter control is completely inadequate. These cases require continuous tuning of the controller [7.1].

Adaptive and learning control are two important methods of advanced control. Self-tuning and model reference adaptive control are two different techniques in adaptive control. In both approaches, the main idea is based on continuous tuning of the gain values of the controller using a dynamic model of the

system. However, learning control is based on tuning the input for the system using the experience of past responses of the system when executing repetitive tasks.

In this chapter, the determination of initial gain values of a controller for a closed loop system is explained. Different adaptive learning techniques to minimize the error between command and response are examined and discussed. In particular, the development of a new approach for learning is presented in detail. It has been implemented on-line for the improvement of trajectory tracing for several examples and quite satisfactory results have been obtained.

The main subject of the work, trajectory planning, is shown to have an important bearing on trajectory tracing. It is shown experimentally that the prototype gives better responses for a trajectory which is planned carefully with high order smoothness, low peak velocity and low peak acceleration than a trajectory planned more conventionally.

7.2 Control techniques

Advances in microprocessor technology have made digital control more attractive then before. Digital control methods offer many advantages compared with analogue control. They allow complicated control laws to be implemented and the resulting system performance to be much more accurate [7.2].

The dynamics of a manipulator can be characterised by a set of highly coupled nonlinear differential equations. Based on such dynamic models, the control of manipulators has been extensively studied in recent years and two control design approaches have been proposed, non-adaptive and adaptive control [7.3]. Non-adaptive control is based on an exact knowledge of the complex system dynamic equations. In other words, if the dynamics of the process and the characteristics of the disturbances affecting it are known, then the controller which will yield the desired performance can be designed. However, this model usually needs more complicated control structures, thus

incurring higher costs when put to practical use. Many parameters effect the accuracy of the dynamic model such as link inertias, mass centres, friction and air resistance etc. Some of these are uncertain and change with time and therefore cannot be obtained exactly in advance. The explicit use of these parameters may degenerate the control performance and may lead to instability.

In order to solve this problem, some adaptive control schemes for manipulators have also been proposed. Adaptive control has become one of the most popular techniques in modern automatic control. An adaptive controller can alter its behaviour in response to changes in a process and dynamic disturbances unlike, an ordinary feedback system whose control action remains fixed. Adaptive control can be defined as "a control system which continuously and automatically measures the dynamic characteristics (such as the transfer function or state equation) of the system, compares them with the desired dynamic characteristics, and uses the difference to modify the signal so that the optimal performance can be maintained regardless of the environmental changes; an alternative to such a system may continuously measure its own parameters so as to maintain optimal performance regardless of the environmental changes" [7.4]. One of the basic features of adaptive control is that the system must have its own self-organizing features, if the adjustment of the parameters is done only by direct measurement of the environment, the system is not adaptive. Iterative learning is another type of control which has been applied in recent years. Its simplicity and effectiveness have made it popular. It has great advantages over adaptive control approaches particularly if the system is driven at high speeds. In the following sections two basic approaches using adaptive control schemes together with a new scheme based on iterative learning are discussed.

7.2.1 Self-tuning

A recent development gaining increasing importance is so-called self-tuning control. This is an adaptive control method designed to control a process

(see Fig.7.1) which may or may not be contaminated with noise and whose parameters are unknown but are either constant or varying slowly with time [7.5]. Unknown parameters can be estimated using a parameter identification algorithm such as the recursive least squares method. As soon as new parameters have been obtained they are employed in a controller synthesis stage which produces the coefficients of the controller (gains). For example, these parameters correspond to proportional, integral and derivative gains if the controller is based on a PID control system.

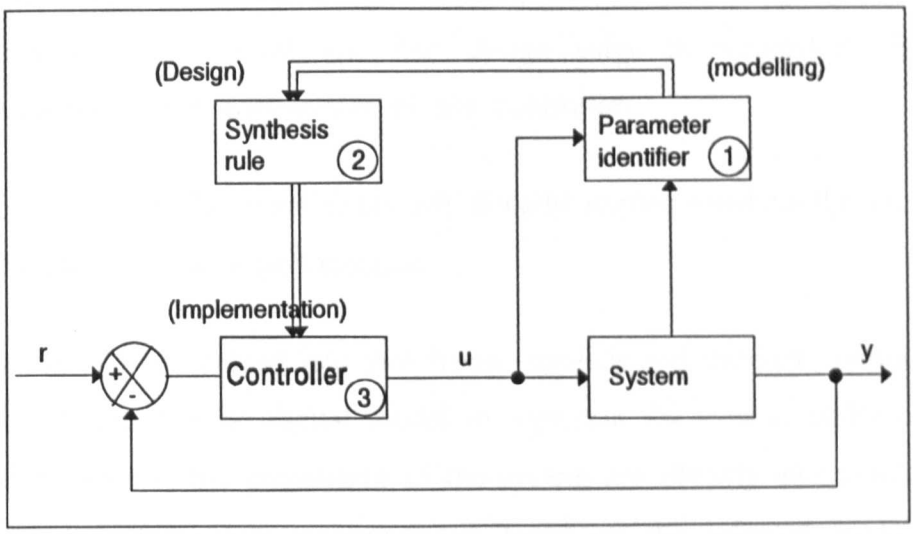


Fig.7.1. Self-tuning controller.

Once the new control input has been calculated by means of the controller parameters, and applied to the system, then the algorithm waits for the next sampling pulse before taking another input value and repeating the parameter estimation process. To ensure good stability and convergence characteristics, certain conditions must be satisfied and a number of assumptions must be made about the process and its environment [7.6]. Specifically, the basic aim of self-tuning control is the automatic adjustment of the parameters of a digital controller to meet a particular requirement with the minimum compulsory knowledge of the dynamic state of a controlled process [7.7]. This is done through the adjustment of the coefficients of the corresponding controller.

In order to simplify the design procedure for a self-tuning system, the principal stages can be sequenced as follows:

1)Modelling: The dynamic model of the system (manipulator) is formed at this stage. Most of the parameters of the model are unknown apart from the input and the output for the system. The unknown parameters are then estimated using a parameter estimation method.

2)Design: The mathematical model is manipulated in order to synthesize the controller using one of the design methods, such as pole assignment control, minimum variance control etc. The design stage is completed with the determination of the coefficients of the controller.

3)Implementation: The coefficients are directly implemented to the controller to maintain a required performance.

Self-tuning can be divided into two forms: explicit and implicit control. Both schemes require a valid digital model to represent the system under control. In explicit control, the parameters of the system are directly estimated. Then, the controller parameters are computed based on the selected control law, and on the digital system model. On the other hand, with the implicit method, the controller parameters are indirectly identified. Hence, the implicit method requires less computational effort as compared to the explicit method. Since the parameters are calculated in real time, the saving in computational time is important in fast response applications [7.8].

The adaptive controller, based on a least squares estimation method, was first described by Kalman in 1958 [7.9]. A similar controller, based on least squares estimation and minimum-variance method, in which the uncertainties of the estimation were considered, was published by Peterka in 1970 [7.10]. However, the first actual self-tuning controller was introduced by Astrom and Wittenmark in 1973 [7.11].

There are several new methods for self-tuning and they can be classified into the following three groups:

(i) general pole placement self-tuning (GPP)
(ii) self-tuning robust controller
(iii) hybrid self-tuning.

For more information about these methods the following publications are recommended [7.12]-[7.20]

Despite its advantages, it has also been found that in some cases self-tuning may introduce problems. For example, the response of a system with an offset between the command and the response can be a major problem with self-tuning [7.12] and [7.21].

7.2.2 Model reference adaptive control (MRAC)

Model reference adaptive control is another approach to adaptive control. The aim is to express the desired performance in terms of a reference model, which specifies the desired response of the control system. The feedback of the system is used to calculate the error which is the difference between the model output and the system output. The parameters of the controller are adjusted according to the adaptation rule and the error. The overall aim is to tune the controller so that the system output follows the reference model output closely. Fig.7.2 illustrates this approach.

The model reference adaptive method is a general approach for adjusting the parameters of a controller so that the closed-loop transfer function will be close to a prescribed model [7.22]. The performance of the model is described by a mathematical model which may be linear or non-linear. The parameters of the model may be unknown in which case they can be estimated similarly to those given for the self-tuning method. This model can be solved using a digital controller design method such as pole placement design. The new parameters are then adjusted in order to minimize the error.

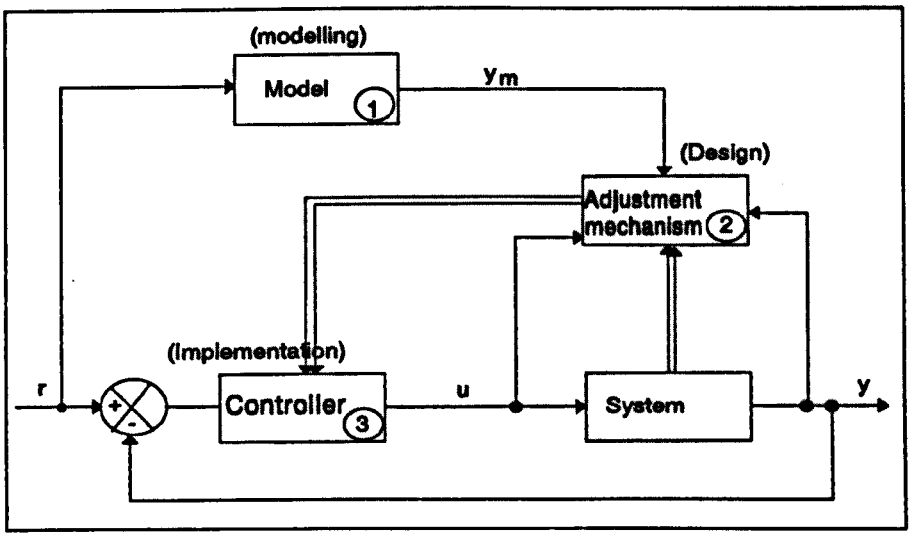


Fig.7.2. Model reference adaptive control.

The following design procedure can be formalized:

1)Modelling: The dynamic model of the reference model is formed at this stage. Most of the parameters of the model are unknown and they are estimated by using a parameter estimation method.

2)Design: The mathematical model is manipulated in order to synthesize the controller using one of the available design methods, such as pole assignment control, minimum variance control etc. The design stage is completed with the determination of the coefficients of the controller.

3)Implementation: The coefficients are directly implemented to the controller to maintain a required performance.

The MRAC technique was first proposed by Whitaker [7.23]. Several modifications were later proposed for MRAC in following studies. These may be divided into three main approaches. The first is Whitaker's approach which is often called the MIT rule since the work was done at the Instrumentation Laboratory at MIT. The MIT rule has been very popular due to its simplicity in practical implementation, although it may require a large number of sensitivity filters for multiparameter adjustments [7.24]. The need for sensitivity filters is avoided in the investigation by Dressler [7.25].

Another rule is suggested by Butchard and Shackcloth in which the Lyapunov function is used to satisfy the system conditions [7.26]. The main advance of this approach is that system stability is guaranteed for all inputs. However, the entire state vector must be available for measurement which it is often not possible [7.25].

Passivity theory is another approach for MRAC, as suggested by Monopoli [7.27] in which the number of differantiators is reduced in order to modify the adaptation mechanism.

7.2.3 Learning

In general, it is very difficult for a programmable system to follow a desired trajectory perfectly due to the existence of dynamic interferences among the mechanical links and other unknown disturbances. To overcome these difficulties, control methods such as self-tuning and reference model adaptive control can be considered. However, these methods require a series of calculations to be made, for example, parameter estimation for the dynamic model and the determination of gain values for the controller. The complexity of the adaptive control algorithm limits the real time use of the adaptive control approach in a trajectory tracking control task especially for systems which are required to follow trajectories at high-speeds. In addition, such approaches are found to be inappropriate when handling trajectory tasks since adaptive control does not guarantee that the system outputs meet the desired values along the entire length of the trajectory, due to the asymptotic convergence property [7.28].

Iterative learning provides a new approach for the control of repetitive programmable machine systems. The method can be easily applied to systems which are non-linear and time variant for which even the mathematical model is unknown. The block diagram for the learning control is illustrated in Fig.7.3. In this scheme, the inputs for the system are tuned systematically until the desired outputs from the system are achieved. To do this, the

inputs for the present cycle are modified using the experience of the previous cycle, therefore the method is called "learning". The learning operation continues until the response approaches the desired output within an acceptable tolerance band. Unlike adaptive control, a dynamic model of the system is not needed and the parameters of the controller are not changed. As a consequence, learning control can be applied more easily in real time for high-speed systems.

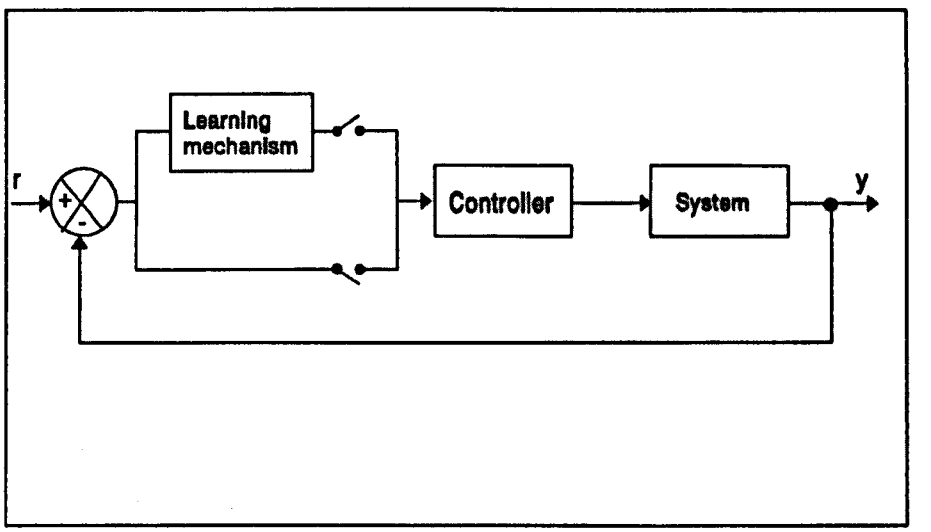


Fig.7.3. Learning control.

The common feature with adaptive control and learning is that the modification of the input is based on the output of the system for both methods. However, there are also some important differences such as the requirement in adaptive control to adjust the parameters of the controller whereas in learning control the input command only is adjusted. Adaptive control also needs a mathematical model of the system which it is not the case in learning control.

Learning control has been suggested by several authors [7.29]-[7.33]. But, it was originally developed by Arimota and his colleagues for PID type controllers [7.34]-[7.36]. Although the method has many advantages over other approaches it may suffer from saturation problems for the actuators after a few cycles depending on the complexity of the trajectory being

followed [3.1] and [3.2]. In this investigation, a learning method is proposed that is based on Aritomo's iterative learning method, however, the method is modified in order to prevent saturation of the actuators occurring.

As mentioned before, the gains of the controller are constant for learning control. However, the response of the system is directly related to the values set for the gains. In practice, the gains are initially preset to some values for the standard controller, however they are not the best values since the gains depend on the dynamics of the driven mechanism. Thus, the parameters of the controller have to be initially adjusted in order to get optimum performance from the system and so to reduce the number of learning cycles. This process is known as tuning.

7.2.3.1 Tuning the gains of the controller

Most controller modules for closed loop servo systems are usually based on PID (proportional, integral and derivative) systems. The parameters of these controllers can be tuned in order to force the system to follow the command closely. Tuning can be described as the process of adjustment of the gains of a servo system in order to decrease the error between input and output without running into instability. In general, the values of the gains determine how hard the system tries to reduce the error. A servo mechanism and its load both have inertia which the motors must accelerate and decelerate while attempting to follow a change in the input. The inertia effects will tend to result in over-correction, with the system oscillating either side of the target (see Fig.7.4). This oscillation must be dampened but too much damping will cause the response to be sluggish. Therefore, the gains of the system should be adjusted to achieve the fastest response with little or no overshoot for proper tuning.

It is commonly the case that once they are tuned manually the parameters remain constant with respect to time, and this is often found to be sufficient to produce a reasonable control action. The initial values for the gains can

be determined by deriving a mathematical model of the system as in adaptive control and then applying one of the control system design methods such as root locus, compensation or frequency response. However, if the mathematical model is too complicated an analytical approach may not be possible. Then the designer must apply an experimental approach to determine the gain values.

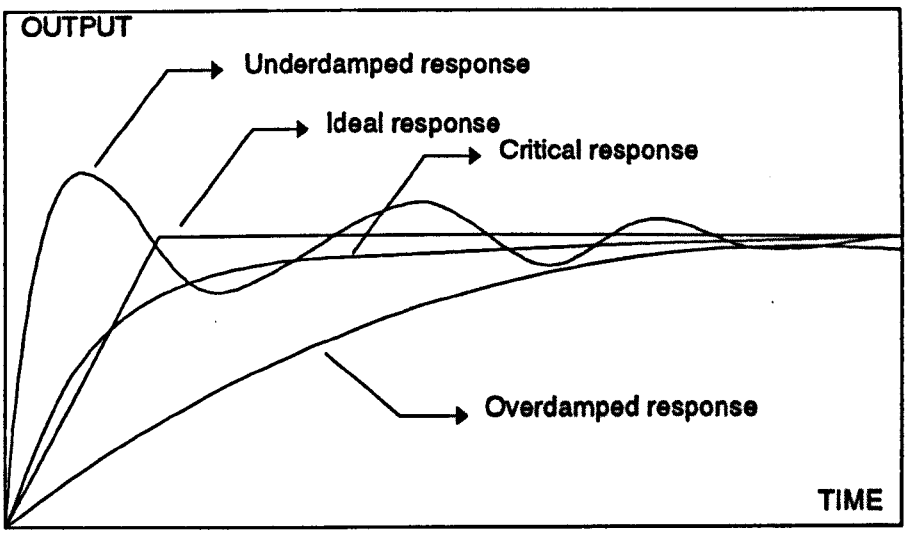


Fig.7.4. Response of a system under different damping factors.

Each gain makes a different contribution to the overall reduction in the amount of the error. For example, in a PID controller the proportional gain affects the positional error of the system. Also, it controls the overall response of the system and the magnitude of the positional following error. The integral gain controls the positional error of the system in the steady state condition, as well as affecting the final accuracy and stiffness of the motors. Finally, the derivative gain controls the positional error due to the velocity of the system and also it controls the damping affect of the motor shafts due to high acceleration rates.

