Evolutionary Design Automation for Control Systems with Practical Constraints

by

Wenyuan Feng

A thesis of the requirement for

The degree of

Doctor of Philosophy

July 2000

Department of Electronics and Electrical Engineering

University of Glasgow

Glasgow G12 8LT

United Kingdom

Dedications

To:

Father	Feng Shichang
Mother	Wang changming
Wife	Li Yi

For their supports throughout the course in UK

Acknowledgements

I would like to express my sincere gratitude to my supervisor, Dr. Yun Li for his guidance, encouragement, inspiration and patience throughout the course of my PhD work. In particular, his vision in, and commitments to, the research and his sharing of knowledge and time that has made the research a success.

Special acknowledgement and appreciation are due to Mr. Tom O'Hara for his enthusiastic assistance during my laboratory work. Great thanks should be given to Dr Wenhua Chen, since he contributed so much interesting suggestion and comment.

I would like to thank my wife, my daughter, my father, my mother, my sister, my bother-in-law, my father-in-law and mother-in-law. Without their love and moral support, nothing would have been achieved.

I am also grateful to the University of Glasgow and CVCP for their financial support in the form of a Postgraduate Scholarship and an Overseas Research Scheme Award, respectively.

Finally, I want to thank all the friends in Evolutionary Computing and Control group for their friendliness, help and companionship. In particular, Dr. Kay Chen Tan, Dr. Euan McCookin, Dr. Jun Chen, Lipton Chan, Yingpeng Zhao, Cindy Goh and Gregory Chong.

Abstract

The aim of this work is to explore the potential and to enhance the capability of evolutionary computation in the development of novel and advanced methodologies that enable control system structural optimisation and design automation for practical applications.

Current design and optimisation methods adopted in control systems engineering are in essence based upon conventional numerical techniques that require derivative information of performance indices. These techniques lack robustness in solving practical engineering problems, which are often of a multi-dimensional, multi-modal nature. Using those techniques can often achieve neither global nor structural optimisation. In contrast, evolutionary mechanism learning tools have the ability to search in a multi-dimensional, multi-modal space, but they can not approach a local optimum as a conventional calculus-based method. The first objective of this research is to develop a reliable and effective evolutionary algorithm for engineering applications.

In this thesis, a globally optimal evolutionary methodology and environment for control system structuring and design automation is developed, which requires no design indices to be differentiable. This is based on the development of a hybridised GA search engine, whose local tuning is tremendously enhanced by the incorporation of Hill-Climbing (HC), Simulated Annealing (SA) and Simplex techniques to improve the performance in search and design. A Lamarckian inheritance technique is also developed to improve crossover and mutation operations in GAs. Benchmark tests have shown that the enhanced hybrid GA is accurate, and reliable.

Based on this search engine and optimisation core, a linear and nonlinear control system design automation suite is developed in a Java based platform-independent format, which

can be readily available for design and design collaboration over corporate Intranets and the Internet. Since it has also made cost function unnecessary to be differentiable, hybridised indices combining time and frequency domain measurement and accommodating practical constraints can now be incorporated in the design. Such type of novel indices are proposed in the thesis and incorporated in the design suite.

The Proportional plus Integral plus Derivative (PID) controller is very popular in real world control applications. The development of new PID tuning rules remains an area of active research. Many researchers, such as Åström and Hägglund, Ho, Zhuang and Atherton, have suggested many methods. However, their methods still suffer from poor load disturbance rejection, poor stability or shutting of the derivative control etc. In this thesis, Systematic and batch optimisation of PID controllers to meet practical requirements is achieved using the developed design automation suite. A novel cost function is designed to take disturbance rejection, stability in terms of gain and phase margins and other specifications into account in the same time. Comparisons made with Ho's method confirm that the derivative action can play an important role to improve load disturbance rejection yet maintaining the same stability margins. Comparisons made with Åström's method confirm that the results from this thesis are superior not only in load disturbance rejection but also in terms of stability margins.

Further robustness issues are addressed by extending the PID structure to a free form transfer function. This is realised by achieving design automation. Quantitative Feedback Theory (QFT) method offers a direct frequency-domain design technique for uncertain plants, which can deal non-conservatively with different types of uncertainty models and specifications. QFT design problems are often multi-modal and multi-dimensional, where loop shaping is the most challenging part. Global solutions can hardly be obtained using analytical and convex or linear programming techniques. In addition, these types of conventional methods

often impose unrealistic or unpractical assumptions and often lead to very conservative designs. In this thesis, GA-based automatic loop shaping for QFT controllers suggested by the Research Group is being furthered. A new index is developed for the design which can describe stability, load rejection and reduction of high frequency gains, which has not been achieved with existing methods. The corresponding prefilter can also be systematically designed if tracking is one of the specifications. The results from the evolutionary computing based design automation suite show that the evolutionary technique is much better than numerical methods and manual designs, i.e., 'high frequency gain' and controller order have been significantly reduced. Time domain simulations show that the designed QFT controller combined with the corresponding prefilter performs more satisfactorily.

For nonlinear plants, the PID structuring and design strategy is extended to a nonlinear format. This is structured as a building block based on artificial neural networks. The automation design environment is employed to optimise the neurocontroller. Here, *special learning* is employed in the design of a feedforward path neurocontroller, in which the network can be trained from a plant model directly. The automated design suite also facilitates the new controller design directly from plant step response data without a model, where convolution is used. In order to arrive at the simplest structure of a network, growth training method is developed for the design suite. Through three applications, it is found that if there is a rate limiter in a practical control loop, the automatically designed neurocontroller outperforms an optimised linear controller; under amplitude limit, the linear controllers achieved from LQR are outperformed by the neurocontroller; and the neurocontroller can be trained to cancel steady state errors for different operating points of a nonlinear double tank system. They show the power of such feedforward path nonlinear controllers, whose designs are only enabled by the evolutionary computing base design suite.

List of Publications

1. Brune, T., Chew, K. G., Feng, W., *et al.* (1998). Evolving Generic Feedforward Path Nonlinear Controllers from Open-loop Plant Response Data, Asia and Pacific Conference for Control and Measurement. Dunhuang, China, 217-222.

2. Li, Y., Feng, W., Tan, K. C., *et al.* (1998). PIDeasy™ and automated generation of optimal PID controllers, *Asia and Pacific Conference for Control and Measurement.* Dunhuang, China, 1998, 29-34.

3. Feng, W., Brune, T., and Chan, L. (1998). Benchmarks for Testing Evolutionary Algorithms, *Asia and Pacific Conference for Control and Measurement*, Dunhuang, China, 134-139.

4. Chen, W., Ballance, D., Feng, W., and Li, Y. (1999). Genetic Algorithm Enabled Computer-Automated Design of QFT Control Systems, *Proceeding of the IEEE International Symposium on Computer-Aided Control System Design*, Hawai'i, USA, 492-497.

5. Feng, W., and Li, Y. (1999). Performance Indices in Evolutionary CACSD Automation with Application to Batch PID Generation, *Proceeding of the IEEE International Symposium on Computer-Aided Control System Design*, Hawai'i, USA, 486-491.

6. Feng, W., Li, Y., Chen, W. and Ballance, D. J.: Automating QFT Control System Design for Industrial Applications, in preparation.

7. Feng, W., and Li, Y.: Batch PID Tuning for Optimal Load Disturbance Rejection, Sensitivity and Stability Margins, in preparation.

8. Feng, W., and Li, Y.: Feedforward-Path Nonlinear PD Controller Optimisation with Practical Constraints, submitted.

Table of Contents

List of Figures

List of Tables

Chapter 1 Introduction

1.1 PID, QFT and Neural Control Systems

The Proportional plus Integral plus Derivative (PID) controller is regarded as the "bread and butter" of control engineering (Levine, 1996). Such a 'three term' controller offers the ability to eliminate steady-state offsets of type 0 systems through its integral action and the ability to anticipate changes and to improve transient and stability through its derivative action. PID control is thus found sufficient for many control applications, such as process control, particularly when process dynamics are benign and performance requirements are modest (Levine, 1996; Åström and Hägglund, 1995). Nowadays, PID modules are manufactured by the hundred of thousands yearly for stand-alone applications (Levine, 1996; Åström and Hägglund, 1995; Seborg *et al.*, 1989). PID control also forms an important ingredient of distributed control systems. For example, in process control, more than 95% control loops are of the PID or PI type. Combined with logic, sequential machines, selectors, and simple function blocks, PID controllers are used to build complicated automation systems for use in energy production, transportation and manufacturing (Levine, 1996).

Although a PID controller shapes the overall frequency response, the proportional, integral and derivative actions offer limited capability in robust loop shaping for uncertain plants. Using an unrestricted transfer function, Quantitative Feedback Theory (QFT) provides a more sophisticated tool for robust loop shaping uncertain plants (Horowitz, 1972; Chait, 1997), where plant dynamics may be described conveniently by its frequency response either as a fixed model, or as a model with parametric, non-parametric or mixed uncertainties. Similar to PID, QFT provides a familiar Bode-Nyquist margin based classical approach. An advantage of QFT over other design methods, such as H_∞ and Linear Quadratic Gaussian

(LQG) or Linear Quadratic Regulation (LQR) optimal control, is its ability to deal non-conservatively with different types of uncertainty models and specifications (Horowitz, 1972; Chait, 1997). This is achieved by translating closed-loop performance specifications into QFT bounds at a set of frequency points. These bounds, typically displayed on a Nichols chart, serve as a guide to shaping the nominal loop, so that in a QFT design, one only needs to shape the nominal loop to satisfy those bounds. This method has already found applications in robust flight control systems design (Keating *et. al.*, 1997; and Wu *et al.*, 1998) and active noise control in duct (Chai *et al.*, 1997). It now attracts practising engineers in robust control system design and receives more attention in research (Levine, 1996).

Another application-driven and rapidly expanding area of research is neurocontrol (Rogers and Li, 1993; Mei *et al.*,1998; Hrycej, 1994). Neural networks provide an innovative method to expand the PID idea to nonlinear control, although this has not been achieved widely elsewhere. Nevertheless, neurocontrol has shown incontestable success in solving practical control problems and forms one of the most significant applications of neural networks, because (Hrycej, 1997)

(i) Applications such as filtering coincide well with linear solutions and approximations and are useful to controller synthesis;

(ii) Applications such as pattern recognition and classification relate to both nonlinear problems and provide well-established special nonlinear algorithms that are useful to system identification;

(iii) Almost all the difficult control problems are nonlinear, but few other nonlinear control design approaches are universally applicable in practice.

Hence, neurocontrol also forms a major part of this thesis.

1.2 Problems in Designing PID, QFT and Neural Control Systems

Despite its popularity, PID control is not always designed or used in the best way. In practice, the controllers are often poorly tuned (Levine, 1996). It thus becomes quite common that the derivative action is not used. Reasons are that it is difficult to tune the three coefficients simultaneously, and that the derivative action does not always stabilise the system (in contrast to the common conception) (Li *et al.*, 1997). Therefore, in the past half-century, significant effort has been made in tuning of PID controllers. Among those, the Ziegler and Nichols (1942) methods are the most popular and benchmarked. Because they give too poor damping to the closed-loop system (Åström, 1991), many methods have been developed to improve them. The CHR method described by Chien *et al.* (1952) is a modification of the Ziegler-Nichols method. An analytical tuning method was first proposed by Newton, Jr. *et al.* (1957), followed by the λ tuning method by Dahlin (1968) and Higham (1968). Tuning techniques developed by Smith and Murill (1966), Pemberton (1972), Hwang and Chang (1987) are also based on the analytical approach. Then optimisation methods to design PID controllers have been developed (Rovira *et al.*, 1969; Lopez *et al.*, 1969; and Zhuang and Atherton, 1993). Zhuang and Atherton's tuning rules are also used in more recent papers (Majhi and Atherton, 1999; Atherton 2000). However, there are still defects in these methods, as Åström *et al.* (1993) pointed out. There is too much transient in the designs given by CHR method and the load disturbance methods by Zhuang and Atherton (1995). Further, analytical methods and the set-point following methods by Zhuang and Atherton (1995) may lead to pole-zero cancellation.

Since set-point weighting can improve set-point following (Hang *et al.*, 1991), the problem is simplified as that of achieving the best load disturbance rejection under given stability margins. To tackle the problem, Ho *et al.*, (1995) used the gain and phase margins as

specification to design PI controller for the widely used first order plus dead time system (FOPDT). Åström and Hägglund (1995) presented their designs using maximum sensitivity as specification. These designs have improved the performances largely, but there is still room to make the tuning better, because Ho's method does not include the derivative action and Åström and Hägglund's method does not work when the normalised delay is more than 0.7 (Åström and Hägglund, 1995). There are two problems with these PID solutions:

(i) These methods cannot take time domain and frequency domain requirements into account at the same time;

(ii) There exists no global optimality in these methods.

Optimisation on load disturbance rejection of PID controllers can be regarded as a loop shaping method. This means that under the frequency domain limits (maximum sensitivity, gain and phase margins), Integral of Time weighted by Absolute Error (ITAE) for load disturbance rejection is minimised. PID designs have now addressed gain and phase margins and maximum sensitivity for robust control, but their capability is limited by the PID structure. It is because that the PID control just has 'three term' to adjust and may too simple to control a complicated and uncertain process.

Robust loop shaping can be achieved explicitly by QFT. In QFT design, however, the loop shaping part can be challenging. Many methods have been proposed to solve this problem. Those include, for example, Bode integrals in an iterative approach to loop shaping by Horowitz and Gera (1980), and Ballance and Gawthrop (1991); Thompson and Nwokah's analytical approach (1994); an automatic technique by Chait (1997), which overcomes the non-convexity of the bounds on the open-loop transmission; and linear programming based optimisation approach to automatic loop shaping by Bryant and Halikias (1995) and Chait (1999). The existing approaches have their merits and deficiencies. Since the QFT design problem is often multi-dimensional and multi-modal, global solutions can hardly be obtained

using analytical and convex or linear programming techniques. Further, existing methods are often mathematically oriented and can impose unpractical or unrealistic assumptions. These often lead to very conservative designs, and appear engineer-unfriendly, losing its original attractiveness.

Not that, although for both PID and QFT design, the ideas originated from loop shaping, there exist the following differences:

(i) A PID controller is a fixed structure controller. A QFT controller can be of a high order, although a lower order is preferred;

(ii) In the loop shaping process, the PID controller often pursues the best load disturbance rejection in time domain under stability margins in frequency domain, but the QFT controller pursue the lowest high frequency gain under stability, disturbance and tracking requirements, which are generalised in frequency domain.

These differences call for different treatment in the methodologies developed in this thesis.

Similarly, for neurocontrol system design, different treatment is also required. Since the traditional frequency methods do not apply to the nonlinear system, time domain optimisation is adopted. There exist some problems to be solved for neurocontrol. One of these is the optimisation of the controller to be extended in the feedforward path as in the same way as a conventional controller. Optimisation with back-propagation suffers from problems of local optimality, requirement of differentiable performance index and difficulty in structure optimisation. Another problem is how to achieve a simple network structure. These problems are to be addressed in this thesis.

In general, if a plant is linear or has a good linearised model, a linear controller such as PID or QFT controller, can be used to achieve good results (Schultz *et al.*, 1997), if the controller

can be designed optimally. When the plant involves with significant nonlinearities, the nonlinear building block based neurocontroller may be used in place of a PID or QFT controller. This thesis will address present problems and difficulties associated with designing these controllers for practical applications, where hard nonlinearities are common.

1.3 Evolutionary Methodology for Control System Structuring and Global Optimisation

Based on Charles Darwin's biological observations, the means of natural selection and the principle of *survival-of the fittest* have led to today's success in evolutionary computation. Evolutionary Algorithms (EAs) such as Genetic Algorithms (GAs), Genetic Programming (GP), Evolutionary Strategy (ES) and related life strategies have been developed upon the synthesis of natural evolution. They form the paradigm of evolutionary computation and have been found particularly effective in searching poorly understood, irregular and complex spaces for optimisation and machine learning (Fogel, 1995; Goldberg 1989; Holland, 1975; Michalewicz, 1994). Unlike conventional gradient-guided search techniques, which are *a-priori*, EAs are *a-posteriori* and require no derivative information at the search points. These algorithms are probabilistic in nature and, based on *a-posteriori* information obtained by computerised trial-and-error, require no direct guidance and thus no stringent conditions on the cost functions (Fogel, 1995; Michalewicz, 1994). Therefore, the index function can be constructed in a way that satisfies the need of engineering systems most and not the need of analytical or numerical tools to be employed.

EAs exhibit global search capability by simultaneously evaluating performances at multiple points in the solution space. Supported by the Schema Theory (Goldberg, 1989a; Holland, 1975), it has been shown that evolutionary algorithms offer an exponentially reduced search

time of the order of $O(n^m)$, $m < \infty$, compared with exhaustive search, which requires a total evaluation time of $O(p^n)$, where n is the number of parameters to be optimised in the search and p is the number of possible choices of each parameters. EAs can handle multiple objectives (MO) without the need to define a composite scalar objective function (Goldberg, 1989a). The multiple search nature of reproductive and evolving population indicates that EAs are a natural parallel paradigm (Goldberg, 1989a; Li *et al.*, 1997). Other features of EAs include robustness of search, capability to incorporate a-priori knowledge and adaptability (Goldberg, 1989; Li, 1999; Michalewicz, 1994). These non-deterministic algorithms could become even more reliable and accurate if interactive fine-tuning, such as simplex tuning, is incorporated (Feng *et al.*, 1998). The evolution process can also be speeded up several times when existing design experience is included in the initial design 'database' for intelligent design-reuse (Ng, 1995).

To summarise, EAs differ from conventional optimisation and search algorithms in several ways (Goldberg, 1989a):

(i) EAs use probabilistic rules to make decisions. This has introduced intellectual capability in EAs and transformed a deterministic problem into a non-deterministic.

(ii) EAs evaluate multiple points in the solution space simultaneously, instead of a single point. Therefore, it is capable of avoiding many local optima.

(iii) EAs use pay-off (objective function) information to guide the search and thus they are more robust in achieving optimal solution compared with *a-priori* optimisation techniques.

(iv) EAs have more computation burden and are non-deterministic methods.

In summary, EAs have been found to be very effective and powerful in searching poorly understood, irregular, complex or non-differentiable spaces for optimisation and machine learning (Goldberg, 1989a; Holland, 1975). They can thus provide feasible solutions to automated control system design. Control engineers' existing knowledge and experience can be included in the EAs to assist in fast design. This technique has been successfully applied to controller order reduction (Caponetto *et al.*, 1994; Tan and Li, 1996); optimal control (Fleming and Fonseca, 1993; Hunt, 1992), linear control system unification and design automation (Li *et al.*, 1995, 1996b; Tan and Li, 1997), robust control and stability analysis (Dakev *et al.*, 1995; Goh *et al.*, 1996; Hunt, 1992; Murdock *et al.*, 1991; Patton and Liu, 1994), fault detection (Patton *et al.*, 1995), fuzzy logic control (Karr, 1992; Linkens and Abbod, 1992; Ng, 1995), and sliding mode control (Li *et al.*, 1996a; Ng, 1995).

Although there exist many publications on PID control using EA-based design methods, most of them are for *ad hoc* tuning (Rensburg *et al.*, 1998; Vlachos *et al.*, 1998; Wang and Kwok 1992). In this thesis, a hybridised GA has been developed for systematically batch-optimising practical PID controllers for the popular FOPDT systems with a wide range of normalised delay. The index function can take the frequency and time domain information into account in the same time.

With a view to tackling QFT control problems and orienting the design towards industrial applications, a computerised trial-and-error approach based on GAs has been proposed (Chen, Ballance, and Li, 1996). To further this research, the hybridised GA developed in this thesis is used to automate loop shaping for QFT controllers and prefilter design.

Built from existing work on EA-based optimisation of neurocontrollers (Brune *et al.*, 1998; Li and Häußler, 1996; Ng, 1995), a novel neurocontroller method is developed using the hybridised GA developed in this thesis. A novel nonlinear PID type neurocontroller is

designed to solve linear and nonlinear models based design problems. In particular, saturation and rate limits can be included in the designs. Comparison between the PID-neurocontroller and linear controllers is made in the thesis.

It is known that what highlights implementation difficulties and design inconvenience is the mapping from mathematics of designed controllers to into machine-dependent code. Java (Symantec Corporation, December 1996 and Eckel, 1997), a new computer language developed for platform-independent object-oriented programming (OOP) overcomes this problem largely. It has been most successfully applied in Internet and multimedia, but its original aim was to provide platform-independent modular code for embedded micro-controller applications in domestic appliances/consumer electronics, such as cameras, videocassette recorders and washing machines. Development of a Java based hybridised program can lead to the development of a platform-independent optimal design and implementation design suite for optimal control system in one go. Hence, the Java technologies are adopted in the development of EA-based optimal control system design automation suite in this thesis.

1.4 Contributions of This Thesis

(i) A globally optimal evolutionary methodology and environment for control system structure selection and design automation is developed, which requires no design indices to be differentiable.

(ii) This evolutionary environment is based on a hybridised GA search engine, whose local tuning is tremendously enhanced by the incorporation of Hill-Climbing (HC), Simulated Annealing (SA) and Simplex techniques. A Lamarkian inheritance technique is also developed to improve crossover and mutation operations in GAs. Benchmark tests show that this novel hybrid GA

is accurate, effective and reliable.

(iii) Based on this search engine and optimisation core, the linear and nonlinear control system design automation suite is developed in a Java based platform-independent format, which is readily available for design and design collaboration over corporate Intranets and the Internet.

(iv) Since EAs liberate the cost function or performance index used in the optimal control beyond the usual differentiable indices, specification based indices are investigated for practical control system designs. A thorough study on the merits and deficiencies of existing optimal control indices are carried out and, based on the findings, new indices are proposed, which can approach different damping ratios. Hybridised indices combining time and frequency domain measurement and accommodating practical constraints are proposed and incorporated into the design suite.

(v) Systematic and batch optimisation of PID controllers to meet practical requirements is achieved using the developed design automation suite. New cost function is designed to take disturbance rejection, stability in terms of gain and phase margins and other specifications into account in the same time. The results have shown that

- The derivative action can play a role in improving load disturbance rejection while maintaining or improve stability margins;
- Compared with Åström's (Levine, 1996) and Ho's (1995) method, the performances achieved by this method are much better not only in load disturbance rejection but in stability margins.

(vi) Further robustness issues are addressed by extending the PID structure to a free form transfer function. This is realised by achieving QFT design automation. The index used in design can now describe stability, load rejection and reduction of high frequency gains, which has not been achieved with existing methods. The corresponding prefilter can also be systematically designed if tracking is one of the specifications. The design results have shown that

- Controllers achieved by the design automation suite can offer a lower order and a lower 'high frequency gain' than the results published elsewhere so far (Chait *et al.*, 1999; Borghesani *et al.*, 1995);
- The designed controller combined with the corresponding prefilter performs more satisfactorily in time domain.

(vii) For nonlinear plants, the PID structuring and design strategy has been extended to a nonlinear format. This is structured as a building block based on neural networks. The automation design environment is employed to design and optimise the neurocontrollers. The design results have shown that

- The design suite make direct training of feedforward neurocontroller be possible;
- The neurocontroller can be designed directly from plant step response data without a model, where convolution is used;
- Growing method can optimise the structure and lead to the simplest;

- If a rate limiter is required in a practical control loop, the designed neurocontroller outperforms an optimised linear controller;
- The neurocontrollers can cope with amplitude limit better than the linear controllers achieved from LQR;
- Steady-state errors at different operating points of a nonlinear double tank system can be cancelled by a single trained neurocontroller.

1.5 Contents of the Thesis

In Chapter 2, GA is reviewed. The basic idea is illustrated by flowcharts and figures. Operators such as selection, crossover and mutation are explained. The theories behind GAs, such as the Schema Theory, niches and species are given. In addition to GAs, tuning methods, such as Simplex, SA and HC are illustrated for the development of the hybridised GA developed in the next chapter.

In Chapter 3, the development of the hybridised GA will be given. A Lamarckian inheritance technique is developed to replace crossover and mutation. Details on coding system are given. Then benchmarks are proposed and many algorithms are compared by benchmark tests. Following the establishment of the hybridised GA, what is necessary to employ this method in the control system optimisation, is a proper index, which can describe specifications for practical applications. Usual specifications widely accepted by control practitioners are given first, followed by an investigation into relationship between basic indices and the specifications. This chapter also develops some new indices. Hybridised non-differential indices are also developed, which may be used in the GA-based automation suite.

Chapter 4 is concentrated with PID controller design and optimisation. Since there exist many tuning rules obtained by numerous researchers, the defects with the existing rules are investigated in this chapter. Then based on gain and phase margins methods (Ho *et al.*, 1995), batch optimisation of PID controllers by the design automation suite is carried. The results are compared with the method by Åström (Levine, 1996).

In Chapter 5, loop shaping for QFT controller design is developed. The process of designing evaluation function is given. Examples of optimisation are presented, together with comparisons.

In Chapter 6, PID structure based nonlinear controllers are developed. The method of achieving the simplest neurocontroller is investigated. Examples of designing PID type neurocontrollers are given, especially with nonlinear limits, such as saturation and rate limits. A comparison with linear controllers is given to illustrate the usefulness of neurocontrollers.

Conclusions are drawn and future work is suggested in Chapter 7.

Chapter 2 Background on Genetic and Evolutionary Algorithms

2.1 Genetic Algorithms

The most widely applied EA is the GA, a coding version of EAs (Goldberg, 1989a). For a control system design problem, the "genetic codes" enable possible representation and adjustment of the system structure in addition to parameters of structure in the same evolution process (Li *et al.*, 1997). The algorithm uses three operators, namely, reproduction, crossover and mutation. Although another operator inversion may be used, it can be included by crossover and mutation (Michalewiz, 1994), and is thus not commonly used in a GA. This algorithm is based on an analogy to the genetic code in our own DNA structure, where the coded chromosome is composed of many genes, having 64 values ($64=4^3$ being the total number of different *words* permuted from 3 different alphabets out of A, C, G and T representing the 4 nitrogen-containing *bases*).

The initial population of parameters sets can be generated by random candidate solutions including, although unnecessary, *a-priori* parameters, which may lead to a faster convergence (Ng, 1995). In a population of individuals, a GA conducts multiple searches in parallel by effective exchange of co-ordinate information (parameters) through crossover. At each stage of evolution, the parameter values are altered randomly by crossover and mutation. Then the performance of all candidate parameter sets are evaluated and the whole generation is guided *a posteriori* to evolve in a "survival-of-the fittest" manner. Hence superior parameter sets would receive more attention for replication refinement from generation to generation according to the Schema Theorem (Goldberg, 1989a). The basic operation of a simple GA is shown in Figure 2.1. To illustrate this, a GA example is shown in Figure 2.2 (Li, 1999b). The operation details of Figure 2.2 are explained in the Table 2.1.

Make initial population;

Evaluate the fitness of all initial chromosomes

Repeat

Reproduce children from parents;

Apply crossover and mutation with the children

Extend the population with the children;

Select fittest chromosomes of the extended population for next circle;

Continue until satisfactory result found or maximum generation number reached;

Decode the best chromosome found

Figure 2.1 Basic operations in a simple GA

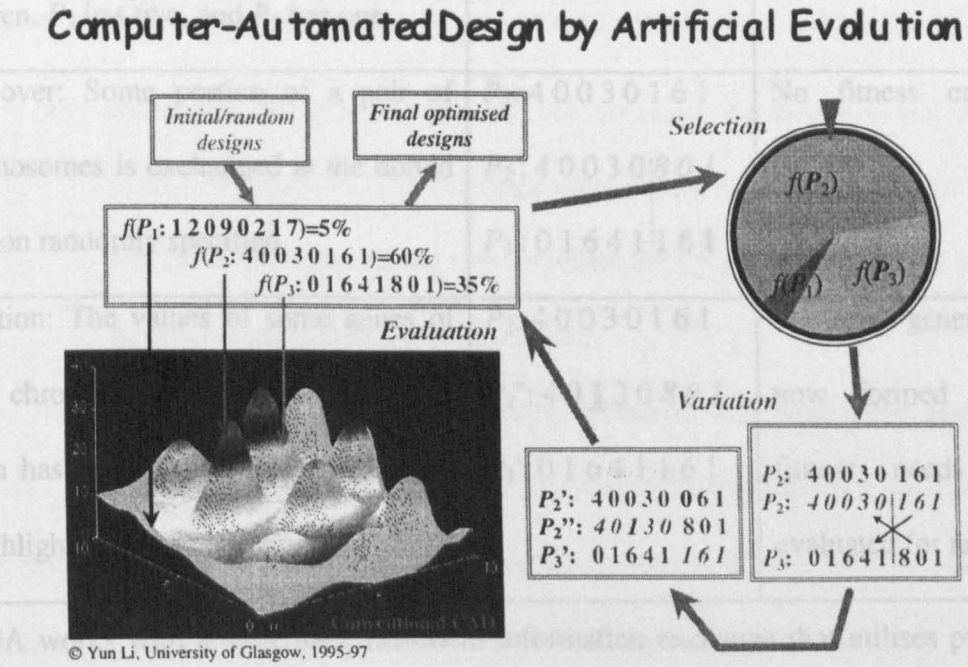

Figure 2.2 Genetic evolution of a parameter set

Table 2.1 Evolutionary process of a GA program.

Operations	Chromosomes	Fitness
Initial population: Example of coded parameter sets forming an initial population with size 3. The performance of each parameter set is simulated and then assigned fitness.	P_1: 1 2 0 9 0 2 1 7 P_2: 4 0 0 3 0 1 6 1 P_3: 0 1 6 4 1 8 0 1	$f(p_1) = 5\%$ $f(p_2) = 60\%$ $f(p_3) = 35\%$
Reproduction: A simple scheme is to allow the chromosomes to reproduce offspring according to their respective fitness. Thus, here P_1 doesn't have children. P_2 has two, and P_3 has one.	P_2: 4 0 0 3 0 1 6 1 P_2: 4 0 0 3 0 **1 6 1** P_3: 0 1 6 4 1 *8 0 1*	Evolution in progress
Crossover: Some portion of a pair of chromosomes is exchanged at the dotted position randomly specified.	P_2: 4 0 0 3 0 1 6 1 P_2': 4 0 0 3 0 *8 0 1* P_3': 0 1 6 4 1 **1 6 1**	No fitness calculation needed here.
Mutation: The values of some genes of some chromosomes changed. The value which has been changed as an example is highlighted by an underline.	P_2: 4 0 0 3 0 1 6 1 P_2'': 4 0 **1** 3 0 *8 0 1* P_3': 0 1 6 4 1 1 6 1	A new generation is now formed and the fitness needs to be evaluated for next cycle.

The GA works with a systematic historical information exchange that utilises probabilistic decisions to locate new points in the search space with improved performance. In comparison with conventional search algorithms, GAs differ in several ways:

1. GAs consider a population of points in the search space simultaneously, instead of a single point. Therefore, it is possible to avoid many local hills.

2. GAs work directly with a coding of the parameter set, instead of the parameters themselves. The method permits the optimisation of the whole parameter set simultaneously as concatenated coding is used.
3. GAs use payoff (objective function) information to guide the search, not derivative or other auxiliary knowledge. Therefore, GAs are much more robust in achieving optimal solution.
4. GAs use probabilistic rules to make decisions, not deterministic rules. This has introduced intellectual capability in GAs and transforms a deterministic non-polynomial problem to a non-deterministic polynomial problem.

Under the general framework of GAs described by Holland, the performance of a GA in optimisation tasks is characterised by the following parameters: namely, the size of the population; the crossover rate; the mutation rate. These parameters are called the control parameters of a GA. However, they have pros and cons effects in the GAs given by:

(a) Increasing the population size can increase its diversity and reduce the probability that the GAs converge prematurely to a local optimum, but it also increases the computional time required for the genetic algorithm to converge to optimal regions in the search space;
(b) Increasing the crossover rate can introduce new and more search spaces through recombination but it also increases the disruption of good strings;
(c) Increasing the mutation rate tends to transform the genetic search into a random search, but it helps to restore lost genetic material.

The setting of these control parameters may depend on user's experience and prior knowledge about the problem on hand. This is a drawback of evolutionary algorithms. However, a common choice of the population size may be set to about 10 times the

complexity of the solution space and the crossover and mutation rates are usually set at 50-80% and 0.5%-2% of the population size, respectively.

Various further enhancements of the simple GAs originally developed by Holland (1975) and later Goldberg (1989a) have been widely reported. Tournament, rank based and Boltzmann selection schemes (Baker, 1985; Sirag and Weisser, 1987; Srinivas and Patnaik, 1994) have been proposed to replace standard roulette-wheel selection for better diversity and efficiency. Extensions of a single point crossover to two point, multiple point or uniform crossover have also been reported by Spears and DeJong (1991). Adaptive mutation and multiple range decoding schemes have been proposed (Ng, 1995), which need less prior experience in fixing the mutation rate and parameter range. Generation gap hypothesis (Grefenstette, 1986) was proposed to let the parents and children coexist in the same population and allow good genetic materials to be kept. Some of these techniques are detailed in the following sections.

2.2 Genetic Operators

2.2.1 Selection and Reproduction Schemes

Reproduction is used once the initial population involving a fixed number of chromosomes representing candidate designs is formed. In the reproduction process, a new generation of population is formed by randomly selecting individuals from an existing generation, according to their fitness, to breed. This fitness test is accomplished by adopting a selection scheme in which higher fitness individuals are selected to contribute off-springs in the next generation.

Roulette Wheel Selection

One of the standard selection methods is the roulette wheel selection scheme in the simple genetic algorithm (SGA) proposed by Goldberg (1989a). This is done by generating a probability that the individual in question can be selected to reproduce itself within the fixed size of population in each generation. Each chromosome is allocated a sector (slot) of roulette wheel with the angle equal to $2\pi f_i / \hat{f}_i$, where f_i is the fitness value a chromosome in population i and $\hat{f}_i$ is the total fitness value of the population. A chromosome is selected for reproduction if a randomly generated number in the range of 0 to 2π falls in the sector corresponding to the chromosome. The algorithm selects chromosomes in this fashion until it has generated the entire population of the next generation. Although this selection scheme is easy to implement, several relatively high fitness individuals are always being selected in each roulette spin and dominating the whole reproduction process, which could lead to a premature convergence in the evolution. A different problem also arises in the later stages of the evolution when the population has converged and the variance in fitness becomes small. In this case, the selection can fail to identify two chromosomes with small variance in fitness as they occupy almost the similar sector size. These problems can be overcome by using scaling mechanisms or other selection schemes such as rank-based (Baker, 1985) or tournament selection schemes (Srinivas and Patnaik, 1994).

Rank-Based Selection

In ranking selection (Baker, 1985; Goldberg, 1989a), the population is sorted according to objective function value. Individuals are then assigned an offspring count using a predefined function. This approach provides a consistent means for offspring allocation and avoids the scaling problems encountered in the roulette wheel selection. Two types of ranking, in the

form of arithmetic and geometric series ranking schemes respectively are illustrated in Figure 2.3.

