The Synthesis of Hybrid Mechanisms Using Genetic Algorithms

by

Andrew Miles Connor

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

Mechanisms & Machines Research Group
Liverpool John Moores University

October 1996

I have undertaken a labour, a labour out of love for the world and to comfort noble hearts : those that I hold dear, and the world to which my heart goes out. Not the common world do I mean, of those who (as I have heard) cannot bear grief and desire to bathe in bliss (May God then let them dwell in bliss!). Their world and manner of life my tale does not regard : its life and mine lie apart. Another world do I hold in mind, which bears together in one heart its bitter sweetness and its dear grief, its heart's delight and its pain of longing, dear life and sorrowful death, dear death and sorrowful life. In this world let me have my world, to be damned with it, or to be saved.

Gottfried Von Strassburg

Table of Contents

Acknowledgements

Firstly, I must thank Steve Douglas who, as my Director of Studies, has guided m y footsteps whenever they have faltered and has given invaluable assistance without which this PhD could not have been completed. Similar thanks must also go to Mike Gilmartin, who's efforts have been in no way less useful, and George Rooney who advised me concerning certain aspects of motion design. I must also thank Prof. Mike Lalor and Graham Alexander for their generosity during funding difficulties.

Other people have contributed to the success of this project. Most notable are the technicians and librarians at LJMU, the efforts of whom are normally not recognised. In particular, I must thank Paul Wright, Steve Ebbrell, Tony McKenna, Steve Gotts and John Carrier for their advice concerning the design and installation of the experimental rig described in Chapter Five. I must also thank Colin Pryor and Len Geekie of Quin Systems for bringing their experience of servo systems to my aid during the trouble shooting period of the installation.

Finally, I would like to thank all of my friends and family who have endured the trials and tribulations presented by my obsession with the completion of this thesis, who now have the pleasure of seeing life return to at least some vestige of normality.

To one and all, many thanks.

Abstract

This thesis presents a novel design methodology for the synthesis of hybrid mechanisms using Genetic Algorithms. GAs are a search and optimisation method which model the mechanics of population genetics to give a truly global search method.

In parallel to the development of a suitable GA, the work also develops novel objective function criteria which go some way to providing an approximation to dynamic criteria whilst using only kinematic properties during calculations. This has considerable effect in reducing the time required to find a feasible solution.

The thesis presents a set of results which validate the proposed methodology, both in terms of speed of convergence and quality of the final solutions obtained. The application chosen is the synthesis of a hybrid five bar path generating mechanism.

A description is given of the development of a practical machine for a given test case, so as to illustrate that the solutions produced are feasible in terms of real world implementation. Results are presented which show the effectiveness of the machine.

Finally, a critical analysis of both the methodology and the results is carried out. This highlights some areas in which the methodology could be improved by future work.

List of Symbols and Notation

CV	Constant Velocity
P	Power
GA	Genetic Algorithm
o(H)	Schema Order
$\delta(H)$	Schema Defining Length
f(H)	Schema fitness
p_s	Probability of Survival
L	Length of String
p_c	Probability of Crossover
p_m	Probability of Mutation
$\underline{p}, \underline{q}, \underline{r}, \underline{s}, \underline{t}$	Mechanism Link Vectors
θ_i	Angular Displacement of Link
ω_i	Angular Velocity of Link
α_i	Angular Acceleration of Link
x, y	Cartesian Coordinates
Z	Kinematic Constant
Q	Generalised Force
K	Kinetic Energy
U	Potential Energy
I	Inertia
ϕ_i	Contribution of Link i to Generalised Force
T	Torque
D	Diameter
F_L	Friction Load
F_M	Friction of Motor
V	Voltage
K_P	Proportional Gain
K_I	Integral Gain
K_D	Differnetial Gain

K_V	Velocity Feedback Gain
K_F	Velocity Feedforward Gain
e_i	Position Error
d_i	Demand Position
p_i	Measured Position

Chapter One

Introduction

1.1 Project Background

This thesis presents a novel methodology for the synthesis of multi-degree of freedom mechanisms, with particular emphasis on hybrid machine applications. A mechanism is a device that transforms a given input motion to a desired output motion. A machine is a device that contains mechanisms, provides complex output motions and transmits significant forces and power.

Traditionally, mechanisms were driven by motors that operate at constant velocity and have only possessed a single degree of freedom. Because of this the mechanism has driven the load with a fixed output motion. To change the output motion required a change in the mechanism itself. These changes involve the manual adjustment of machine components and resulted in large down times for production.

1.1.1 The Need for Flexibility

The nature of the modern production environment has introduced the demand that flexibility must be high and that down time must be low for a manufacturer to remain competitive. This has resulted in changing the way in which machinery is designed. Conventional design methods consist of inserting cams and linkages into the driving mechanisms. Whilst a degree of flexibility is given by having interchangeable parts, this results in high down times. However, traditional machines have good transmission and the potential for energy regeneration.

In an attempt to achieve high flexibility and low down time, programmable electric motors have been used to drive the output directly, as opposed to through a linkage.

Although there are a variety of programmable motors, the basic principles of operation are similar. The output motion is generated by varying the potential difference across a rotor that is surrounded by a magnetic field. The rotor then experiences a torque due to the changes in magnetic flux. Altering the rate at which the voltage changes allows the speed, and direction, of the output shaft to be regulated to suit the output motion requirements. The constantly changing current in the coils of the motor results in the generation of considerable heat. This manifests itself in a higher power rating for the motor and can, occasionally, lead to damage within the circuitry of the motor. This is particularly true for applications that follow non-uniform trajectories and demand high rates of change of torque.

Generally, a programmable motor used in conjunction with closed loop control is referred to as a servo motor. Servo motor systems have high flexibility but may suffer from poor energy regeneration. Constant velocity (CV) motors, and traditional machines, have good transmission and the potential for good energy regeneration. This is often achieved by the inclusion of a flywheel on the drive shaft. An approach that combines the desirable characteristics of both systems is the use of a hybrid machine.

1.1.2 Hybrid Machines

The initial concept of a hybrid machine investigated by **Tokuz** [1,2], who developed a system in which the outputs of a constant velocity motor and a programmable servo motor were combined through a differential gearbox to drive a slider-crank mechanism. Figure 1.1 shows a schematic diagram of the experimental rig. The aim of the work was to consider the efficiency of the hybrid rig with the case where the servo motor drives the slider-crank directly. This is an example of a programmable machine, as opposed to a hybrid machine.

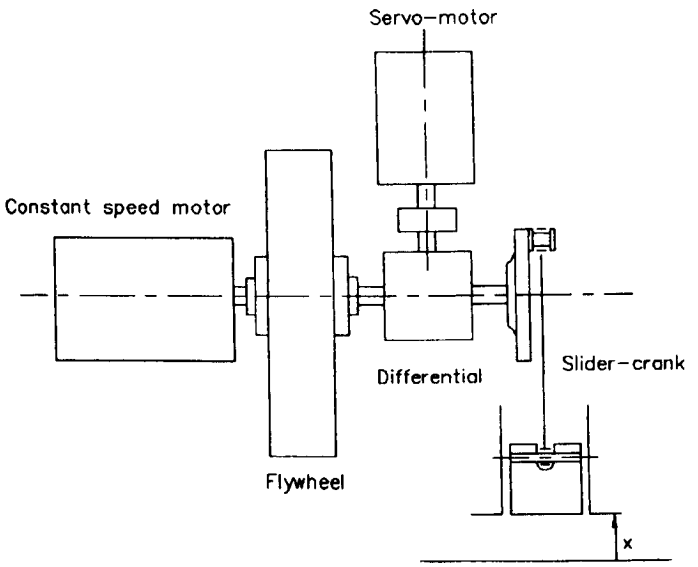

Figure 1.1 : Hybrid Arrangement

Several different slider motions were investigated, including a Rise-Return (R-R), a Rise-Dwell-Return (R-D-R) and a Rise-Return-Dwell (R-R-D). These motions were designed using polynomial laws to define segments of the curve and boundary conditions were selected to ensure continuity of derivatives across segments. The three motions are shown in Figure 1.2.

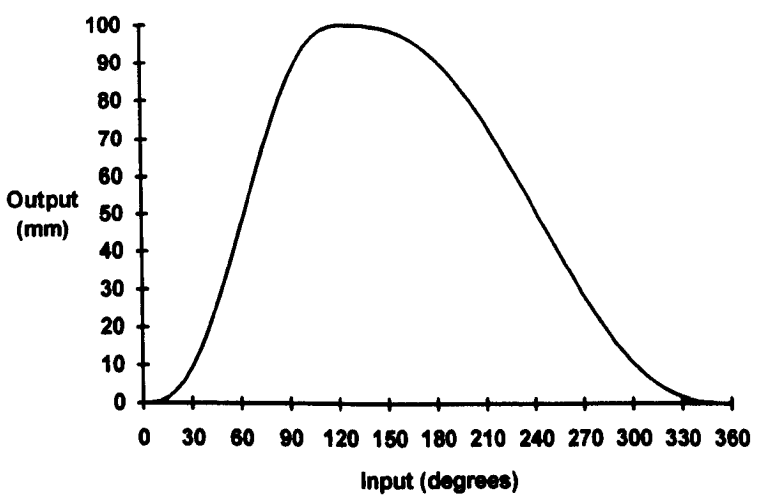

Figure 1.2a : Rise-Return Motion

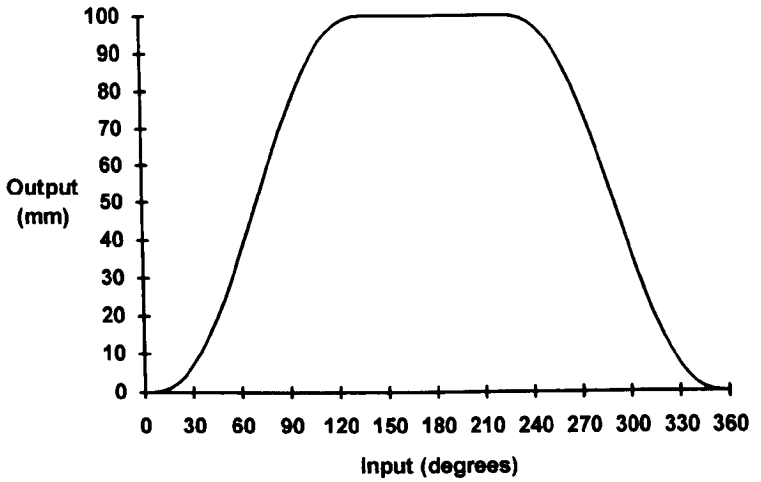

Figure 1.2b : Rise-Dwell-Return Motion

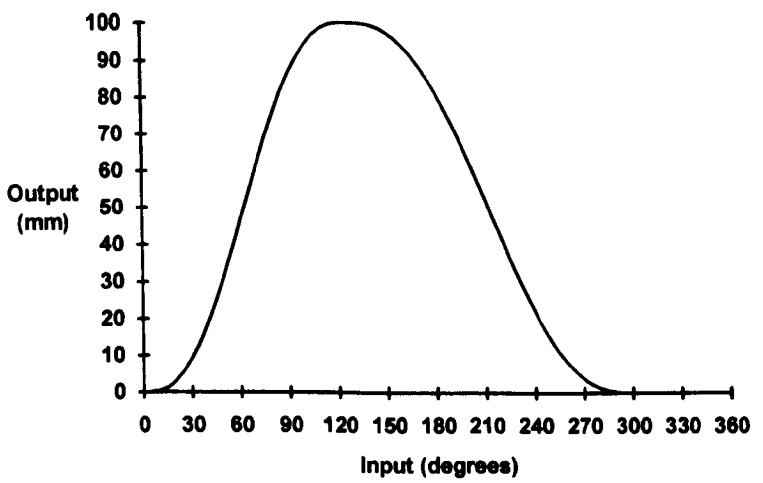

Figure 1.2c : Rise-Return-Dwell Motion

For the R-R motion, the hybrid approach showed considerable power savings over the programmable machine. However, for the other motions the hybrid machine required more power to produce the output motion. This can be attributed to the dwell in both the R-D-R and the R-R-D motions. To obtain a dwell, the servo motor must oppose the motion of the constant velocity motor directly. This produces high currents in the servo motor and this energy must be dissipated as heat. This leads to the high power consumption. Figure 1.3 shows the comparison of power

consumption (watts) for both machine configurations in each of the three motion cases.

	R-R	R-D-R	R-R-D
Hybrid P_{max}	22.02	35.47	33.49
Hybrid P_{min}	-47.97	-40.68	-56.46
Programmable P_{max}	82.83	13.64	25.48
Programmable P_{min}	-131.77	-19.32	-35.11

Figure 1.3 : Peak Power Values

Later work by **Greenough** [3] and **Bradshaw** [4] attempted to overcome the difficulties of producing complex output motions by replacing the differential gearbox with a multi-degree of freedom linkage to act as a non-linear gearbox. In this arrangement, the servo motor need not directly oppose the constant velocity motor to obtain a dwell.

This work investigated the case of a hybrid machine designed to have two fully rotational angular inputs and one fully rotational angular output. After researching several alternatives, a Svoboda type II mechanism was used. A schematic diagram of this mechanism is shown in Figure 1.4. In using this as a hybrid mechanism, the input θ_1 is provided by the CV motor, whilst θ_3 is provided by the programmable servo motor. The desired output function is generated as θ_2.

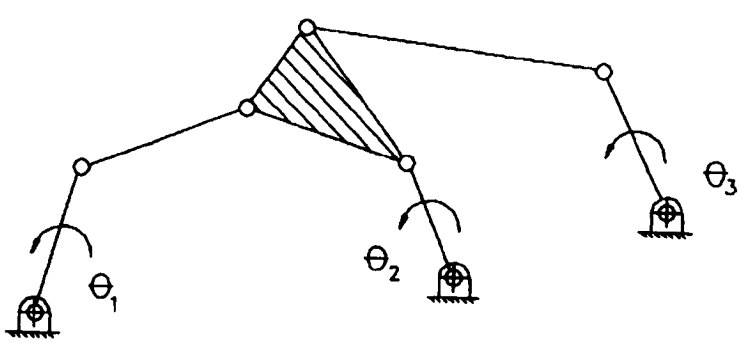

Figure 1.4 : Svoboda Type II Mechanism

The aim of the research was to investigate issues concerning the design, control and applications of hybrid machines of this configuration. One particular application was for a cut to length machine. Figure 1.5 shows a diagram of such a machine.

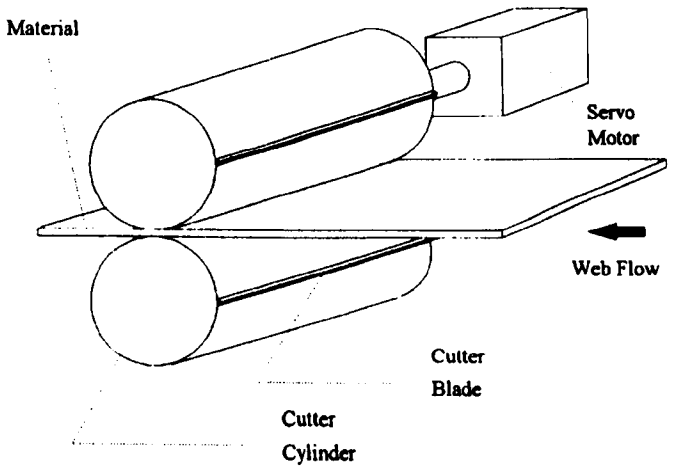

Figure 1.5 : Cut To Length Machine

The machine consists of a pair of high inertia cylinders, each of which is driven by a programmable servo motor. Each cylinder has a blade attached to it. A shearing action is produced when the blades meet which cuts the material. By matching the speed of the blade to the linear speed of the web, a high quality cut is produced.

The programmability of the machine is provided by the motions with which the cylinders are driven, as each different cut requires a unique cylinder motion. In this case, the programmable servo motors were replaced by a hybrid machine module to investigate any potential advantages to the system. The hybrid machine modulates the speed, but not the direction, of the cylinders to ensure that the desired speed matching occurs.

Introducing the hybrid machine module reduced the peak power by approximately 70%. There was also a slight increase in speed of operation. The machine module was also tested on a number of intermittent motions, such as the R-D-R and R-R-D motions. In these cases the power savings were less significant.

Whilst the results obtained were significantly better than previous work, problems were encountered in several areas. Firstly, the automatic design method used to calculate the link length and motion profile requirements was susceptible to locating sub-optimal solutions due to the highly non-linear solution space. Secondly, problems were encountered with the control algorithms used in the machine module. This was particularly noticeable at high speeds of operation. The final problem encountered in this study was the degree of flexibility offered by the machine module. The hybrid machine was capable of producing a range of motions with similar characteristics. However, it was found that a single mechanism was not able to produce a wide variety of motions.

Hybrid machines are a novel machine design solution currently unique to the Mechanisms & Machines Research Group at Liverpool John Moores University. Whilst previous work has settled some of the issues concerning hybrid machine design and application, there is still not a complete understanding of the benefits of the approach. Considerable research is still required if the hybrid concept is to become an acceptable industrial solution for the generation of non-uniform motion. The current work is intended to provide the next step in the process of turning hybrid machines into an industrially viable machine option by investigating a robust design methodology.

1.2 Project Objectives

The main objective of this research is to tackle the first problem indicated above. The development of a robust automatic design method is essential if the hybrid machine concept is to be turned into an industrially viable machine design option. **Greenough** [5] suggests some areas in which work could be concentrated, including

- Improving synthesis results
- Objective function criteria
- Machine balancing

The aim is to develop a robust design methodology for multi-degree of freedom mechanisms with specific applications in the field of hybrid machine design. The new methodology is intended to cover the first two of the above suggestions. In particular, it is intended that a robust optimisation strategy will be investigated along with additional design objectives which complement those already developed in previous work.

It is proposed that using frequency domain information will enable the objective function to be simplified so that the effectiveness of the search will be increased and the time taken to find a solution is reduced.

1.2.1 The Design Methodology

The entire design methodology is based around a simple statement of the desired optimal condition for a hybrid machine. This optimal condition is when the bulk of the motion is provided by the CV motor. When this occurs the power consumption of the servo motor will be much lower than that of the CV motor. This allows for considerable energy regeneration potential by including a flywheel on the CV drive shaft. The action of the servo motor provides a degree of modulation to the motion and hence introduces a degree of flexibility to the system.

The conditions that are necessary for the servo motor power consumption to be low are dependent on the kinematic properties of the mechanism and servo motor motion profiles as well as dynamic considerations such as load inertia and link inertias. Rather than utilise a computationally expensive dynamic design objective, the methodology attempts to approximate dynamic optimality using kinematic properties only.

It is assumed that the dynamic performance of a hybrid mechanism is a function of the kinematic properties of the servo motor motion profiles. The methodology is based around utilising a Fourier transform in the synthesis routine to ensure that the

displacement profiles have a low harmonic content. As the magnitude of high order harmonics is diminished, the servo motor motion trends towards simple harmonic motion. As this trend continues, acceleration profiles will trend towards being smooth and continuous. By combining this with the effect of other design objectives a mechanism with low torque requirement is found.

The actual search technique used in the methodology is a Genetic Algorithm. Genetic Algorithms were chosen from a number of alternative AI and heuristic based algorithms as they have been shown to provide an effective search of complex and discontinuous functions and are ideally suited to mechanism synthesis problems which have a highly non-linear search space. The methodology acts as a "black box" so that once implemented it can be used by non-specialist machine designers who do not have experience of the use of optimisation methods based around Genetic Algorithms. This is a very important consideration if the method is to become accepted in an industrial environment.

1.3 Thesis Structure

This thesis is split into seven Chapters that describe the overall research program and the development of the resulting design methodology. Chapter Two contains a summary of a comprehensive literature search that was carried out throughout the duration of the project. The aim of this Chapter is to highlight existing work and illustrate how the new methodology fits into the existing knowledge base. Chapter Three gives an introduction to the AI search method used in the design methodology, namely Genetic Algorithms. Chapter Four shows how a theoretical analysis of a mechanism can be derived and used to implement the methodology as a practical search tool. This Chapter also presents results of the method using different design objectives and illustrates the effectiveness of the final multi-criteria objective on several examples. Chapter Five presents results obtained from a local search method which uses the solutions found by the Genetic Algorithm as a seed. This Chapter also outlines the construction of the experimental test rig. Chapter Six is a discussion of

the results of the project in relation to the project objectives. Finally, Chapter Seven concludes the thesis and also includes a number of suggestions for possible future research in this field. The Appendices include the references cited in the thesis, program code for the Genetic Algorithm and synthesis objective function and copies of published papers concerning early work in this area.

Chapter Two

Literature Review

2.1 Introduction

This Chapter reviews some of the current literature in the fields of mechanism design and novel optimisation techniques. Essentially, the review of this literature can be split into two sections. The aim of the first section is to highlight previous work in mechanism synthesis and optimisation to show the trend towards more general and more robust methods.

The second section compares optimisation techniques, including novel methods which could be applied to mechanism design problems to produce an improvement over existing methods. A more detailed review is given for Genetic Algorithms, the method selected for use in this study.

2.2 Mechanism Design, Analysis and Synthesis

This section is split into several sub-sections, each of which deals with a different aspect of mechanism design. Section 2.2.1 deals with the traditional approach to mechanism design and defines the objectives of the distinct stages of type synthesis and dimensional synthesis. Section 2.2.2 deals exclusively with the automation of type synthesis. This has no direct relationship to this study but is a very important aspect of machine design. Section 2.2.3 outlines several methodologies for the synthesis of single degree of freedom mechanisms. This is not an exhaustive selection as a plethora of techniques have been developed, but those reviewed here have been chosen to highlight the current trends in mechanism design. Section 2.2.4 considers several additional objectives in machine design which have been suggested for inclusion in the synthesis of mechanisms to improve the overall quality of the

final design. Finally, section 2.2.5 deals specifically with multi-degree of freedom mechanisms and covers both the analysis and synthesis of this type of mechanism. However, it is important to state that very little work has been carried out in this area due to the inherent complexities associated with design and control of multi-degree of freedom systems.

2.2.1 Mechanism Design : The Traditional Approach

In general, mechanisms are designed to fulfil one of three roles, either path, motion or function generation. Path generation is defined as the control of a point in a plane so that it traces a desired trajectory. Path generation problems are essentially a subset of motion generation problems which are defined as the control of a line in a plane so that it assumes a desired set of sequential positions. Function generation is defined where the motion of an output member of a mechanism corresponds to a desired function.

These definitions are entirely independent of mechanism type and so the first stage to the design of a mechanism is known as type synthesis. Type synthesis is the selection of a mechanism type and configuration based on the consideration of the defined motion requirement. For example, there are a variety of ways in which a linear (reciprocating) function may be generated, including the use of straight line linkages, slider-crank linkages and cam-follower arrangements. Essentially, the purpose of type synthesis is to decide which configuration is best able to generate the desired motion.

Once a mechanism type has been selected it is then necessary to choose the desired link lengths so that the actual output motion meets the desired output motion requirements. This stage is known as dimensional synthesis.

The traditional approach to mechanism design utilised the experience of the designer to select a mechanism configuration in the type synthesis phase and graphical

methods in the dimensional synthesis. These graphical methods typically enabled the mechanism link lengths to be found for a motion defined by up to three precision points. Once the link lengths were specified, it would be possible to fully analyse the motion of the mechanism. If the output of the mechanism did not satisfactorily approximate the desired motion then the whole process would be repeated.

Some graphical synthesis methods have been developed for more than three positions, though they are particularly complex. Other techniques for dimensional synthesis were developed which include both analytical and numerical synthesis. Some systems have also been developed for type synthesis of mechanisms. Each of this areas will now be considered in greater depth.

2.2.2 Automated Type Synthesis of Mechanisms

Several researchers have investigated the possibility of encapsulating designer experience in an expert system approach to automate the type synthesis procedure. Examples of this include the TYSES program developed by **Thomson *et al*** [6,7] where the user interface consists of a series of questions concerning the nature of the desired motion. The software then uses graph theory to propose different mechanism configurations.

Several attempts have been made to catalogue mechanism coupler curves and use a selection process to choose a mechanism which generates the desired output. One of the most recent, and most effective, is the CAMFORD package developed by **McGarva & Mullineux** [8,9,10]. A variety of different mechanism configurations, with a variety of different link ratios were analysed and the output motions expressed as a set of discrete points. These output motions were then stored in a library as a set of Fourier coefficients which take into account differences in scale, rotation etc. The user interface consists of a sketch pad editor where the user defines a desired motion in terms of a set of points. This is then converted into the Fourier form and the library searched for close matches. The program then displays a number of

mechanisms which may be used to approximately generate the motion. One of these may then be chosen for use in an optimisation routine which uses the gradient based method developed by **Powell** [11]. A similar method has been utilised by **Hoeltzel *et al*** [12,13] in the Pattern Matching Synthesis approach. In this method a Hopfield Neural Network has been used to classify digitised images of complete coupler curves (as opposed to a set of distinct points) and once again a selection is made by considering the correlation between the desired curve and those in a library.

There have also been many methods developed for specific applications. Examples of hese include a knowledge based system for the design of dwell mechanisms by **Rosen *et al*** [14] and a method for designing circuit breaker mechanisms by **Jobes *et al*** [15].

Whilst type synthesis is outside the scope of this program of research, it is important to note that all of the above techniques have only been successfully applied to single degree of freedom mechanisms and that the greater complexity of multi degree of freedom systems does not lend itself easily to simple expert systems. This is partly due to the lack of in depth knowledge concerning the design and synthesis of multi degree of freedom mechanisms.

2.2.3 Dimensional Synthesis of Mechanisms

Root & Ragsdell [16] provide a complete survey of optimisation methods applied to mechanism design up to the year of publication. This paper provides the basis for this literature review, and is essential reading. Some of the more important methods are described briefly in this section, along with some of the more recent methods in an effort to show the complimentary development of the two fields of optimisation and mechanism synthesis.

Dimensional synthesis can be sub-divided into one of three categories : graphical, analytical and numerical. Of these, numerical methods are the most popular as they

can easily be carried out using digital computers. Analytical methods can also be programmed into software but have too many inherent limitations to be widely used for complex systems. Graphical methods are now virtually obsolete.

In analytical methods, the number of precision points for which the mechanism can be synthesised is limited by the number of equations for solution. For example, the four bar mechanism can be synthesised for up to five positions for motion or path generation, or up to seven points for function generation. Analytical synthesis techniques produce a solution which is capable of reaching all the desired positions, but do not guarantee that the mechanism will be able to complete a continuous cycle through them all. Even if the mechanism is able to complete a cycle, there is no guarantee that significant excursions will not occur between the precision points. One novel analytical method has been developed by **Bogdan & Larionescu** [17,18] which synthesises a mechanism by considering it in terms of the structural groups, or dyads, which form the mechanism in terms of a complex Fourier series. The method can be applied to both transmission and guidance mechanisms and produces a minimum quadratic mean error. Another early method, as used by **Soni *et al*** [19], was developed to synthesise six bar mechanisms for a variety of output motions. This method of synthesis solves the displacement matrices of the mechanism to find the required link lengths of the mechanism to produce a motion defined by five precision points.

In contrast to analytical methods, numerical methods have no limit on the number of precision points that can be used to define the required output motion, but generally only produce an approximation to the desired motion. The majority of numerical methods utilise an error function which describes the difference between the desired and actual outputs of the mechanism. The synthesis can then be expressed as an optimisation problem, where the aim is to minimise the value returned by this error function.

As an example of early numerical synthesis, **Fox & Wilmert** [20] approach the synthesis of a four bar mechanism as a mathematical programming problem. The objective is to synthesise a mechanism where the coupler point closely approximates

the desired motion. The design variables are constrained to ensure that the result will be a four bar linkage with low internal forces and torques. The method utilises a simple iterative search process based around the minimisation method developed by **Fletcher & Powell** [21].

Many other synthesis methods have been applied to the four bar mechanism, as this simple mechanism is also one of the most versatile, having many applications in both path generation and function generation. **Nolle** [22] introduced the concept of a multi-dimensional objective function surface which represented the relationship between the quality of the solution to a problem and the independent design variables, and then developed an algorithm to search for the optimum four bar mechanism to act as a function generator for a given problem. **Garrett & Hall** [23] developed an approach where mechanisms were randomly generated for a given problem and ranked with respect to their optimality. This could even be viewed as a crude precursor of the new methodology, where the random generation has been give a degree of direction by modelling population genetics.

Kwak & Haug [24] introduce the parametric optimal design (POD) technique to mechanism design problems. This technique provides an effective method for the solution of the Chebyshev mechanism optimisation problem. The precision point approach can only give approximate solutions to this problem, since the precision points must be selected intuitively and will generally not provide a bound on the maximum error. The POD technique, however, determines and adjusts the critical points and so it provides a means of directly solving the Chebyshev problem.

Game theory has been applied to the synthesis of function generating four bar mechanisms by **Rao & Hati** [25]. Another approach to the same problem is given by **Rao** [26] which utilises geometric programming. In both examples, the objective is the minimisation of an error function. There have also been a number of approaches to the synthesis of path generating mechanisms, including an approach based on matrix algebra by **Ion & Cezar** [27]. Many of these methods have either assumed that there are no constraints upon the solution, or transform the constrained problem into an unconstrained solution. **Paradis & Wilmert** [28], however, used the Gauss

constrained method for solving mechanism problems where the objective functions are the sum of squares quantities with linear constraints.

Throughout this continuing development, the aim has been to develop more general and more effective synthesis techniques. **Ma & Angeles** [29] developed a general technique for the synthesis of path generating linkages which could be applied to planar, spherical and spatial four bar linkages and **Cleghorn *et al*** [30] developed a general synthesis method based on conventional non-linear programming techniques for a four bar path generating mechanism. The main advantage of the method is that it can easily be applied to practical problems and readily expanded to cover trajectory approximation problems.

The search for more effective synthesis techniques has led to many diverse approaches to the problem, and it is impossible to produce an in-depth review of all the methodologies that have been investigated. However, several of the more recent techniques will be outlined here before moving on to consider other aspects of mechanism design and synthesis. Many of the numerical methods available for precision point synthesis exhibit many shortcomings. Two of the more important are that the convergence of the solution depends on the quality of the initial approximations for the solutions and that the methods used tend to converge to just a single solution dependant on the initial estimate. The continuation method, as used by **Subian & Flugrad** [31], is a procedure that theoretically assures convergence to a solution without any initial solution estimates and also produces the complete solution set for a given system of non-linear equations. This method has been used for up to five precision points and could be used for six or more points, though the efficiency of the method reduces significantly. This technique has also been applied to five bar mechanisms by **Starns & Flugrad** [32]. The method is further extended to spatial mechanisms and six bar mechanisms by **Subian & Flugrad** [33,34]. A continuation method (or Homotopy method) has also been used by **Tsai & Lu** [35].

Jain & Agogino [36] applied a Simulated Annealing algorithm to three problems in the field of mechanism design. These were the synthesis of a four bar function generator, the synthesis of a rotary to linear recording mechanism and the multi-

objective optimisation of multi-speed gearboxes. For the first two cases the results were comparable to those obtained from other methods but in the third case an improvement over existing techniques was shown. One other attempt at using novel optimisation techniques was the use of a Genetic Algorithm to synthesise a four bar path generating mechanism by **Connor *et al*** [37,38]. The results of this early work indicated that the GA worked well on simple desired motions, but did not perform so well on complex motion requirements.

Another method which attempts to find "global" solutions is that developed by **Ogot & Alag** [39]. The methodology employs an analytical synthesis approach, based on complex number theory, to find a mechanism which satisfies a small number of prescribed hard precision points. This mechanism is then numerically optimised for an arbitrary number of soft precision points to improve the quality of the output motion. Hard precision points are defined as design points where the output motion passes exactly through the point at a given time. Soft precision points do not have to be satisfied so rigorously. By using a stochastic numerical approach, the search is guided through the highly non-linear solution space and local minima are often avoided allowing the process to be classed as fully automated.

2.2.4 Additional Objectives for Mechanism Synthesis

In addition to synthesising mechanisms based purely upon the simple kinematic output of the mechanism, many researchers have considered broadening the synthesis process to include other factors. Examples of these objectives include the work of **Malik & Dhande** [40], where the optimal kinematic synthesis of mechanisms takes into account possible manufacturing tolerances in the joints. This problem is also tackled by **Yin & Wu** [41].

Another design process which ensures that a planar linkage will be free from interference during the cycle of its motion has been developed by **Ling** [42]. Others, such as **Kochev** [43] and **Rao & Kaplan** [44] have considered mechanism

optimisation in terms of both the balancing and torque requirements within a mechanism.

Dhingra & Rao [45] have suggested that this kinetostatic optimisation should be combined with the kinetic synthesis to produce an optimal mechanism in a single design stage. They have proposed an integrated approach which does this by utilising fuzzy theories. Whilst this approach is suitable for single degree of freedom mechanisms, as both the kinematic and kinetostatic properties of the mechanism depend upon the single input it is questionable whether it can be applied to multi degree of freedom mechanisms as the kinematics and dynamic properties of the mechanism are often controlled by different dyads and inputs to the mechanism.

Lee *et al* [46,47,48] have extended the concept of applying both kinematic, kinetostatic and dynamic criteria to multi degree of freedom mechanisms and have discussed the development of performance tools and objectives for the design and synthesis of multi degree of freedom systems.

2.2.5 Multi Degree of Freedom Mechanisms

The majority of mechanism analysis and synthesis literature currently available tends to deal with single degree of freedom mechanisms such as the four bar mechanism. In contrast, relatively little work has been published in the area of multi degree of freedom mechanisms.

Multi-degree of freedom mechanisms of degree "n" require "n" independent variables to be specified to fully analyse the output motions. Some examples of multi-degree of freedom mechanisms include the five bar mechanism, the adjustable crank mechanisms investigated by **Cheunchom & Kota** [49] and the seven bar mechanisms of **Svoboda** [50].

Early work, such as that by **Pollit** [51], tends to deal with the analysis of the more simple multi degree of freedom mechanisms such as the geared five bar mechanism. In this mechanism, two inputs are required to fully analyse the output motion. These inputs are supplied by a definition of the position of one of the input cranks, a gearing ratio which specifies the relative motions between the rotations of the two inputs and a phase angle which describes the relative positions. Later work by authors such as **Kramer & Sandor**[52], **Rooney** [53] and **Erdman & Sandor** [54] approached the synthesis of this type of mechanism to provide a desired output motion. Most of these early synthesis techniques utilised an analytical closed form solution where the links of the mechanism were represented in complex number form.

Some early work was also carried out on mechanisms other than the geared five bar. **Pafelias & Sandor** [55] presented a general closed form solution method for the synthesis of a path generating linkage with "n" links, whilst **Kohli & Soni** [56] considered several classes of seven bar mechanisms and showed that for some cases the synthesis could be approached analytically but in other cases the highly non-linear equations had to be solved numerically.

One important area which has been considered is the mobility of multi degree of freedom mechanisms. **Freudenstein & Lee** [57] developed design charts for crank rotatability and optimisation of transmission angles in geared five bar mechanisms. **Ting & Tsai** [58,59] developed an effective and simple mobility criteria, similar to the Grashof criteria for the four bar mechanism, which can be applied to classify five bar mechanisms into two types of double crank linkages and non double crank linkages. This has important applications when considering programmable linkages where the input motions are independent.

Basu & Farang [60] have shown that five bar mechanisms which have relatively short input cranks have small output link motions. These motions are approximately describable as simple harmonic functions and so the output link velocities and accelerations also exhibit an approximation to simple harmonic motion. Therefore the inertial forces in the linkage are greatly reduced. A design method was developed

to synthesise mechanisms which not only generate a desired motion but also exhibit good dynamic behaviour.

Several other attempts have been made to synthesise multi-degree of freedom mechanisms. The methods used are varied and include the continuation method of **Starns & Flugrad** [32] and a penalty function approach by **Alizade *et al*** [61]. In this method, the synthesis is formulated as a mathematical programming problem. The penalty function is used to express the difference between the desired output and the generated output.

One particularly novel method, developed by **Sugimoto & Hara** [62] uses the theory of connecting chains. In this method, the synthesis is based on the concepts of mobile and quasi mobile workspace of the mechanism. **Shiller & Sunder** [63,63] have proposed a method for the optimisation of multi-degree of freedom mechanisms for near time optimal motions and optimal dynamic performance. In this work, dynamic performance is considered in terms of time requirements rather than torque or power requirements.

A great deal of the research into multi degree of freedom mechanisms considered the use of geared mechanisms, although some researchers had considered purely programmable drives. For example, **Huissoon & Wang** [65] considered the design of a direct drive five bar mechanism. **Kirecchi** [66] also considered the five bar mechanism, but concentrated on the classification and application of motion design principles to the programmable drive inputs.

One of the earliest applications of a hybrid machine can be found in the work of **Tokuz** [1,2]. In this work, a hybrid machine has been defined as a constant velocity motor acting in combination with a servo motor through a mechanism to produce an output motion. The hybrid arrangement developed consisted of a constant velocity motor and a servo motor acting through a differential gearbox to produce complex output motions. One of the drawbacks of this arrangement is that to produce a dwell in the motion, it is necessary for the servo motor to oppose the constant velocity motor, which gives rise to large power consumption.

Later work on hybrid machines by **Greenough** [3] and **Bradshaw** [4] has tried to eliminate this drawback in the generation of angular functions by replacing the differential gearbox with a two degree of freedom mechanism to act as a non linear gearbox. This work has been constrained so that the two inputs to the mechanism must be angular and that the output motion must also be angular. The linkage used in this study was a Svoboda seven bar linkage and the aim was to use a single linkage to generate a number of different output motions.

In this work, the synthesis of the mechanism was carried out using a combination of traditional optimisation techniques, including Powell's method, and the Golden Section Search. The use of traditional methods resulted in long computation times and the necessity of including complex penalty functions to force the solution towards feasible areas of the solution space.