Ziegler and Nichols [7.37] proposed rules for determining values of the gains for PID controller based on the transient response characteristics of a given system. They aimed at obtaining a 25% maximum overshoot for a step response. The response of the system for a unit step can be obtained experimentally. If the system has neither integrator(s) nor dominant com-

plex-conjugate poles, then such a unit-step response may look like an S-shaped curve [7.4], as shown in Fig.7.5. The S-shaped curve may be characterized by two constants, the delay time L and the time constant T. These two constants can be obtained graphically by drawing a tangent line at the inflection point of the curve. The interval between the origin of the co-ordinate axis and intersection point of the tangent line with the time axis gives the time delay L. T is the interval which starts from the end of the time delay and ends at the intersection point of the tangent line with the line c(t)=K as shown in Fig.7.2. In the Ziegler-Nichols method, it is recommended to set the values of proportional gain=1.2T/L, integral gain=2L and derivative gain=0.5L.

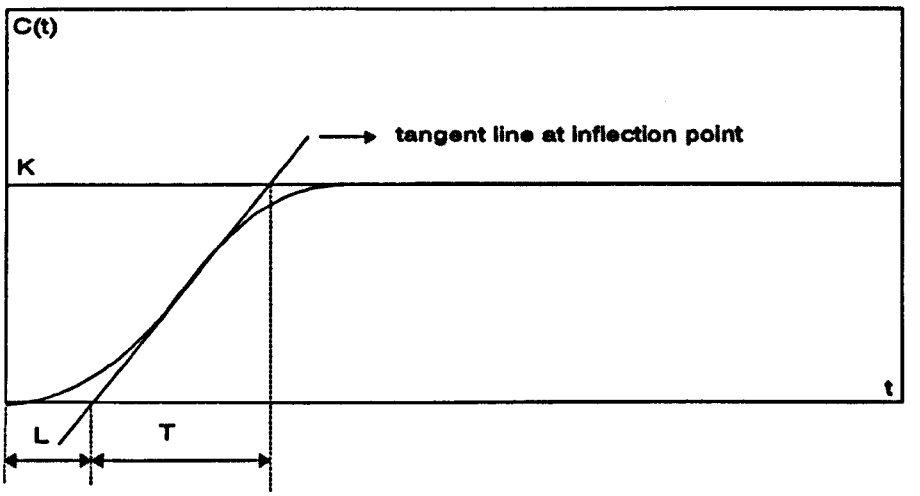


Fig.7.5. Response of a system for a unit step-input without integrator and complex - conjugate.

In the prototype used for this investigation the controller was not based on PID control and there were therefore no established guide-lines for the initial tuning of the gain values of the controller. Despite this, a method based on trial an error was applied to tune the gain values of the controller and a satisfactory result was obtained. After the tuning operation, the resulting response of one of the servo-motors for a unit step input appeared as shown in Fig.7.6.

For this step-response the following parameters were used:

Digital filter gain, $K = 50$

Digital filter zero, $A = 256$

Digital filter pole, $B = 250$

Sample time, $t = 416$ microseconds

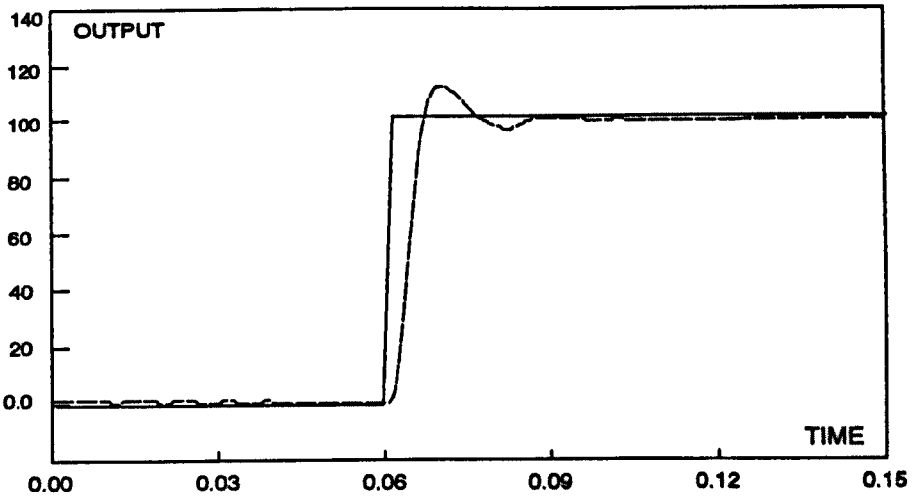


Fig.7.6. Response of a servo motor for a step-input.

7.2.3.2 Iterative learning control algorithm

Using the Lagrangian formulation, the dynamic equation of a manipulator which has n joints driven by n actuators, can be expressed by

$$I(\theta)\ddot{\theta} + N(\theta, \dot{\theta}) + G(\theta) = u \qquad (7.1)$$

Where

I = manipulator inertia matrix

θ = vector of joint co-ordinates

N = matrix of manipulator Coriolis and centrifugal coefficients

G = vector of gravitational coefficients

u = vector of joint torques

The input for the system u can be determined by calculating the above parameters where the joint angles and their derivatives are calculated either using the motion equations of the end-effector or using co-ordinates which

are pre-planned and stored in a data base using the motion design software. However, there may be still a small deviation of the system response with the desired motion due to friction and other unknown disturbances. Therefore, an additional term $e(\theta)$ must be introduced to the control function in order to reduce this deviation. Arimoto's learning algorithm [7.34] is based on this additional term. This term is calculated from the difference between the previous response and the reference input. The above control function then can be rearranged to

$$I(\theta)\ddot{\theta} + N(\theta, \dot{\theta}) + G(\theta) + e(\theta) = u \tag{7.2}$$

This equation is used in adaptive control in order either to calculate the parameters of the controller or to calculate the input torque values of a system that is under torque control. However, the dynamic equation of the manipulator is not necessary when using learning control and inputs for learning control can be directly calculated from the end-effector trajectory. The learning algorithm for position control can be simplified in the following way:

$$u_n = u_{n-1} + e \tag{7.3}$$

Where

u = position input for actuator in terms of encoder counters

e = error factor is based on the experience of the previous cycle.

The error factor e is expressed [7.36] as:

$$e = \lambda(r - y_{n-1}) \tag{7.4}$$

Where

r = reference input (desired output)

y = actual output of the system

λ = learning gain where $0 < \lambda < 1$

n = index for cycle number.

Since the cycle includes a number of data the algorithm can be rewritten for each data point of the trajectory as:

$$u_{n,i} = u_{n-1,i} + \lambda(r_i - y_{n-1,i}) \tag{7.5}$$

Where

i = index for data number in a cycle.

The effectiveness of the above learning algorithm depends on the selection of the learning gain. The servo motors of the system may achieve a saturation point after a few learning cycles if the learning gain is improper [7.38], [3.1] and [3.2]. However, it is observed experimentally that whatever the gain values, the servo motors achieve saturation after 7-10 cycles.

The main reason for the saturation of the actuators is abrupt changes in the shape of trajectory after tuning using the learning algorithm. For example, the motion curves of Fig.7.7 are produced from a polynomial function and the curves are smooth and continuous up to the acceleration. The position curve of this trajectory is used in order to control the system and the above learning algorithm is applied to tune the position curve in order to obtain the desired response from the system. In this case, the learning gain value is selected as $\lambda = 0.3$ and the speed of the system is 150 rpm. After nine cycles of learning the tuned command (dashed) curve is obtained as shown in the figure. It is obvious that this learning algorithm is a simple addition and subtraction. As a result of this calculation, the smoothness of the position curve is disturbed. Since even small discontinuities in the velocity curve may cause extremely large changes in the acceleration curve saturation will result no matter how large the actuator sizes.

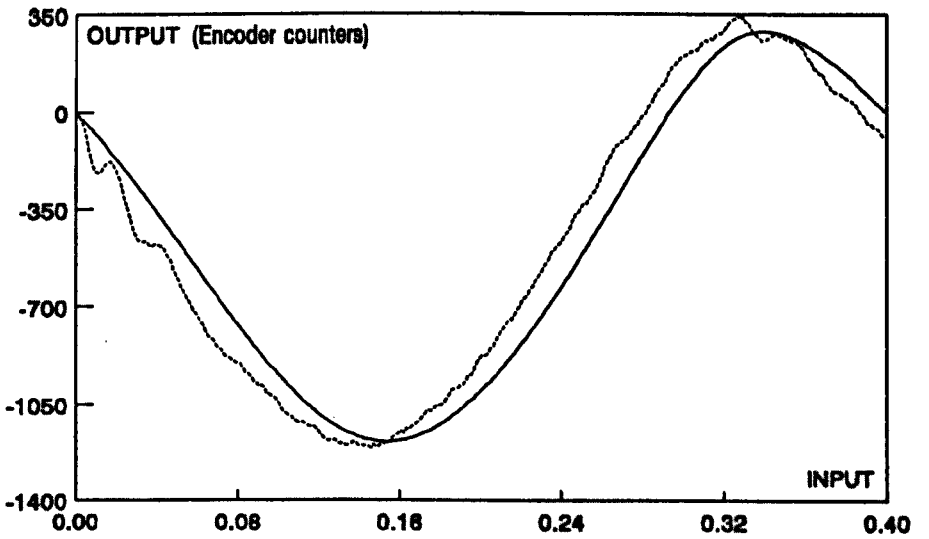


Fig.7.7. Saturation of an actuator after nine learning cycles.

In order to prevent saturation, the inputs must be filtered after each application of the learning algorithm using a digital filter. A digital filter can be described as a process that generates a cleaned output from a contaminated input. The purpose of filtering is to remove extraneous components from the input, for example, extraneous noise can be easily cleaned from data by a filtering operation. Some other curve fitting methods such as least mean squares can also be used to re-smooth the inputs, but curve fitting methods either give unique solutions which may not match the delicate turning points in the required curve or they may need more calculations than for a filtering operation. Filtering is simple and can produce various outputs when different cut-off frequencies are applied.

Non-recursive and recursive filters are commonly used digital filter types. A non-recursive filter is a function of the given inputs producing its output by simply weighting the inputs by constants and then summing the weighted inputs. The constants are called coefficients and these determine the filter. However, a recursive filter is not only a function of inputs, but also depends on the past outputs. In this study a non-recursive filter has been applied for data smoothing. A non-recursive filter can be expressed as [7.39]-[7.41]:

$$u_i = \sum_{k=-m}^{m} u_{i+k} \quad C_k \tag{7.6}$$

Where

C_k = coefficients of the filter

m = half length of the filter

i = index for filtered data

k = index for filter coefficients.

Fig.7.8 is given to explain more easily the algorithm of a non-recursive filter. As seen in the figure the length of the filter is $2m$. A longer filter generates more faithful output because it uses more data. However, since the operation is required in real time the number of the coefficients must be restricted. The coefficients of the filter are determined as [7.38]:

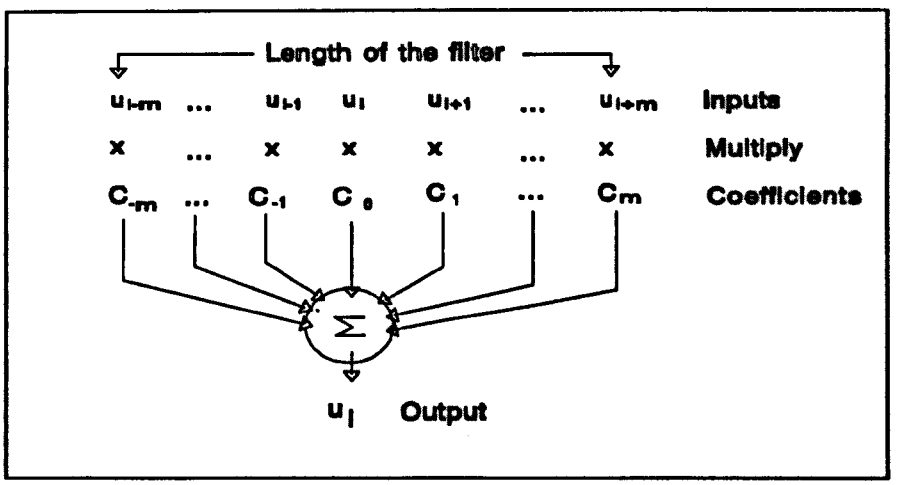


Fig.7.8. Non-recursive filtering algorithm.

$$C_k = \frac{1}{2\pi} \int_{-\omega_c}^{\omega_c} \omega_c \; e^{j\omega k} \; d\omega = \frac{1}{k\pi} \mathrm{Sin}(\omega_c \; k) \qquad (7.7)$$

where

ω_c = cut-off frequency for the filter (rad/s).

Computing the zeroth term is sensitive because both the denominator and numerator are zero. It is determined as:

$$C_0 = \omega_c \tag{7.8}$$

The length of the filter depends on the applied cut-off frequency. It can be determined from the following equation:

$$\sum_{k=-m}^{m} C_k = 1 \tag{7.9}$$

The summation term in equation (7.9) may not always be equal to one. Therefore, the summation is continued until the value is greater or equal to one. However, an error results if the summation is not equal to one. Therefore, a remainder term is used to compensate for this error.

$$R = \sum_{k=-m}^{m} C_k - 1 \tag{7.10}$$

Hence, the filter equation becomes

$$u_i = \left(\sum_{k=-m}^{m} u_{i-k} \; C_k \right) \; / \; R \tag{7.11}$$

An application of filtering to the noisy data set used in Fig.7.7 is illustrated in Fig.7.9.

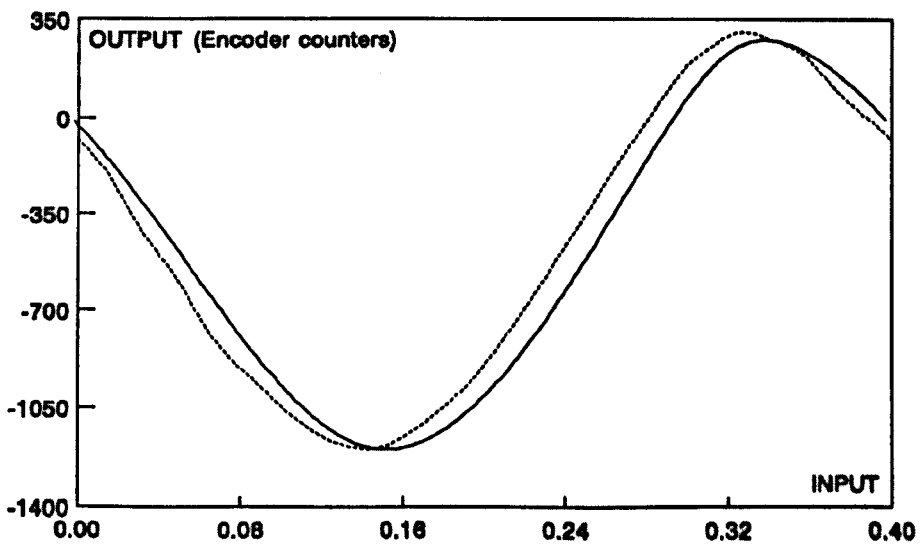


Fig.7.9. Filtering of a noisy data.

As seen from the figure the filter produces quite smooth outputs. In this particular example the filtering is applied to data which is the output of the system after nine learning cycles. Normally, the filtering operation is applied after each learning cycle. Therefore, the output of the filter will be much smoother in a real application than that seen in the figure. In this example, the cut-off frequency was selected as 16 rad/s.

The learning algorithm is combined with the above filtering algorithm in order to prevent saturation of the actuators. The following algorithm is then obtained:

$$u_{n,i} = \left(\sum_{k=i-m}^{i+m} \{u_{n-1,k} + \lambda(r_k - y_{n-1,k})\} * C_{k-i} \right) \quad / \quad R \qquad (7.12)$$

In order to clarify the process of learning control two three-dimensional graphs are used (see Fig.7.10 and Fig.7.11). The first graph shows the improvement of the tracking error cycle by cycle for one axis of the manipulator. The actual motion of this axis can be seen in Fig.5.13(a). Normally, the axis responds to the command with an error shown in the first cycle in the graph. However, the error of the axis has been reduced dramatically by learning control. The maximum value of the tracking error is about (10-15) encoder counts after 53 learning cycles. The encoder resolution is 8000 counts/cycle. Despite the high speed of the manipulator (160 rpm) and the large number of learning cycles, saturation is not observed.

The second example is given to compare the response of the system for smooth and difficult motions. This graph displays the tracking errors of the same axis but the command curve is much smoother then the command used in the first example. The actual command for this axis can be seen in Fig.5.15(a). Both examples have the same speed and the sizes of the paths are similar however the tracking error is modified in 21 learning

cycles. Normally, a few learning cycles (8-10) are enough for reasonable tracking, but it can be continued for several additional cycles to achieve more precise tracking.

7.3 System control software

One of the objectives of this study is to examine experimentally the response of the system for different trajectories and different speeds particularly high speeds. Therefore, control software has been designed in order to manipulate the servo system. The software is a general rather than a specific control program, such that all the parameters including the trajectories are changeable on-line (in real time). Furthermore, the user can examine the curves which correspond to the command input and response of the system. In the following sections, the features of the control software are explained and some examples of trajectory tracing are presented graphically.

7.3.1 Initialization of the system

The system is initialised under the control of the user. The program reads the position data of the trajectories from the described path. Five different trajectories can be loaded simultaneously having previously been planned and stored using the motion design package "MOTDES". The next step is to detect marker positions on each of the drive axes to determine the initial start up position for the manipulator. To accomplish this, the user can rotate the shafts of the servo motors slowly in both directions by pressing left and right keys (← →) on the keyboard until the encoders register the marker positions. This position is called the "home" position for the system. The system is reset at this position in order to make the encoder values zero. However, this position may not be a good starting point from which to follow the trajectories. Therefore, the mechanism has to be moved to a more suitable starting point. The starting point of the system is either determined automatically by pressing the key (A) or the user is allowed to do it manually using the arrow keys.