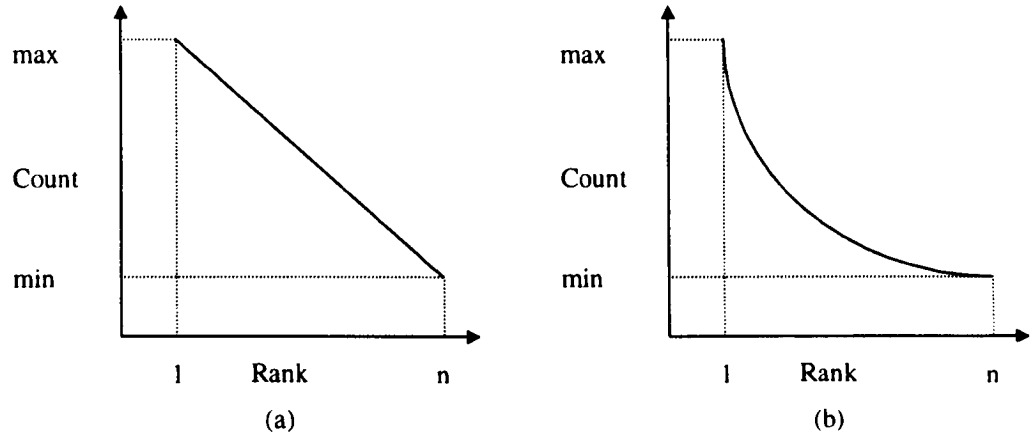

Figure 2.3 Rank-based selection scheme: (a) Arithmetic series ranking; (b) Geometric series ranking

Tournament Selection

In tournament selection as shown in Figure 2.4, the individuals are divided into subgroups and individuals with the best fitness among the subgroups are selected for reproduction. The subgroups can be of any size within population. However, a usual choice is two or three for good diversity and preventing premature convergence of the GA. A tournament selection scheme has the following advantages over standard roulette wheel selection criteria (Srinivas and Patnaik, 1994):

1. The scheme is deterministic;
2. No scaling of fitness is required;
3. Tournament size can vary;
4. Good diversity and efficiency.

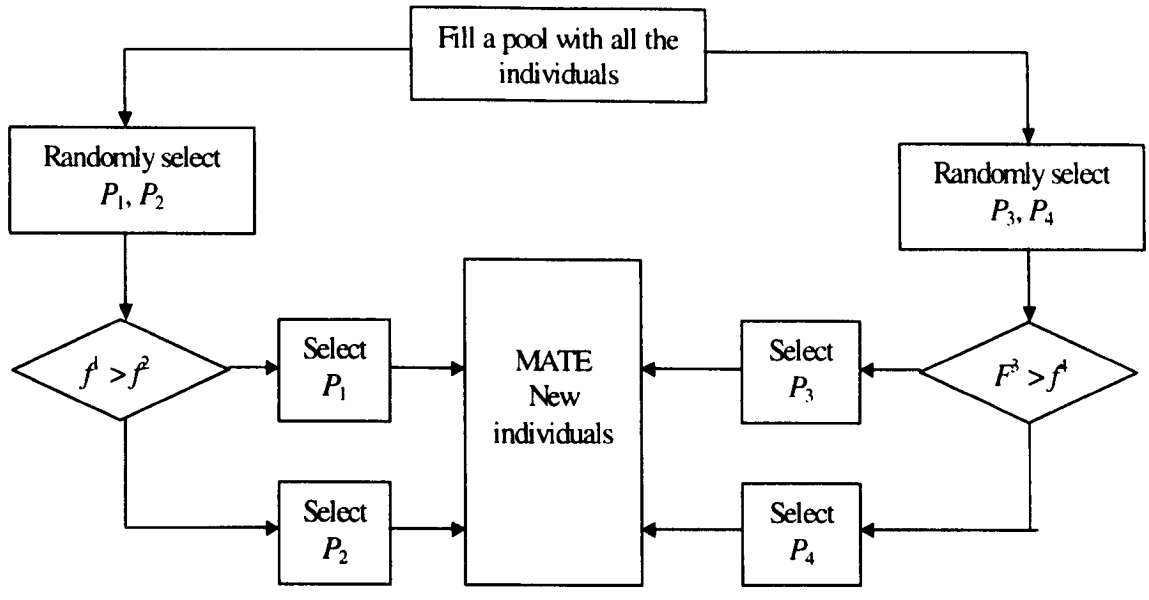

Figure 2.4 Tournament selection scheme with subgroups of size two

2.2.2 *Crossover*

Reproduction is in fact a selection process, in which two parents are chosen for mating and does not generate novel individuals in the population. Therefore, as in natural sexual reproduction, the crossover operator is used to produce offsprings that are different from their parents but inherit their parents' genetic material. Under this operator, a selected chromosome is split at the same crossover point. An example of this crossover operation is illustrated in Figure 2.2. In addition, this operation can also be applied to multiple random points.

Based on the "Choice Theorem", Zhang (1995) has reported that the best crossover point within a chromosome of N genes is $N/2$, provided the co-variance of parent chromosomes is uniformly distributed. Obviously, it is not appropriate to fix the crossover point on the centre of a chromosome, since this can lead to chromosome stagnation as the population evolves. It is because that some genes are always involved in crossover operation, but others are not. Due to this, Zhang (1995) proposed a ring type "Sufficient Exchange" optimal

uniform crossover, in which crossover is performed by first joining together the first gene of the parent chromosomes with its last gene to form a ring structure of the chromosome. Then, the chromosomes are cut into two portions upon a randomly generated diameter and crossover is realised by exchanging the first or second portion of one parent chromosome with the first or second portion of another parent chromosome. An operation of the ring type crossover is shown in Figure 2.5.

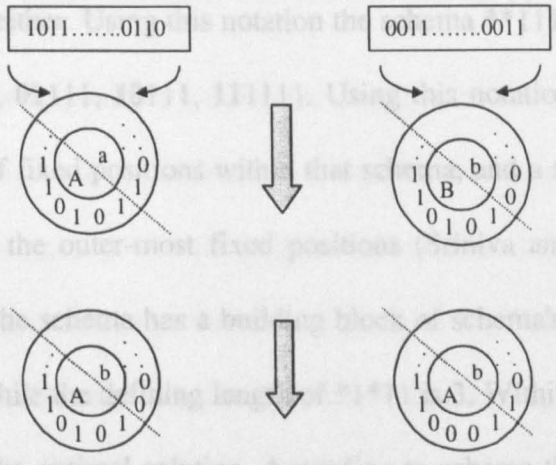

Figure 2.5 Ring type crossover

2.2.3 *Mutation*

Mutation in a chromosome is used to provide new genetic materials. This serves to keep the diversity in the population and searches the neighbouring solution space, leading to an optimal answer. In a binary GA, the mutated genes are randomly selected and subjected to inversion of its value. Decimal coding GAs can perform the mutation operation by changing the value of a gene randomly or to its adjacent value (Ng, 1995), which is shown in Figure 2.2. An adaptive mutation scheme, which varies the mutation rate upon chromosome co-variance in an evolution strategy, has also been reported by Ng (1995). The mutation operator is implemented by the Boltzmann learning technique (Tan *et al.*, 1995).

2.3 The Schema Theory

The *Schema Theorem* was developed by Holland (1975), and offers a better understanding of the convergence process of a GA (Goldberg, 1989a; Holland, 1975). A schema is defined as a template describing a subset of chromosomes with similarities at certain chromosome positions. For example, in binary bit chromosome representation, a schema matches a chromosome if at every location in the schema a '1' matches a '1', a '0' matches '0', or a '*' ("do not care") matches either. Using this notation the schema **111 describes a subset with four individuals {**00**111, **01**111, **10**111, **11**111}. Using this notation, the schema's order is defined as the number of fixed positions within that schema, and a schema's defining length is the distance between the outer-most fixed positions (Sriniva and Patnaik 1994). In the earlier example **111, the schema has a building block of schema's length of 5, order of 3, and defining length 2, while the defining length of *1*11 is 3. Within this schema space, one of these strings can be the optimal solution. According to schema theory (Goldberg 1989a), GAs find the solution by finding as many building blocks as possible, then recombining them together to give the highest fitness. The theory has also shown that a GA requires an exponentially reduced search time, compared with the exhaustive search that requires a total evaluation time of $O(p^n)$, with n being the number of parameters to be optimised in the search and p the number of possible choices of each parameters (Goldberg 1989a; Li *et al.*, 1997).

2.4 Niches and Species

For many optimisation problems there may be multiple, equal or unequal, optimal solutions. A simple GA cannot maintain stable populations at different optima of such functions. In case of optimal solutions with equal fitness, sampling errors in evaluation cause the

population to converge to a single solution. However, in the case of unequal optimal solutions, the population converges to the better.

The availability of alternate solutions is of practical value, particularly in arriving at 'robust' or multiple solutions. To achieve this objective, it is essential to introduce a controlled competition among different solutions near every locally optimal region. This would maintain stable sub-population at such optimal regions. This could be achieved by incorporating concepts of 'niche' and 'species' in the GA search process.

A niche is viewed as an organism's (individual member of the population) environment (fitness function) and a species is a collection of organisms with similar features. A simple GA with no niching converges to a single optimum although multiple peaks of equal quality may exist. Nature addresses such a problem through the formation of stable sub-population near global and local optima by introducing competition among different solutions near every local optimal region.

Niching in general is implemented by using a sharing function. The sharing function creats subdivisions of the environment by degrading an organism's fitness proportional to the number of other members in its neighbourhood. In an n-dimension space, the amount of sharing contributed by each organism $\mathbf{x}_i$ into its neighbour $\mathbf{x}_j$ is determined by their proximity in the decoded parameter space based on a relative distance measure d_{ij}. Given n parameters of unequal boundaries over a parameter range $[x_{min}, x_{max}]$,

$$d_{ij} = \left\| \frac{\mathbf{x}_i - \mathbf{x}_j}{\mathbf{x}_{k,\max} - \mathbf{x}_{k,\min}} \right\| = \sqrt{\sum_{k=1}^{n} \left(\frac{x_{k,i} - x_{k,j}}{x_{k,\max} - x_{k,\min}} \right)^2} \tag{2.1}$$

where, without loss of generality, the distance is measured on the Euclidean metric and

$x_{k,i}$ = k-th parameter of individual i;

$x_{k,j}$ = k-th parameter of individual j;

$x_{k,\max}$ = Maximum allowable value for k-th parameter; and

$x_{k,\min}$ = Minimum allowable value for k-th parameter.

For each d_{ij}, the sharing function $s(d_{ij})$ is given by the equation: (Goldberg and Richardson, 1987)

$$s(d_{ij}) = \begin{cases} 1 - (d_{ij}/\sigma_{share})^{\omega} & \text{if } d_{ij} < \sigma \\ 0 & \text{otherwise} \end{cases} \tag{2.2}$$

σ_{share} is the limiting distance between the individuals to be shared and is usually fixed by the user at some estimate of the minimal separation desired or expected between each niche in the solution space. It can be calculated using (Krishnakumar and Satyadas, 1996)

$$\sigma_{share} = 0.5n^{\left(-\frac{1}{p}\right)} \tag{2.3}$$

where n is the number of assumed peaks in the solution space, p is the number of parameters. ω = 1 in Equation (2.2) is suggested, since it allows equal degree of sharing between the neighbouring individuals. The shared fitness of the i-th individual is given as

$$f_{share} = \frac{f_{true}}{\sum_{j} s(d_{ij})} \tag{2.4}$$

2.5 Encoding of System Parameters and Structures

Encoding is the methods used to describe the system to be optimised by the chromosomes, which could be evolved by the evolutionary computations. Since in the optimisation process, it is desirable that both the parameters and structure of the system should be optimised.

Genetic Programming (GP) was invented under such a background, and it has been used for design of structures and parameters. It has been used to optimise structures and parameters of systems successfully in many cases. For example, filter designs are reported by Koza *at al.* (1997), and Uesaka and Kawamata (1999). However, it suffers from slowness of the evolutionary process. However, for many cases in engineering, GA could be used to deal with structure optimisation if a proper encoding method is employed. Kishida *et al.* (1996) have reported a GA optimisation of an IIR filter.

Another example is that of a feedforward neurocontroller which is optimised in Li and HäuBler (1997). In this case, all the weights of neurons are coded by two digits, and the coding of the hidden neurons is augmented by one additional digit (gene). For the odd values of this gene, the neuron and all of its local connections are interpreted as 'exist', and for the even values, they are not. When decoding, the existence of a hidden neuron is signalled by a flag that represents either "true" or "false" for its existence. Figure 2.6 shows the structure of such a chromosome, where w_{ij} is the jth digit of weight i.

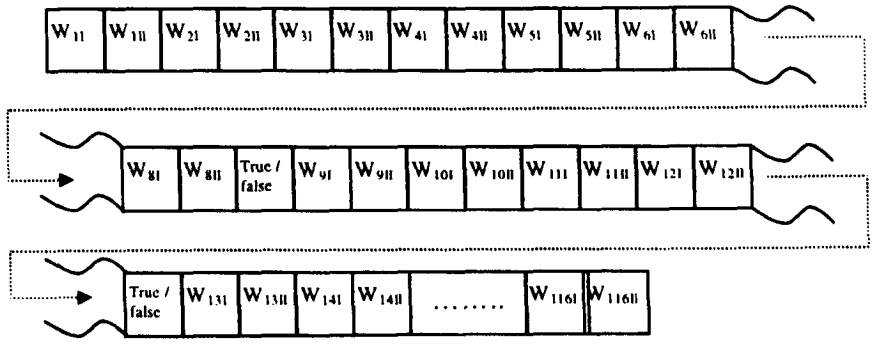

Figure 2.6 Structure of a chromosome for encoding a hidden neuron

2.6 Fitness Evaluation

Fitness function in a GA is similar to the inverse of a cost-function in any optimisation technique. The choice of fitness function in a GA is very crucial. It reflects the objectives or

specifications of the application, which directs the searching process and thus the rate of convergence. Its applications in control system are discussed in Chapter 3.

2.7 Paradigms Employed in Tuning

Despite recent techniques developed to improve the performance of a GA as discussed in the previous sections, it is well known that existing GAs are weak in local exploration and thus poor in finding the exact optima at each generation (Kwong *et al.*, 1995; Li *et al.*, 1996b; Michalewicz, 1994; and Tan, 1997). The underlying reason of this is that, in a GA, there is a lack of "biological diversity" resulting from interaction with the evolution environment. In addition, as mutation is usually set very low to avoid the GAs becoming random search, individual may become increasingly homogeneous as the GAs converge. Hybrid GA is suggested to incorporate traditional methods in local fine tuning (Tan *et al.*, 1995 and Li, 1999b). Here are these paradigms for tuning, among which simplex method is suggested in this thesis for tuning first.

2.7.1 Local Simplex Method

The *downhill simplex* method is due to Nelder and Mead (1965). This method requires only function evaluations, not derivatives. It is not very efficient in terms of the number of function evaluations that it requires. Gradient guidance based methods are faster than this method in most applications. However, the downhill simplex method may frequently be simpler to apply to a problem whose computational burden is small.

A *simplex* is the geometrical figure consisting, in N dimensions, of N+1 points (vertices) and all their interconnecting line segments, polygonal faces, etc. In two dimensions, a simplex is a triangle. In three dimensions, it is a tetrahedron, not necessarily the regular tetrahedron.

More than three dimensions, It is super space. In general, we are only interested in simplexes that are nondegenerate, If any point of a nondegenerate simplex is taken as the origin, then the N other points define vector directions that span the N-dimensional vector space.

For one-dimensional minimisation, it is possible to bracket a minimum, so that the success of a subsequent isolation is guaranteed. There is no analogous procedure in multidimensional space. For multidimensional minimisation, the best we can do is to give the algorithm a starting guess. That is, an n-vector of independent variables as the first point to try. The algorithm is then supposed to make its own way downhill through the unimaginable complexity of an n-dimensional topography, until it encounters a local minimum at least.

The downhill simplex method must be started not just with a single point, but with N+1 points, defining an initial simplex. If one of these points is thought as being the initial starting point $\mathbf{P}_0$, the other N points could be taken as

$$\mathbf{P}_i = \mathbf{P}_0 + \lambda \mathbf{e} \tag{2.5}$$

where the $\mathbf{e}_i$s are N unit vectors, and where λ is a constant which should keep $\mathbf{P}_i$ in a single modal area. Alternatively, having different λ_i for each vector direction is allowed.

The downhill simplex method now takes a series of steps. Most steps are just moving the point of the simplex where the function is largest ("highest point") though the opposite of the simplex to a lower point. These steps are called reflections, and they are constructed to conserve the volume of the simplex (hence to maintain its nondegeneracy). When it can do so, the method expands the simplex in one or another direction to take larger steps. When it reaches a "valley floor", the method contracts itself in the transverse direction and tries to ooze down the valley. If there is a situation where the simplex is trying to "pass through the

eye of a needle," it contracts itself in all directions, pulling itself in around its lowest point. The basic moves are shown in the Figure 2.7. The flowchart is shown in Figure 2.8.

Termination criteria can be delicate in any multidimensional minimisation routine. Without bracketing, and with more than one independent variable, There is no longer the option of choosing a certain tolerance for a single independent variable. One "cycle" or "step" of the multidimensional algorithm can be identified. It is then possible to terminate when the vector distance moves in a step that is fractionally smaller in magnitude than a small constant which could be chosen according to required accuracy or limitation of the computer. Alternately, when the decrease in the function value is fractionally smaller than machine constant, the search could be stopped as well.

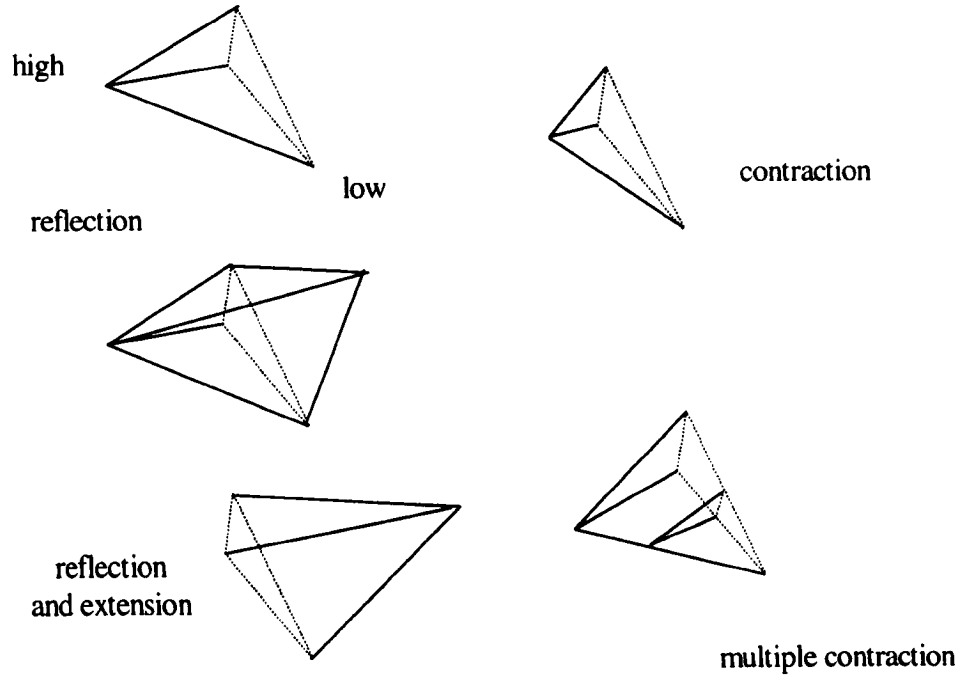

Figure 2.7 Search operations in Simplex

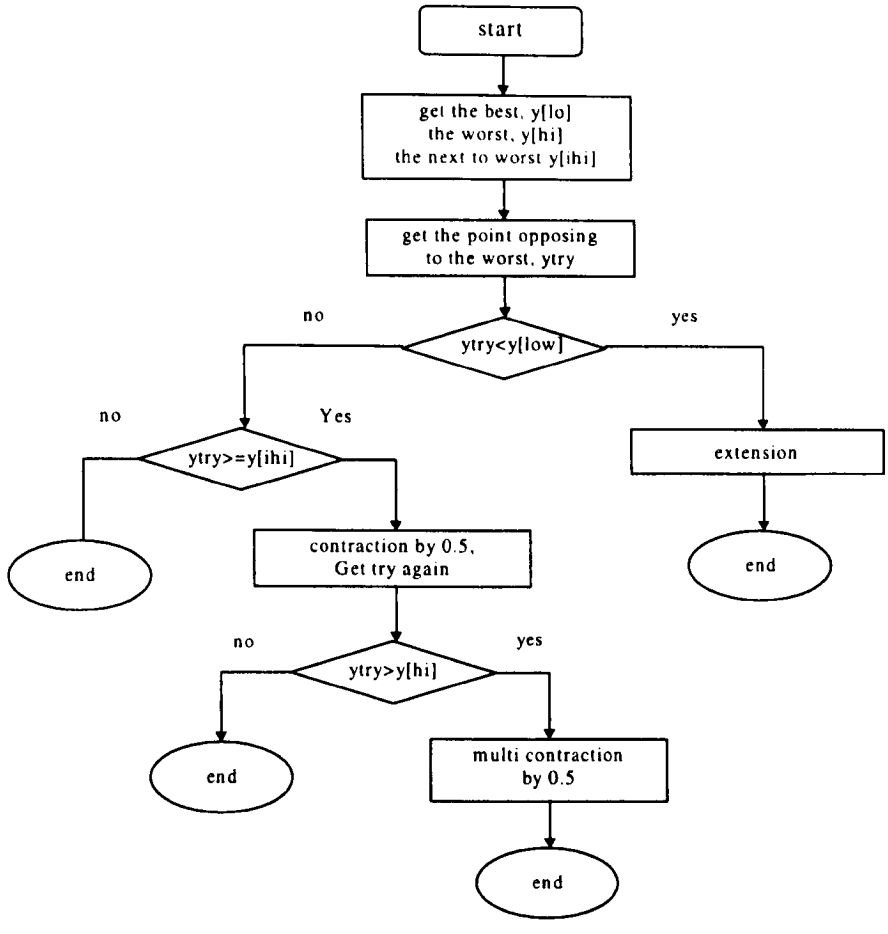

Figure 2.8 Flowchart of Simplex

2.7.2 *Non-Deterministic Hill-Climbing*

Hill-Climbing (HC) works by randomly choosing a few points to evaluate and from there adjusting some parameters in the combinational solution and re-evaluating. This is repeated until the best combination is found. This, however, may not necessarily be a very good combination and may lead to local optimum solution. This may be overcome by increasing the number of initial random points. Obviously, the greater the number of initial points the better the chances of obtaining a global optimum. Nonetheless, this increase in accuracy would be at the expense of evaluation time. This would make HC a poor method as it would not be any much better off than *Exhaustive Search,* since the time required to produce a global optimum solution would be long.

2.7.3 *Simulated Annealing*

At the heart of the method, Simulated Annealing (SA) is an analogy to thermodynamics, specifically to the way that liquids freeze and crystallise, or metals cool and anneal. At high temperatures, the molecules of a liquid move freely with respect to each other. If the liquid is cooled slowly, thermal mobility is lost. The atoms are often able to line themselves up and form a pure crystal that is completely ordered. This crystal is the state of minimum energy for this system. The fact is that nature is able to find this minimum energy state for slowly cooled systems. However, if a liquid metal is cooled quickly, it does not reach this state but rather ends up in a polycrystalline or amorphous state having higher energy. So the essence of the process is slow cooling. The *Boltzmann's* energy distribution

$$\text{Prob}(E) \propto \exp(-E/kT), \tag{2.6}$$

expresses the idea that a system in thermal equilibrium at temperature T has its energy probabilistically distributed among all different energy states E. Even at low temperature, there is a chance for the system to get out of a local minimum in favour of find a better one. The constant k is Boltzmann constant.

It was Metropolis *et al.* (1953) who incorporated this theory into numerical optimisation. When offered some options, a simulated thermodynamic system is assumed to change its configuration from E_1 to E_2 with probability

$$p = \exp\left[-\left(E_2 - E_1\right)/kT\right]. \tag{2.7}$$

Notice that if $E_1 > E_2$, this probability is greater than unity, in such cases the change probability is arbitrarily assigned as p = 1, i.e. the system always takes such an option. This general scheme is of always taking a downhill step while sometimes taking an uphill step.

Making use of the Metropolis algorithm for other than thermodynamic systems, the following elements are needed (Kirkpatrick *et al.*, 1984):

1. A description of possible system configuration;
2. A generator of random changes in the configuration;
3. An objective function E whose minimisation is the goal of the procedure;
4. An artificial temperature parameter T and an annealing schedule which control how to lowering T.

In general, this technique allows some inferior-neighbouring position to replace the current one for possible correct direction leading to the global optimum. But, as the artificial temperature decreases, the *Boltzmann* distribution concentrates on the states of lowest energy, which make SA almost the same as HC. This method, however, suffers the same disadvantages as in HC in its dependency on a set of good initial random points to obtain global optimum. However, this method is relatively better in its capability in getting a global optimum solution, as compared to HC (Kirkpatrick *et al.*, 1984).

2.8 Summary

In this chapter, the basic process of evolutionary computing have been presented with a simple example. Advantages of non-binary coding strategy are also given. Then the operators including selection, crossover and mutation employed are given with some analysis. Selection techniques including roulette wheel selection, rank-based selection and tournament selection are illustrated. The theories behind crossover and mutation are also highlighted. An insight of convergence process is given through the schema theorem. These are followed by the niching method to improve GA in retaining local optimal. Defects in the GA are also pointed out. Mechanisms of some traditional paradigms, which could be

employed to tune GA, are discussed. Among of these, simplex tuning is suggested in this thesis, owing to its speed and accuracy for local refinements.

Chapter 3 Structuring and Global Optimisation Evolutionary Environment for Control Systems

It is recognised that a classical GA can perform much better if some forms of local fine-tuning process can be incorporated (Renders and Bersini, 1994; Li *et al.*, 1995 and Tan, 1997). In this thesis, an EA strategy, which includes three tuning paradigms (HC, SA and Simplex), is developed. It can be considered as an improved or hybridised GA. However, it should be stressed that the new strategy is still based on search techniques inspired by natural science and most concepts, such as population, genes, etc, are retained in the hybridised GA for global and structural search. Here, the encoding and decoding facilities are improved from conventional GAs, and objective functions are designed to indicate control system performance. In this chapter, Section 3.1 presents a Lamarckian inheritance strategy and tuning methods to be used. Some details of the hybridised GA including coding strategy are given in Section 3.2. In Section 3.3, benchmarks are investigated, after which many EA methods are tested including the hybridised GA developed here. To use the hybridised GA environment to evolve control systems for best specifications, performance indices are analysed and new ones are developed in Section 3.4. Section 3.5 presents a summary.

3.1 Improving Both Global Search and Local Tuning

3.1.1 Improving Convergence by Lamarckian Inheritance

In this work a Lamarckian inheritance technique is used to enhance evolutionary process achieved by crossover and selection used by general GAs, which have been discussed in Chapter 2. Figure 3.1 shows the mechanics of the process. In this scheme, 2 individuals (parents) are randomly selected from the population. Thereafter, an *Inheritance Ratio* (Li, 1995; Tan, 1997) is computed, which is defined as:

$$Inheritance\ Ratio = \frac{Fitter\ Parent's\ Fitness}{Total\ Fitness\ of\ Parents} \tag{3.1}$$

This determines how much of the genetic materials from the fitter parent is imparted to the weaker in a random fashion at the same gene location. The fitter parent undergoes the tuning process, which will be presented shortly. The whole process continues until all the chromosomes within the population are evaluated.

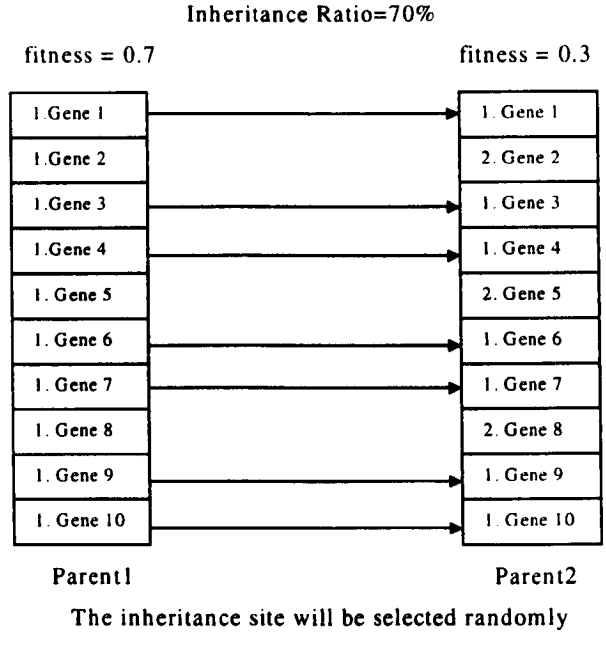

Figure 3.1 Schematic of the Lamarckian inheritance scheme

3.1.2 Mutation with HC or SA

The inheritance operator itself is not sufficient for the hybridised GA to be complete. Yet other operators are needed with the GA to guide the search towards a global optimum. Thus, HC or SA are introduced.

The scheme in this thesis allows the HC or SA to mutate at any random values negatively or positively at generations before half of specified total generation. After this cutoff, the perturbation is limited to only a 10% range, but still in both directions. This allows the fine-

tuning process to be achieved at later part of the evolution (assuming that convergence occurs in the midway of evolution). Unlike traditional GAs the mutation operator is omitted. However, HC or SA here act as a mutation operator, which similarly introduce new materials into the chromosome. Figure 3.2 shows a simple flow chart of Lamarckian Inheritance scheme.

3.1.3 Fine Tuning with Simplex

There is another tuning method, which is Simplex. Just like what is investigated in Chapter 2, it is a reliable vehicle to approach a local optimal, if the initial points are put on a single modal. However, if the initial points in different modals, it could jump between the modals and spend much time in settling down. So in the hybridised GA, it is not used in every generation. Just when there appears a new better chromosome Simplex is applied to tuning it to a better optimal. The flow chart for one generation of the hybridised GA is given as

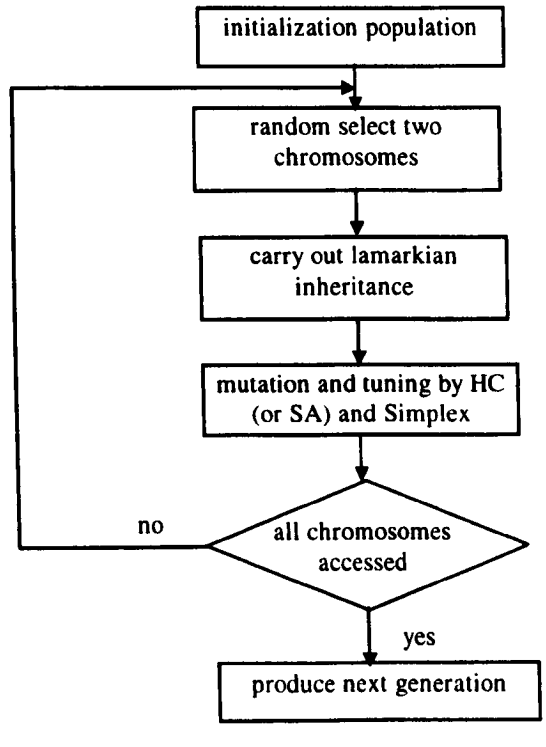

Figure 3.2 Flowchart of a hybridised GA

In summary, the evolutionary environment developed in this thesis is different from existing evolutionary methods in following points:

1. Lamarkian inheritance is used to enhance evolutionary process achieved by crossover and selection in general GAs.

2. HC or SA is used for mutation (For the application in Chapters 4 to 6, HC is used) and the Simplex is used for fine-tuning.

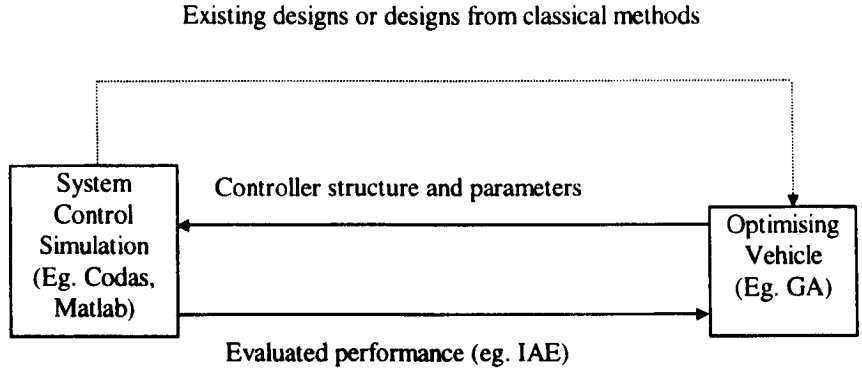

Figure 3.3 Basic structure of optimisation environment

3.2 GA Environment in Java

Java was developed based on C++ language. Unlike other language, output of a Java compiler is not executable code, but bytecode. It is a highly optimised set of instructions designed to be executed by the Java run-time system, which is called the Java Virtual Machine (JVM). Translating a Java program into bytecode helps make it much easier to run a program in a variety of environments, because only the JVM needs to be implemented for each platform (Symantec Corporation, December 1996 and Eckel, 1997). Thus, Java is employed to program here, because of consideration of future portability. Here are the classes in the hybridised GA Java Environment.

3.2.1 Class of Individuals

The Individual Class consists of three main private members. The first is *Gene*, which is an array representing a gene, and is allocated with corresponding memory. The following is *FitnessValue*, which specifies the degree of fitness of this specific of chromosome strand. And the last is *IndNum*, which is the number of this individual in the population. To evaluate this chromosome, the *Objective* member function is used. The function, which needs to be coded by a user, is called once the content of a chromosome is changed (see later in GA class). The member function of *StreamResults* is used to show fitness of a chromosome.

3.2.2 Class of Populations

The Population class holds the collection of all the individuals. It monitors the average fitness among the individuals and also keeps track of the best of them. The population is stored in 2D array where items in the array are gene arrays. At the very beginning of evolution, the initial population is randomly set up by its member function of evolution *InitPop*.

3.2.3 Class of GAs

This is the base class of our hybridised GA. It contains four important operator member functions:

- Inherits (Individual, Individual)
- HC (Individual)
- SA (Individual)
- Simplex (Individual)

The *Inherits* member function takes in two individual objects and does the necessary imparting of genes. *HC* and *SA* use the member function *Perturb* to carry out perturbation process of a single individual. *Simplex* is used only when a new best point is found. The constructor of this class assigns member variables to their default values. However, these values can be changed or accessed by their associating member functions.

3.2.4 Coding for Control System Parameters and Structures

A Traditional GA uses a binary coding scheme. This method utilises only a binary bit (i.e. 'a' 'o' or'1') for each gene. Since an engineering problems today require high precision and may involve a very large memory size for the coding of a chromosome. What binary coding can do in the past is impractical for present.

One-integer-parameter coding, a technique invented in Glasgow (Li, 1995 and Tan, 1997), utilises only one gene per parameter. This enables us to save up to 32 times amount of memory space and processing time. A short integer data type within the Java language uses 15+1 bits (15 for data and 1 for the sign). That allows us to have a range [-32k~32k] for the value of each gene. The random initialisation of the population sets the genes to within such a range. While finding a particular chromosome's fitness requires the real value of the parameters, each gene goes through a decoding process. This is merely a mapping process, which uses a precise constant.

The advantage of coding is that logic values and operators can also be coded in a chromosome, which makes the search more versatile and complete. It makes the coding structure possible. For example, a link in neural network architecture can be encoded within a single gene. When the gene is even number, the link exists, or it does not. However, the disadvantage of coding is that decoding to a phenotype is needed before fitness can be

evaluated, which takes unnecessary time. In order to achieve a simple optimised structure, growth method is used in this thesis. To be simple, it means that growing the length of chromosomes systematically to achieve the simplest structure. This idea is detailed in Chapter 6.

3.3 Benchmarks and Tests of Search Engines

During the past four decades, many evolutionary and other global optimisation and search algorithms have been developed. Those algorithms have been shown to be very useful in enabling control systems design automation, highly accurate and high-performance measurements. There exists, however, a lack of systematic benchmark measures that may be used to assess the merit and performance of these algorithms. Such benchmarks should be simple to use and should result in little program overhead. This thesis attempts to formalise, and to promote discussions on this issue. Here, definitions about benchmarks in the terms of (i) optimality; (ii) accuracy; (iii) convergence; and (iv) total number of evaluations and (v) program overhead are formalised.