2.3 Optimisation Techniques

The purpose of this section is to compare a variety of optimisation techniques. There are many different methods available for evaluating solutions to a given problem and searching for an optimal solution. These vary from calculus based techniques through to modern combinatorial methods. There are four main types of optimisation methods.

2.3.1 Classical Optimisation Theory

Classical optimisation theory develops the use of differential calculus to determine maxima and minima for unconstrained and constrained functions. Classical techniques subdivide into two main classes, direct and indirect. Indirect methods search for optimum points by solving the equations that are obtained by setting the first derivative of the objective function equal to zero. Direct methods search for

optima by taking a point on the objective function surface and moving in a direction determined by the gradient at that point.

Neither of these general methods are particularly useful outside of the simple domain where the objective function is uni-modal, smooth and continuous. They act mainly as a local search and are prone to becoming trapped on sub-optimal peaks. Classical techniques are not robust to provide useful solutions to complex problems.

2.3.2 Linear Programming

Linear programming is a technique that is applicable for the solution of problems where the objective function and constraints are formed as linear functions of the independent design variables. Linear programming methods can easily deal with both equality and inequality constraints.

The most general of the linear programming techniques, and the most widely used, is the Simplex Method. This method can deal with large numbers of design variables and is easily implemented as a numerical computation. For simple, linear optimisation problems the simplex method is a powerful tool which finds the optimum point in a multi-variate feasible region. Unfortunately, many optimisation problems are non-linear and other methods are required to find solutions.

2.3.3 Non-Linear Programming

Non-linear programming methods fall into both uni-variate and multi-variate categories. Essentially, non-linear programming methods are search techniques where an algorithm directs the search towards an optimal solution. Because the methods are based on searching, as opposed to calculus, the objective functions do not need to be smooth or continuous. Because of this non-linear programming

methods have become widely used and many attempts at robust mechanism synthesis have been based on these methods.

Examples of uni-variate methods include the Golden Section Search and Rosenbrock's Success/Failure search. These simple searches have been extended into the multi-variate domains to give rise to methods such as Powell's Method of Conjugate Directions and the Hooke and Jeeves Pattern Search. Using these methods, constraints are normally dealt with by the use of penalty functions.

Non-linear programming methods have many advantages, including ease of numerical computation, but also suffer in that they are generally local search methods. Without resorting to numerous and complex penalty functions they are not robust enough to search extremely convoluted objective function surfaces to any degree of accuracy. It is necessary to check that the methods are finding truly optimum solutions and this process of reiteration prolongs the time required for an analysis.

2.3.4 Simulation and Heuristic Methods

Recent research into optimisation techniques has concentrated on the development of new methods which are either more robust or computationally more efficient than the accepted non-linear programming methods. There are a proliferation of new algorithms that have been suggested and some of the more promising techniques are outlined below. Most of these techniques are well proven on problems such as the Travelling Salesman Problem, and some attempts have been made to apply the techniques to more practical problems.

2.3.4.1 Simulated Annealing

Simulated Annealing algorithms model the process of annealing in solids to optimise complex functions or systems. A full description of such algorithms is given by **Kirkpatrick *et al*** [67]. Annealing is accomplished by heating a solid to an elevated temperature and then allowing it to cool slowly enough so that the thermal equilibrium is maintained. Atoms in the material then assume a globally minimum energy state. Simulated Annealing algorithms have been successfully applied to a variety of problems including the optimal design of mechanisms by **Jain & Agogino** [36].

The operation of the algorithms is fairly simple. The algorithm starts with an initial set of design variables. A new design is then generated from the neighbourhood of the initial design by changing one or more of the design variables by a small amount. The objective function is then evaluated for the new design and if it gives rise to a better solution it is retained. If the design is a worse solution, then the probability that it is retained is found from the Boltzmann probability function;

$$P(\Delta E) = \exp(-\Delta E/T)$$

where;

ΔE = change in energy

T = temperature

The change in energy is expressed as the change in objective function value, whilst the temperature is merely a control parameter, with the same units as the objective function, which sets the probability of selection. The temperature is held constant for a prescribed number of iterations, to allow the system to gain "thermal equilibrium" and is then decreased in accordance to a cooling curve. As the temperature decreases, so does the probability that a poor design will be retained. This forces the algorithm to converge to an optimal, or near optimal, solution. Simulated Annealing algorithms

are reasonable robust provided that the parameters controlling the cooling curve are assigned values that reflect the complexity of the problem.

2.3.4.2 The Great Deluge Algorithm

As with most of the modern optimisation algorithms, the Great Deluge Algorithm, as described by **Sinclair** [68], is based on a simple idea. Assuming that the maximum of a function is to be found then the objective function is visualised as a land surface. As it "starts to rain" the algorithm searches the surface on the available "dry" areas. As the water level increases the end point of the algorithm will be forced on to a high point or optimal solution. This bears some resemblance to the Simulated Annealing algorithm where the temperature has been replaced by the water level.

2.3.4.3 Flexible Polyhedron Search

The Flexible Polyhedron Search described by **Borup and Parkinson** [69], and the very similar Polytope Algorithm described by **Gill *et al*** [70], minimises a function of "n" independent design variables by selecting a succession of "n+1" vertices of a polyhedron, or simplex, in a manner that improves the objective function evaluation. The strength of the method lies in that the shape of the polyhedron is allowed to change, giving rise to the name of the technique.

The algorithm begins by evaluating the objective function at each vertex of the polyhedron. The evaluation is then successively lowered by replacing the vertex with the highest value by better points generated using four operations. These operations are reflection, expansion, contraction and reduction. The scaling coefficients assigned to each of these operators, and the initial simplex size, defines both the search method and the efficiency of the search.

2.3.4.4 Tabu Search

The Tabu Search concept, initially proposed by **Glover** [71,72], is a heuristic procedure designed to guide other methods to avoid local optimality. Tabu Search has been shown to be effective on a wide variety of classical optimisation problems, such as graph colouring and Travelling Salesman Problems, and has also been applied to practical problems such as scheduling and electronic circuit design by **Bland & Dawson** [73]. The method uses constraint conditions, such as aspiration levels and tabu restrictions, and a number of flexible memories with different time cycles. The flexible memories allow search information to be exploited more thoroughly than rigid memory or memoryless systems, and can be used to either intensify or diversify the search to force the method to find optimum solutions.

2.3.4.5 Artificial Neural Networks

Artificial Neural Networks are composed of many simple elements operating in parallel and are modelled on biological nervous systems. The network function is determined largely by the connections between the elements. It is possible to train a neural network to perform a variety of functions by adjusting the relative weights assigned to these connections.

Neural networks have been trained to perform complex functions in a wide variety of fields including systems analysis by **Bardou & Sidahmed** [74], corrosion prediction by **Sanyal** [75] and the design of control systems by **Stylios & Sotomi** [76]. Neural networks can be applied to optimisation problems and performance can be enhanced by changing the learning rules. For example, it is possible to alter the normal back propagation training rule to include a "momentum" feature which helps the network avoid local minima.

2.3.4.6 Genetic Algorithms

Goldberg [77] describes Genetic Algorithms as a non-derivative based optimisation method based on the Darwinian theory of natural selection and survival of the fittest. The method starts with a initial population of randomly generated solutions and each successive generation is formed by mimicking the genetic operators that occur in natural systems. These operators are reproduction, crossover and mutation. As the number of generations increases the population "evolves" towards the global optimum solution.

Genetic Algorithms are a broad and effective search method which have been applied to a wide range of practical problems. Examples of this include gearbox design by **Pham & Yang** [78], the optimisation of the structure of laminates by **Ball *et al*** [79] and the design of simple machine elements by **Chen & Tsao** [79].

2.4 Comparison of Novel Optimisation Techniques

A number of studies, such as those of **Sinclair** [68], **Borup & Parkinson** [69], **Stuckman *et al*** [81] and **Bramlette & Cusic** [82], have been carried out which contrast and compare several of the novel optimisation techniques outlined in the previous sections. In general, these show that most of the techniques are equally effective in terms of the finding of globally optimum solutions. What differentiates the methods is the computational efficiency and ease of implementation.

Considering the results of these studies, as well as results obtained by **Connor *et al*** [37,38] in an early investigation, it was decided to continue to use a Genetic Algorithm as a synthesis tool. Genetic Algorithms are somewhat more computationally expensive, but they have a much more broad approach to the search which brings dividends in individual, as opposed to average, results. Also, GAs are more robust in terms of control parameters than other novel methods.

The following sections contain the results of a further literature review into the applications and effectiveness of the chosen method.

2.5 Applications of Genetic Algorithms

Early applications, such as the work of **Hollstein** [83] concentrated on the use of Genetic Algorithms as function optimisers. However, as the power of the method became clear it has been successfully applied to a wide variety of practical problems. Genetic Algorithms are now being applied in situations such as optimal control by **Krishnakumar & Goldberg** [84]. Other applications include maze learning by **Sychen & Kiichi** [85], vibration control by **Curtis** [86] and container packing problems such as presented by **Lin *et al*** [87]. **Gibson & Byrne** [88] have even applied Genetic Algorithms, in conjunction with an artificial neural network, in the field of musical composition.

2.6 Improvements to the Simple Genetic Algorithm

Many of the applications of Genetic Algorithms have only utilised a simple Genetic Algorithm which uses just the three main genetic operators of reproduction, crossover and mutation. In many cases these algorithms are prone to premature convergence or are not sophisticated enough to deal with complex functions. There are a variety of advanced techniques for use in Genetic Algorithm based search and optimisation which have been proposed to overcome these problems.

In previous work, **Connor *et al*** [37,38] carried out an initial investigation into the suitability of Genetic Algorithms in the field of mechanism design and optimisation. In general, the results show that the performance of basic Genetic Algorithms are adequate on simple problems, but poor on more complex problems. In this case, poor performance is attributed to the premature convergence of the population to a sub-optimal solution. However, later work by **Connor *et al*** [89] has shown a great

improvement on complex problems due to the removal of explicit constraints on the search. These results are in agreement with the work of **Rajeev & Krishnamoorthy** [90] who also used a penalty function to transfer an unconstrained search into a constrained domain and showed that performance was enhanced.

The techniques used to improve performance can be split into three categories. Firstly, there are those which tend to improve performance by preventing, or limiting, premature convergence. Secondly, there are those techniques which improve performance through the use of advanced genetic operators and modifications to the underlying form of the Genetic Algorithm. Finally, there are methods which try to reduce the effect of any inherent bias in the Genetic Algorithm due to a poor problem representation or selection method.

2.6.1 Prevention of Premature Convergence in Genetic Algorithms

Some of the earliest techniques used to prevent premature convergence modelled the effects of niche exploitation and speciation that occur in natural evolution. The general principle underlying all of these methods is that in natural systems, species develop to take advantage of an unused resource. This is niche exploitation. However, most natural resources can only support a limited amount of use. When a niche becomes overpopulated, the result can be a decline in size of the species, but can also lead to a sharing of resources through crowding and conflict.

Cavicchio [91] was one of the first researchers to attempt to induce niche exploitation and species behaviour in a Genetic Algorithm. He developed a mechanism called preselection. In this scheme, a "child" solution replaces its inferior "parent" if it has a fitness greater than that of the parent. Because of this, the diversity of the population is maintained because new solutions tend to replace solutions that are similar to themselves. Cavicchio claimed that preselection maintained a more diverse population over a given number of iterations when the population size was small.

De Jong [92] generalised preselection into a scheme he called crowding. In this method, individual solutions replace existing solutions according to their similarity with other solutions in an overlapping population. This is virtually the same methodology as preselection, but in practice this method does not restrict the new solution to replace one of its parents. Each individual solution is compared to a randomly selected sub-population of CF (crowding factor) solutions. The new solution replaces the most similar member of the sub-population. The similarity is usually calculated on a "bit-by-bit" basis. The effects of crowding are the same as with preselection. Diversity is maintained by the replacement of "like by like".

Schaffer [93] utilised fixed sub-populations in his use of multi-criteria objective functions, where each sub-population characterises a single criterion of the objective function. In each of the sub-populations, separate reproductive processes are carried out and there is no mixing between them. However, Schaffer expressed some concern about the nature of the search and to whether the end solutions were truly "globally" optimum. It is also unclear how effective this technique would be on the more usual single criterion objective functions.

Perry [94] carried out an investigation into biological niche theory by utilising a Genetic Algorithm. One of the key aspects of this work was the use of external schemata. These are special similarity templates, defined by the user, to characterise different species. This technique is of little practical use in generalised optimisation problems due to the necessary intervention and the requirement for problem specific knowledge. However, when heuristics are available to guide the search, the use of external schemata leads to a much more efficient search.

Another investigation which had a biological orientation was that of **Grosso** [95], who studied the formation of explicit sub-populations and the migration of individuals between them. The results are not directly applicable to generalised genetic search methods, but the study did show that limited migration between sub-populations was preferable to either homogenous mixing or isolation.

One technique exists which utilises the sharing of resources directly, as opposed to the implicit sharing found in other techniques. This is the sharing function developed by **Goldberg** [77,96]. The underlying principle of sharing is to force individuals located close to one another to share the "local" available resources until an equilibrium between different areas is reached. The result of this is the formation of several quasi-stable sub-populations distributed around several peaks on the objective function surface. The size of each sub-population is roughly determined by the size of the objective function peak.

All of the methods outlined above attempt to prevent premature convergence by the modelling of speciation and the associated development of sub-populations. However, there are a variety of other methods used to prevent premature convergence. Many of these are not modelled on naturally occurring events, but use a "common sense" approach to stop the population being dominated by a particular solution and as such are population management strategies.

The first of these methods is the uniqueness operator proposed by **Maudlin** [97]. This operator artificially maintains the diversity of the population by ensuring that when a new individual is substituted into a population it is not too similar to any existing solution. He also suggested at the start of the search the similarity measure should be greater than towards the end of the search. This bears some resemblance to Simulated Annealing algorithms. The effect is that the search becomes more localised as it progresses. The combination of uniqueness with crowding has been shown to greatly enhance the performance of Genetic Algorithms.

Pham & Yang [78] carried out an investigation where the intention was to discover as many feasible solutions to a problem as possible. To enhance the Genetic Algorithm they utilised a variety of techniques. The first, sharing, has already been discussed. The other three were deflation, identical string elimination and the use of heuristics. Deflation is a method of controlling the number of times that an individual is chosen for reproduction, thus limiting the chances of any one individual dominating the population. Each time an individual reproduces, its fitness is reduced by a fixed amount so reducing its chance of further selection. This method can also

been used in conjunction with a ranked fitness method. Each time an individual is selected, its position in the ranked population is reduced. It is possible that an accelerating deflation operator may be of great benefit.

Identical string elimination bears some resemblance to the uniqueness operator, and can be applied in several ways. The first, and most simple, is to subject identical strings to additional mutation or other operators. An alternative is to penalise all but one of the identical strings by an amount proportional to the number of identical strings in the population. The effect of this is, once again, to reduce the chance of any one solution dominating the population.

The use of heuristics can aid a Genetic Algorithm in a variety of ways. Initially, they may be used to place the initial population somewhere in the region of the optimal solution. Secondly, they may be used to give direct guidance to the search by utilising problem specific knowledge. Heuristics are of limited use due to the necessity of intervention and can often negatively bias a search causing it to locate a sub-optimal solution.

Whilst these three techniques have not been explicitly used to prevent premature convergence, the fact that they have been designed to maximise the total number of feasible solutions explored implies that the population will be diverse.

The final methods utilise a variety of mating schemes to prevent premature convergence. **Goldberg & Deb** [98] have shown that the inclusion of a mating scheme can further improve the performance of Genetic Algorithms which already utilise speciation. The essence of mating schemes is to prevent the formation of "lethal" solutions formed by reproduction between individuals which inhabit different peaks of the objective function surface. In natural systems, the mating restrictions are implicitly contained in speciation. There are a variety of mating schemes which have been used successfully by authors such as **Goldberg** [77], **Deb** [99] and **Booker** [100,101].

One mating strategy has been proposed by **Eshelman & Schaffer** [102] which contradicts the generally held view that "like should mate with like". These authors suggest that sharing functions have the effect of inhibiting, rather than promoting, within species mating for the dominant species. Instead of preventing diverse individuals from mating, similar individuals belonging to the largest species are less likely to be selected for reproduction with each other. Incest prevention is a more direct method for ensuring that similar individuals will not mate. Individuals are selected randomly for mating, but are only mated if their Hamming distance (a measure of dissimilarity) is above a given threshold. This threshold level is decreased as the number of generations increases. The effect of mating strategies that pair diverse individuals is that crossover becomes more disruptive of schemata. This contradicts the theory of Genetic Algorithms, but empirical evidence shows that it can lead to improved performance.

2.6.2 Improved Genetic Operators and Search Strategies

Whilst the randomised nature of Genetic Algorithms is highly desirable for complex objective functions, it implies that the convergence to optimum solutions is often slow. This may be offset slightly by the correct choice of control parameters but many researchers have investigated modifications to the basic Genetic Algorithm intended to improve performance.

Two examples of improved genetic operators are the directed mutation operator of **Bhandari *et al*** [103] and multi-point crossover, which is described by **Goldberg** [77] and **Booker** [101]. Multi-point crossover is more disruptive of schemata, which contradicts the essential basis of Genetic Algorithms, though there is some empirical evidence that it does lead to improved performance. This is because certain schemata are more easily formed using this as opposed to traditional crossover. Unfortunately, not all functions benefit from multi-point crossover, as these specific schemata are dependant on the problem representation.

Directed mutation deterministically introduces new solutions in the population guided by the information acquired in the previous generations. Essentially, it is a form of accelerated search as new points are explored "intuitively" based on previously explored regions. Links can be drawn between directed mutation and the intermediate memory cycle of the Tabu Search method of **Glover** [71,72]. Other aspects of Tabu Search may have applications in genetic optimisation

Several researchers have investigated the use of a biased random walk as a mutation operator, as opposed to the more widely accepted bit inversion. The use of this operator forces the Genetic Algorithm to explore more regions of the search space and as such often aids the search to discover the global optimum more quickly.

Yao [104] carried out an empirical study of the effects of using different genetic operators on the performance of a Genetic Algorithm used to optimise a Travelling Salesman Problem. Different operators and selection algorithms were investigated and the results indicated that "greedy" crossover and "hard" selection, combined with a low mutation rate, often led to enhanced performance. This combination of operators causes the search to become more aggressive.

As well as improved operators, there are a variety of search strategies that have been implemented to improve Genetic Algorithm performance. **Lin & Hajela** [105] have proposed a multi-stage search where the precision of the search is increased as the number of generations is increased. This increase in precision is obtained by either reducing the limits of the search space or by increasing the length of the string representation of the solutions. The first method has several drawbacks and can block the search from exploring promising areas of the objective function surface that were not explored in the previous stage. An integral part of the strategy is the initialisation of the population for each stage of the search to ensure that solution space is explored fully.

A similar approach is used by **Whitley *et al*** [106] in their delta coding approach. Delta coding achieves a trade off between fast search and sustaining diversity. Diversity is reintroduced by the random generation of new populations as the search

progresses. The delta coding approach is more refined than many other re-initialisation methods. Examples of these include cataclysmic mutation used by **Eshelman** [107] and the micro-Genetic Algorithms of **Krishnakumar** [108].

2.6.3 Reduction of Bias in Genetic Algorithms

The probabilistic nature of Genetic Algorithms makes the technique susceptible to stochastic sampling errors and biases due to poorly thought out solution representation. **Goldberg** [109] has outlined a broad methodology for the design of effective Genetic Algorithms.

One of the most important aspects of Genetic Algorithm design is the choice of an appropriate solution representation. Whilst, to a certain extent, the solution representation is determined by the nature of the problem there are a variety of techniques available which can be used to reduce the effects of any bias irrespective of the actual representation chosen.

Caruana & Schaffer [110] have shown that the use of Gray, as opposed to binary, coding can lead to a more effective search. Gray coding uses a binary alphabet but the order of decoding differs from normal binary notation. Their study suggests that Gray coding eliminates the "Hamming cliff" problem that makes some transitions difficult if a binary notation is used. This problem arises due to the nature of binary decoding. The Hamming cliff arises due to the substantial differences between binary coding of similar decimal numbers. This is just one instance of hidden bias that emerges from an interaction between search control and knowledge representation.

An alternative technique for reducing coding bias has been suggested by **Levenick** [111] by considering the way in which genetic coding occur in natural systems. There are sections of DNA called introns, which are non-functional. That is, these sections of DNA do not effect the genetic characteristics of an individual. Levenick has shown that inserting introns into a Genetic Algorithm solution coding can

improve performance by up to a factor of ten. It is suggested that the reason for this is that these non-functional sections of the binary string makes crossover less disruptive.

One other source of bias in Genetic Algorithms is the presence of sampling errors in the selection algorithm. **Baker** [112,113] has investigated the effects of different selection algorithms on the performance of Genetic Algorithms. These algorithms include standard selection, ranked selection and stochastic sampling methods. The results of this study show that the choice of an appropriate selection algorithm not only reduces the bias inherent in selection but also inhibits premature convergence.

The final consideration to the improvement of Genetic Algorithm performance is the correct choice of control parameters. **Grefenstette** [114], **Goldberg** [115] and **Schaffer *et al*** [116] have carried out investigations to identify the optimal control parameter set. However, it is unclear to what extent the results of these studies are applicable to other optimisation tasks, as the definition of a new objective function will effect how the Genetic Algorithm will perform.

2.7 Summary

This Chapter has covered some of the recent literature that is relevant to this research. It has shown that the trends in mechanism design and synthesis requires tools that are even more robust and effective.

This requirement is being met by the development of optimisation methods based on simulation and heuristics which exhibit less deficiencies than methods based on gradient search or non linear programming techniques.

Chapter Three

The Fundamentals of Genetic Algorithms

3.1 Introduction

The purpose of this Chapter is to provide an introduction to Genetic Algorithms the search technique chosen for use in the proposed design methodology. In the following sections the mechanics of GAs are explained and an attempt is made to describe not only how they work, but also to answer the questions of "why do they work?" and "why are they so effective?".

3.2 Genetic Algorithms

This section will describe the differences between GAs and other search techniques and explain both how, and why, GAs operate. **Goldberg** [77] introduces GAs as;

> "Genetic Algorithms are search algorithms based on the mechanics of natural selection and population genetics. They combine survival of the fittest among string structures with a structured, yet randomised information exchange to form a search algorithm with some of the innovative flair of human search."

This is a succinct description of what has grown to be a powerful technique with many applications in search, optimisation and machine learning. Much of the work carried out today would not have been possible if it had not been for the pioneering work of **Holland** [117] at the University of Michigan into adaptive systems. His aim

was to develop the theory and procedures necessary for the creation of general programs and machines with unlimited capability to adapt to arbitrary environments.

Before explaining the exact mechanics of how GAs work, it is important to build up a conceptual image of the technique. In essence, as with all search techniques, an objective function is defined which encapsulates the desirable characteristics of a design solution. This objective function can be viewed as defining an environment in which a population of solutions exists. By breeding solutions to form new populations the method can be encouraged to evolve optimal solutions.

3.2.1 The Differences Between Genetic Algorithms and Other Search Methods

GAs differ from most other search methods in many ways and it is these differences which make them as robust and applicable as they are. Several main differences are given below;

1. GAs work with a coding of the parameter set, not the parameters themselves.

2. GAs search from a population of points, not a single point

3. GAs use payoff (objective function) data directly, and do not rely on secondary information such as gradients.

4. GAs use probabilistic transition rule, not deterministic rules.

These differences give rise to a variety of advantages to genetic searching, other than robustness, and these will be explored in the appropriate sections. However, these advantages will be briefly outlined here.

The schema theorem of **Holland** [117] will be explored fully in section 3.4. However, this theorem explains the efficiency of the GA search techniques in terms of similarity templates or schemata. Each schema represents a hyperplane of solutions, and testing a single schema may be representative of testing the entire hyperplane. This implies that a large solution space can be efficiently searched by testing only a relatively small number of solutions. For complex optimisation problems, such as mechanism synthesis, this provides a very efficient search technique as it reduces the number of computations required to search the solution space.

GAs do not rely on secondary information, such as derivatives, to guide the search through a given solution space. Because the search is guided by an "intuitive" mechanism which utilises objective function information directly, it is possible to search complex and multi-modal functions without becoming trapped in local optima.

Another reason that GAs avoid local optima is the fact that they utilise a population of solutions. The solutions can be encouraged to populate all of the optima on an objective function surface, where the number of solutions on each peak is approximately related to the magnitude of the peak. This ensures that the global optimum solution is found.

It should be noted that, due to their probabilistic nature, GAs do not necessarily converge to a specific point or solution, but will normlly continue to generate solutions which populate the near optimal regions.

3.2.2 Solution Coding, Schemata, Genetic Operators and Fitness

When discussing the mechanics of GAs it is necessary to understand several concepts simultaneously. These concepts are solution coding, schemata, genetic operators and fitness. The purpose of this section is to briefly describe these concepts

and how they are applied to GAs. Later sections will discuss the concepts in greater depth and discuss the implications with respect to genetic based searching.

3.2.2.1 Solution Coding

GAs are based on the mechanics of natural selection and survival of the fittest. Essentially, GA terminology has been derived from the field of natural genetics. Solutions are coded as chromosomes. These chromosomes are composed of genes, which describe characteristics of the solution, and can take on one of several different values called alleles. The position of a gene in a chromosome is known as its locus.

This terminology is best explained in terms of an analogy to a typical natural system. Consider a genotype which represents the characteristics of a human being. The genotype consists of a set of forty six chromosomes. Each chromosome contains several genes and it possible to isolate a specific gene by its locus. For example, assume that the gene with locus, or position, x represents the eye colour of an individual. The alleles of this gene may be green, grey, blue or brown. The table in Figure 3.1 compares both the natural genetic terminology and adaptations for GA work. These terms are often intermixed.

Natural Terminology	**GA Terminology**
Chromosome	String
Gene	Feature, character or detector
Allele	Feature value
Locus	String position
Genotype	Structure
Phenotype	Parameter set or complete solution

Figure 3.1 : Comparison of Terminology

Most implementations of GAs utilise a binary notation for the solution strings. The reasons for this will be explained in section 3.4.

3.2.2.2 Genetic Operators

Genetic operators are the mechanisms by which GAs simulate evolution and the processes of natural selection. There are three main operators which are used in simple GAs which are analogous to those found in natural systems. These are reproduction, crossover and mutation. There are many other operators that have been developed for specific tasks which will be covered in later sections.

Reproduction is a method for selecting individuals, or strings, from a given population to act as parents for the successive generation. There are many different selection schemes but the most common is "roulette wheel" selection, where a probability of selection is assigned to a string that is porportional to its fitness, or quality. This method of selection implicitly contains the natural selection mechanism as only strings with above average fitness are selected and the genes of these strings are propagated from generation to generation. Once two parent strings have been selected they may be subjected to both crossover and mutation depending on the respective probabilities of each occurring. If no crossover or mutation occurs, the parent strings are resubstituted into the new population. This is a simple form of genetic preservation.

Crossover is the method by which parental genes are passed on to child solutions. Many crossover mechanisms have been developed for different types of problem, however the most commonly used is single point crossover. If crossover occurs, the genes of the two parent strings are intermixed to form two new children that are substituted into the new population. Crossover occurs between two strings around a randomly selected crossover point. The exact mechanism of crossover can be illustrated by the use of an example. Consider two parent strings in binary notation;

5 4 3 2 1 0	Position
1 1 1 1 1 1	Parent 1
0 0 0 0 0 0	Parent 2

Given that the crossover site is position 3, the two children formed are;

5 4 3 2 1 0	Position
1 1 1 0 0 0	Child 1
0 0 0 1 1 1	Child 2

As has already been stated, crossover allows the exchange of information between different strings and propagates genetic information from generation to generation. However, a GA that utilises crossover alone will rarely converge to a globally optimum solution as it very difficult to expand the search beyond the portion of the solution space represented by the initial population.

Mutation maintains the global nature of the search by introducing new information into the population and hence expanding the region of the solution space that can be explored. In a binary GA, mutation can easily be implemented as a Boolean bit inversion. Each bit in a child solution is tested against the probability of mutation occurring. If mutation occurs the value of the bit changes. In binary notation, a one becomes a zero or a zero becomes a one.

3.2.2.3 An Introduction to Schemata

A schema (pl. schemata) or similarity template is used to describe a subset of solutions with similarities at certain positions. Consider a five bit binary coding of a solution. One schema for this string could be {1 0 0 # 1} where the # symbol

represents a "don't care" so that the position can be either a one or a zero. The exact value is deemed irrelevant.

This schema represents a subset of two strings which are instances of the schema. These two strings are;

{ 1 0 0 0 1 , 1 0 0 1 1 }

Schemata give a powerful and compact method for analysing similarities between different solution structures as well as an insight into the reasons why GAs work. For strings of length L, formed with an alphabet of cardinality M there are $(M+1)^L$ schemata. As an example, consider the five bit binary string which has $2^5 = 32$ combinations of strings and $3^5 = 243$ schemata. A given binary string will contain M^L schemata, as each position can be represented as a # or as its defined value. A population of N strings will contain between M^L and $N.M^L$ schemata depending on the diversity of the population.

Reproduction ensures that strings with high fitness have a higher probability of survival. In terms of schemata, this can be expressed by saying that strings which contain effective schemata are more likely to reproduce.

Crossover allows for the combination of strings that exhibit different schemata. However, crossover may affect the structure of a given schema. For example, the schema { 1 0 # # 1 } is much more likely to be disrupted by crossover than the schema { 1 1 # # # }.

Normally, mutation rates are very low. **De Jong** [92] suggested that the mutation rate should vary inversely with the population size. Mutation does not readily affect the integrity of a given schema.

As a result of reproduction and crossover, highly fit schemata with a short defining length, known as building blocks, are propagated from one generation to the next. Also, these schemata are combined with other building blocks to form more highly

fit solutions. This handling of multiple schemata is known as Implicit Parallelism and is a vital concept in understanding how Genetic Algorithms work. This will be discussed in greater depth in section 3.4.3.3, but for now it is sufficient to realise that the efficiency of genetic search is based on the number of schemata that are processed. **Goldberg** [77] indicates that for a population of N strings, approximately N^3 schemata are usefully processed in each generation.

3.2.2.3 Fitness Function

Fitness is an important concept both in natural genetics and in Genetic Algorithms. Put simply, fitness is a measure of how good an individual is. Individuals with high fitness are more adapted for survival in their given environment and are more likely to reproduce.

The fitness function is used to determine the fitness of an individual string. In general, the fitness function should always supply a positive numerical answer, where the higher the number the greater the fitness. Because the nature of many problems leads to an objective function that results in negative values, it is often necessary to map the fitness function to the objective function. There are a variety of ways in which fitness values are assigned to an individual., including standard fitness, rank method and rank space method.

3.3 A Basic Genetic Algorithm

The three operators of reproduction, crossover and mutation form the basis of a basic Genetic Algorithm. When combined with fitness and decoding functions they allow subsequent generations of string populations to be created. Section 3.3.1 shows how a Genetic Algorithm works by "hand working" a solution to a simple problem.

The operation of the Genetic Algorithm can be written as a 'pseudocode' algorithm;

```
generation=0;
create initial population;
repeat
        evaluate fitness;
        test termination criteria;
        reproduce, crossover & mutate;
        generation=generation+ 1;
end
```

3.3.1 A Hand Worked Genetic Algorithm Solution

The aim of this section is to illustrate the workings of a basic Genetic Algorithm in an optimisation problem. The grid in Figure 3.2 shows the fitness values for all combinations of the two variables X and Y. The optima occurs when X=5 and Y=3.

	7	1	2	3	4	5	6	5	4
	6	2	3	4	5	6	7	6	5
	5	3	4	5	6	7	8	7	6
	4	4	5	6	7	8	9	8	7
Y	**3**	5	6	7	8	9	(10)	9	8
	2	4	5	6	7	8	9	8	7
	1	3	4	5	6	7	8	7	6
	0	2	3	4	5	6	7	6	5
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
					X				

Figure 3.2 : Values of Fitness for Two Variable Function

The variables are coded as binary integers between the range of 0 and 7. The structure of a typical solution is given below.

3 0 1 1 0 1 0 7

The first digit is the number of the solution in the population. For this problem the size of the population is six. The first three digits of the string form the X chromosome and the second three digits form the Y chromosome. The final integer value is the fitness of the decoded parameters. In this specific case, X=3 and Y=2. An initial population of six strings are randomly generated by tossing a coin. The size of the population is maintained at six throughout the optimisation process, with each pair of parents creating one pair of children. The initial strings are shown below;

1	0 1 1 0 0 0	5
2	1 0 1 1 1 0	7
3	0 0 1 0 1 1	6
4	1 1 1 1 0 1	6
5	0 1 0 1 1 1	3
6	1 0 0 1 0 0	8

The next generation is created from the initial strings. Reproduction is based on roulette wheel selection, so strings with high fitness are more likely to be selected. The second and third generations are shown below, along with the number of the parent strings from the previous generation. Mutated values are shown in bold and the crossover points are shown by the : symbol.

1	1 0 0 1 0 0	8	Parents 6,4	No crossover
2	1 1 1 1 0 1	6	Parents 6,4	No crossover
3	1 0 1 1 1:1	6	Parents 2,3	
4	0 0 1 1 1:0	7	Parents 2,3	
5	1 0 0:1 1 0	6	Parents 2,6	
6	1 0 1:1 0 0	9	Parents 2,6	

1	1 0 1 1 0 0	9	Parents 6,2	No crossover
2	1 1 1 1 0 1	6	Parents 6,2	No crossover
3	1 0:1 1 1 0	7	Parents 1,4	
4	0 0:0 1 0 0	4	Parents 1,4	
5	1 0 1 **0** 1 1	10	Parents 3,4	No crossover
6	0 0 1 1 1 0	3	Parents 3,4	No crossover

The optimum value is reached in the third generation, although it should be realised that in a real optimisation problem the optimal solution would not be known and the termination criteria would be based on a combination of the number of generations and some form of performance evaluation criteria.

One of the advantages of Genetic Algorithms is that if the process is allowed to continue, the average fitness of the population will increase although the optimum value may not be replicated in each subsequent generation. To illustrate this, consider an additional generation.

1	1 0 1 1:0 0	9	Parents 1,3	
2	1 0 1 1:1 0	7	Parents 1,3	
3	1 1 1 0 1 1	10	Parents 2,5	No crossover
4	1 **0** 1 1 0 1	8	Parents 2,5	No crossover
5	1 1 1:1 **0** 0	7	Parents 3,2	
6	1 0 1:1 0 1	8	Parents 3,2	

In this case, the optimal string is reintroduced into the population because crossover did not occur. Because of some lucky mutations, the other strings are also converging towards this value. This is because of the propagation of effective schemata through the generations.

3.4 The Schema Theorem

The previous section illustrated how Genetic Algorithms work. To describe why they work demands a more rigorous approach. This is presented as the schema theorem. Whilst it is not necessary to fully understand the schema theorem to utilise Genetic Algorithms, it provides a mathematical foundation which cannot be ignored.

The schema theorem was described generally in section 3.2.2.3, where it was suggested that highly fit schema are propagated through the populations of each generation. This statement may be substantiated by using basic probability theory and equations developed which can be used to calculate the rate of growth of a given schema.

3.4.1 Schema Order And Defining Length

The concept of schemata was introduced in section 3.2.2.3. A schema, or similarity template represents a subset of solutions. However, some schemata are more specific than others. For example, the schema { 0 1 1 # 1 } is a more specific instance of the schema { 0 # # # # }. Similarly, some schema span more of the string length than others. For example, the schema { 1 # # # 0 } spans a larger portion of the string than the schema { 1 1 # # # }. To quantify this, two properties of schemata have to be defined.

The order of a schema H, denoted by o(H), is simply the number of fixed positions in a schema. The defining length of a schema, denoted by $\delta(H)$, is the distance between the first and last specified positions. Consider the schema H ={ 1 # # 0 1 # } which can now be seen to have order, o(H) = 3 and defining length, $\delta(H) = 4$.

3.4.2 Mathematical Formulation Of The Schema Theorem

The reproductive schema growth equation can be formally stated as;

$$m(H,t+1) = m(H,t)\frac{f(H)}{\bar{f}} \qquad ...(3.1)$$

where;

m(H,t)= number of examples of the schema H at time t.

f(H)= average fitness of strings representing schema H at time t.

$\bar{f}$ = average fitness of the entire population.

This equation shows that a particular schema grows as the ratio of the average fitness of the schema to the average fitness of the population. This implies that schemata with fitness values higher than the population average will receive an increasing number of samples in the next generation. The opposite applies to schemata with fitness below the population average.

Suppose a schema H remains above average by an amount $c\bar{f}$ where c is a constant. The schema growth equation can be rewritten as;

$$m(H,t+1) = m(H,t)\frac{(\bar{f}+c\bar{f})}{\bar{f}} = (1+c)m(H,t) \qquad ...(3.2)$$

Now, starting at t = 0 and assuming that c has a stationary value;

$$m(H,t) = m(H,0)(1+c)^t \qquad ...(3.3)$$

This equation describes a geometric progression. The effect of reproduction is now clear. It allocates exponentially increasing numbers of trials to above average

schemata. The opposite holds true for below average schemata. Crossover and mutation will effect this allocation of trials, but the growth equation is easily adjusted to take these effects into consideration.

During crossover, the probability that a schema will survive, p_s, is given as;

$$p_s = 1 - \frac{\delta(H)}{(L-1)} \qquad ...(3.4)$$

where;

L is the length of the string.

If crossover occurs with a probability p_c then the chance of survival can be written as;

$$p_s \geq 1 - p_c \frac{\delta(H)}{(L-1)} \qquad ...(3.5)$$

It can be seen that eqn. 3.5 reduces to eqn. 3.4 when $p_c = 1$.