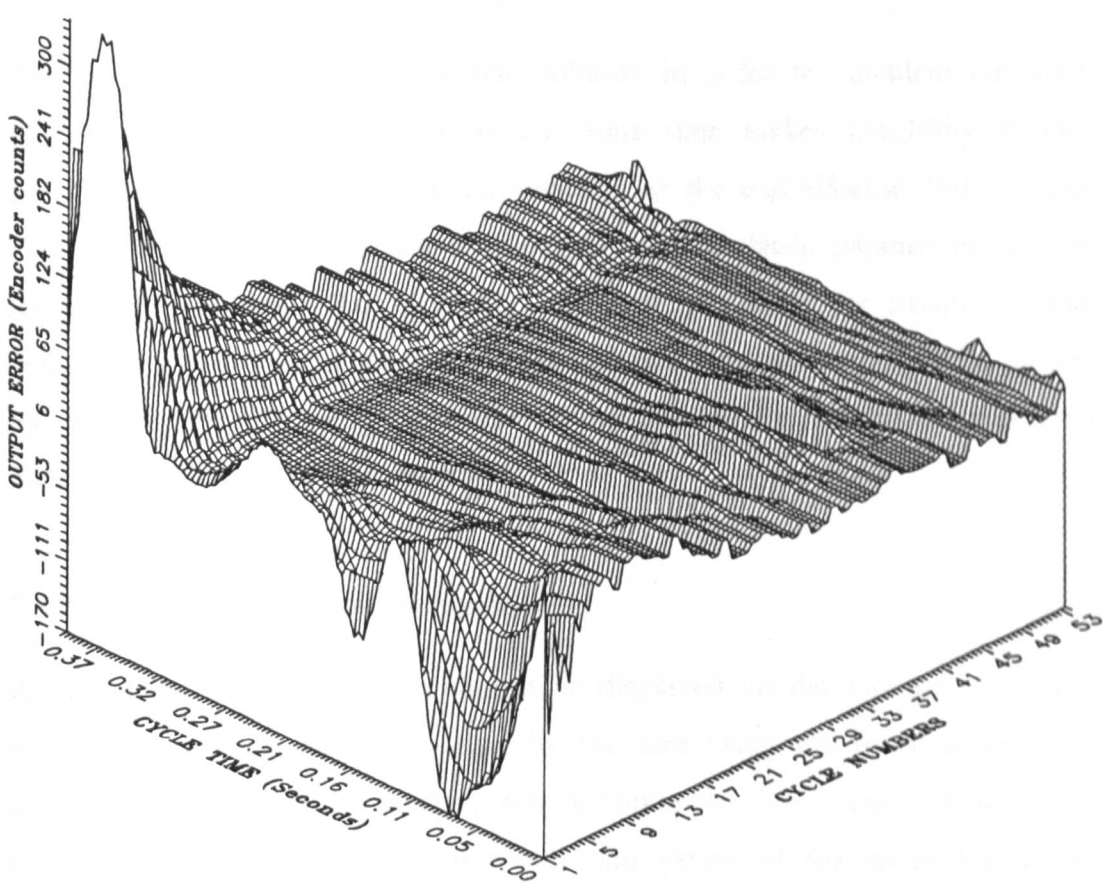


Fig.7.10. Improvement of tracing errors by the application of learning control for 53 cycles.

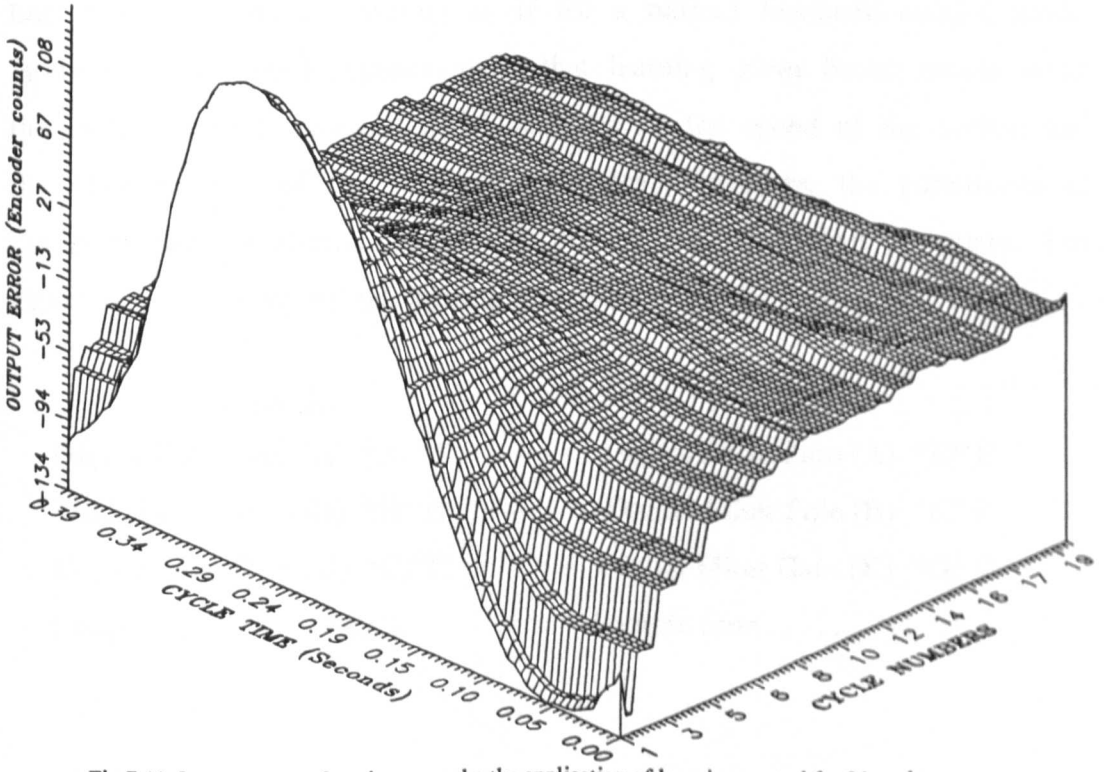


Fig.7.11. Improvement of tracing errors by the application of learning control for 21 cycles.

7.3.2 Inverse solution and feasibility checks

The program determines the inverse solution in order to calculate the joint co-ordinates of the axes while at the same time makes feasibility checks for all the trajectories. Position co-ordinates of the end-effector, link lengths and the initial position of the manipulator are the basic parameters needed for inverse solution. The feasibility check is performed by examining the end-effector positions of the trajectories and checking whether they lie within the work-space of the manipulator. If the checks are valid the system is then ready to execute the motions.

7.3.3 Changing the system parameters

All the parameters of the system can be displayed on the monitor and their values can be changed at any time by the user using the function keys on the keyboard without interrupting the operation of the system. These parameters consist of the sample time and gain values of the controller which control the response of the system.

The gain values of the controller are tuned initially in order to obtain the best response from the system as if for a normal feedback control mode. However, it is found experimentally that learning gives better results when the system is under-damped which depends on the speed of the system and the characteristics of the followed trajectory. Therefore, the parameters of the controller are allowed to be changed by the user when necessary. The function keys to adjust the system parameters on-line are given below:

Axis -1	Axis -2
Digital Filter Zero (A) *A/^A	Digital Filter Zero (A) *E/^E
Digital Filter Pole (B) *B/^B	Digital Filter Pole (B) *F/^F
Digital Filter Gain (K) *C/^C	Digital Filter Gain (K) *G/^G
Sample time ..... *D/^D	Sample time ..... *H/^H

For example, pressing *A (Alt-A) increases the Digital Filter Zero for one unit whereas ^A (Ctrl-A) decreases the same parameter for one unit.

As previously noted the program is capable of executing up to five different motions which have been previously designed using **MOTDES**. The inverse kinematic solution for each trajectory has been calculated within the program and is ready to be used as input commands. Each motion can be executed by the manipulator without it stopping i.e. motions can be switched "on the fly". The motions are introduced using the five function keys **(F1,..,F5)**. However, the user can also switch the motion after stopping the manipulator. The action of the system can be stopped by pressing the **(Delete)** key and reactivated by the **(Insert)** key.

The program displays the curves of the followed path automatically when a new trajectory is activated.

7.3.4 Monitoring and switching of motions

The motion of the axes are displayed graphically on the monitor screen in order to observe the response of the system to see whether it follows the required curve or not. This is especially vital for learning operations since the process is terminated by the user when the response of the system is close enough to the reference input. Therefore, the curves of each axis can be monitored independently using the keys **(Alt-X, Alt-Y and Alt-T)**. Where, **(Alt-X)** and **(Alt-Y)** show the curves of the first and second axes respectively. However, the actual path followed by the end-effector is shown on the monitor when the **(Alt-T)** key is pressed. The user can see three different curves for each axis, they represent the reference curve, command curve and the response of the axis. In fact, the reference curve and command curve are identical initially, but then the command curve starts to deviate from the reference curve in order to reduce the error between the reference and response curves with the application of the learning process.

Another key, (S) is used to activate the learning process for one cycle. One cycle is selected deliberately, because saturation of the servo motors may happen and therefore the system may go out of control at high speeds if the gain values are not appropriate. The learning process continues as long as the key (S) is pressed. Another key **(Home)** initialises the current command. This key is needed when restarting the learning process, for example, if after a number of learning cycles the user wishes to restart the learning process with different gain values the command must be re-initialised.

7.3.5 Adjusting of the speed of the system

Adjusting the speed of the system in real time is one of the very useful features of the control software. The current speed is displayed on the monitor all the time. There are several methods for adjusting the speed of the system and they are given below:

1) Changing the sampling time:

$$rpm = t * n * 60 \tag{7.13}$$

Where

t = sample time in seconds

n = number of data points on the trajectory.

In this equation the adjustable parameter is the sampling time t. But, there are some limitations with the sampling time, for example, it is restricted to a value between 64-2048 micro seconds for the control card used. Furthermore, it is suggested by the manufacturer of the control card that the sample time should be set near to the minimum value for stability of the system.

2) Changing the number of data in the trajectory:

In the above equation the number of data n can also be changed to adjust the speed of the system. However, changing the number of data is not an

easy task in real time. First of all, it needs a large number of data points to achieve quite slow speeds and the memory of the host computer for static data is limited (65 kb maximum). Also, this approach is expensive in terms of computer time which adversely affects real time operation.

3) Using delay time:

$$rpm = (t + T) \ * \ n \ * \ 60 \qquad (7.14)$$

Where

T = delay time in seconds

In this method it is not necessary to change either the sample time t or number of data n in order to adjust the speed of the system. A delay time T is used to adjust the speed of the system. A specific cycle time can be achieved by using the above equation or it can be done on-line by changing the delay time T systematically using the keys **(Alt-N/Ctrl-N)**. Here, pressing the keys **(Alt-N)** increases cycle time and the keys **(Ctrl-N)** decreases the cycle time.

In the experimental arrangement, the first and third methods are used together. To this end, the user is required to keep pressing the appropriate keys while observing the displayed speed of the system until it reaches the desired value. Alternatively, the sample time is changed.

7.4 Implementation of example trajectories

Several example trajectories have been implemented using the experimental system. These have been planned and presented graphically in the Chapter 5. In this section the responses of the system for these trajectories are examined.

Since position control only is used in the arrangement, the velocity or acceleration values of the trajectory are not needed. This situation allows the freedom of adjustment of the speed of the system. Clearly, the shape of the position curves do not change by changing the cycle time.

Example-1 This example is one of the severe trajectories to be implemented in this study. Four operations are planned to be carried out in a cycle and each operation is required to be executed with the end-effector stationary. Stopping of the manipulator at some points when following a trajectory causes abrupt changes in its motion as seen in Fig.5.13 particularly in the acceleration curve. Therefore, such trajectories are not easily implemented especially at high speeds.

The output of the system is given in Fig.7.12 where the speed of the system is 90 rpm. The first two figures (a) and (b) in the first column show the command (solid) and the response (dashed) for each axis of the system. The third figure (c) displays the desired path and the actual path at the end-effector level. It is seen that the system responds to the command with a noticeable amount of error when normal feedback control is applied. An acceptable execution of the job process cannot be expected from the system with this amount of error.

The second column of Fig.7.12 presents another output for this example with the same format but this time with learning applied to control the tracing error. All the figures in this column (d), (e) and (f) indicate that the response is much better than before.

Example-2 In contrast to the first example, the curves of this trajectory are very smooth. They can be seen in Fig.5.15 where the continuity of the curves between the segments is maintained up to the level of jerk.

When this motion is implemented on the system, the manipulator follows the desired command perfectly at slow speeds (60-70 rpm) using normal feedback control. However, as the speed is increased (180 rpm) the response

deviates from the command as seen in the Fig.7.13 (a-c). Activating learning for a few cycles improves the response of the system and it matches with the desired path as shown in Fig.7.13(d-f).

The same motion is performed at higher speed (400 rpm). The normal output of the system is given in Fig.7.14 (a-c). As seen, the response is far away from the desired output. The tuning of the command using the learning control technique improves the response as seen in Fig.7.14. (D-f) however some error still remains.

As a conclusion of this example we can say that the response of a programmable system depends on the characteristics of the motion curves (i.e. smoothness), the control technique used and the speed of the system. A programmable system can be easily controlled with normal feedback techniques and quite acceptable responses can be obtained at slow speeds, however, the system can be very sensitive when its speed is increased. Therefore, the smoothness of the planned trajectory can become the most important factor affecting the control of a programmable system at higher speeds.

Example-3 The manipulator is tested for a square path which is planned in Fig.5.19. The corners of the square are rounded in order to obtain smooth motion curves.

The output of the system is given in Fig.7.15 which is obtained when the manipulator is running at 180 rpm. The first and second columns show respectively the outputs of the system with and without learning control. Despite the difficulty of the motion, a satisfactory result is obtained at this speed, when learning is used.

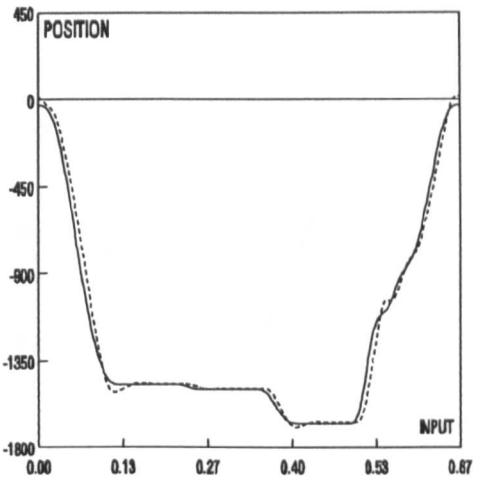
Example-4 An interesting output of the system is presented in Fig.7.16 and Fig.7.17. They correspond to trajectories planned earlier (see Fig.5.21 and 5.22) to show the effect of the arbitrary power polynomial which has been developed.

These paths are tested at different speeds and it is observed that the system responds better to the trajectory produced using arbitrary power polynomials at lower speeds. When the speed of the system is increased to 180 rpm very similar output is produced for both trajectories. However, if the speed is increased further (350-400) then the system becomes more sensitive and gives a better response for the command which is produced using a normal polynomial function. This can be explained by the smoothness of trajectories. The curves of the trajectory which are produced using normal polynomial interpolation are smoother than the motion curves of the other trajectory.

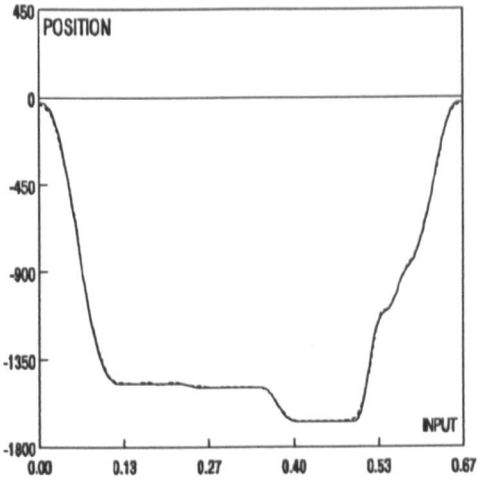
Example-5 The meandering model is illustrated in this example. Its trajectory is planned in Fig.5.25 to show the meandering of a polynomial function. As shown in the figure the effect of interpolation is to produce longer paths than expected. This drawback had been improved by using an arbitrary power polynomial function (see Fig.5.26).

When these two trajectories are applied to the system, the outputs support the results which are obtained in example-4. The results are given in Fig.7.18-19.

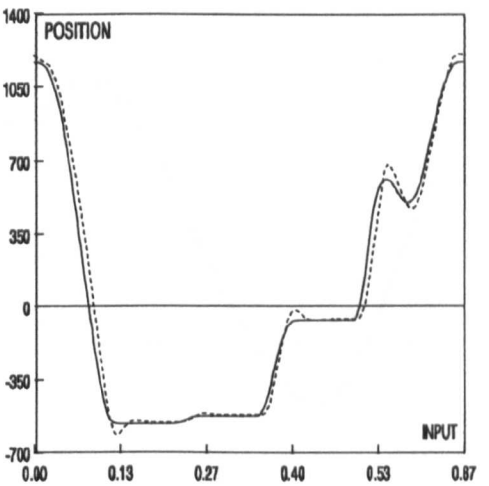
Example-6 This is an example of another complex trajectory. However, it is successfully planned (see Fig.5.27) and implemented by the system. The responses which are obtained with and without learning are displayed in Fig.7.20.



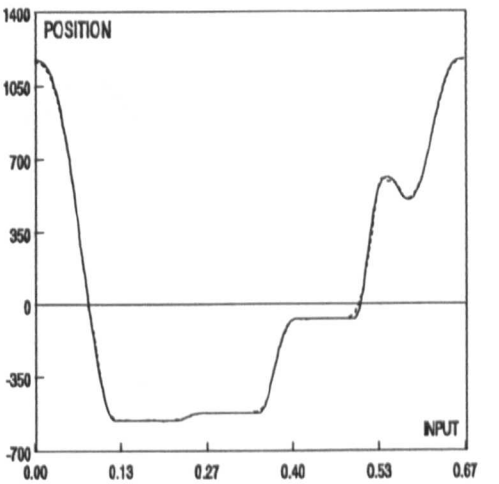
a) Motion of axis-1 before tuning.



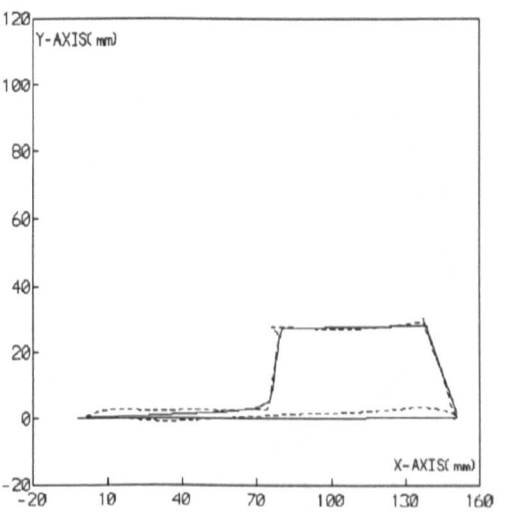
d) Motion of axis-1 after tuning.