3.3.1 Test Function and its Objectives

Consider a benchmark problem for testing an optimisation, learning or search algorithm. Suppose that its *objective function* (cost function, performance index or fitness function) is $f(\boldsymbol{x})$: $\boldsymbol{X} \rightarrow \boldsymbol{F}$, which may be evaluated via analytical calculations or numerical simulations. Here $\boldsymbol{X} \subseteq \mathbf{R}^n$ represents the entire search or possible solution space in n dimensions, $\boldsymbol{x} \in \boldsymbol{X}$ represents the n collective variables or parameters to be optimised, $\boldsymbol{F} \subseteq \mathbf{R}^m$ represents the m dimensional space of all possible objective values, and $\boldsymbol{f} \in \boldsymbol{F}$ represents the collection of m objective elements.

Denote the *theoretical objective* vector that may be ultimately reached as

$$f_0 = \text{objective } \{f(x)\} \in F \tag{3.2}$$

Note that elements in f_0 can have separate objectives, i.e., some for *maximisation* and some for *minimisation*. Note also that a non-numerical objective element, such as a "logic" objective, may only take the value of 0 or 1. An $x_0 \in X$ that satisfies

$$f(x_0) = f_0 \tag{3.3}$$

is said to be a corresponding *theoretical solution* to the optimisation problem. Note that, for a *non-dominant* or *non-commensurate* multi-objective optimisation problem, f_0 represents a collection of individual theoretical objectives that may only be reached separately by different solutions. In this case, there does not exist a single, or dominant, solution and hence a *quasi-theoretical solution* needs to be defined (See (3.3.2.2)).

Denote an *objective reached* by the optimisation algorithm as $\hat{f}_0$. An $\hat{x}_0 \in X$ satisfying

$$f(\hat{x}_0) = \hat{f}_0 \tag{3.4}$$

represents a corresponding *solution found.*

3.3.2 Benchmarks

Based on the above notation, benchmarks conforming to simplicity, wide applicability and reliability are to be formalised. The benchmarks should also be designed such that little testing overhead may be added to the algorithms being probed. Note that mean values of test results over multiple runs must be used, since most global optimisation algorithms are often non-deterministic.

3.3.2.1 Optimality

Optimality of an objective reached represents its relative closeness to the theoretical objective. It can be defined as:

$$Optimality\ (\hat{f}_0)\Big|_a = 1 - \frac{\left\| f_0 - \hat{f}_0 \right\|_a}{\left\| \overline{f} - \underline{f} \right\|_a} \in [0, 1] \tag{3.5}$$

where $\underline{f}$ is the lower bound of f and $\overline{f}$ is the upper bound.

Remarks

1. Any popular norm, such as the 1-norm (sum of absolute), the 2-norm ($\sqrt{m}$ times root mean squares, where m is number of objectives) and the ∞-norm (maximum of absolute), defined in a Banach (i.e., normed and complete) space may apply to (3.5). The optimality thus defined is termed '*Banach optimality*', which represents a Banach distance to the goal (i.e., the theoretical objective), regardless of whether the optimisation problem is for maximisation or is for minimisation, and thus unifies this benchmark.
2. In engineering applications, the 2-norm (Euclidean metric) is most commonly adopted for such a metric on $\mathbf{R}^m$ and the optimality thus defined is termed '*Euclidean optimality*'.

Single Objective

For a single-objective problem (i.e., $m = 1$), all a-norms are identical. Consider a maximisation problem with an objective bounded by $[f_{min}, f_{max}]$ as an example. Then, by (3.5) the optimality measure can be simplified to:

$$Optimality\big|_{\max} = 1 - \frac{\left|f_{\max} - \hat{f}_0\right|}{f_{\max} - f_{\min}} = \frac{\hat{f}_0 - f_{\min}}{f_{\max} - f_{\min}} \tag{3.6}$$

Similarly, for a single-objective minimisation problem, the optimality measure can be simplified to:

$$Optimality\big|_{\min} = 1 - \frac{\left|f_{\min} - \hat{f}_0\right|}{f_{\max} - f_{\min}} = \frac{f_{\max} - \hat{f}_0}{f_{\max} - f_{\min}} \tag{3.7}$$

Multiple Objectives

For a dominant multi-objective problem, definition (3.5) suffices. For a non-dominant or non-commensurate multi-objective solution, if each objective of a multi-objective problem or algorithm needs to be assessed separately, then (3.6) may be applied m times individually to replace (3.5). However, it is difficult to assess an *overall optimality* by a single quantity without combining all objectives to form a composite optimality. One method adopted to achieve this is to measure the 'distance to demands' (Battiti, 1993).

This method can be implemented easily in (3.5) by setting the ultimate goals, f_0, as the 'demand levels', as illustrated in Figure 3.4 for a 2-objective solution using the Euclidean distance. Therefore, the smaller the distance is the higher the overall optimality. Whether the solution is dominant or non-dominant, this definition of optimality preserves and extends the concept of '*Pareto optimality*' (Michalewicz, 1992 and Goldberg, 1989a). As shown in Figure 3.4, a higher optimality guarantees that the corresponding solution is closer to at least one goal.

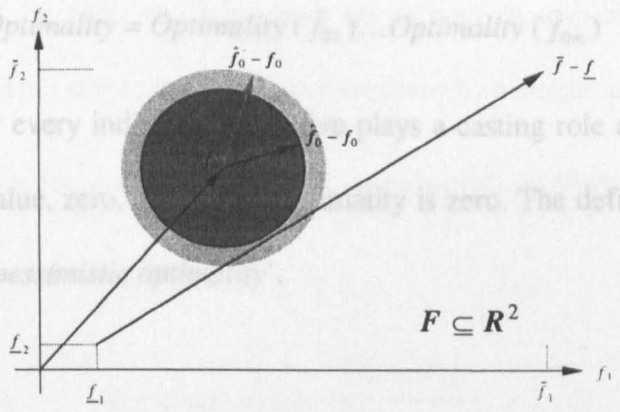

Figure 3.4 Non-dominant solution $\hat{f}_0$ having an f_2 closer to the goal than $\hat{f}_0'$ but an f_1 that is farther; Overall $\hat{f}_0$ has a shorter 'distance to demand'.

Note that, if it is required to weight each single distance or objective distinctively, this should be done before calculating the norms. For this, the objective vector, $\boldsymbol{f}$, should be replaced by:

$$\boldsymbol{wf} = [w_1 f_1 \;\; w_2 f_2 \ldots \;\; w_m f_m]^T \tag{3.8}$$

where

$$\boldsymbol{w} = \text{diag}\,[w_1, w_2, \ldots w_m] \tag{3.9}$$

represents the weighting vector.

Remarks

1. The value of a 'logic' objective, say f_{01}, can only be either 1 or 0. If this individual objective is deemed as uncompromising, it may be separated from the others such that (3.5) may be replaced by:

$$\textit{Optimality} = \textit{Optimality}\,(\hat{f}_{01}) \cdot \textit{Optimality}\,(\hat{f}_{02}, \cdots, \hat{f}_{0m}) \tag{3.10}$$

where Optimality($\hat{f}_{01}$) plays a casting role in assessing the overall optimality.

2. To extend from (3.10), the definition of (3.5) may be replaced by the product of all individual optimalities, i.e.,

$$Optimality = Optimality\,(\hat{f}_{01})\ldots Optimality\,(\hat{f}_{0m}) \tag{3.11}$$

3. This means that every individual objective plays a casting role and if any one falls to the minimum value, zero, the overall optimality is zero. The definition given by (3.11) is thus termed '*pessimistic optimality*'.

3.3.2.2 Accuracy

Accuracy represents the relative closeness of a solution found to the theoretical solution. It can be defined as:

$$Accuracy\ (\hat{\boldsymbol{x}}_0) = 1 - \frac{\left\|\boldsymbol{x}_0 - \hat{\boldsymbol{x}}_0\right\|}{\left\|\bar{\boldsymbol{x}} - \underline{\boldsymbol{x}}\right\|} \in [0, 1] \tag{3.12}$$

where $\underline{x}$ and $\bar{x}$ are the lower and upper bounds of $\boldsymbol{x}$, representing the search range. This benchmark is particularly useful if the solution space is noisy, there exist multiple optima or "niching" is used.

Remarks

1. A more sophisticated measure is to replace the denominator in (3.12) with the maximal distance to the theoretical solution in $\boldsymbol{X}$.
2. There may be distinctive solutions corresponding to the same objective value, with or without niching. In this case, the highest accuracy calculated should be used.
3. For a non-dominant multi-objective problem, the concept of 'accuracy' is no longer valid, as there does not exist a (dominant) theoretical solution. Assessing a solution found would be subjective and may only be carried out separately on each element of the solution, if the user or designer takes a 'minimum commitment principle' (Guan *et al.*, 1996) in the optimisation. However, a solution that results in the maximal overall

optimality given by (3.5) or (3.11) may be defined as a *quasi-theoretical solution* benchmark, $\tilde{x}_0$. This allows measuring the accuracy by a single quantity.

By (3.12), the accuracy for a single-parameter optimisation problem within $[x_{\min}, x_{\max}]$ is measured by:

$$Accuracy = 1 - \frac{|x_0 - \hat{x}_0|}{x_{\max} - x_{\min}} \tag{3.13}$$

3.3.2.3 Sensitivity

When the values of optimal parameters found are perturbed or manufacturing tolerance in accuracy is taken into account, the actual optimality may change. This affects the design robustness of an engineering system. To measure how much a "small" relative change in the designed parameters (solution found) lead to a relative change in the quality (objective value reached), the usual definition of 'sensitivity' may apply. For example, the sensitivity of a single-objective problem may be defined as:

$$Sensitivity = \lim_{\|\Delta x\| \to 0} \frac{\|\Delta f\| / \|\hat{f}_0\|}{\|\Delta x\| / \|\hat{x}_0\|} = \frac{\|\hat{x}_0\|}{|\hat{f}_0|} \lim_{|\Delta x_i| \to 0} \left\{ \frac{|\Delta f|^a}{\sum_i |\Delta x_i|^a} \right\}_{x = \hat{x}_0}^{1/a} \le \frac{\|\hat{x}_0\|}{|\hat{f}_0|} \lim_{|\Delta x_i| \to 0} \left\{ \sum_i \left| \frac{\Delta f}{\Delta x_i} \right|^a \right\}^{1/a}$$

$$= \frac{\|\hat{x}_0\|}{|\hat{f}_0|} \left\{ \sum_i \left| \lim_{|\Delta x_i| \to 0} \frac{\Delta f}{\Delta x_i} \right|^a \right\}^{1/a} = \frac{\|\hat{x}_0\|}{|\hat{f}_0|} \left\{ \sum_i \left| \frac{\partial f}{\partial x_i} \right|^a \right\}^{1/a} = \frac{\|\hat{x}_0\|}{|\hat{f}_0|} \|\nabla f\|_{x = \hat{x}_0} \tag{3.14}$$

which has a value of zero at the theoretical solution if the objective function is differentiable.

In a test, a simple approximation of (3.14) can be obtained by perturbing the solution found by, for example, 0.1%, i.e.:

$$Sensitivity = \frac{\|\Delta f\| / \|\hat{f}_0\|}{0.001} = 1000 \frac{\|f(1.001\hat{x}_0) - \hat{f}_0\|}{\|\hat{f}_0\|} \tag{3.15}$$

Note that sensitivity is related to "relative gradient". It is thus dependent upon the test objective function and not directly upon the algorithm. It indicates the nature of the problem and its 'fitness landscape' (Goldberg, 1989a). Sensitivity would be a more useful indicator in a practical design than in a benchmark test. If design robustness needs to be optimised during an evolution process, sensitivity could be used as an additional objective of the design. Nevertheless, this benchmark provides another indicator on how close the solution found is to the theoretical solution that might be obtained by *gradient-guidance*.

3.3.2.4 Convergence

Generational Convergence

In a GA, the average fitness of the entire population is used to assess the convergence trend qualitatively, for the mutation rate in a GA is relatively very low. This fitness is, however, often oscillatory when the evolution reaches a 'steady-state' or a relatively high mutation rate is used as in the case of EP or ES. Therefore, it differs from the concept of 'convergence' adopted in conventional optimisation paradigms and can hardly fulfil the role as a quantitative indicator or benchmark of convergence. Hence, the traces of the following are used to indicate the generational convergence:

1. The highest 'optimality' or fitness in every generation;
2. The highest 'accuracy' or the parameter values of the individual solution that have the highest fitness in every generation.

Reach-Time

To quantify the convergence benchmark with respect to a GA, define

$$Reach\text{-}time\big|_b = C^b \tag{3.16}$$

to represent the total number of 'function evaluations' conducted after which the optimality of the best individual first reaches $b \in [0, 1]$. This also means that the relative distance to the theoretical objective first drops to $1 - b$ by the 'reach-time'. For example, the following two reach-times may be useful indicators:

- $C^{0.999}$
- $C^{0.632}$

The former would be perhaps the more significant indicator. The latter means a convergence 'time-constant', by which an optimality of 63.2% is first reached in a similar manner to a first-order dynamical system.

NP-Convergence

The power of an EA is that it reduces exponential computational time needed by an exhaustive search algorithm to a non-deterministic polynomial (NP) computational time. To estimate the order of the polynomial, $C^{0.999}$ may be plotted against the number of parameters being optimised, n, as revised in:

$$NP\text{-}time\ (\mathrm{n}) = C^{0.999}(n) \tag{3.17}$$

Total Number of Evaluations

During the entire optimisation process, the optimality of 99.9% may not be reached by certain algorithms under test. The total number of evaluations is the number of function evaluations, search trials or simulations performed in the entire optimisation process until

termination. This should be kept the same for all the algorithms compared in a benchmark test, such as $400mn^2$. It may be more informatively defined as

$$N = \min\left\{C^{0.999}, 400mn^2\right\} \tag{3.18}$$

which implies that a benchmark test should terminate either when the goal has been reached or $20n$ generations of a size of 20 $n \times m$ have been evolved. This also means that there is faith that GAs should not perform worse than an $O(n^2)$ algorithm in terms of computational time.

Remarks

1. A polynomial quantitatively representing the NP-convergence may be obtained by curve-fitting the convergence trace.
2. The higher the threshold b is, the more meaningful the convergence indicator could be, but it is more difficult to reach.
3. Note that these definitions concerning "convergence" are not proposed to replace theoretical proofs of convergence of an optimisation algorithm, but are only proposed for use as a benchmark for assessing the performance of the algorithm statistically.
4. Alternative to highest optimality, highest accuracy may be used as the individual threshold b in C^b.

3.3.2.5 Optimiser Overhead

Alternative to or in addition to the 'total number of evaluations', the 'total CPU time' may be used in a benchmark test. This would be useful in indicating how long an optimisation or simulated evolution process would take in real world and to indicate the amount of program overhead as a result of the optimisation manipulations such as those by EA operators. More quantitatively, the optimiser overhead may be calculated by:

$$\text{Optimiser overhead} = \frac{\text{Totaltime taken} - T_{RS}}{T_{RS}} \tag{3.19}$$

where T_{RS} is the mean time taken in a *random search* by evaluating the test function N times and retaining the best solution found while search progresses.

Before carrying out benchmark tests against evolutionary methods in Section 3.3.3, Not that the five benchmarks formalised here are used for the widely studied benchmark problems, to which the theoretical solution are known. So for control application problems in Chapters 4 to 6, the optimisation results are not tested against the benchmarks formalised here, since no theoretical optima can be obtained. That is because real applications are involved with nonlinearities and uncertainty. The benchmarks are used to test optimising ability of different algorithms in theory. Then the better ones can be chosen for real applications.

3.3.3 An n-D Benchmark Problem and its Test

The objective function of an n-dimensional maximisation problem that was introduced in (Michalewicz, 1992) and further studied in (Renders and Bersini, 1994 and Feng *et al.*, 1998) is given by:

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} f_i(x_i) = \sum_{i=1}^{n} \sin(x_i) \sin^{2k}\left(\frac{i x_i^2}{\pi}\right) \quad \forall\, \boldsymbol{x} \in [0, \pi]^n \tag{3.20}$$

It is composed of a family of amplitude-modulated sine-waves whose frequencies are linearly modulated.

Table 3.1 Theoretical solutions and objectives of a benchmark problem

I	1	2	3	4	5	6	7	8	9	10
x_{i0}	2.072	1.571	1.305	1.916	1.718	1.571	1.458	1.755	1.655	1.571
f_{i0}	.8409	1.000	.9619	.9396	.9890	1.000	.9933	.9830	.9964	1.000

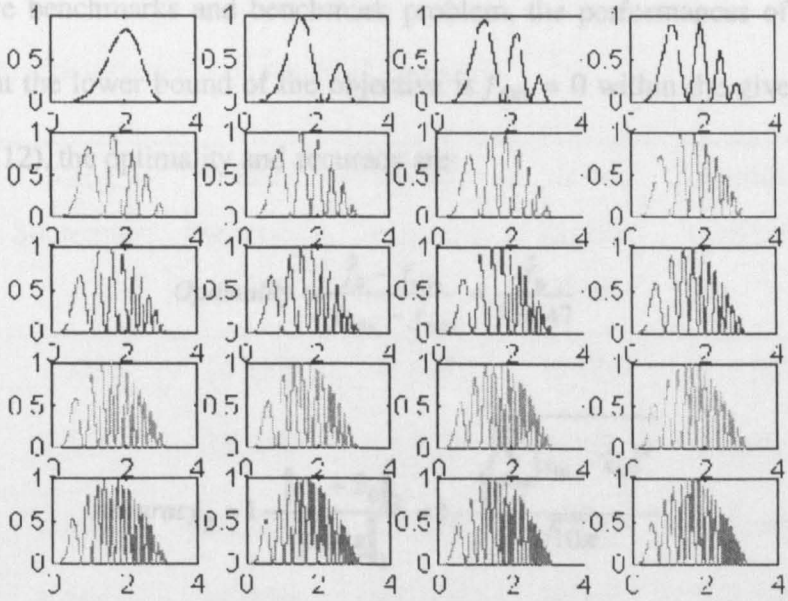

Figure 3.5 The n independent uni-dimensional functions that form the 20-D objective function

This objective function is, in effect, de-coupled in every dimension by $f_i(x_i)$. Every such member function is independent and is shown in Figure 3.5 for k = 1 and n = 20. This characteristic yields the following properties:

1. The theoretical benchmark solution to this n-dimensional optimisation problem may be obtained by maximising n independent uni-dimensional functions, f_i, the fact of which is however unknown to an optimisation algorithm being tested. The results for k = 1 and n = 10 are shown in

2. The larger the product kn is, the sharper the landscape becomes.

3. There are $n! = 2.4329\times10^{18}$ local maxima within the search space $[0, \pi]^n$.

4. The ease of obtaining theoretical benchmarks regardless of n makes it ideal for studying NP characteristics of the algorithms being tested.

Using the above benchmarks and benchmark problem, the performances of some EAs are tested. Note that the lower bound of the objective is $f_{\min} = 0$ within the given search space. By (3.5) and (3.12), the optimality and accuracy are:

$$Optimality = \frac{\hat{f}_0 - f_{\min}}{f_{\max} - f_{\min}} = \frac{\hat{f}_0}{9.6547} \tag{3.21}$$

$$Accuracy\Big|_2 = 1 - \frac{\left\|\boldsymbol{x}_0 - \hat{\boldsymbol{x}}_0\right\|_2}{\left\|\overline{\boldsymbol{x}} - \underline{\boldsymbol{x}}\right\|_2} = 1 - \frac{\sqrt{\sum_{i=1}^{10} \left|x_{0i} - \hat{x}_{0i}\right|^2}}{\sqrt{10}\pi} \tag{3.22}$$

Tests are carried out for nine types of GA, e.g.,

(i) Simplex;

(ii) Random search for best;

(iii) 150-bit Simple GA (Goldberg, 1989a; and Li, 1999);

(iv) Floating-point GA (FPGA) (Tan, 1997);

(v) FlexTool (GA) Toolbox (Flexible Intelligence group, 1995), 0.0001 resolution;

(vi) Messy GA (Goldberg, 1989b; and Chowdhury, 1998) with single-integer coding,

(vii) Simulated annealing hybrid FPGA (Tan, 1997);

(viii) A-posteriori hill-climbing hybrid FPGA (Feng, 1998) and

(ix) A-posteriori hill-climbing plus simplex tuning FPGA.

Among these methods, (viii) and (ix) are designed and programmed in this project. (ix) is used in the practical control design in Chapters 4 to 6. (i), (ii) and (iii) are implemented by Java in this project. (iv) to (vii) are existing software, which are tested in this project. Note that (iii) to (vii) are evolutionary algorithms which adopt strategies presented in Section 2.1. For each method, 10 repeated experiments are carried out with randomly generated initial populations. The results of objective reached, optimality, accuracy, total number of function evaluations or reach-time, and optimiser overhead are shown in Table 3.2.

Table 3.2 Benchmark test results on the 10-D problem

Algorithm Tested	Mean Supremum	Mean Optimality	Mean Accuracy	*N* or Reach-Time	Optimiser Overhead	Search Time(Sec.)
(i)	1.5588	16.15%	71.49%	40,000	112.30%	2.37776
(ii)	3.2514	33.68%	72.38%	40,000	8.40%[2]	1.21408[1]
(iii)	6.3932	66.22%	77.45%	40,000	828.40%	10.39808
(iv)	8.7684	90.82%	89.24%	40,000	246.40%	3.87968
(v)	9.2081	95.37%	89.05%	40,000	1170.36%	14.22803
(vi)	9.3743	97.10%	96.44%	40,000	1058.66%	12.97699
(vii)	9.6302	99.75%	98.50%	39,100	74.33%	1.908565
(viii)	9.6344	99.79%	98.79%	40,000	121.00%	2.4752
(ix)	9.6451	99.99%	98.73%	38,200	123.00%	2.385208
Best	(x)	(x)	(viii)	(x)	(ii)	(ii)
Theoretical values	9.6547	100.00%	100.00%	40,000 Max		

NB:

(1) This is the CPU time taken for 40,000 function evaluations in random search using Java with Symantec JIT 2.1 compiler in Window 95 on a 266 MHz Pentium processor with 64 MB RAM and 128K cache.

(2) This represents the testing overhead incurred by recording necessary figures on-line, which are required by calculating the benchmarks off-line. It should be zero in theory, i.e., if there exits no testing overhead.

From Table 3.2, it can be seen that (ix) is the best one for objective reached, optimality and total number of evaluations. For accuracy, (ix) is just below (viii), but better than others. Though for optimiser overhead and search time (ix) is not the best, far better than (v), (vi), (iii) and (iv), but worse than (ii), (i), (vii) and (viii), the cost is acceptable.

3.4 Control System Design Objectives and Indices

In the last sections, the evolutionary optimisation environment is established. But the application of the environment to the optimisation of control systems needs to be achieved by a good performance index, which can distinguish the best control system from others. Employing the index as objective function, optimisation of control system could be carried out. However, the first thing in our discussion is design specifications in control systems.

3.4.1 Specifications

Consider a generic unity negative feedback control system of a given plant $G(s)$. Refer to Figure 3.6 for the notation. Then, for the case where $F(s) = 1$ without loss of generality,

$$E(s) = R(s) - Y(s) = \frac{1}{1 + H(s)G(s)}\left[R(s) - G(s)D(s)\right]. \tag{3.23}$$

The ultimate (but theoretically unachievable) objective of a control system design is to find an $H(s)$ such

$$E(s) = 0, \forall s, \forall D(s) \tag{3.24}$$

or

$$. e(t) = \mathcal{L}^{-1}\{E(s)\} = 0, \quad \forall t, \forall d(t) \tag{3.25}$$

This means (3.24) or (3.25) needs to be satisfied regardless of plant uncertainties, which can be modelled in $D(s)$. Note that, in practical control system designs, strictly satisfying (3.24) or (3.25) is impossible.

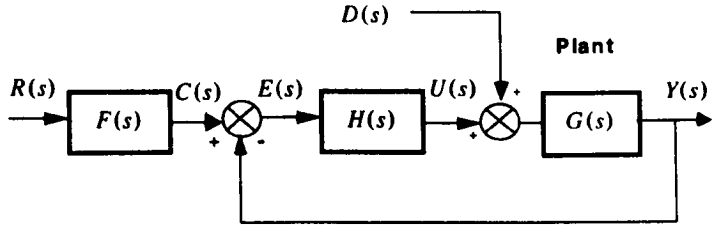

Figure 3.6 A feedback control system with model following

Hence, a performance index J: $\mathbf{R}^n \rightarrow \mathbf{R}^+$, is often used to measure *how close* the above ultimate objective is met, where n is the number of parameters that need to be determined in the design. For this, performance indices and specifications need to reflect the following qualitative requirements (Åström, 1991; Li and Häußler, 1996; Li *et al.*, 1996; Levine 1996):

1. Stability;
2. Excellent steady state accuracy in terms of small steady state errors;
3. Excellent transient response in terms of rise time, overshoots and settling-time;

4. Attenuation of load disturbance;

5. Sensitivity to measurement and robustness to model uncertainty.

Some of the specifications such as attenuation and sensitivity to measurement errors are conflicting, and others such as set-point following and load disturbance rejection are non-conflicting. For process control applications set-point following is often less important than load disturbance attenuation. Set-point changes are often made when production rate is altered. Furthermore, the response to set-point changes can be improved by introducing a set-point weighting.

3.4.2 Basic Performance Indices and Stability

Performance indices should reflect all specifications that need to be considered in practice. It should also address the issue of interpreting human engineers' perception of merit into a form that may be utilised for any controller design automation. Indices can be in the form of an overall composite objective or cost function with practical constraints, as commonly adopted by control engineers. They can also be in the form of multiple independent objectives, if a 'least commitment' principle is to be adopted at an early stage of design (Guan and MacCallum, 1996). These objectives should be easy to understand and utilise in computerised trial-and-error based search or optimisation. Thus, for a given application, a control system can be automatically designed or invented if the search and optimisation engine can accommodate these objectives.

3.4.2.1 Basic Performance Index in Time and Frequency Domains

In a design exercise, the closed-loop performance can be inverse-indexed conveniently by a basic 'cost function' in the form of an Euclidean norm:

$$J_f(H) = \|E(j\omega)\|_x \tag{3.26}$$

or

$$J_t(H) = \|e(t)\|_x . \tag{3.27}$$

As these performance indices are mainly for use with numerical simulations in the optimisation of controller, discrete summation over a bounded number of points is often used in place of integration. Note that all the linear metrics are equivalent, i.e., Euclidean norms are bounded linearly by one another. Hence, any one of the common norms may be used here. However, their selectivity in indexing can be different and an index based on L_∞, for example, loses selectivity completely for systems whose maximum error falls below $e(0)$. Two commonly used basic indices are (Åström, 1995):

1. Integral of Absolute Error (IAE)

$$J_{\text{IAE}} = \sum_t |e(t)| = \|e(t)\|_1 \ ; \tag{3.28}$$

2. Integral of Square Error (ISE)

$$J_{\text{ISE}} = \sum_t e^2(t) = \|e(t)\|_2^2 = \frac{1}{N}\|E(j\omega)\|_2^2 . \tag{3.29}$$

where N denotes the number of samples in both the time and the frequency domains. Note that the last equation is obtained from Parseval's energy equivalence theorem in both domains.

This implies that time and frequency domain indices can be equivalent and also that the design of a linear time-invariant (LTI) control system under this index can be unified in one

domain. Note also that minimising an index of an L_2 norm as in (3.29) is equivalent to minimising the root mean square (rms) error.

In the context of evolutionary computation, a performance index is often termed a 'fitness function' and 'maximising a fitness function' is more commonly encountered than 'minimising a cost function', although an evolutionary algorithm can do both maximisation and minimisation in one process. For convenience, however, a cost function can be converted easily into a fitness function by, for example, f: $\mathbf{R}^+ \rightarrow \mathbf{R}^+$

$$f(H) = \frac{1}{1+J(H)} \in (0, 1]. \tag{3.30}$$

3.4.2.2 Implicit Index to Robust Stability

For a linear control system, if the open-loop system is stable, then the Nyquist plot of the denominator in (3.23) does not encircle its origin in any way. This means that for relatively large stability margins, the denominator plot should be relatively far away from its origin and its magnitude should have a relatively large value. Hence, minimising the basic index indirectly leads to robust stability, owing to the norm equivalence. Note that L_∞ stable also means that the system is bounded-input and bounded-output stable.

However, for cases where specific gain and phase margins are necessarily required, minimising a basic index may not lead to satisfied results. Hence stability margins should be added to a composite index or form a second, independent index in non-committal multi-objective optimisation. An example of this is illustrated in Chapter 4.

3.4.3 Time Domain Specifications

3.4.3.1 Set-Point Following

Specifications on set-point following may include requirements on rise time, settling time, decay ratio, overshoot, and steady-state offset for step changes in set-point. The definitions for them are generally as

1. The rising time t_r is either defined as the inverse of the largest slope of the step response or the time it takes the step to pass from 10% to 90% of its steady state value (Dorf, 1992).

2. The settling time t_s is time it takes before the step response remains with $p\%$ of it steady state value. The value 2% is commonly used.

3. The decay ratio d is the ratio between two consecutive maxima of the error for a step change in set-point or load.

4 The overshoot o is the ratio between the difference between the first peak and the steady state value of the step response. In industry control applications it is common to specify an overshoot of 8% to 10%. But, in many situations, it is desirable to have an overdamped response.

5. The steady-state error e_{ss} is value of control error e in steady state. With the integral action in the controller, the steady-state error is always zero.

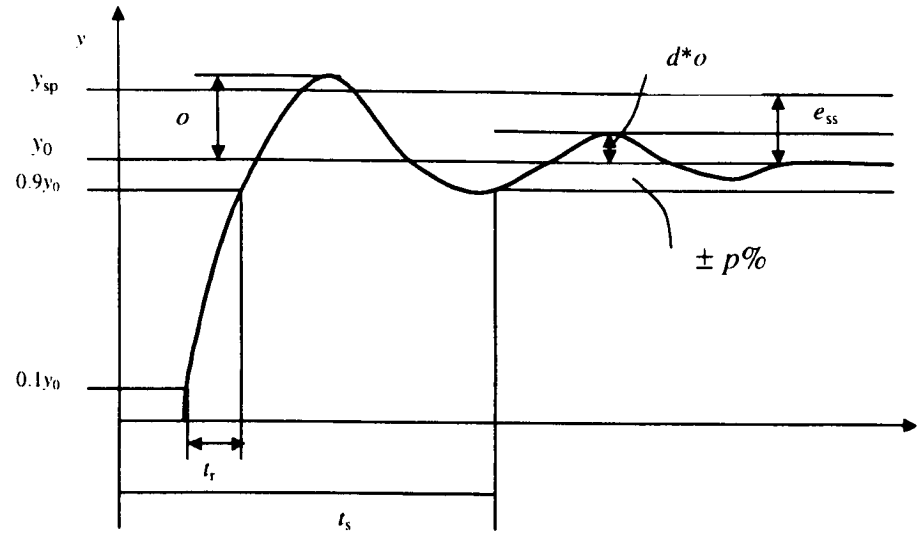

Figure 3.7 Practitioner's graphical specifications

3.4.3.2 Weighting Steady-State Errors by Time

If the command signal $r(t)$ is a step of size A, then

$$|e(\infty)| = \left| \frac{A}{1 + H(0)G(0)} \right|. \tag{3.31}$$

Hence, in either the time or the frequency domain, simple weighting against the steady-state error can be applied by adding an index building block (3.31) to a basic index in either the time or the frequency domain. Another simple 'weighting' against steady-state errors is to use the L_∞ norm for a basic index in the frequency domain or to use L_1 in the time domain. Since evolution does not require direct gradient-guidance, the weighting function design becomes much more relaxed and flexible.

Note that if the L_∞ norm is used to replace L_2, an emphasis is placed on the maximum magnitude of the spectrum that occurs near the dc frequency, where static steady-state errors contribute most. Similarly, the time domain cost can be in L_1, which tends to accumulate the absolute values of errors that are significantly contributed $\forall\ t \to \infty$.

The time itself forms a simple gradual, ramp weighting function. Inversely, dividing a frequency-domain index by frequency itself should also achieve a similar effect of emphasising the steady-state response. Time weighting is used in two commonly adopted indices (Åström, 1996):

3. Integral of Time Weighted Absolute Error (ITAE)

$$J_{\text{ITAE}} = \sum_t t\,|e(t)| \; ; \tag{3.32}$$

4. Integral of Time Weighted Square Error (ITSE)

$$J_{\text{ITSE}} = \sum_t t\,e^2(t) \; ; \tag{3.33}$$

and in (Zhuang and Atherton, 1993):

5. Integral of Square Time Weighted Square Error (ISTSE)

$$J_{\text{ISTSE}} = \sum_t t^2 e^2(t) \, . \tag{3.34}$$

3.4.3.3 Weighting Transients by Frequency

If suppressing overshoots and undershoots are required, weighting against the transient may be realised in either the time or the frequency domain by adding to a basic index:

$$|e(0)| = \left| \frac{A}{1 + H(\infty)G(\infty)} \right| = |A| \, . \tag{3.35}$$

Similarly, another simple 'weighting' against overshoots and undershoots is to use the L_1 norm for a basic index in the frequency domain or to use L_∞ in the time domain. When L_1 norm is used, it tends significantly to accumulate frequency response values $\forall\ \omega \rightarrow \infty$,

which are contributed most at transients. The L_∞ norm in the time domain places an emphasis on the maximum amplitude of errors, which usually occurs at $t \rightarrow 0$ for a 'hard-start' command such as a step.

Note that, for a 'hard' command, transients already contribute a relatively large amount of error and are, hence, seldom weighted in practice. However, the change of error instead of the error itself may be used to highlight the transient performance and/or to penalise chattering. This is equivalent to multiplying by frequency, as transients constitute high frequencies. Such indices are studied here:

6. Integral of Absolute Error Derivative (IAED)

$$J_{\text{IAED}} = \sum_t |\dot{e}(t)| = \|\dot{e}(t)\|_1 ; \tag{3.36}$$

7. Integral of Square Error Derivative (ISED)

$$J_{\text{ISED}} = \sum_t \dot{e}^2(t) = \|\dot{e}(t)\|_2^2 = \frac{1}{N}\|\omega E(j\omega)\|_2^2; \tag{3.37}$$

8. Integral of Time Weighted Absolute Error Derivative (ITAED)

$$J_{\text{ITAED}} = \sum_t t\,|\dot{e}(t)| ; \tag{3.38}$$

9. Integral of Time Weighted Square Error Derivative (ITSED)

$$J_{\text{ITSED}} = \sum_t t\,\dot{e}^2(t) ; \tag{3.39}$$

10. Integral of Square Time Weighted Square Error Derivative (ISTSED)

$$J_{\text{ISTSED}} = \sum_{t} t^2 \dot{e}^2(t) \ . \tag{3.40}$$

The index values of ten indices are plotted to the selectivity in terms of damping ratio in Section 3.4.6 for both hard-start and soft-start command.

3.4.4 Load disturbance

3.4.4.1 Direct Index

Load disturbances are disturbances that drive the process variables away from their desired values. Attenuation of load disturbances is of primary concern for process control. This is particularly the case for regulation problems where the processes are running in steady state with constant set-point for a long time. Load disturbances are often of low frequencies. Step signals are often used as prototype disturbances. The disturbances may enter the system in many different ways. If nothing else is known, it is often assumed that the disturbances enter at the process input. In Figure 3.6, D is the step input. Let e to be the error caused by a unit step disturbance at process. The integrated absolute error, which is defined by

$$J_{IAE} = \int_0^{\infty} |e(t)| dt \tag{3.41}$$

The criterion IAE is in many cases a natural choice, at least for control quality variables. Though, it was thought time consuming, with much fast computer now, the numerical calculation of it seems possible.