The combined effect of reproduction and crossover can now be considered by multiplying the expected number of schemata for reproduction alone by the survival probability. The number of a particular schema H in the next generation can be calculated using eqn. 3.6.

$$m(H,t+1) \geq m(H,t)\frac{f(H)}{\bar{f}}\left[1 - p_c \frac{\delta(H)}{(L-1)}\right] \qquad ...(3.6)$$

The schema H will grow (or decline) according to some multiplication factor. With both reproduction and crossover the factor depends on whether the schema has above

or below average fitness and whether the schema has relatively short defining length. Those schemata with above average fitness and short defining lengths are going to be sampled at exponentially increasing rates.

Mutation occurs in a large population relatively frequently, but even so its effect on the sampling of schemata cannot be ignored. A schema will survive mutation if all the specified positions in the schema are not mutated. If mutation occurs at a position with probability p_m, the probability that any given allele will survive is $(1-p_m)$. The probability that a schema of order o(H) will survive is given by;

$$p_s = (1-p_m)^{o(H)} \qquad ...(3.7)$$

For small values of p_m this may be approximated by a simpler expression;

$$p_s = 1-o(H)p_m \qquad ...(3.8)$$

The mutation operator can then be included in the growth equation.

$$m(H,t+1) \geq m(H,t)\frac{f(H)}{\bar{f}}\left\{1-p_c\frac{\delta(H)}{(L-1)}-o(H)p_m\right\} \qquad ...(3.9)$$

Eqn. 3.9 is only an approximation, as some cross product terms have been ignored. However it is accurate enough to substantiate the claim given in section 3.2.2.3 that low order schema with short defining length and above average fitness will be rapidly propagated throughout the population. This is the schema theorem developed by **Holland** [117] which is also known as the fundamental theorem of Genetic Algorithms.

3.4.3 Implications Of The Schema Theorem

The implications of the schema theorem are wide ranging and consideration should always be given to the design of a Genetic Algorithm and the solution representation used. A full analysis is given by **Holland** [117] and **Goldberg** [77]. Three specific cases will be dealt with briefly here.

3.4.3.1 The Building Block Hypothesis

Schema processing, or the efficiency of GAs is related to the coding of a solution. In order to maximise the rate of growth of highly fit schema, there are two principles which should be adhered to. These are the principle of meaningful building blocks and the principle of minimal alphabets.

The principle of meaningful building blocks states;

> "The user should select a coding so that short, low order schemata are relevant to the underlying problem and relatively unrelated to schemata over other fixed positions."

After considering the nature of schemata, and schema processing, the logic behind this statement is easily understood. An optimal solution consists of several schemata with high fitness. The role of the schemata is to subdivide the solution space and hence force the search towards the optimal solution. This is the building block hypothesis which is covered by **Goldberg** [77], but in greater depth by **Bethke** [118].
The principle of minimal alphabets states;

> "The user should select the smallest alphabet that permits a natural expression of the problem."

The smaller the cardinality of the alphabet, the easier it is to 'spot' similarities in strings with high fitness. That is, it is easier to construct relevant schema for a given problem. It is easy to show that a binary alphabet offers the maximum number of schemata per bit of information of any alternative coding system.

Despite the fact that a binary alphabet is the most useful in a Genetic Algorithm coding, there are still subdivisions of coding using this scheme. It has been suggested by **Caruana & Schaffer** [110] that Gray coded numbers offer improved performance than standard binary coded numbers. Gray coding uses a binary alphabet, but a different coding sequence which leads to a reduced Hamming distance, which is a measure of how the change of a single binary digit effects the decoded decimal result. In binary coding the Hamming distance varies, and as Hamming distance increases, schemata similarity decreases. With Gray coding, the Hamming distance is constant.

3.4.3.2 Schemata As Hyperplanes

To illustrate how schemata act as hyperplanes, consider the strings and schemata of three bit length. The search space is easily visualised in three dimensions as a cube. Points on the cube are represented as schemata of order three, lines in space are represented by schemata of order two and planes in the solution space are represented by schemata of order one. The entire solution space is represented by the schemata of order zero (ie {# # #}).

This generalises to longer strings, and higher dimensional search spaces. Points, lines and planes described in three dimensional space generalise to hyperplanes of varying dimensionality. Searching a given schema may be thought of as searching the entire hyperplane to which the schema is representative. A Genetic Algorithm can be thought of as cutting across different hyperplanes to search for an improved performance.

3.4.3.3 Implicit Parallelism

Implicit parallelism was first introduced in section 3.2.2.3, and is essentially a description of how Genetic Algorithm based searches are effective. It has been described by **Grefenstette & Baker** [119] in the following manner;

> "The power of a Genetic Algorithm derives largely from it's implicit parallelism, i.e., the simultaneous allocation of search effort to many regions of the search space..."

Implicit parallelism is inseparable from the schema theorem. The searching of regions of the solution space simultaneously is due to the propagation of highly fit schemata through out the population. Therefore, implicit parallelism describes the action of both the building block hypothesis and the expression of schemata as hyperplanes.

3.5 Advanced Genetic Techniques

The previous sections have outlined the necessary operators, techniques and theory required to implement a basic Genetic Algorithm as an optimisation tool. However, there are a variety of advanced techniques and operators which have been developed to either deal with specific problems or to improve the efficiency of the basic Genetic Algorithm.

3.5.1 Diploidy And Dominance

In nature there are two types of genotype, haploid and diploid. Haploid chromosomes contain only one set of data and are the only type considered so far. However, diploid chromosomes are much more common in nature and have several advantages over haploid structures. In diploid chromosomes, two datasets are carried in parallel. These

contain different data for the same characteristics. This representation introduces redundancy, and a decision must be made as to what data is to be used.

Dominance is the method used to choose between the datasets. Dominance occurs at the gene level, not at the chromosome level. This means that the chromosome used is a combination of the two datasets carried. Consider the diploid chromosome below, where capital letters indicate a dominant gene, and different letter represents a different allele or value;

0	1	2	3	4	Position
A	b	C	D	e	
a	B	c	d	e	

The dominant genes are 'carried forward' and form the dominant chromosome or dataset. In this case, the dominant coding is {A B C D e}.

Diplold chromosome structures have an advantage over simple haplold structures in that they retain the recessive genes as a form of memory. Should the external conditions change significantly then the recessive gene may become dominant. In nature this leads to an adaptable species. An example of how nature uses diploid chromosomes to increase adaptability is given by **Goldberg** [77], who describes the change of colouring of the peppered moth during the industrial revolution.

Diplold chromosomes may be used in Genetic Algorithms to carry additional data throughout the optimisation. The selection of the dominant gene is based on schemata fitness. Diploid GAs lead to a more flexible search and improved convergence, particular for time dependent problems where the second set of genes allows for quick adaptation.

3.5.2 Alternative Genetic Operators

Section 3.2.2.2 introduced the three most basic genetic operators, reproduction, crossover and mutation. Reproduction is a selection method which determines which solutions are allowed to breed. Crossover allows the transfer of data between different solutions. Mutation preserves the global nature of the search by introducing random changes in a solution.

There are many other possible genetic operators which may be included into a Genetic Algorithm, some of which will be considered below.

3.5.2.1 Elitism

Elitism is a form of preservation, where highly fit strings from one generation are reintroduced into a new generation directly, as well as through reproduction. Introducing elitism into genetic searching has a dramatic effect. In general, elitism will improve the local searching of a Genetic Algorithm, but will have a detrimental effect on global searching power. This is due to the bias incurred by the propagation of a given individual into the next generation which may lead to premature convergence.

3.5.2.2 Inversion

Inversion is the primary operator that allows for the recoding of a particular string representation. Two inversion sites are chosen at random and the middle section reordered. Consider the string below, where the symbol : marks the inversion sites;

0 1 2 3 4 5 6 7	Position
1 1:0 1 1 1:0 1	
1 1:1 1 1 0:0 1	Inverted string

There are several methods which combine the inversion, or reordering, operator with the crossover operator. These include partially matched crossover, order crossover and cycle crossover.

3.5.2.3 Partially Matched Crossover

Partially matched crossover was created to solve the traveling salesman problem, which involves minimising the distance traveled between a number of cities. Two strings are aligned and then two cross sites are chosen. Data is transferred between the strings on a bitwise exchange. For example;

0	1	2	3	4	5	6	7	Position
9	:	7	3	5	7	2	:	8	8
1	:	4	6	4	9	1	:	3	1

The resulting two strings are given below. The data is transferred directly between the strings, into the same locus it occupied in the parent string;

0	1	2	3	4	5	6	7	Position
9	4	6	4	9	1	8	8	
1	7	3	5	7	2	3	1	

Order crossover and cycle crossover have also been created to help solve the traveling salesman problem, and are both also "artificial" operators. That is, there is no equivalent operator found in natural systems. Both order and cycle crossover are similar to partially matched crossover in application, but have different effects.

3.5.2.4 Duplication And Deletion

Duplication and deletion are two similar micro operators that can be used in Genetic Algorithms. Duplication copies a given gene and puts it alongside its progenitor on the chromosome. Deletion removes duplicate genes. In essence, duplication increases the effective mutation rate, whilst deletion reduces the effective mutation rate.

3.5.3 Breeding Schemes

A variety of breeding schemes are available which restrict reproduction between certain chromosomes. The most simple are based on gender, where a certain characteristic of a solution causes it to be either male or female. Reproduction then occurs between chromosomes of different gender.

The specialisation permitted by sexual differentiation is carried further in nature through speciation and niche exploitation. A species is a class of organisms with common characteristics, whilst a niche is an organisms role. The introduction of species and niche exploitation into Genetic Algorithms can be used to direct a search towards either sub-optimal peaks or multiple peaks in a multi modal function.

In problems of this nature, it is essential that the Genetic Algorithm maintains the diversity amongst the population. This is achieved by changing the reproduction rules, allowing the reproduction of individuals in particular sub-populations to take place.

3.5.3.1 Crowding

Crowding is a technique that was developed by **De Jong** [92] to promote the development of sub-populations and hence maintain population diversity. De Jong reasoned that in natural systems, as similar individuals begin to dominate a niche,

increased competition for limited resources decreases life expectancy and birth rates. Less crowded niches exhibit life expectancies nearer to their potential.

The method of applying crowding to GAs is as follows. One of the most important differences is that in each iteration the generation of solutions is not entirely replaced. The proportion of the population that is replaced is governed by the generation gap. Another new parameter must also be defined. The crowding factor (CF) is essentially the size of the sub-population of solutions.

When an individual is born, one individual from the existing population is chosen to die. The dying individual is chosen from a subset of CF members chosen from the full population at random. The exact individual to die is chosen from the subset on the basis of similarity, where the member of the subset which most resembles the new individual, on a bit-by-bit count, is replaced. Essentially, crowding is a more general form of the preselection technique developed by **Cavicchio** [91].

3.5.3.2 Sharing

In practice, neither preselection or crowding explicitly utilise the concept of niche generation due to the exploitation of environmental resources. Sharing is one method, proposed by **Goldberg** [77], which does do this. In this method, a sharing function is defined to determine the neighbourhood and degree of sharing for each string in the population. For a given individual, the degree of sharing is determined by summing the sharing function values contributed by all other strings in the population. Strings close to an individual require a high degree of sharing and vice versa. After accumulating the total nature of shares, an individuals actual fitness is calculated by taking the potential fitness and dividing through by the total number of shares. In this way environmental resources are shared between individuals and sub-populations, or niches, are formed.

In addition to crowding and sharing, there are many methods that can be used to artificially maintain the diversity of a population. These include fitness reduction,

identical string elimination and the use of heuristics as outlined by **Pham & Yang** [78]. Another method is the uniqueness operators of **Maudlin** [97].

3.5.4 Fitness Evaluation

Up until now fitness has been assumed to be a positive number which describes how good a solution is. Whilst this is true, there are several methods which may be used to calculate fitness. These are standard fitness, rank fitness and rank-space fitness. Scaling, or normalisation, methods can also be used to ensure that selection probabilities are such that there are no biases in the selection algorithm.

3.5.4.1 Standard Fitness

The standard method of evaluating the fitness of an individual is to divide the 'quality' (objective function value) of that individual and by the sum of the populations quality.

$$f_i = \frac{q_i}{\sum_{j} q_j} \qquad \text{...(3.10)}$$

The standard fitness computes the fitness values relative to the population and returns a value between zero and one.

3.5.4.2 Rank Method

Standard fitness has several disadvantages. One of these is the inability to influence the selection of individuals from the population. Rank fitness controls the bias towards the best performing individuals and also counteracts against the implicit bias following a poor choice of measuring scale. The rank method only uses the quality of an individual

to rank it in order within the population. Once the population has been ranked, a selection probability is assigned to each individual according to it's position in the ranked population.

3.5.4.3 Rank Space Method

Neither of the two previous methods have explicitly considered maintaining the diversity of the population. The rank space method achieves this by linking fitness to diversity and quality rank. When selecting an individual for inclusion into the reproduction scheme, the diversity can be measured by calculating the inverse squared distance between the individual and the previously selected members. The diversity rank of an individual is determined by the inverse squared distance sum.

3.5.4.3 Fitness Scaling

Essentially, fitness scaling is a technique that ensures that all fitness values are positive and that the probabilities of selection of the individuals in the population are such that there are no biases in the selection algorithm. This prevents any "super fit" individuals from dominating the population and causing the method to locate a sub-optimal solution.

The method is fairly simple. The range of fitness is calculated and these values are shifted into the positive region. Scaling can then occur by using either linear or logarithmic rules so that the range is increased or decreased to reduce any bias in selection.

3.6 Genetic Algorithm Used in This Study

The GA used in previous work utilised a simple GA based on the three main operators of selection, crossover and mutation and operated under an elitist strategy. The effectiveness of the GA was assessed by testing it on several standard numerical objective functions, such as Rosenbrock's Banana function as well as on several problems involving the generation of a desired coupler curve of a four bar mechanism. Initially, a ranked fitness method was used but this was shown to bias the GA due to a poor method of dealing with constraints. Changing the fitness evaluation to a standard fitness methods and introducing penalty functions for various constraints improved the performance dramatically, even though the GA was being applied to more complex problems.

The GA used in this study is a refined version of that used in early work. It is based upon the three main operators and also runs under an elitist strategy. An inversion operator is included in an effort to eliminate identical strings from the population. As each new child string is created, it is tested against all strings currently in the new population. If an identical string is found, the new child string is inverted.

In addition to this enhancement, the method of fitness evaluation has been changed to include a simple fitness scaling routine. In this routine, the fitness of each individual is scaled with respect to the sum of the fitnesses for the current generation. As the number of generations increases the average fitness of the population improves. Therefore, as the population becomes more fit, the difference between similar solutions becomes more pronounced. At the start of a solution run this has great effect in preventing the GA from becoming dominated from super individuals and converging to a sub-optimal solution. Towards the end of a solution run, the differences between individuals becomes more pronounced, so improving the local search power of the method.

In section 2.6, a wide variety of improvements for Genetic Algorithms were outlined. Some of these are more relevant to mechanism synthesis than others. The following methods should be considered to be included into the method in future work. The

sharing function of **Goldberg** [77] or the crowding method developed by **De Jong** [92] could be included to promote speciation with in the GA population. This would improve the robustness of the method as a varied population tends to occupy all local optima with in the search space. Including the use of introns in conjunction with crowding, as proposed by **Levenick** [111], has been shown to vastly improve search power. One other possibility that should be considered for future work would be the development of mutation operators either based upon localised hill climbing algorithm or the inclusion of a degree of intelligence in the form of heuristic rules.

3.7 Summary

This Chapter has introduced some concepts behind the working of GAs. During early work by **Connor *et al*** [37,38,89], a GA was developed and tested on a number of standard test functions such as Rosenbrock's banana function. The performance of this GA was analysed and improved by eliminating biases in the coding and solution representation. The GA was then tested on a number of mechanism synthesis problems, and further refined until performance on complex problems was acceptable.

GAs can be summarised by the following points;

- GAs are a randomised, but not random, search method based on the mechanics of natural selection and survival of the fittest.

- GAs optimise the trade off between exploring new points in the search space and exploiting the information discovered thus far.

- GAs exhibit the property of implicit parallelism. This means that an extensive search of the hyperplanes of the solution space can be carried out without having to search all the hyperplane values. This is an implication of the schema theorem, where each schema represents a hyperplane.

- GAs are a randomised, but not random, procedure. They utilise operators that are governed by probabilistic rules, not deterministic rules.

- GAs operate on a population of solutions, not a single solution. The use of multiple solutions makes the search less susceptible to becoming trapped in local optima.

Chapter Four

Hybrid Five Bar Mechanism Analysis and Synthesis

4.1 Introduction

The purpose of this Chapter is to outline the techniques used in the analysis of hybrid five bar mechanisms and show how such analysis routines can be use in conjunction with a Genetic Algorithm to provide an effective synthesis tool. This Chapter describes the kinematic analysis of five bar mechanisms based upon the vector loop equations and also outlines the dynamic analysis using the Lagrangian method. This analysis is customised to the requirements of the synthesis method.

4.2 Five Bar Mechanism Notation

Figure 4.1 shows the notation now in use for the analysis of the five bar mechanism. Each 'stick' in the diagram represents a link between revolute joints. The link t is the ground link, and so all motions in the mechanism are relative to this fixed datum. The link p is the CV input and so rotates fully around it's ground point. The link s is the servo motor input and has a programmable motion.

As the five bar is a two degree of freedom mechanism, it requires two inputs to be defined if the mechanism is to be fully analysed. In the following analysis, these two inputs are θ_2 and θ_3. θ_2 is the input angle associated with the CV motor position. θ_3 can be calculated for each position of the CV motor. This is done by referring to the desired output motion of the end effector as described in section 4.7.1. The analysis is then based around calculating the other angles of the mechanism, θ_4 and θ_5.

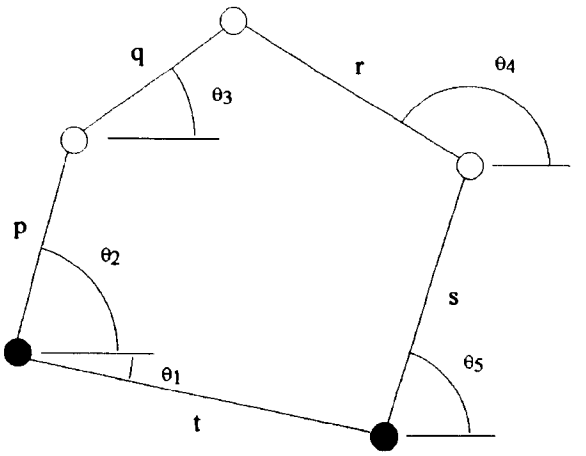

Figure 4.1 : Five Bar Notation

4.3 Vector Loop Displacement Analysis

Figure 4.2 shows a five bar mechanism represented as a set of vectors. The addition of the vectors leads to the vector loop equation;

$$\underline{p} + \underline{q} - \underline{r} - \underline{s} - \underline{t} = 0 \qquad \ldots(4.1)$$

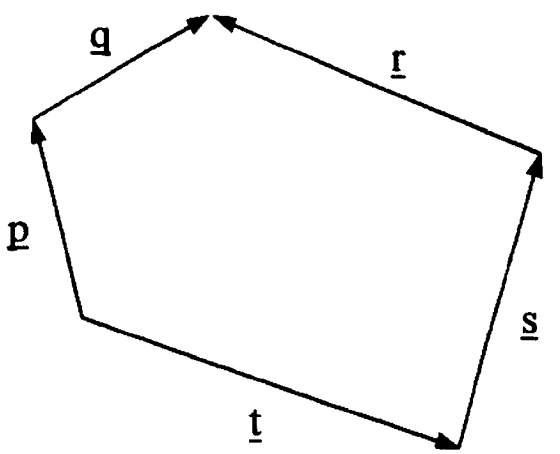

Figure 4.2 : Vector Loop Representation

The lengths of the vectors are defined by the link lengths of the mechanism, and the angles are shown in Figure 4.1. Of these angles, all but θ_4 and θ_5 are known. By expanding the loop equation it is possible to solve for these angles.

Expressing the equation in complex polar form;

$$pe^{j\theta 2} + qe^{j\theta 3} - re^{j\theta 4} - se^{j\theta 5} - te^{j\theta 1} = 0 \qquad ...(4.2)$$

Substituting the Euler equivalent form and separating into real and imaginary parts;

$$p\cos\theta_2 + q\cos\theta_3 - r\cos\theta_4 - s\cos\theta_5 - t\cos\theta_1 = 0 \quad \{real\} \qquad ...(4.3)$$

$$p\sin\theta_2 + q\sin\theta_3 - r\sin\theta_4 - s\sin\theta_5 - t\sin\theta_1 = 0 \quad \{imaginary\} \qquad ...(4.4)$$

This set of two equations in two unknowns may be solved for the unknown terms θ_5 and θ_4. It is first necessary to calculate the servo motor angle, θ_5 for use in the closure tracking algorithm to determine the mechanism closure for each step. Therefore, θ_4 is isolated.

$$r\cos\theta_4 = p\cos\theta_2 + q\cos\theta_3 - s\cos\theta_5 - t\cos\theta_1 \qquad ...(4.5)$$

$$r\sin\theta_4 = p\sin\theta_2 + q\sin\theta_3 - s\sin\theta_5 - t\sin\theta_1 \qquad ...(4.6)$$

θ_4 can be eliminated by squaring and adding each equation. This action yields the following equation.

$$r^2 = p^2 + q^2 + s^2 + t^2 + 2(p\cos\theta_2\, q\cos\theta_3) + 2(p\sin\theta_2\, q\sin\theta_3) - 2(p\cos\theta_2\, s\cos\theta_5) + 2(p\sin\theta_2\, s\sin\theta_5) - 2(p\cos\theta_2\, t\cos\theta_1) - 2(p\sin\theta_2\, t\sin\theta_1) - 2(q\cos\theta_3\, s\cos\theta_5) + 2(q\sin\theta_3\, s\sin\theta_5) - 2(q\cos\theta_3\, t\cos\theta_1) - 2(q\sin\theta_3\, t\sin\theta_1) + 2(s\cos\theta_5\, t\cos\theta_1) + 2(s\sin\theta_5\, t\sin\theta_1) \quad ...(4.7)$$

By collecting terms together and substituting the trigonometric identity;

$$(\cos\theta\cos\phi) + (\sin\theta\sin\phi) \equiv \cos(\theta - \phi)$$

this can be simplified to;

$$r^2 = p^2 + q^2 + s^2 + t^2 + 2pq\cos(\theta_2 - \theta_3) - 2pt\cos(\theta_2 - \theta_1) - 2qt\cos(\theta_3 - \theta_1) - 2ps\{(\cos\theta_2\cos\theta_5) + (\sin\theta_2\sin\theta_5)\} - 2qs\{(\cos\theta_3\cos\theta_5) + (\sin\theta_3\sin\theta_5)\} + 2st\{(\cos\theta_5\cos\theta_1) + (\sin\theta_5\sin\theta_1)\} \quad ...(4.8)$$

The known terms in this equation can be collected into a single term.

$$Z = r^2 - p^2 - q^2 - s^2 - t^2 - 2pq\cos(\theta_2 - \theta_3) - 2pt\cos(\theta_2 - \theta_1) + 2qt\cos(\theta_3 - \theta_1) \quad ...(4.9)$$

So eqn. 4.8 can be expressed as;

$$Z = -2ps\{(\cos\theta_2\cos\theta_5) + (\sin\theta_2\sin\theta_5)\} - 2qs\{(\cos\theta_3\cos\theta_5) + (\sin\theta_3\sin\theta_5)\} + 2st\{(\cos\theta_5\cos\theta_1) + (\sin\theta_5\sin\theta_1)\} \quad ...(4.10)$$

By substituting appropriate half angle formulae, eqn. 4.10 can be rearranged and solved as a quadratic equation.

The following half angle formulae are used;

$$\sin\theta_5 = 2x_5/(1+x_5^2)$$
$$\cos\theta_5 = (1-x_5^2)/(1+x_5^2)$$

where;

$$x_5 = \tan\frac{\theta_5}{2}.$$

The quadratic equation is then;

$$(2ps\cos\theta_2 + 2qs\cos\theta_3 - 2st\cos\theta_1 - Z)\,x_5^2 + (4st\sin\theta_1 - 4ps\sin\theta_2 - 4qs\sin\theta_3)\,x_5 + 2ps\cos\theta_2 - 2qs\cos\theta_3 + 2st\cos\theta_1 - Z = 0 \quad \ldots(4.11)$$

Eqn. 4.11 can be solved using the standard formula and then the value for θ_5 calculated from the half angle formulae. Similarly, θ_4 can be calculated from the following equation.

$$x_4 = \frac{1 - \{(p\cos\theta_2 + q\cos\theta_3 - s\cos\theta_5 - t\cos\theta_1)/r\}}{\{(p\sin\theta_2 + q\sin\theta_3 - s\sin\theta_5 - t\sin\theta_1)/r\}} \quad \ldots(4.12)$$

where;

$$x_4 = \tan\frac{\theta_4}{2}.$$

Once all angles are known, for each position, it is possible to calculate the angular velocities and accelerations of each link by using the derivatives of the vector loop equations.

4.4 Velocity Analysis

The position loop equation for the five bar mechanism was expressed in polar form in eqn. 4.7. Differentiating this with respect to time, noting that θ_1 is constant, leads to the velocity loop equation, where ω_n is the angular velocity of link n.

$$p\,\omega_2 j e^{j\theta 2} + q\,\omega_3 j e^{j\theta 3} - r\,\omega_4 j e^{j\theta 4} - s\,\omega_5 j e^{j\theta 5} = 0$$

...(4.13)

Substituting the Euler equivalent form and separating into real and imaginary parts;

$$-p\,\omega_2 \sin\theta_2 - q\,\omega_3 \sin\theta_3 + r\,\omega_4 \sin\theta_4 + s\,\omega_5 \sin\theta_5 = 0 \quad \{\text{real}\}$$

...(4.14)

$$p\,\omega_2 \cos\theta_2 + q\,\omega_3 \cos\theta_3 - r\,\omega_4 \cos\theta_4 - s\,\omega_5 \cos\theta_5 = 0 \quad \{\text{imaginary}\}$$

...(4.15)

For the hybrid configuration, ω_2 is the angular velocity of the CV motor and as such is known. ω_5 is the velocity of the programmable servo motor, and because the discrete displacements of the motor are known it is possible to calculate the velocity at each step by using a central difference formula. Therefore, the only unknowns are ω_3 and ω_4. As there are two equations in terms of two unknowns it is possible to calculate ω_3 and ω_4.

$$\omega_3 = \frac{2 \sin\theta_3 \{p\,\omega_2 \sin(\theta_2 - \theta_4) + s\,\omega_5 \sin(\theta_4 - \theta_5)\}}{q\,\{\cos(\theta_3 - 2\theta_4) - \cos\theta_3\}}$$

...(4.16)

and

$$\omega_4 = \frac{p\,\omega_2 \sin\theta_2 + q\,\omega_3 \sin\theta_3 - s\,\omega_5 \sin\theta_5}{r \sin\theta_4}$$

...(4.17)

4.5 Acceleration Analysis

The velocity vector loop equation (eqn. 4.13) is given as;

$$p\,\omega_2 j e^{j\theta 2} + q\,\omega_3 j e^{j\theta 3} - r\,\omega_4 j e^{j\theta 4} - s\,\omega_5 j e^{j\theta 5} = 0$$

Differentiating this equation with respect to time leads to the acceleration vector loop equation where α_n is the angular acceleration of link n.

$$(p\,\alpha_2 j e^{j\theta 2} - p\,\omega_2^2 e^{j\theta 2}) + (q\,\alpha_3 j e^{j\theta 3} - q\,\omega_3^2 e^{j\theta 3}) - (r\,\alpha_4 j e^{j\theta 4} - r\,\omega_4^2 e^{j\theta 4}) - (s\,\alpha_5 j e^{j\theta 5} - s\,\omega_5^2 e^{j\theta 5}) = 0 \qquad ...(4.18)$$

Note that the angular acceleration α_2, associated with link p, is zero, as this link has constant angular velocity. It is the input to the mechanism provided by the CV motor.

Substituting the Euler equivalents and separating into real and imaginary parts;

$$p\,\omega_2^2 \cos\theta_2 - q\,\alpha_3 \sin\theta_3 - q\,\omega_3^2 \cos\theta_3 + r\,\alpha_4 \sin\theta_4 + r\,\omega_4^2 \cos\theta_4 + s\,\alpha_5 \sin\theta_5 + s\,\omega_5^2 \cos\theta_5 = 0 \quad \{\text{real}\} \qquad ...(4.19)$$

$$p\,\omega_2^2 \sin\theta_2 + q\,\alpha_3 \cos\theta_3 - q\,\omega_3^2 \sin\theta_3 - r\,\alpha_4 \cos\theta_4 + r\,\omega_4^2 \sin\theta_4 - s\,\alpha_5 \cos\theta_5 + s\,\omega_5^2 \sin\theta_5 = 0 \quad \{\text{imaginary}\} \qquad ...(4.20)$$

The only two unknowns are α_3 and α_4. α_3 can be found by direct substitution.

$$\alpha_3 = \frac{[-p\,\omega_2^2 \cos(\theta_2-\theta_4) - q\,\omega_3^2 \cos(\theta_3 - \theta_4) + s\,\omega_5^2 \cos(\theta_5 - \theta_4) + s\,\alpha_5 \sin(\theta_5 - \theta_4) + r\,\omega_4^2\,]}{q \sin(\theta_3 - \theta_4)} \qquad ...(4.21)$$

α_4 can now be calculated directly.

$$\alpha_4 = \frac{[-p\,\omega_2^2 \sin\theta_2 + q\,\alpha_3 \cos\theta_3 - q\,\omega_3^2 \sin\theta_3 + r\,\omega_4^2 \sin\theta_4 - s\,\alpha_5 \cos\theta_5 + s\,\omega_5^2 \sin\theta_5]}{r\cos\theta_4}$$

...(4.22)

Once all the displacements, angular velocities and angular accelerations of the mechanism are known for all stages of the cycle, it is possible to start analysing the dynamic performance of the mechanism.

4.6 Dynamic Analysis of Five Bar Mechanisms

The dynamic analysis of mechanisms can be carried out using a variety of different techniques. The Newtonian kinetostatic method involves the inversion of large matrices but gives the values of joint forces. However, as only the values for motor torque are required a Lagrangian approach has been used. The Lagrange formulation equation is given as eqn. 4.23.

$$\frac{d}{dt}\left\{\frac{\partial K}{\partial \dot{q}}\right\} - \frac{\partial (K-U)}{\partial q} = Q$$

...(4.23)

where;

Q is the generalised force associated with q

K is the total kinetic energy of the system

U is the total potential energy of the system

q_i is an independent generalised co-ordinate

For each link there is a value of K and U, each of which may be summed together to produce the systems total kinetic and potential energy. Hence, for a link of mass m_i, having an inertia I_i, a joint rotation θ_i and displacements x_i and y_i at its centre of mass, the following is true;

$$K_i = \frac{1}{2} I_i \dot{\theta}_i^2 + \frac{1}{2} m_i (\dot{x}_i^2 + \dot{y}_i^2)$$

...(4.24)

Differentiating eqn. 4.24 with respect to $\dot{q}_i$ gives;

$$\frac{\partial K_i}{\partial \dot{q}_i} = I_i \dot{\theta}_i \frac{\partial \dot{\theta}_i}{\partial \dot{q}_i} + m_i \left\{ \dot{x}_i \frac{\partial \dot{x}_i}{\partial \dot{q}_i} + \dot{y}_i \frac{\partial \dot{y}_i}{\partial \dot{q}_i} \right\}$$

...(4.25)

By cancelling the dot terms a simpler equation is produced;

$$\frac{\partial K_i}{\partial \dot{q}_i} = I_i \dot{\theta}_i \frac{\partial \theta_i}{\partial q_i} + m_i \left\{ \dot{x}_i \frac{\partial x_i}{\partial q_i} + \dot{y}_i \frac{\partial y_i}{\partial q_i} \right\}$$

...(4.26)

Further differentiation of eqn. 4.26 with respect to time yields;

$$\frac{d}{dt}\left\{\frac{\partial K_i}{\partial q}\right\} = m_i \left\{ \ddot{x}_i \frac{\partial x_i}{\partial q_i} + \dot{x}_i \frac{d}{dt}\left(\frac{\partial x_i}{\partial q_i}\right) + \ddot{y}_i \frac{\partial y_i}{\partial q_i} + \dot{y}_i \frac{d}{dt}\left(\frac{\partial y_i}{\partial q_i}\right) \right\} + I_i \left\{ \ddot{\theta}_i \frac{\partial \theta_i}{\partial q_i} + \dot{\theta}_i \frac{d}{dt}\left(\frac{\partial \theta_i}{\partial q_i}\right) \right\}$$

...(4.27)

Also, assuming that the linkage is in the vertical plane;

$$U_i = m_i g y_i$$

...(4.28)

Then the following equation can be derived;

$$\frac{\partial (K-U)_i}{\partial q_i} = m_i \left\{ \dot{x}_i \frac{d}{dt}\left(\frac{\partial x_i}{\partial q_i}\right) + \dot{y}_i \frac{d}{dt}\left(\frac{\partial y_i}{\partial q_i}\right) \right\} - m_i g \frac{\partial y_i}{\partial q_i} + I_i \dot{\theta}_i \frac{d}{dt}\left\{\frac{\partial \theta_i}{\partial q_i}\right\}$$

...(4.29)

By combining eqn. 4.29 with eqn. 4.27 according to the relationship given in eqn. 4.23 the contribution that each link makes to the generalised force Q_i can be written as ϕ_i, where;

$$\phi_i = I_i\alpha_i \frac{\partial\theta_i}{\partial q_i} + m_i\left\{\ddot{x}_i \frac{\partial x_i}{\partial q_i} + \ddot{y}_i \frac{\partial y_i}{\partial q_i}\right\} + m_i g \frac{\partial y_i}{\partial q_i}$$

...(4.30)

At this point in the analysis, the rotational displacements of the motor driven input links, θ_2 and θ_5, will be considered as the independent variables, since the torque requirements of these two positions are required.

Before eqn. 4.30 can be solved, the following terms must be calculated;

$$\frac{\partial\theta_i}{\partial q_i}, \frac{\partial x_i}{\partial q_i}, \frac{\partial y_i}{\partial q_i}, \ddot{x}_i \text{ and } \ddot{y}_i$$

The rate at which each joint rotation varies with the independent variables can be found by differentiating the vector loop equations with respect to θ_5, the servo input;

$$p\cos\theta_2 \frac{\partial\theta_2}{\partial\theta_5} + q\cos\theta_3 \frac{\partial\theta_3}{\partial\theta_5} - r\cos\theta_4 \frac{\partial\theta_4}{\partial\theta_5} - s\cos\theta_5 = 0$$

...(4.31)

$$-p\sin\theta_2 \frac{\partial\theta_2}{\partial\theta_5} - q\sin\theta_3 \frac{\partial\theta_3}{\partial\theta_5} + r\sin\theta_4 \frac{\partial\theta_4}{\partial\theta_5} + s\sin\theta_5 = 0$$

...(4.32)

These are two equations with two unknowns, and can be solved to find $\frac{\partial\theta_3}{\partial\theta_5}$ and $\frac{\partial\theta_4}{\partial\theta_5}$.

In addition to these values, it is necessary to calculate the partial linear derivatives and the linear accelerations. All links of the mechanism are assumed to be symmetrical and the centre of mass is located at the midpoint. Given that the co-

ordinates of the CV motor are (e,f) and the co-ordinates of the servo motor are (g,h), the co-ordinates of the centre of masses for each link are found by;

$$x_2 = e + (0.5\ p \cos\theta_2)$$

...(4.33)

$$y_2 = f + (0.5\ p \sin\theta_2)$$

...(4.34)

$$x_3 = e + 2x_2 + (0.5\ q \cos\theta_3)$$

...(4.35)

$$y_3 = f + 2y_2 + (0.5\ q \sin\theta_3)$$

...(4.36)

$$x_5 = g + (0.5\ s \cos\theta_5)$$

...(4.37)

$$y_5 = h + (0.5\ s \sin\theta_5)$$

...(4.38)

$$x_4 = g + 2x_5 + (0.5\ r \cos\theta_5)$$

...(4.39)

$$y_4 = h + 2y_5 + (0.5\ r \sin\theta_4)$$

...(4.40)

Differentiating these equations twice with respect to time;

$$\ddot{x}_2 = -0.5p\left\{\ddot{\theta}_2 \sin\theta_2 + \dot{\theta}_2^2 \cos\theta_2\right\}$$

...(4.41)

$$\ddot{y}_2 = 0.5p\left\{\ddot{\theta}_2 \cos\theta_2 - \dot{\theta}_2^2 \sin\theta_2\right\}$$

...(4.42)

$$\ddot{x}_3 = 2\ddot{x}_2 - 0.5q\left\{\ddot{\theta}_3 \sin\theta_3 + \theta_3^2 \cos\theta_3\right\}$$

...(4.43)

$$\ddot{y}_3 = 2\ddot{y}_2 + 0.5q\left\{\ddot{\theta}_3 \cos\theta_3 - \theta_3^2 \sin\theta_3\right\}$$

...(4.44)

$$\ddot{x}_5 = -0.5s\left\{\ddot{\theta}_5 \sin\theta_5 + \dot{\theta}_5^2 \cos\theta_5\right\}$$

...(4.45)

$$\ddot{y}_5 = 0.5s\left\{\ddot{\theta}_5 \cos\theta_5 - \dot{\theta}_5^2 \sin\theta_5\right\}$$

...(4.46)

$$\ddot{x}_4 = 2\ddot{x}_5 - 0.5r\left\{\ddot{\theta}_4 \sin\theta_4 + \theta_4^2 \cos\theta_4\right\}$$

...(4.47)

$$\ddot{y}_4 = 2\ddot{y}_5 + 0.5r\left\{\ddot{\theta}_4 \cos\theta_4 - \theta_4^2 \sin\theta_4\right\}$$

...(4.48)

This set of equations give a comprehensive set of values for the linear accelerations in the mechanism. Differentiating eqns. 4.33 to 4.40 with respect to the rotational displacement of the servo motor, θ_5, gives;

$$\frac{\partial x_2}{\partial \theta_5} = -0.5p \sin\theta_2 \frac{\partial \theta_2}{\partial \theta_5}$$

...(4.49)

$$\frac{\partial y_2}{\partial \theta_5} = 0.5p \cos\theta_2 \frac{\partial \theta_2}{\partial \theta_5}$$

...(4.50)

$$\frac{\partial x_3}{\partial \theta_5} = 2\frac{\partial x_2}{\partial \theta_5} - 0.5q \sin\theta_3 \frac{\partial \theta_3}{\partial \theta_5}$$

...(4.51)

$$\frac{\partial y_3}{\partial \theta_5} = 2\frac{\partial y_2}{\partial \theta_5} + 0.5q \cos\theta_3 \frac{\partial \theta_3}{\partial \theta_5}$$

...(4.52)

$$\frac{\partial x_5}{\partial \theta_5} = -0.5s \sin\theta_5$$

...(4.53)

$$\frac{\partial y_5}{\partial \theta_5} = 0.5s \cos\theta_5$$

...(4.54)

$$\frac{\partial x_4}{\partial \theta_5} = 2\frac{\partial x_5}{\partial \theta_5} - 0.5r\sin\theta_4 \frac{\partial \theta_4}{\partial \theta_5}$$

...(4.55)

$$\frac{\partial y_4}{\partial \theta_5} = 2\frac{\partial y_5}{\partial \theta_5} + 0.5r\cos\theta_4 \frac{\partial \theta_4}{\partial \theta_5}$$

...(4.56)

As all of the partial angular derivatives for θ_5 are known each of the previous equations can be evaluated. As each unknown system variable can now be evaluated for each link it is now possible to calculate the total torque required at the servo motor input by summing all the values of ϕ_i as calculated using eqn. 4.30.