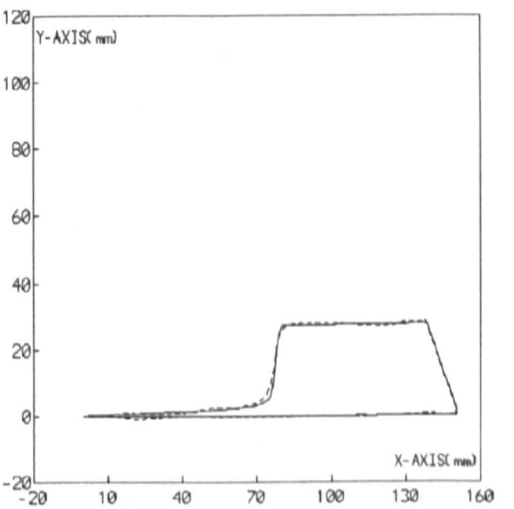
b) Motion of axis-2 before tuning.



e) Motion of axis-2 after tuning.



c) Path of the end-effector before tuning.

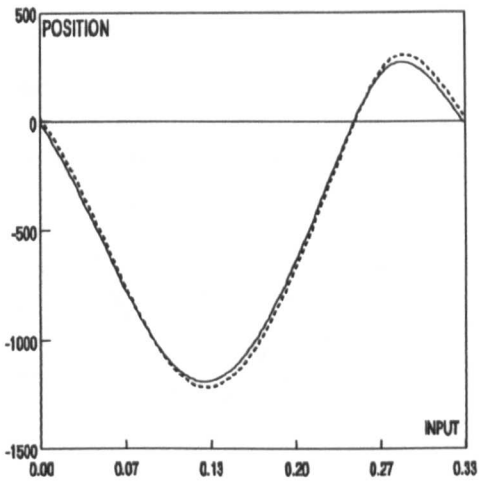


f) Path of the end-effector after tuning.

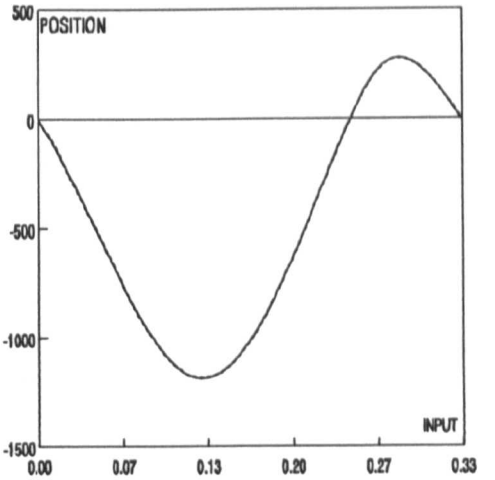
____ COMMAND ---- RESPONSE

POSITION:Encoder counts, INPUT:Seconds.

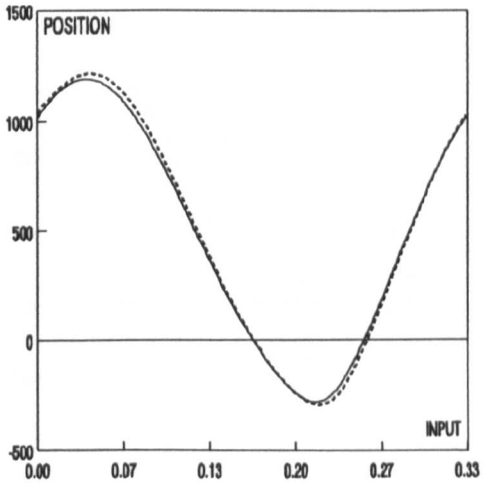
Fig.7.12. Command and Response of the system for example-1.



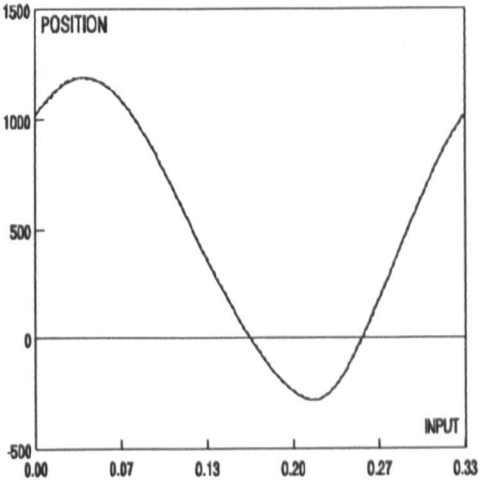
a) Motion of axis-1 before tuning.



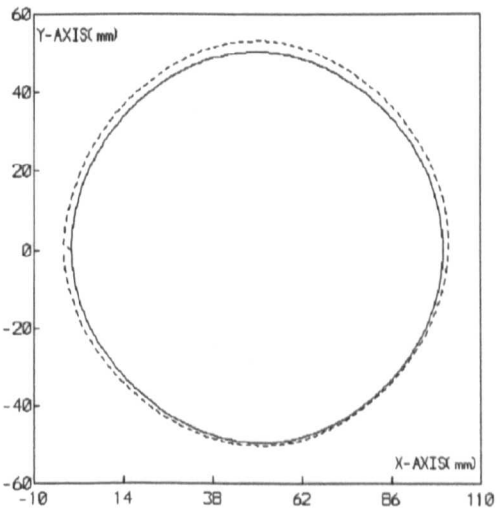
d) Motion of axis-1 after tuning.



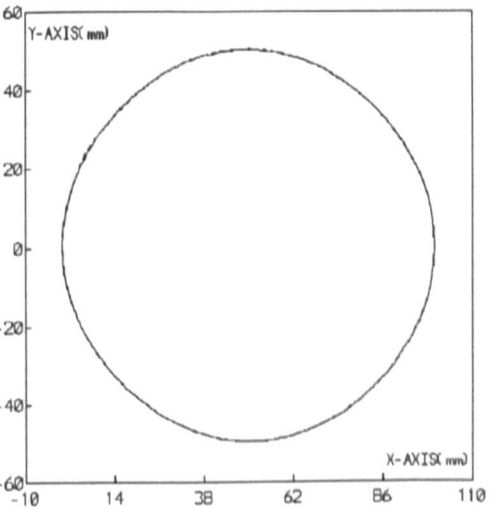
b) Motion of axis-2 before tuning.



e) Motion of axis-2 after tuning.



c) Path of the end-effector before tuning.

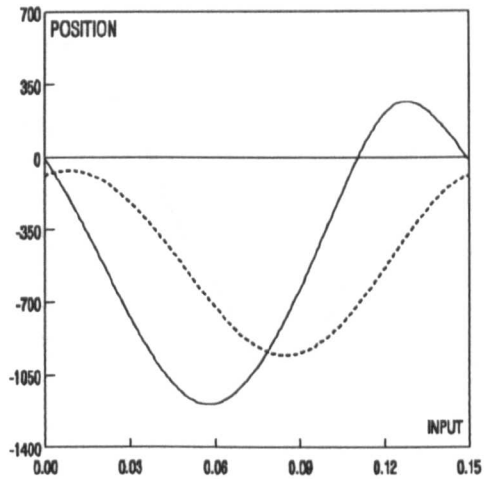


f) Path of the end-effector after tuning.

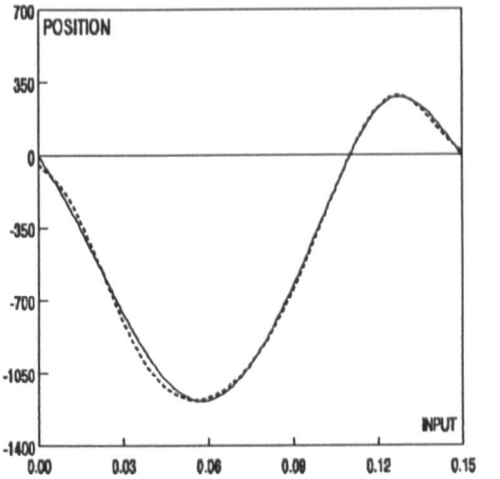
____ COMMAND ---- RESPONSE

POSITION:Encoder counts, INPUT:Seconds.

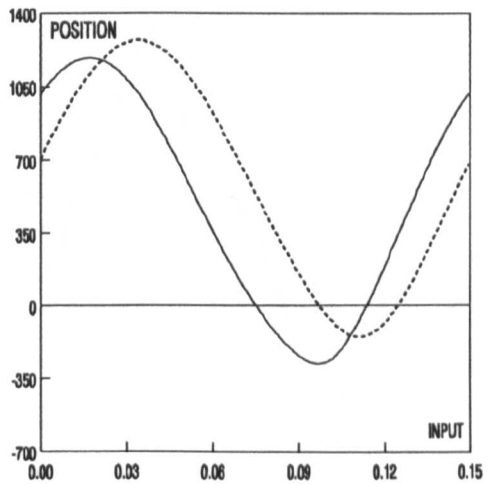
Fig.7.13. Command and Response of the system for example-2.



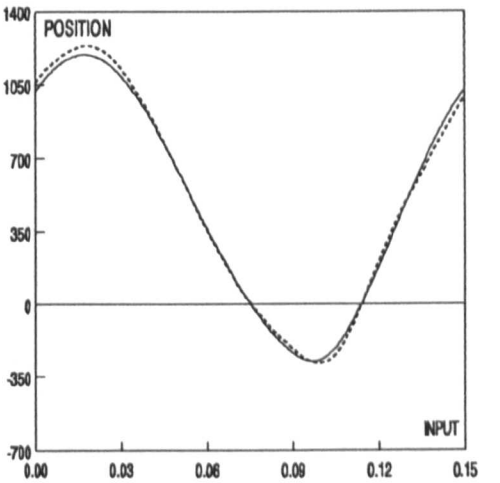
a) Motion of axis-1 before tuning.



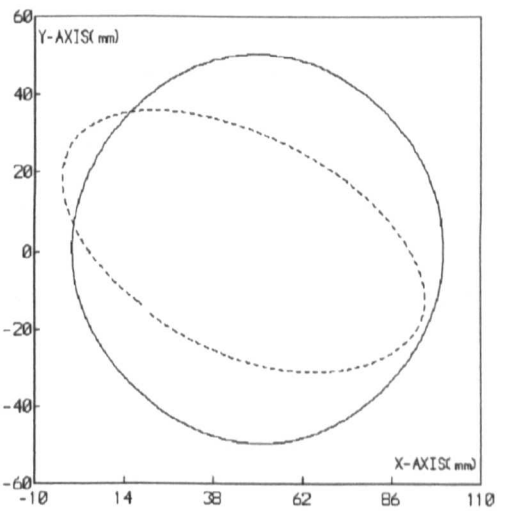
d) Motion of axis-1 after tuning.



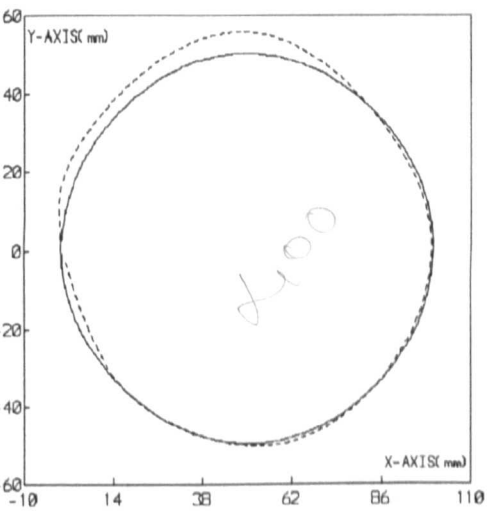
b) Motion of axis-2 before tuning.



e) Motion of axis-2 after tuning.



c) Path of the end-effector before tuning.

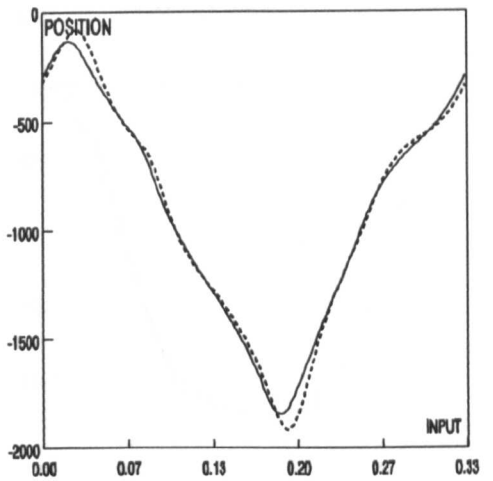


f) Path of the end-effector after tuning.

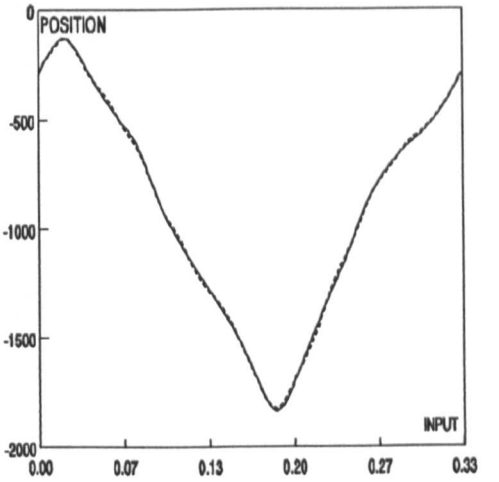
____ COMMAND ---- RESPONSE

POSITION:Encoder counts, INPUT:Seconds.

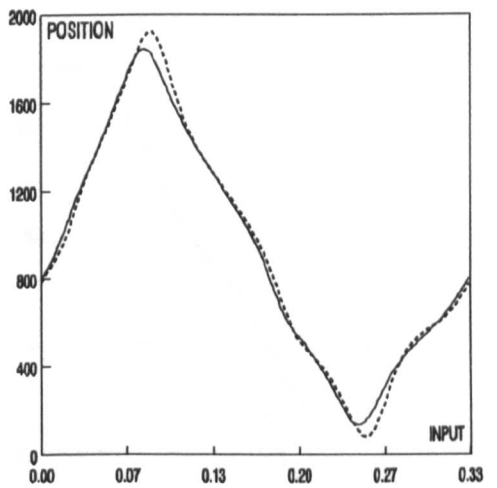
Fig.7.14. Command and Response of the system for example-2.



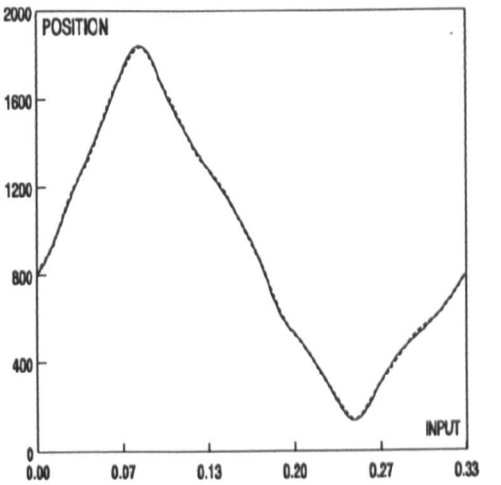
a) Motion of axis-1 before tuning.



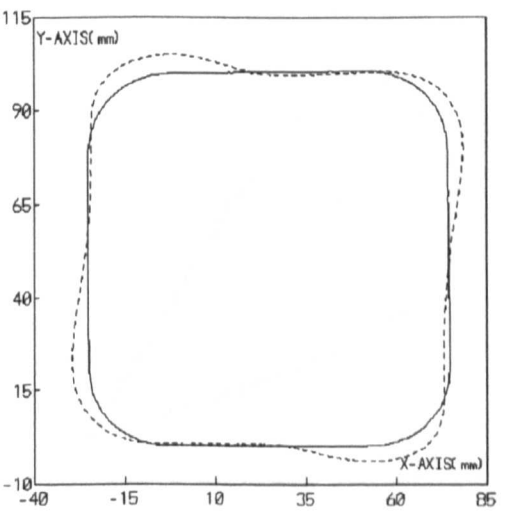
d) Motion of axis-1 after tuning.



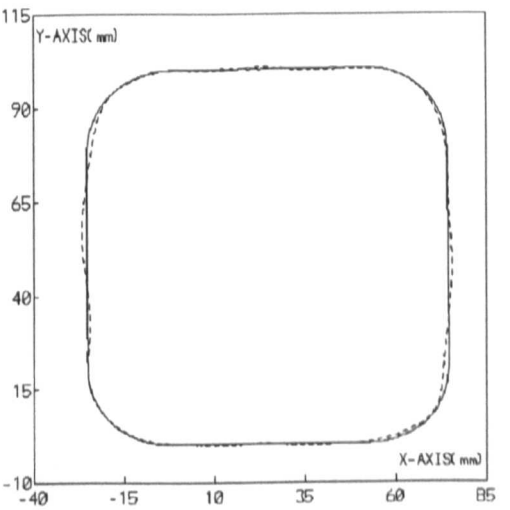
b) Motion of axis-2 before tuning.



e) Motion of axis-2 after tuning.



c) Path of the end-effector before tuning.

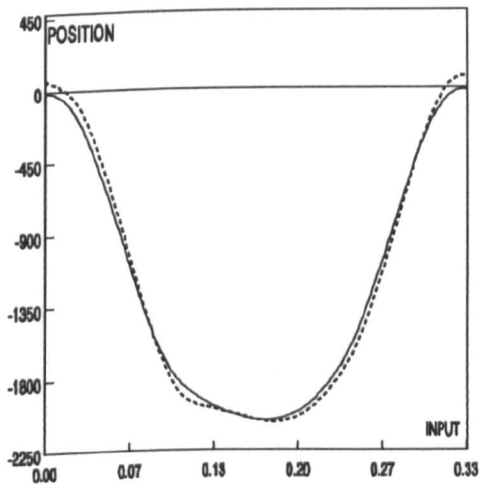


f) Path of the end-effector after tuning.

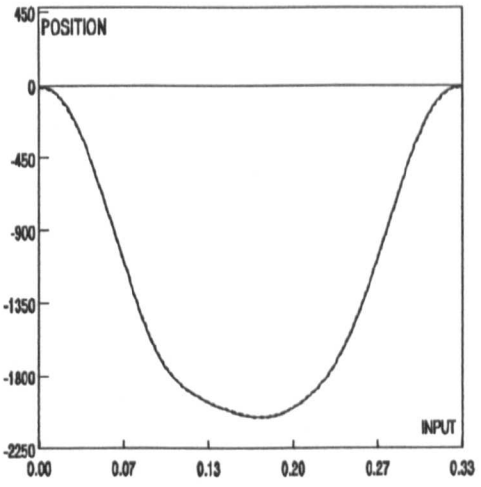
____ COMMAND ---- RESPONSE

POSITION:Encoder counts, INPUT:Seconds.

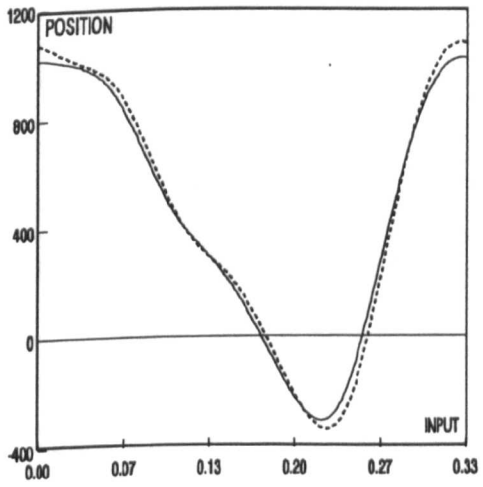
Fig.7.15. Command and Response of the system for example-3.