3.4.4.2 Implicit Index to Disturbance Rejection

Refer to Figure 3.6 again. The magnitude of the disturbance transfer to the closed-loop output is give by

$$\left\|\frac{Y(j\omega)}{D(j\omega)}\right\| = \left\|\frac{1}{1+H(j\omega)G(j\omega)}\right\| \|G(j\omega)\| \quad . \tag{3.42}$$

Comparing this with (3.23), it can be inferred that the load disturbance rejection is satisfied, if the basic index (3.24) or (3.25) is satisfied. Similarly, therefore, the rejection is maximised if a basic index is minimised, largely meeting Requirement 4. Note that, however, best set-point following does not necessarily mean best load disturbance rejection (Åström and Hagglund, 1995), which is highlighted in Chapter 4, when PID controllers are designed by different specifications.

3.4.5 Sensitivity

3.4.5.1 Sensitivity to Measurement Noise

Measurement noise is typically of high frequency. Care should always be taken to reduce noise by appropriate filtering. In QFT controller design, reduction of high frequency gain is the optimisation objective. So if in PID control, the derivative part of a PID controller is generally modified as

$$D(s) = K_p \frac{T_d s}{1+\frac{T_d s}{N}} e(s) \tag{3.43}$$

The high-frequency gain of such a PID controller is

$$K_{hf} = K_p(1+N) \tag{3.44}$$

N is typically chosen to be about 10.

3.4.5.2 Sensitivity to Process Characteristics

The controller parameters are typically matched to the process characteristics. Since the process may change, it is important that the controller parameters are chosen in such a way that the closed-loop system is not too sensitive to variations in process dynamics. There are many ways to specify the sensitivity. Many different criteria are conveniently expressed in terms of the Nyquist curve of the loop transfer function $G_l(s) = H(s)G(s)$. Maximum sensitivity can be described as

$$M_s = \max_{0 \le \omega \le \infty} \left| \frac{1}{1 + H(j\omega)G(j\omega)} \right| \tag{3.45}$$

The quantity M_s is simply the inverse of the shortest distance from the Nyquist curve to the critical point −1. Reasonable values of M_s are in the range from 1.3 to 2.

From the Figure 3.8, it is clear that M_s guarantees that the distance from the critical point to the Nyquist curve is always greater than $1/M_s$. Gain margin and phase margin are defined as

$$A_m = \frac{1}{\left| G_l(j\omega_u) \right|} \tag{3.46}$$

$$\varphi_m = \pi + \arg G_l(j\omega_g) \tag{3.47}$$

where the ultimate frequency ω_u is the frequency where $\arg G_l(j\omega) = -\pi$ and the gain cross-over frequency ω_g is the frequency where $\left| G_l(j\omega) \right| = 1$. Typical values of φ_m range 30° to 60°, and gain margin can vary from 2 to 5.

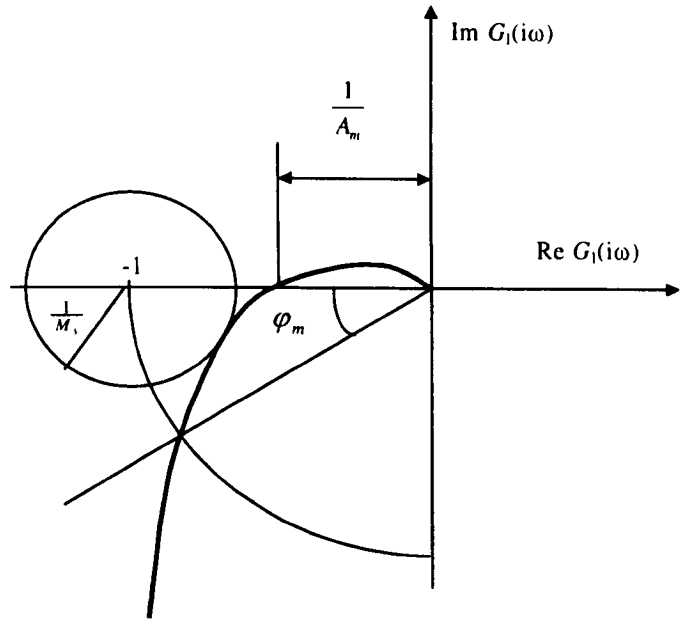

Figure 3.8 Definition of maximum sensitivity M_s, gain margin A_m, and phase margin φ_m

3.4.5.3 Implicit Index to Robustness Against Plant Uncertainty

In Figure 3.8, the magnitude of the sensitivity of the closed-loop transfer function to the plant transfer function is given by

$$\left\| S_G^{G_c} \right\| = \left\| \lim_{\Delta G \to 0} \frac{\Delta G_c(j\omega)/G_c(j\omega)}{\Delta G(j\omega)/G(j\omega)} \right\| = \left\| \frac{1}{1+H(j\omega)G(j\omega)} \right\|. \tag{3.48}$$

Hence, the closed-loop sensitivity to the plant uncertainty is minimised if the basic index is minimised. Often, the L_∞ norm is used here to represent maximum sensitivity. In Chapter 4, an example considering sensitivity in PID control systems design is presented.

3.4.6 Merit and Selectivity of Indices

As an LTI system can generally be decomposed of first and second order subsystems, its dominant dynamics are hence often represented in practice by a second order system. Suppose that a design results in an overall closed-loop system that behaves close to a unity-gain second-order system. Then the performance of the closed-loop system is regarded as too

sluggish if it behaves 'over-damped'. If it is too much 'under-damped', however, the transient is unsatisfactory. Often, the damping factor, ζ, is regarded as 'good' if it is of a value between that resulting in a critically-damped system ($\zeta = 1.0$) and that resulting in a resonance ($\zeta = 0.707$).

3.4.6.1 Hard-Start Command

Controllers obtained by minimising different indices could result in different damping ratios. Hence, the ability of an index in selecting an optimal controller that minimises the index should be assessed. Refer to (Graham and Lathrop, 1953; Zhuang and Atherton, 1993) for IAE, ISE and ITAE. The selectivity of ITSE and ISTSE and the derivative versions of the all five indices are compared here.

In this regard, index values resulting from step following are studied here and are plotted to the selectivity in terms of damping ratio in Figure 3.9. It can be seen that, if the resultant closed-loop system is of a second-order dominant, as found in most practical control systems, the use of different indices results in a damping ratio ranging from 0.50 (ISE) to 1.00 (ISTED), extending to the infinity. In optimisation, clearly, the use of the ISTSE and ITAE indices would offer the sharpest selectivity, at $\zeta = 0.67$ and 0.75, respectively. An ITAE selected controller should offer a high and near-resonant damping. Nevertheless, different performance indices should be used for different purposes. For example, combining different indices together should provide a composite index that meets different needs of a design. But the index just with derivative error can't work independently, since it can make the designed system to be fixed to zero.

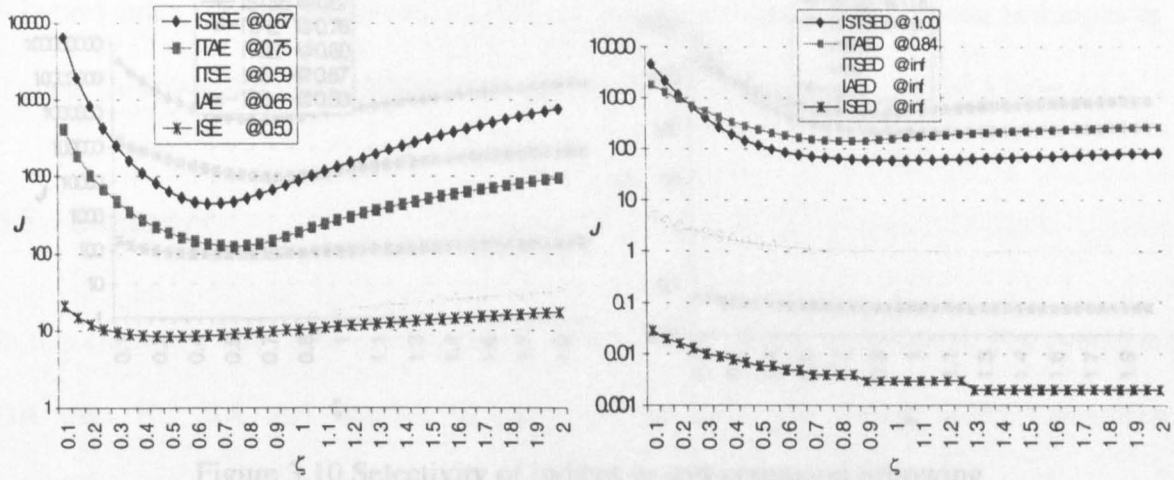

Figure 3.9 Selectivity of indices in terms of damping ratios

3.4.6.2 Soft-Start Command

Refer to $F(s)$ and $C(s)$ in Figure 3.6. In practice, a step response $C(s)$ of a critically damped second-order system $F(s)$, as opposed to the step $R(s)$ itself, is often used as a 'soft-start' command to follow, i.e., the dynamics of the closed-loop system is desired to follow a critically damped system $F(s)$. This 'model-following' control strategy (Åström 1996) is to avoid sharp acceleration in course-keep or aircraft control, for example. This is also to avoid actuator saturation and infinite current is not practically available to support a hard-command.

To study index selectivity for such applications without loss of generality, suppose that the natural frequency of the model to follow is ten times higher than that of the plant to be controlled. The results are shown in Figure 3.10. As can be seen, the selectivity of the indices almost remains the same.

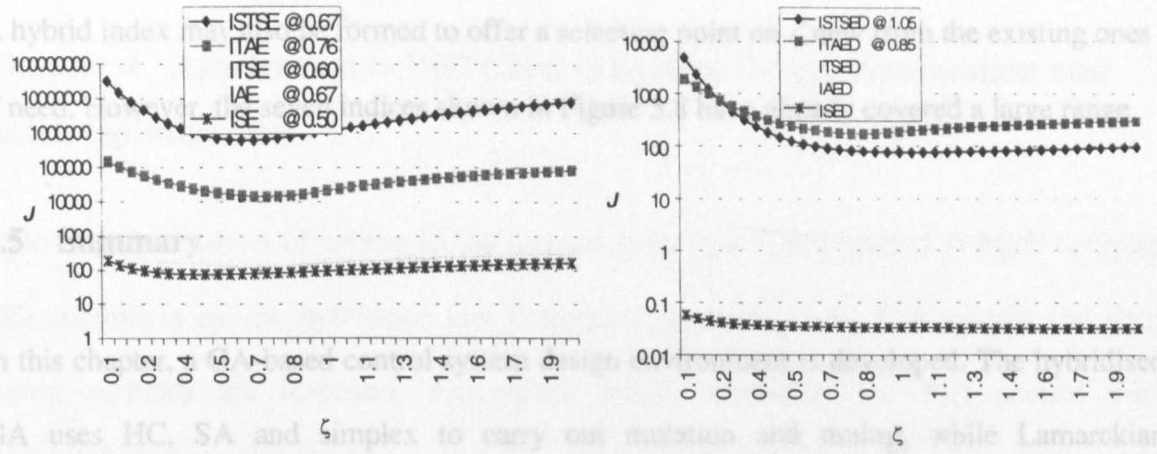

Figure 3.10 Selectivity of indices in soft-command following

3.4.7 Reconciling Accuracy and Chattering with Hybrids

It has been discussed that the steady-state response can be emphasised by time weighting and transients by frequency. These two weightings can thus be combined together to tackle both the steady-state and transient problems. An example of hybridised index for this is:

$$J_{\mathrm{TF}} = \sum_{t} te^2(t) + \sum_{\omega} \omega E^2(\omega) . \tag{3.49}$$

Here the frequency weighting may be replaced by derivatives. A simple such hybrid that places an emphasis on both tracking accuracy and actuator chattering is:

$$J_{\mathrm{H}} = J_{\mathrm{ITSE}} + J_{\mathrm{ITSED}} \tag{3.50}$$

which is shown to be very effective in evolving control systems (Li *et al.*, 1995 and 1996a). In this thesis, this index is employed in Chapter 6 to evolve a neurocontroller. Another hybrid example may be constructed in the same domain by multiplying the basic index with a 'notch filter':

$$J_{\mathrm{Notch}} = \sum_{\omega} \left(\omega + \frac{1}{\omega} \right) E(j\omega) E(-j\omega) . \tag{3.51}$$

A hybrid index may also be formed to offer a selecting point on ζ new from the existing ones if need. However, the seven indices shown in Figure 3.8 have already covered a large range.

3.5 Summary

In this chapter, a GA-based control system design environment is developed. The hybridised GA uses HC, SA and simplex to carry out mutation and tuning, while Lamarckian inheritance is introduced to replace the selection and crossover. Java technologies are adopted in the development of EA based optimal control system design automation environment. Therefore, the environment can be portable in the Internet.

Benchmark test principles of a GA algorithm are presented in detail. The concepts such as optimality, accuracy, sensitivity, convergence and optimiser overhead are proposed. For a benchmark test problem, various algorithms are tested. It confirms that the hybridised algorithm developed in this thesis is better.

Specifications such stability, transient response, steady state error, robustness and sensitivity in control systems are detailed. Based on specification for a control system, performances of all types of indices including those from the error and the derivative of error are investigated. In addition, soft model following cases are investigated as well. In the following chapters, the environment developed here is employed as automation design suite to achieve or improve different controllers with practical constraints, i.e., PID, QFT, neural controllers.

Chapter 4 Application to PID Control System Design Automation and Batch Optimisation

Following the analysis of indices in last chapter, some results are applied to batch optimise PID controllers by the hybridised GA design environment. Here, PID control and their tuning methods are reviewed. Automated design techniques for PID control with specifications in gain and phase margins are developed. Differences between PID and PI control are highlighted. Comparison with results of Åström and Hägglund 's (1995) method is also given.

4.1 Introduction

As highlighted in Chapter 1, PID is still very popular in industrial control. In a PID controller, the control action is generated as the sum of three terms, namely,

$$u(t) = u_p(t) + u_i(t) + u_d(t) \tag{4.1}$$

where u is the control variable, and u_p is the proportional part, u_i the integral part and u_d the derivative part. Not that, a generic practical PID controller is different from this simple version.

4.1.1 Proportional Control

The proportional control part is a simple feedback

$$u_p(t) = K_p e(t) \tag{4.2}$$

where e is the control error, and K_p is the controller gain. The error is defined as the difference between the set-point y_{sp} and the process output y, i.e.,

$$e(t) = y_{sp}(t) - y(t) \tag{4.3}$$

In many cases, u_p is modified as

$$e(t) = K_p\left(by_{sp}(t) - y(t)\right) \tag{4.4}$$

where b is called set-point weight, it can influence the set-point following without impact on load disturbance rejection. (Åström and Hägglund, 1995 and Seborg *et al.*, 1989).

4.1.2 Integral Control

Since the proportional control always gives a type 0 system steady-state error (Dorf, 1992), the integral action is introduced to remove this. The integral action has the form:

$$u_i(t) = \frac{K_p}{T_i}\int_t e(s)ds \tag{4.5}$$

This idea is simply that the control action should be taken even if there is a very small error, provided the error is the same sign over a long period. It can eliminate the steady-state error.

However, there is a 'wind-up' problem caused by the integral action. If a practical actuator that realises the control action has a range limit (Figure 4.1), then the integrator may well saturate. The future correction is ignored until the saturation is offset. This causes low-frequency oscillations and my may lead to instability (Åström, 1991 and Li *et al.*, 1998).

A usual measure taken to counteract this effect is 'anti-windup'. This is realised by negative-feeding the excess amount of the integral action back to the integrator so that saturation is taken out. A simple anti-windup is realised by modifying the integral action to:

$$\begin{aligned} U_i &= K_p\frac{1}{T_i s}E(s) - \frac{1}{\gamma T_i s}\left[U(s) - \tilde{U}(s)\right] \\ &= \frac{1}{T_i s}\left[K_p E(s) - \frac{U(s) - \tilde{U}(s)}{\gamma}\right] \end{aligned} \tag{4.6}$$

where $\tilde{U}(s)$ represents the saturated control action and γ is a correcting factor.

4.1.3 *Derivative Control*

The derivative action control is used to provide predictive action (Åström, 1991 and Li *et al.*, 1998). A simple form is

$$u_d = K_p T_d \frac{de(t)}{dt} \tag{4.7}$$

The combination of proportional and derivative action is then

$$u_p(t) + u_d(t) = K_p\left[e(t) + T_d \frac{de(t)}{dt}\right] \tag{4.8}$$

This means that the control action is based on linear extrapolation of the error T_d time units ahead. Parameter T_d is called derivative time. The main difference between a PD controller and a more complex controller is that a dynamic model can have better prediction than linear extrapolation.

In many practical applications, the set-point is a constant. This means that the derivative of the set-point is zero except for those time instances when the set-point is changed. At that moment, the derivative action becomes infinitely large. In addition, for the requirement of reduction of high frequency gain, which has been discussed in Chapter 3, a better realisation of the derivative action is,

$$U_d(s) = \frac{K_p T_d s}{1 + \frac{sT_d}{N}}\left[cY_{sp}(s) - Y(s)\right] \tag{4.9}$$

Supposing $e(s) = cY_{sp}(s) - Y(s)$, (4.9) is the same as (4.7) plus a low pass filter. The low pass filter can reduce the high frequency gain. Parameter c is a set-point weighting, which is often set to zero. It does not have impact on load distance rejection as well. Except the typical low pass filter to reduce the high frequency gain, a nonlinear *median filtering* technique was

reported by Li *et al.* (1998). This method can filter out the unusual spike type of noise more easily.

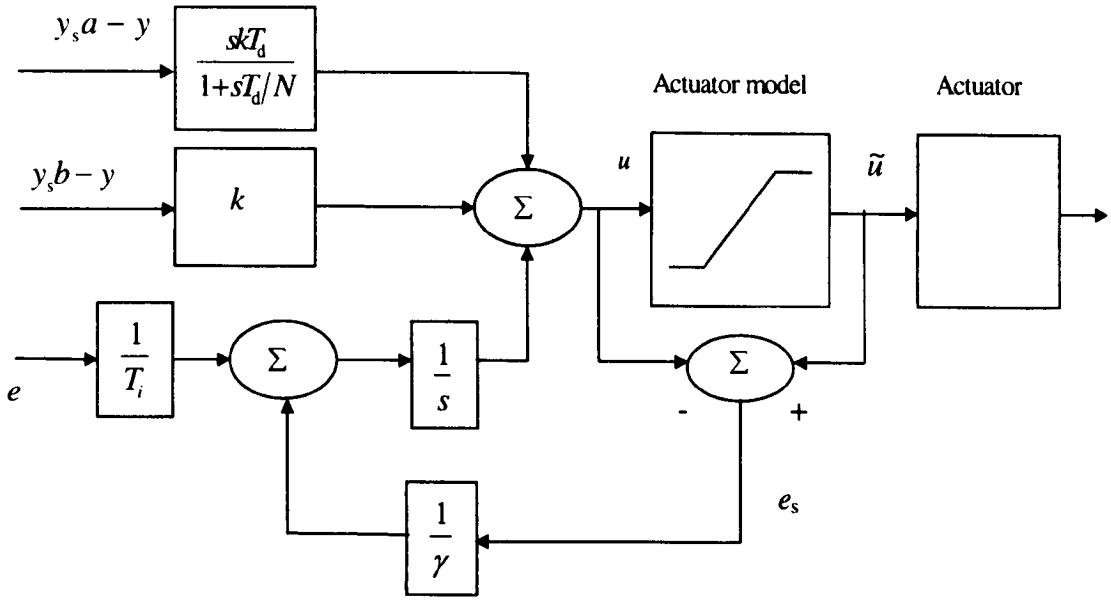

Figure 4.1 A generic practical PID controller

4.2 PID Controller Design Methods

During the past 50 years, many methods for determining PID controller coefficients have been developed. Some methods employ information from the open-loop step response, other methods use some knowledge of the Nyquist curve of the plant, for example, the Ziegler-Nichols frequency response method. These methods use simple tuning laws to determine the PID controller coefficients, and some commercial autotuners based on these methods have been available since 1981. However, these tuning methods use only a small amount of information about the dynamic behaviour of the system, and often do not provide good tuning. For instance, the Ziegler-Nichols tuning laws usually result in rather oscillatory set-point responses, although they have been widely used as heuristic methods to determine PID controller coefficients in the process control industry. It is the reason why there is an investigation of PID control.

4.2.1 Ziegler-Nichols and Related Methods

Ziegler and Nichols (1942) developed their rules from experiments and by analysing various industrial processes. Using the integral of absolute error criterion with a unit step response, they found that controllers adjusted according to the rules usually have a step response that was oscillatory but with enough damping so that the second overshoot is less than 25% of the first (peak) overshoot. This is the quarter-decay criterion, and is sometimes used as a specification.

4.2.1.1 Step Response Method

This method used relies on the fact that many processes have an open-loop step response of the form shown in Figure 4.2. This process signal is characterised by 3 parameters namely L, τ and A, where L is the delay; τ is the time constant, which is the inverse of the maximum gradient; and A is the steady-state gain of the plant. The Ziegler-Nichols recommendations are given in the Table 4.1 in terms of these parameters.

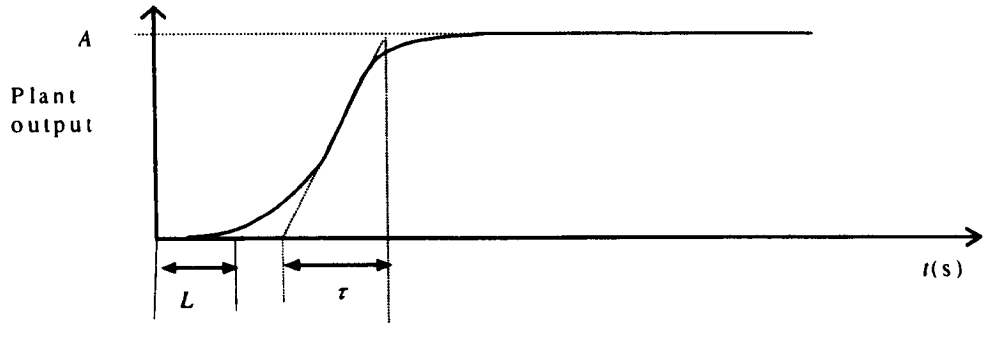

Figure 4.2 Open-loop response of a plant

Table 4.1 PID controller coefficients obtained from Ziegler-Nichols step response method

Controller Type	K_p	T_i	T_d
P	$\frac{\tau}{AL}$	—	—

PI	$0.9\frac{\tau}{AL}$	$3L$	—
PID	$1.2\frac{\tau}{AL}$	$2L$	$0.5L$

4.2.1.2 Frequency Response Method

This method is also based on a simple characterisation of the process dynamics. The design is based on knowledge of the point on the Nyquist curve of the process transfer function $G(s)$, where the Nyquist curve intersects the negative real axis. For historical reasons this point is characterised by the parameter K_u and T_u, which are called the *ultimate gain* and *ultimate period*. These parameters can be determined in the following way. Connect a controller to the process, set the parameters so that control action is proportional, i.e., $T_d = 0$, $T_i = \infty$. After approaching the state status, increase the gain slowly until the process starts to oscillate. K_u is the gain when this occurs, and T_u is the period of the oscillation. The controller coefficients are given in Table 4.2.

Table 4.2 PID controller coefficients obtained from Ziegler-Nichols frequency response method

Control Type	K_p	T_I	T_d
P	$0.5K_u$	----	-----
PI	$0.4K_u$	$0.8T_u$	-----
PID	$0.6K_u$	$0.5T_u$	$0.125T_u$

4.2.1.3 Modified Ziegler-Nichols Methods

There have been many suggestions of modifications of the Ziegler-Nichols methods. Chien, Hrones and Reswick (CHR) (Chien *et al.*, 1952) changed the step response method to give

better damped closed-loop systems. They proposed to use "quickest response without overshoot" or "quickest response with 20% over shoot" as design criteria. They also found that tuning for set-point tracking is different from load disturbance response rejection. Two sets of coefficients for different overshoot specifications are given in Table 4.3.

Table 4.3 CHR method load for disturbance rejection response

Overshoot	0%			20%		
Controller	K_p	T_I	T_d	K_p	T_I	T_d
P	$0.3\frac{\tau}{AL}$	--------	---------	$0.7\frac{\tau}{AL}$	----------	---------
PI	$0.6\frac{\tau}{AL}$	$4L$	---------	$0.7\frac{\tau}{AL}$	$2.3L$	---------
PID	$0.95\frac{\tau}{AL}$	$2.4L$	$0.42L$	$1.2\frac{\tau}{AL}$	$2L$	$0.42L$

4.2.2 Analytical Tuning Methods

There are several analytical tuning methods where the controller transfer function is obtained from the specifications by direct calculation. Let G_p be the transfer functions of the process, H for the controller and G_c for the closed-loop. The closed-loop transfer function obtained with error feedback is then

$$G_c = \frac{G_p H}{1 + G_p H} \tag{4.10}$$

Solving this equation,

$$H = \frac{1}{G_p} \frac{G_c}{1 - G_c} \tag{4.11}$$

If the closed-loop transfer function G_c is specified and G_p is known, it is thus easy to compute H. The essential problem is to find reasonable ways to determine H based on engineering specifications of the system.

It follows from (4.10) that all process poles and zeros are cancelled by the controller. Åström and Hägglund (1995) argued that the method should not be applied when the process has poorly damped poles and zeros. The method also gives a poor load disturbance response when slow process poles are cancelled. Tuning methods developed from this scheme including λ-Tuning, Haalman and Internal Model Controller (IMC) (Aström, 1993).

4.2.3 Optimisation Based Methods

Since Zielger-Nichols and modified methods are not satisfactory in many cases (too much transients) (Zhuang and Atherton, 1993; Voda and Landau, 1995), and analytical methods do not work with load disturbances, people have searched novel ways to tune PID controllers. Zhuang and Atherton's solution (1993) is to design tuning methods using integral performance criteria, e.g. IST^{2}E. These tuning rules are also used in more recent papers (Majhi and Atherton, 1999; Atherton 2000). In the thesis, it is found that the optimisation method works very well for the performance of set-point following, and it has acceptable robustness in terms of the gain margin and phase margin, and sensitivity as well. But, for the design of the rejection of load disturbance, the results of this method can lead to too small stability margins, which was pointed out by Åström and Hägglund (1995). Tuning rules achieved from optimisation method given by Zhuang and Atherton (1993) are in Table 4.4. Corresponding sensitivity and stability margins are given in Figures 4.3 to 4.5.

Table 4.4 Tuning formulae for PID control obtained by the step response method. The table gives coefficients of functions of the form $AK_p = a_1\left(\frac{L}{\tau}\right)^{b_1}$, $T_i/\tau = \frac{1}{a_2 + b_2(L/\tau)}$ (set point following), $T_i/\tau = \frac{1}{a_2}\left(\frac{L}{\tau}\right)^{-b_2}$ (load disturbance rejection) and $T_d/\tau = a_3\left(\frac{L}{\tau}\right)^{b_3}$ for the model given in Figure 4.2.

	a_1	B_1	a_2	b_2	a_3	c_3
Set point following	0.968	-0.904	0.977	-0.263	0.316	0.892
Load disturbance	1.531	-0.960	0.971	-0.746	0.413	0.933

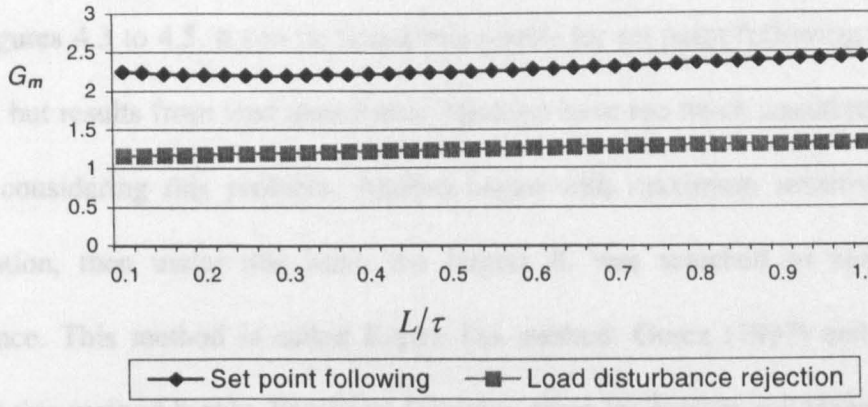

Figure 4.3 Gain margin resulting from Zhuang and Atherton's design formulae

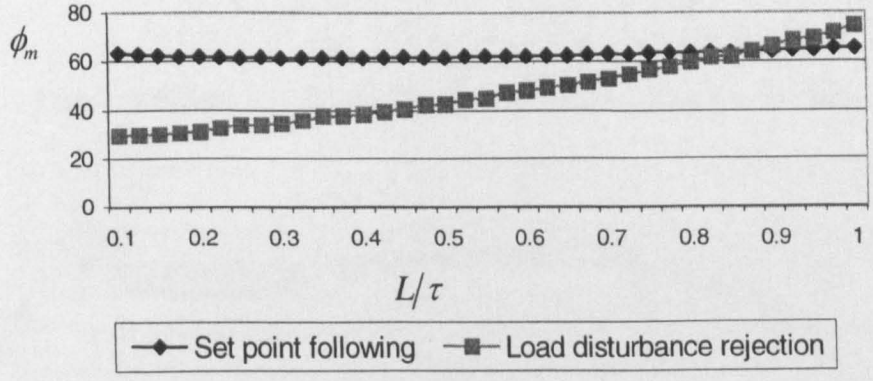

Figure 4.4. Phase margin resulting from Zhuang and Atherton's design formulae

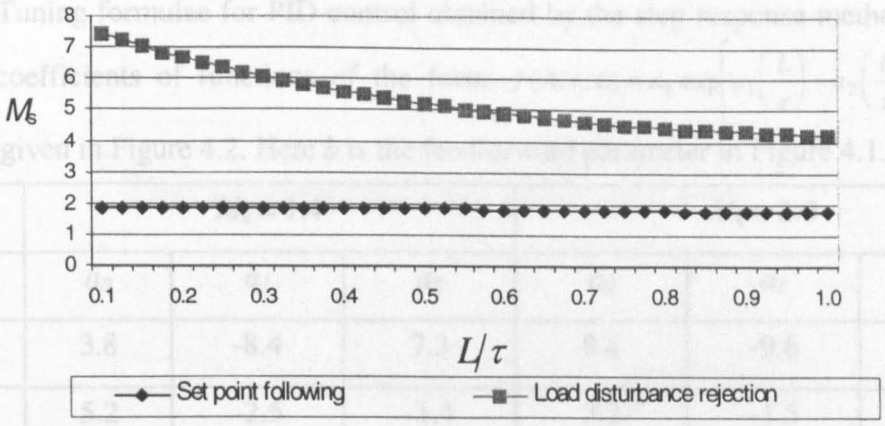

Figure 4.5 Maximum sensitivity resulting from Åström 's design formulae

From Figures 4.3 to 4.5, it can be found that results for set point following have acceptable stability, but results from load disturbance rejection have too much sensitivity (e.g. Ms > 4). Having considering this problem, Åström began with maximum sensitivity as a design specification, then under this limit, the largest K_i was searched to minimise the load disturbance. This method is called Kappa-Tau method. Gorez (1997) and Becerra (2000) regarded this method highly. Empirical formulae given by Åström and Hägglund (1995) and Levine (1996) are in Table 4.5. Results of the sensitivity and margins are given in Figures 4.6 to 4.8.

Table 4.5 Tuning formulae for PID control obtained by the step response method. The table gives coefficients of functions of the form $f(A,\tau,L)=a_0\exp\left[a_1\left(\frac{L}{\tau}\right)+a_2\left(\frac{L}{\tau}\right)^2\right]$ for the model given in Figure 4.2. Here b is the feedforward parameter in Figure 4.1.

	$M_s = 1.4$			$M_s = 2.0$		
	a_0	a_1	a_2	a_0	a_1	a_2
AK_p	3.8	-8.4	7.3	8.4	-9.6	9.8
T_i/L	5.2	-2.5	-1.4	3.2	-1.5	-0.93
T_i/τ	0.46	-2.8	-2.1	0.28	3.8	-1.6
T_d/L	0.89	-0.37	-4.1	0.86	-1.9	-.044
T_d/τ	0.077	5.0	-4.8	0.076	3.4	-1.1
B	0.41	0.18	2.8	0.22	0.65	0.051

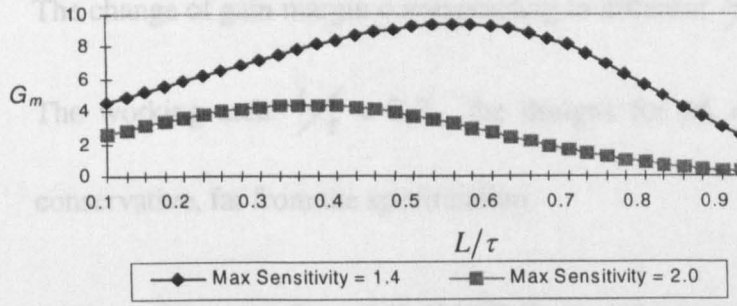

Figure 4.6 Gain margin resulting from Åström's design formulae

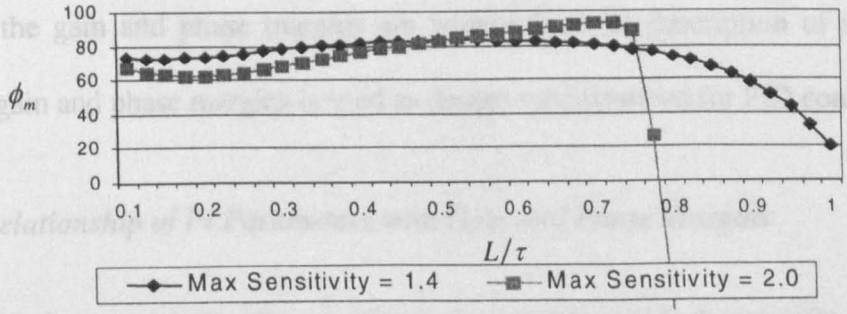

Figure 4.7. Phase margin resulting from Åström 's design formulae

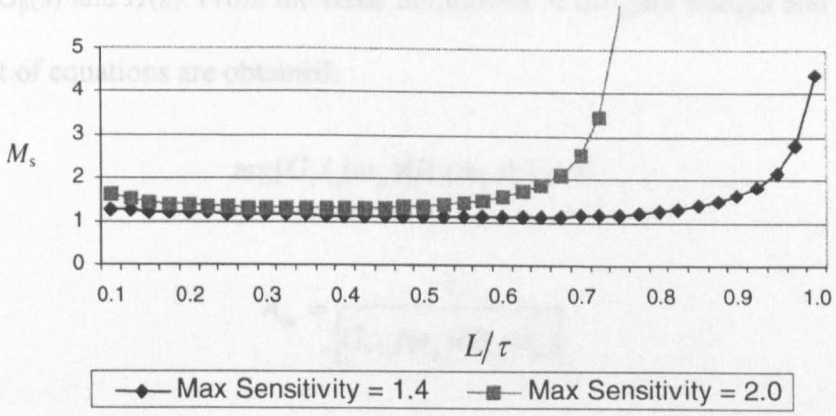

Figure 4.8 Maximum sensitivity resulting from Åström 's design formulae

From the Figure 4.6 to 4.8, it is easy to find that there are three defects in the empirical formulae in stability performance.

1. When L/τ approaches to 1, the formulae do not work, and the resultant systems have too much sensitivity.
2. The change of gain margin corresponding to different L/τ is too much.
3. The working area $L/\tau < 0.7$, the designs for M_s = 2.0 seem too much conservative, far from the specification.

4.3 Gain and Phase Margins Based Full PID Design Automation

Because the results from specifications on the maximum sensitivity have their unsatisfactory parts, and the gain and phase margins are widely used for description of stability, in this thesis, the gain and phase margins is used as design specifications for PID control.