Using this approach it is possible to calculate the total torque requirement for the servo motor around the cycle of the mechanism. It is also possible to calculate the CV motor torque requirements using a similar method.

The rest of this Chapter illustrates how kinematic objective functions can be developed and synthesis using a GA can be carried out. Dynamic verification of some results are included.

4.7 The Synthesis of Five Bar Mechanisms

The five bar mechanism is a two degree of freedom mechanism which requires two inputs to fully define the output motion. Such a mechanism was shown in Figure 4.1, where the two inputs to the mechanism were provided by links p and s with reference to the fixed link t.

For the sake of the synthesis it is assumed that the motion of the servo motor is not known. Therefore, to fully analyse the mechanism it is necessary to define an alternative input. To do this it is possible to use the desired position of the end effector. The synthesis problem can then be expressed as a search for the mechanism

link lengths and the servo motor input displacements which provide the desired end effector motion.

The actual parameters which have been selected for use as variables are the lengths of the links p,q,r and s. During the search, the lengths of these links are constrained between 10 and 266 units. The length of the ground link t is not an explicit variable. This length is defined by the ground positions of the inputs. Each of these is defined by x,y co-ordinates constrained within a 16 by 16 unit constraint envelope. The global position of each constraint envelope is defined by the user.

When a solution consisting of eight variables (four link lengths, two pairs of x,y co-ordinates) is represented in a binary string so that the variables are constrained as outlined above the length of the binary string is 48 bits. This relates to a solution space of 2^{48}-1 possible solutions.

Once the problem has been expressed in these terms, it is possible to describe how an objective function may be constructed so that an appropriate mechanism may be synthesised. the following sections outline several criteria that can be used in the objective function. In section 4.9, a number of experiments are presented that show which criteria are most effective.

4.7.1 Coupler Curve Error

For the sake of the analysis and synthesis of the five bar mechanism, it is assumed that the end effector is the revolute joint between links q and r. The input motion for link p is known, and so by defining the actual position of the end effector it is possible to fully describe the motions in the mechanism. Figure 4.3 illustrates how the positions of links p and q can be used to define the required angles of the mechanism so that it may be analysed fully and also calculate an error score.

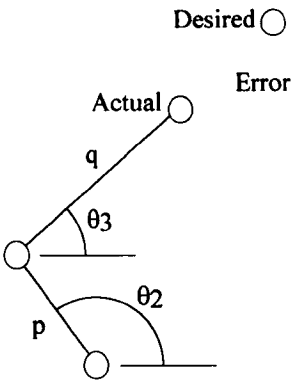

Figure 4.3 : Calculation of Error

A vector is defined from the end of link p to the desired position of the end effector for the given value of θ_2. The actual position of this point is then calculated, given the length of link q, and the error between the two points found. This error is summed around the cycle for the input as described in eqn. 4.57, where the subscripts *d* and *a* correspond to desired and actual coordinate positions and n is the number of precision points.

$$error = \sum_{1}^{n} \sqrt{(x_d - x_a)^2 + (y_d - y_a)^2} \qquad ...(4.57)$$

Once the position of the end effector is known, it is possible to calculate the value of θ_3 and then calculate the other angles of the mechanism.

Once the error score around the cycle has been calculated, it can be used to calculate the objective function in many ways. Experimentation has shown that both speed of convergence and quality of solution are improved by raising the error score to a given power. However, if too high a power is used, then in a multiple objective search the function is dominated by the error score. All results in this thesis have been obtained using the objective function criterion given in eqn.4.58.

$$obj_{err} = error^2 \qquad ...(4.58)$$

4.7.2 Mechanism Mobility

In section 4.7.1 it was shown how the two links, p and q, form a dyad which can be used to evaluate the error at a given point for a given input angle. Similarly, the remaining two mobile links, r and s, form another dyad which is used to calculate a penalty function criteria of the overall objective function.

This penalty function is based on the mobility of the mechanism. For a truly mobile mechanism, the dyad formed by the links r and s should be able to 'close' for all given positions of the CV input crank. This means that the position of the common revolute joint, when considered as part of the dyad formed by r and s, should be able to reach the actual position defined by the dyad formed by the links p and q.

For each position that this dyad cannot close, the mobility counter is increased. It is important to realise that for each position of the input link, there are two possible closures of the dyad. This is illustrated in Figure 4.4.

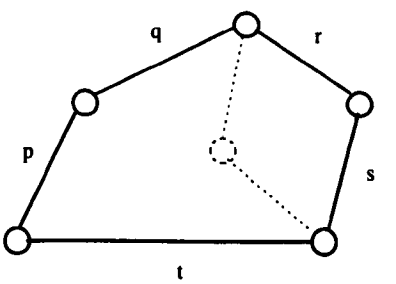

Figure 4.4 : Multiple Closures of the Mechanism

The implication of this, at this stage, is that for each position of the CV input link the mobility score may increase by two. The mobility penalty score is summed throughout the cycle and, for twenty four precision points, may range between zero and forty eight. As with the error score value, it has been found that raising the mobility penalty to a power acts so that the search is directed more towards feasible solutions.

The following expression is used to calculate the contribution of the mobility penalty function to the overall objective function.

$$obj_{mob} = mobility^3 \qquad ...(4.59)$$

4.7.3 Servo Motor Displacements

Once two inputs to the mechanism are known, in this case θ_2 and θ_3, it is possible to calculate any other angle in the mechanism. In this case, θ_5 is calculated using the equations derived in section 4.3.

The values for θ_5 around the cycle of the mechanism can be used to calculate a number of figures of merit which describe the quality of the motion profile. However, the situation is complicated by the presence of multiple closures in the mechanism. For each input position, two values for θ_5 can be calculated and the question which arises is "which value should be chosen?".

4.7.3.1 Closure Tracking Algorithm

An algorithm has been developed to choose the set of servo motor displacements for a given mechanism. The algorithm is based on the fact that it is desirable to have smooth velocity profiles for the servo motor input.

The algorithm optimises the displacement profile for a given mechanism by choosing between the two available closures for each defined position around the cycle. The algorithm acts as follows. It is assumed that the mechanism starts in the "open" closure. As the input crank angle is incremented a decision is made between the two available closures based on the previous positions selected.

The first decision is based purely on magnitude of displacement. That is, the closure where the change in displacement is smallest is selected. Subsequent decisions are based on changes in velocity and trends in direction. If the two possible closures are such that the trend in displacement is continued, then the closure with the smallest change in velocity is chosen. The trend in displacement is only broken if the change in velocity of the alternative closure is very much larger than for that which involves a change of direction.

The action of this algorithm may be explained in a more simple manner by referring to Figure 4.5.

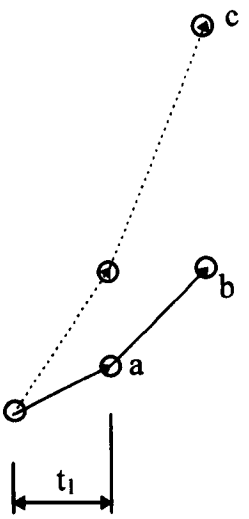

Figure 4.5a : Initial Tracking Step

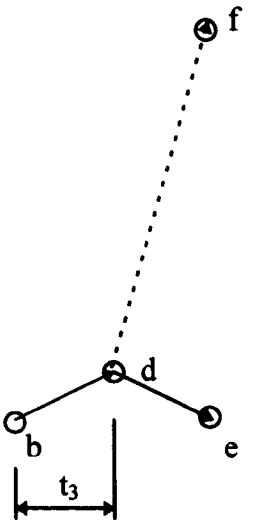

Figure 4.5b : Change of Direction

In Figure 4.5a, point a is chosen as the initial point as it has the smallest displacement from the starting position. Point b is chosen over point c at the second time step as change in velocity is smaller.

Figure 4.5b illustrates a change in direction. From point d, point e is chosen over point f, even though the algorithm is forced to maintain the same direction if possible. This rule has been over ridden as the change in velocity from d to f is too great.

Given that the servo motor displacement is now known, it is possible to calculate a figure of merit which describes the quality of the motion profile. Early work by **Connor *et al*** [89] utilised the RMS value of the displacements so that the magnitude of the servo actuation was minimised. However, the results obtained were not satisfactory due to oscillatory motions. Several alternatives have been investigated, including the area swept by link s and also the harmonic content of the motion profile.

4.7.3.2 Motion Swept Area

The concept of motion swept area was inspired by early work of **Connor *et al*** [89], where minimising the RMS of the motor displacements forced the search to locate mechanisms where the links in the closing dyad were very much longer than those in the input dyad. A mechanism with such link length ratios is unlikely to produce even an approximation to good dynamic performance due to the large torque requirements of the servo motor. In calculating the swept area term, the RMS value of the displacements is multiplied by the length of link s. By doing this, an implicit compromise is achieved between the magnitude of the displacements and the length of link s. The definition of the objective function component is given in eqn.4.60.

$$obj_{swept} = s\sqrt{\sum_{i=1}^{n}\theta_{5i}^{\,2}} \qquad \text{...(4.60)}$$

4.7.3.2 Motion Harmonic Content

In addition to investigating the use of a swept area term in the objective function, it was also decided to experiment using a term based on the harmonic content of the motion profile. This decision was also based on previous work by **Connor *et al*** [89], where minimising the RMS of the motion produced a displacement profile which

oscillated considerably. This effect was worsened by the use of a closure tracking algorithm which did not take into account direction of travel, but only differences in magnitude of velocity.

It is desirable to have smooth motion profiles, as oscillatory displacement profiles tend to produce sharp or discontinuous acceleration profiles and hence large torque and power requirements. Penalising profiles with high magnitudes in high order harmonics should direct the search towards a solution where the servo displacement profile is trending towards simple harmonic motion.

Calculating the harmonic content of a motion profile can be achieved by utilising a Fourier transform. Only a statement of numerical harmonic analysis will be given here but it is important to understand the relationship between analytical and numerical Fourier transforms as the discrete Fourier transform (DFT) is derived from such an analytical base. Essentially, the only difference between the DFT and analytical techniques is that the coefficients relating to the Fourier series are calculated by numerical integration. The required series is denoted by;

$$f(x) = \frac{1}{2}a_0 + \sum_{n=1}^{\infty}\{a_n \cos nx + b_n \sin nx\}$$

...(4.61)

The individual coefficients are calculated using the following expressions.

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

...(4.62)

$$a_n = \frac{1}{\pi}\int_0^{2\pi} f(x)\cos nx\,dx$$

...(4.63)

$$b_n = \frac{1}{\pi}\int_0^{2\pi} f(x)\sin nx\,dx$$

...(4.64)

Examination of the Fourier coefficients allows a considerable amount of information concerning the function waveform to be ascertained. In this instance the function waveform is defined as the servo motor displacement profile. For example, a simple sinusoidal waveform contains only a single (fundamental) harmonic whilst more complex waveforms contain higher order harmonics. Essentially, each harmonic of a waveform represents a sinusoid of a set frequency so that when the resulting sinusoids for a series are added by the principle of superposition the initial waveform results.

Bearing this information in mind, it is easy to visualise an objective function criteria based on harmonic content. Minimising the magnitude of the higher order harmonics will trend towards solutions with smoother servo motor profiles whilst minimising the magnitude of the fundamental harmonic will trend towards solutions where the magnitude of the servo motor displacements are small. A proposed objective function based on harmonic content is given in eqn.4.64;

$$obj_{harm} = \sum_{i=1}^{n}\left(\sqrt{a_n^2 + b_n^2}\right)^{n+1}$$

...(4.65)

In this function the magnitude of the harmonic of order "n" is raised to the power "n+1". This penalises solutions that have both large servo displacements and motions that are not smooth. To reduce the amount of calculations required, only the first five harmonics are calculated. This provides a sufficiently accurate approximation to the motion.

4.8 Motion Design

Before presenting results pertaining to the analysis of the design objectives outlined above, it is important to outline the motion design principles utilised in the development of mechanism models and associated calculations.

The output from the developed synthesis software consists of a set of mechanism dimensions and the corresponding servo motor input angles for the twenty four prescribed positions in the motion. Using only twenty four precision points ensures that the synthesis process is not impeded by excessive calculations of, for example, Fourier coefficients. However, twenty four points is insufficient to define the servo motor profile in such a way that it can easily be transferred to a practical machine.

In general, motion design can be viewed as the process by which a profile defined by a number of finite points can be transferred into terms where the position is calculable at all points of the machine cycle. This is normally achieved by the use of curve fitting techniques such as polynomial interpolation or the use of cubic splines.

The motion design principles used here are similar to those used by **Tokuz** [1], in that a given curve is split into several segments and these segments joined at boundaries by defining velocity and acceleration conditions. The curves used in this study are polynomials of degree eleven. These polynomials are of the general form;

$$y = a_0 + a_1x + a_2x^2 + ... + a_{11}x^{11} \qquad ...(4.66)$$

By using a polynomial of this degree, up to twelve coefficients are determined by the specified boundary conditions. The maximum number of boundary coefficients that can be specified is equal to the order of the polynomial, so allows not only velocity and acceleration conditions specified, but also the jerk condition. However, the more conditions specified, the higher the order of the polynomial used. In general, the

lowest order of polynomial is used for each segment which matches the boundary conditions.

The twenty four prescribed positions are transferred into a segmented polynomial approximation in the following way. Initially, a cubic spline interpolation is carried out to calculate 360 data points. This enables an approximation to the velocity, acceleration and jerk profiles to be calculated. However, the cubic spline does not take into account "wrap around" at the end of the machine cycle so often a velocity discontinuity occurs. The velocity and acceleration profiles are used to estimate the boundary conditions for the segmented profiles used in this work. The table in Figure 4.6 illustrates a typical set of boundary conditions for a seven segment approximation.

Segment no.	Input Change	Motion Constraint	Start	End
1	0° - 30°	Position	0.00	-8.35
		Velocity	-2.44	-0.76
		Acceleration	33.95	30.94
2	30° - 120°	Position	-8.35	7.95
		Velocity	-0.76	1.87
		Acceleration	30.94	-0.28
3	120° - 270°	Position	7.95	40.15
		Velocity	1.87	0.61
		Acceleration	-0.28	-2.23
4	270° - 285°	Position	40.15	41.56
		Velocity	0.61	0.63
		Acceleration	-2.23	5.31
5	285° - 318°	Position	41.56	35.02
		Velocity	0.63	-5.71
		Acceleration	5.31	134.02
6	318° - 333°	Position	35.02	19.05
		Velocity	-5.71	-6.54
		Acceleration	134.02	40.77
7	333° - 360°	Position	19.05	0.00
		Velocity	-6.54	-2.44
		Acceleration	40.77	33.95

Figure 4.6 : Segmented Polynomial Motion Profile

This is the motion calculated for the results presented in section 4.9.1.1. In this table, position data is given in degrees and velocity and acceleration data in radians per second and radians per second^2.

4.9 Computational Experiments for Objective Function Evaluation

In order to compare the effectiveness of the different objective function criteria, a simple computational experiment has been carried out. A motion curve has been defined which is required to be traced by the end effector of the mechanism. For each objective function, the GA based search was run a fixed number of generations. The search was repeated ten times for each function and the best solution from the total number of runs selected. This process of experimentation was selected because of the randomised nature of GA based search. Selecting the best solution from a number of search runs ensures that each objective function receives a fair trial and is not penalised by a single poor run.

A fairly demanding curve was defined which contained two cusps, or points of zero velocity, such as may be generated by a pick and place mechanism The following objective functions were tested on the desired curve.

1. Error and mobility
2. Error, mobility and swept area
3. Error, mobility and harmonic content
4. Error, mobility, swept area and harmonic content

The first objective function produces a benchmark result, where no criteria are used to assess the quality of servo motor profile. The other functions all use some manner of quality assessment.

The following sections presented the results for each objective function, where the method and objective functions are constant. Similarly, the tests are all carried out for the same desired output curve.

4.9.1 Results

As the purpose of the experiment was simply to assess the quality of final solution that each objective function produced, full convergence trends will not be shown. Suffice to say, that in all cases the search followed a typical GA trend with rapid initial improvement followed by gradual, but definite, improvement until the search was terminated by reaching the maximum number of generations.

The results will first be presented as a set of mechanism dimensions. These results will then be assessed by considering the error between the desired and actual curves and the torque requirements and distribution between CV and servo motor. The results and assessment will be shown for each individual mechanism then comparisons drawn in section 4.9.2.

4.9.1.1 Objective Function No. 1

As has been previously stated, this is the most simple objective function and no method was used to assess servo motor motion quality. The following mechanism dimensions were obtained;

$(x,y)_{cv} = -1,0$ $\quad$ $p = 12$

$(x,y)_{servo} = 32,1$ $\quad$ $q = 27$

$r = 236$

$s = 248$

Figure 4.7 shows the servo motor displacement profile required for this mechanism to approximate the desired output curve.

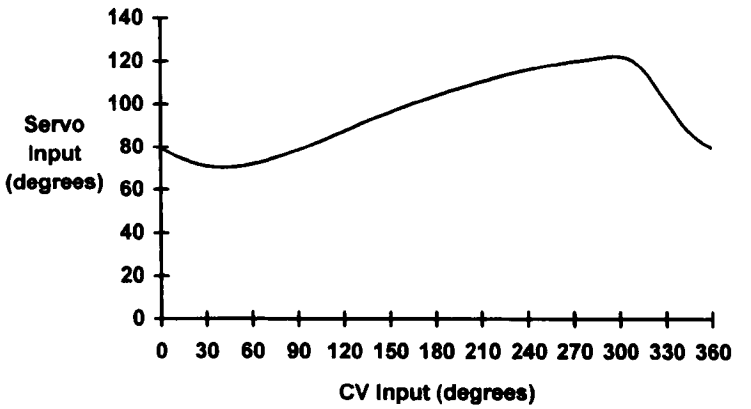

Figure 4.7 : Servo Motor Input Requirement

The end effector motion is shown in Figure 4.8. The continuous line shows the path generated by the mechanism, whilst the points are those used to define the motion. Whilst the generated curve bears a slight resemblance to the desired curve, and is within the same region of workspace, it can be deduced that this objective function has not generated a sufficiently directed search to produce a high quality output.

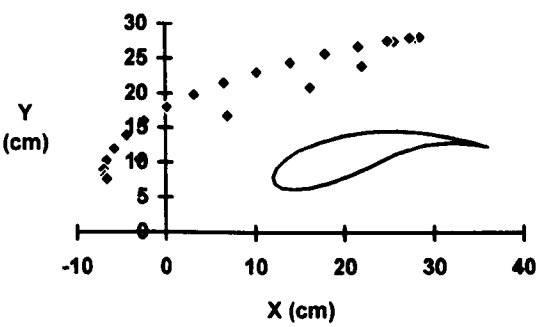

Figure 4.8 : End Effector Motion

Despite the fact that the search has produced a poor approximation to the desired motion, the torque requirements of this mechanism have been calculated. This is so

that any benefit in dynamic performance offered by alternative functions can be assessed. Figure 4.9 shows the torque requirement for both the CV and servo motors.

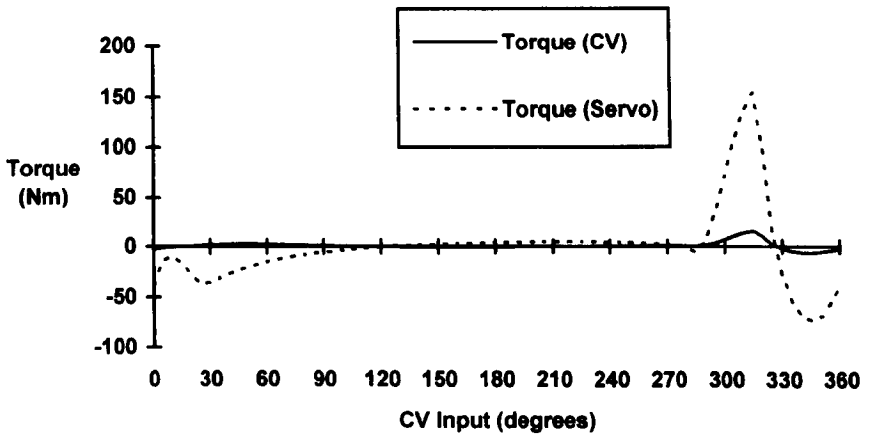

Figure 4.9 : Torque Requirements

It can be seen that this mechanism does not exhibit desirable torque characteristics. The servo motor requirement is very much larger than that of the CV motor. The torque requirement of the servo is beyond the range normally associated with servo motors, and so this is an unfeasible mechanism. This conclusion is logical, when viewed in light of the mechanism dimensions. The links in the closing dyad are very long. Therefore, whilst the servo motor displacements are quite small, a large torque is required. In addition to this, it is worth mentioning that due to the link length ratios, this mechanism is unlikely to exhibit desirable transmission characteristics.

4.9.1.2 Objective Function No. 2

In this objective function an attempt is made to reduce the length of the links in the closing dyad by incorporating the swept area of link s into the objective function, and so improve the dynamic characteristics of the mechanism. The following mechanism dimensions were obtained;

$(x,y)_{cv}$ = -2,-2 p = 13

$(x,y)_{servo}$ = 32,8 q = 27

r = 45

s = 27

Figure 4.10 shows the servo motor displacement profile required for this mechanism to approximate the desired output curve.

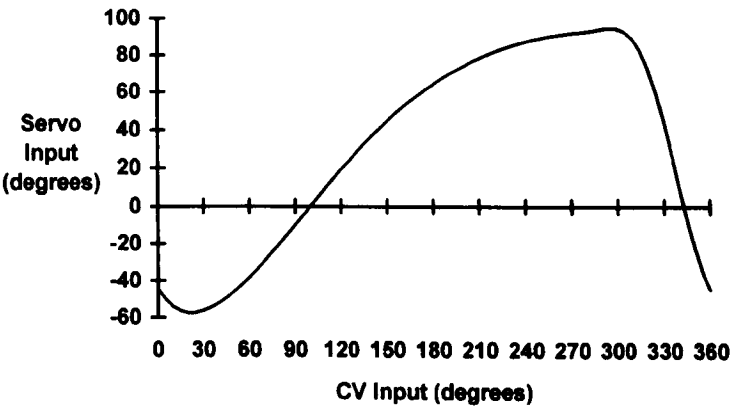

Figure 4.10 : Servo Motor Input Requirement

This mechanism requires much greater servo motor displacements to produce the output motion, as should be expected due to the shorter links in the closing dyad. The end effector motion is shown in Figure 4.11. The continuous line shows the path generated by the mechanism, whilst the points are those used to define the motion. The generated curve is a much better approximation to the desired motion.

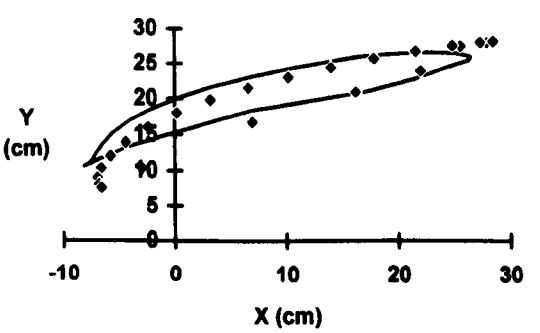

Figure 4.11 : End Effector Motion

Figure 4.12 shows the torque requirement for both the CV and servo motors. For this mechanism, the torque requirements are much lower and the distribution of torque between CV and servo motors is more desirable as the servo motor requirement is slightly smaller than that of the CV motor.

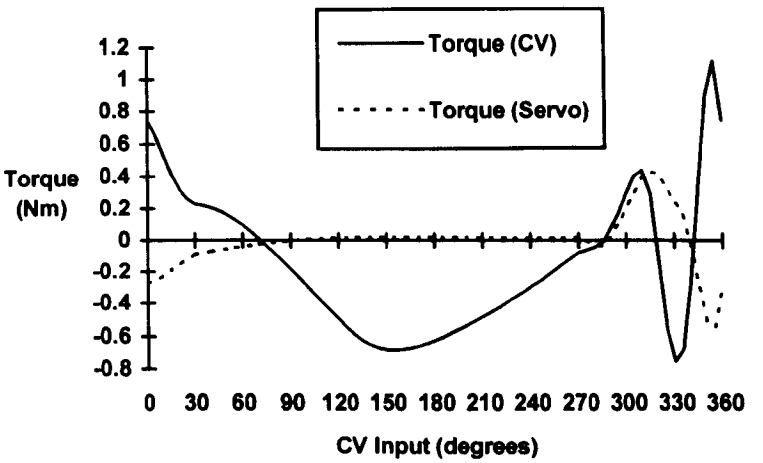

Figure 4.12 : Torque requirements

4.9.1.3 Objective Function No. 3

In this objective function an attempt is made to produce a more compact mechanism with a servo motor displacement profile with low harmonic content, and so improve the dynamic characteristics of the mechanism. The following mechanism dimensions were obtained;

$(x,y)_{cv}$ = -2,-2

$(x,y)_{servo}$ = 30,-7

p = 14

q = 26

r = 30

s = 40

Figure 4.13 shows the servo motor displacement profile required for this mechanism to approximate the desired output curve.

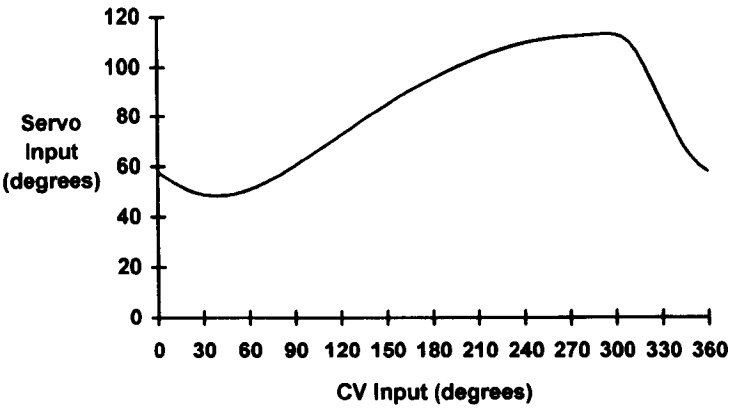

Figure 4.13 : Servo Motor Input Requirement

This motion profile is more compact than that generated using the previous objective function. The effect of raising the harmonic value to the power of the order plus one is forcing the motion profile to become both smooth and have smaller magnitude of displacements. However, the cost is that the links in the closing dyad are slightly longer The end effector motion is shown in Figure 4.14. The continuous line shows the path generated by the mechanism, whilst the points are those used to define the motion.

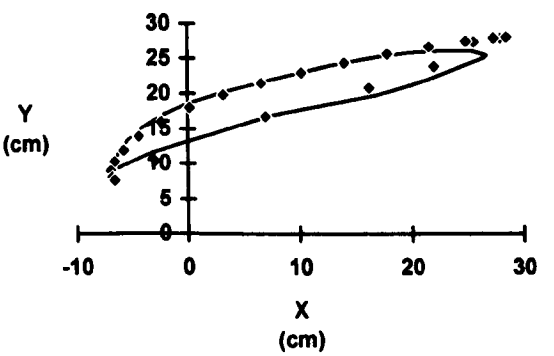

Figure 4.14 : End Effector Motion

Figure 4.15 shows the torque requirement for both the CV and servo motors. For this mechanism, the torque requirement of the servo motor is higher than that required by

the mechanism in section 4.9.1.2. This is probably due to the increase in link lengths. However, the torque distribution between CV and servo motors is still reasonable.

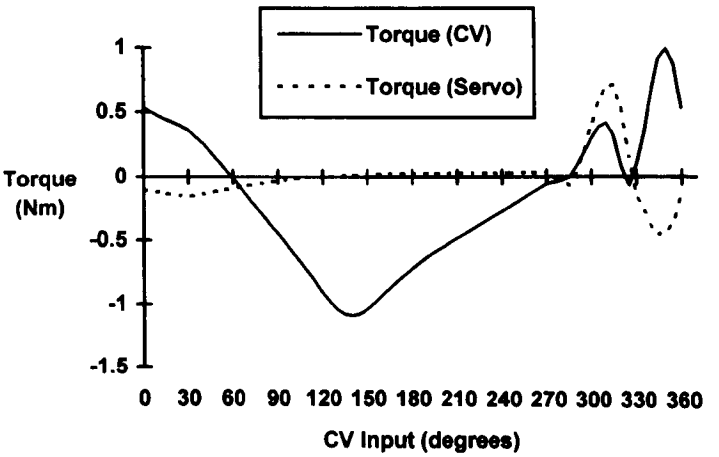

Figure 4.15 : Torque Requirements

4.9.1.4 Objective Function No. 4

In this objective function an attempt is made to combine the benefits of both minimising the harmonic content of the motion profile, as well as the swept area of the link a_{45}. By doing this, the aim is to find a compact mechanism with acceptable motion profiles and dynamic characteristics. The following dimensions were obtained;

$(x,y)_{cv} = 0,1$ $\quad$ $p = 16$

$(x,y)_{servo} = 27,-2$ $\quad$ $q = 24$

$r = 30$

$s = 16$

Figure 4.16 shows the servo motor displacement profile required for this mechanism.

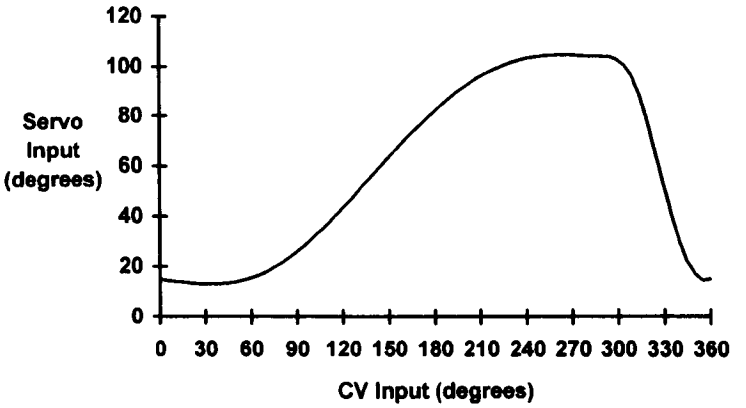

Figure 4.16 : Servo Motor Input Requirement

The end effector motion is shown in Figure 4.17. The continuous line shows the path generated by the mechanism, whilst the points are those used to define the motion.

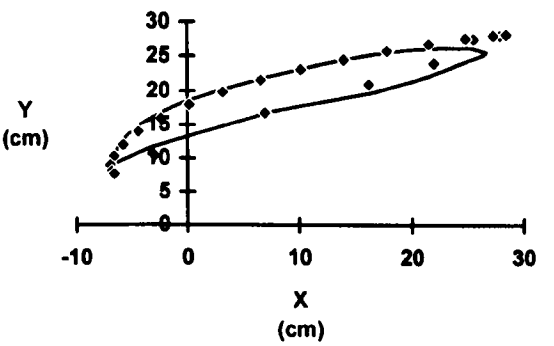

Figure 4.17 : End Effector Motion

Figure 4.18 shows the torque requirement for both the CV and servo motors. For this mechanism, the torque requirement of the servo motor is much lower than that required by the CV motor.

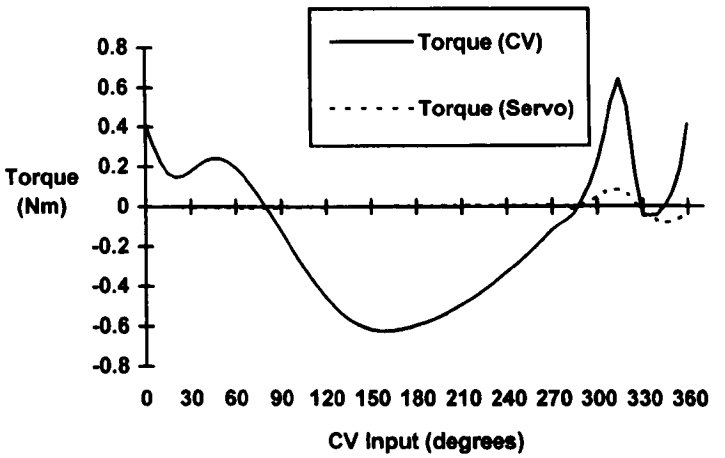

Figure 4.18 : Torque Requirements

4.9.2 Comparison of Results

These initial results can be used to assess the effectiveness of each of the objective functions by comparing the quality of the solutions obtained. In section 1.2.1 an assumption for dynamic optimality was stated. This is where the CV motor torque requirement is much larger than the servo motor torque requirement. The table in Figure 4.19 shows the minimum, maximum and RMS torque requirements.

	Function 1	**Function 2**	**Function 3**	**Function 4**
Error	67.49	39.37	34.69	16.78
CV T_{min}	-6.9011	-0.7559	-1.0869	-0.6247
CV T_{max}	15.4815	1.1128	0.9962	0.6402
CV T_{RMS}	6.3901	0.8016	0.9811	0.6491
Servo T_{min}	-73.0237	-0.5428	-0.4474	-0.0786
Servo T_{max}	153.243	0.4285	0.7106	0.0834
Servo T_{RMS}	65.5106	0.2775	0.3429	0.0449

Figure 4.19 : Motor Torque Requirements

These results show that the solution found using the final objective function exhibits both the lowest servo motor torque requirement and also the best torque distribution.

4.10 Further Experimentation

In addition to the results highlighted above, several other computational experiments have been carried out. Not all results will be presented here, though the implications of some of the early experiments will be explained.

Initial experiments were carried out to try and find the optimum weightings for the different components in the selected objective function. These experiments consisted of a trial and error approach, where the method was run several times on different problems utilising the same weighting parameters and the results analysed. It was observed that the GA was quite robust in terms of variations in the weighting parameters, and also the internal control parameters. The following weightings and parameter settings were found to usually give rise to feasible solutions.

GA Control Parameters	Objective Function Weightings
Population size : 40	Error : 1.0
Crossover rate : 0.85	Mobility : 1.0
Mutation rate : 0.03	Swept Area : 0.75
	Harmonic content : 0.5

Results will now be presented for the method using these parameter settings on two different problems. The method used is slightly different from that used in section 4.9. In these results, the GA was not run for a fixed number of generations, but termination criteria were introduced so that the search finished after finding the first feasible solution. This enables the speed of the search method to be realised, though of course leaving the search to continue may, and often will, locate even better solutions. The effect of this may be seen by comparing the results found in section

4.9.1.4 with those given in section 4.10.1, as this the method is being run on the same problem in each section. The termination condition used consists of a logical statement that the error between the desired and actual coupler curves must be below a given threshold and that the link length ratios within the mechanism must be such that no link is more than five times longer than the CV input crank.

4.10.1 Experiment 1

The desired curve used in this experiment is the same as was used in evaluating the different objective function in section 4.9. The graph in Figure 4.20 shows the convergence of five solution runs for this problem.