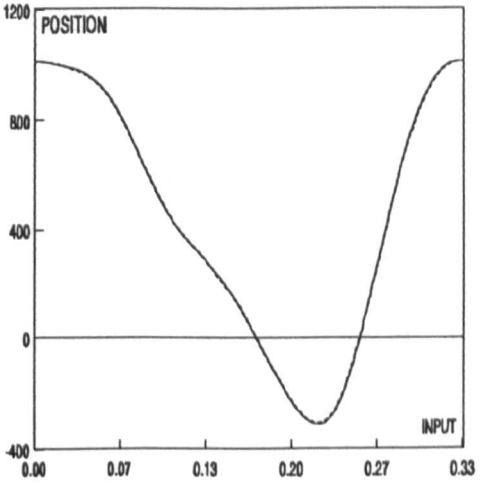
a) Motion of axis-1 before tuning.



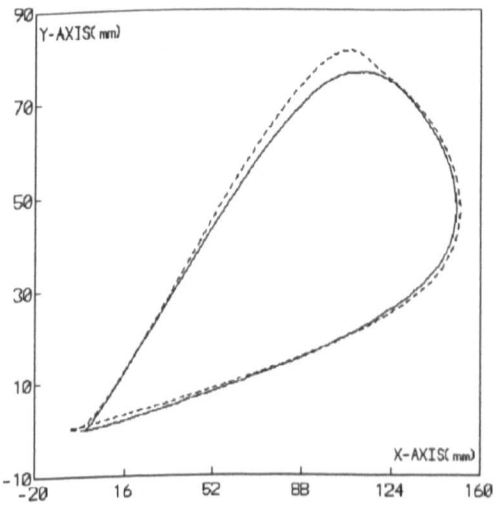
d) Motion of axis-1 after tuning.



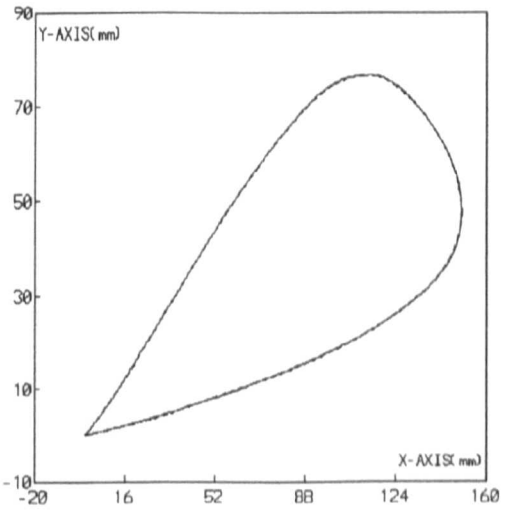
b) Motion of axis-2 before tuning.



e) Motion of axis-2 after tuning.



c) Path of the end-effector before tuning.

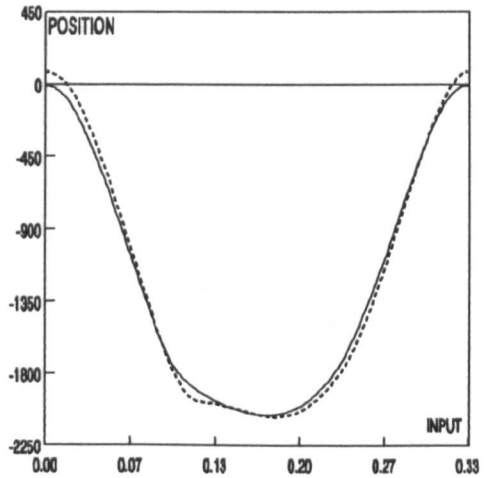


f) Path of the end-effector after tuning.

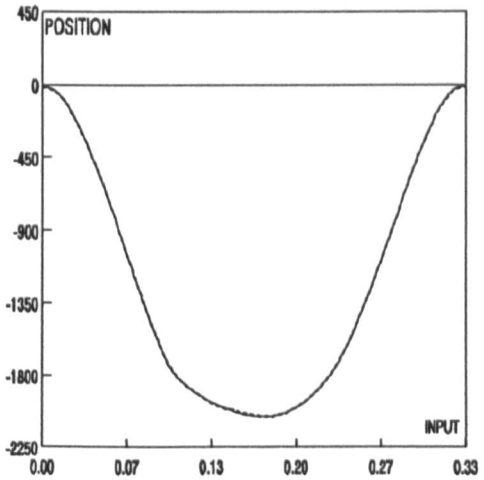
___ COMMAND ---- RESPONSE

POSITION:Encoder counts, INPUT:Seconds.

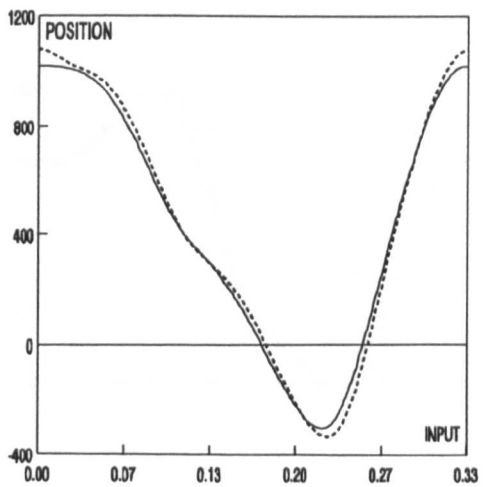
Fig.7.16. Command and Response of the system for example-4.



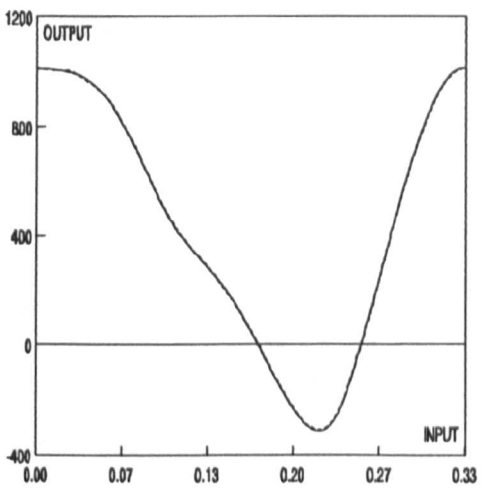
a) Motion of axis-1 before tuning.



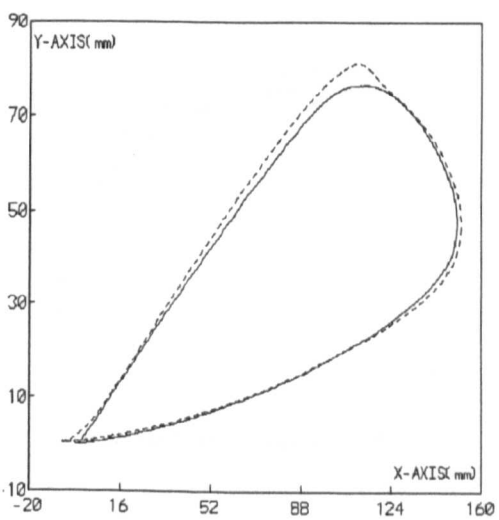
d) Motion of axis-1 after tuning.



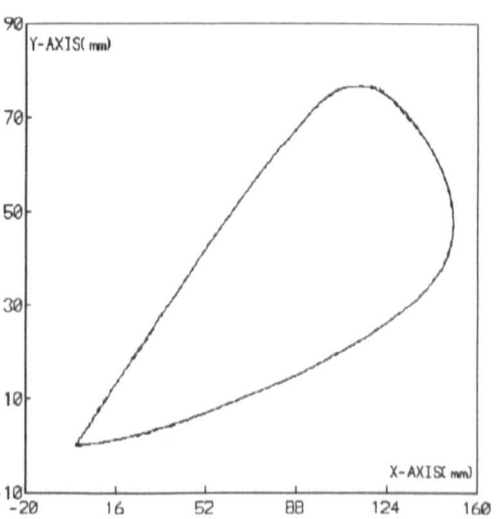
b) Motion of axis-2 before tuning.



e) Motion of axis-2 after tuning.



c) Path of the end-effector before tuning.

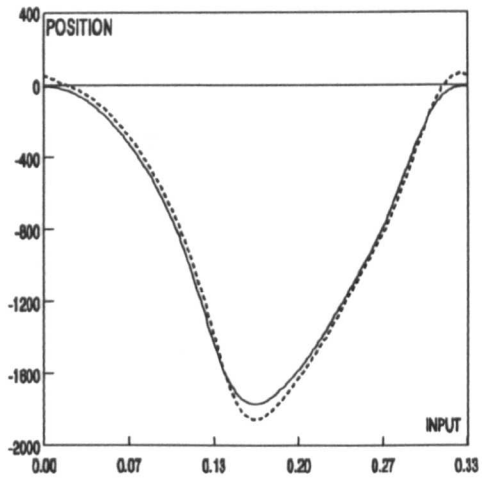


f) Path of the end-effector after tuning.

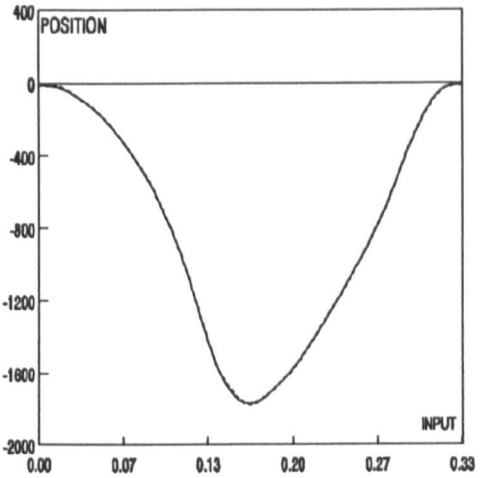
____ COMMAND ---- RESPONSE

POSITION:Encoder counts, INPUT:Seconds.

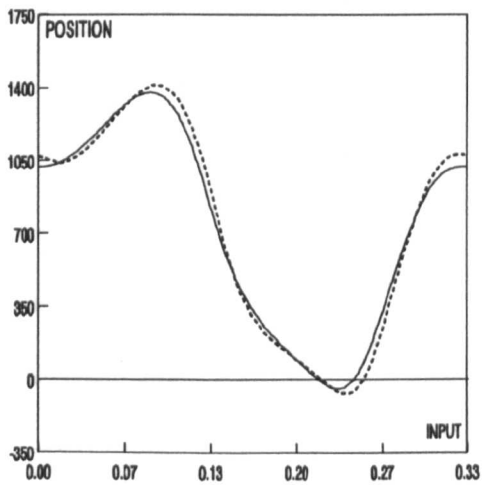
Fig.7.17. Command and Response of the system for example-4.



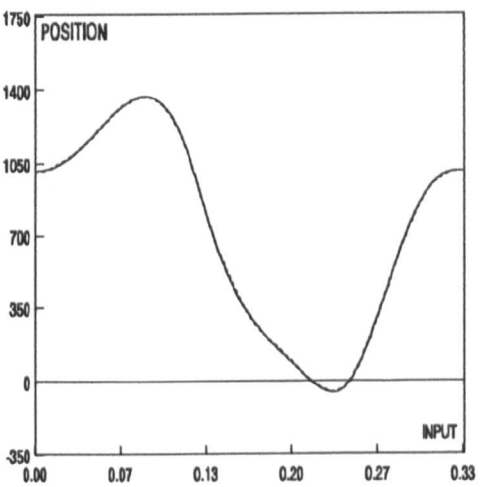
a) Motion of axis-1 before tuning.



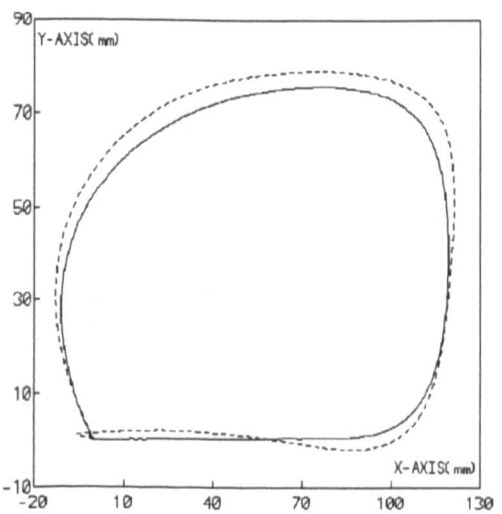
d) Motion of axis-1 after tuning.



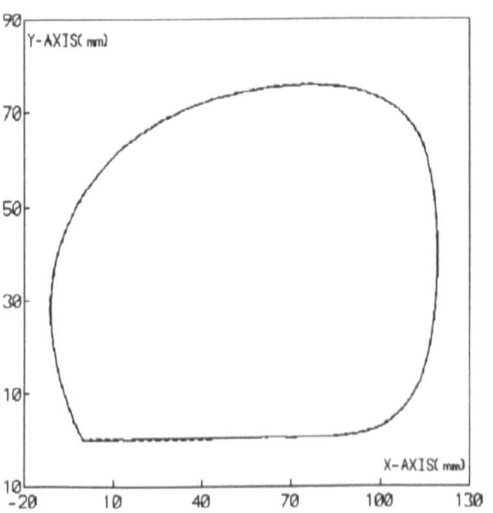
b) Motion of axis-2 before tuning.



e) Motion of axis-2 after tuning.



c) Path of the end-effector before tuning.

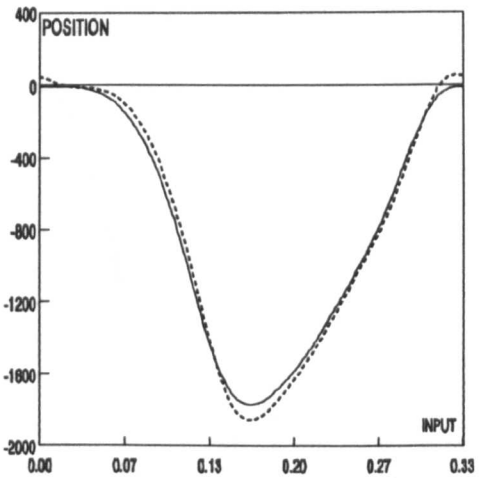


f) Path of the end-effector after tuning.

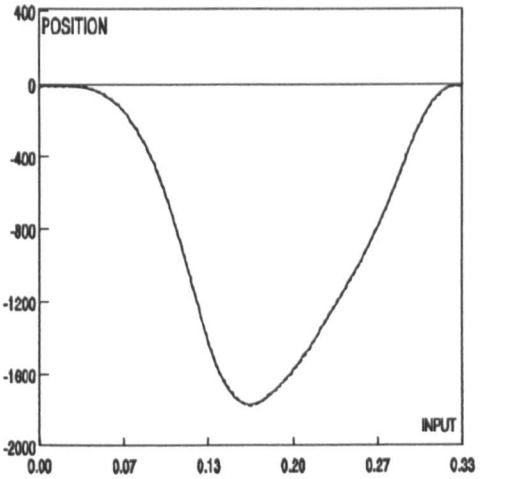
____ COMMAND ---- RESPONSE

POSITION:Encoder counts, INPUT:Seconds.

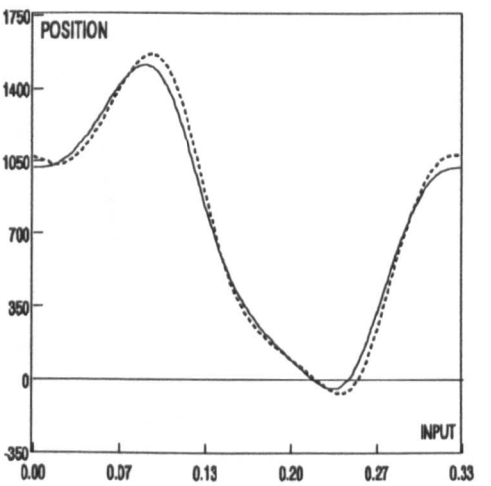
Fig.7.18. Command and Response of the system for example-5.



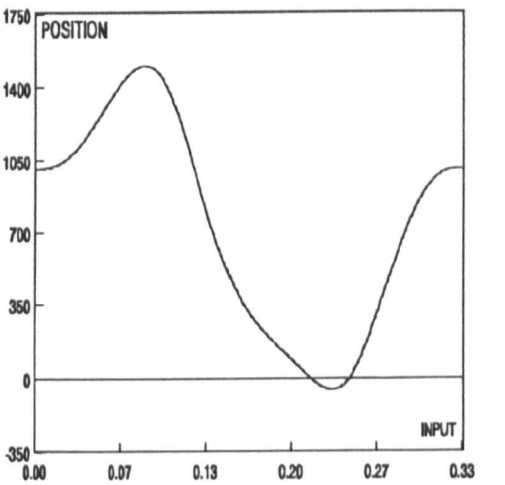
a) Motion of axis-1 before tuning.



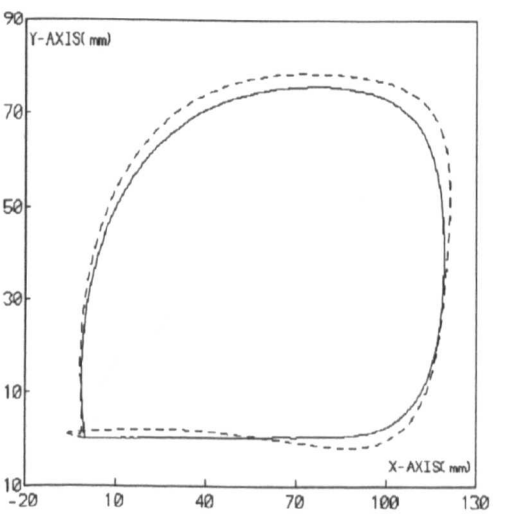
d) Motion of axis-1 after tuning.



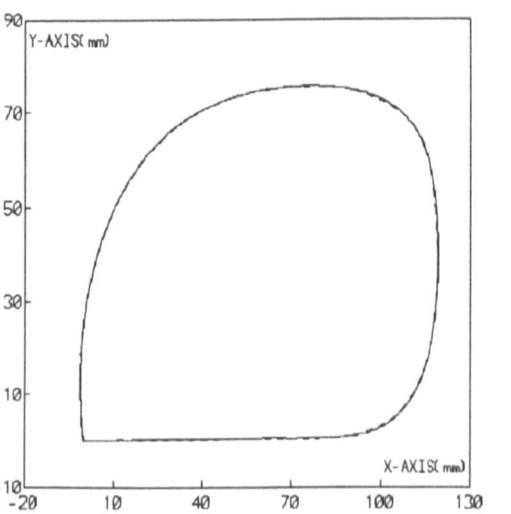
b) Motion of axis-2 before tuning.



e) Motion of axis-2 after tuning.



c) Path of the end-effector before tuning.