4.3.1 Relationship of PI Parameters with Gain and Phase Margins

For FOPDT plant model, if a PI controller is designed to satisfy the specific gain margin and phase margin, through solving a set of equations, K_p and T_i, could be achieved. Denote the specified gain and phase margins by A_m and ϕ_m, and the process and controller transfer

function by $G_p(s)$ and $H(s)$. From the basic definitions of the gain margin and phase gain, the following set of equations are obtained:

$$\arg[G_c(j\omega_p)G(j\omega_p)] = -\pi, \tag{4.12}$$

$$A_m = \frac{1}{\left|G_c(j\omega_p)G(j\omega_p)\right|}, \tag{4.13}$$

$$\left|G_c(j\omega_g)G(j\omega_g)\right| = 1, \tag{4.14}$$

$$\phi_m = \arg\left[G_c(j\omega_g)G_p(j\omega_g)\right] + \pi. \tag{4.15}$$

Then include PI controller and the process,

$$G_c(s) = k_c(1 + \frac{1}{sT_i}), \tag{4.16}$$

$$G_p(s) = \frac{k_p}{1+s\tau}e^{-sL}. \tag{4.17}$$

Substituting (4.16) and (4.17) into (4.12) – (4.15) gives

$$\frac{1}{2}\pi + \arctan\omega_p T_i - \arctan\omega_p \tau - \omega_p L = 0, \tag{4.18}$$

$$A_m k_c k_p = \omega_p T_i \sqrt{\frac{\omega_p^2\tau^2 + 1}{\omega_p^2 T_i^2 + 1}}, \tag{4.19}$$

$$k_c k_p = \omega_g T_i \sqrt{\frac{\omega_g^2\tau^2 + 1}{\omega_g^2 T_i^2 + 1}}, \tag{4.20}$$

$$\phi_m = \frac{1}{2}\pi + \arctan\omega_g T_i - \arctan\omega_g \tau - \omega_g L. \tag{4.21}$$

Finally, through the approximation of function arctan, a solution of K_p, and T_i was given by Ho *et al.* (1995),

$$K_p = \frac{\omega_p \tau}{A_m k_p}, \tag{4.22}$$

$$T_i = (2\omega_p - \frac{4\omega_p^2 L}{\pi} + \frac{1}{\tau})^{-1}, \tag{4.23}$$

where

$$\omega_p = \frac{A_m \phi_m + \frac{1}{2}\pi A_m (A_m - 1)}{(A_m^2 - 1)L}. \tag{4.24}$$

4.3.2 Improvement with Derivative Action

Although Ho's method is very easy for us to get a PI controller to satisfy the specific gain margin and phase margin, derivative control is not considered in this solution. Predictive action played by the derivative control has been explained in Section 4.1.2. It is expected that the derivative action could play a part for the system to have the best performance under the limitation of the gain margin and phase margin. Li *et al.* (1998) has investigated this problem. In general, adding a derivative term to the proportional term means increasing the gain by:

$$\left|1 + j\omega T_d\right| = \sqrt{1 + \omega^2 T_d^2} > 1, \quad \forall \omega \tag{4.25}$$

times, which alone tends to decrease the gain margin. But the phase is improved by:

$$\angle\left(1 + j\omega T_d\right) = \arctan\frac{\omega T_d}{1} \in \left[0, \pi/2\right], \quad \forall \omega \tag{4.26}$$

From (4.25) and (4.26), there is possible improvement of stability margins to be achieved by involving the derivative term, especially, when there is apparent time delay in the plant to be controlled, because the time delay will lead to the reduction of phase margin.

4.3.3 Design Results and Comparison

Since it has been discussed that load disturbance rejection is more important than set-point following (Åström and Hägglund), and Zhuang and Atherton's method has a good result for set point following, here, just load disturbance rejection is considered as the objective to be achieved. And ITAE is used as the index, instead of IAE, because IAE could lead to underdamped design results, which has been detailed in Chapter 3. But what should be noted is that the optimisation is under constraints of gain and phase margins. The hybridised GA based environment developed in Chapter 3 is used to deal with the problem. The cost function designed here is,

$$J = \begin{cases} J_{\text{ITAE}} + K\left(\left(A_m^d - A_m\right) + \left(\phi_m^d - \phi_m\right)\right) & A_m^d > A_m \vee \phi_m^d > \phi_m \\ J_{\text{ITAE}} & \text{otherwise} \end{cases}, \tag{4.27}$$

where A_m^d and ϕ_m^d are the desired gain margin and phase margin, A_m and ϕ_m are candidates' corresponding margins. K is a big constant compared with J_{ITAE} in order to make sure the stability requirement to be satisfied. Because there are just three coefficients in the system, the hybridised GA finishes the optimisation in 5 generations with the population size 50. Optimisation results are given in Tables 4.6 to 4.11. Time domain simulation are given in Figures 4.9 to 4.20 (the darker line for PID):

Case 1: $G_p(s) = e^{-0.1s} / (1+s)$

Table 4.6 PID controller coefficients from the hybridised GA with $L/\tau = 0.1$

Specified					Resultant			
A_m	ϕ_m	K	T_i	T_d	A_m^*	ϕ_m^*	IATE	IAE
3	45	5.43	0.35	0.016	2.99	44.78	0.029	0.065
5	45	3.29	0.32	0.011	4.91	44.71	0.079	0.121
3	60	4.65	0.43	0.045	3.02	59.25	0.056	0.096
5	60	3.24	0.52	0.015	5.00	59.45	0.117	0.162

Table 4.7 PI controller coefficients from the Ho's formulae with $L/\tau = 0.1$

Specified				Resultant			
A_m	ϕ_m	K	T_I	A_m^*	ϕ_m^*	IATE	IAE
3	45	4.91	0.35	2.91	41.6	0.033	0.073
5	45	0.295	0.35	4.83	46.6	0.097	0.142
3	60	5.24	1.00	3.00	60.0	0.231	0.191
5	60	3.05	0.54	4.94	58.5	0.132	0.178

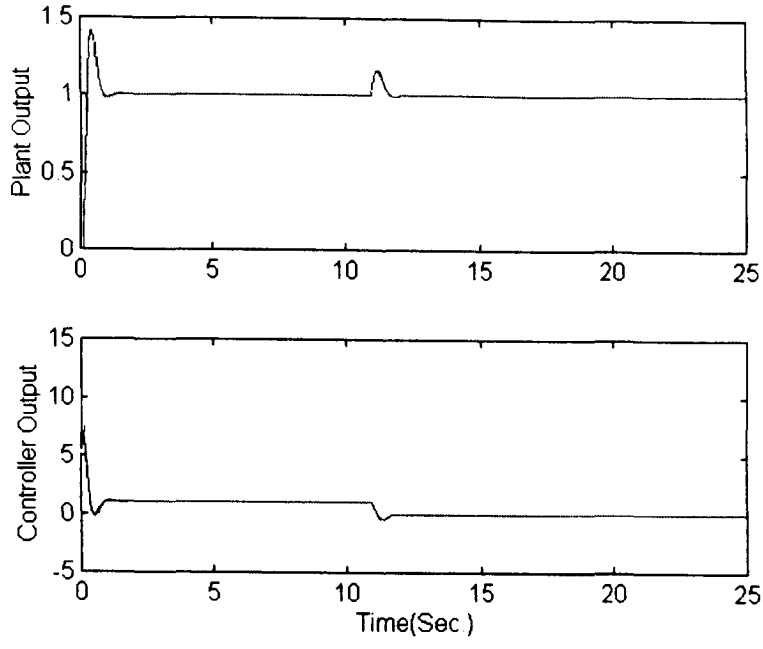

Figure 4.9 Comparison between PID and PI with $A_m = 3$, $\phi_m = 45$ and $L/\tau = 0.1$

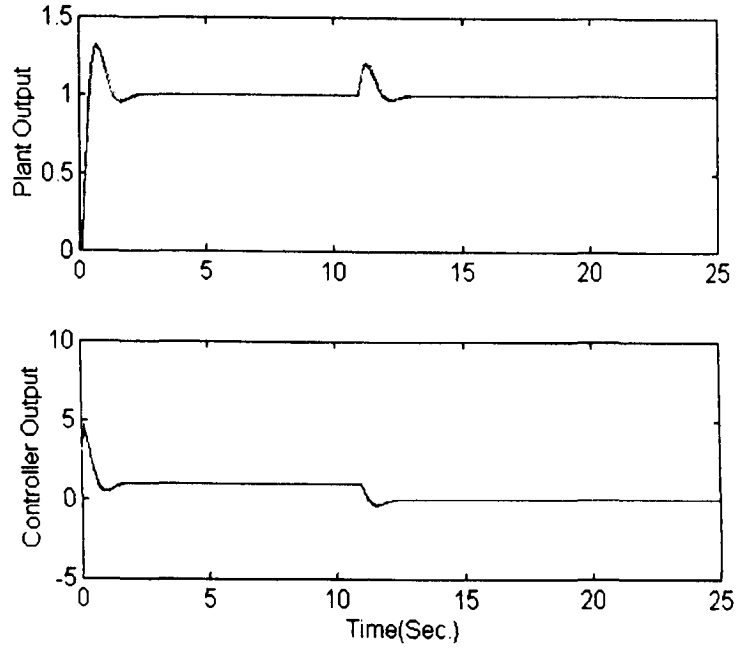

Figure 4.10 Comparison between PID and PI with $A_m = 5$, $\phi_m = 45$ and $L/\tau = 0.1$

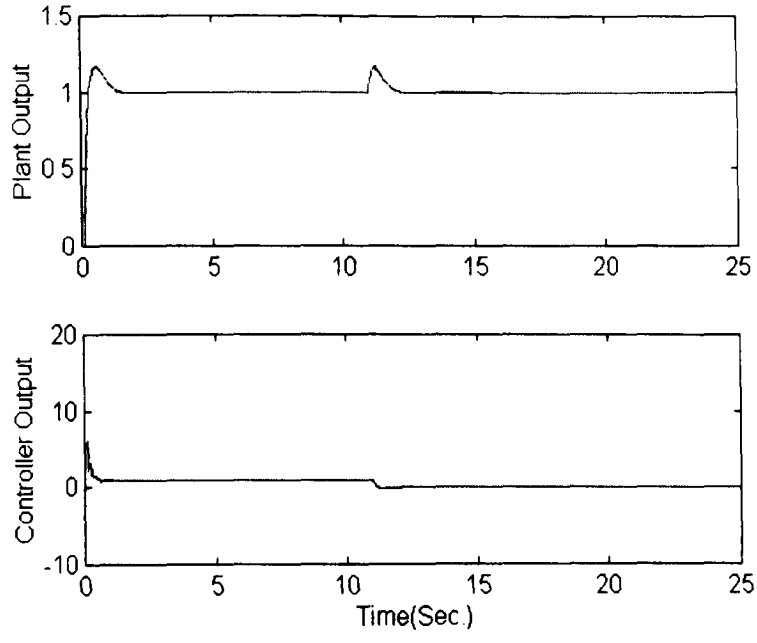

Figure 4.11 Comparison between PID and PI with $A_m = 3$, $\phi_m = 60$ and $L/\tau = 0.1$

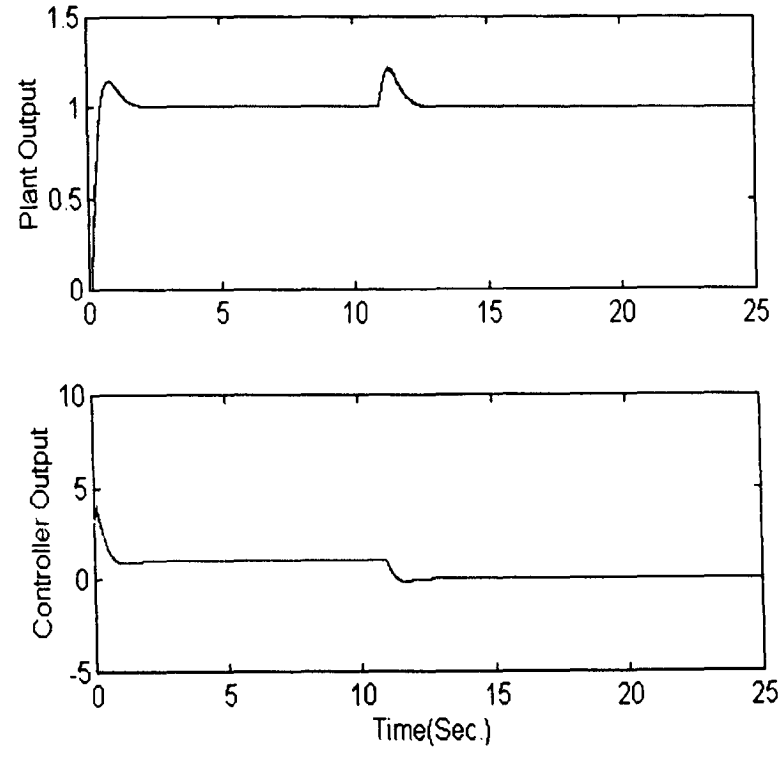

Figure 4.12 Comparison between PID and PI with $A_m = 5$, $\phi_m = 60$ and $L/\tau = 0.1$

Case 2: $G_p(s) = e^{-0.4s}/(1+s)$

Table 4.8 PID controller coefficients from the hybridised GA with $L/\tau = 0.4$

Specified					Resultant			
A_m	ϕ_m	K	T_i	T_d	A_m^*	ϕ_m^*	ITAE	IAE
3	45	1.53	0.69	0.066	2.94	45.2	0.749	0.520
5	45	0.92	0.60	0.084	5.00	44.9	1.561	0.821
3	60	1.50	0.93	0.100	3.02	59.3	0.974	0.620
5	60	0.92	0.75	0.080	5.00	59.5	1.635	0.881

Table 4.9 PI controller coefficients from the Ho's formulae with $L/\tau = 0.4$

Specified				Resultant			
A_m	ϕ_m	K	T_i	A_m^*	ϕ_m^*	ITAE	IAE
3	45	1.23	0.68	2.86	46.2	0.993	0.640
5	45	0.73	0.68	4.81	58.2	1.992	1.019
3	60	1.31	1.00	3.00	60.1	1.360	0.764
5	60	0.76	0.83	4.96	65.0	1.097	2.191

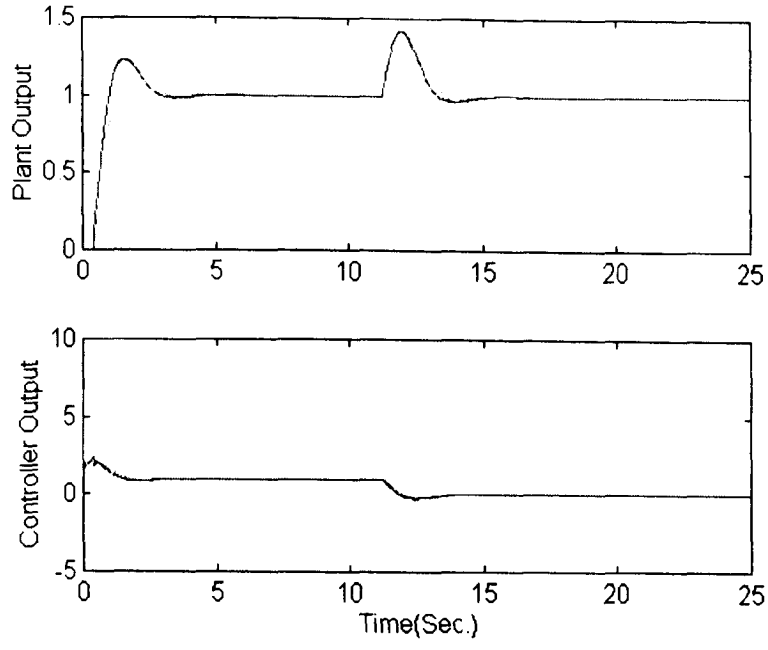

Figure 4.13 Comparison between PID and PI with $A_m = 3$, $\phi_m = 45$ and $L/\tau = 0.4$

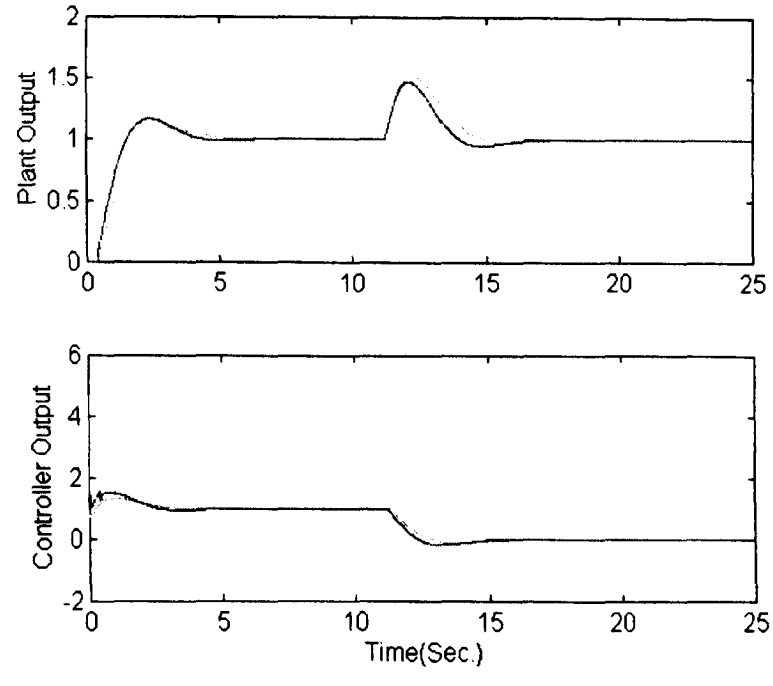

Figure 4.14 Comparison between PID and PI with $A_m = 5$, $\phi_m = 45$ and $L/\tau = 0.4$

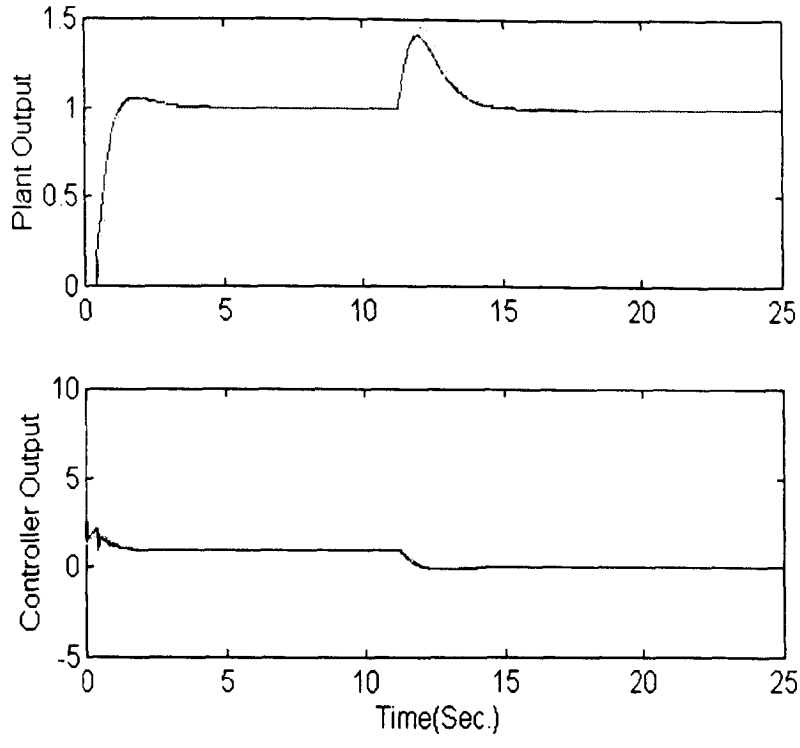

Figure 4.15 Comparison between PID and PI with $A_m = 3$, $\phi_m = 60$ and $L/\tau = 0.4$

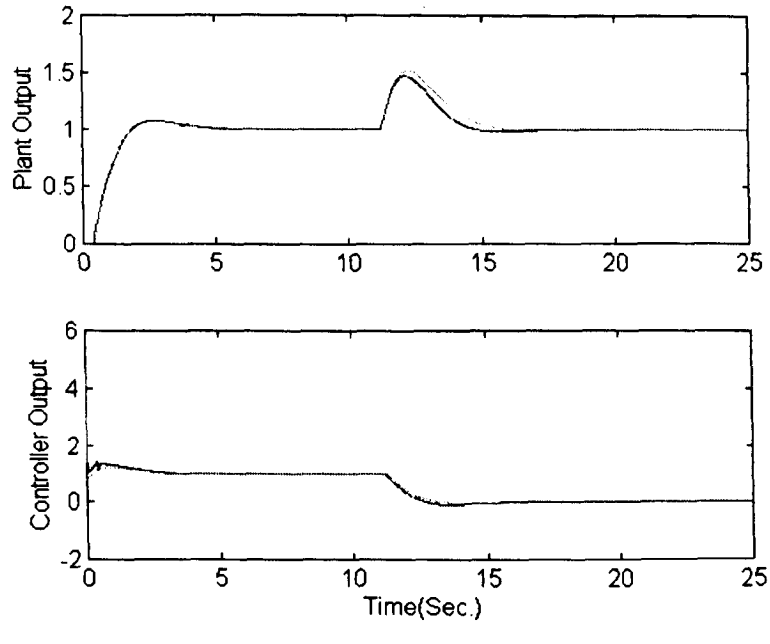

Figure 4.16 Comparison between PID and PI with $A_m = 5$, $\phi_m = 60$ and $L/\tau = 0.4$

Case 3: $G_p(s) = e^{-s}/(1+s)$

Table 4.10 PID controller coefficients from the hybridised GA with $L/\tau = 1$

Specified					Resultant			
A_m	ϕ_m	K	T_i	T_d	A_m^*	ϕ_m^*	ITAE	IAE
3	45	0.75	1.11	0.22	3.01	57.6	4.441	1.561
5	45	0.45	0.81	0.29	4.98	57.6	7.491	2.139
3	60	0.70	1.11	0.20	3.10	60.3	4.781	1.650
5	60	0.44	0.84	0.30	5.00	60.0	7.633	2.182

Table 4.11 PI controller coefficients from the Ho's formulas with $L/\tau = 1$

Specified				Resultant			
A_m	ϕ_m	K	T_i	A_m^*	ϕ_m^*	ITAE	IAE
3	45	0.49	0.84	2.87	53.9	6.685	2.034
5	45	0.29	0.84	4.85	67.5	10.97	2.894
3	60	0.52	1.00	3.02	60.2	6.042	1.970
5	60	0.30	0.92	4.97	70.0	11.95	3.019

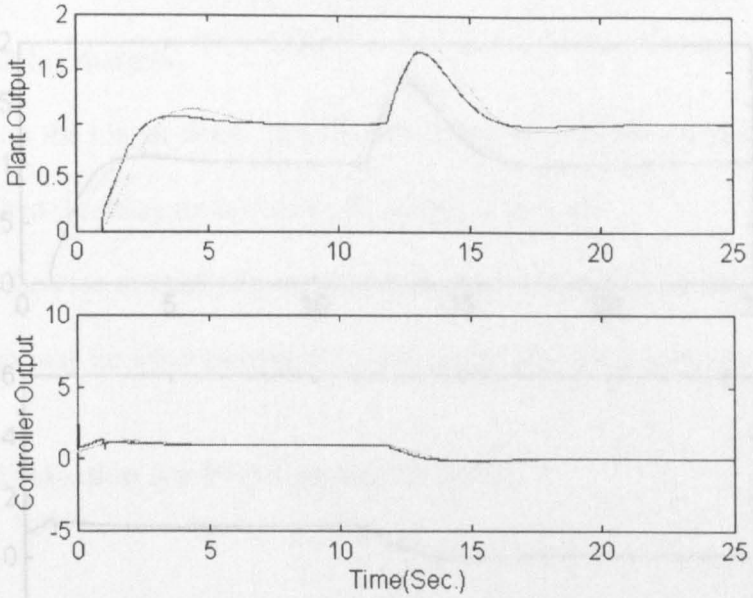

Figure 4.17 Comparison between PID and PI with $A_m = 3$, $\phi_m = 45$ and $L/\tau = 1$

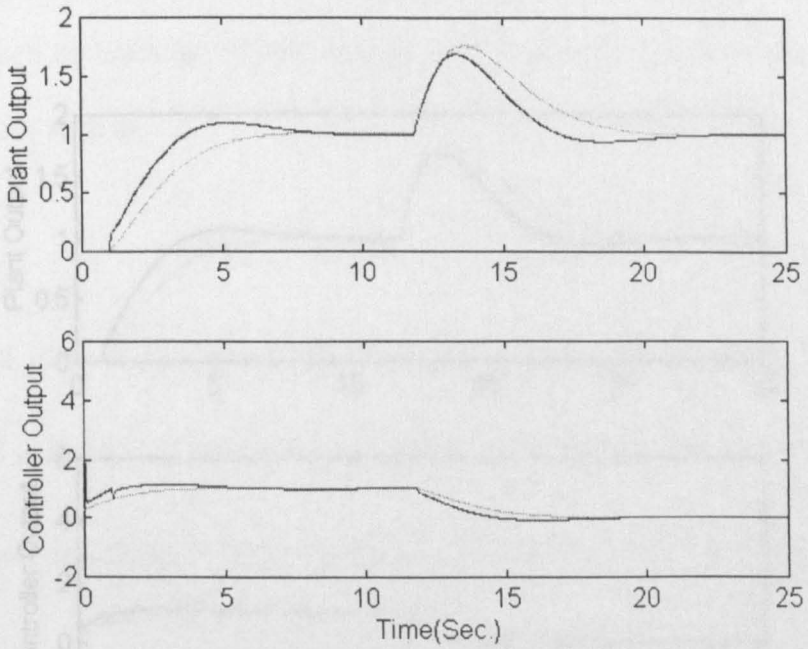

Figure 4.18 Comparison between PID and PI with $A_m = 5$, $\phi_m = 45$ and $L/\tau = 1$

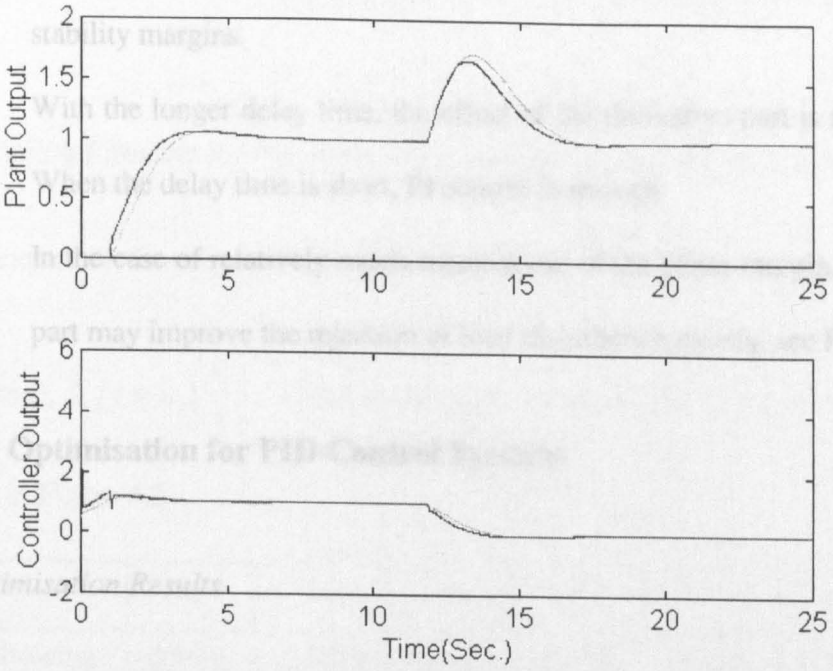

Figure 4.19 Comparison between PID and PI with $A_m = 3$, $\phi_m = 60$ and $L/\tau = 1$

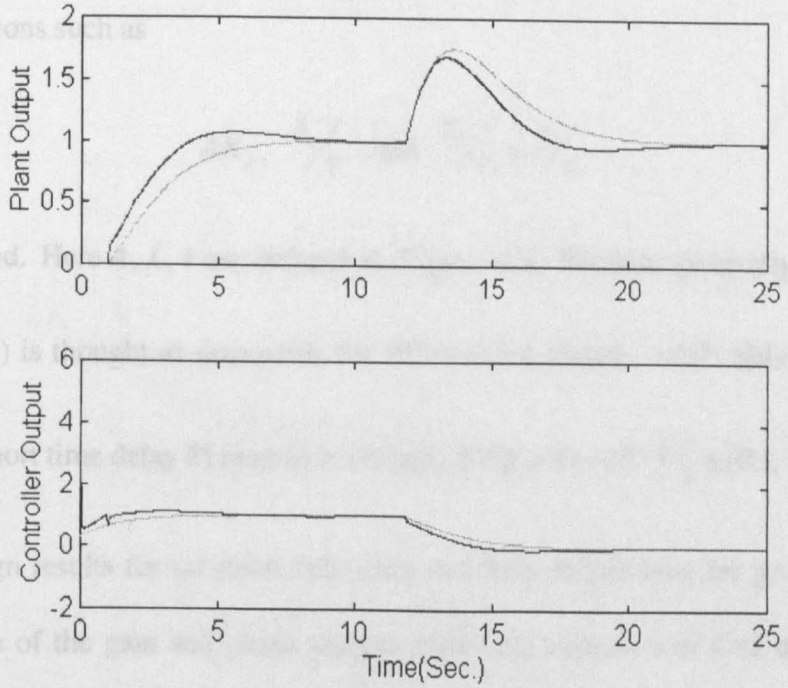

Figure 4.20 Comparison between PID and PI with $A_m = 5$, $\phi_m = 60$ and $L/\tau = 1$

From the these simulation results, it can be observed:

1. The derivative part can reduce the load disturbance under the limits of stability margins.
2. With the longer delay time, the effect of the derivative part is more apparent. When the delay time is short, PI control is enough.
3. In the case of relatively much requirement of the phase margin, the derivative part may improve the rejection of load disturbance greatly, see Figure 4.11.

4.4 Batch Optimisation for PID Control Systems

4.4.1 Optimisation Results

Since in Section 4.3, PID controllers have better rejection of load disturbance than PI controllers do when the gain and phase margins are employed as design specifications. In this section, batch optimisation of PID controllers is carried out. Just as other optimisation methods, relations such as

$$AK_p,\ {}^{T_i}\!/_{\tau} \text{ and } {}^{T_d}\!/_{\tau} \propto {}^{L}\!/_{\tau}$$

are investigated. Here A, L, τ are defined in Figure 4.2. Because generally too much time delay (${}^{L}\!/_{\tau} \geq 1$) is thought as unsuitable for PID control (Smith, 1957; Seborg, 1989) and if there is very short time delay PI control is enough, the plants with ${}^{L}\!/_{\tau} \in [0.1, 1]$ are designed.

Different design results for set-point following and load disturbance are given. To highlight the importance of the gain and phase margin limits, the rejection of load disturbances with and without limits are designed to make a comparison. The design requirement for gain margin is 3, and phase margin is 45 degree. The design results are given in Figure 4.21 to

4.23. Tuning rules given by approximation of points in Figures 4.21 to 4.23 are given in Table 4.12.

Table 4.12 Tuning formulae for PID control obtained by the step response method. The table gives coefficients of functions of the form $AK_p = a_1\left(\frac{L}{\tau}\right)^{b_1}$, $T_i/\tau = a_1\left(\frac{L}{\tau} - b_2\right) + c_2$ (set point following), $T_i/\tau = a_2\left(\frac{L}{\tau}\right)^{-b_2}$ (load disturbance rejection) and $T_d/\tau = a_3\left(\frac{L}{\tau}\right)^{b_3}$ for the model given in Figure 4.2.

	a_1	b_2	A_2	B_2	c_2	a_3	b_3
Set point following	0.9619	-0.8528	0.3463	0.4176	1.155	0.2888	0.8745
Load disturbance with stability margin	0.7189	-0.8536	1.164	0.4842	-	0.1979	1.252
Load disturbance	1.355	-0.9503	1.229	0.7383	-	0.3710	0.9478

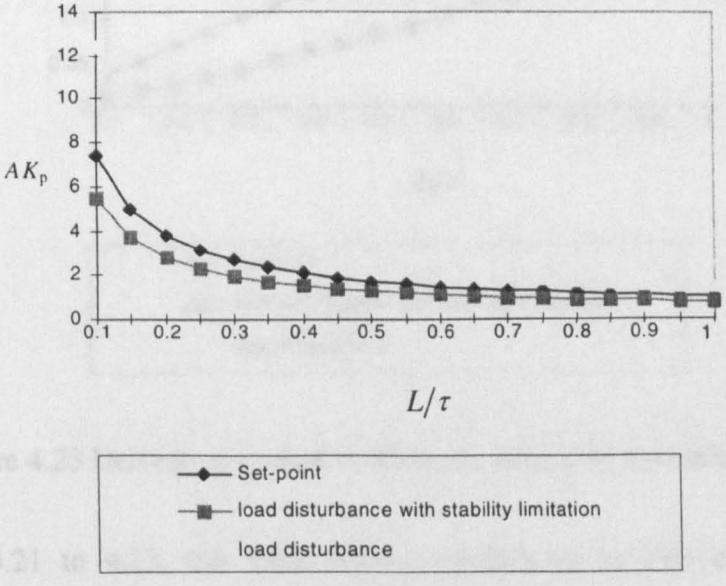

Figure 4.21 Normalised Proportional control coefficients related to normalised delay

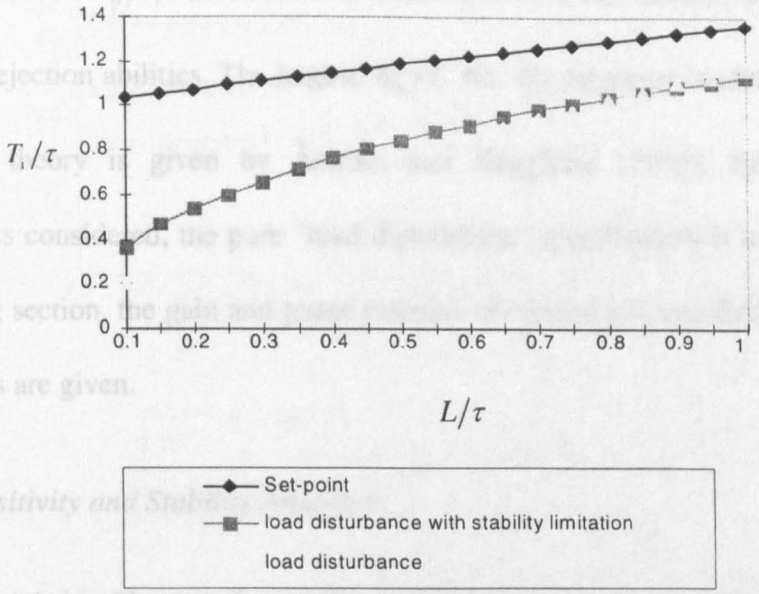

Figure 4.22 Integration control coefficients related to normalised delay

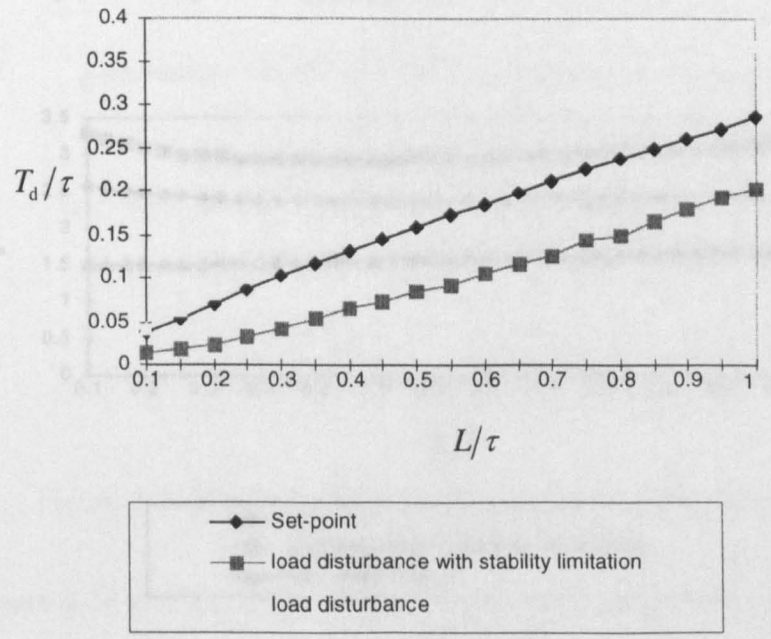

Figure 4.23 Derivative control coefficients related to normalised delay

From Figure 4.21 to 4.23, the three control coefficients in PID control for different specifications are presented. It could be found that for the specification "load disturbance" the control system has the largest K_p/T_i, "load disturbance with stability limitation" the second and "set-point" the last. It can be also found that with the increase of the normalised

delay, the values of K_p/T_i are close to one another. These differences hint that different load disturbance rejection abilities. The largest K_p/T_i has the strongest load disturbance rejection ability. The theory is given by Åström and Hägglund (1995). However, if stability requirement is considered, the pure "load disturbance" specification is not a good choice. In the following section, the gain and phase margins of control systems designed from different specifications are given.