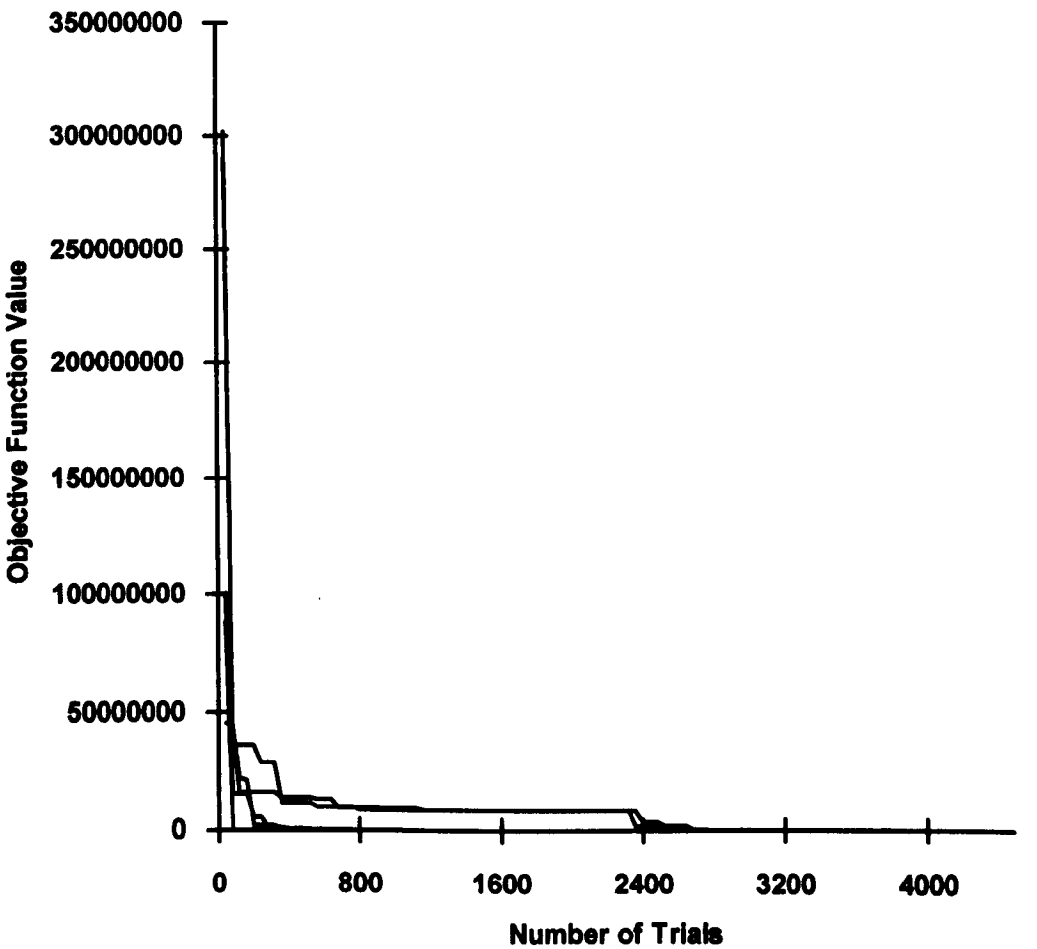

Figure 4.20 : Convergence

All of these solution runs exhibit the typical characteristics of GA based searches, that is, a very fast initial reduction in objective function value, followed by a slow but progressive reduction as the search continues. The table in Figure 4.21 compares the final solutions for each of these sample runs.

This comparison highlights many important points concerning the nature of GA based search and also the complexity of the objective function. The first point to note is that whilst the method has been used on the same problem, the solution obtained in each run is different. This is due to the probabilistic nature of the GA. The second point to note is that the solutions obtained, particularly in the first and third tests, have similar fitness function values but quite different parameter sets. This is often encountered in multi-objective search and is known as pareto-optimality. By using a Genetic Algorithm, problems associated with pareto-optimality are often aggravated by the parallel nature of the search.

	Test 1	**Test 2**	**Test 3**	**Test 4**	**Test 5**
No. Trials	4480	4040	4440	4280	960
CV Co-ords	-5,-5	0,5	0,1	-1,0	2,4
Servo Co-ords	19,-6	19,3	32,-1	32,3	24,-8
p	16	16	16	16	16
q	31	21	24	25	21
r	27	54	39	23	35
s	10	56	28	30	54
Fitness Value	13491.86	30386.93	13557.56	10169.35	28641.97

Figure 4.21 : Comparison of Solutions

This table shows the value of the scaled fitness function used in the GA for selecting between solutions. This scaled fitness differs from the actual objective function value due to the fitness scaling embedded into the GA. However, in this case, the values for fitness function provide a direct maping to dynamic performance. The best solution obtained in these experiments was found in the fourth test after 107

generations (4280 trials). Figure 4.22 shows the servo motor requirement for this mechanism.

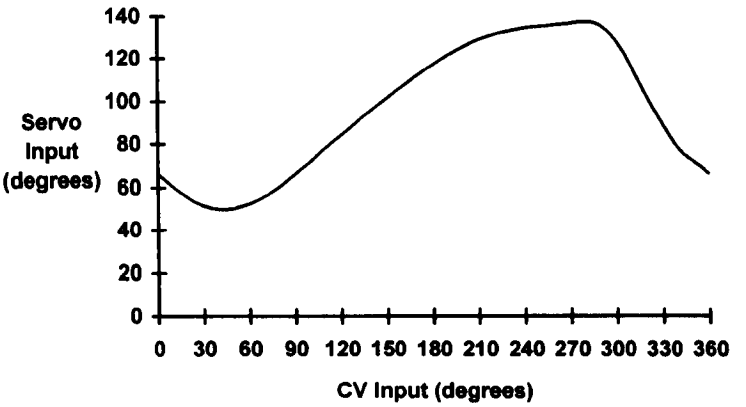

Figure 4.22 : Servo Motor Input Requirement

The end effector motion is shown in Figure 4.23. The continuous line shows the path generated by the mechanism, whilst the points are those used to define the motion.

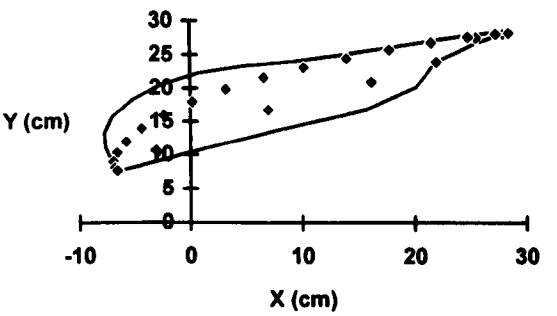

Figure 4.23 : End Effector Motion

Figure 4.24 shows the torque requirement for both motors. For this mechanism, the torque requirement of the servo motor is much lower than that of the CV motor.

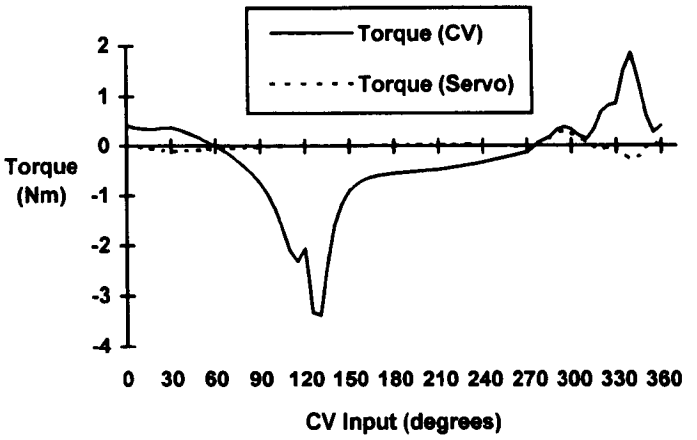

Figure 4.24 : Torque Requirements

The torque requirements are expressed numerically in the table shown in Figure 4.25.

	CV Motor	Servo Motor
Maximum Torque	1.8568	0.2973
Minimum Torque	-3.3906	-0.2782
RMS Torque	1.7160	0.1520

Figure 4.25 : Torque Requirements

This mechanism does not exhibit as desirable torque characteristics as that located in section 4.9.1.4, and indeed suffers from a higher position error around the machine cycle which can be seen by comparing Figure 4.23 with Figure 4.17. Again it is important to state that this is due to the probabilistic nature of GA search. However, one other factor must be taken into account. By introducing termination criteria, the search is finished as soon as a feasible solution is located. This feasible solution is not necessarily the best solution. By continuing to run the GA for further generations then better solutions may be found.

4.10.2 Experiment 2

For the second experiment it was decided to define a complex output curve which contains a double point. This often leads to a lack of search resolution. In addition to this, a complex curve should require a complex servo motor input profile. The purpose of this is to test how well the method works on less than simple problems. The graph in Figure 4.26 shows the convergence of five solution runs for this problem. All of these solution runs exhibit the typical characteristics of GA based searches, that is, a very fast initial reduction in objective function value, followed by a slow but definite reduction as the search continues. One of the trial runs was terminated before a feasible solution had been found as it appeared to have become trapped within a local optima.

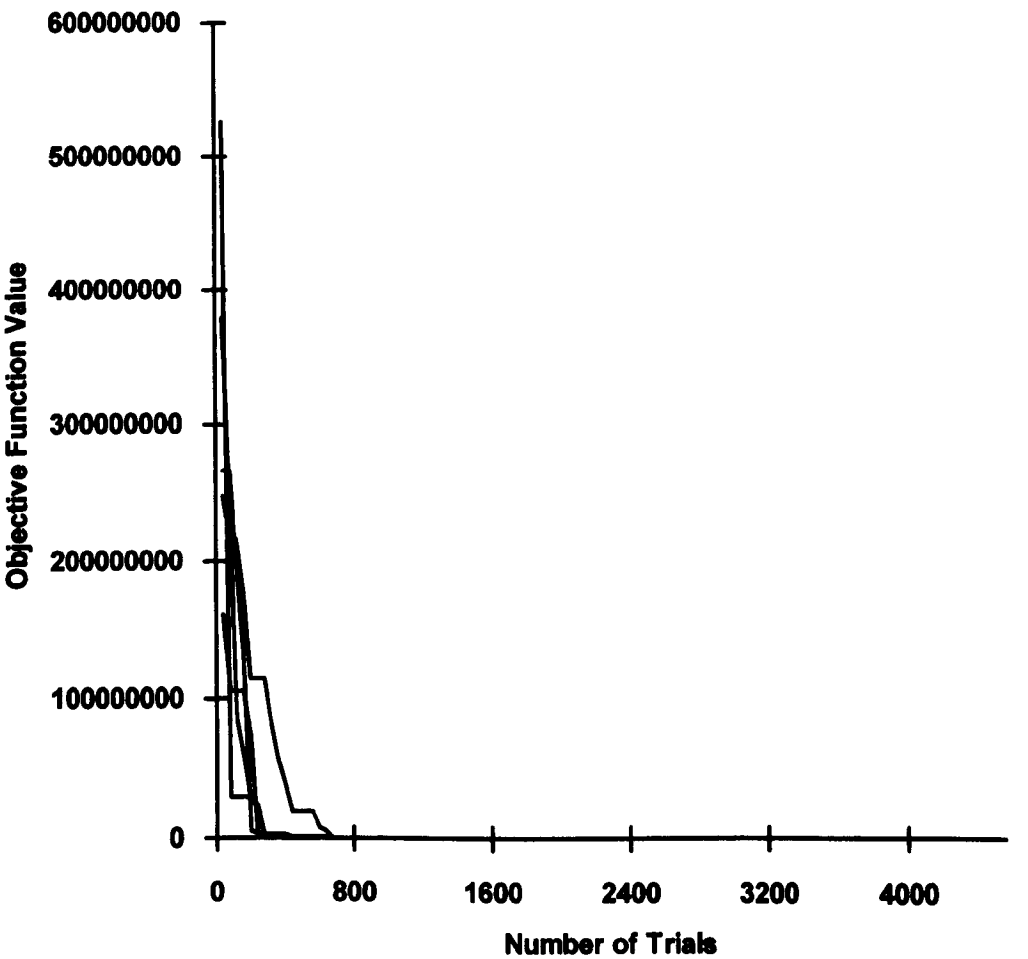

Figure 4.26 : Convergence

The table in Figure 4.27 compares the final solutions for each of these sample runs.

	Test 1	Test 2	Test 3	Test 4	Test 5
No. Trials	680	1840	2480	4560	4520
CV Co-ords	3,-3	8,9	4,-1	5,-1	5,-3
Servo Co-ords	32,13	40,10	27,10	36,10	26,10
p	19	18	19	19	19
q	53	41	51	51	53
r	63	45	71	61	53
s	56	21	74	59	48
Fitness Value	3593.17	4586.98	3405.44	2638.66	2217.74

Figure 4.27 : Comparison of Solutions

The results presented here also suffer from pareto-optimality as well as a lack of objective function resolution. That is, the scaled fitness function values do not directly correspond to dynamic performance. This may be due to the fitness scaling routin embedded into the GA. To illustrate this, torque requirements have been calculated for the best and the worst solutions from these results.

4.10.2.1 Analysis of Test 5

The best solution obtained in these experiments was found in the fifth test after 113 generations. Figure 4.28 shows the servo motor requirement for this mechanism.

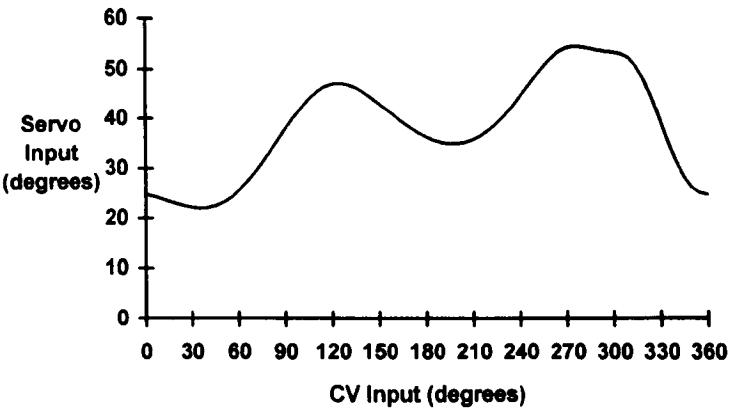

Figure 4.28 : Servo Motor Input Requirement

The end effector motion is shown in Figure 4.29. The continuous line shows the path generated by the mechanism, whilst the points are those used to define the motion.

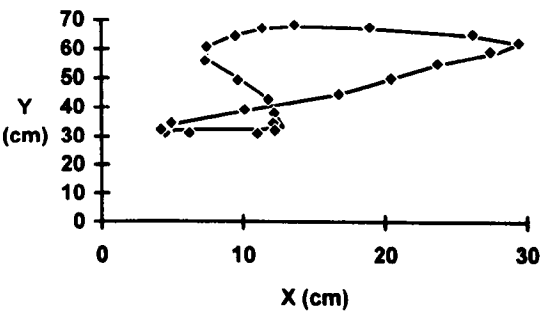

Figure 4.29 : End Effector Motion

Figure 4.30 shows the torque requirement for both the CV and servo motors. For this mechanism, the torque requirement of the servo motor is much lower than that required by the CV motor.

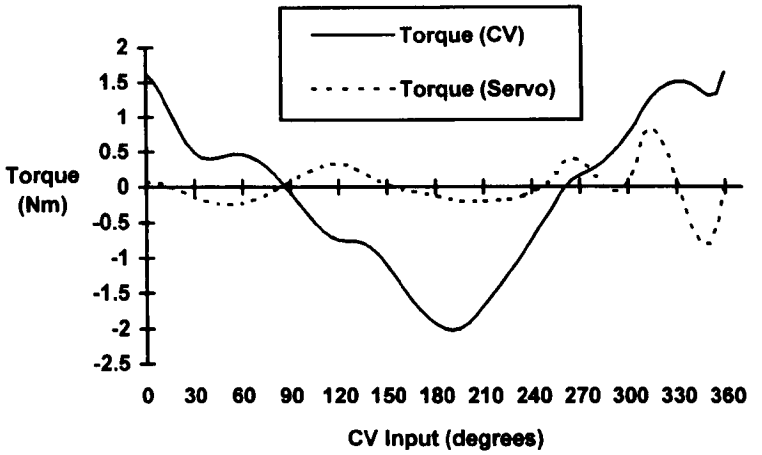

Figure 4.30 : Torque Requirements

The torque requirements are expressed numerically in the table shown in Figure 4.30.

	CV Motor	Servo Motor
Maximum Torque	1.6351	0.8369
Minimum Torque	-2.0237	-0.8063
RMS Torque	1.9168	0.5077

Figure 4.31 : Torque Requirements

4.10.2.2 Analysis of Test 2

The worst solution obtained in these experiments was found in the second test after only 46 generations. Figure 4.32 shows the servo motor requirement for this mechanism.

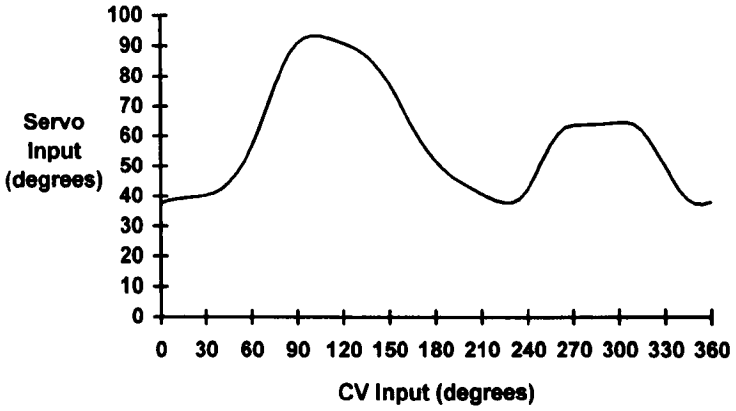

Figure 4.32 : Servo Motor Input Requirement

The end effector motion is shown in Figure 4.33. The continuous line shows the path generated by the mechanism, whilst the points are those used to define the motion.

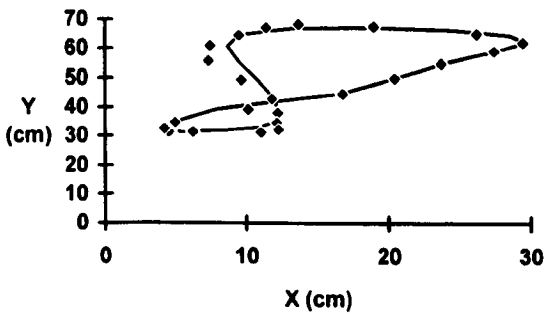

Figure 4.33 : End Effector Motion

Figure 4.34 shows the torque requirement for both the CV and servo motors. For this mechanism, the torque requirement of the servo motor is much lower than that required by the CV motor.

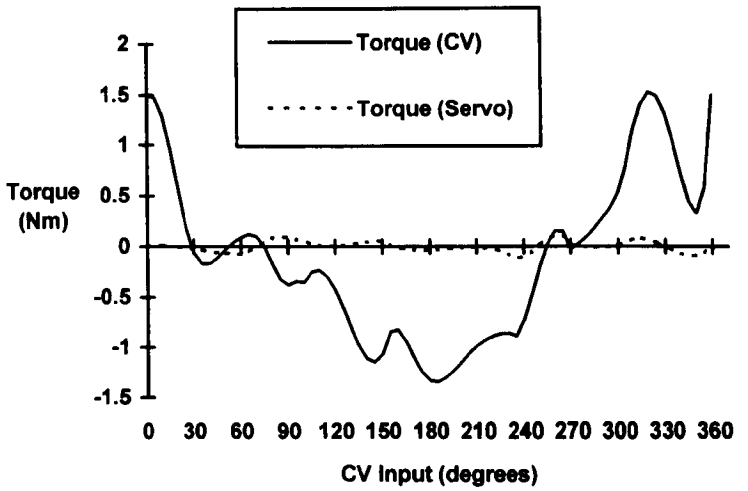

Figure 4.34 : Torque Requirements

The torque requirements are expressed numerically in the table shown in Figure 4.34.

	CV Motor	Servo Motor
Maximum Torque	1.5242	0.1120
Minimum Torque	-1.3431	-0.1163
RMS Torque	1.4290	0.0893

Figure 4.35 : Torque Requirements

Comparing these results with those in Figure 4.31 suggest that this mechanism, whilst having a higher objective function value exhibits more desirable torque characteristics. The implication of this is that the objective lacks resolution between different design objectives. The table in Figure 4.36 compares the values for the design objectives.

	Error	Harmonic Content	Swept Area
Test 2	14.236	10410531.72	28344.59
Test 5	2.967	1726988.74	40840.66

Figure 4.36 : Design Objective Values

It is important to state that the objective function value calculated from these values cannot be compared directly to the fitness function values given in Figure 4.27. The fitness of any given solution is dependant on both th objective function value of that individual solution and the objective function solutions of all the other solutions in the current population.

By considering these values, it can be seen that the solution in test 5 is trading off a high swept area value for a low error. In contrast the solution in test 2 is trading off a high harmonic content for a low swept area value. The implication is that of the design objectives used in this study, the swept area term is the most important in producing low servo motor torque requirement for complex motions. This conclusion is in conflict with the results presented in section 4.9, where a combination of harmonic content and swept area produced the best results. However, this result may have been due to the effects of reducing the magnitude of the fundamental harmonic on the magnitude of the actual displacements.

Some discrepancies also arise due to the fitness scaling routine embedded in the GA. Eliminating this routine may lead to better definition between objective funtion components, but is also likely to lead to premature convergence.

4.11 Summary

This Chapter has developed a design methodology based on the analysis of the five bar mechanism which has produced results that exhibit good dynamic characteristics without the necessity of computationally expensive dynamic objective functions. The results are promising, but more work is required to investigate the conflict that occurs between different criteria within a multi-criteria objective function. It will then be possible to propose methods which deal effectively with complex motions. In addition, it may be neccesary to remove the fitness scaling routine from the GA in order to obtain a more consistent mapping between fitness function value and true

dynamic performance. The performance of the GA may then have to be enhanced bu utilising other methods such as sharing.

Chapter Five

Experimental Verification

5.1 Introduction

In order to illustrate that the method outlined in Chapter Four produces feasible mechanisms, a certain amount of experimental work has been carried out. By developing a prototype machine system, real life issues can be addressed which are not otherwise immediately apparent. In this Chapter, some results are presented for a practical machine based around one of the linkages developed in Chapter Four.

5.2 Local Search Strategy and Theoretical Results

The mechanisms located in section 4.1.9.4 and in section 4.10.1 were used as seed mechanisms in a local search strategy based on gradient methods. In general, a GA will quickly locate solutions in the region near to the globally optimum solution. However, even if running under an elitist selection strategy, GAs often lack local search power. In this work, a novel double pass synthesis strategy has been used. By combining both GA based search and traditional techniques, a true globally optimum solution is found without excessive numerical calculations.

The local search method used in this study was an algorithm based on a steepest descent method. These methods are somewhat more computationally expensive than algorithms such as Powell's method. However, steepest descent algorithms are logically more simple. As the aim was not to compare methods, but to simply search the region around the solutions located by the GA for slight improvements then this method can be deemed to be suitable.

The mechanism located after the search was carried out has the following dimensions;

p = 15
q = 25
r = 30
s = 15

The ground point co-ordinates are fixed at;

CV = (0,0)
Servo = (25,0)

This compares to the seed mechanism, which had the following link lengths and motor location co-ordinates.

p = 16
q = 24
r = 30
s = 16

The ground point co-ordinates are fixed at;

CV = (0,1)
Servo = (27,-2)

By comparing the two sets of parameters, it can be seen that the GA has located a solution near the optimal region and that only small improvements have been found by the local search engine.

The servo motor input motion displacement profile of the test mechanism is shown in Figure 5.1.

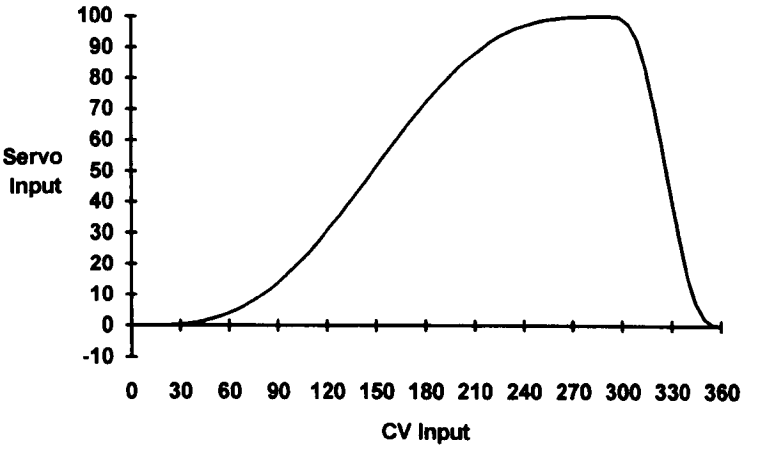

Figure 5.1 : Servo Motor Displacement

The velocity and acceleration profiles are shown in Figure 5.2 and Figure 5.3 respectively.

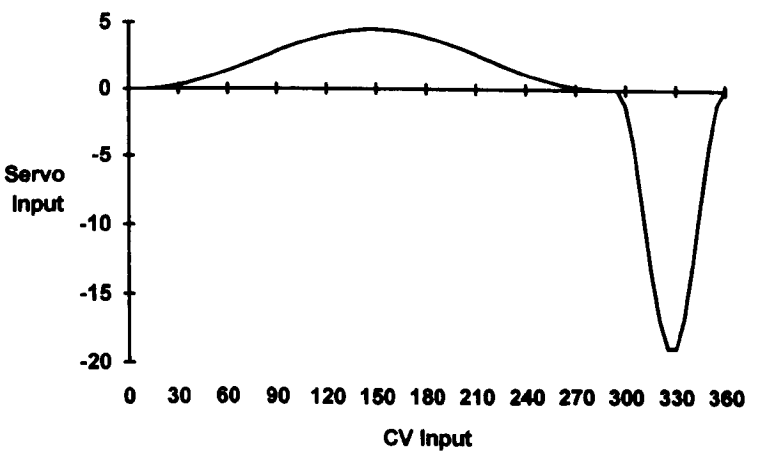

Figure 5.2 : Servo Motor Velocity

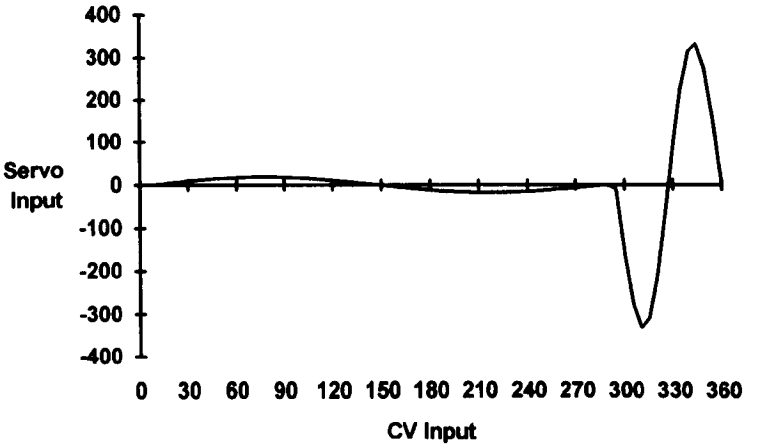

Figure 5.3 : Servo Motor Acceleration

The motion of the end effector is shown in Figure 5.4. By examining this motion, it can be seen that the use of a local search engine has reduced the error between the desired curve and the actual curve significantly. This is due to the re-weighting of the objective function criteria to make the error the most significant criterion.

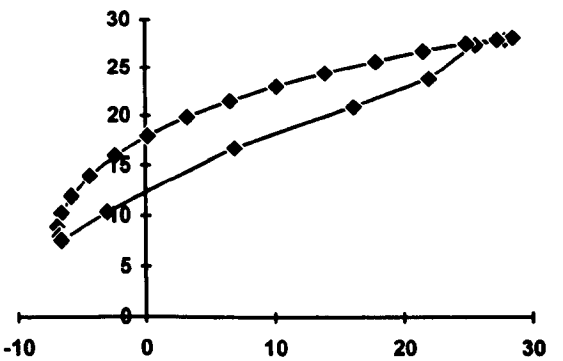

Figure 5.4 : End Effector Motion

The torque requirements of the motors are shown in the graph in Figure 5.5. There is a slight difference in torque requirement over the results presented in Chapter Four.

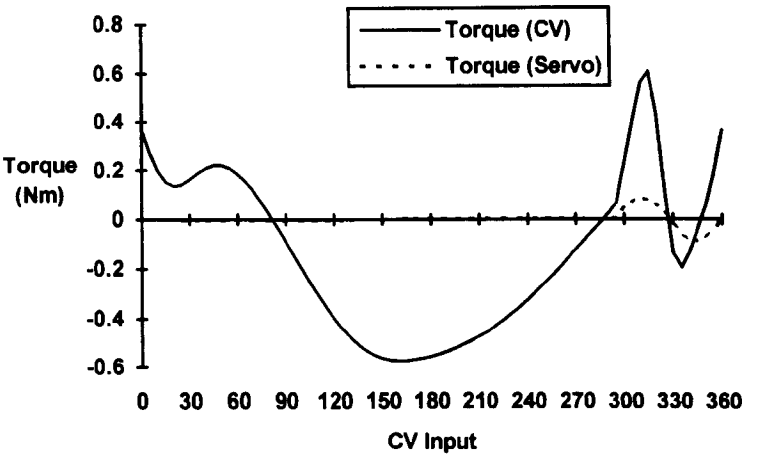

Figure 5.5 : Torque Requirements

The minimum, maximum and RMS values for the torque requirement of each motor are given in the table in Figure 5.6.

	T_{min}	T_{max}	T_{RMS}
CV Motor	-0.5739	0.6107	0.6043
Servo Motor	-0.0859	0.0862	0.0461

Figure 5.6 : Motor Torque Values

5.3 Experimental Apparatus

The experimental apparatus consists of two servo motors fixed in position. These two motors are joined together by a five bar linkage of dimensions given in section 5.2. One of these motors is forced to act as a CV motor whilst the other is programmed to follow a given motion profile. The output of the end effector is traced using a camera. An LED is located on the end effector and a long exposure time used to ensures that the trajectory is recorded on film.

The sketch in Figure 5.7 shows the arrangement of the motors, housing, misalignment coupling and bearings.

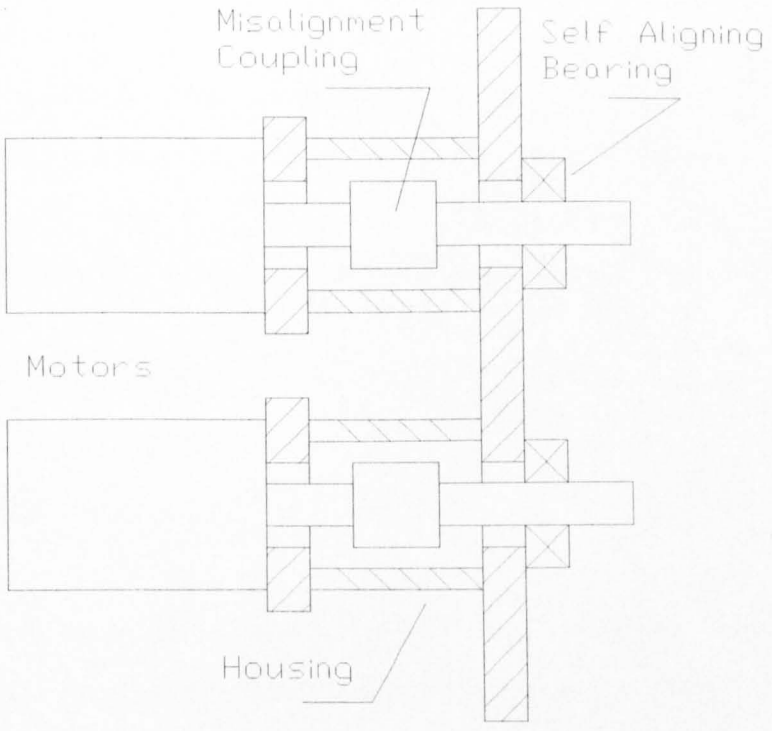

Figure 5.7 : Schematic Arrangement

The linkage mechanism is attached to the output shafts. A sketch of the linkage is given in Figure 5.8.

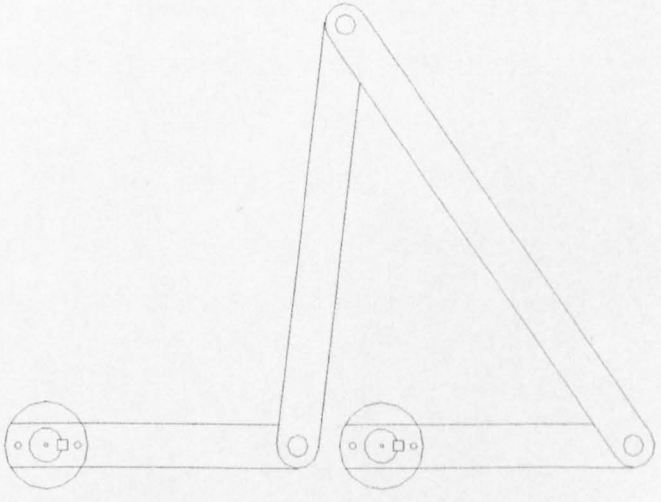

Figure 5.8 : Five Bar Linkage

An in depth description of each of the components in the system will now be given.

5.3.1 Servodrives

Essentially, a servodrive consists of a motor with low rotor inertia and an amplifier which is used to modulate the potential difference across the motor coils so as to produce a variable speed output. A controller is used to interpret feedback information and provide command signals to modulate the potential difference as required.

5.3.1.1 Servo Motors

The motors used in the experimental apparatus are 3 phase AC servo motors with electronic commutation and resolver feedback. The motors are brushless and require a 3 phase power supply.

By varying the voltage across the motor coils, the servo amplifiers induce a torque in the rotor of the motor by means of electromagnetic induction. The level of torque produced is a characteristic of the motor that is termed mechanical torque constant. In producing this torque the motor experiences an opposing voltage across its coils that is proportional to the rate of the rotor movement. The motor therefore has another important characteristic known as the back emf constant.

Other properties of the motor such as armature resistance, inductance and inertia all contribute to the response that the motor is capable of producing under excitation. External constraints such as friction and load inertia also effect the response.

5.3.1.2 Choosing Motors

The choice of a motor for a particular application depends on several factors. Typically, these include the following;

- Maximum torque required
- Continuous torque required
- Maximum motor shaft speed
- Maximum acceleration rate

The torque is the turning effort required from the motor in order to accelerate the mechanical load or system at the desired rate. In order to calculate the torque required from the system, it is necessary to find out the following;

- The reflected total inertia of the load at the motor shaft
- The reflected total friction of the load
- The maximum motor inertia and friction
- The maximum acceleration rate of the motor
- Any gear or pulley ratios in the system

For example, consider a motor driving a load via a belt and pulley. The total torque requirement of the motor is given by eqn. 5.1.

$$T = \left[\left(\frac{D_1}{D_2}\right)^2 I_L + I_M\right]\frac{d^2\theta}{dt^2} + \frac{D_1}{D_2}F_L + F_M \qquad ...(5.1)$$

where;

T = Total motor torque required

D_1 = Diameter of motor pulley

D_2 = Diameter of load pulley

I_L = Inertia of load

I_M = Inertia of motor

$\frac{d^2\theta}{dt^2}$ = Acceleration at motor shaft

F_L = Friction torque of load

F_M = Friction of motor

In most cases, the motor inertia and friction can be assumed constant, unless the system has a changing load. The required velocity profile of the motor should be determined to allow the acceleration to be calculated. The maximum acceleration is found where the gradient of the velocity profile is maximum.

This can be repeated for all points so as to find the RMS continuous torque requirement. Servo motors are generally specified with both a peak torque and continuous torque rating and they should be chosen that the torque requirement is within the range of the motor.

If too large a motor is selected then the motor inertia is higher than for a smaller motor. This affects the maximum acceleration that the motor can produce. It is not always the largest or most powerful motor that accelerates the load at the quickest rate. This gives rise to the concept of inertia matching. Whenever possible, the motor inertia should be similar to the load inertia. The maximum power transfer from the motor to the load is obtained if the motor inertia and the reflected load inertia are similar.

In section 5.2, the torque requirements for a typical hybrid five bar mechanism were calculated using the equations derived in Chapter Four. The motors used in the experimental apparatus are both 6.8NM continuous stalling torque motors. This is somewhat larger than necessary for the experiments outlined in this Chapter as the experiments use a rather slow CV input speed of 60 rpm and there is no external load inertia other than the linkage itself. The choice was made on the basis that the apparatus may in future be used for testing different linkages and motion profiles at different speeds, or with varying masses located at the point of the end effector. This is particularly relevant as the five bar mechanism is suited to tasks such as pick and place. The greater motor capacity allows such flexibility at a slight compromise of current performance.

5.3.1.3 Servo Amplifiers

The servo amplifiers used in the experimental apparatus are Q-Drives, developed by Quin Systems. The use of Q-Drives entails a seamless compatibility with the MiniPTS controller. Essentially, the Q-Drive utilises a 16 bit Digital Signal Processor to emulate an incremental encoder from the resolver feedback information from the motor. An incremental encoder allows not only speed information to be ascertained, but also direction information that is essential to the operation of non-uniform motion generating servodrives.

Information from the resolver is therefore transferred into an output of an incremental encoder. This information is passed through a serial link to the MiniPTS controller where it is interpreted and command signals returned to the Q-Drive. These command signals are determined from the motion requirements downloaded to the controller and are returned to the Q-Drive. The Q-Drive then modulates the servo motor voltages as required.

The use of DSP technology allows faster conversion times than more conventional ADC and DAC technology. This enables the system to operate at high speeds without loss of resolution.

In addition to modulating the servo voltages, the Q-Drive has setup parameters which are initialised for each system. These include limits on the maximum current, the nominal current and the resolver resolution. These parameters effect the performance of the system and must be correctly setup. Details of this process are available in the Q-Drive Installation Manual.

5.3.1.4 Controller

The controller used in the experimental apparatus is a MiniPTS system, also designed by Quin Systems. The system comprises hardware and software to control

up to four servo motors in conjunction with suitable high power drive systems. The hardware is highly modular, allowing the system to be easily expanded or upgraded. The software provides full control over all aspects of the system but has a simple high-level user interface. The advantages of the MiniPTS is its ease of installation and use. All communications between the software/PC and the MiniPTS are via a standard RS232 serial communications link. The front end software allows motion profiles to be easily defined using a number of methods and provides means to slave different axes of control together. This is perfect for the hybrid concept, where the servo motor motion profile must be executed in relation to the CV input position.

5.3.1.5 Control Algorithm

The MiniPTS control system operates by sampling the position of the motor at regular intervals and calculating a motor demand signal according to the control algorithm. The algorithm used is of the following form;

$$V_{out} = K_P e_i + K_I \sum e_i + K_D(e_i - e_{i-1}) - K_V(p_i - p_{i-1}) + K_F(d_i - d_{i-1}) \quad ...(5.2)$$

where;

K_P = Proportional gain

K_I = Integral gain

K_D = Differential gain

K_V = Velocity feedback gain

K_F = Velocity feedforward gain

e_i = position error

d_i = demand position

p_i - measured position

The actual scaling between error and output voltage, for proportional gain only, is shown below. The other gain terms have similar scaling factors.

$$V_{out} = e_i \times \frac{K_P}{256} \times \frac{10}{2048}$$

...(5.3)

Previous work by **Bradshaw *et al*** [4,120] has shown that the use of a PID controller with a velocity feedforward component based on an inverse model provides the best response for hybrid systems to date. Whilst this type of controller is not as robust as H∞ controllers, the predictable nature of the system implies that the feedforward component will produce low steady state errors and system stability.