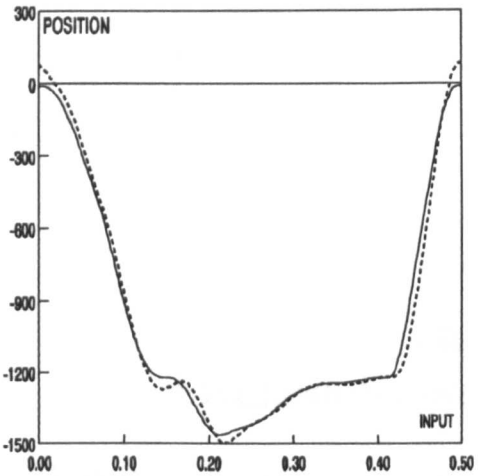


f) Path of the end-effector after tuning.

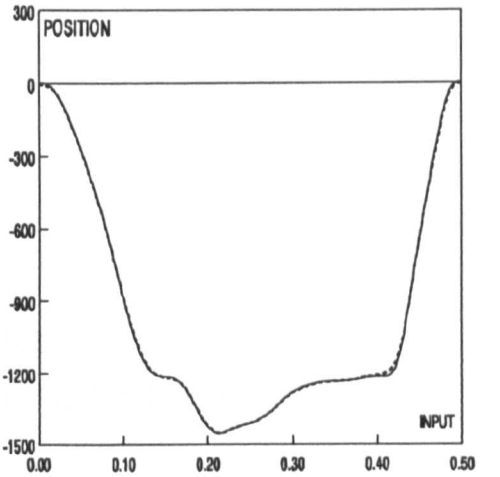
____ COMMAND ---- RESPONSE

POSITION:Encoder counts, INPUT:Seconds.

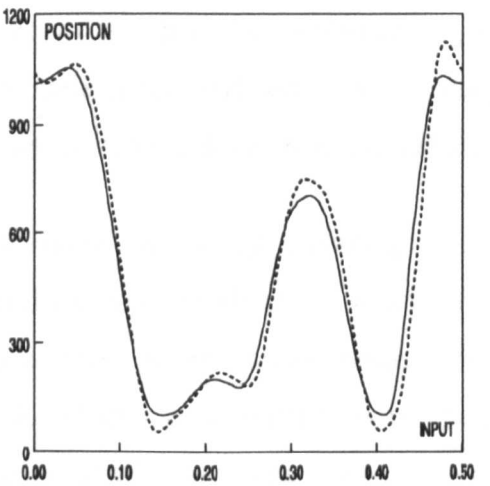
Fig.7.19. Command and Response of the system for example-5.



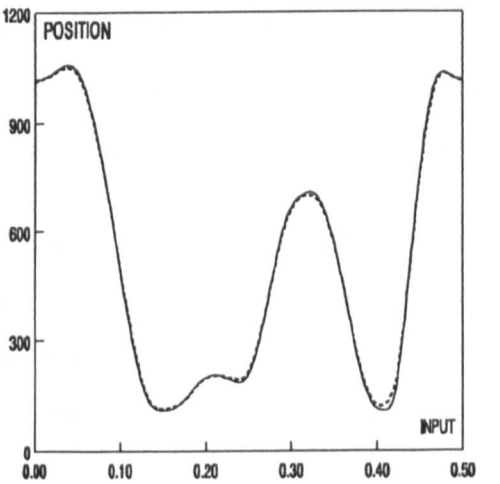
a) Motion of axis-1 before tuning.



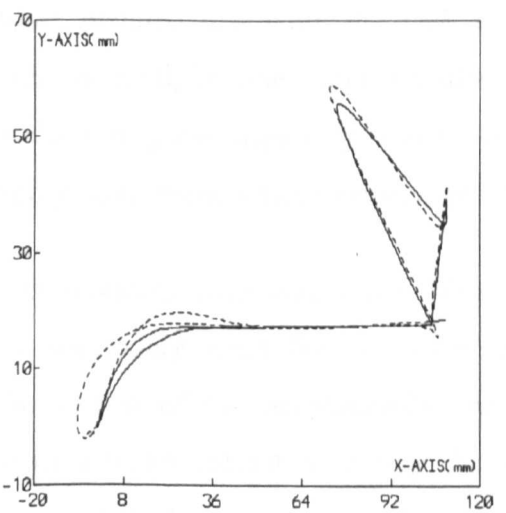
d) Motion of axis-1 after tuning.



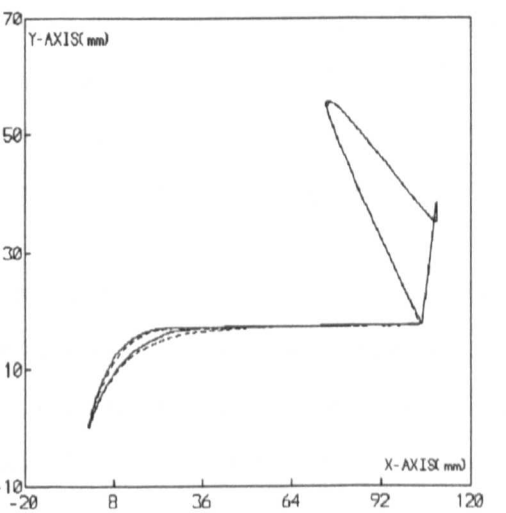
b) Motion of axis-2 before tuning.



e) Motion of axis-2 after tuning.



c) Path of the end-effector before tuning.



f) Path of the end-effector after tuning.

____ COMMAND ---- RESPONSE

POSITION:Encoder counts, INPUT:Seconds.

Fig.7.20. Command and Response of the system for example-6.

CHAPTER 8
CONCLUSIONS AND RECOMMENDATIONS

The dynamic performance of a programmable system depends on both the motion profile to be followed and the feedback control method applied. In order to improve the performance of a system, more specifically to increase the cycle speed and reduce the tracing errors, the trajectory has to be planned at an advanced level and an efficient control method has to be employed.

Traditional trajectory planning methods, such as manual planning, using standard cam motions or spline functions are not very effective methods since they either cannot realise the task specification or they cause high peak velocity or accelerations in the curves. Therefore, programmable systems cannot justify their use economically because of these improper trajectories.

When planned manually, the path is normally composed of simple elements, such as straight lines and circular arcs. Such planning generally fails to produce a good trajectory since no account is taken of the effect of the highly non-linear characteristics of the motion of the manipulators.

Cam motions, especially those with constant acceleration motion (bang-bang) are commonly used for the trajectory planning of industrial manipulators. The values of the accelerations and decelerations are generally selected to avoid actuator saturation during the motion. However, designing a trajectory using constant acceleration and deceleration segments produces a discontinuous acceleration curve and results in infinite jerk values at the connection points of the segments. Such conditions produce large disturbances for a system especially at high speeds.

Cubic spline functions mostly produce satisfactory curves, but, one important complication effecting their use is the common requirement that the motion of a manipulator may depend on the motion of another manipulator which operate together in processing a product. In such cases there must be synchronization between the motion of the machines that may require specification of position, velocity and even acceleration boundary conditions. Specification of arbitrary boundary conditions for derivatives of the function is not possible with cubic spline functions.

Many mathematical functions have been investigated in order to determine whether an appropriate function may be used in a general trajectory planning study. As a result, it was found that every function has its own drawbacks. However, the use of polynomials offers advantages over other mathematical functions. They give the freedom of specifying arbitrary boundary conditions for any derivative of the function, thus the user can use velocity and acceleration boundary conditions as well as position boundary conditions to solve the synchronization problem and to improve the motion curves. In spite of these advantages, polynomials functions may produce undesirable oscillations between design points. Turning points in the motion profile, the degree of the function used and the severity of the boundary conditions specified were determined as the causes of undesirable oscillations.

However, the effects of this drawback can be prevented by means of a number of methods. The first method was to use dummy boundary conditions. Contrarily, using dummy velocity and acceleration boundary conditions increases the degree of the function but at the same time they provide control for the slope and curvature of the function where they are applied and therefore they can prevent oscillations in the motion curves. Practically, however this method is difficult to apply.

The second method was to divide the whole motion into smaller segments and then to apply a number of lower order polynomial functions for these segments. Adding segments to one another to complete the trajectory produces

a segmented polynomial solution. If the curve is divided into segments at turning points an improved result is always retained. However, dividing a curve into segments between the design points was found to be more practical for computer application.

Unfortunately, in spite of dividing the motion into segments polynomials may still produce curves which do not lie within acceptable tolerance envelopes. A method has however been developed to bring these curves within the specified tolerance envelopes. The method is not only appropriate for modifying poor trajectories which suffer from meandering but it can also be used to improve normal curves.

This technique was based on the fact that changing the powers of a polynomial function changes the path between the design points. Normally, the powers of a polynomial function start from zero and increase one by one. This rule has been changed and real numbers were used instead of integer numbers so that the powers of the function could be varied with small intervals such as 0.1 or 0.01 or with any other value. Reducing the powers artificially shows an important characteristic that it forces the main undulations towards the start of the segment in the acceleration curve while the remaining part of the curve stays flat.

Similarly, the powers of a polynomial function can be replaced with large numbers. This approach reverses completely the effect of low order polynomial, that is, it pushes the main undulation to the end of the curve.

In general, it was found that oscillation in polynomials can be better controlled if the undulations in the acceleration curves were pushed towards the two ends of the segment with the minimum amount of disturbance between. Therefore the combination of reduced and raised powers was used in order to achieve this result. Also, it has been shown that polynomial functions with arbitrary powers produce the lowest peak velocity curve among the

motion laws for the same conditions. The other advantage of arbitrary powers is that the shape of a curve can be adjusted without changing any boundary conditions.

There are two approaches for trajectory planning, namely on-line and off-line planning. On-line trajectory planning refers to the determination of the history of a motion by means of on-board sensory equipment and then the generation and execution of the motion in real time. This means that the constraints for the trajectory including constraints for obstacles along the path and constraints needed for co-ordination of motion with other machines will be determined by means of sensory equipment. In practice, such planning cannot be easily undertaken because of the inherent dynamic complexities associated with its implementation, high cost and limited application areas.

In the off-line planning the development of trajectories takes place without access to the manipulator itself and without recourse to real time operation. Off-line planning offers some potential benefits. Since every action taken in the planning is performed by the user, many of the problems peculiar to trajectory planning can be controlled within the computer environment. For example, unexpected oscillations in the trajectory, high peak velocity and acceleration values and discontinuities in the motion curves are such problems which are tackled.

An off-line motion design program has been developed, this program is capable of producing motion curves for the end-effector of a mechanism which can perform a body motion in a plane. Several motion types have been included to provide a wide degree of flexibility for the user. The program includes many facilities to achieve a desired motion. The necessary boundary conditions comprise the main input for the software. Other inter-activity is based on mouse click selection. The generated motion curves are displayed on the monitor all the time. This allows the user to examine the motion curves and modify any undesirable behaviour of the trajectory while in the computer-graphic environment. The trajectories of all three axes (X,Y,θ)

can be designed simultaneously. The program also includes a simulation option which shows how the end-effector moves along the path without needing to access the machine system.

A prototype has been designed and built to implement different motions in the context of a computer controlled system. The objective was to investigate the practical problems of trajectory control particularly when the system was running at high speeds.

In general, it is very difficult for a programmable system to follow a desired trajectory perfectly due to the existence of dynamic interferences among mechanical links and unknown disturbances. When a desired signal is applied to a servo-system it follows the command with an error. The gains' settings of a controller are the main parameters that determine the response of the system. The gains' values can be optimized, however, controllers with fixed gains are not always efficient. There are several cases where precise tracing of trajectory under different payloads require more advanced control techniques. Adaptive control and learning are the two main methods applied for trajectory control problems of programmable systems. There are two main approaches for adaptive control, these are self-tuning and model reference adaptive control.

Self-tuning is based on the idea of continuous tuning of the gain values of the controller. It is supposed to control a process whose parameters are unknown and they are either constant or slowly time varying. The unknown parameters of the dynamic model of the system are estimated using a parameter identification method.

In the model reference adaptive control, the desired performance is expressed in term of reference model. The feedback of the system is used to calculate the error which is the difference between the model output and the system output. The parameters of the controller are adjusted according to the adaptation rule and the error. The overall aim is to force the controller so that the system output follows the reference model output closely.

Iterative learning is another type of control for repetitive tasks that has been developed in recent years. Its simplicity and effectiveness have made it popular. The input of the present cycle is calculated using the experience of the previous cycle so the response approaches the desired output slowly but positively. The main advantages of this system over adaptive control is that a dynamic model of the system is not needed and the parameters of the controller are not changed. Therefore, learning control can be applied more easily in real time for high-speed systems. A problem with a learning algorithm however is that saturation of the servo motors after a few learning cycles may occur due to a lack of smoothness in the command curve caused by the learning algorithm. In order to avoid saturation a new approach has been developed. This is achieved by filtering the input for the system after the application of each learning cycle using a digital filter.

It was found that trajectory control is more difficult when the speed of the system was increased. This implies that adaptive control methods are even less suitable for high-speed systems due to the need for additional calculations. Furthermore such motions can be inappropriate for handling trajectory tasks since adaptive control does not guarantee that the system output meets the desired one along the entire period of trajectory. In this context, learning is a much simpler and effective control technique compared with adaptive control. Therefore, it has been applied exclusively for trajectory control in this work with good results having been obtained for several widely varying examples.

Recommendations

In this study it is found that rational interpolation is one of the most important interpolation methods. It can produce smoother curves that those obtained with other interpolations. However, it is not included in the motion design software because of difficulties with the manipulation of this type of function. It may be possible to improve their use in future work.

Off-line trajectory planning can be extended to high-level planning. In such a system, the user can design the manipulator and its environment by selecting them from a menu and specifying their dimensions and co-ordinates using the computer-graphics environment. Then, the user may state only the objective instead of taking every action. Examples of objectives could be "transfer the products" or "paint the objects". All the remaining work can be performed by the computer. This can be achieved by providing automated solutions to various sub-tasks and letting the programmer use them to explore various options in the simulated environment.

In order to improve the trajectory tracing of the manipulator, learning control is applied and satisfactory outputs are obtained in all cases. However, the effect of changing of the payload has not been investigated. This is further work which needs to be done.

REFERENCES

[1.1] J.E. Bobrow, *Optimal Robot Path Planning Using the Minimum-Time Criterion,* IEEE Journal of Robotics and Automation, V.4, N.4, pp. 443-450, August 1988.

[1.2] K.G. Shin and N.D. McKay, *Minimum-Time Control of Robotic Manipulators with Geometric Path Constraints,* IEEE Transactions on Automatic Control, V.AC-30, N.6, pp. 531-541, June 1985.

[1.3] L.C. Tokuz, *Hybrid Machine Modelling and Control,* Phd. Thesis, School of Engineering and Technology Management, Liverpool Polytechnic, 1992

[1.4] R.P. Podhorodeski and W.L. Cleghorn, *Optimization of manipulator point to point motion considering actuator output capability constraints,* Mech. Mach. Theory, V.23, N.5, pp. 409-419, 1988.

[1.5] H.Sehitoglu, *Design of a Trajectory Controller for Industrial Robots Using Bang-Bang and Cycloidal Motion profiles,* American Society of Mechanical Engineers, Dynamic System and Control Division. Publ. By ASME NewYork NY, USA, pp. 169-175, 1986.

[1.6] J.R. Jones, G.T. Rooney, and P.J. Fisher, *Dynamically Smoother Motion for Industrial Robots, Design and Implementation,* IMechE, 1984.

[1.7] E. Isaacson and H.B. Keller, *Analysis of numerical methods,* NewYork, Wiley, 1996.

[1.8] J. Tood, *Survey of numerical analysis,* NewYork, McGraw-Hill, 1962.

[1.9] L.L. Schumaker, *Spline Function, Basic Theory,* NewYork, Wiley, 1981.

[2.1] G. Ammar, W. Dayawansa, C. Martin W, *Exponential Interpolation: Theory and Numerical Algorithms,* Applied Mathematics and Computation, V.41, pp. 189-232, 1990.

[2.2] E. Kreyszig, *Advanced Engineering Mathematics,* John Wiley and Sons, NewYork 1988.

[2.3] A. Dodes, *Numerical Analysis for Computer Science,* Elsevier North-Holland, Inc. 1978

[2.4] T.R.F. Nonweiler, *Computational Mathematics an Introduction to numerical approximation,* Ellis Horwood Ltd, England 1984

[2.5] E.Soylemez, *The Dynamic Synthesis, Analysis, and Design of Modelled Cam Systems,* Lexington Books, 1976.

[2.6] W. Cheney, D. Kincaid, *Numerical Mathematics and Computing,* Brooks/Cole Publishing Company California USA, 1985.

[2.7] Schwarz H.R. *Numerical Analysis, a Comprehensive introduction,* John Wiley and Sons Ltd, New York, U.S.A, 1989.

[2.8] I. Jacques and C.judd, *Numerical Analysis*, Chapman Hall, NewYork, USA, 1987

[2.9] L.W. Johnson and R.D Riess, *Numerical Analysis*, Addison-Wesley Publishing Company, Inc. U.S 1987.

[2.10] X. Pintado, E. Fiume, *Grafields: Field-Directed Dynamic splines for Interactive Motion Control*, Computer and Graphics, V.13, N.1, pp. 77-82, 1989.

[2.11] A.K. Cline, *Scalar-and Planar Valued Curve Fitting Using Splines Under Tension*, Communications of the ACM, V.17, N.1, pp. 218-223, 1974.

[2.12] D. G. Schweikert, *An Introduction curve Using a Spline in Tension*, Journal of Maths. And Physics, V.45, pp. 189-312-317, 1966.

[2.13] Y. Fletcher and D.F. McAllister, *Automatic Tension Adjustment for Interpolatory Splines*, IEEE Computer Graphics and Applications, pp. 10-17, January 1990.

[2.14] A D. Luca, L.Lanari and G.Oriolo,*A sensitive Approach to Optimal Spline Robot Trajectories*, Applied Mathematics and Computation, V.41, pp. 189-232, 1990.

[2.15] M.K. Jouaneh, Z. Wang and D.A. Dornfeld, *Trajectory Planning for Co-ordinated Motion of Robot and a Positioning Table: Part 1-Path Specification*, IEEE Transactions on Robotics and Automation, V.6, N.6, pp. 735-744, 1990.