4.4.2 Sensitivity and Stability Analyses

As investigated in Chapter 3, stability specifications can be indicated by maximum sensitivity, or gain and phase margins. Therefore, design results from these indices against $L/\tau \in [0.1, 1]$ are given in Figure 4.24 to 4.26.

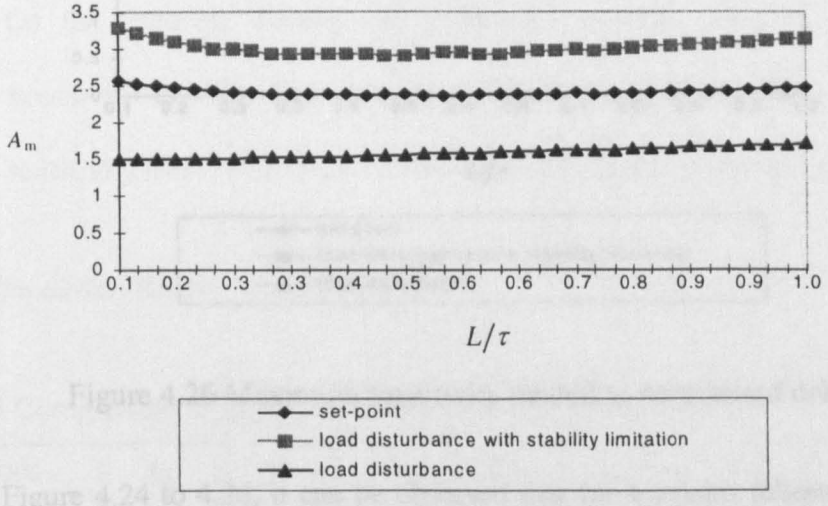

Figure 4.24 Gain margins related to normalised delay

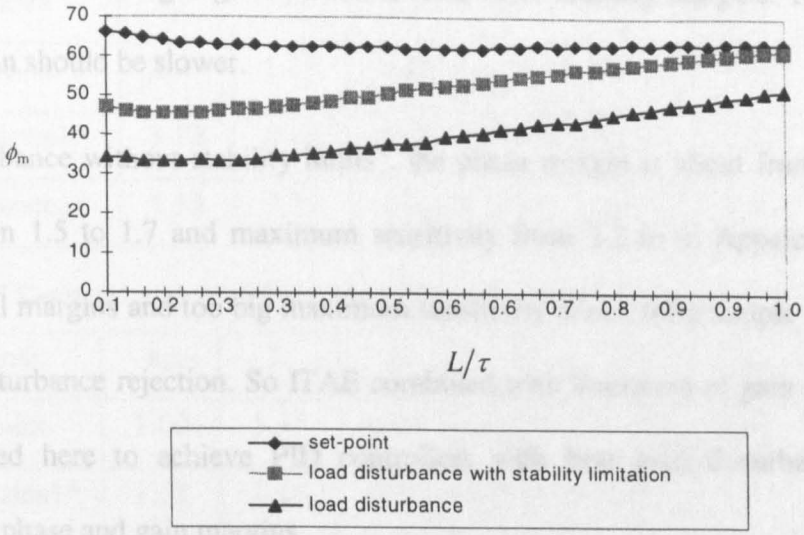

Figure 4.25 Phase margins related to normalised delay

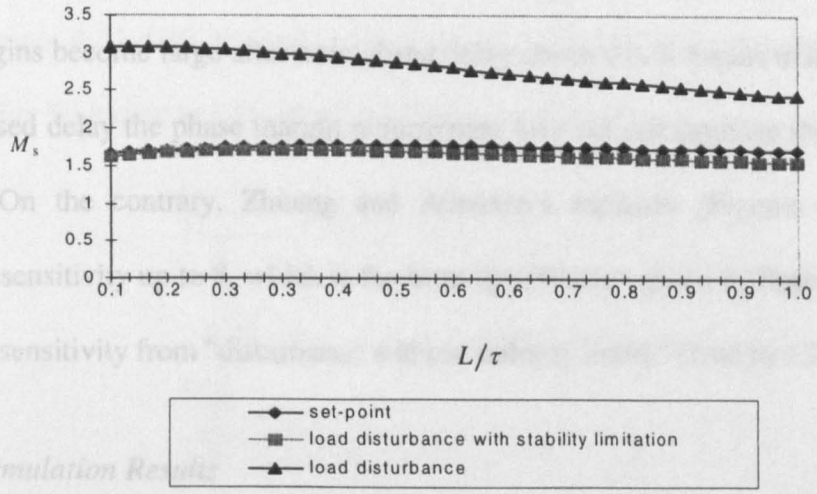

Figure 4.26 Maximum sensitivity related to normalised delay

From the Figure 4.24 to 4.26, it can be observed that for set-point following case, the gain margin is 2.4 to 2.6, the phase margin from 62 to 66 degree and maximum sensitivity from 1.66 to 1.78. This is similar to that of Zhuang and Atherton's methods, which are given in Figure 4.3 to 4.5: the gain margin is 2.3 to 2.5, the phase margin from 60 to 64 degree and maximum sensitivity from 1.8 to 1.9. Basically both of them meet general rules: the gain margin 2.5 to 4; the phase margin 40 to 65 degree, and the maximum sensitivity 1.2 to 2

(Levine 1996), but designs given in thesis have more stability margins. The response in the time domain should be slower.

For "disturbance without stability limits", the phase margin is about from 35 to 55 degree, gain margin 1.5 to 1.7 and maximum sensitivity from 2.2 to 3. Apparently, there are too much small margins and too big maximum sensitivity arisen from simple ITAE optimisation of load disturbance rejection. So ITAE combined with limitation of gain and phase margins is employed here to achieve PID controllers with best load disturbance rejection and acceptable phase and gain margins.

It is clear under our design strategy that the gain and phase margins basically satisfy the initial design requirements (Gain margin 3, phase margin 45 degree). For this type of design, phase margins become large after normalised delay above 0.6. It means that with the increase of normalised delay the phase margin requirement may not compromise the load disturbance rejection. On the contrary, Zhuang and Atherton's methods (Figures 4.3 to 4.5) have maximum sensitivity up to 8, which is far from specification given in Page 99 (1.2 to 2). But maximum sensitivity from "disturbance without stability limits" (1.66 to 1.78) meets it.

4.4.3 Simulation Results

Table 4.13 Comparisons of different specification with $L = 0.1$, $\tau = 1$

Methods	k_p	T_i	T_d	J_1**	J_2**	A_m	ϕ_m	M_s
Set-point	7.42	1.07	0.036	0.194	0.009	2.57	66.2	1.67
Disturbance1*	12.41	0.21	0.041	0.005	0.038	1.50	34.4	3.07
Disturbance2*	5.46	0.36	0.016	0.029	0.072	2.99	44.78	1.65

Table 4.14 Comparisons of different specification with $L = 0.4$, $\tau = 1$

Methods	k_p	T_I	T_d	J_1**	J_2**	A_m	P_m	M_s
Set-point	2.07	1.15	0.13	1.052	0.495	2.37	62.5	1.78
Disturbance1*	3.18	0.65	0.16	0.412	0.216	1.53	35.8	2.94
Disturbance2*	1.53	0.77	0.064	0.749	0.053	2.94	45.2	1.70

Table 4.15 Comparisons of different specification with $L = 1$, $\tau = 1$

Methods	k_p	T_I	T_d	J_1**	J_2**	A_m	P_m	M_s
Set-point	1.00	1.37	0.29	3.779	0.340	2.47	63.3	1.71
Disturbance1*	1.39	1.16	0.37	2.028	1.487	1.71	51.4	2.44
Disturbance2*	0.75	1.11	0.20	4.441	1.145	3.01	57.6	1.56

*Disturbance1: load disturbance design without phase and gain margins limitation. Disturbance 2 limited by $A_m > 3$ and $P_m = 45$.

** J_1 is ITAE for load disturbance and J_2 is ITAE for set-point following.

In the Figures 4.27-4.29, the red is for "load disturbance", the green for "load disturbance with stability limitation and the black for "set-point following".

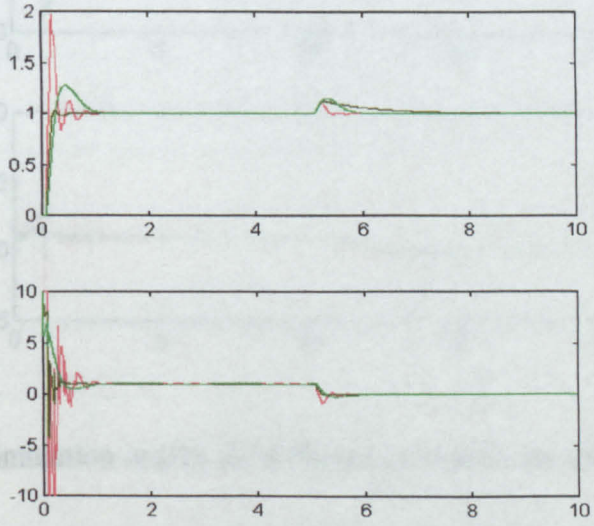

Figure 4.27 Simulation results for different specifications with $L = 0.1$, $\tau = 1$

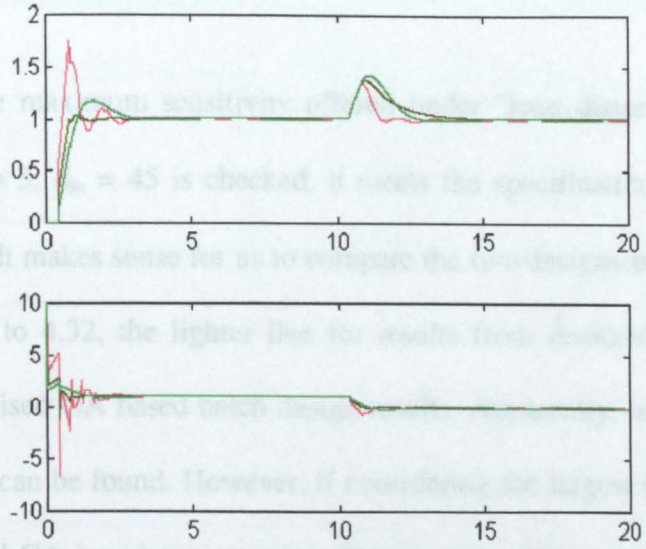

Figure 4.28 Simulation results for different specifications with $L = 0.4$, $\tau = 1$

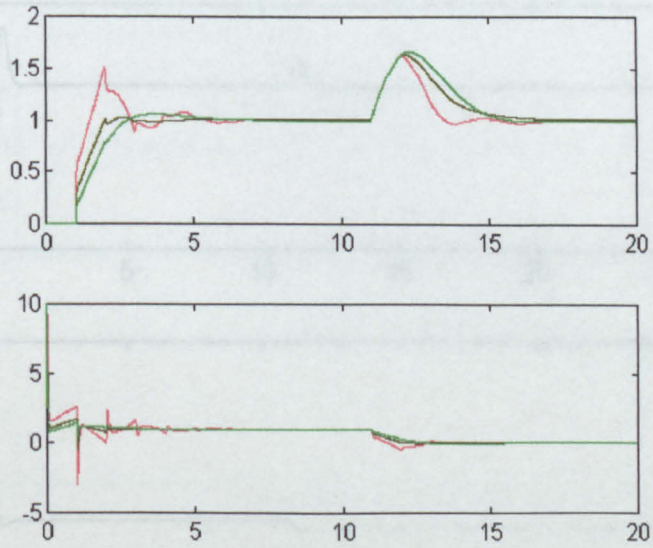

Figure 4.29 Simulation results for different specifications with $L = 1$, $\tau = 1$

From the simulation results, it can be observed that with the increase of the delay, Designs from "set-point " have the close or better load disturbance rejection than those from "load disturbance with stability margin". However, it should be noticed that this is at the cost of phase margin (Table 4.13 to 4.15).

4.4.4 Comparison

In Figure 4.26, if the maximum sensitivity offered under "load disturbance with stability limitation" with $A_m = 3$, $\phi_m = 45$ is checked, it meets the specification $M_s = 2.0$ given by Åström (Figure 4.8). It makes sense for us to compare the two designs in the time domain as well. In Figure 4.30 to 4.32, the lighter line for results from Åström's formulae, and the darker line for hybridised GA based batch design results. Apparently, improvement for load disturbance rejection can be found. However, if considering the largest $M_s = 1.7$ for systems evolved by hybridised GA based environment, there is still some room to improve the load disturbance rejection under the specification $M_s = 2.0$.

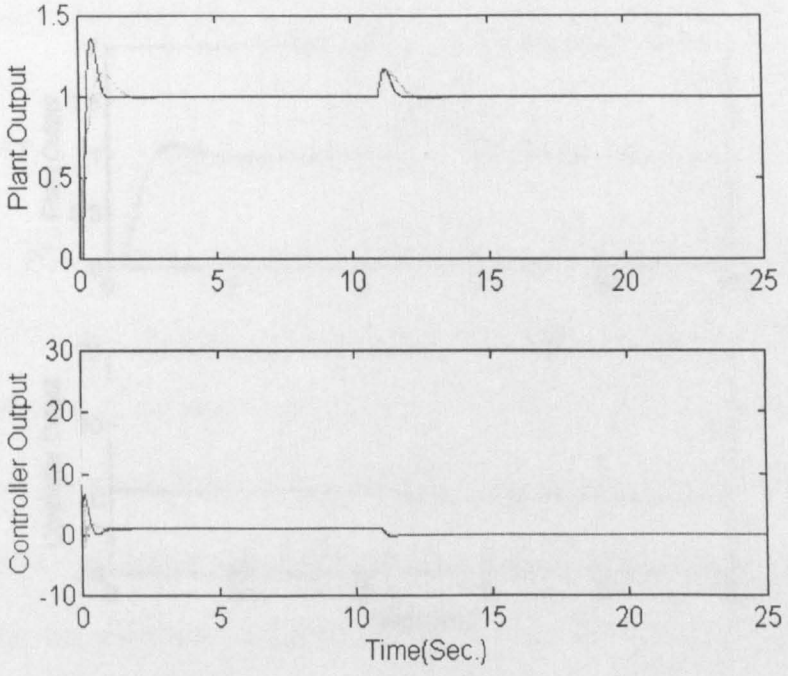

Figure 4.30 Results of designs with EA and Åström 's formulae with $T_d = 0.1$ and $\tau = 1$

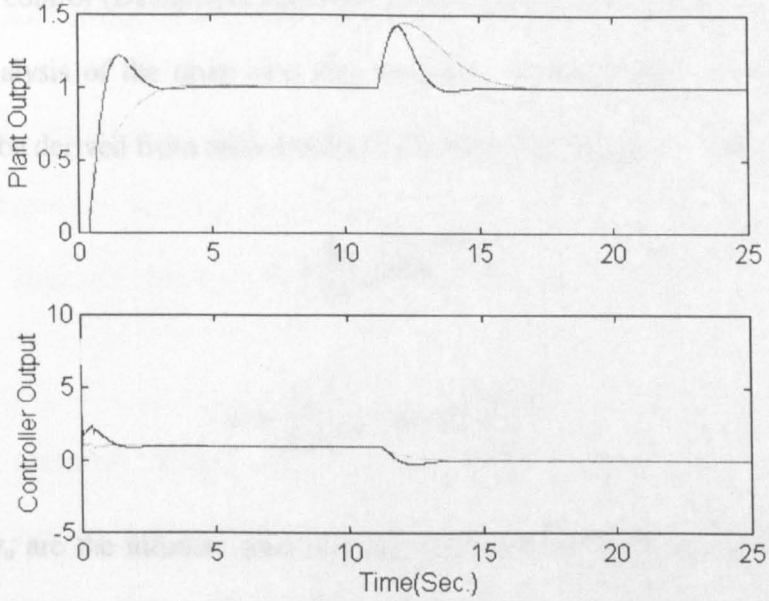

Figure 4.31 Results of designs with EA and Åström 's formulae with $T_d = 0.4$ and $\tau = 1$

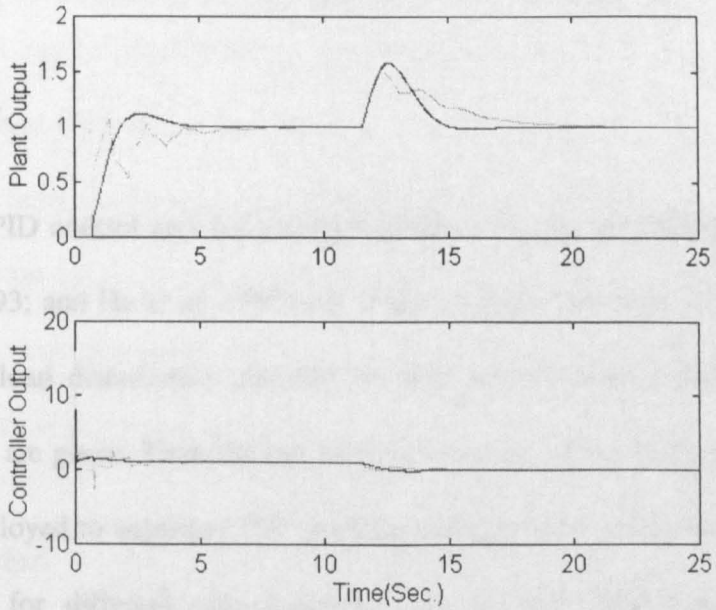

Figure 4.32 Results of designs with EA and Åström's formulae with $T_d = 0.7$ and $\tau = 1$

4.4.5 Simple relay PID auto-tuning

The tuning rules given in Table 4.12 are well suited for PID auto-tuning and adaptive control. There are many identification techniques that can be used to obtain the FOPDT

model for PID control (Deshpande and Ash, 1988; Åström *et al.* 1993). A simple method is through the analysis of the open-loop step response. Alternatively, the FOPDT model of Figure 4.2 can be derived from relay feedback (Åström and Hägglund, 1984, 1988):

$$\tau = \frac{t_u}{2\pi}\sqrt{k_u^2 k_p^2 - 1} \tag{4.28}$$

$$L = \frac{t_u}{2\pi}\left(\pi - \text{ar tan}\frac{2\pi\tau}{t_u}\right) \tag{4.29}$$

where k_u and t_u are the ultimate gain and ultimate period, which can be obtained from the relay experiment. k_p is the corresponding static gain. Since there are advantages of relay feedback identification (Majhi and Atherton, 1999; Atherton 2000), (4.28) and (4.29) are very useful relations between frequency model and step response model.

4.5 Summary

In this chapter, PID control and the tuning methods (Åström and Hägglund, 1995; Zhuang and Atherton 1993; and Ho *et al.* 1995) are analysed. Based on these contributions, detailed comparisons of load disturbance rejection by PID and PI control designs under specific stability margins are given. Then the hybridised GA based design environment developed in the thesis is employed to automate PID controller designs in batch mode. The optimisations are carried out for different specifications, e.g., set-point following, load disturbance rejection and load disturbance with stability limitation. Finally, comparisons with designs from Åström's formulae are given. It is found that PID controllers achieved have better load disturbance rejection than counterpart from Åström's formulae, while the stability margins can still be maintained or improved.

Chapter 5 Automating Robust Loop Shaping and QFT Design

Although a PID controller shapes the overall frequency response, this simple control strategy offers limited capability in robust loop shaping for uncertain plants due to the limitations of its 'three term' structure. However, QFT provides a more sophisticated tool for robust loop shaping.

In this chapter, therefore, further robust issues are addressed by extending the PID structure to a free form transfer function, which is achieved by employing the hybridised GA based design environment to carry out automatic loop shaping for more complicated linear controllers in QFT. Since EA is considered as one of powerful techniques to deal with complicated search space, to solve the complicated loop shaping problem, the search power of hybridised GA can be highlighted. In this chapter, the QFT problem is first presented in Section 5.1. General QFT design procedures are outlined in Section 5.2. The hybridised GA based automated design technique is detailed in Section 5.3. This is illustrated by benchmark examples in Section 5.4, in which manual loop shaping and linear programming based loop shaping methods are compared with automated loop shaping developed in this thesis. Finally, a summary is given in Section 5.5.

5.1 Introduction

Since QFT was developed by Horowitz (1973, 1992), it has found many successful applications in control engineering practice, such as control of wastewater treatment plant (Ostolaza and GarciaSanz, 1998), robust flight control systems design (Keating *et. al.*, 1997; and Wu *et al.*, 1998) and active noise control in duct (Chai *et al.*, 1997). The underlying principle of QFT is to transform plant uncertainties and closed-loop design specifications

into robust stability and performance bounds, so as to design a robust controller using simple gain-phase loop shaping techniques with the nominal system.

From all sorts of applications, the most important feature of QFT is its ability to tackle design problems concerning complicated uncertain plants. At present, loop shaping is performed manually in a computer-aided control system design (CACSD) environment. The main advantages of this method are that the design procedure is transparent and the designer can consider factors that might be difficult to represent by analytical expressions or quantitatively. Since most CACSD packages (such as MATLAB toolboxes) are simulation packages, however, the QFT design procedure often falls in a trial-and-error process (Chait, 1997). Whether the design is successful or not is thus dependent upon the experience and/or intelligence of the human designer. Moreover, for uncertain, unstable or non-minimum phase plants, it is difficult to design a controller that may satisfy all specifications manually, sometimes even in the case where the plant is merely a stable one. This is also true for systems with a large number of resonance, pure delays, etc., where a high-order and/or complex controller is necessary. These contribute to the complexity and difficulties in manual loop shaping, where mutually interactive factors need to be taken into account.

To solve this design problem and unleash the power of QFT, optimisation and 'automatic design' techniques have recently been investigated and developed. These techniques include:

1. The use of Bode integrals in an iterative approach to loop shaping by Horowitz and Gera (1980) and Ballance and Gawthrop (1991).
2. Thompson and Nwokah's analytical approach (1994) to loop shaping, if the templates may be approximated by boxes or an initial QFT controller already exists.
3. A linear programming approach to automatic loop shaping by Bryant and

Halikias (1995).

4. An automatic technique by Chait (1997), which over-comes the non-convexity of the bounds on the open-loop transmission, whilst the design is based on the closed-loop bounds.

The major disadvantage of these approaches is, however, the inability in solving a complicated nonlinear optimisation problem. The QFT design problem is multi-modal and multi-dimensional. Global solutions can hardly be obtained using analytical and/or convex or linear programming techniques. Further, this type of conventional methods often imposes unrealistic or unpractical assumptions and often leads to very conservative designs. For example, the denominator of the closed-loop transfer function of a QFT control system must be specified in advance if the approach of Chait (1994) is to be used.

In view of the multi-dimensional non-convexity of the loop shaping problem in QFT designs, a globally optimal QFT design automation technique using the evolutionary computation paradigm was first proposed in (Chen, *et al.*, 1998). The evolutionary algorithm computerises the trial-and-error process 'intelligently'. It can globally optimise for multiple objectives efficiently in a multivariate multi-modal space. In our research, this method is furthered by using the hybridised GA-based design suite in presented in Chapter 3. Thus, QFT controllers are easily designed from scratch for uncertain industrial plants such that the cost of feedback is minimised and all robust stability and performance specifications are satisfied. Improvement of existing designs is tried as well. In particular, the automated QFT design technique presented here consists of two steps. The first is closed-loop shaping, where a robust controller tackling uncertain plants is evolved such that the cost of feedback is minimised and all robust stability and performance bounds are satisfied. Then a prefilter is automatically designed to meet open-loop performance specifications if there is the requirement for tracking.

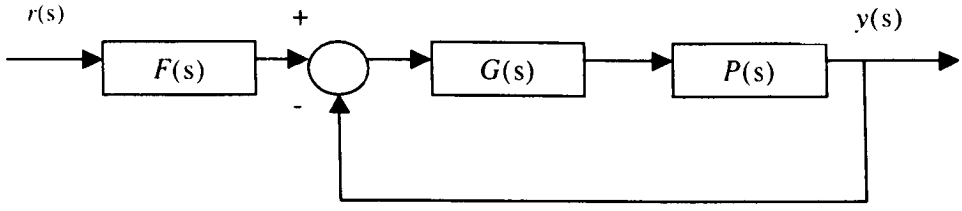

Figure 5.1 Control system configuration in QFT

5.2 Robust Loop Shaping by QFT

5.2.1 QFT Review

The two-degree-freedom feedback system configuration of QFT is given in Figure 5.1 where $G(s)$ and $F(s)$ are referred to as the controller and the prefilter respectively. $P(s)$ denotes the uncertain plant, which belongs to a given plant family $\mathcal{P}$. It can contain structured, unstructured or mixed uncertainties.

One view of the QFT approach is that if there are no uncertainties and noise, the feedback is unnecessary and we can achieve the prescribed performance specification by the prefilter $F(s)$, which can be designed via open-loop shaping. The main role of the controller, $G(s)$, is therefore to reduce uncertainties and disturbances by using feedback. The QFT design is thereby divided into two steps. The first step is to design the controller, $G(s)$, such that uncertainties and noise on the closed-loop system are reduced to an acceptable level which is determined by the closed-loop robust stability and performance specifications. The prefilter is then designed to achieve the desired frequency responses.

In general three kinds of specification are considered in QFT:

1. Robust Stability Margin

$$\left|\frac{L(j\omega)}{1+L(j\omega)}\right| \leq \gamma \ \forall \ P \in \mathcal{P}, \quad \omega \in [0,\infty) \tag{5.1}$$

2. Tracking Performance

$$|a(\omega)| \leq \left|F(j\omega)\frac{L(j\omega)}{1+L(j\omega)}\right| \leq |b(\omega)| \ \forall \ P \in \mathcal{P}, \quad \omega \in [0,\omega_l] \tag{5.2}$$

3. Disturbance Attenuation Performance

$$a(\omega) = \frac{0.6854(j\omega + 30)}{(j\omega)^2 + 4(j\omega) + 19.752} \tag{5.7}$$

$$b(\omega) = \frac{120}{(j\omega)^3 + (j\omega)^2 + 828(j\omega) + 120} \tag{5.8}$$

and

$$\omega_l = 15\text{rad/s} \tag{5.9}$$

Based on such a problem, a common QFT Design procedure is outlined below:

1. Generating templates. A given uncertain plant $P(s) \in \mathcal{P}$ select a series of frequency points according to the plant characteristics and specifications. Calculate $P(j\omega)$, the plant templates, at all required frequency points. The 14 points in Figure 5.2 can be thought as characteristic points, because the corresponding points in the complex plane just enclose an area. So the 14 plants represented by the points could be thought as suitable templates. In Figure 5.3, frequency characteristics of plants at specific frequency are shown, including amplitude and phase. In order to check characteristics on different frequencies, more templates are calculated in Figure 5.4,

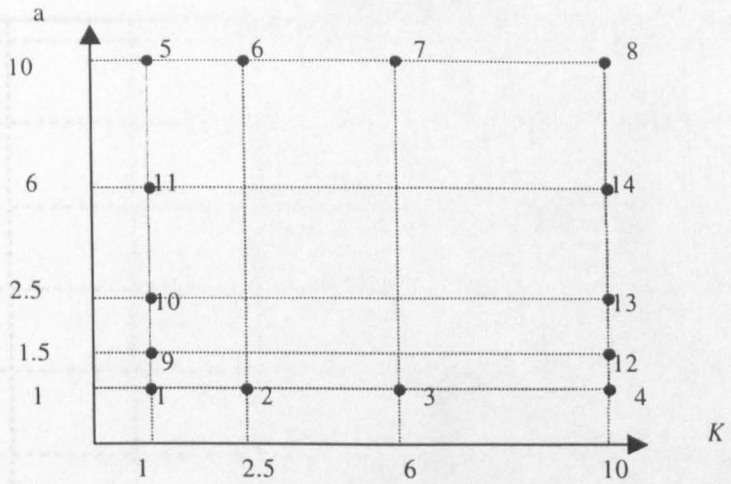

Figure 5.2 Varied area of parameters and template points

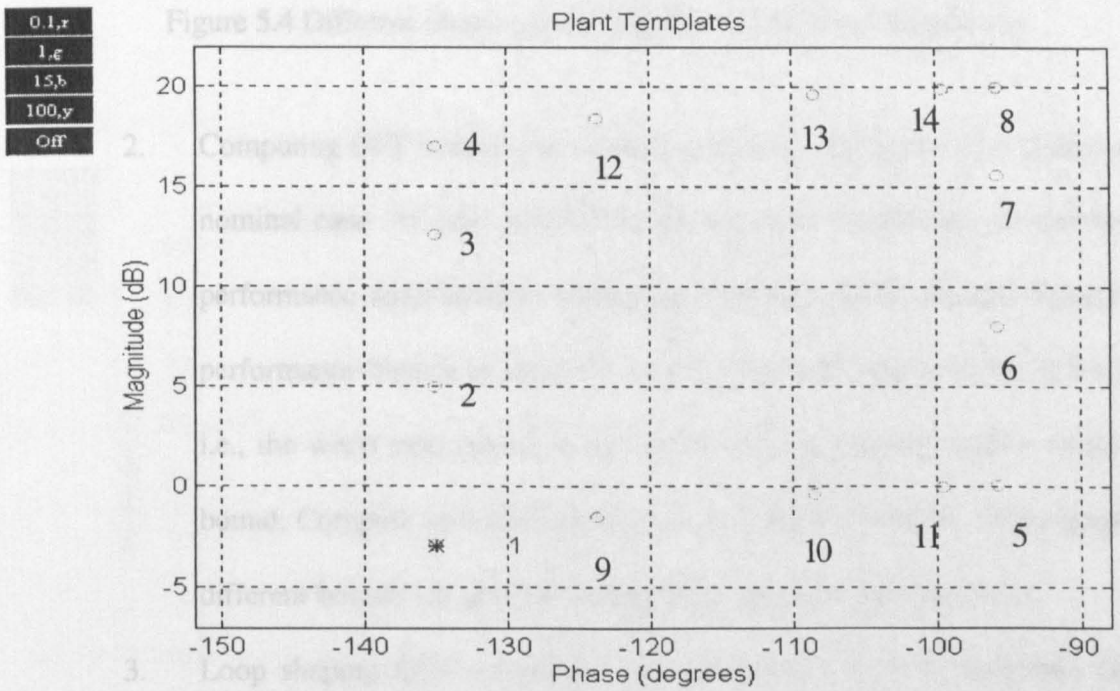

Figure 5.3 Corresponding points of template plants in the complex plane with $\omega = 1$

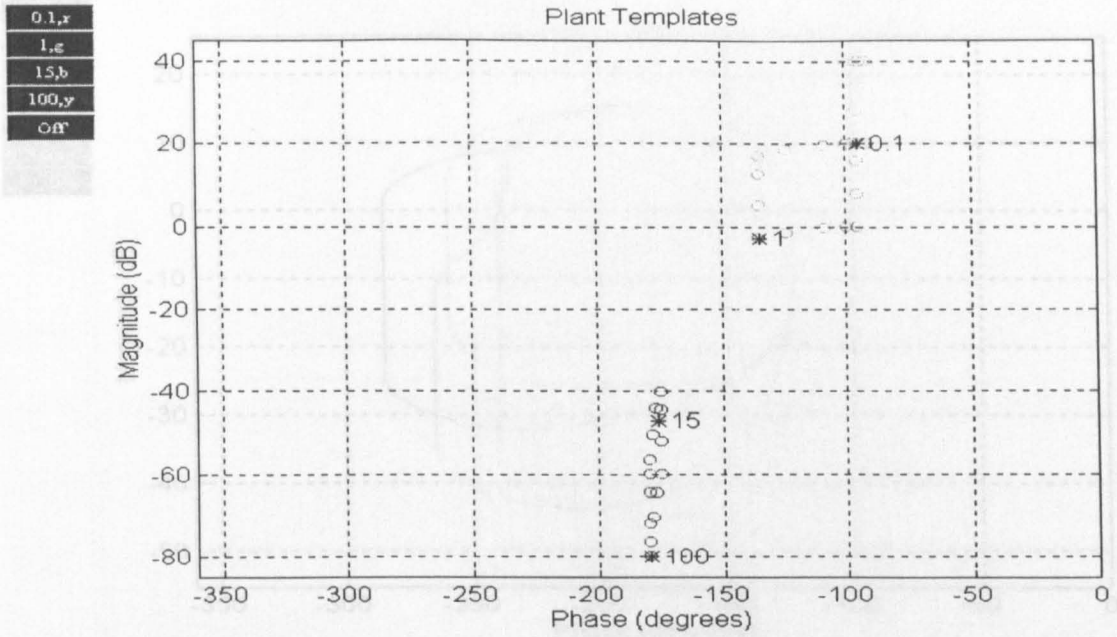

Figure 5.4 Different shapes of the templates for different frequencies

2. Computing QFT bounds. An arbitrary member in the plant set is chosen as the nominal case. At each selected frequency point, combining the stability and performance specifications with plant templates yields stability margins and performance bounds in term of nominal case. Intersection of all such bounds, i.e., the worst case bound, at the same frequency point yields a single QFT bound. Compute such QFT bounds for all frequency points. Some graphs for different bounds are given in Figure 5.5, Figure 5.6 and Figure 5.7.

3. Loop shaping QFT controllers. The design of the QFT controller, $G(s)$, is accomplished on the Nichols Chart. The phase gain loop shaping technique is employed to design controllers, until the QFT bounds at all frequencies are satisfied, while the closed-loop nominal system is kept stable.

4. Designing prefilters. The final step in QFT is to design the prefilter, $F(s)$, such that the performance specifications are satisfied.