5.3.1.6 Monitoring System Performance

The MiniPTS controller comes complete with a full range of system software. Within this software there are a variety of ways in which the performance of the system can be accurately monitored.

The first of these methods is the continuous display mode offered through the serial communications link. This mode is initiated through the terminal and the desired parameters are displayed on the screen numerically.

The most useful method, however, uses the PTS Scope software. Essentially, this is a software oscilloscope which allows the desired parameters to be displayed graphically as the machine runs through its cycle. In addition to this, there is an option which allows the scope to act in a data logging mode so that data can be stored and displayed at a later date.

5.4 Controller Tuning

This section presents two methods for tuning single motors, as well as a simple algorithm for the tuning of a hybrid machine. All of these methods assume that the transfer function of the system is unknown. If the transfer function is known, then there are a number of analytical methods that can be used to determine the gains of the controller so that the system is both stable and exhibits a satisfactory response.

5.4.1 Zeigler-Nichols Tuning Rules

A simple set of tuning rules were initially developed by **Zeigler & Nichols** [121] and have since been further refined so that they can be applied to modern control systems. The method essentially revolves around observing the response of a system to a step input. The proportional gain is slowly increased until the system exhibits a response that is a non-decaying oscillation. This value for the proportional gain is known as the critical value, K_{PC}. The period of the oscillation is T_P.

Depending upon the type of controller used, the proportional gain (K_P) and time constants (T_I & T_D) are calculated as shown in the table in Figure 5.7.

Controller Type	K	T_I	T_D
P	0.5 K_{PC}	∞	0
PI	0.45 K_{PC}	$T_P/1.2$	0
PID	0.6 K_{PC}	$T_P/2$	T_P

Figure 5.9 : Zeigler-Nichols Tuning Rules

Recent work by **De Paor** [122] has shown that these tuning rules are not as empirical as they first seem, and their validity can be assessed by using frequency domain techniques.

5.4.2 Oscilloscope Methods

Osilloscope methods are a common approach to the practical tuning of many types of controller. In general, the rely on forcing the controlled system into a given motion and observing the error between the actual and desired outputs as the tuning gains are altered. A method for use with the Quin controller is described in Appendix B and provides the basis for the hybrid machine tuning method described in the next section.

5.4.3 Hybrid Machine Tuning

The tuning of a single motor is a relatively simple process when compared to the tuning of a multi-axis machine, where the inputs are mechanically coupled. In this case, the output from one motor effects the position of other motors in the machine and incorrect tuning can lead to ineffective machines with poor stability. Tuning a multi-axis machine, such as the hybrid five bar developed in this study, is often done during the normal running operation. This is because the system does not often respond well to step inputs and can often be damaged by severe oscillations.

The flowchart in Figure 5.10 shows the tuning process used for the machine developed in this study and is based around a two axis machine, where axis 1 is acting as a CV motor and axis 2 carries out a given motion profile slaved to the position of axis 1.

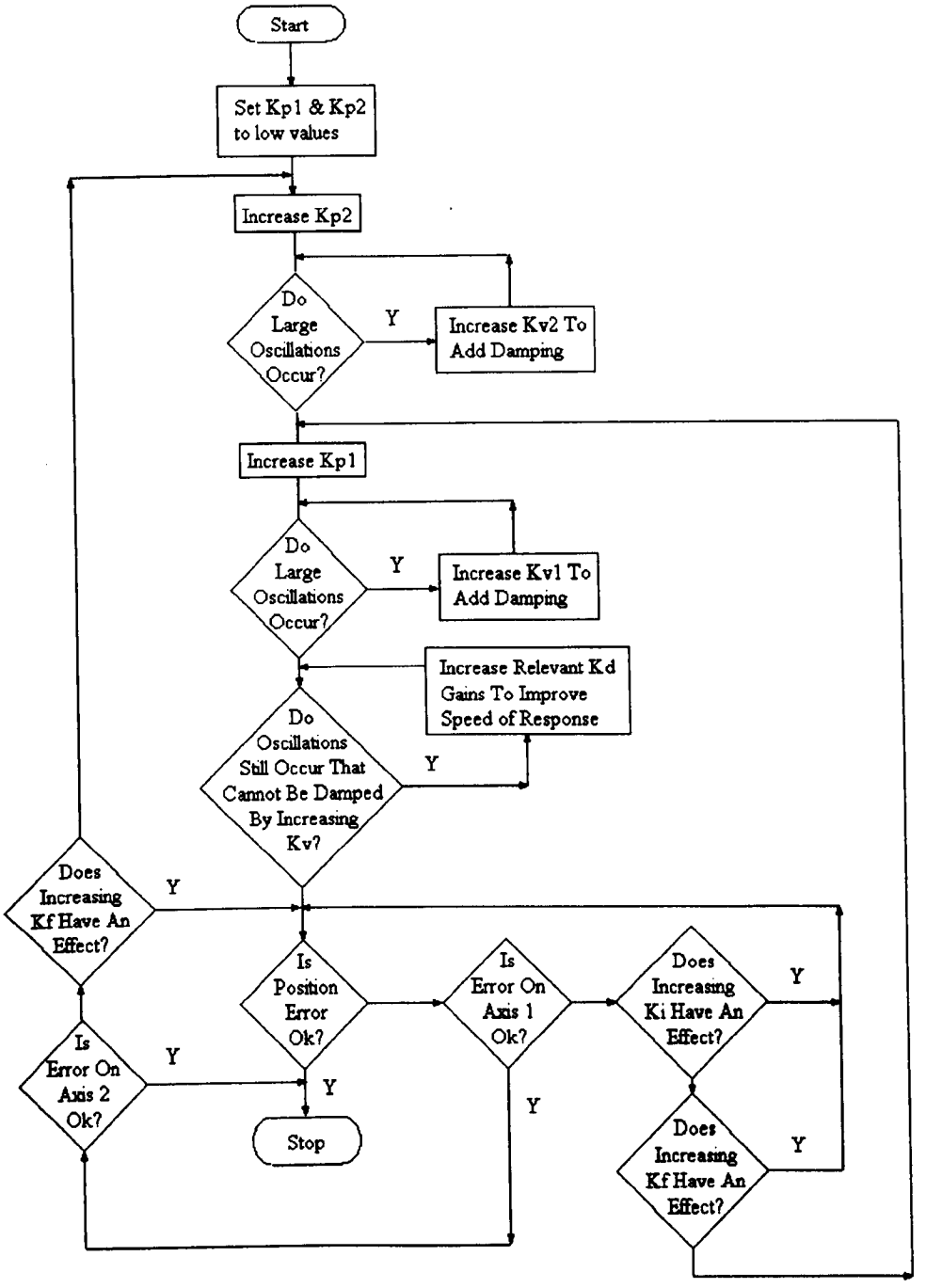

Figure 5.10 : Tuning Flowchart

Essentially, this method is a refinement of the method described in section 5.4.2, where the normal running operation of the machine is used. During the practical tuning of the machine, it is important to realise the logical deficiencies of this flowchart. For example, if the error on the CV axis is deemed acceptable, but the

error on the servo axis is not, and the chart leads back to the start it is obviously unnecessary to increase the proportional gain of the CV axis unless significant errors are reintroduced by changing the tuning parameters of the servo axis.

It is also important to realise that the tuning process must be repeated whenever the operating conditions are changed. However, in general, a set of control parameters tuned for a high speed will generally produce good performance at lower speeds.

For the results presented in section 5.5, the tuning process was halted as soon as the end effector motion appeared to approximate the desired motion and very little oscillation was observed on either axis at an input speed of 60 rpm. By tightening the termination criteria of the tuning process, improved performance could be produced.

5.4.4 Tuning Summary

Tuning any control system is not an easy process, and whilst the above procedure can be used as a guideline it is important to realise the effects of each of the gain constants so that a good tuning setup is achieved.

Proportional gain is used to decrease steady state error, as the controller ouput signal is directly proportional to the error. Unfortunately, high proportional gains lead to oscillatory motions and unstable systems. When Integral control is used, the system integrates the position error by adding the current error to a running total. Integral gain is useful to remove a constant position error, due to a steady load or friction, and so have a zero steady state error without overly high proportional gains. Derivative gain uses the differential of the position error which represents the velocity error of the system. This is useful where the position error is changing rapidly, as increasing the derivative gain improves the speed of response of the system. Derivative control uses the rate of change of error, whilst Velocity Feedback uses the rate of change of absolute position. Adding Velocity Feedback is similar to the effect of an externally connected tachogenerator and results in increased damping in the system. Finally,

Velocity Feedforward uses the demand velocity as opposed to the measured velocity and is particularly useful when following a defined profile. If the system is using Proportional gain only, then there will be a steady positioning error, known as velocity lag, when running at constant velocity. The feedforward gain has the effect of reducing the velocity lag by adding a component dependant on the demand velocity into the controller output.

5.5 Experimental Results

This section presents results for the purpose of evaluating the performance of the hybrid machine. The experiments consist of running the mechanism through a number of cycles and logging position, and velocity data for both the CV and servo motors as well as observing the trajectory of the end effector with the LED and camera combination.

Figure 5.11 shows the actual end effector motion of the real machine in comparison to the precision points used to define the desired motion. This plot was obtained from long exposure photographs, where the trajectory of the end effector was traced by an LED. The trace obtained was copied onto acetate, and then projected and traced onto graph paper so that the precision points could be plotted. The desired start position of the mechanism is shown, as is the direction of rotation.

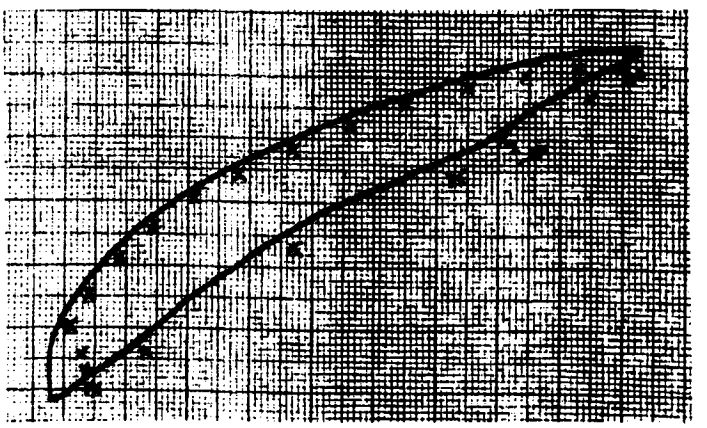

Figure 5.11 : End Effector Motion

In addition to this, the following images were obtained by the use of a video camera. The mechanism was filmed around the cycle, and the images grabbed from the appropriate frames. The purpose of these images is to show the actual end effector position for a given CV motor input angle and so compare this to the desired position. This differs from the information shown in Figure 5.11, where the timing of the input is not shown. Three images are given here which correspond to different CV motor input angles.

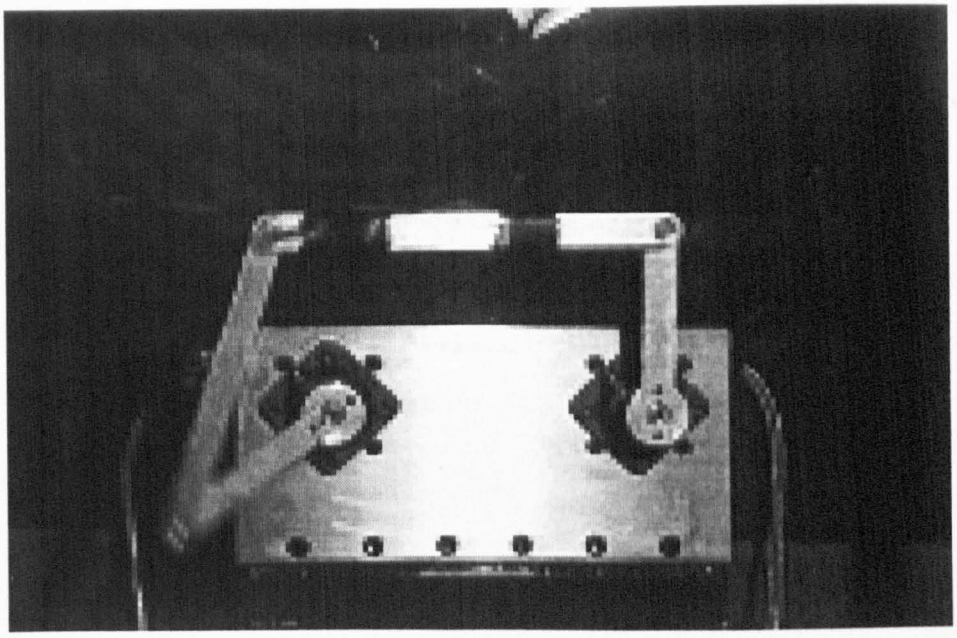

Figure 5.12 : End Effector Position (217°)

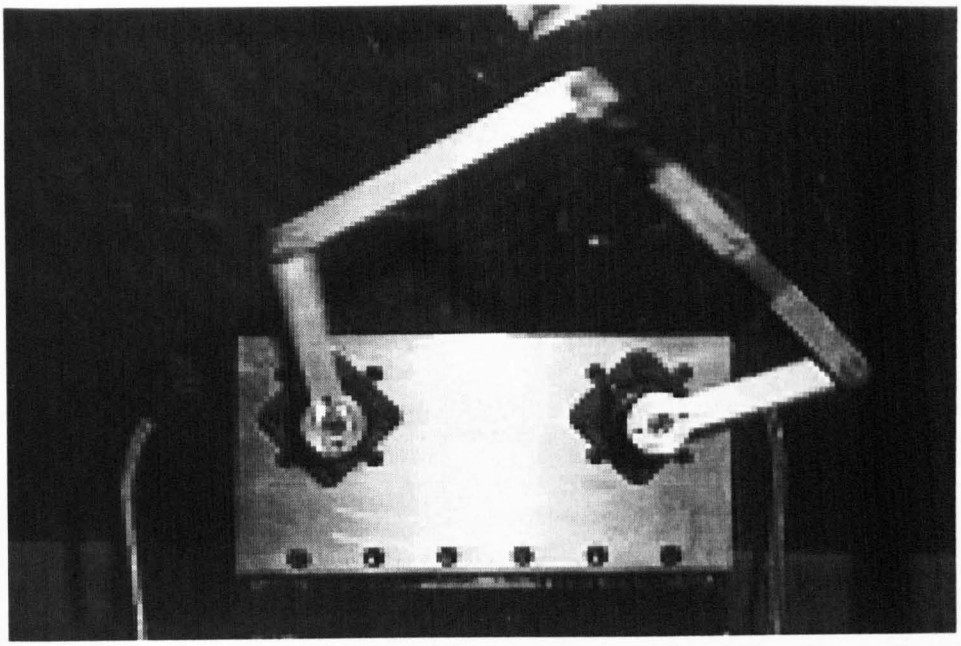

Figure 5.13 : End Effector Position (102°)

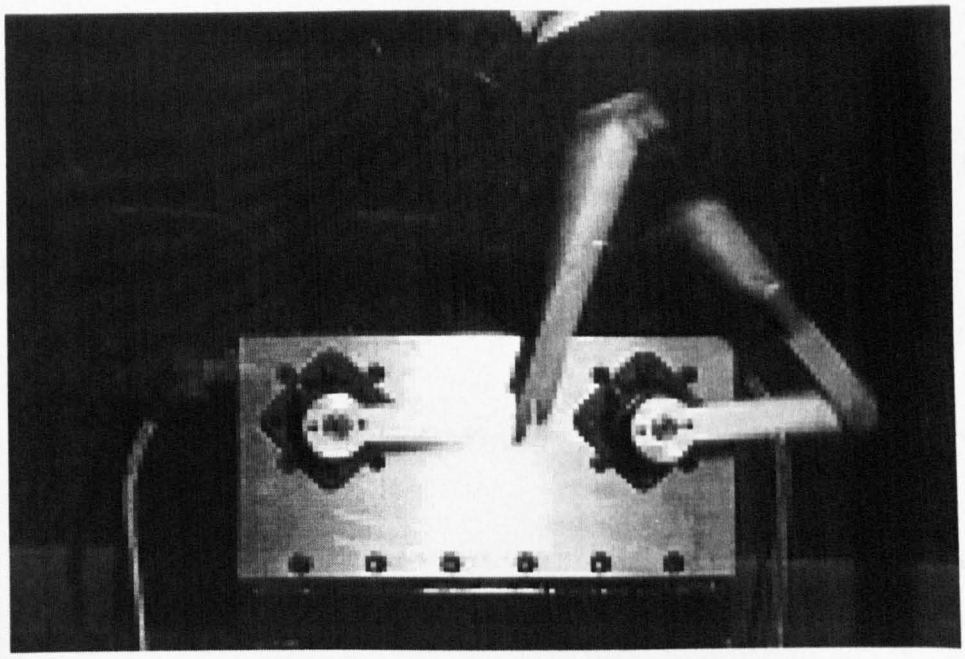

Figure 5.14 : End Effector Position (4°)

The table in Figure 5.15 compares the actual co-ordinates of the end effector with the desired co-ordinates. The co-ordinates are expressed in millimetres relative to the CV motor position.

CV Motor Input	Actual Co-ordinates	Desired Co-ordinates
217°	-52,147	-46,148
102°	194,262	186,270
4°	242,246	230,247

Figure 5.15 : Comparison of End Effector Positions

The graph in Figure 5.16 shows the data logged from the experimental machine for demand position, actual position and position error for the servo axis in units of resolver counts. The servo amplifiers are set to a resolution of 4096 counts per cycle. By examining this data it can be seen that for nearly all of the machine cycle, the actual position lags the demand position. This explains the position of the actual coupler curve with relation to the desired coupler curve observed in Figure 5.11.
The maximum position error occurs at the start of the return period of the motion. The value at this point is approximately -115 resolver counts, or approximately 10°.

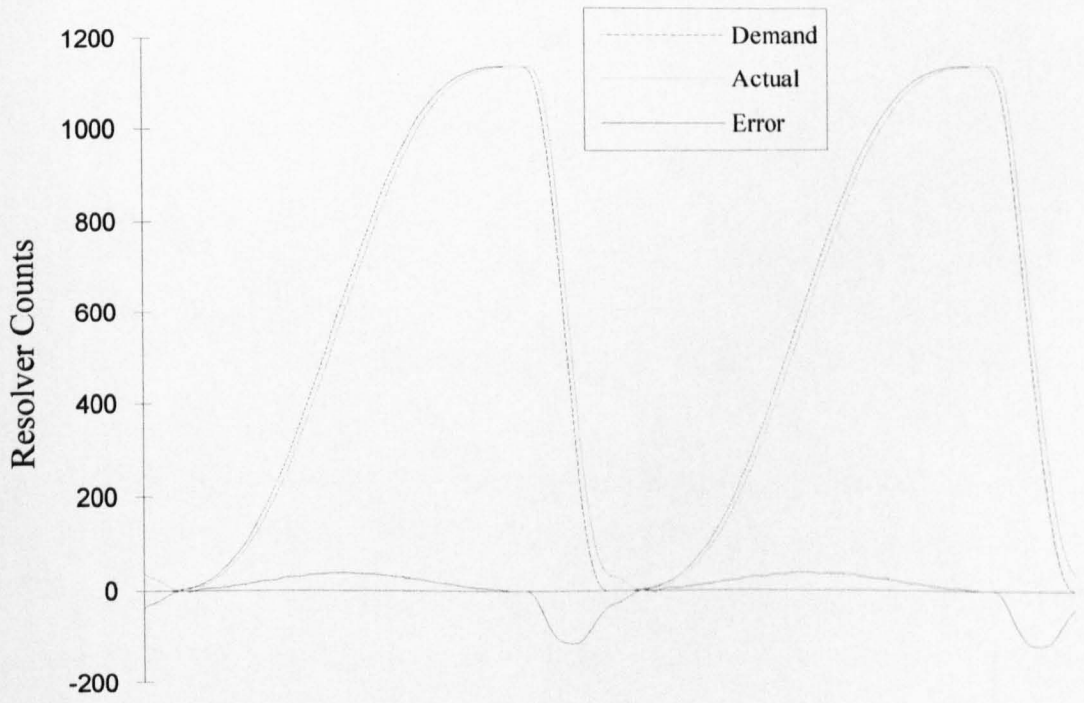

Figure 5.16 : Servo Motor Position Data

The maximum actual position of the servo motor is 1135 resolver counts. This is equivalent to 99.76°. This compares to the theoretical prediction of 99.78°.

The graph in Figure 5.17 shows the demand and actual velocity of the servo axis, in units of resolver counts per second. There is a certain amount of "noise" on the demand signal, which is caused by the controller trying to correct position error. This casues the actual velocity signal to be quite oscillatory in nature.

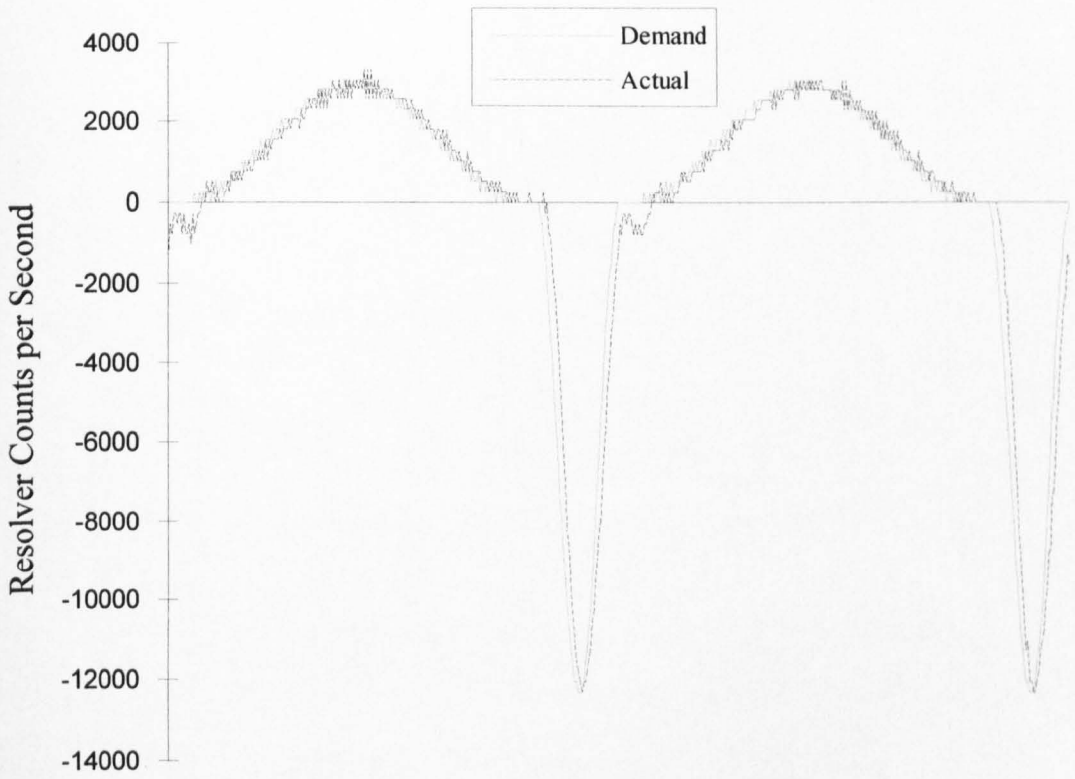

Figure 5.17 : Servo Motor Velocity Data

The maximum actual velocity is 3072 resolver counts per second. This is equivalent to 4.71 radians per second. This compares to the theoretical prediction of 4.48 radians per second. Similarly, the minimum actual velocity is -12288 resolver counts per second. This is equivalent to -18.85 radians per second. This compares to the theoretical prediction of -18.96 radians per second.

Due to the uniform motion on the CV axis, the differences between demand and actual position are less significant. The graph in Figure 5.18 shows the position error only for this axis. The maximum value occurs at the beginning of the cycle and has a value of approximately 19 resolver counts, or 1.7°.

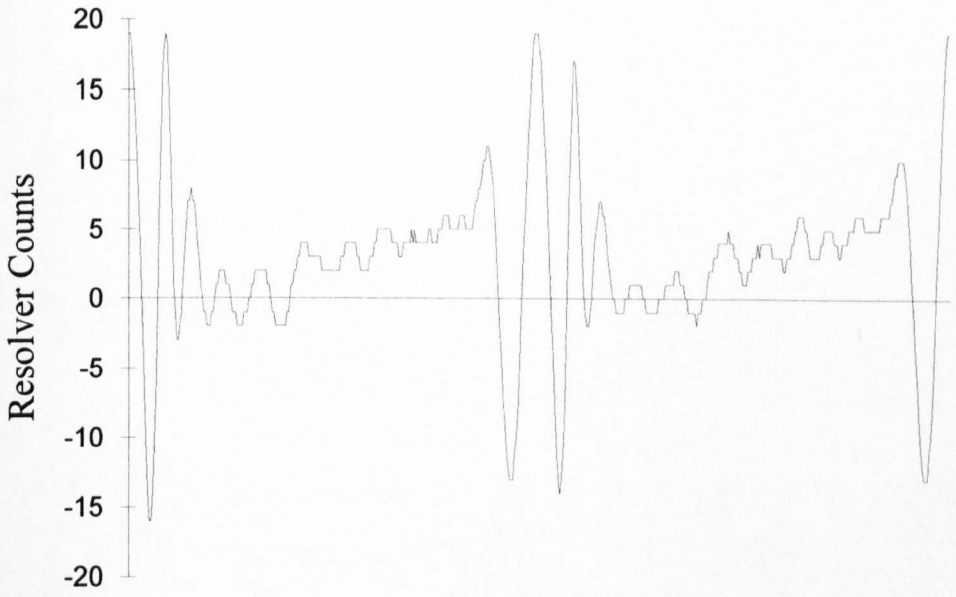

Figure 5.18 : Position Error on CV Axis

The graph in Figure 5.19 shows the constant demand and the actual velocity for the CV axis. Again, the units are in resolver counts per second, where the demand velocity of 4096 cps is equal to 60 rpm.

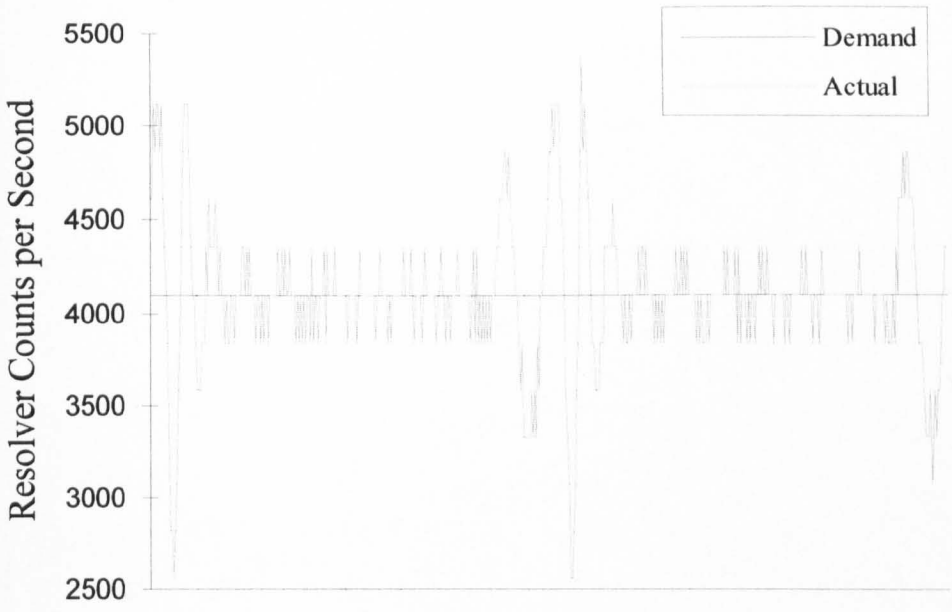

Figure 5.19 : Demand and Actual Velocity for CV

In addition to logging kinematic data concerning the mechanism to validate that the output motions are similar to the desired motions, some simple power measurements were carried out to show the power distribution between CV and servo motor. These power measurements were taken using a simple hand held power meter adjusted for use on a three phase supply by the inclusion of a reference voltage.

Readings of RMS power were taken from a single phase of each motor supply, and averaged for several cycles of the mechanism. A three phase compensation was included which calculated the total power by referencing this phase to the remaining phases. The power drawn by the CV motor is 3.3429 kW, whilst the power drawn by the servo motor is 1.5696 kW. These figures can only be viewed as being approximations to the motor power consumption, as they are actually measurements of the power drawn by the servo amplifier on each axis. However, they do indicate that the CV axis is contributing more to the overall motion than the servo axis. The inclusion of torque transducers into the mechanism would allow for more accurate assessment to be carried out.

5.6 Discussion of Results

The results presented in section 5.5 illustrate that the developed design methodology does produce results that can be turned into practical machines. However, several factors need to be highlighted. The most important of these is that the tuning of the controller is critical to producing the desired motions.

The experimental tuning process 5.4.3 was halted as soon as the end effector of the mechanism appeared to generate the desired output motion, and there was no observable oscillation on either axis as the machine completed one cycle. The process was halted due to the tedious nature of finding optimal gain settings. The search for optimal gain settings is complex due to effect changing one gain term has on possible values for the other gain terms.

There is one example of this which can be used to illustrate this effect. Setting a high proportional gain will often cause excessive oscillation at the end of a move. However, increasing the velocity feedback gain adds damping to the system and so allows the proportional gain to be further increased. If an experimental tuning approach is chosen for future work, then it is recommended that a more rigorous understanding of the effects of changing gains terms is obtained through careful experimentation.

For the sake of this study, where control issues do not hold prime importance, the tuning algorithm developed in section 5.4.3 produces acceptable results even if the termination criteria on position error are not overly rigorous. Continuing to tune the controller on tighter termination criteria should produce better results, though the time taken to tune the controller for excellent performance may be excessive. Considering that the controller must be retuned if the servo motor motion profile is changed, then an experimental tuning approach is not recommended.

However, by examining the graphs in Figure 5.18 and Figure 5.19, it can be seen that the position error is essentially cyclic. This implies that better control could be

achieved by developing models of the hybrid machine. These models could be used to replace the current feedforward component in the controller with an inverse model feedforward component, as used by **Bradshaw** [4,120].

As an alternative to this, it should be possible to determine the transfer function of the mechanism by comparing the response of the machine to a set of standard inputs. This is achieved by plotting an experimental Bode diagram, and hence determining the transfer function. This approach has several merits, including the possibility of de-coupling the mechanism by finding the inverse transfer function. This is achieved by considering the effect on one input when a standard input is applied to the other machine input.

It is important to realise that any improvements to the control algorithms, such as those outlined above, can only be guaranteed to improve the response of the motors. It is entirely different to ensure that the end effector is accurately placed at all stages of the cycle. By examining the data in Figure 5.11 to Figure 5.15 it can be seen that this does not always occur. By using sophisticated image processing equipment, it may be possible to include the actual position of the end effector into the control algorithm governing the operation of the machine. Such a development would involve considerable computing power and would be a completely novel approach to the control of path generating mechanisms.

An alternative to the use of image processing equipment would be to attach wires to the end effector. These cables could then be passed through fixed linear encoders, so as the mechanism completes a cycle, the position of end effector could be calculated by the amount of wire drawn through the encoder. To include this information into the control algorithm would require less computing power than the image processing approach but lacks elegance. In addition, it may not be a practical solution for an industrially viable machine.

5.7 Summary

This Chapter has described the experimental apparatus developed to test that the developed design methodology produces solutions that can be turned into practical machines. Results have been presented which show that the real machine produces an adequate approximation to the desired motion. Errors between the desired and actual motion can be attributed to a set of non-optimal controller gains and the effects of a cross coupled system. The real machine shows a power distribution between CV and servo motors of approximately 2:1. This indicates that the bulk of the motion is being provided by the CV motor, whilst the servo motor is providing the flexibility to ensure that the desired coupler curve is being produced. This proves that the machine is acting as a hybrid device.

Chapter Six

Discussion

6.1 Introduction

The aim of this Chapter is to discuss the results presented in Chapter Four and Chapter Five, and hence assess the merit of the methodology. Essentially, this Chapter consists of two sections. In the first section, the results presented in Chapter Four will be analysed. The aim is to assess the merits of the methodology in terms of a theoretical base. In the second section, the result of Chapter Five will be discussed, so assessing the manner in which theoretical results can be turned into practical machines. Both of these sections contain various recommendations for future work which will be reconsidered in the appropriate section of Chapter Seven.

6.2 Theoretical Results

In order to discuss the results presented in Chapter Four in a rational manner, it is first important to restate the main objectives of this work. The aim was to develop a novel design and synthesis methodology for hybrid machine design. The purpose for this new methodology was to enable a more robust approach to automatic design. As a secondary objective, the methodology was intended to encapsulate novel design objectives which can be used to augment or replace previously used design objectives which were considered to be computationally expensive and inelegant.

A large amount of effort has been put into the development of a suitable Genetic Algorithm for use in the synthesis method. To ensure that the performance of the GA is acceptable, it is necessary to understand how certain implementations introduce bias into the method which results in poor performance. These biases, which were present in early work involving the synthesis of four bar mechanisms, have been

eliminated by the inclusion of penalty functions to replace explicit heuristic constraint filters and the inclusion of a fitness scaling to help differentiate between similarly fit individuals. Other biases, including those which relate to poor random number generation have also been eliminated.

By analysing the results in Chapter Four, it can be seen that the proposed methodology is considerably more robust than previous methods based on conventional search algorithms. In terms of GA based searching, it is also reasonably efficient. When coded into a binary string, each solution representation consists of a 48 bit binary string. This results in a search space of 2^{48}-1 possible solutions. Yet the method is locating feasible solutions after approximately 4000 trials.

In all cases the method located a solution that was feasible in terms of the objective function in use. However, the results presented in section 4.10.2 show that the objective function used to approximate dynamic performance does not correlate directly to true dynamic performance. This may be due to not including the effects of the intermediate links into the function approximation.

However, in many respects, this lack of direct correlation is not a major issue. In both cases analysed the results show a desirable torque distribution between CV and servo motors. However, if a truly dynamic solution is required then the novel methodology could be used to generate a seed mechanism for use in a search using an objective function based on either torque or power calculations. This double pass methodology has considerable merit as it is likely to produce solutions that are dynamically optimal without long computation times.

One other point which needs to be discussed with respect to the lack of correlation between approximate dynamic conditions and truly dynamic conditions is that this lack of correlation only occurs in the second trial case. This second experimental curve was defined expressly so that the servo motor displacement profile was likely to be very complex. This was intended to test parts of the design methodology such as the closure tracking algorithm. With the limited data that is available the following conclusions may be drawn.

In the objective function used, the harmonic content of the servo displacement profile is weighted so that it is more important than the swept area term. When the harmonic content of the servo motor displacement profile is low, this produces solutions that are very good approximations to dynamically optimum solutions and a direct correlation between objective function value and dynamic performance. However, when the desired output curve necessitates a servo motor displacement profile with high harmonic content then this correlation is not as exact, though high quality solutions are still produced.

Conclusions of early work by **Connor *et al*** [37,38] suggested that poor performance on certain coupler curves was due to the fact that the coupler curve defined the objective function surface and this conclusion can be paralleled here. In the second case, when harmonic content must be high, then it is the swept area term which has the most significance on determining the dynamic performance.

Whilst further research must be carried out to verify this proposition, it is possible to suggest a method for producing a general objective function which could be applied to all problems. In this method, the objective function weightings for the swept area and harmonic content terms would be calculated automatically after considering the nature of the desired curve.

The simplest way of achieving this would be to consider the harmonic content of the closed curve, in a similar manner to **McGarva & Mullineux** [8], and utilising this in terms of a transfer function between output curve and servo motor displacement profile. This transfer function is based upon a simple assumption that the higher the harmonic content of the desired coupler curve, then the servo motor displacement profile must have a higher harmonic content in order to produce that curve. The higher the harmonic content of the servo motor displacement curve, then the more the swept area term becomes significant. This simple assumption does not take into account the effect of varying the link lengths of the mechanism but should still provide sufficient approximation to select the objective function weightings.

The development of such transfer function may also provide an insight into other areas in which it could be applied. By examining the servo motor displacement profiles for the first test motion in Chapter Four, it can be seen that these profiles exhibit similar natures despite differing link lengths and motor positions.

Given sufficient information, it may be possible to develop a knowledge based system which utilises the relationship between the desired motion and the family of curves which are required to produce the motion for different link lengths. This could be used to either guide the GA towards high quality solutions, or aid the designer during other stages of the design process.

6.2.1 The Genetic Algorithm

Whilst the most important aspect of the method is the quality of the solutions produced, it is also relevant to discuss the effectiveness of the GA used. In this study, the GA was based around the three main operators of reproduction, crossover and mutation. The GA used an elitist strategy and a secondary inversion operator called to eliminate identical strings from the population. Analysis of the GA performance can be split into two areas. These are the nature of the convergence and the quality of the solution.

In this case, the nature of the convergence was studied briefly by examining the population of solutions for a number of test runs as it changed from generation to generation. It is not necessary to reproduce a full population set for the number of generations for which the solution was run, as a number of generalised conclusions can be drawn from the observations.

In all cases the method did not fall foul of the two main reasons which cause genetic based searches to fail. The first of these is domination of early generations by "super individuals", leading to premature convergence. In GA terms, premature convergence is defined as when the population consists of a significant number of

identical solutions which are not located on the globally optimum peak of the objective function. This avoidance of premature convergence can be attributed to two factors. These are the use of identical string elimination and a simple fitness scaling routine.