[2.16] M.K. Jouaneh, D.A. Dornfeld and M. Tomizuka, *Trajectory Planning for Co-ordinated Motion of Robot and a Positioning Table: Part 2-Path Specification*, IEEE Transactions on Robotics and Automation, V.6, N.6, pp. 746-759, December 1990.

[2.17] T.A. Foley, *Interpolations with Interval and Point Tension Controls Using Cubic Weighted V-Splines*, ACM Transactions on Mathematical Software, V.13, N.1, pp. 68-96, 1987.

[2.18] B.L. MacCarthy and N.D. Burns, *An Evaluation of spline Function for Use in Cam Design*, IMecE, V.199, N.C3, 1985.

[2.19] B.L. MacCarthy, *Quntic Splines for Kinematic Design*, Computer Aided Design, V.20 N.7, 1988.

[3.1] G.W. Vernon, Motion Design, *Control and Implementation in Robot Manipulators*, PhD. Thesis, School of Engineering and Technology Management, Liverpool Polytechnic, 1988.

[3.2] K.J. Stamp, *The Mechanism of Carton Erection*, PhD. Thesis, School of Engineering and Technology Management, Liverpool Polytechnic, 1989.

[3.3] W.M. Dudley, *A New Approach to Cam Design*, Machine Design, pp. 143-160, July 1947.

[3.4] A.D. Stoddart, *Poldyne Cam Design*, Machine Design, 121-136, Jan. 1953.

[3.5] A.D. Stoddart, *Poldyne Cam Design*, Machine Design, 146-154, Feb. 1953.

[3.6] A.D. Stoddart, *Poldyne Cam Design*, Machine Design, 149-164, March. 1953.

[3.7] D.Tesar and G.K.Matthew, *The Dynamic Synthesis, Analysis, and Design of Modelled Cam Systems*, Lexington Books, 1976.

[4.1] J.E. Bobrow, S. Dubowsky and J.S. Gibson, *Time-Optimal Control of Robotic Manipulators Along Specified Paths*, The international Journal of Robotics Research, V.4, N.3, pp. 3-17, 1985.

[4.2] K.G. Shin and N.D. McKay, *A Dynamic Programming Approach To Trajectory Planning of Robotic Manipulators*, IEEE Transactions on Automatic Control, V.AC-31, N.6, pp. 491-500, June 1986.

[4.3] S.K. Singh and M.C. Leu, *Manipulator Motion Planning in the Presence of Obstacles and Dynamic Constraints*, The International Journal of Robotics Research, V.10, N.2, pp. 171-187, April 1991.

[4.4] E.G. Gilbert and D.W. Johnson, *Distance Functions and Their Applications to Robot Path Planning in the presence of Obstacles*, IEEE Journal of Robotics and Automation, V.RA-1, N.1, March 1985.

[4.5] A.M.S. Zalzala and A.S. Morris, *Distributed robot Control on Transputer Network*, IEE Proceedings-E, V.183, N.4, pp. 169-176, July 1991.

[4.6] Z. Shiller and S. Dubowsky, *Robot Path Planning with Obstacles, Actuator, Gripper, and Payload Constraints*, The International Journal of Robotics Research, V.8, N.6, pp. 3-17, December 1989.

[4.7] L.V. Aken and H.V. Brussel, *On-line Robot Trajectory Control in Joint Co-ordinates by Means of Imposed Acceleration Profiles*, Proceeding of the 15'th International Symposium of Industrial Robots, 1003-1010,1984.

[4.8] L. Weiping and J.J.E. Slotine, *A Unified Approach to Compliant Motion Control*, Proceedings 1989 American Control Conference, V.3, pp. 1944-1949, 1989.

[4.9] J.J. Craig, *Introduction to Robotics*, Addison-wesley publishing company 1989.

[4.10] A. Ekrekli, *Control and Identification of a Two Link Robot Manipulator*, Ph.D Thesis, University of Liverpool, March 1992.

[4.11] S. Chand and K.L. Doth, *On-Line Polynomial Trajectories for Robot Manipulators*, The International Journal of Robotics Research, V.4, N.2, pp. 38-48, Summer 1985.

[4.12] C.S. Lin, P.R. Chang and J.Y.S. Luh, *Formulation and Optimization of Cubic Polynomial Joint Trajectories of Industrial Robots*, IEEE Transactions on Automatic Control, V.AC-28, N.12, pp. 1066-1074, December 1983.

[4.13] K.G. Shin and N.D. McKay, *Selection of Minimum Time Geometric Paths for Robotic Manipulators*, IEEE Transactions on Automatic Control, V.AC-31, N.6, pp. 501-511, June 1986.

[4.14] B.K. Kim and K.G. Shin, *Minimum-Time Path Planning for Robot Arms and Their Dynamics*, IEE Transactions on System, Man, and Cybernetics, V.SCM-15, N.2, pp. 38-48, March/April 1985.

[4.15] J. Angeles, A. Rojas and C.S.L. Cajun, *Trajectory Planning in Robotics Continuous-Path Applications*, IEEE Journal of Robotics and Automation, V.4, N.4, pp. 380-385, August 1988.

[4.16] F. Peiffer and R. Johanni, *A Concept for Manipulator Trajectory Planning*, IEEE Journal of Robotics and Automation, V.RA-3, N.2, pp. 115-123, April 1987.

[5.1] R. Wilson, *An Introduction to Dynamic Data Structures*, McGraw-Hill Ltd. 1988.

[5.2] N. Wirth, *Algorithms and Data Structures*, Prentice-Hall International, Inc. 1987.

[5.3] Daven, *An Introduction to Dynamic Data Structures*, McGraw-Hill Ltd. 1988.

[5.4] D. Osypiw, *High Speed Automatic Assembly of Fuselink*, Mphil/Phd. Transfer Report, School of Engineering and Technology Management, Liverpool Polytechnic, 1992.

[6.1] N.H.J. Udoakang, *Robots Aiding New Developments of Manipulative Machinery* , Phd. Thesis, Loughborough University of Technology, 1983.

[7.1] D.W. Clarke and P.J. Gawthrop, *Self-tuning Control*, PROC. IEE, V.126, N.6, pp. 633-640, June 1979.

[7.2] Q. Wang and D.R. Broome, *A New Simulation Scheme for Self-tuning Adaptive Control of Robot Manipulator*, Robotica, V.9, pp. 335-339, 1991.

[7.3] Mei-Hua Liu W. Lin, *Multivariable Self-tuning Control with Decoupling for Robotic Manipulators*, IEE proceedings, V.135, N.1, pp. 43-48, January 1988.

[7.4] K. Ogata, *Modern Control Engineering*, Prentice-Hall, Inc. 1990.

[7.5] A.M Zikic, *Practical Digital Control*, Ellis Horwood limited 1989.

[7.6] S.P. Banks, *Control Systems Engineering*, Prentice-Hall, Inc. 1986.

[7.7] P.E Wellstead and M.B. Zarrop, *Self-tuning Systems Control and Signal Processing*, John Wiley and Sons Ltd, 1991.

[7.8] M.A. El-Sharkawi S. Weerasooriya, *Development and Implementation of Self-tuning Tracing for DC Motors*, IEEE transactions on energy conversation, V.5, N.1, pp. 122-128, March 1990.

[7.9] R.E Kalman, *Design of a Self-optimizing Control Systems*, Trans. ASME, N.80, pp. 468-478, 1958.

[7.10] V. Peterka, *Adaptive Digital regulation of Noisy Systems*, IFAC Symposium on Identification and Process Parameter Estimation, Prague, 1970.

[7.11] K.J. Astrom B. Wittenmark, *On Self-tuning Regulators*, Automatica, N.9, pp. 185-199, 1973.

[7.12] J.K. Pieper, B.W. Surgenor and J.Z. Liu, *On Self-tuning Control of Processes with Time Varying Dead Time*, Proceedings of the 1991 American Control Conference, V.3, pp. 2166-2171, 1991.

[7.13] G. Mingkun and S. Zengqi, *A Force/Position Hybrid Self-tuning Control*, IEE, pp. 271-275 1988.

[7.14] A.P. Tzes, *Intelligent Self-tuning Controllers for Robot Manipulators*, IEEE, pp. 5-8, 1990.

[7.15] Z.S. Tumeh, *Hybrid Nonlinear Self-tuning Discrete Manipulator Control*, Proceedings of the 29 th Conference on Decision and Control Honolulu Hawaii, pp. 2648-2649, December 1990.

[7.16] B. McAskill and W.G. Dunford, *Self-tuning Pole Placement Control of a Manipulator with Flexible Joints*, PECS '88 record, pp. 445-451, April 1988.

[7.17] M.B. Zarrop, *A Generalised Pole Placement Self-tuning Controller*, IEE Fourth Workshop on Self-tuning and Adaptive Control, pp.(11-1,11-9), March 1987.

[7.18] Z.S. Tumek, *Hybrid Nonlinear Self-tuning Discrete Manipulator Control*, Proceedings of the 29 th Conference on Decision and Control Honolulu Hawaii, pp. 2648-2649, December 1990.

[7.19] D.E. Miller, E.J. Davison, *The Self-tuning Robust Servomechanism Problem*, IEEE Trans. on Automatic Control, V.34, N.5, pp. 511-523, May 1989.

[7.20] H.K. Song, S.L. Shah and D.G. Fisher, *A Self-tuning Robust Controller*, Automatica, V.22, N.5, pp. 521-531, 1986.

[7.21] M.T. Tham and A.J Morris, *An Introduction to Self-tuning Control*, IEE Fourth workshop on Self-tuning and adaptive control, pp. 1/1-1/32, 22 to 24 March 1987.

[7.22] K.J. Astrom B. Wittenmark, *Adaptive Control*, Addison-wesley publishing company, 1989.

[7.23] P.V. Osburn., H.P. Whitaker and A. Kezer, *New Developments in the Design of Adaptive Control*, Inst. Aeronautical Sciences, paper 61-39, 1961.

[7.24] C.C. Hang and P.C. Parks, *Comperative Studies of Model Reference Adaptive Control Systems*, IEEE Transactions on Automatic Control, V.AC-18, N.5, pp. 419-427, March 1990.

[7.25] M. Dressler, *An Approach to Model Reference Adaptive Control Systems*, IEEE Transactions on Automatic Control, V.AC-12, pp. 75-80, Feb. 1967.

[7.26] R.L. Butchard and B. Shackcloth, *Synthesis of Model Reference Adaptive Control Systems by Lyapunov's Second Law*, IFAC Symposium on Adaptive Control, pp. 145-152, England, 1966.

[7.27] R.V. Monopoli, J.W. Gilbart and W.D. Thayer, *Model Reference Adaptive Control Based on Lyapunov-like techniques*, IFAC Symposium System Sensitivity and Adaptivity, Yugoslavia, pp. 401-405, 1968.

[7.28] Z. Geng, J.D. Lee, R.L. Carroll and X. Lie, *Learning Control System Design for a Parallel Link Robot Manipulator*, Proceedings IEEE international symposium on Intelligent Control, pp. 102-114, 1989.

[7.29] Z. Geng, M. Jamshidi, *Expert Self-learning Controller for Robot Manipulator*, Proceedings of the 27th Conference on Decision and Control, pp. 1090-1095, 1988.

[7.30] N. Sadegh and K. Guglielmo, *A New Learning Controller for Mechanical Manipulators*, Proceedings 1989 American Control Conference (IEEE) V.3, pp. 2827-2833, 1989.

[7.31] R. Horowitz and W. Messner, *Learning Optimal Control Using Integral Equations*, Proceedings 1989 American Control Conference (IEEE) V.3, pp. 2147-2152, 1989.

[7.32] K. Ramamurthi and M. Ahmadian, *Model Referenced and Learning Model Adaptive Control for Robotic Manipulator*, American Society of Mechanical Engineers, Dynamic Systems and Control Division, pp. 345-353, 1987.

[7.33] E.G. Harokopos, *Optimal Learning Control of Mechanical Manipulators in Repetitive Motions*, Proceedings IEEE International Conference on Robotics and Automation, pp. 396-401, 1986.

[7.34] S. Arimoto, S. Kawamura and F. Miyazaki, *Bettering Operation of Robots by Learning*, J. Robotic Systems, V.1, pp. 123-140, 1984.

[7.35] S. Arimoto, S. Kawamura, F. Miyazaki and S. Tamaki, *Learning Control Theory for Dynamical Systems*, Proc. IEEE 24th Conference on Decision and Control, pp. 1375-1380, 1985.

[7.36] S. Kawamura, F. Miyazaki, and S. Kawamura, *Realization of Robot Motion Based on a Learning Method*, IEEE Transactions on Systems Man and Cybernetics, V.18, N.1, pp. 126-134, 1988.

[7.37] J.G. Ziegler and N.B. Nichols, *Optimum setting for Automatic Controllers*, ASME Trans. V.64, pp. 759-768, 1942.

[7.38] H.J. Park, H.S. Cho and S.R. Oh, *A Discrete Iterative Learning Control Method with Application to Electric Servo Motor Control*, Proceedings 1989 American Control Conference (IEEE), V.3, pp. 2640-2645, 1989.

[7.39] C.S. Williams, *Designing Digital Filters*, Prentice-Hall, Inc. 1986.

[7.40] A. Antoniou, *Digital Filters Analysis and Design*, McGraw-Hill, Inc. 1979.

[7.41] S.M. Bozic, *Digital and Kalman Filtering*, Edward Arnold Ltd. 1979.

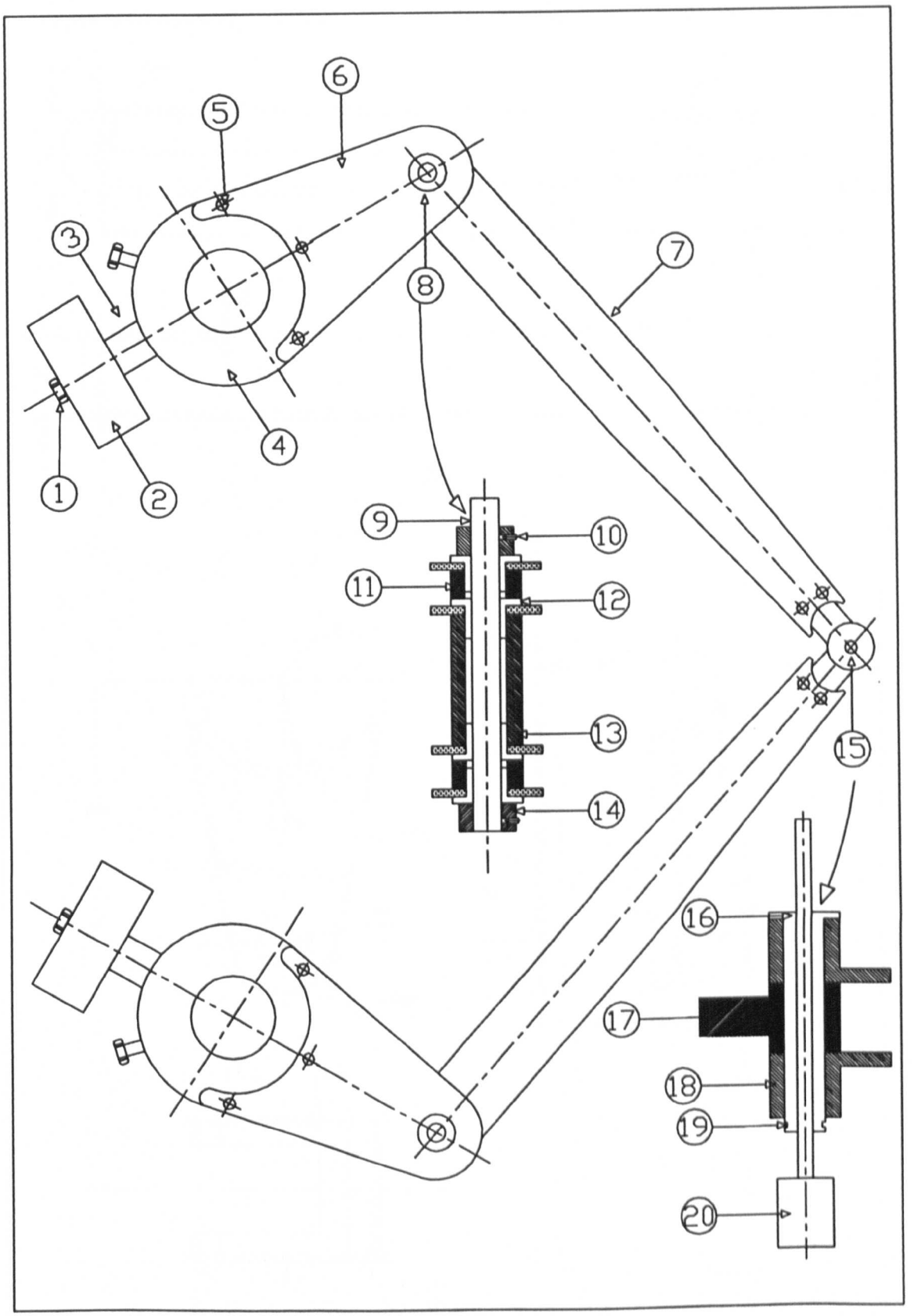


Fig.A.1. Assembly drawing of the arrangement.