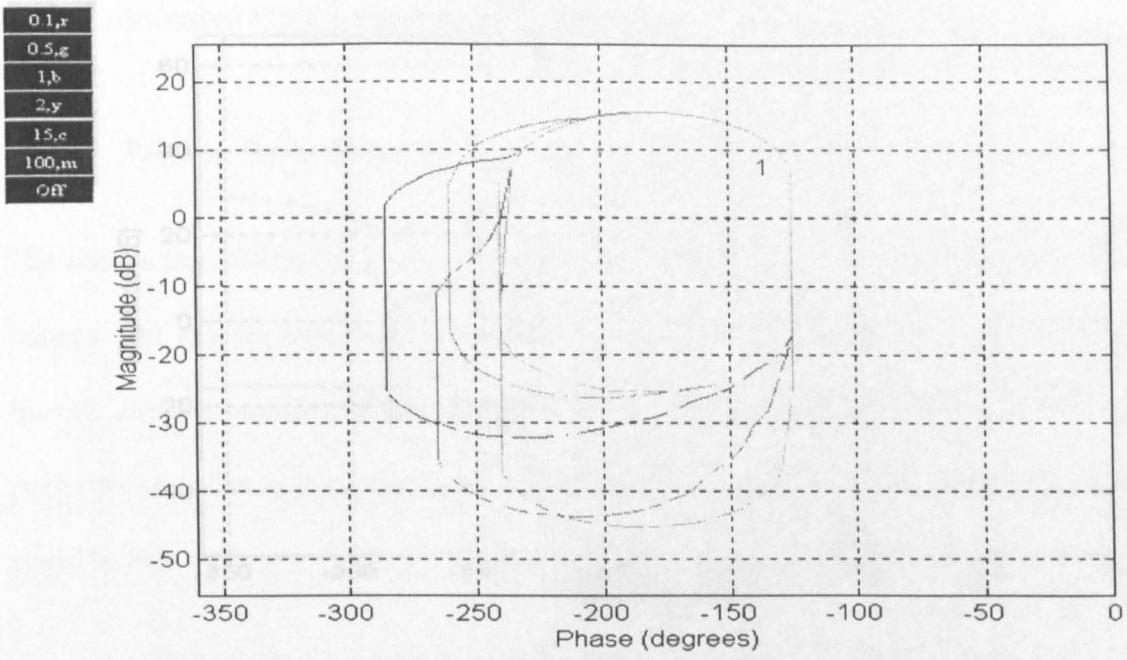

Figure 5.5 Stability margin bounds

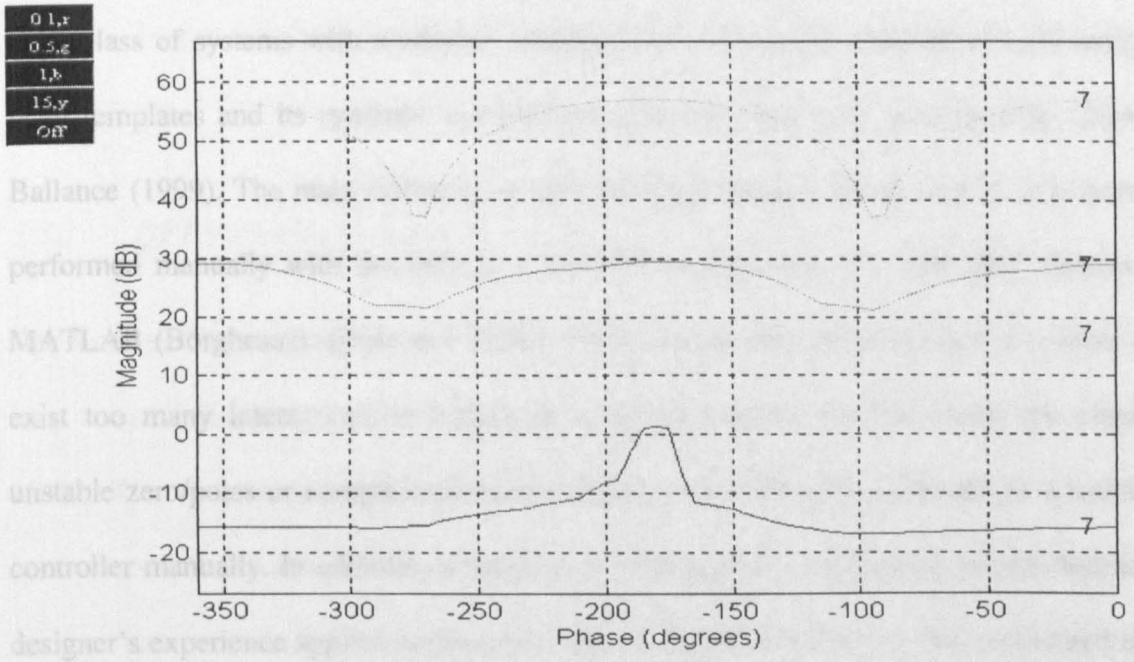

Figure 5.6 Tracking specification bounds

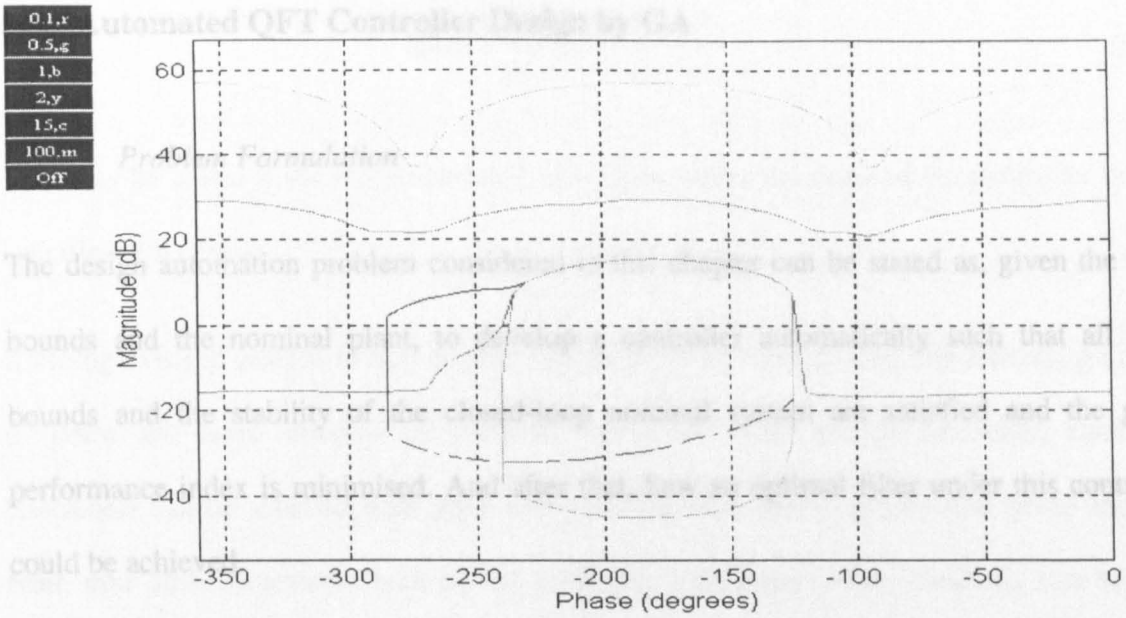

Figure 5.7 Intersections of bounds

In general the first two steps can be carried out by numerical evaluations on computer. For a large class of systems with nonlinear uncertainties, a systematic method for generating the plant templates and its symbolic computation procedure has been developed by Chen and Ballance (1999). The main difficulty in QFT design procedure lies in Step 3. It is normally performed manually with the help of a CACSD environment, e.g., the QFT Toolbox for MATLAB (Borghesani, Chait and Yaniv, 1995). As pointed out in Section 5.1, there often exist too many interactions to handle in a design process. Further, when the plant has unstable zero/poles or complicated characteristics, it may be difficult to design a stabilising controller manually. In addition, whether or not the design is successful mainly depends on designer's experience applied to the trials. Thus, the QFT controller is first considered in this chapter. After the controllers, which ensure all QFT bounds are satisfied, have been worked out in step 3, a systematic way to achieve the optimal filter is presented in this chapter as well.

5.3 Automated QFT Controller Design by GA

5.3.1 Problem Formulation

The design automation problem considered in this chapter can be stated as, given the QFT bounds and the nominal plant, to develop a controller automatically such that all QFT bounds and the stability of the closed-loop nominal system are satisfied and the given performance index is minimised. And after that, how an optimal filter under this controller could be achieved.

A good automatic design procedure, we believe, should be flexible and transparent to the designer. The designer should know how to control the optimisation process to achieve the specific requirements for a problem in hand by adjusting the parameters provided by the optimisation procedure, for example, the order the controller, whether or not an integral is included, etc.

5.3.2 QFT Variables to Optimise

The work of Horowitz has shown that the optimal QFT design is achieved when the open-loop transmission lies on the corresponding QFT bound at each frequency point (Horowitz, 1992). This is incorporated in the hybridised GA automated procedure. Any irrational controllers or those with unstable pole and non-minimum phase zero cancellation may not be allowed to survive in the evolution. Since QFT bounds are given in terms of the open-loop transmission, it is thus convenient to limit the minimum of that of the controller order to that of the plant plus that of the resultant open-loop transmission. The controller can be evolved by polynomials, as in:

$$G(s) = \frac{b_r s^r + \cdots + b_1 s + b_0}{a_m s^m + \cdots a_1 s + a_0} \tag{5.10}$$

This may be useful if the design starts from scratch, where the order of the controller can be specified if necessary

If tuning existing designs is only required, controllers could be evolved by refining positions of poles and zeros directly. In order to reduce the order of the controller, candidate controllers can be assessed with poles and zeros far from the imaginary axis being omitted. Note that other structures, such as the realisable (non-ideal) PID structures, can be also imposed in the search.

5.3.3 Stability of the Nominal Case

It is well known that a sufficient and necessary condition for the robust stability of closed-loop systems is that the nominal system is stable and the open-loop transmission under the prescribed plant set does not intersect the $-1+j0$ point in the complex plane. The latter is guaranteed in QFT by the robust stability margin condition.

In manual designs, stability is checked in the Nichols Chart graphically. For an automated design, the roots of the characteristic equation can be checked. A simple cost function to penalise unstable designs is:

$$J_{sta} = \begin{cases} 0 & \textit{if stable} \\ d_{sta} & \textit{if unstable} \end{cases} \tag{5.11}$$

where d_{sta} is the distance to the imaginary axis in the complex plane.

5.3.4 *Right Half Plane Pole/Zero Cancellation*

In order to ensure internal stability, it is desired that a minimum phase and stable controller be designed. This can guarantee the internal stability and no unstable pole and non-minimum phase zero cancellations. For an automated design, the necessary condition is utilised first to limit all coefficients of the transfer function $G(s)$ to be positive. If necessary, the poles and the zeros of the controllers can be calculated explicitly to avoid right half pole/zero cancellation by comparing them with all right-half plane poles/zeros (if any) of the nominal case. Alternatively, the Horowitz method for QFT design of unstable and non-minimum phase plants can be used, i.e., to translate QFT bounds for an unstable/non-minimum phase nominal plant to that for a stable and minimum phase plant (Horowitz, 1992). This avoids right half plane pole/zero cancellations since the new nominal plant is stable and is of minimum phase.

5.3.5 *QFT Bounds*

It is difficult to give analytical expressions of the QFT bounds (Thompson and Nwokah, 1994) since in general the QFT bounds are very complicated and are non-convex (Horowitz, 1992). In our research, the QFT bounds are generated first using the QFT Toolbox. Then, with the capability of an evolutionary algorithm, these numerical bounds can be used directly in an automated design. At each frequency point, the gain and phase of the open-loop transmission $L(j\omega_i)$ is calculated and then checked to see whether the QFT bound at this frequency is satisfied by interpolation. A simple bound index is given by

$$J_{bi} = \begin{cases} 0 & \textit{if QFT bound at } \omega_i \textit{ satisfied} \\ d_{bi} & \textit{otherwise} \end{cases} \tag{5.12}$$

where d_{hj} is the distance to the QFT bound at jth frequency point. A Universal High-frequency Bound (UHB) is widely used in QFT. To ensure the open-loop transmission does not intersect the UHB, a number of frequency points near or greater than the largest frequency are added. The gain and phase of the open-loop transmission is computed and the UHB is tested at those frequency points. This does not add much to the computational burden since no new QFT bounds need to be calculated.

5.3.6 *Performance Index of QFT Controller Design*

The optimum in QFT is taken to be any $L(j\omega)$ whose magnitude as a function of frequency decreases as fast as possible (Horowitz, 1992). The justification for this is to consider the effects of high-frequency sensor noise and the unmodelled high-frequency dynamics/harmonics, which may result, with unnecessarily large bandwidth, in actuator saturation and instability. It follows that the cost-function to be minimised is the high-frequency gain of the open-loop transmission L(s), which is termed the cost of feedback in QFT. Since the nominal is fixed, this is equivalent to the high frequency gain of the controller, given by

$$J_{hg} = b_r / a_m \tag{5.13}$$

This performance index is widely adopted in QFT optimisation (Horowitz, 1992; Thompson and Nwokah, 1994; Chait, 1997; and Bryant and Halikias, 1995) and is used as another cost in guiding the EA search. Since the stability and bounds are hard conditions to satisfy in the design, it is difficult to optimise the QFT controller for all three objectives. This also means that it is counter-productive if a multi-objective EA is applied, as no compromise may be made to the stability and bounds goals. Thus a single composite cost is formed for the EA search for the QFT design, as given by

$$J = \log J_{hg} + \sum_{i=1}^{h} \gamma_i J_{bi} + \gamma_0 J_{sta} \tag{5.14}$$

5.3.7 Prefilter Design

While robust controllers have emerged, prefilters can be evolved to satisfy the tracking requirements. In QFT, the tracking performance is represented as desired frequency response bounds. The objective of the design of the prefilter is to fit the frequency response of closed-loop systems within these desired frequency bounds. Starting the prefilter design, frequency response bounds of the closed-loop systems consisting of the designed QFT controller and the uncertain plant are calculated. According to our QFT bounds calculation for robust tracking, in required frequency band, the close-loop could be fitted in the bounds. Just like in the Figure 5.8, the two solid lines represent the extremes of close-loop responses without any filtering, the two dash lines represent the limitation given by specifications. Through a filter, the solid lines could be shifted in the limits of dash lines. In our design for fitness function, the middle line of the bounds is calculated, and the middle line of the closed-loop responses is calculated. The performance of the ideal filter we are to design is the difference between the two middle lines. Then hybridised GA environment is employed to design a filter to approximate the performance of the ideal filter under the specific frequency range. It is apparent that in the lower frequency range, the filter performance should be 0 dB, so the filter can be supposed be as

$$F(s) = \frac{b_n s^n + \cdots + b_1 s + 1}{a_m s^m + \cdots + a_1 s + 1} \tag{5.15}$$

Where $n < m$, since it is a lower pass filter. EA is used to find proper parameters of the filer. Cost function employed is

$$J = \sum_{i=1}^{m} \left| 20\log\left|F(f_i)\right| - 20\log\left|F_d(f_i)\right| \right| \tag{5.16}$$

Where F_d is the desired filter from calculation, f_i is logarithmically spaced frequency array. It means that lower frequencies area should be emphasised.

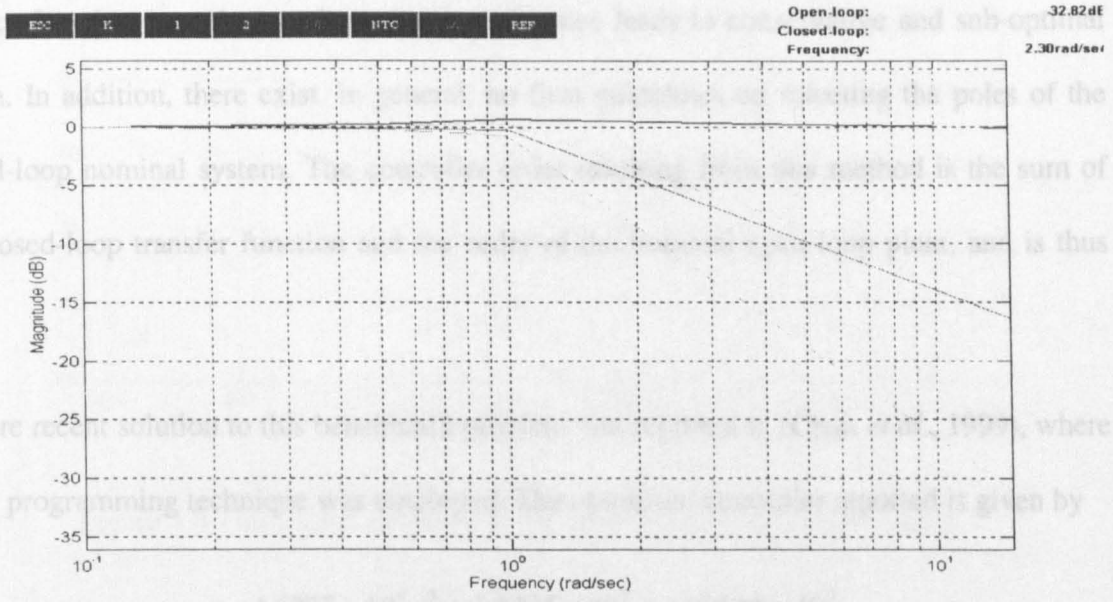

Figure 5.8 QFT filter design bounds and the extremes of response without a prefilter

5.4 Design Examples

5.4.1 Benchmark Example

For the benchmark problem introduced in Section 5.2.2, with the controller order prefixed to three, and the first benchmark design was reported by (Borghesani, *et al.*, 1995) using the QFT Toolbox for MATLAB, given by

$$G(s) = \frac{3.0787 \times 10^6 s^2 + 3.537 \times 10^8 s + 3.853 \times 10^8}{s^3 + 1.529 \times 10^3 s^2 + 1.064 \times 10^6 s + 4.281 \times 10^7} \tag{5.17}$$

The parameters of the controller given in the QFT Toolbox vary in a very large range (from 1.0 to 10^8) and hence represent a challenging problem to automation design (Chait, 1997).

Optimising QFT designs for this plant was first investigated by Chait (1997), Chait's methods, however, imposes the requirement that the poles of the nominal closed-loop transfer function must be pre-determined, and hence leads to conservative and sub-optimal design. In addition, there exist, in general, no firm guidelines on selecting the poles of the closed-loop nominal system. The controller order resulting from this method is the sum of the closed-loop transfer function and the order of the nominal open-loop plant, and is thus high.

A more recent solution to this benchmark problem was reported in (Chait *et al.*, 1999), where linear programming technique was employed. The optimised controller reported is given by

$$G(s) = \frac{1.6823 \times 10^7 s^2 + 5.9444 \times 10^7 s + 6.9046 \times 10^7}{s^3 + 5..4770 \times 10^3 s^2 + 6.6782 \times 10^6 s + 9.3003 \times 10^6} \tag{5.18}$$

whose loop shaping results are shown in Figure 5.9.

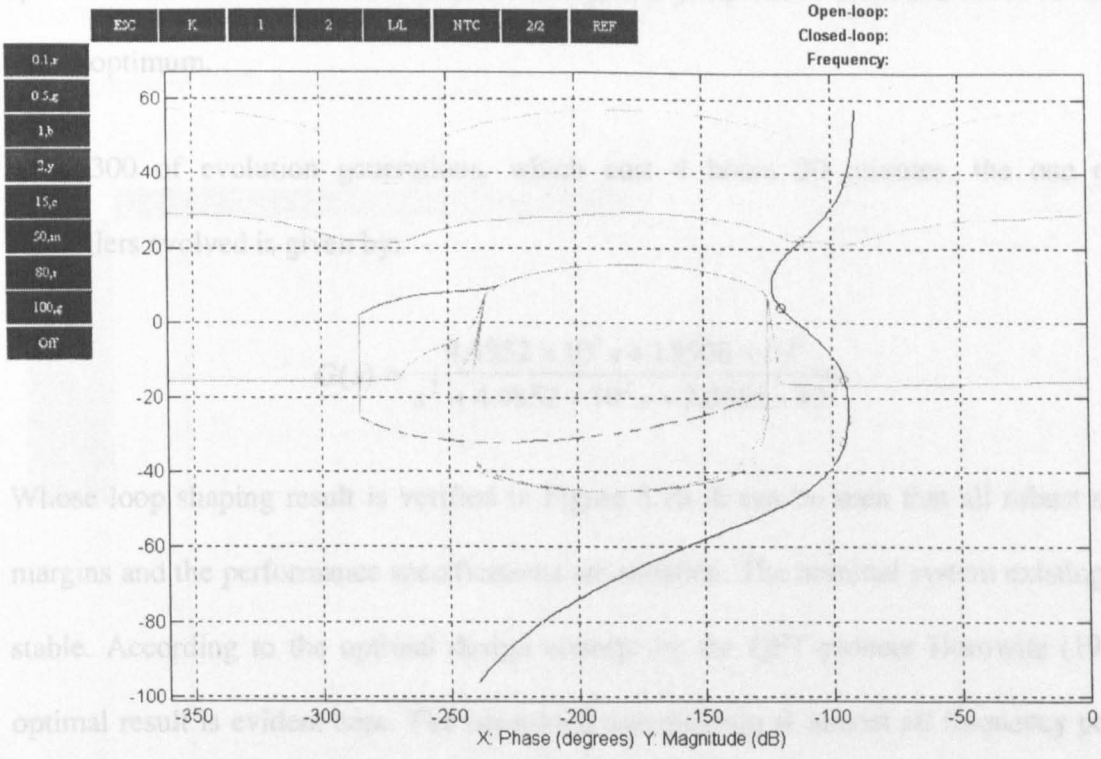

Figure 5.9 Third order controller loop shaping by linear programming

To validate the EA based QFT design-automation technique here, the existing frequency points $\omega \in \{0.1, 0.5, 1, 2, 15, 100\}$ in (Borghesani *et al.*, 1995) and (Chait *et al.*, 1999) are retained. Two more {50, 80} are considered. To check the UHB, 100 more frequency points beyond 100 rad./sec. are added using logarithmic spacing.

To permit optimisation over the order, it is also encoded in the EA. This and all coefficients are allowed to change. Starting from scratch, large ranges of the parameter values are coded in logarithmic scale. The improvements for the controller and filter over 300 generations are shown in Figure 5.14. This also confirms that designing the prefilter less challenging than designing the controller, which appears more open to traps of local optimality. A typical coefficient search history of the controller representing the best in 50 candidates is shown in Figure 5.15. It can be see that the hybridised GA has managed to find and stay with locally

optimal niches. Nevertheless, it has also managed to jump out of them and move towards the global optimum.

After 300 of evolution generations, which cost 4 hours 20 minutes, the one of best controllers evolved is given by:

$$G(s) = \frac{4.4852 \times 10^5 s + 1.5508 \times 10^6}{s^2 + 4.4852 \times 10^2 s + 2.0655 \times 10^5} \tag{5.19}$$

Whose loop shaping result is verified in Figure 5.10. It can be seen that all robust stability margins and the performance specifications are satisfied. The nominal system existing is also stable. According to the optimal design concept by the QFT pioneer Horowitz (1973), an optimal result is evident here. The open-loop transmission at almost all frequency points lie on the corresponding QFT bounds. Comparing with Figure 5.9, it is clear that this automatically evolved second order controller performs better than the best third-order one manually designed us numbering the QFT Toolbox. Also, the high frequency gain of 4.4852×10^5 is smaller than that of the benchmark one of 3.0787×10^6. This means that the evolved controller needs less control effort and is less sensitive to high frequency noise.

While the robust controller is evolved, a prefilter is designed by EA method described in Section 5.3.7:

$$F(s) = \frac{0.29s + 1}{0.095s^2 + 0.040s + 1} \tag{5.20}$$

The ideal and achieved prefilters are shown in Figure 5.11. The closed-loop system gain and the desired frequency response bounds are shown in Figure 5.12, which are indeed very close. Step responses of the overall closed-loop systems with the uncertain plant are verified in the Figure 5.13, which validate the optimal design in the time domain, although the plant

parameters vary in a large range, again robust performance is achieved with the hybridised GA automated QFT design procedure.

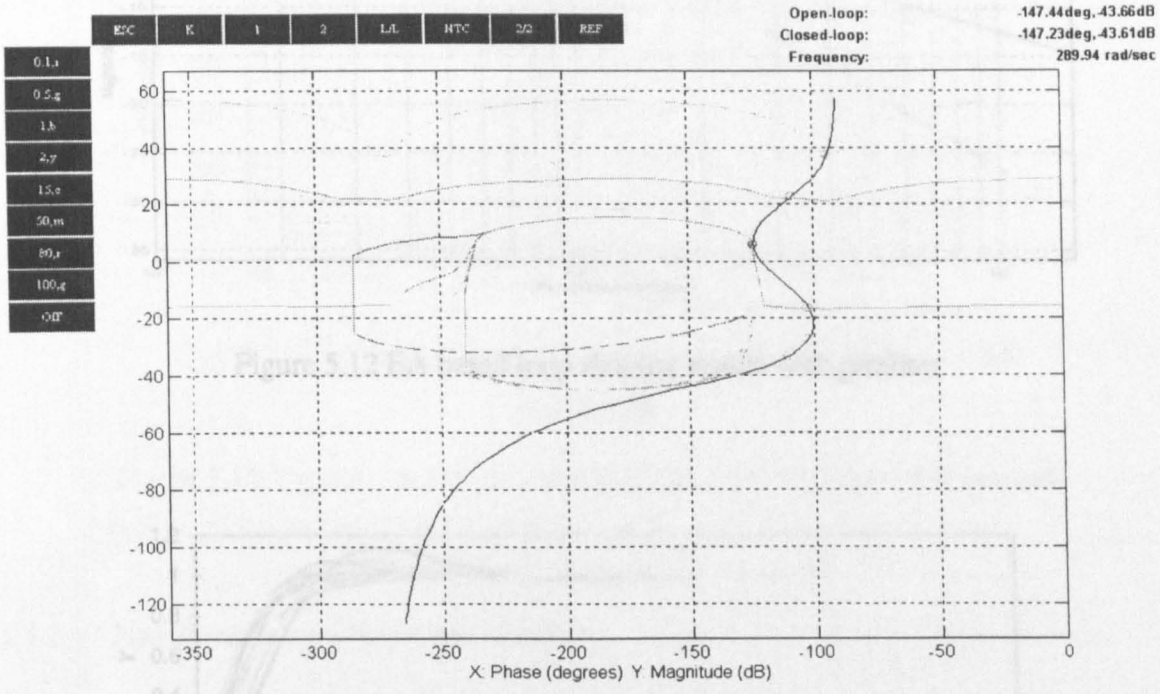

Figure 5.10 Second order loop shaping by an EA

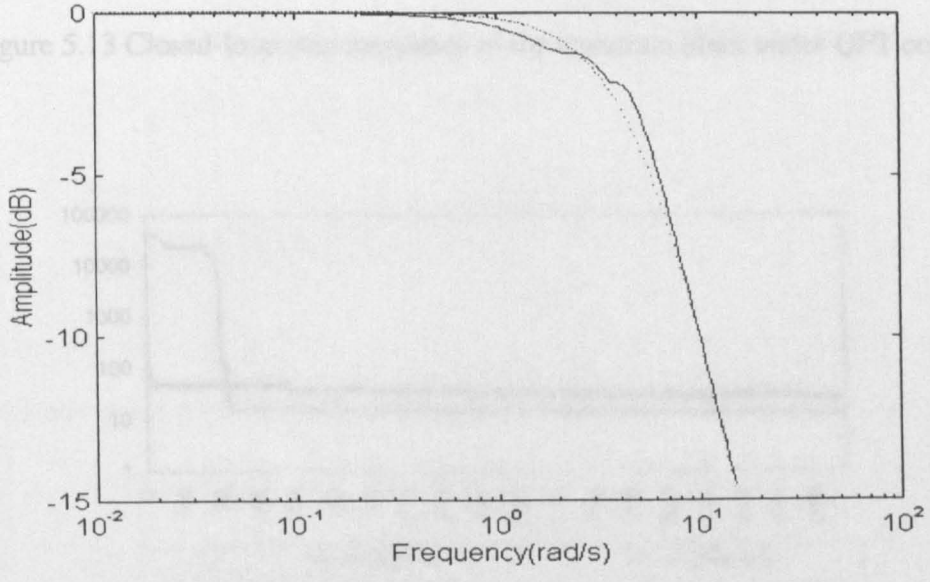

Figure 5.11 Comparison between the ideal (solid line) and best prefiler found

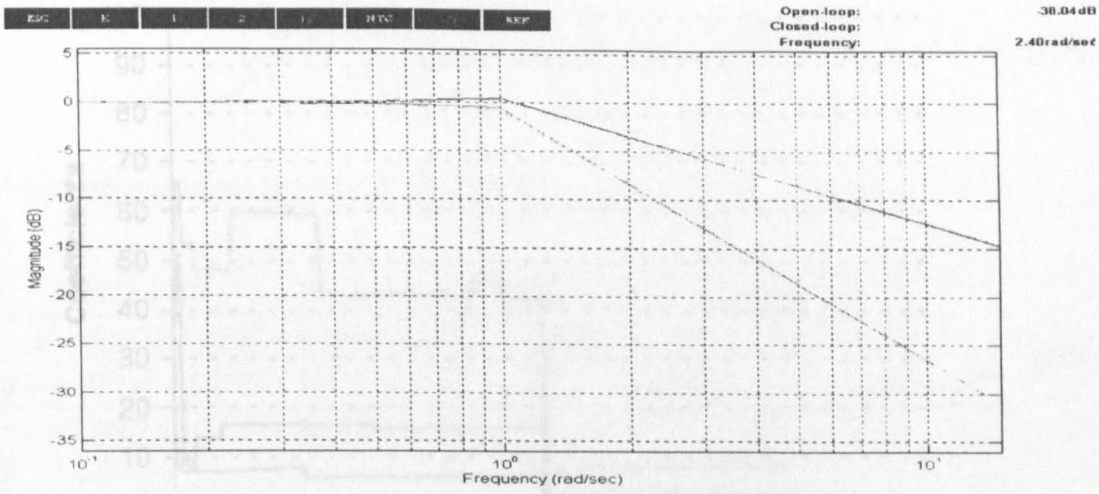

Figure 5.12 EA based loop shaping results with prefilter

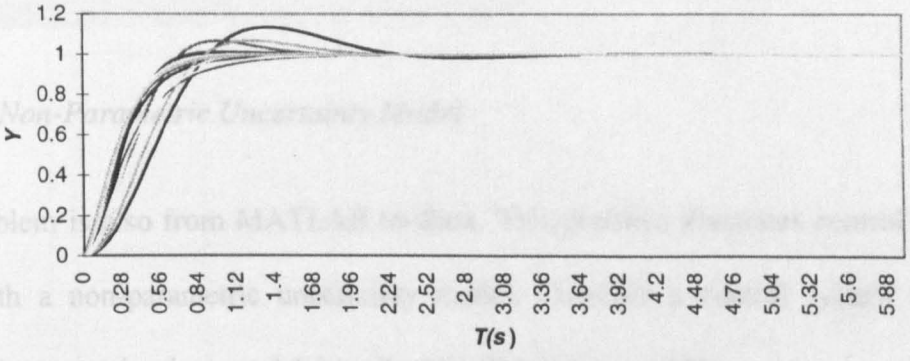

Figure 5.13 Closed-loop step responses of the uncertain plant under QFT control

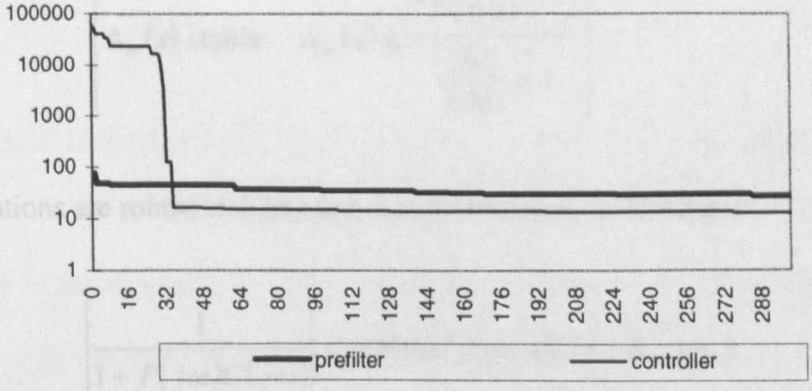

Figure 5.14 Evolution of a second order controller and its prefilter

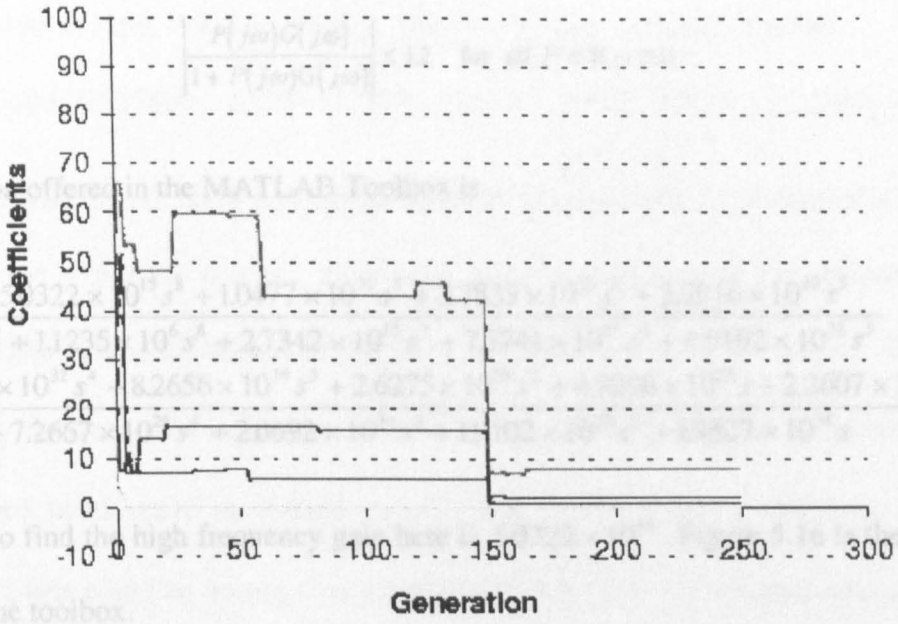

Figure 5.15 Typical convergence of parameters in the optimisation process

5.4.2 *Non-Parametric Uncertainty Model*

This problem is also from MATLAB toolbox. This problem illustrates control design for a plant with a non-parametric uncertainty model. Consider a control system with a non-parametric uncertain plant model described by (Borghesani, 1995),

$$\mathcal{P} = \left\{ \begin{array}{l} P(s) = \dfrac{10}{s(0.1s+1)}\left(1+\Delta_m(s)\right): \\ \\ \Delta_m(s) \text{ stable.} \quad \Delta_m(s) < \dfrac{0.9\left(\dfrac{j\omega}{0.91}+1\right)}{\dfrac{j\omega}{1.001}+1} \end{array} \right\} \tag{5.21}$$

The specifications are robust stability and robust sensitivity according to

$$\left| \frac{1}{1+P(j\omega)G(j\omega)} \right| \le 0.089\omega^2, \text{ for all } P \in \mathcal{P}, \omega \le 5 \tag{5.22}$$

The associated QFT robust stability constraint is given by

$$\left|\frac{P(j\omega)G(j\omega)}{1+P(j\omega)G(j\omega)}\right| \le 1.2 \quad \text{for all } P \in \mathcal{P}, \omega \ge 0 \tag{5.23}$$

The solution offered in the MATLAB Toolbox is

$$G(s) = \frac{\begin{array}{l}5.0322\times10^{15}s^8 + 1.0477\times10^{21}s^7 + 2.7833\times10^{25}s^6 + 2.0916\times10^{29}s^5 \\ + 3.1804\times10^{32}s^4 + 8.2656\times10^{34}s^3 + 2.6275\times10^{36}s^2 + 4.9298\times10^{36}s + 2.2607\times10^{36}\end{array}}{\begin{array}{l}s^9 + 1.1235\times10^6 s^8 + 2.7342\times10^{12}s^7 + 7.5741\times10^{17}s^6 + 4.9192\times10^{22}s^5 \\ + 7.2667\times10^{26}s^4 + 2.6692\times10^{30}s^3 + 1.9102\times10^{33}s^2 + 1.9327\times10^{35}s\end{array}} \tag{5.24}$$

It is easy to find the high frequency gain here is 5.0322×10^{15}. Figure 5.16 is the loop shape given by the toolbox.