The second cause of failure in genetic search is the lack of definition between similarly fit solutions towards the end of a solution run. This has been partially avoided by the use of the fitness scaling routine. However the results presented suffer from pareto-optimality caused by trade off between different design objectives. Pareto-optimality is when solutions have similar objective function values but considerably different parameter sets.

6.2.2 Pareto-Optimality

Analysing the results presented in section 4.10 show that in most cases different solutions have similar objective function values but considerably different parameter sets. This state is known as pareto-optimality and is generally found in multi-objective searches where different objectives contribute different amounts to the overall objective function value.

The problem caused by pareto-optimal solutions is simply to choose which parameter set is truly the best. In the case of the problems presented in this study, this can be answered by simply calculating the torque requirements of the resulting mechanism. However, for this to be introduced into the search means a return to computationally expensive objective functions.

There are several other methods which could be applied in this case to help the search decide the relative merits of otherwise pareto-optimal solutions during the search. The simplest of these is to introduce a degree of intelligence into the objective function based around simple knowledge based rules. One such rule has already been proposed in section 6.2.

This rule can be expressed as a heuristic;

IF (desired curve harmonic content) is high.
THEN (swept area) is more significant

If this direction is chosen for future work, then it is obvious that more knowledge needs to be acquired about hybrid machines in order to generate sufficient heuristics to allow a suitable objective function to be formed.

One other method for dealing with pareto-optimality is the definition of major and minor functions as used by **Donne *et al*** [123]. As an example, the objectives defined in this study could be defined as major functions. Minor functions can then be used to further differentiate between pareto-optimal solutions. One possible approach would be to calculate the transmission angles within the mechanism as it completes the cycle, then select between two mechanisms based on this secondary consideration.

Perhaps the most common manner of dealing with pareto-optimal solutions is to not use a weighted objective function but to store all solutions that match feasibility criteria based on objective function component values. These solutions are then presented to the designer to select the best solution.

This approach has some merit in that by storing different solution parameter sets it is possible to predict the pareto-optimal boundary front and hence predict other possible solutions by examination. However, it is not really a viable option for complex functions as the pareto-optimal boundary front is often so large that the time taken to store all pareto-optimal solutions becomes more significant than the search time itself.

6.3 Practical Implementation of a Real Machine

In Chapter Five, a local search method based upon a steepest descent method is used to improve upon solutions obtained in Chapter Four. This combination of search methods produces a high quality theoretical solution. This double pass methodology is a new approach to mechanism synthesis.

However, it is very important to consider the effectiveness of the method by producing a real machine so that practical issues can be assessed. These practical issues can be divided into two areas. These are problems associated with the implementation and commissioning of the machine and also problems associated with the control of the machine.

6.3.1 Commissioning

During the commissioning of the machine several problems were encountered. These problems arose due to a number of reasons, mainly an inexperience of commissioning servo driven systems as well as errors and inconsistencies in the installation manuals provided by the servo drive manufacturer.

These problems highlighted several important factors. The first of these was that it is essential to ensure that no local earthing loops are present in the power circuitry. Such earthing loops give rise to differences in electrical potential which can cause the motors to randomly shudder about the demand position. Such motor shudder can also be due to improper shielding of power, resolver and encoder simulation cables.

Whatever the cause, motor shudder cannot be tolerated in a servo system. This is particularly true in a system such as the hybrid machine where two inputs are mechanically coupled together. When a motor shudders away from the desired position, the other motor is pulled from it's desired position due to the linkage

between them. This effect is known as cross coupling and is essentially a control problem.

6.3.2 Control

One of the major problems associated with the control of hybrid machines is the effects due to cross coupling. If a noisy signal causes one motor to shudder about its desired position the mechanical coupling causes the other motor to be pulled from its desired position. The control algorithm will then try to correct both positions, often causing a higher position error. This normally results in an increasing oscillation that ultimately leads to an unstable system.

The effects of cross coupling can be reduced by correct system implementation to reduce any shudder on each axis. It is also essential to have a correctly tuned control algorithm. In this respect, the control algorithm should be considered in both the speed loop and the position loop.

This highlights one of the most important aspects of hybrid machine implementation. Previous work by **Bradshaw *et al*** [120] has shown that a PID controller with an IMFF (Inverse Model Feed Forward) component provides adequate control. However, it does not deal with how best to assign control parameters. Tuning rules, such as those outlined in Chapter Five, may not always provide a sufficiently rigorous approach to selecting optimal control gains.

This is particularly true if a very accurate output motion is required. Experimental tuning methods suffer from the disadvantage that as each tuning parameter is altered, then the bandwidth for acceptable settings of other parameters changes. For example, if the proportional gain is set to the critical value then increasing the velocity feedback gain will allow even higher values for the proportional gain to be used. The implication of this is that experimental tuning processes require an increasing degree

of effort to implement as the tolerance on the output motion precision is made tighter.

A full consideration of control issues was always beyond the scope of this study. However, it is important to stress the importance of having a robust controller. In addition to establishing methods for tuning PID controllers, more work is required to investigate other robust control techniques and could concentrate on the use of adaptive or state space methods. One other possibility is to design a control scheduling system. In control scheduling, the gains of the controller are adjusted as the mechanism completes each cycle. This could be very simple to achieve as the results in Chapter Five indicate that the errors associated with the inputs to the mechanism are generally cyclic in nature. This is similar to the Inverse Model Feed Forward approach, in that system specific information is used to improve the control system.

Another alternative is to introduce into the control algorithm information concerning the actual position of the end effector. This could possibly be achieved by tracking the trajectory of the end effector using image processing hardware then using this to predict the position of the end effector. Comparing this predicted position with the desired position should allow the velocities in the machine to be adjusted accordingly. A simpler alternative could be developed where tensioned wires are attached to the end effector. Linear encoders on these wires allow the end effector position to be calculated. This method may not be applicable in an industrial implementation and lacks some of the elegance of using image processing technology. However, it requires less computing power and so may be the most feasible option.

6.3.3 Machine Evaluation

Whilst the output of the machine has no doubt been affected by the non-optimal gain settings used in the position controller it is still reasonably accurate. The desired

coupler curve is approximated, despite significant errors on the inputs to the machine. However, more work is required to understand how an error on the motor inputs affects the position of the end effector. This could include further use of video imaging equipment which operates at speeds higher than currently used. Alternatively, models could be developed which show not only the affects of a fluctuating input, but also predict the response of the control algorithm.

Despite the limitations of the current control implementation, the machine can be said to be practically implemented. The implication of this is that the design methodology developed produces results that can be turned into real machines. These machine solutions also exhibit desirable characteristics, both theoretically and practically

The power distribution between CV and servo motors is such that the CV motor provides the bulk of the motion whilst the servo motor provides a degree of modulation which allows the desired output coupler curve to be generated. When this occurs, there is more potential for energy regeneration by utilising a flywheel on the CV motor axis.

In addition to running the machine at the test speed of 60 rpm, some experimentation has been carried out to observe the effects of increasing the speed of operation. At 90 rpm, without altering the controller tuning gains, the response of the mechanism is poor. The end effector still traces a crude approximation to the desired coupler curve, but the oscillations on the servo motor at the end of each cycle are so large so as to produce significant error on the CV axis. However, it is possible to retune the controller so that performance is improved. As the speed of operation is further increased, it becomes increasingly difficult to tune the controller so that adequate performance is maintained. This may be due to the increase in magnitude of the out of balance forces and the severity of the servo motor motion profile. It is suggested that future work considers the balancing of the mechanism to enable higher speeds to be reached. As the speed of operation is increased, it also becomes increasingly more important to change tuning gains by fine steps, as severe oscillations can occur when gains are adjust by large amounts. These large oscillations often flick the mechanism

into its other closure which leads to the possibility of damage to the machine due to the interference of the links.

The implication is that more research is required before the level of understanding of hybrid machines is sufficiently understood so that they could become industrially viable machine design solutions. This must include work concerned with the safety aspects of hybrid machines and in particular methods which inhibit the possibility of closure changes. Controlling the mechanism so that good performance is achieved, but changes of closure do not occur, presents a particularly difficult problem.

6.4 Summary

This Chapter has discussed the results presented in earlier Chapters. By examining these results it has been shown that the novel design methodology developed during this study has gone some way to providing a robust synthesis tool for hybrid mechanisms.

The practical results indicate that the solutions obtained can be turned into real machines which approximate the desired motions with desirable power and torque characteristics. What errors that have occurred have been attributed to the tuning of the controller.

Chapter Seven

Conclusions

7.1 Introduction

The purpose of this Chapter is to summarise the development of the novel design methodology based on Genetic Algorithms, briefly state the importance of the results achieved and draw conclusions concerning the merit of the methodology based on the discussion in Chapter Six.

Whilst this Chapter concludes this thesis, it is important to realise that this thesis does not cover all possible avenues for research in this area.

7.2 Summary of Design Methodology

In Chapter One, it was stated that the aim of this research was to develop a robust design methodology for multi-degree of freedom mechanisms with specific applications in the field of hybrid machine design. Initial investigations showed that Genetic Algorithms could prove to be a suitable automatic design optimisation method for use in this area.

Chapter Three provides an introduction to the mechanics of Genetic Algorithms. In essence, GAs are a method based upon the principles of Darwinian evolution and natural selection. By defining a problem as an environment in terms of an objective function, the method seeks to "evolve" good solutions by modelling the main processes by which genetic information is passed from parents to children in natural systems. These processes, or operators, are reproduction, crossover and mutation. In addition to these, the GA used in this study utilises an unnatural operator known as

inversion. This has been included so as to remove identical solutions from the population so as to maximise the search potential of each generation.

The objective function used in obtaining the results presented in Chapter Four was a multi-objective function based around the following criteria;

1. Least squares error
2. Mobility penalty function
3. Servo link swept area
4. Harmonic content of servo displacement

Each of these criteria was proposed for inclusion for different reasons, though all were intended to force the method towards high quality solutions. The least squares error between the curve generated by the trial mechanism and the desired curve should be viewed as the most important component for applications where precision location of the end effector is required. In addition to this, it is also important to ensure that the mechanism is feasible in terms of mobility. This is achieved by including an accelerating penalty function based on the number of mechanism positions around the cycle that cannot be reached.

The final two components of the objective function were included so as to try and approximate a dynamic objective function. Designing mechanisms where the bulk of the motion is provided by the CV motor has many implications. For example, it allows the use of a flywheel on the CV motor axis to maximise the energy regeneration potential of the system and reduces the size of the servo motor. This results in a reduced installation cost and implies that higher speeds of operation may be possible or that savings in power consumption can be made.

However, using such dynamic criteria in a parallel search method, such as a Genetic Algorithm, would be computationally expensive. The swept area term in the kinematic approximation function is included because the motor torque requirement is a function of the motor displacement profile and the link lengths of the mechanism. The swept area term is calculated by multiplying the RMS of the servo

displacement profile by the length of the link attached to the servo motor. This is not a true approximation of dynamic conditions as the effects of the other links and the CV motor are not accounted for.

The harmonic content of the servo displacement is included for similar reasons to the swept area term. The torque requirement of the motor is dependent on the acceleration profile of the servo motor and so a smooth displacement profile should produce lower torques. The importance of the harmonic content of the displacement profile can be seen by considering an abstract example. Simple harmonic motion is infinitely differentiable without introducing discontinuities in high order derivatives. Therefore, lowering the magnitude of high order harmonics trends the displacement profile towards simple harmonic motion. This leads to an improvement in torque requirement.

7.3 Critical Analysis of Results

The results presented in Chapter Four indicate that some success has been achieved in developing the novel methodology. The considerable time spent developing a robust GA has resulted in an effective synthesis method. In all cases, the method has produced high quality mechanisms with desirable torque distributions. However, by analysing the objective function values for different mechanisms and comparing each mechanisms torque characteristics it can be seen that the objective function used does not provide a direct mapping between kinematic approximation and truly dynamic optimality. This may be partly due to the exclusion of the effects of the intermediate links of the mechanism on the criteria used to assess the quality of the servo motor input and also partly due to the weightings used in the objective function. Whilst this direct mapping does not exist, the method does locate acceptable solutions in most cases.

In Chapter Five, results are presented for a real machine designed around one of the solution sets presented in Chapter Four. These results show that not only does the

methodology locate feasible solutions but that these solutions can be turned into working machines. However, a greater understanding of the control of hybrid machines is required before such machine modules can be classed as industrially robust design solutions.

7.4 Suggestions for Future Work

The work described in this thesis should be viewed as an inital exploration of the application of an AI search method to the synthesis of hybrid mechanisms. Whilst a degree of sucess has been acheived in developing the method, including some novel design objectives, there is still a large amount of potential research in this field.

7.4.1 Design Methodology

The ideas behind the design methodology presented in this work are essentially valid, but there is still scope for improvements to be made to the methodology. In terms of the Genetic Algorithm, there are several ways in which the performance could be improved. This includes the development of a parallel algorithm as well as the inclusion of a sharing scheme.

In a parallel GA, separate sub-populations are maintained and each gene pool produces local solutions. Poor individuals in one sub-population are often replaced by good individuals from another. Such migration ensures that all sub-populations contain highly fit solutions. It may be appropriate to utilise a similar approach to **Schaffer** [93], where each sub-population was used to represent each objective function criteria.

A parallel GA may be of benefit to the methodology as they have been shown to contain a more diverse set of solutions than a serial GA. This effect can also be

achieved by using sharing in a serial GA, though the development of sub-populations is in a much more implicit manner and it is more difficult to control any migration.

Increased diversity within the population, no matter how it is achieved, implies that the method will locate more pareto-optimal solutions. Therefore, along with this development, work should be carried out into dealing with complex multi-criteria objective functions and the pareto-optimality condition. This may be best achieved by introducing a degree of intelligence into the objective function to replace the static weightings currently used. This degree of intelligence could be in the form of heuristics such as suggested in section 6.2.2.

In addition to developments within the workings of the method, the scope of the method should also be expanded. By applying the method to a Svoboda seven bar function generator the effectiveness of the dynamic approximate objectives can be assessed by comparing results to those achieved by **Greenough** [3,5].

Should the results lack the quality of previous work, the method can then be altered to a double pass methodology where the output from the GA search is used in a local search based around the dynamic objectives already developed. Should this prove necessary, then it may be best to utilise a Tabu search algorithm as the local search engine. Such a double pass methodology is likely to produce very high quality solutions without the high computation time associated with previous work.

In addition to considering the current design objectives, work should also be carried out to develop yet more novel objectives. These could include objectives which minimise the out of balance forces for a given mechanism. One possibility, other than simple balancing, is to use simple stress analysis techniques to consider the stresses in each link caused by the joint forces. Once the stresses are known, it would be possible to redesign the shape of each link so that the mass is minimised. This would then reduce the out of balance forces.

Future work should also consider the degree of modulation that is available to a given linkage by changing the servo motor motion profile. An understanding of how

much this profile can be changed, without causing interference in the linkage, and how much these changes effect the output coupler curve will enable a design approach to be developed which maximises the flexibility of a given mechanism.

Finally, the method could be generalised so that the designer can select the type of output required and so select a desired mechanism type as well as design objectives. To achieve this, the nature of the GA code must be changed to eliminate the use of symbolic constants to define chromosome lengths and parameter definitions. The most difficult area of this generalisation would be to allow the user to specify the objective function. One possible approach is to compile the GA as object code or a .DLL file and allow the user to write their own objective function to link to the GA. An alternative approach is to provide the user with a library of pre-written functions.

Future work should also consider how the current methodology can be extended into a comprehensive automated machine design toolbox. Generalising the GA, and providing the user with options concerning the mechanism type and output motion required, as well as options concerning the design objectives, goes some way towards such a toolbox. However, it may be applicable to provide a knowledge based user interface which interprets the designers motion requirements using a natural language parser so providing an automated type synthesis method.

7.4.2 Control Techniques

This study has shown that the experimental tuning of hybrid machines can produce adequate performance, though the setting of the gain parameters can be tedious and time consuming. It is recommended that further research into the control of hybrid machines should be carried out to develop methods which do not require such tedious tuning approaches. Possible avenues include the use of adaptive control and state space methods.

There are several other techniques which could also be investigated. The first of these is to implement control scheduling into the machine so that improved performance is achieved. An alternative method for improving performance would be to introduce the actual position of the end effector into the control algorithm.

Finally, it may be possible to develop complex models of the mechanism and controller. These models could be used in a variety of ways. Firstly, they could be used to gain a better understanding of the effects of cross coupling in the system. They could also be embedded into the control system user interface and be used to judge the effects of changing controller gains or motion profiles before they are physically changed, This would be of great use in terms of the safety aspects of introducing hybrid machines into an industrial environment.

7.5 Conclusions

In this thesis, a totally novel approach to mechanism design has been formulated for use in the synthesis of hybrid mechanisms. The methodology utilises multi-criteria design objectives to approximate a dynamically optimum hybrid mechanism. The results presented in this thesis show that the novel methodology produces acceptable results, though more research is required in a number of areas if the hybrid machine approach is to become an industrially viable option for non uniform motion generation.

Appendix A

Bibliography

A.1 Cited References

[1] Tokuz, L.C. (1992)
Hybrid Machine Modelling and Control
Ph.D. Thesis, Liverpool Polytechnic

[2] Tokuz, L.C. & Rees Jones, J. (1991)
Programmable Modulation of Motion Using Hybrid Machines
Institution of Mechanical Engineers, C414/071, pp 85-91

[3] Greenough, J.D. (1993)
Design of High Speed Machines
M.Phil/Ph.D. Transfer report, Liverpool John Moores University

[4] Bradshaw, W.K. (1993)
Control of Hybrid Machine Modules
M.Phil/Ph.D. Transfer report, Liverpool John Moores University

[5] Greenough, J.D. (1994)
Design and Control of Hybrid Machines
Final Report (Internal Document), Liverpool John Moores University

[6] Thomson, T.R., Riley, D.R. & Erdman, A.G. (1985)
An Expert System Approach to Type Synthesis of Mechanisms
Proceedings of the 1985 ASME Computers in Engineering Conference, Vol 2, pp 71-75

[7] Erdman, A.G., Thomson, T.R. & Riley, D.R. (1988)
Type Selection of Robot and Gripper Kinematic Topology Using Expert Systems
International Journal of Robotics Research, Vol 5, No2, pp 183-189

[8] McGarva, J.R. & Mullineux, G. (1993)
Harmonic Representation of Closed Curves
Applied Mathematical Modelling, Vol 17, No 4, pp 213-218

[9] McGarva, J.R. & Mullineux, G. (1992)
A New Methodology for Rapid Synthesis of Function Generators
Proceedings of the IMechE, Part C, Vol 206, No 6, pp 391-398

[10] McGarva, J.R. (1994)
Rapid Search and Selection of Path Generating Mechanisms from a Library
Mechanisms & Machine Theory Vol 29, No 2, pp 223-235

[11] Powell, M.J.D. (1965)
A Method for Minimising a Sum of Squares of Non-Linear Functions Without Calculating the Derivatives
The Computer Journal, Vol 7, pp 303-307

[12] Hoeltzel, D.A. & Chieng, W-H. (1990)
Pattern Matching Synthesis as an Automated Approach Knowledge *to Mechanisms Design*
Transactions of the ASME : Journal of Mechanical Design,, Vol 112, No 2, pp 190-199

[13] Hoelztel, D.A., Quadracci, H.R. & Chieng, W-H. (1990)
Path Generation Based Pattern Matching for Automated Mechanism Synthesis
Proceedings the 21st ASME Biennial Conference : Mechanism Synthesis & Analysis, Vol DE v25, pp 341-351

[14] Rosen, D., Riley, D. & Erdman, A. (1991)
A Knowledge Based Dwell Mechanism Assistant Designer
Journal of Mechanical Design, Transactions of the ASME, Vol 113, No 3, pp 205-212

[15] Jobes, C.C., Palmer, G.M. & Means, K.H. (1990)
Synthesis of a Controllable Circuit Breaker Mechanism
Transactions of the ASME : Journal of Mechanical Design, Vol 112, pp 205-212

[16] Root, R.R. & Ragsdell, K.M. (1976)
A Survey of Optimisation Methods Applied to the Design of Mechanisms
Transactions of the ASME : Journal of Engineering for Industry, pp 1036-1041

[17] Bogdan, R.C. & Larionescu, D. (1971)
Complex Harmonic Synthesis of the Binary Structural Groups
Proceedings of the Third World Congress for the Theory of Machines and Mechanisms, Vol D, Paper D2

[18] Bogdan, R.C. & Larionescu, D. (1971)
Applications of the Complex Harmonic Synthesis in Five and Four Bar Mechanisms of I-Aspect
Proceedings of the Third World Congress for the Theory of Machines and Mechanisms, Vol C, Paper C3

[19] Soni, A.H., Kohli, D., Srivastv, D., Dewey, G. & Hall, D. (1972)
Synthesis of Six-Link Mechanisms for Rigid Body Guidance and Function Generation
ASME Publication No. : 72-Mech-81

[20] Fox, R.L. & Willmert, K.D. (1967)
Optimum Design of Curve Generating Linkages with Inequality Constraints
Transactions of the ASME : Journal of Engineering for Industry, Vol 89, No 1, pp 144-152

[21] Fletcher, R. & Powell, M.J.D. (1963)
A Rapidly Convergent Descent Method for Minimisation
The Computer Journal, Vol 6, pp 163-168

[22] Nolle, H. (1967)
On the Capability of the Four Bar Mechanism as a Function Genertor
Mechanical and Chemical Engineering Transactions, Institution of Engineers, Australia, Vol MC 3, pp 207-213

[23] Garrett, R.E. & Hall, A.S. (1968)
Optimum Synthesis of Randomly Generated Linkages
Transactions of the ASME : Journal of Engineering for Industry, Vol 90, pp 475-480

[24] Kwak, B.M. & Haug, E.J. (1976)
Optimal Synthesis of Planar Mechanisms by Parametric Design Techniques
Engineering Optimisation, Vol 2, pp 55-63

[25] Rao, S.S. & Hati, S.K. (1979)
Game Theory Approach in Multicriteria Optimisation of Function Generating Mechanisms
Transactions of the ASME : Journal of Mechanical Design, Vol 101, No 3, pp 398-406

[26] Rao, A.C. (1979)
Synthesis of Four Bar Function Generators Using Geometric Programming
Mechanisms and Machine Theory, Vol 14, No 2, pp 141-149

[27] Ion, S. & Cezar, D. (1979)
On a General Method for the Synthesis of the Path Approximation Mechanism
Mechanisms & Machine Theory, Vol 14, No 5, pp 289-298

[28] Paradis, M.J. & Wimert, K.D. (1983)
Optimal Mechanism Design Using the Gauss Constrained Method
Transactions of the ASME : Journal of Mechamisms, Transmissions and Automation in Design, Vol 105, No 2, pp 187-196

[29] Ma, O. & Angeles, J. (1988)
Performance Evaluation of Path Generating Planar, Spherical and Spatial Four Bar Linkages
Mechanisms & Machine Theory, Vol 23, No 4, pp 257-268

[30] Cleghorn, W.L., Fenton, R.G. & Fu, J-F. (1990)
A General Method for Synthesis of the Coupler Point Path of Planar Four Bar Mechanisms
Proceedings of the 21st ASME Biennial Conference : Mechanism Synthesis and Analysis, Vol DE v25, pp 209-215

[31] Subbian, T. & Flugrad, D.R. (1991)
Four Bar Path Generation Synthesis by a Continuation Method
Transactions of the ASME : Journal of Mechanical Design, Vol 113, pp 63-69

[32] Starns, G.K. & Flugrad, D.R. (1993)
Five Bar Path Generation Synthesis by Continuation Methods
Transactions of the ASME : Journal of Mechanical Design, Vol 115, pp 988-994

[33] Subbian, T. & Flugrad, D.R. (1992)
Synthesis of an RSSR-SS Mechanism Using a Continuation Method
Proceedings of the 22nd ASME Biennial Conference : Mechanism Design and Synthesis, Vol DE v46, pp 579-586

[34] Subbian, T. & Flugrad, D.R. (1993)
Five Position Triad Synthesis With Applications to Four and Six Bar Mechanisms
Transactions of the ASME : Journal of Mechanical Design, Vol 115, No 2, pp 262-268

[35] Tsai, L-W. & Lu, J-J. (1990)
Coupler Point Curve Synthesis Using Homotopy Methods
Transactions of the ASME : Journal of Mechanisms, Transimissions and Automation in Design, Vol 112, No 3, 384-389

[36] Jain, P. & Agogino, A.M. (1988)
Optimal Design of Mechanisms Using Simulated Annealing : Theory and Applications
Proceedings of the 14th ASME Design Automation Conference, pp 233-240

[37] Connor, A.M. (1994)
The Use of Genetic Algorithms in Optimisation
M.Sc. Dissertation, Liverpool John Moores University

[38] Connor, A.M., Douglas, S.S. & Gilmartin, M.J. (1995)
The Synthesis of Path Generating Mechanisms Using Genetic Algorithms
Proceedings of the 10th International Conference of Applications of Artificial Intelligence, pp 237-244

[39] Ogot, M.M. & Alag, S. (1993)
A Stochastic Methodology for the Optimal Analytical Synthesis of Planar Mechanisms
Proceedings of the 19th Annual ASME Design Automation Conference, Vol DE v65, Part 1, pp 449-458

[40] Malik, A.K. & Dhande, S.G. (1987)
Analysis and Synthesis of Mechanical Error in Path Generating Linkages Using a Stochastic Approach
Mechanisms and Machine Theory, Vol 22, No.2, pp 115-123

[41] Yin, Z.W. & Wu, J.K. (1990)
An Optimal Synthesis of Linkages Considering Structural Error and Clearances
Proceedings the 21st ASME Biennial Conference : Mechanism Synthesis & Analysis, Vol DE v25, pp 295-299

[42] Ling, Z.K. (1991)
Technique for the Design of an Interference Free Complex Planar Mechanism
Proceedings of the 17th ASME Design Automation Conference, Vol DE v32, Part 2, pp 435-441

[43] Kochev, I.S. (1990)
General Method for Active Balancing of Combined Shaking Moment and Torque Fluctuations in Planar Linkages
Mechanisms and Machine Theory, Vol 25, No 6, pp 679-687

[44] Rao, S.S. & Kaplan, R.L. (1986)
Optimal Balancing of High Speed Linkages Using Multiobjective Programming Techniques
Transactions of the ASME : Journal of Mechanisms, Transmissions and Automation in Design, Vol 108, pp 454-460

[45] Dhingra, A.K. & Rao, S.S. (1991)
An Integrated Kinematic-Kinetostatic Approach to Optimal Design of Planar Mechanisms Using Fuzzy Theories
Transactions of the ASME : Journal of Mechanical Design, Vol 113, pp 306-311

[46] Lee, M-Y., Erdman, A.G. & Gutman, Y. (1993)
Development of Kinematic/Kinetic Performance Tools in Synthesis of Multi Degree of Freedom Mechanisms
Transactions of the ASME : Journal of Mechanical Design, Vol 115, pp 462-471

[47] Lee, M-Y., Erdman, A.G. & Gutman, Y. (1991)
Applications of Kinematic/Kinetic Performance Tools in Synthesis of Multi Degree of Freedom Mechanisms
Proceedings of the 17th ASME Design Automation Conference, Vol DE v32, Part 2, pp 279-290

[48] Lee, M-Y., Erdman, A.G. & Gutman, Y. (1993)
Kinematic/Kinetic Performance Analysis and Synthesis Measures of Multi Degree of Freedom Mechanisms
Mechanisms & Machines Theory, Vol 28, No 5, pp 651-670

[49] Chuenchom, T. & Kota, S. (1992)
Generalised Synthesis of Adjustable Mechanisms
Proceedings of the 22nd ASME Biennial Conference : Mechanism Design and Synthesis, Vol DE v46, pp 253-260

[50] Svoboda, A. (1948)
Computing Mechanisms & Linkages
McGraw-Hill

[51] Pollit, E.P. (1962)
Five Bar Linkages With Two Drive Cranks
Machine Design, January, pp 168-179

[52] Kramer, S.N. & Sandor,G.N. (1969)
Finite Kinematic Synthesis of a Cycloidal Crank Mechanism for Function Generation
ASME Publication No. : 69-WA/DE-1

[53] Rooney, G.T. (1970)
Synthesis of Five Bar Mechanisms as Function Generators
Ph.D. Thesis, Liverpool Polytechnic

[54] Erdman, A.G. & Sandor, G.N. (1970)
Kinematic Synthesis of a Geared Five Bar Function Generator
ASME Publication No. : 70-Mech-2

[55] Pafelias, T.A. & Sandor, G.N. (1972)
Synthesis of a Geared 'N' Bar Linkage for Path Generation
ASME Publication No. : 72-Mech-9

[56] Kohli, D. & Soni, A.H. (1972)
Synthesis of Seven Link Mechanisms
ASME Publication No. : 72-Mech-35

[57] Freudenstein, F. & Lee, T.W. (1978)
Design of Geared Five Bar Mechanisms for Unlimited Crank Rotations and Optimum Transmissions
Mechanisms and Machine Theory, Vol 13, No 2, pp 235-244

[58] Ting, K.L. & Tsai, G.H. (1985)
Mobility and Synthesis of Five Bar Programmable Linkages
Proceedings of the 9th OSU Applied Mechanisms Conference, Part III, pp 1-8

[59] Ting, K.L. (1986)
Five Bar Grashof Criteria
Transactions of the ASME : Journal of Mechanisms, Transmissions and Automation in Design, Vol 108, pp 533-537

[60] Basu, P.S. & Farhang, K. (1992)
Kinematic Analysis and Design of Two Input, Five Bar Mechanisms Driven by Relatively Short Cranks
Proceedings of the 22nd ASME Biennial Conference : Flexible Mechanisms Dynamics and Analysis, Vol DE v47, pp 377-386

[61] Alizde, R.I., Mohan Rao, A.V. & Sandor G.N. (1975)
Optimum Synthesis of Two Degree of Freedom Planar and Spatial Function Generating Mechanisms Using the Penalty Function Approach
Transactions of the ASME : Journal of Engineering for Industry, May, pp 629-634

[62] Sugimoto, K. & Hara, A. (1991)
Synthesis of Multi-DOF Mechanisms by Using Connecting Chains
Advanced Robotics, Vol.6, No.1, pp 95-108

[63] Sunder, S. & Shiller, Z. (1994)
Constrained Optimisation of Multi-Degree of Freedom Mechanisms for Near Time Optimal Motions
Transactions of the ASME : Journal of Mechanical Design, Vol 116, pp 412-418

[64] Shiller, Z. & Sunder S. (1993)
Design of Multi-Degree of Freedom Mechanisms for Optimal Dynamic Performance
Transactions of the ASME : Journal of Mechanical Design, Vol 115, pp 199-206

[65] Huissoon, J.P. & Wang, D. (1991)
On the Design of a Direct Drive 5-Bar-Linkage Manipulator
Robotica, Vol.9, pp 441-446

[66] Kirecchi, A. (1993)
Motion Design for High Speed Machinery
Ph.D. Thesis, Liverpool John Moores University

[67] Kirkpatrick, S., Gelatt, C. & Vecchi, M. (1983)
Optimisation by Simulated Annealing
Science, Vol 220, pp 671-680

[68] Sinclair, M. (1993)
Comparison of the Performance of Modern Heuristics for Combinatorial Optimisation on Real Data
Computers & Operations Research, Vol 20, No 7, pp 687-695

[69] Borup, L. & Parkinson, A. (1992)
Comparison of Four Non-Derivative Optimisation Methods on Two Problems Containing Heuristic and Analytic Knowledge
Proceedings of the 18th ASME Design Automation Conference, Vol DE v44, pp 137-143

[70] Gill, P.E., Murray, W. & Wright, M.H. (1981)
Practical Optimisation
Academic Press

[71] Glover, F. (1989)
Tabu Search - Part I
ORSA Journal on Computing, Vol 1, No 3, pp 190-206

[72] Glover, F. (1990)
Tabu Search - Part II
ORSA Journal on Computing, Vol 2, No 1, pp 4-32

[73] Bland, J.A. & Dawson, G.P. (1991)
Tabu Search and Design Optimisation
Computer Aided Design, Vol 23, No 3, pp 195-201

[74] Bardou, O. & Sidahmed, M. (1994)
Early Detection of Leakages in the Exhaust and Discharge Systems of Reciprocating Machines by Vibration Analysis
Mechanical Systems & Signal Processing, Vol 8, No 5, pp 551-570

[75] Sanyal, A.J. (1995)
The Use of Artificial Neural Networks for Corrosion Prediction
M.Sc. Dissertation, Liverpool John Moores University

[76] Stylios, G. & Sotomi, O.J. (1994)
A Neuro-Fuzzy Control System for an Intelligent Sewing Machine
Proceedings of the Fifth IEEE Conference on Artificial Intelligence Applications, IEEE Publication No 398, pp 543-551

[77] Goldberg, D.E. (1989)
Genetic Algorithms in Search, Optimisation & Machine Learning
Addison-Wesley

[78] Pham, D.T. & Yang, Y. (1993)
Optimisation of Multi-Modal Discrete Functions Using Genetic Algorithms
Proceedings of the IMechE, Part D, Vol 207, pp 53-59

[79] Ball, N.R., Sargent, P.M. & Ige, D.O. (1993)
Genetic Algorithm Representation for Laminate Layups
Artificial Intelligence in Engineering, Vol 8, pp 99-108

[80] Chen, J.L. & Tsao, Y-C. (1993)
Optimal Design of Machine Elements Using Genetic Algorithms
Journal of the Chinese Society of Mechanical Engineers, Vol 14, No 2, pp 193-199

[81] Stuckman, B., Evans, G. & Mollaghasemi, M. (1991)
Comparison of Global Search Methods for Design Optimisation Using Simulation
Proceedings of the IEEE Winter Simulation Conference, pp 937-944

[82] Bramlette, M.F. & Cusic, R. (1989)
A Comparative Evaluation of Search Methods Applied to Parametric Design of Aircraft
Proceedings of the Third International Conference on Genetic Algorithms, pp 213-218

[83] Hollstein, R.B. (1971)
Artificial Genetic Adaptation in Computer Control Systems
Ph.D. Thesis, University of Michigan

[84] Krishnakumar, K. & Goldberg, D.E. (1992)
Control System Optimisation Using Genetic Algorithms
Journal of Guidance, Control & Dynamics, Vol 15, No 3, pp 735-740

[85] Suchen, S. & Kiichi, T. (1993)
Learning of a Maze Using a Genetic Algorithm
Proceedings of the 19th Annual International Conference on Industrial Electronics, Control and Instrumentation, Vol 1, pp 376-379

[86] Curtis, A.R.D. (1991)
Application of Genetic Algorithms to Active Vibration Control
Journal of Intelligent Material Systems and Structures, Vol 2, No 4, pp 472-481

[87] Lin, J-L., Foote, B., Pulat, S., Chang, C-H. & Cheung, J.Y. (1993)
Hybrid Genetic Algorithm for Container Packing in Three Dimensions
Proceedings of the 9th IEEE Conference on Artificial Intelligence for Applications, pp 353-359

[88] Gibson, P.M. & Byrne, J.A. (1991)
Neurogen : Musical Composition Using a Genetic Algorithm and Cooperating Neural Networks
Proceedings of the 2nd International Conference on Artificial Neural Networks, pp 309-313

[89] Connor, A.M., Douglas, S.S. & Gilmartin, M.J. (1995)
The Synthesis of Hybrid Five Bar Path Generating Mechanisms Using Genetic Algorithms
Proceedings of the 1st IEE Conference on Genetic Algorithms in Engineering Systems : Innovations and Applications, IEE Publication No 414, pp 313-318

[90] Rajeev, S. & Krishnamoorthy, C.S. (1992)
Discrete Optimisation of Structures Using Genetic Algorithms
Journal of Strutural Engineering, Vol 118, No 5, pp 1233-1250

[91] Cavicchio, D.J. (1970)
Adaptive Search Using Simulated Evolution
Ph.D. Thesis, University of Michigan

[92] De Jong, K.A. (1975)
An Analysis of the Behaviour of a Class of Genetic Algorithms
Ph.D. Thesis, University of Michigan

[93] Schaffer, J.D. (1984)
Some Experiments in Machine Learning Using Vector Evaluated Genetic Algorithms
Ph.D. Thesis, Vanderbilt University

[94] Perry,Z.A. (1984)
Experimental Study of Speciation in Ecological Niche Theory using Genetic Algorithms
Ph.D. Thesis, University of Michigan

[95] Grosso, P.B. (1985)
Computer Simulation of Genetic Adaptation : Parallel Subcomponent Interaction in a Multi-Locus Model
Ph.D. Thesis, University of Michigan

[96] Goldberg, D.E. & Richardson, J. (1987)
Genetic Algorithms with Sharing for Multi-Modal Function Optimisation
Proceedings of the Second International Conference on Genetic Algorithms, pp 41-49

[97] Maudlin, M.L. (1984)
Maintaining Diversity in Genetic Search
Proceedings of the National Conference on Artificial Intelligence, pp 247-250

[98] Deb, K. & Goldberg, D.E. (1989)
An Investigation of Niche and Species Formation in Genetic Function Optimisation
Proceedings of the Third International Conference on Genetic Algorithms, pp 42-50

[99] Deb, K. (1989)
Genetic Algorithms as Multi-Modal Function Optimisers
M.Sc. Dissertation, University of Alabama

[100] Booker, L.B. (1982)
Intelligent Behaviour as an Adaptation to the Task Environment
Ph.D. Thesis, University of Michigan

[101] Booker, L.B. (1984)
Improving The Performance of Genetic Algorithms in Classifier Systems
Proceedings of an International Conference on Genetic Algorithms and Their Applications, pp 80 -92

[102] Eshelman, L.J. & Schaffer, J.D. (1991)
Preventing Premature Convergence of Genetic Algorithms by Preventing Incest
Proceedings of the Fourth International Conference on Genetic Algorithms, pp 115-122

[103] Bhandari, D. et al (1994)
Directed Mutation in Genetic Algorithms
Information Sciences, Vol 79, pp 251-270

[104] Yao, X (1993)
An Empirical Study of Genetic Operators in Genetic Algorithms
Microprocessing and Microprogramming, Vol 38, pp 707-714

[105] Lin, C.Y. & Hajela, P. (1993)
Genetic Search Strategies in Large Scale Optimisation
Proceedings of the AIAA 34th Conference on Structures, Structural Dynamics and Materials, pp 2437-2447

[106] Whitley, D. et al (1991)
Delta Coding : An Iterative Search Strategy for Genetic Algorithms
Proceedings of the Fourth International Conference on Genetic Algorithms, pp 77-84

[107] Eshelman, L. (1991)
The CHC Adaptive Search Algorithm : How to have Safe Search when Engaging in Nontraditional Genetic Recombination
Proceedings of the Fourth International Conference on Genetic Algorithms, pp 85-92

[108] Krishnakumar, K. (1989)
Micro-Genetic Algorithms for Stationary and Non-Stationary Function Optimisation
SPIE Conference on Intelligent Control and Adaptive Systems, pp 106-112

[109] Goldberg, D.E. (1989)
Zen and the Art of Genetic Algorithms
Proceedings of the Third International Conference on Genetic Algorithms, pp 80-85

[110] Caruana, R.A. & Schaffer, J.D. (1988)
Representation and Hidden Bias : Gray vs. Binary Coding for Genetic Algorithms
Proceedings of the Fifth International Conference on Machine Learning, pp 152-161

[111] Levenick, J.R. (1991)
Inserting Introns Improves Genetic Algorithm Success Rate : Taking a Cue from Biology
Proceedings of the Fourth International Conference on Genetic Algorithms, pp 123-127

[112] Baker, J.E. (1987)

Reducing Bias in the Selection Algorithm

Proceedings of the Second International Conference on Genetic Algorithms, pp 14-21

[113] Baker, J.E. (1985)

Adaptive Selection Methods for Genetic Algorithms

Proceedings of an International Conference on Genetic Algorithms and Their Applications, pp 101-111

[114] Grefensttete, J.J. (1986)

Optimisation of Control Parameters for Genetic Algorithms

Journal of Systems, Man and Cybernetics, Vol16, No 1, pp 5-11

[115] Goldberg, D.E. (1989)

Sizing Populations for Serial and Parallel Genetic Algorithms

Proceedings of the Third International Conference on Genetic Algorithms, pp 20-29

[116] Schaffer, J.D. et al (1989)

A Study of Control Parameters Affecting Online Performacne of Genetic Algorithms for Function Optimisation

Proceedings of the Third International Conference on Genetic Algorithms, pp 51-60

[117] Holland, J.H. (1975)

Adaptation in Natural and Artificial Systems

The University of Michigan Press

[118] Bethke, A.D. (1981)

Genetic Algorithms as Function Optimisers

Ph.D. Thesis, University of Michigan

[119] Grefenstette, J.J. & Baker, J.E. (1989)

How Genetic Algorithms Work : A Critical Look at Implicit Parallelism

Proceedings of the Third International Conference on Genetic Algorithms, pp 12-19

[120] Bradshaw, W.K et al (1995)
Comparison of Three Algorithms for the Control of a Servo Motor
Proceedings of the Ninth World Congress on the Theory of Machines and Mechanisms, Part 2, pp 1355-1359

[121] Zeigler, J.G. & Nichols, N.B. (1942)
Optimal Settings for Automatic Controllers
Transactions of the ASME, Vol 68, Part 8, pp 759-768

[122] De Paor, A.M. (1993)
A Fiftieth Anniversary Celebration of the Zeigler-Nichols PID Controller
International Journal of Electrical Engineering Education, Vol 30, No 4

[123] Donne, M.S., Tilley, D.G. & Richards, W. (1995)
The Use of Multi-Objective Parallel Genetic Algorithms to Aid Fluid Power System Design
Proceedings of the Institution of Mechanical Engineers, Part I , Vol 209, No 1, Journal of Systems and Control Engineering, pp 53-61

A.2 Additional Publications

In addition to the cited references, several other publications were used extensively throughout the duration of the project. These are listed below.