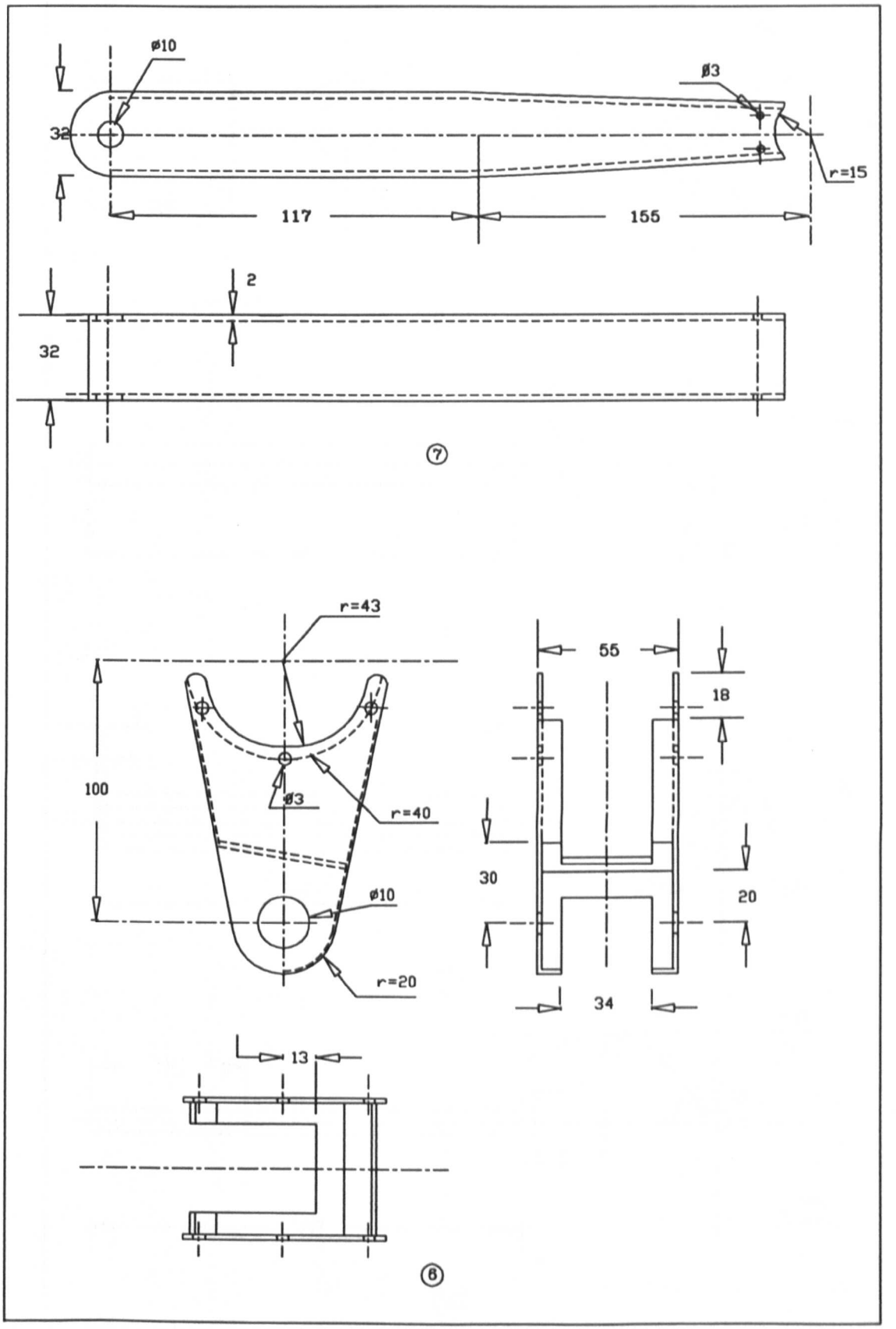


Fig.A.2. Technical drawings of the carbon-fibre crank and coupler link.

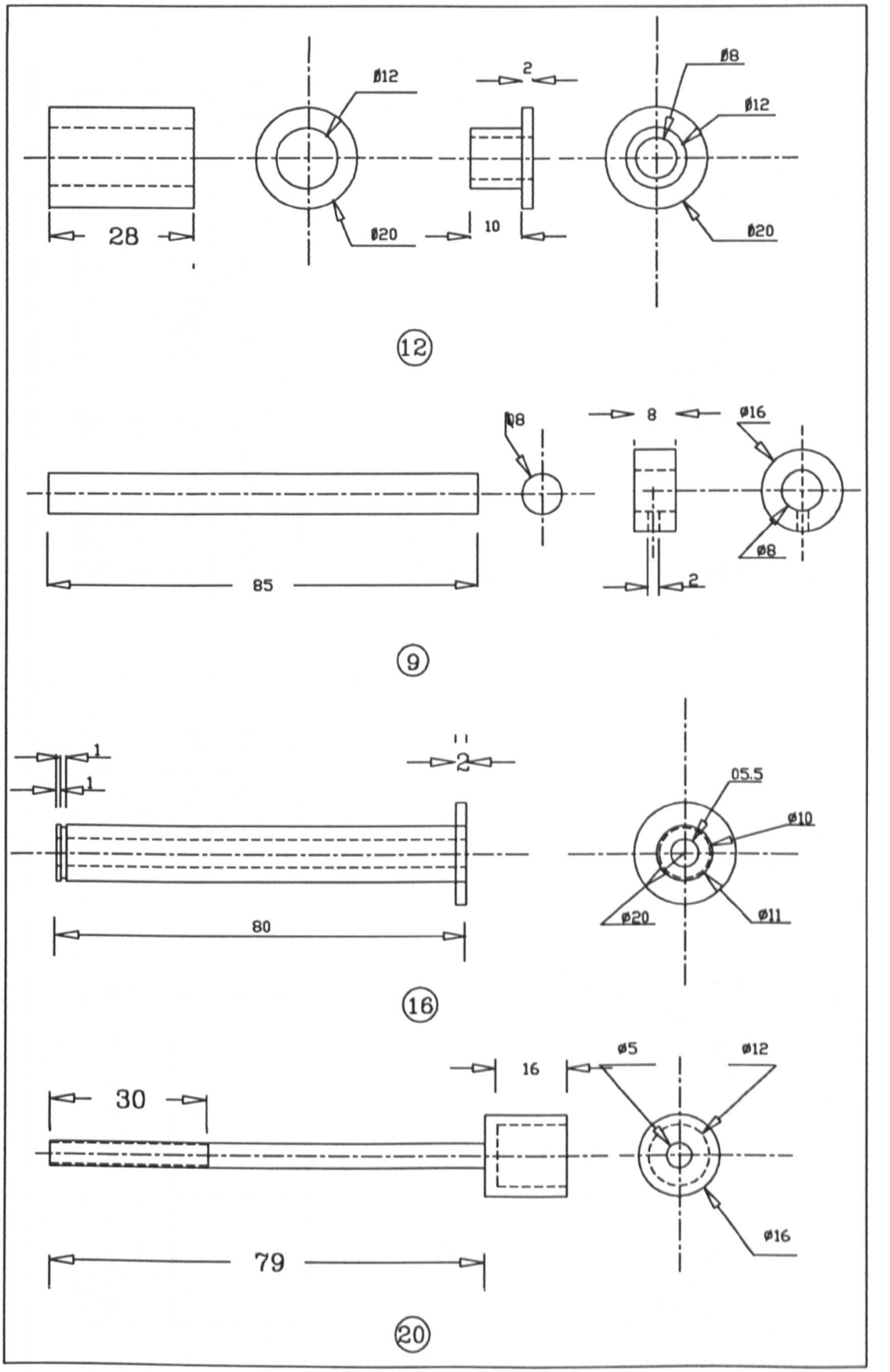


Fig.A.3. Technical drawings of bushes and pen-holder.

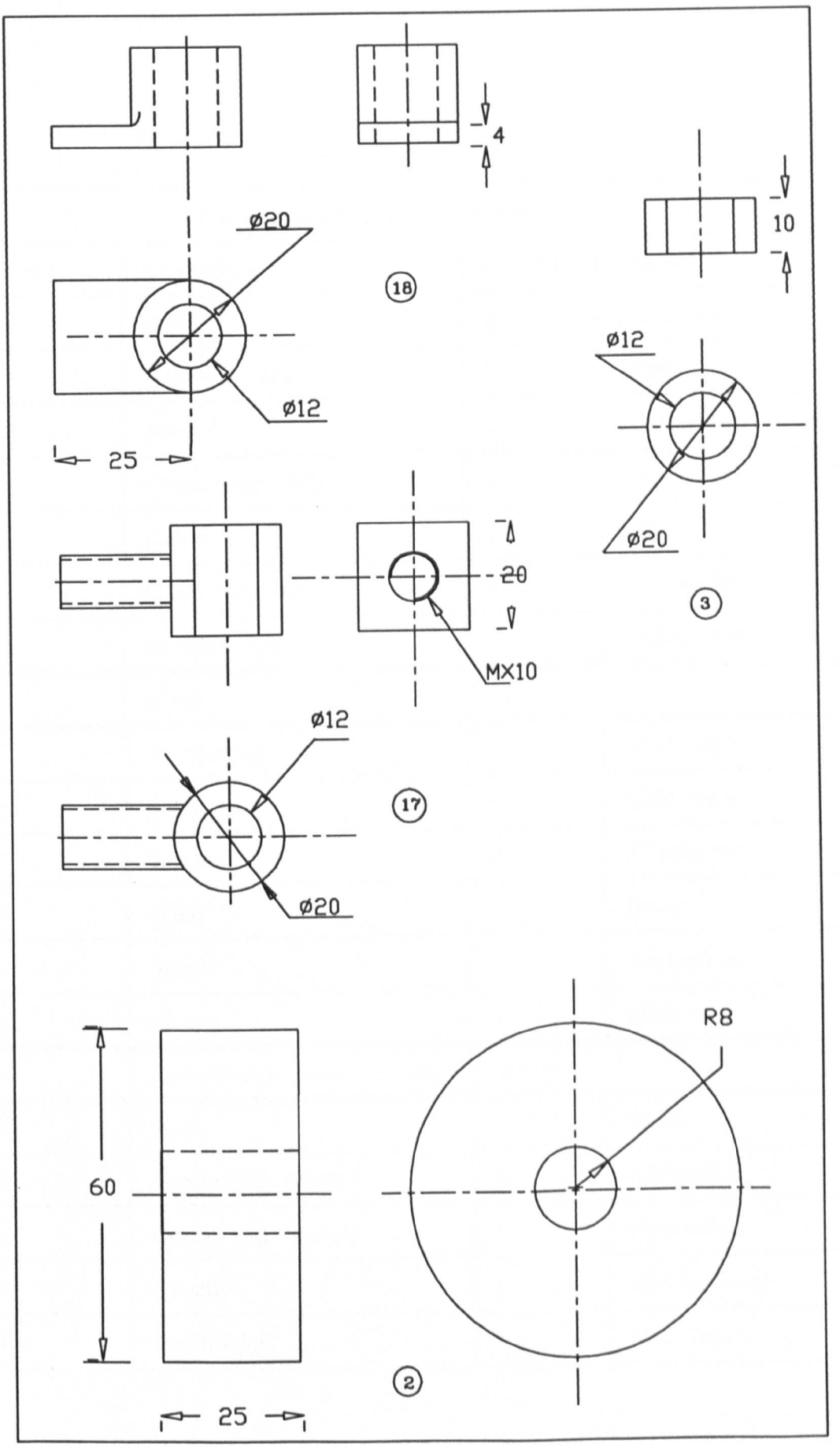


Fig.A.4. Technical drawings for the end–effector and balancing weights.

Table A.1 materials for the mechanism			
Part no	Description	Quantity	Material
1	Bolt	2	Mild steel
2	Balancing mass	2	Mild steel
3	Spacer	1	Aluminium
4	Connecting bush	2	Aluminium
5	Screw	16	Mild steel
6	Connecting arm	2	Carbon-fibre
7	Coupler link	2	Carbon-fibre
8	Pivot		
9	Connecting rod	2	Mild steel
10	Allen bolt	4	Mild steel
11	Washer	2	Aluminium
12	Bush	8	Brass
13	Spacer	2	Aluminium
14	Sleeve	4	Mild steel
15	End-effector pivot (double lap pivot)		
16	Bush	1	Brass
17	Connecting block-1	2	Aluminium
18	Connecting block-2	1	Aluminium
19	Cir-clip	1	Carbon steel
20	Pen-holder	1	Mild steel

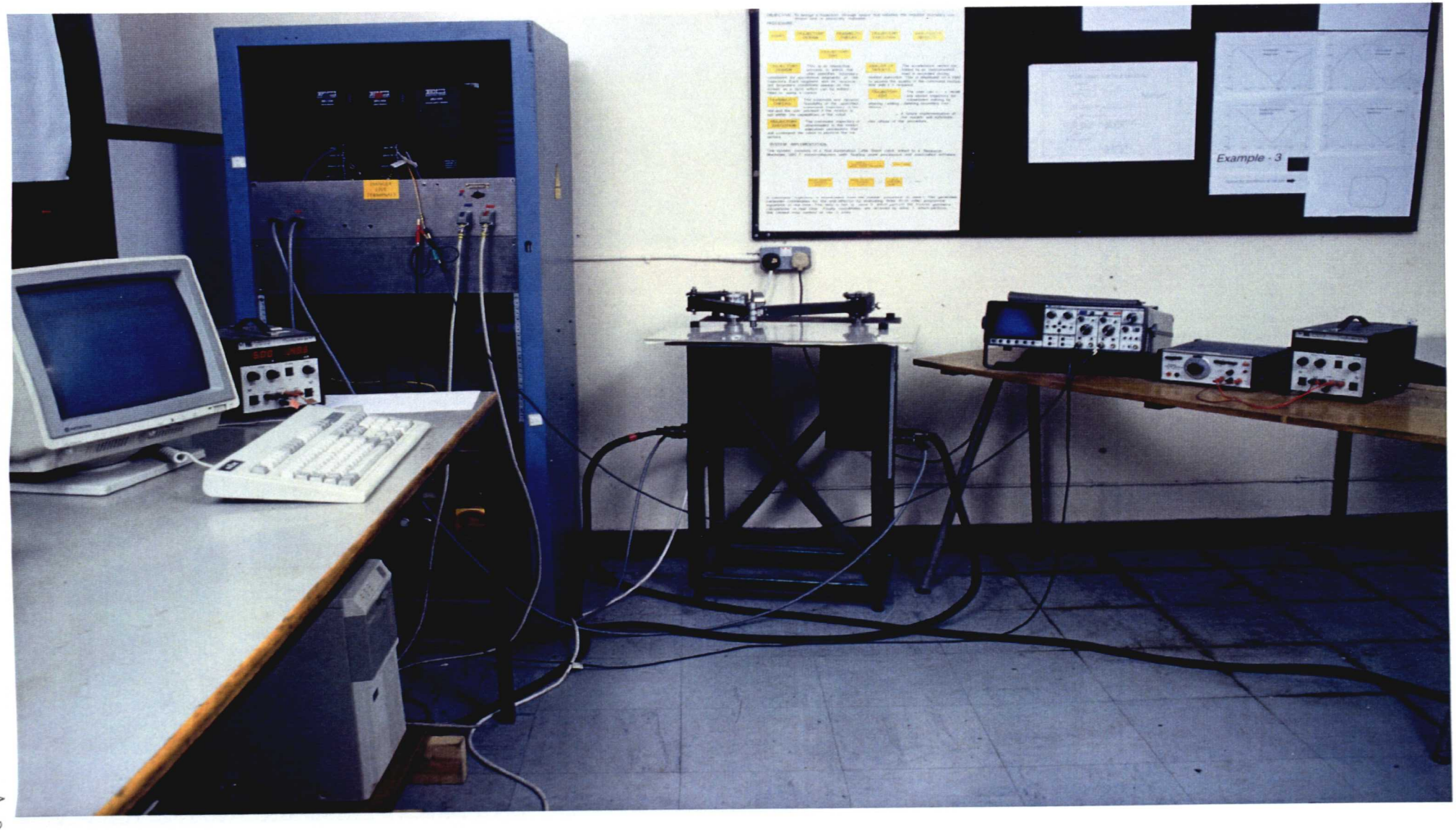


Plate 1. The experimental arrangement used in the study.

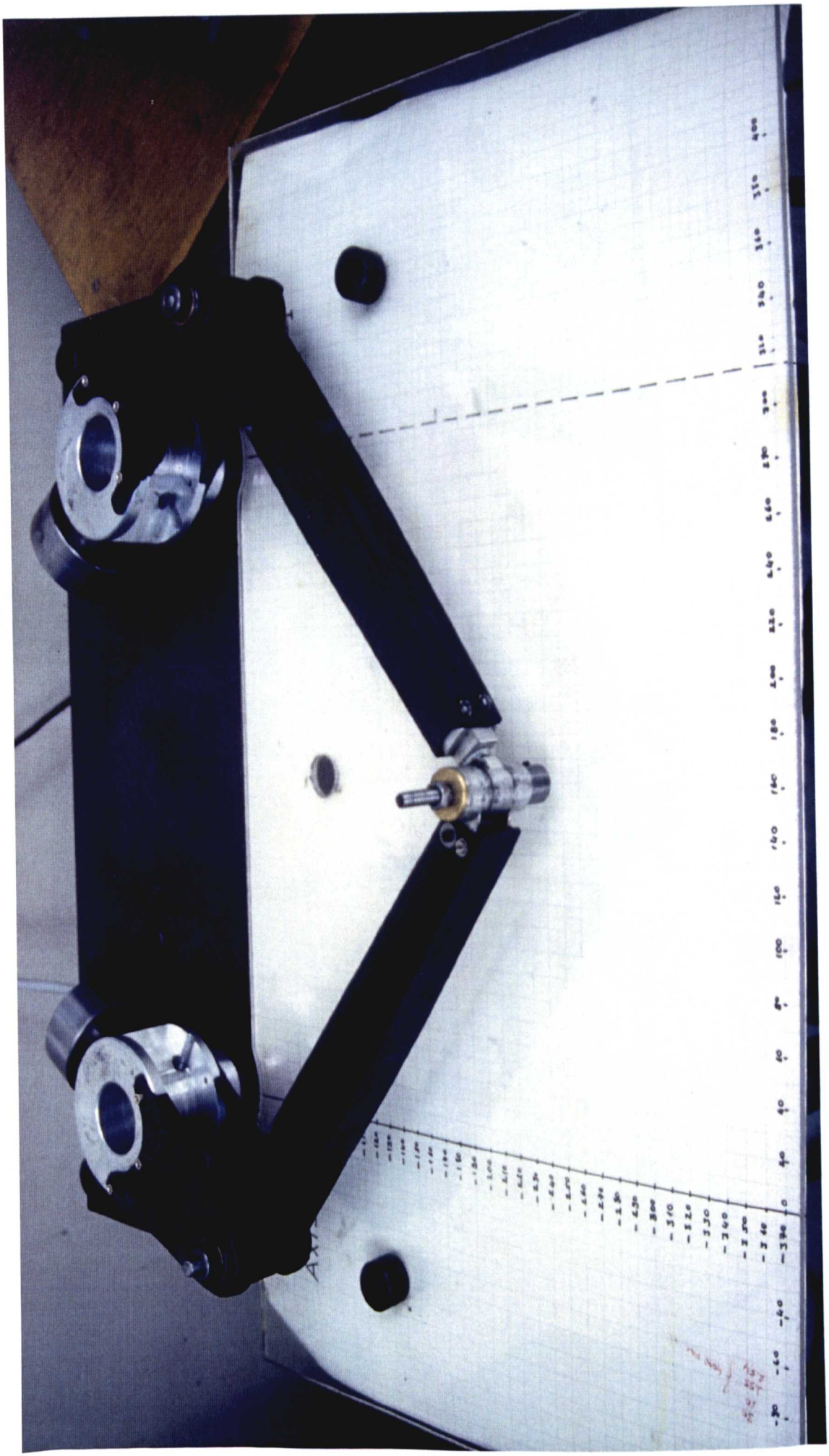


Plate 2. A view of the five-bar manipulator.