[124] Norton, R.L. (1992)
Design of Machinery
McGraw-Hill

[125] Rzevski, G. (1995)
Mechatronics : Designing Intelligent Machines
Volume I : Perception, Coginition & Execution
Butterworth-Heineman

[126] Johnson, J. & Piction, P. (1995)
Mechatronics : Designing Intelligent Machines
Volume II :Concepts in Artificial Intelligence
Butterworth-Heineman

[127] Shigley, J.E. & Uicker, J.J. (1980)
Theory of Machines & Mechanisms
McGraw-Hill

Appendix B

Quin Systems Tuning Method

The following method is based upon recommendations of Quin Systems for the tuning of a single motor. It is important to realise that this procedure involves trying to set the system into oscillation in order to find an upper limit on the gain parameters. If this is likely to cause any problems, or damage to the system, then this procedure should not be used.

1. Set the proportional gain to a low value, for example 50, and set all other gain terms to zero. The default settings for K_P is 256 and for all other values it is zero. Set the velocity and acceleration with the SV and SA commands to a suitable low value. These values depend upon the resolution of the encoder/resolver.

2. Enable the position control action with the PC command. If the motor immediately runs at high speed in one direction and then stops, giving a "motor position error" message, then the sense of the encoder is reversed. This can be corrected by swapping one pair of encoder signal wires. The exact method for doing this is dependant on the type of amplifier and encoder/resolver fitted. The following stages assume that the system has been correctly set up and is able to control the motor position.

3. Try executing some simple move commands, such as MR 1000. The motor should move as instructed. If at any time the motor starts to vibrate or oscillate beyond the desired move then the gain is already too high. Reduce it by halves until the vibrations stop. The monitor output should show something approximating a trapezoidal or triangular velocity profile for the move.

4. When the motor is following some simple slow move commands correctly, the next stage is to try some cyclic moves. An example command string is;

MR1000/WT128/MR-1000/WT128/RP

This command string sets up a loop where the motor moves 1000 counts positive, pauses for half a second, moves back 1000 counts negative to its start position and then waits a further half a second before repeating the move.

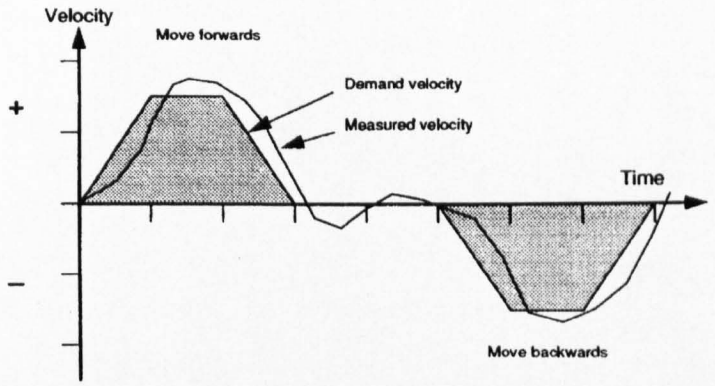

Figure B.1 : Response to a Slow Cyclic Move for a Detuned System

5. Increase the acceleration to a larger value. Set the system to repeatedly execute sudden cyclic moves, with a pause between each move to allow the system to settle.

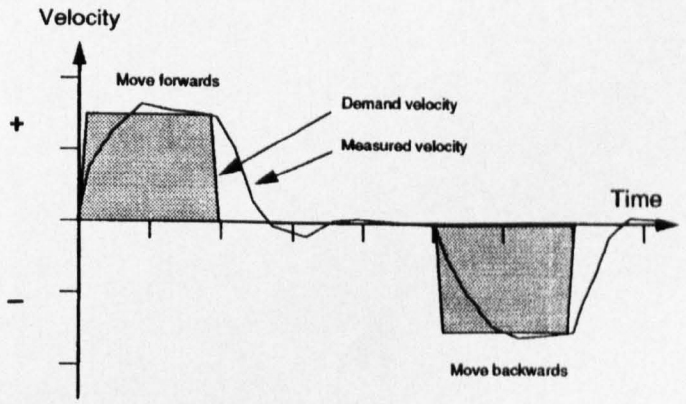

Figure B.2 : Response to a Fast Cyclic Move for a Detuned System

6. While the system is executing this move sequence, slowly increase the proportional gain with the KP command until the actual velocity begins to overshoot the desired velocity at the end of a move. This indicates that the system is beginning to become unstable. It is possible to continue increasing the gain to the point where the oscillation is sustained indefinitely. This is the highest useable value of K_P without making the system totally unstable, although it is of no practical use because of the oscillations.

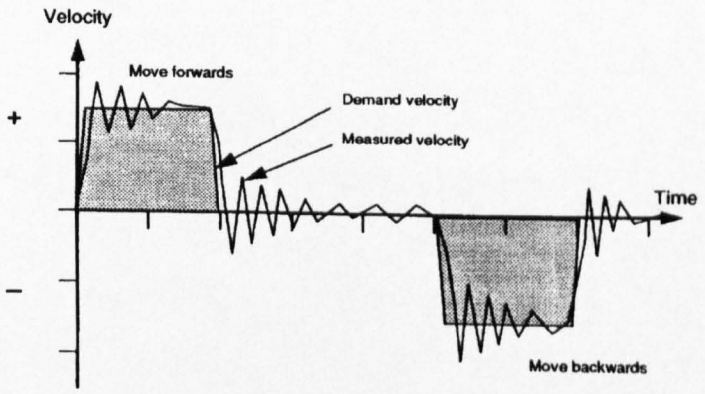

Figure B.3 : Response of an Overtuned Proportional Controller

7. Now increase the velocity feedback gain with the K_V command. Velocity feedback adds damping to the system and should begin to reduce the amplitude of the oscillations. This should be visible on the output display. Continue to increase the K_V value until the oscillation stops and there is little or no ringing at the end of each move. The K_V term can usually be increased to a much higher value than the Kp term. In many systems it is possible to increase K_V to the point where the system is critically damped and the time taken to reach the target position is at a minimum.

On some low load inertia systems, the K_V term may prove ineffective in damping the oscillations and may even make them worse. When this occurs, it is necessary to include an external source of damping such as a tachogenerator. This provides instant velocity feedback to the motor drive and is not subject to the sample time constraints of a digital system.

8. Stop the move sequence with either the AX or ER commands. If it is possible to run the machine at constant speed in one direction, then the K_F feed forward gain may be set up as well. If not, it will have to be set up during normal operation of the machine system. Set the speed to the desired operating speed of the motor. Increase the value of K_F and note that the position error should decrease. K_F may be increased until the position error is approximately zero, at which point the feedforward gain is compensating for the velocity lag present in the system with proportional gain only. The K_F value may be increased further to the point where the motor position is ahead of the demand position if required.

This procedure, although it only describes setting some of the gain terms, is sufficient in many cases to give acceptable performance. However, an acceptable setup for any particular operation may not be ideal for a different operation, so it is useful to experiment with many different moves and profiles to find the best compromise. Clearly, the most important operation for the purpose of tuning motors is the normal operation cycle of the machine. It is possible, however, by using the command sequence facilities on the system to change gain settings automatically, in response to an input signal or according to a set program. This would be used for example, on a robot arm, where the ideal setup depends on the load carried by the arm. This could have great benefit in the hybrid configuration where control parameters could be modulated throughout the machine cycle to obtain truly optimum performance.

Appendix C

Program Code

C.1 Introduction

This appendix contains the prgram code for the Genetic Algorithm developed in this program of research as well as the code for the mechanism synthesis objective function. The code was deveoped using Microsft C/C++ v7.00 and contains several Microsoft (non ANSI) extensions. The code is contained in six files;

1. 5bar_ga.c
2. const.c
3. operator.c
4. util.c
5. userio.c
6. objective.c

C.2 5bar_ga.c

```
/*****************************************************/
/* Five-Bar Mechanism Synthesis Software             */
/* Genetic Algorithm Developed By                    */
/* A.M.Connor        7th June 1996                   */
/* Mechanisms & Machines Research Group              */
/* Liverpool John Moores University                  */
/*****************************************************/

/* Preprocessor directives */

#include <stdio.h>
#include <math.h>
#include <graph.h>
#include <conio.h>
#include <stdlib.h>
#include <time.h>

#include "const.c"
#include "userio.c"
```

```
#include "util.c"
#include "operator.c"
#include "objectiv.c"

void main(void)
{
    int z=0;

    srand((unsigned)time(NULL));

    initialise();
    printf("\a\n\nEnter filename for results output : ");
    scanf("%s",&save);
    generatepop();
    update();
    decode();
    report();
    while(z!=1)
    {
      z=0;
      if(kbhit())
          break;
      generation();
      update();
      decode();
      report();
      maxfit=0.0;     /* Reset generation max fitness */
      ++gen;
      z=termination(&newpop[0].param5,&newpop[0].param6,
                                &newpop[0].param7,&newpop[0].param8);
    }
    printf("\nz=%d\nok",z);
    final_report();
}
```

C.3 const.c

```
/*******************************************************/
/* File CONST.C                                        */
/* Containing the definitions of symbolic constants,   */
/* global variables and function prototypes for        */
/* 5bar_ga.                                            */
/* A.M.Connor    7th March 1996                        */
/*******************************************************/

/* Symbolic constants */

#define CHROM_LENGTH 48
#define POP_SIZE 41
#define CROSS_RATE 0.85
#define MUTATE_RATE 0.05
#define NO_PRECISION_POINTS 24
#define CYCLE 360.0
#define PI 3.141592654
#define STEP 15
#define NO_HARMONICS 5

/* Structure definitions */

struct phenotype{
    int ident;
    int b[CHROM_LENGTH];
```

```
    double param1;
    double param2;
    double param3;
    double param4;
    double param5;
    double param6;
    double param7;
    double param8;
    double fitness;
    double error;
    double harmonics;
    double area;
    int mobility;
    };
typedef struct phenotype solution;

/* Function prototypes */

void initialise(void);
void read_file(void);
void weighting(void);
void generatepop(void);
void update(void);
void decode(void);
void report(void);
double random(void);
int test(double);
void crossover(int,int,int);
void mutation(int,int);
int selection(void);
void generation(void);
void sort(void);
void final_report(void);
double error(double *,double *,double *,double *,double *,
                                    double *,double *,double *,int *);
double angle1(double *,double *,double *,double *,double *,
                                    double *,double *,double *,int *);
double angle2(double *,double *,double *,double *,double *,
                                    double *,double *,double *,int *);
void tracking(double [],double [],double []);
void identical(int string);
void invert(int j);
double fourier(double []);
double area(double [],double *);
int termination(double *,double *,double *,double *);

/* Global variables */

solution oldpop[POP_SIZE],newpop[POP_SIZE];
double g_p_1_x,g_p_1_y,g_p_5_x,g_p_5_y; /* Ground point constraint
origin */
double
x_desired[NO_PRECISION_POINTS],y_desired[NO_PRECISION_POINTS]; /*
Desired curve */
int gen;   /* No. generations */
double sumfit,maxfit,tol,w1,w2,w3,w4;
char save[20];
```

C.4 userio.c

```
/*****************************************************/
/* File USERIO.C                                     */
/* Contains the input/output functions for 5bar_ga   */
/* A.M.Connor    7th March 1996                      */
/*****************************************************/

void initialise(void)
{

/* Sets up the global variables by prompting the user for the
   position of the ground point constraint envelopes, the file name
   of the motion data file and calls the function weighting()to
   find the objective function weights */

    _clearscreen(_GCLEARSCREEN);
    printf("\nGenetic Synthesis of FiveBar Mechanisms");
    printf("\n---------------------------------------");
    printf("\n\nDeveloped by A.M.Connor");
    printf("\nLiverpool John Moores University, 1996");
    printf("\n\nPress any key to start.\n");
    getch();
    printf("\n\nEnter location of local origin for CV crank
                                                constraint envelope");
    printf("\a\nx co-ord : ");
    scanf("%lf",&g_p_1_x);
    printf("\ay co-ord : ");
    scanf("%lf",&g_p_1_y);
    printf("\n\nEnter location of local origin for servo crank
                                                constraint envelope");
    printf("\a\nx co-ord : ");
    scanf("%lf",&g_p_5_x);
    printf("\ay co-ord : ");
    scanf("%lf",&g_p_5_y);
    read_file();
    weighting();
}

void read_file(void)
{

/* Inputs a data file of known length containing the (x,y)
   coordinates which define the desired motion */

    int i;
    char file[20];
    FILE *stream;

    printf("\a\n\nEnter motion data file name (include drive
                                                specifier) : ");
    scanf("%s",&file);
    stream=fopen(file,"r");
    fseek(stream,0L,SEEK_SET);
    if(stream==NULL)
    {
      printf("File does not exist!\nProgram terminated\n");
      exit(0);
    }
    printf("Reading file.....\n");
    for(i=0;i<=NO_PRECISION_POINTS-1;++i)
    {
      fscanf(stream,"%lf",&x_desired[i]);
```

```
        fscanf(stream,"%lf",&y_desired[i]);
    }
    printf("\nData input complete.");
}

void weighting(void)
{
    /* Prompts the user for the weights used in the calculation of
       the objective function value of a solution */

    printf("\nSelect weightings to be used in objective function.");
    printf("\nType '0' to exclude component.");
    printf("\a\nError : ");
    scanf("%lf",&w1);
    printf("\aMobility : ");
    scanf("%lf",&w2);
    printf("\aHarmonics : ");
    scanf("%lf",&w3);
    printf("\aSwept area : ");
    scanf("%lf",&w4);
    printf("\a\nEnter search termination level on curve error : ");
    scanf("%lf",&tol);
}


void report(void)
{
    /* Reports on screen the progress of the synthesis and outputs
       the population data (fitness) to a named file for analysis in
       MATLAB */

    FILE *stream;
    int i;
    double test=0;

    printf("\n\nBest solution for generation %d",gen);
    printf("\n----------------------------------");
    printf("\n\nCV ground point co-ords :
                    (%.2f,%.2f)",newpop[0].param1,newpop[0].param2);
    printf("\nServo ground point co-ords :
                    (%.2f,%.2f)",newpop[0].param3,newpop[0].param4);
    printf("\na12 : %.2f",newpop[0].param5);
    printf("\ta23 : %.2f",newpop[0].param6);
    printf("\na34 : %.2f",newpop[0].param7);
    printf("\ta45 : %.2f",newpop[0].param8);
    printf("\nObjective function value : %f",newpop[0].fitness);
    printf("\nError : %lf\t\tMobility :
                          %d",newpop[0].error,newpop[0].mobility);
    printf("\nHarmonic content : %lf\tSwept area :
                         %lf",newpop[0].harmonics,newpop[0].area);
    printf("\nmaxfit : %f\tsumfit : %lf",maxfit,sumfit);

    stream=fopen(save,"a");
    fprintf(stream,"\n%f",newpop[0].fitness);
    fclose(stream);
}

void final_report(void)
{
    FILE *stream;

    printf("\n\n\nSaving mechanism dimensions.");
    stream=fopen(save,"a");
    fprintf(stream,"\n%lf",newpop[0].param1);
    fprintf(stream,"\n%lf",newpop[0].param2);
    fprintf(stream,"\n%lf",newpop[0].param3);
```

```
    fprintf(stream,"\n%lf",newpop[0].param4);
    fprintf(stream,"\n%lf",newpop[0].param5);
    fprintf(stream,"\n%lf",newpop[0].param6);
    fprintf(stream,"\n%lf",newpop[0].param7);
    fprintf(stream,"\n%lf",newpop[0].param8);
}
```

C.5 operator.c

```
/*******************************************************/
/* File OPERATOR.C                                     */
/* Source code for genetic operators for use in        */
/* 5bar_ga synthesis software                          */
/* A.M.Connor    7th March 1996                        */
/*******************************************************/

void generation(void)
{
    /*  The generation function coordinates the action of the
        genetic operators to form a new population. The main
        operators are selection (reproduction), crossover and
        mutation. An inversion operator is called if a child
        solution string is identical to an existing string */

    int mate1,mate2,j;

    oldpop[0]=newpop[0];     /* Elitism */
    for(j=1;j<=POP_SIZE-1;j+=2)
    {
      mate1=selection();
      mate2=selection();
      crossover(mate1,mate2,j);
      identical(j);
      identical(j+1);
    }
}

int selection(void)
{
    /* The selection operator is analgous to reproduction in natural
       systems. Two parents are choosen from the population by a
       roulette wheel selection method. This ensures that the most
       fit solutions are selected. Individual fitnesses are
       normalised wrt the sum of the generation fitnesses to act as
       a simple scaling routine */

    double partsum,randcheck;
    int j;

    partsum=0.0;
    randcheck=random();
    randcheck*=(POP_SIZE-1)/2.0;
    for(j=0;j<=POP_SIZE-1;++j)
    {
      partsum+=1.0-(newpop[j].fitness/sumfit);
      if(partsum>=randcheck)
          break;
    }
    return j;
}
```

```
void crossover(int mate1,int mate2,int j)
{
    /* Once two parents have been choosen, two child strings are
       formed by crossover and mutation. The mutation operator is
       embedded in the crossover function. During crossover, a cross
       site is randomly selected and the children formed by a
       bitwise exchange of data from the parent strings. */

    int i,cross_site,cross;

    cross=test(CROSS_RATE);
    if(cross==0)
    {
      for(i=0;i<=CHROM_LENGTH-1;++i)
      {
          oldpop[j].b[i]=newpop[mate1].b[i];
          mutation(j,i);
          oldpop[j+1].b[i]=newpop[mate2].b[i];
          mutation(j+1,i);
      }
    }
    else
    {
      cross_site=rand()%CHROM_LENGTH;
      for(i=0;i<=cross_site-1;++i)
      {
          oldpop[j].b[i]=newpop[mate1].b[i];
          mutation(j,i);
          oldpop[j+1].b[i]=newpop[mate2].b[i];
          mutation(j+1,i);
      }
      for(i=cross_site;i<=CHROM_LENGTH-1;++i)
      {
          oldpop[j].b[i]=newpop[mate2].b[i];
          mutation(j,i);
          oldpop[j+1].b[i]=newpop[mate1].b[i];
          mutation(j+1,i);
      }
    }
}

void mutation(int j,int i)
{
    /* The mutation function is called each time data is transferred
       between two strings during crossover. If the probabilty event
       is true, then mutation occurs. In the binary notation
       adopted, mutation is a simple boolean NOT function (ie, a 1
       becomes a 0 or a 0 becomes a 1). Mutation prevents stagnation
       in genetic search */

    int mutate;

    mutate=test(MUTATE_RATE);
    if(mutate==1)
      oldpop[j].b[i]=!oldpop[j].b[i];
}



void invert(int j)
{
    /* The inversion operator is called when ever a string is
       created that is identical to a string that already exists in
       the population. Inversion acts on a single string and
       reorders a section of the string between two inversion sites.
       There is nocheck for mutation during the operation. */
```

```
    int min,max,i,inv_site1,inv_site2;
    solution temp;

    inv_site1=rand()%CHROM_LENGTH;
    inv_site2=rand()%CHROM_LENGTH;
    if(inv_site2==inv_site1)
      return;
    if(inv_site1>inv_site2)
    {
      max=inv_site1;
      min=inv_site2;
    }
    else
    {
      max=inv_site2;
      min=inv_site1;
    }
    temp=oldpop[j];
    for(i=min;i<=max;++i)
    {
      oldpop[j].b[i]=temp.b[max-i];
    }
}
```

C.6 util.c

```
/*********************************************************/
/* File UTIL.C                                           */
/* Contains a variety of random number based             */
/* functions, population functions and test              */
/* functions for 5bar_ga.                                */
/* A.M.Connor    7th March 1996                          */
/*********************************************************/

void generatepop(void)
{
    /* Generates a set of random binary strings which form the
       initial population of solutions. */

    int i,j,k;

    for(i=0;i<=POP_SIZE-1;++i)
    {
      for(j=0;j<=CHROM_LENGTH-1;++j)
      {
          k=test(0.5);
          oldpop[i].b[j]=k;
          mutation(i,j);
      }
      oldpop[i].ident=i;
      oldpop[i].fitness=0.0;
    }
}

void update(void)
{
    /* Update a newly generated population as working population */

    int i;

    for(i=0;i<=POP_SIZE-1;++i)
```

```
    {
      newpop[i]=oldpop[i];
    }
}

void sort(void)
{
    /* The sort function uses a bubble sort algorithm to sort a
       newly generated population by increasing fitness value. */

    int i,j;
    solution temp;

    for(i=0;i<=POP_SIZE-1;++i)
    {
      for(j=POP_SIZE-1;j>=i;--j)
      {
          if(newpop[j-1].fitness>newpop[j].fitness)
          {
            temp=newpop[j-1];
            newpop[j-1]=newpop[j];
            newpop[j]=temp;
          }
      }
    }
    for(i=0;i<=POP_SIZE-1;++i)
    {
      newpop[i].ident=i;
    }
}

double random(void)
{
    /* The random function fills a vector with a set of random
       numbers and returns a randomly selected member that is
       normalised with respect to RAND_MAX so that it's value is a
       real number between 0 and 1. */

    double j;


    j=rand();
    j/=RAND_MAX;
    return j;
}

int test(double x)
{
    /* The test function is used to test whether a probability event
       occurs and works by comparing a random number from the random
       function against the probability of an event occuring. */

    int testval;
    double y;

    if(x==1.0)
      return 1;
    y=random();
    if(y>x)
      testval=0;
    else
      testval=1;
    return testval;
}
```

```
void identical(int string)
{
    /* The identical function is called from the generation function
       and tests to see if an identical string already exists in the
       new population. If an identical string exists then it is
       inverted to forcibly maintain the diversity of the
       population. */

    int i,k,score=0,tempscore;

    for(i=0;i<=string-1;++i)
    {
      tempscore=0;
      for(k=0;k<=CHROM_LENGTH-1;++k)
      {
          if(oldpop[i].b[k]==oldpop[string].b[k])
            ++tempscore;
      }
      if(tempscore>score)
          score=tempscore;
    }
    if(score>=CHROM_LENGTH)
      invert(string);
}

int termination(double *p,double *q,double *r,double *s)
{
    if(((*p/(*q))>0.25)&&((*p/(*q))<2.0))
    {
      if(((*p/(*r))>0.25)&&((*p/(*r))<2.0))
      {
          if(((*p/(*s))>0.25)&&((*p/(*s))<2.0))
          {
            if(newpop[0].mobility==0)
            {
                if(newpop[0].error<tol)
                {
                  return 1;
                }
            }
          }
      }
    }
    return 0;
}
```

C.7 objectiv.c

```
/*******************************************************/
/* File OBJECTIV.C                                     */
/* Contains the decoding and evaluation functions      */
/* for 5bar_ga                                         */
/* A.M.Connor      7th June 1996                       */
/*******************************************************/

void decode(void)
{
    /* The decode function decodes a population of binary strings
       into the actual parameter values. The link lengths are
       constrained to values between 10 and 286 units and the ground
       point positions are expressed as local (x,y) co-ordinates in
```

```
   a constraint envelope 16 by 16 units. The global co-ordinates
   of origin are specified by the user. The decoded values are
   stored in the solution structure. Once all the solutions have
   been decoded and the fitnesses evaluated, the population is
   sorted by fitness value in the sort() function. */

int i,j,power;
double sum1,sum2,sum3,sum4,sum5,sum6,sum7,sum8;

sumfit=0.0;

for(i=0;i<=POP_SIZE-1;++i)
{
  power=1;
  sum1=0.0;
  for(j=0;j<=3;++j)
  {
      if(newpop[i].b[j]==1)
        sum1+=power;
      power*=2;
  }
  sum1+=g_p_1_x;
  power=1;
  sum2=0.0;
  for(j=4;j<=7;++j)
  {
      if(newpop[i].b[j]==1)
        sum2+=power;
      power*=2;
  }
  sum2+=g_p_1_y;
  power=1;
  sum3=0.0;
  for(j=8;j<=11;++j)
  {
      if(newpop[i].b[j]==1)
        sum3+=power;
      power*=2;
  }
  sum3+=g_p_5_x;
  power=1;
  sum4=0.0;
  for(j=12;j<=15;++j)
  {
      if(newpop[i].b[j]==1)
        sum4+=power;
      power*=2;
  }
  sum4+=g_p_5_y;
  power=1;
  sum5=10.0;     /* link length minimum constraint */
  for(j=16;j<=23;++j)
  {
      if(newpop[i].b[j]==1)
        sum5+=power;
      power*=2;
  }
  power=1;
  sum6=10.0;
  for(j=24;j<=31;++j)
  {
      if(newpop[i].b[j]==1)
        sum6+=power;
      power*=2;
  }
  power=1;
```

```
        sum7=10.0;
        for(j=32;j<=39;++j)
        {
            if(newpop[i].b[j]==1)
              sum7+=power;
            power*=2;
        }
        power=1;
        sum8=10.0;
        for(j=40;j<=47;++j)
        {
            if(newpop[i].b[j]==1)
              sum8+=power;
            power*=2;
        }
        /*printf("\nFinished decode");*/
        newpop[i].param1=sum1;
        newpop[i].param2=sum2;
        newpop[i].param3=sum3;
        newpop[i].param4=sum4;
        newpop[i].param5=sum5;
        newpop[i].param6=sum6;
        newpop[i].param7=sum7;
        newpop[i].param8=sum8;
        newpop[i].fitness=error(&sum1,&sum2,&sum3,&sum4,&sum5,&sum6,
                                                    &sum7,&sum8,&i);
        if(newpop[i].fitness>maxfit)
            maxfit=newpop[i].fitness;
        sumfit+=newpop[i].fitness;
      }
      sort();
}

double error(double *x1,double *y1,double *x5,double *y5,
                    double *p,double *q,double *r,double *s,int *z)
{

/* The error() function calculates the objective function based on
   the weightings selected by the user */

      double theta1,theta2,theta3;
      double t,input,x_actual,y_actual;
      double x_coord,y_coord;
      double err,total_error=0;
      int i=(-1),mobility=0;
      double servo1[NO_PRECISION_POINTS]={0},
       servo2[NO_PRECISION_POINTS]={0},servo3[NO_PRECISION_POINTS]={0};
      double harmonics,obj,swept;

      t=sqrt(pow((*x5-*x1),2)+pow((*y5-*y1),2));
      if(t==0.0)
        theta1=0.0;
      else
        theta1=asin((*y5-*y1)/t);

      for(i=0;i<=NO_PRECISION_POINTS-1;++i)
      {
        input=i*15.0;
        theta2=(input*2.0*PI)/CYCLE;
        x_coord=*p*cos(theta2);
        y_coord=*p*sin(theta2);
        theta3=atan2((y_desired[i]-(y_coord+*y1)),
                                        (x_desired[i]-(x_coord+*x1)));
        y_actual=*q*sin(theta3)+y_coord+*y1;
        x_actual=*q*cos(theta3)+x_coord+*x1;
        err=sqrt(pow((x_actual-x_desired[i]),2)
```

```
                                        +pow((y_actual-y_desired[i]),2));
      total_error+=err;
      servo1[i]=angle1(&theta1,&theta2,&theta3,&*p,&*q,&*r,&*s,&t,
                                                        &mobility);
      servo2[i]=angle2(&theta1,&theta2,&theta3,&*p,&*q,&*r,&*s,&t,
                                                        &mobility);
    }
    tracking(servo1,servo2,servo3);
    harmonics=fourier(servo3);
    swept=area(servo3,&*s);
    newpop[*z].harmonics=harmonics;
    newpop[*z].area=swept;
    newpop[*z].error=total_error;
    newpop[*z].mobility=mobility;
    obj=((w1*pow(total_error,3))+(w2*pow(mobility,5)))
                        +(w3*pow(harmonics,0.3))+(w4*pow(swept,0.75));
    return obj;
}


double angle1(double *theta1, double *theta2, double *theta3,
   double *p,double *q,double *r,double *s, double *t,int *mobility)
{
  /* Calculates one closure angle for servo motor based upon actual
     position of end effector. Function angle2() caluates second */

    double k,acoeff,bcoeff,ccoeff;
    double servo1;
    k=(*r**r)-(*p**p)-(*q**q)-(*s**s)-(*t**t)
           -(2**p**q*cos(*theta2-*theta3))
                 +(2**p**t*cos(*theta2-*theta1))
                       +(2**q**t*cos(*theta3-*theta1));
    acoeff=(2**p**s*cos(*theta2))+(2**q**s*cos(*theta3))
                                          -(2**s**t*cos(*theta1))-k;
    bcoeff=((4**s**t*sin(*theta1))-(4**p**s*sin(*theta2))
                                           -(4**q**s*sin(*theta3)));
    ccoeff=(2**s**t*cos(*theta1))-(2**p**s*cos(*theta2))
                                          -(2**q**s*cos(*theta3))-k;

    if((bcoeff*bcoeff)-(4*acoeff*ccoeff)>0.0)
    {
      servo1=2.0*atan(((bcoeff*-1.0)-sqrt((bcoeff*bcoeff)
                                    -(4*acoeff*ccoeff)))/(2.0*acoeff));
      if(fabs(servo1)<0.000001)
          servo1=0.0;
      return servo1;
    }
    else
    {
      *mobility+=1;
      return 0;
    }
}

double angle2(double *theta1, double *theta2, double *theta3,
   double *p,double *q,double *r,double *s, double *t,int *mobility)
{
    double k,acoeff,bcoeff,ccoeff;
    double servo2;

    k=(*r**r)-(*p**p)-(*q**q)-(*s**s)-(*t**t)
           -(2**p**q*cos(*theta2-*theta3))
                 +(2**p**t*cos(*theta2-*theta1))
                       +(2**q**t*cos(*theta3-*theta1));
    acoeff=(2**p**s*cos(*theta2))+(2**q**s*cos(*theta3))
                                          -(2**s**t*cos(*theta1))-k;
```

```
    bcoeff=((4**s**t*sin(*theta1))-(4**p**s*sin(*theta2))
                                          -(4**q**s*sin(*theta3)));
    ccoeff=(2**s**t*cos(*theta1))-(2**p**s*cos(*theta2))
                                         -(2**q**s*cos(*theta3))-k;

    if((bcoeff*bcoeff)-(4*acoeff*ccoeff)>0.0)
    {
      servo2=2.0*atan(((bcoeff*-1.0)+sqrt((bcoeff*bcoeff)
                              -(4*acoeff*ccoeff)))/(2.0*acoeff));
      if(fabs(servo2)<0.000001)
          servo2=0.0;
      return servo2;
    }
    else
    {
      *mobility+=1;
      return 0;
    }
}

void tracking(double s1[NO_PRECISION_POINTS],
      double s2[NO_PRECISION_POINTS],double s3[NO_PRECISION_POINTS])
{
    /* Closure tracking algorithm chooses between two closures
       and updates array s3 {servo3} as selected closure for
       each step as relative increments */

    double last_vel,next_vel_1,next_vel_2,rel=0.0;
    int i,alpha,beta,gamma;

    s3[0]=s1[0]-s1[0];
    if((fabs(s2[1]-rel))<(fabs(s1[1]-rel)))
      s3[1]=s2[1]-s1[0]; /* choose first point on mag. of disp. */
    else
      s3[1]=s1[1]-s1[0];
    for(i=2;i<=NO_PRECISION_POINTS-1;++i)
    {
      last_vel=atan((s3[i-1]-s3[i-2])/(1.0*STEP));
      if(last_vel<0.0)
          alpha=1;
      else
          alpha=0;
      next_vel_1=atan((s1[i]-s3[i-1])/(1.0*STEP));
      if(next_vel_1<0.0)
          beta=1;
      else
          beta=0;
      next_vel_2=atan((s2[i]-s3[i-1])/(1.0*STEP));
      if(next_vel_2<0.0)
          gamma=1;
      else
          gamma=0;
      getch();*/
      if(beta==gamma)
      {
      if(fabs(next_vel_2-last_vel)<fabs(next_vel_1-last_vel))
          s3[i]=s2[i]-s1[0];
      else
          s3[i]=s1[i]-s1[0];
      }
      if(beta!=gamma)
      {

if((gamma==alpha)&&(fabs(next_vel_2)<2.0*fabs(next_vel_1)))
          s3[i]=s2[i]-s1[0];
      else
```

```
            s3[i]=s1[i]-s1[0];
        }
    }
}

double fourier(double angles[NO_PRECISION_POINTS])
{
  /* Calculates harmonics of servo motor displacement for use in
     objective function calculation */

    double a[NO_HARMONICS]={0.0},b[NO_HARMONICS]={0.0};
    double data[NO_PRECISION_POINTS];
    double obj=0;
    int i,j,k;
    for(i=0;i<=NO_PRECISION_POINTS-1;++i)
    {
      data[i]=(CYCLE/2.0*PI)*angles[i];
      a[0]+=data[i];
    }
    a[0]=a[0]/(NO_PRECISION_POINTS/2.0);

    for(j=1;j<=NO_HARMONICS-1;++j)
    {
      for(k=0;k<=NO_PRECISION_POINTS-1;++k)
      {
          a[j]+=data[k]*(cos(STEP*k*j*2.0*PI/CYCLE));
          b[j]+=data[k]*(sin(STEP*k*j*2.0*PI/CYCLE));
      }
      a[j]=a[j]/(NO_PRECISION_POINTS/2.0);
      b[j]=b[j]/(NO_PRECISION_POINTS/2.0);
    }
    for(j=0;j<=NO_HARMONICS-1;++j)
    {
      obj+=pow(sqrt((a[j]*a[j])+(b[j]*b[j])),j+1);
    }
    return obj;
}


double area(double angles[NO_PRECISION_POINTS],double *link)
{
  /* Calcluales swept area term */

    int i;
    double data[NO_PRECISION_POINTS],sum=0.0,rms,value;

    for(i=0;i<=NO_PRECISION_POINTS-1;++i)
    {
      data[i]=(CYCLE/2.0*PI)*angles[i];
      sum+=pow(data[i],2);
    }
    rms=sqrt(sum);
    value=rms**link;
    return value;
}
```