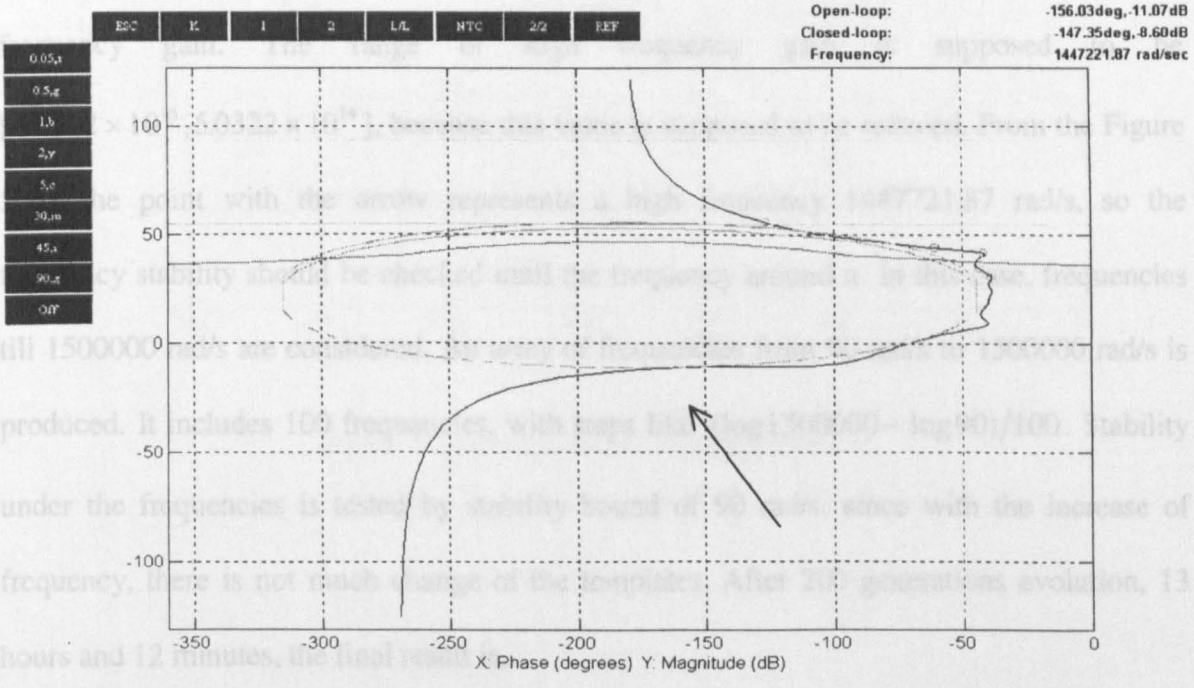

Figure 5.16 Non-parametric uncertainty design results in MATLAB QFT Toolbox

The objective is automatically to find a controller such that all the closed-loop specifications are satisfied with the cost of feedback as small as possible. From the graph, it seems that there is little room to improve the design, because the gain-phase curve is too close to the bounds. After checking the zeros and poles of the design,

Zeros: -178490, -17855, -9923.3, -1597.3, -280.44, -34.585 -1.2129, -0.75590

Poles: $411640 \pm j1515800$, -67553, -13884, -4143.4,-779.63, -120.95, -2.13780, 0

A controller with the 8th order is designed by the EA method, and an integrator is kept in this design. In the evolution process, zeros and poles are chosen as the parameters to be optimised. The ranges of zeros and poles are supposed to be [1/2 * *P or Z*, 2 * *P or Z*], Apparently, the real pole and zero farthest to the origin should be forgotten. The two underdamped poles could be thought as $s^2 + 2\varsigma\omega_n s + \omega_n^2$, when $\varsigma = 0.26, \omega_n = 1570700$, then their ranges could be arranged as $[0.5\omega_n, 2\omega_n]$, and [0.4, 1] for damp ratio according to suggestion given by Horowitz (1992). Another the parameter should be optimised is the high frequency gain. The range of high frequency gain is supposed to be $[5.0322 \times 10^{15}, 5.0322 \times 10^{14}]$, because this value is supposed to be reduced. From the Figure 5.16, the point with the arrow represents a high frequency 1447721.87 rad/s, so the frequency stability should be checked until the frequency around it. In this case, frequencies till 1500000 rad/s are considered. An array of frequencies from 90 rad/s to 1500000 rad/s is produced. It includes 100 frequencies, with steps like $(\log 1500000 - \log 90)/100$. Stability under the frequencies is tested by stability bound of 90 rad/s, since with the increase of frequency, there is not much change of the templates. After 200 generations evolution, 13 hours and 12 minutes, the final result is,

Zero: -18975.0, -6663.35, -1426.03, - 249.770, - 34.3813, - 1.04383, -1.46220

Poles: $-443030 \pm j993470$, -56782.5, - 10403.4, - 3691.90 - 590.484 - 102.043, 0

The zeros and poles confirms that no right plane cancellations. The high frequency gain is 1.00156×10^{15}, which is better, and the pair of underdamped poles represent $\omega_n = 1087780, \varsigma = 0.407782$, the controller could be shown as

$$G(s) = \frac{1.0016\times10^{15}s^7 + 2.7394\times10^{19}s^6 + 1.7103\times10^{23}s^5 + 2.2763\times10^{26}s^4}{s^8 + 9.5764\times10^5 s^7 + 1.2476\times10^{12}s^6 + 8{,}5476\times10^{16}s^5} \quad (5.25)$$
$$\frac{+5.3202\times10^{28}s^3 + 1.6832\times10^{30}s^2 + 3.9605\times10^{30}s + 2.3632\times10^{30}}{+1.0531\times10^{21}s^4 + 3.2744\times10^{21}s^3 + 1.8470\times10^{27}s^2 + 1.5549\times1029s}$$

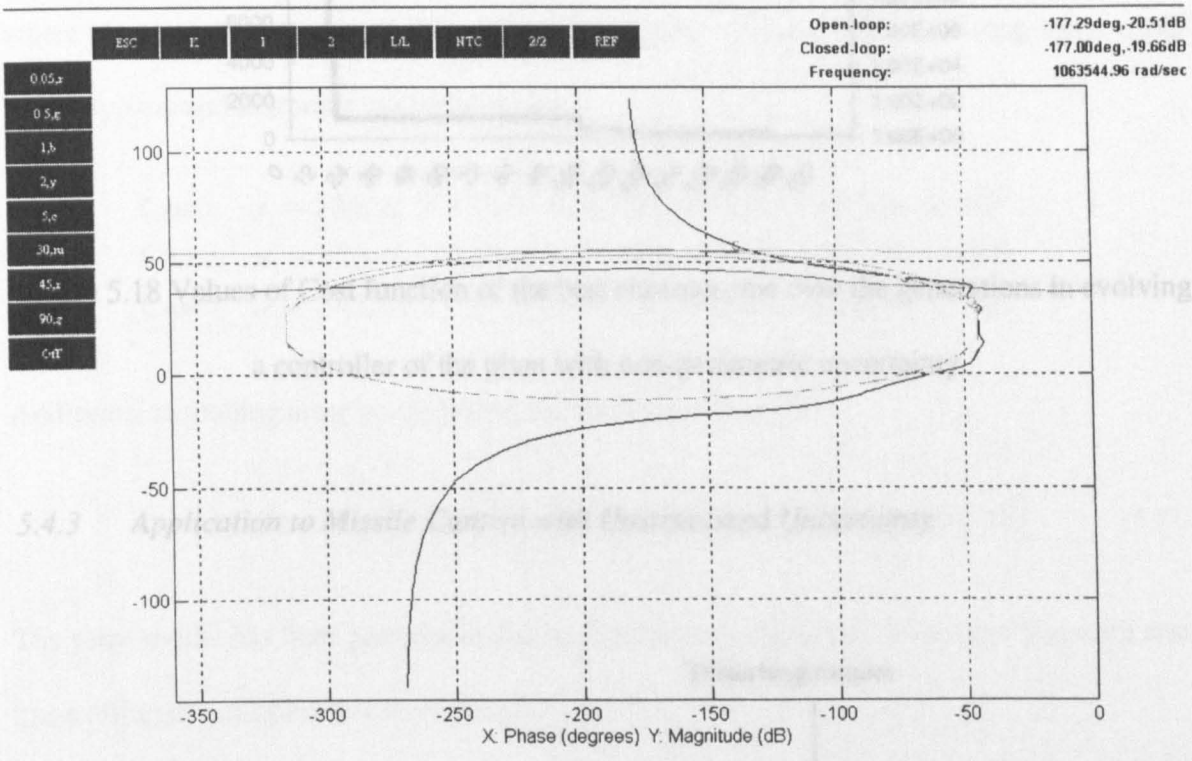

Figure 5.17 Design results from EA method

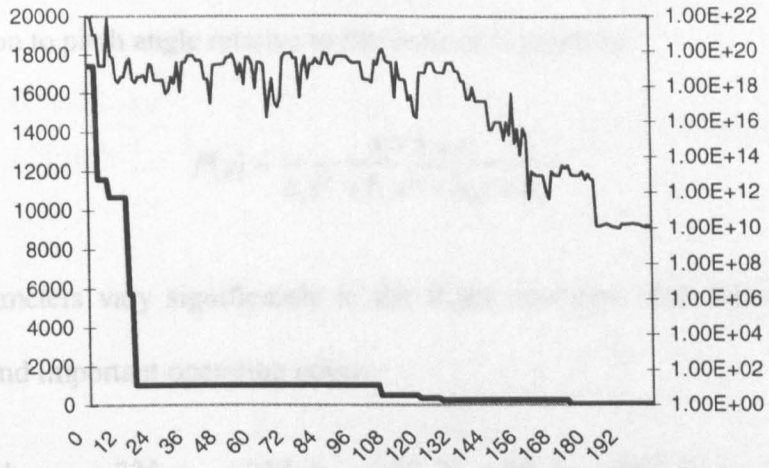

Figure 5.18 Values of Cost function of the best chromosome over the generations in evolving a controller of the plant with non-parametric uncertainty

5.4.3 Application to Missile Control with Unstructured Uncertainty

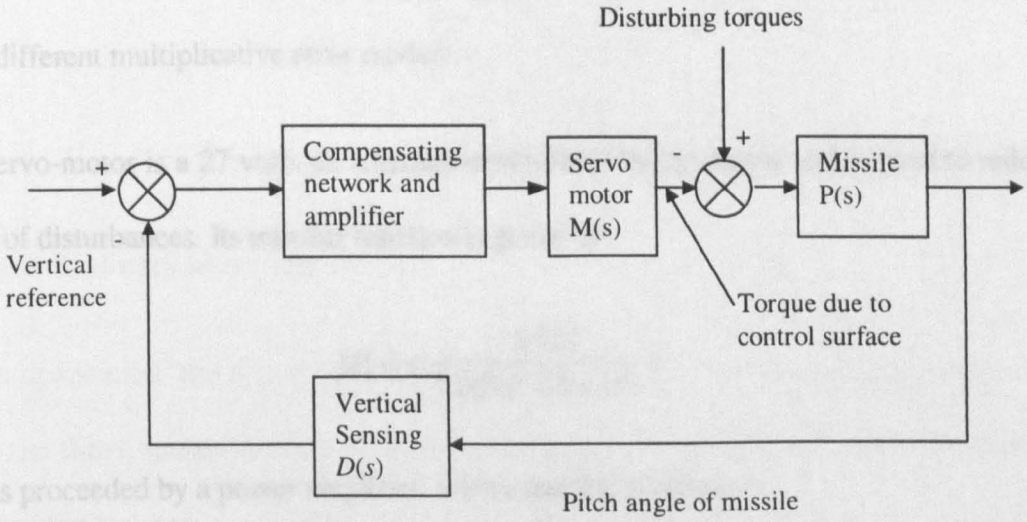

Figure 5.19 Missile control system

A missile control application is shown in Figure 5.19. The control is focused on the vertical trajectory in the design of autopilot. The missile is roll-stabilised and has a cruciform wing configuration. Dynamic modelling of the missile involves aerodynamics, gravitational and

propulsive forces (Gille, 1959). A simplified open-loop transfer function relating the control-surface deflection to pitch angle relative to the vertical is given by

$$P(s) = \frac{a_1 s + a_2}{b_1 s^3 + b_2 s^2 + b_3 s + b_4} \tag{5.26}$$

where the parameters vary significantly in the flight envelope. The following three cases reflect typical and important operating points:

Case1: $a_1 = 335, a_2 = 237, b_1 = 20.7, b_2 = 39, b_3 = 257, b_4 = -9.5$
Case2: $a_1 = 315, a_2 = 227, b_1 = 19.7, b_2 = 37, b_3 = 247, b_4 = -9.0$
Case3: $a_1 = 345, a_2 = 247, b_1 = 23.7, b_2 = 36, b_3 = 267, b_4 = -10.5$

Additional modelling error is covered by the multiplicative form

$$\mathcal{P} = \left\{ P_i(s)\left(1 + \Delta_i(s)\right): \quad \Delta_i(s) \text{ stable}, \left|\Delta_i(s)\right| < R_i, R_i = \left\{0.1, 0.05, 0.075\right\} \right\} \tag{5.27}$$

The plant model has both parametric and unstructured uncertainties. Note also that each case has a different multiplicative error model.

The servo-motor is a 27 volts dc armature-controlled electric motor and is used to reduce the effect of disturbances. Its transfer function is given by

$$M(s) = \frac{1/107}{0.001 s^2 + 0.13 s + 1} \tag{5.28}$$

This is proceeded by a power amplifier, whose transfer function is

$$A(s) = \frac{1}{0.01 s + 1} \tag{5.29}$$

The rate gyro vertical sensor measures pitch angles according to

$$D(s) = 27 \frac{40 s}{s^2 + 1.2 \times 40 s + 40^2} \tag{5.30}$$

The specifications are robust margins

$$\left|\frac{P(j\omega)G(j\omega)M(j\omega)A(j\omega)D(j\omega)}{1+P(j\omega)G(j\omega)M(j\omega)A(j\omega)D(j\omega)}\right| \le W1, \quad \forall\, P \in \mathcal{P}, \omega \ge 0 \tag{5.31}$$

where

$$W1 = \begin{bmatrix} W1_1 \\ W1_2 \\ W1_3 \end{bmatrix} = \begin{bmatrix} 1.3 \\ 1.2 \\ 1.25 \end{bmatrix} \tag{5.32}$$

corresponds to each plant case. The robust input disturbance rejection is

$$\left|\frac{P(j\omega)}{1+P(j\omega)G(j\omega)M(j\omega)A(j\omega)D(j\omega)}\right| \le W2, \quad \text{for all } P \in \mathcal{P}, \omega \in [1,8] \tag{5.33}$$

where

$$W2 = \begin{bmatrix} W2_1 \\ W2_2 \\ W2_3 \end{bmatrix} = \begin{bmatrix} 0.040 \\ 0.036 \\ 0.038 \end{bmatrix} \tag{5.34}$$

corresponds to each plant case.

In this application, the objective is to find a single controller that meets all specifications at all of the three operation points. Extensive manual loop shaping using the MATLAB QFT Toolbox has resulted in a 'good' controller given by (Borghesani, 1995),

$$G(s) = \frac{1.7204\times10^{14}s^8 + 1.1498\times10^{17}s^7 + 2.4023\times10^{19}s^6 + 2.2031\times10^{21}s^5}{\begin{array}{l} s^9 + 1.5552\times10^{4}s^8 + 1.2195\times10^{8}s^7 + 4.4676\times10^{11}s^6 + 9.6083\times10^{14}s^5 \\ + 1.0840\times10^{23}s^4 + 3.1731\times10^{24}s^3 + 4.8509\times10^{25}s^2 + 4.2337\times10^{26}s + 1.3693\times10^{27} \\ + 1.1752\times10^{18}s^4 + 7.2671\times10^{20}s^3 + 1.7063\times10^{23}s^2 - 3.0167\times10^{24}s - 1.1801\times10^{24} \end{array}} \tag{5.35}$$

The loop shaping results in the Nichols Chart is

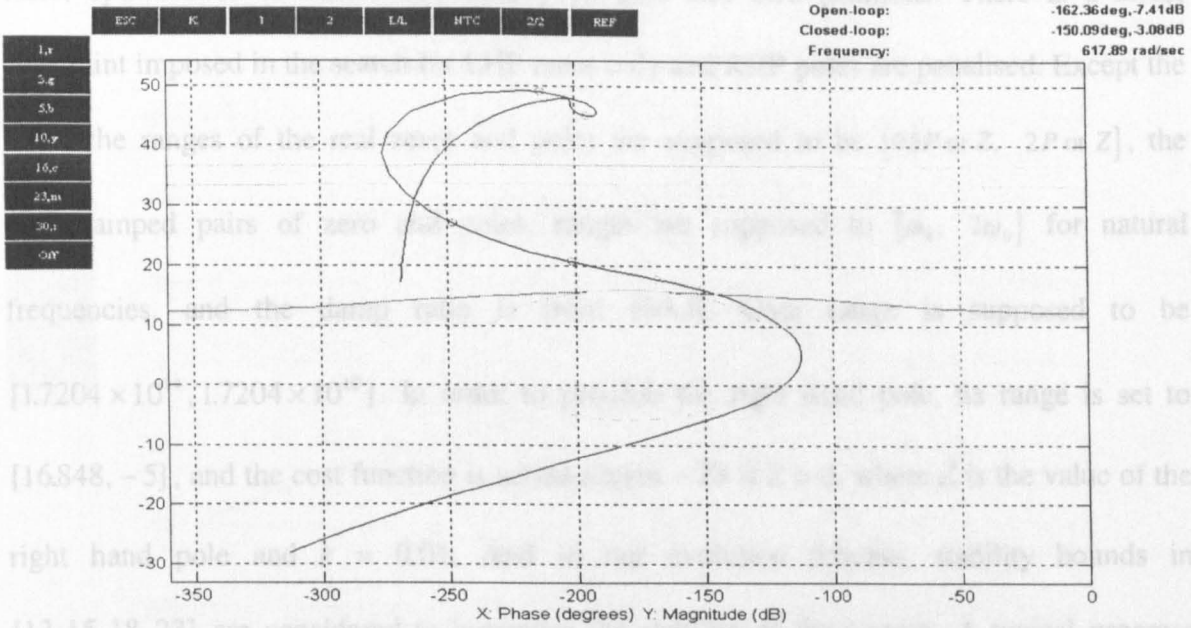

Figure 5.20 Loop shaping for missile controller in MATLAB QFT Toolbox

This design presents a 'high frequency gain' being 1.7204×10^{14}, and the poles and zeros of the design are found as

Zeros:, -377.3, -144.42, $-8.9383 \pm j1.0620$, -7.017, -6.1421, $-2.6176 \pm j3.2764$

Poles: $-5129.3 \pm j5232.9$, $-1270.2 \pm j1693.6$ -1133.0, $-818.36 \pm j35.211$, -3.8289, 16.848

Since there exist no zeros of the plant in the RHP, the right plane pole of the controller has been allowed. It means that there is not RHP pole-zero cancellation. There however exist three under-damped complex pole pairs.

In this application the hybridised GA automated QFT design refinement is employed

1. to reduce the numbers of poles and zeros by one each;
2. to reduce the high-frequency gain ;and
3. to improve damping transmitted to the actuator.

Here, optimisation is carried out directly on pole and zero positions. There is a direct constraint imposed in the search for LHP zeros only and RHP poles are penalised. Except the RHP, the ranges of the real zeros and poles are supposed to be $[0.5P \text{ or } Z,\ 2P \text{ or } Z]$, the underdamped pairs of zero and poles, ranges are supposed to $[\omega_n,\ 2\omega_n]$ for natural frequencies, and the damp ratio is from $[0.4, 1]$. Gain range is supposed to be $[1.7204 \times 10^{14}, 1.7204 \times 10^{10}]$. In order to penalise the right hand pole, its range is set to $[16.848, -5]$, and the cost function is added a term $-Zk$ if $Z > 0$, where Z is the value of the right hand pole and k = 0.01. And in our evolution process, stability bounds in [13, 15, 18, 23] are considered to guarantee the stability of the system. A typical progress chart in terms of minimising the penalties and costs in the evolutionary design automation process is shown in Figure 5.22, where the bold line represent an average controller and the thin the best one in each generation. After 200 generations of artificial evolution, which cost 11 hours 10 minutes, the following poles and zeros have emerged:

Zeros: $-71.375 \pm j74.9428$, $-18.0921 \pm j33.4431$, -9.3194, $-2.7310 \pm j5.3806$

Poles: -1833.1, -1453.3, $-526.80 \pm j571.97$ $-159.65 \pm j243.72$, -0.24929, 0.2490

The controller obtained here is

$$G(s) = \frac{1.1997 \times 10^{11} s^7 + 3.8685 \times 10^{13} s^6 + 4.0048 \times 10^{15} s^5 + 1.7293 \times 10^{17} s^4}{\begin{array}{l} s^8 + 1.7739 \times 10^4 s^7 + 5.0135 \times 10^7 s^6 + 5.3648 \times 10^{10} s^5 + 3.2007 \times 10^{13} s^4 \\ + 5.1598 \times 10^{18} s^3 + 5.7240 \times 10^{19} s^2 + 3.1025 \times 10^{20} s + 1.0493 \times 10^{21} \\ \hline + 8.3662 \times 10^{15} s^3 + 1.3674 \times 10^{18} s^2 - 1.1181 \times 10^{14} s - 8.4878 \times 10^{16} \end{array} \quad (5.36)$$

Automatic loop shaping result is verified in Figure 5.21.

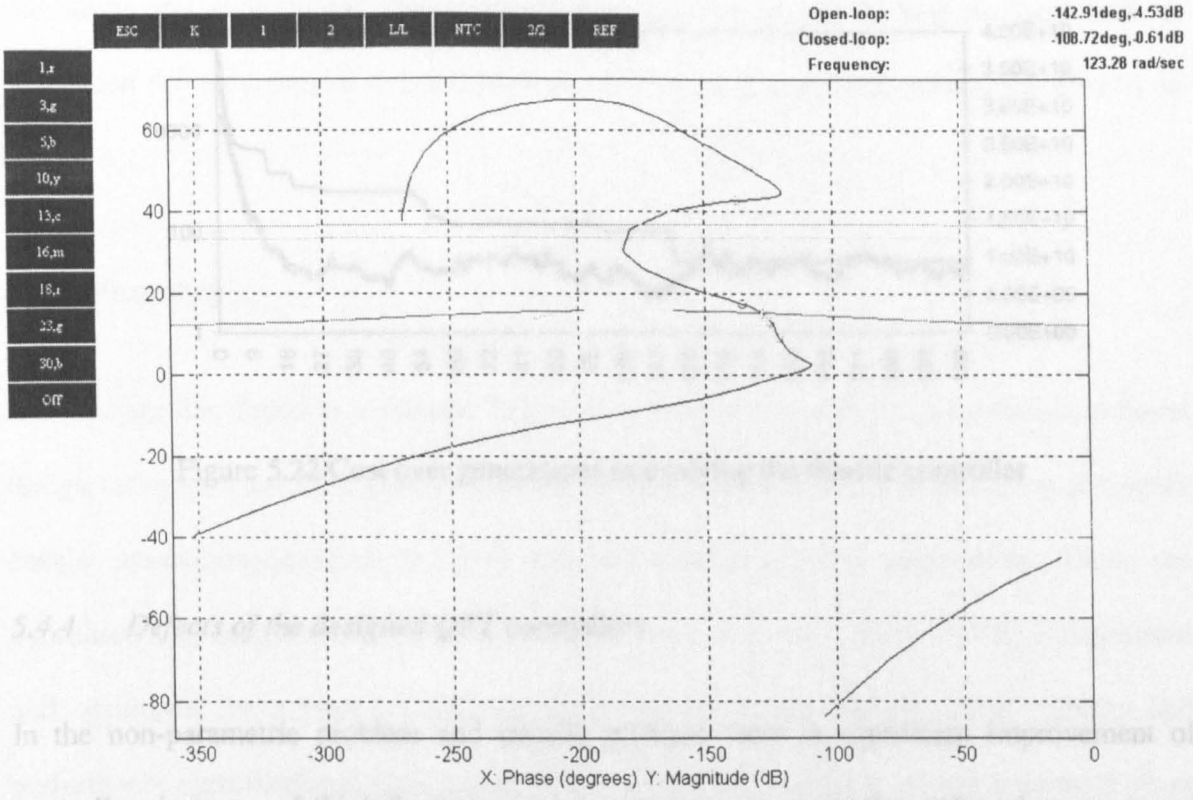

Figure 5.21 EA-based loop shaping for missile controller

It can be seen that the evolutionary computation based optimal and automated design approach offers an improved controller with:

1. A lower order.
2. Better-behaved open-loop frequency response and loop shaping for enhanced robustness.
3. High frequency gain reduced by 63 dB.
4. The under-damped poles reduced by one pair.
5. The relatively more stable RHP controller pole closer to the imaginary axis.

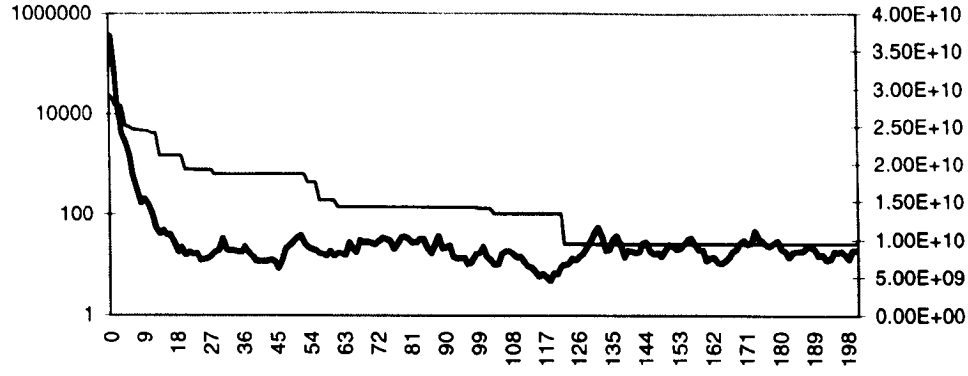

Figure 5.22 Cost over generations in evolving the missile controller

5.4.4 *Defects of the designed QFT controllers*

In the non-parametric problem and missile problem there is significant improvement of controllers in terms of 'high frequency gain' with reducing controller order when compared with designs from MATLAB QFT Toolbox designs (Borghesani, Chait and Yaniv, 1995). However, controllers achieved here still have too much high frequency gain: 1.1997×10^{11} (missile) and 1.7204×10^{14} (parametric) and too high order: 8^{th} (missile and parametric). These controllers can not be manufactured physically. One of choices suggested is the reduction of design requirement. Another choice suggested is alternative method: H_∞ control design strategy. The strategy is to optimise system performance in the worst case. Instead of translating time domain requirement into frequency domain requirement in QFT control, H_∞ control optimises the time domain response directly (Doyle *et al.*, 1989; Zhou *et al.*, 1996). So it is possible to achieve better results.

Another problem in the design is too long time (11 hours 10 minutes for missile and 13 hours and 12 minutes for parametric). The reason for this is that the evaluation process is finished by calling a native code subroutine, which is produced by compilation of MATLAB code into executable file (The MathWorks, Inc., 1997). So exchange of data between the

subroutine and evolutionary engine is achieved by writing and reading the hard disk. A suggestion for reduction of the optimisation time is to program the subroutine directly in Java.

5.5 Summary

This chapter has aimed to overcome deficiencies in manual and existing optimisation based design techniques for QFT control systems. In particular, it has developed an evolutionary design automation approach to QFT, oriented towards practical applications. Using the hybridised GA design suite, the design of QFT controllers for uncertain plant is automated and optimised for a minimal 'cost of feedback' while meeting all robust stability and performance specifications. This is particularly helpful with unstable or non-minimum phase plants or plants for which it is difficult to find a stabilising controller. This chapter has also shown that such an automated design procedure can be used to further tune existing controllers with both reduced order and improved performance. This is particularly useful when any manual loop shaping improvements can only be made by adding more pole-zero sections to the controller. With or without an *a-priori* design, this objective multi-dimensional multi-optimal design technique may be employed to maximise the closed-loop performance under practical constraints. This technique is illustrated and verified through a well-known QFT benchmark design problem, a non-parametric problem and a missile control application example.

Chapter 6 Enabling Neural Control in Forward Path

In the previous two chapters, design automation of linear PID and QFT controllers are extensively studied. However, linear controllers are often inadequate to deal with saturation, if there is a rate limit or other hard nonlinearities, which are encountered in many practical applications.

In this chapter, therefore, the popular PID structure is to be extended to the nonlinear building block based on neural networks to deal with hard nonlinearities and other practical constraints. Novel neural PD and PID type nonlinear controllers are proposed for use in the feedforward path in the same way as conventional linear controllers. They are to be tested with IFAC benchmark problems. In order to tackle local optimum problems, the hybridised GA based design environment developed in this thesis is used to achieve the optimal weights and structures globally. Section 6.1 highlights existing structures and training methods of artificial neural networks used in control. Forward path direct neural control architectures are developed in Section 6.2 and training mechanisms for them are developed in Section 6.3. Section 6.4 validates the methodology proposed for three different plants: a ship regulation problem, an inverted pendulum problem and an asymmetrically nonlinear coupled water tank. Summary is given in Section 6.5.

6.1 Introduction

Artificial neural networks (ANNs) mimic the function of human brain, which are universal and arbitrary function approximators. Similar to their biological equivalent they have capability of learning, storing and judging data. Thus, they have far ranging applications. Most of them are in image processing and pattern recognition. However, they have also been successfully applied to modelling of complicated, irregular, nonlinear, time-varying,

irrational and stochastic systems, including to learning inverse dynamics for controller design (Mei *et al.*, 1998; Li, 1999 and Zurada, 1992).

It is known that most of systems to be controlled are nonlinear systems, but basically they could be described as a linear model plus "hard nonlinearities", such as delay, Coulomb friction, saturation, dead-zones, backlash, and hysteresis (Slotine and Li, 1991). Some simple models, such as FOPPT, are widely used to describe the process plants as well. Here, our discussion focuses on a linear model plus saturation case. The saturation includes amplitude and rate saturation. It is shown that neurocontroller developed here has the advantage than the conventional ones if there is saturation for rate. However, if there is just amplitude limits, a linear controller trained through time domain simulation does perform as well as a neurocontroller.

6.1.1 Existing Neural Control Structures

There are two different ways of applying a neural network to control engineering. One is to use the network to adjust the parameters of a conventional controller (Rogers and Li, 1993), which is shown in Figure 6.1. The other is the use of the ANN as a direct controller which is termed neurocontroller (Rogers and Li, 1993; Psaltis *et al.*, 1989). The latter form is discussed in the thesis.

The most common structure used in ANNs is that of *multi-layer* perceptrons. Within this structure there are several perceptrons arranged in layers. Each perceptron is only connected to one in an adjacent layer. The inputs of the perceptron are weighted. However, in our design, there is not a threshold. That is explained later. Afterwards all inputs are summed up and go through the perceptron's activation function.

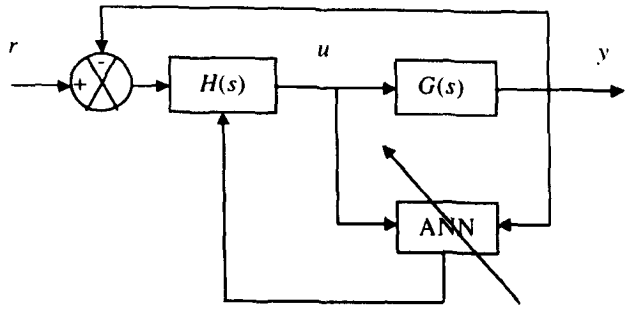

Figure 6.1 ANN for adaptive control

It has been shown that a PD-based non-linear neurocontroller can be enabled by evolutionary training, which produces good results in terms of stability, transient behaviour and robustness. The evolutionary optimisation was structural and parametrical and the controllers could even handle constraints easily afterwards (Li *et al.*, 1996a; Brune, 1998). However, in our research, results that are more interesting are found. Novel neurons without threshold are used. Instead of using threshold to offset the steady-state errors, integral of errors is used as the input, which makes the neurocontroller be able to eliminate the steady-state errors with different operating points. These results have encouraged us to develop a generic neurocontroller that is applicable to any plant.

In order to simulate and evolve a controller, open-loop plant data-based generic design rules for linear controllers have been derived (Li *et al.*, 1996a). It is also proved that convolution can serve as a high fidelity means, in order to get an exact representation of the plant direct from I/O data (Psaltis *et al.*, 1988; Ichikawa *et al.*, 1992; Cluett *et al.*, 1991). These features help to design a generic linear controller for any type of plant, which is supposed to work well in some region around the operating point. Therefore, the convolution method is included in the design of a generic non-linear controller in this chapter. However, it must be pointed out, for an unstable process, the convolution method cannot be used to simulate the process.

6.1.2 *Conventional Means of Training*

There are two different ways of implementing a neurocontroller (Psaltis *et al.*, 1988). The first is called *general learning* and the second *specialised learning*. In *general learning* the controller is trained off-line. ANN learns the inverse dynamics of the plant. Then the trained ANN is put into the control loop, and the control system is supposed to follow any set point command. However, this method suffers from apparent disadvantages. When the plant inverse is not uniquely defined, a major problem arises. This occurs for a plant, when more than one value of u exists that corresponds to one value of y. Figure 6.2 illustrates this limitation of the plant inverse identification for the one-dimensional case. In the discussed case, the neural network modelling the inverse attempts to map a single input y^* to one of the two target responses u_1 or u_2. It may be that the eventual mapping learned would somewhat tend to average the two desired

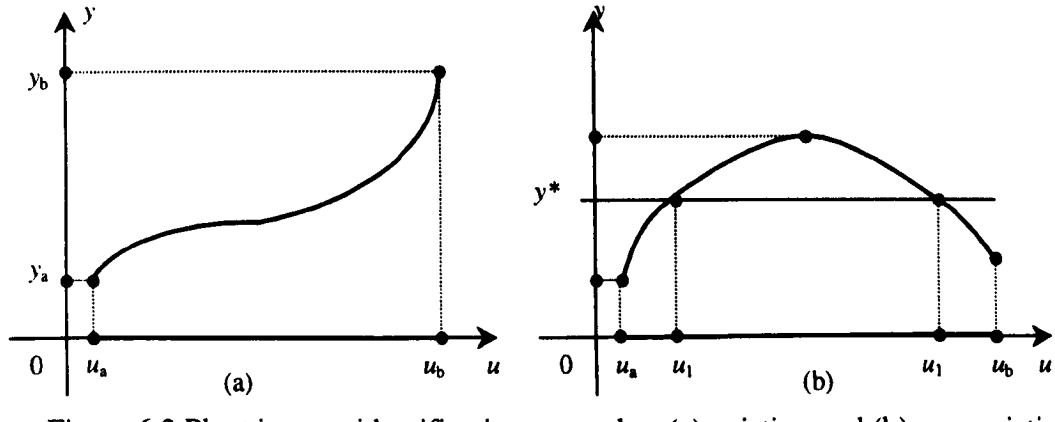

Figure 6.2 Plant inverse identification examples: (a) existing and (b) non-existing

There is another approach in *general learning*. ANNs are trained to behave like a specific form of a conventional controller. Input and output data of the controller in a normal closed-loop fashion in conjunction with the plant are used as a guide for the training. Burns (1995) did some interesting research about the method. Unar (1999) used it for ship steering control. Their results are very interesting. However, it could be expected that these kinds of

controllers' performances are just similar to the conventional controllers, which they are supposed to approximate.

In *special learning* the controller is trained on-line or from plant model directly. Because of the desired control signal, which implies that the desired output of the neural network is unknown, the plant is compared with the command signal and the error is backpropagated through both the plant and neural network.

However, both training methods are based on the error-backward propagation method. In these methods, the error is backward propagated through the neurons to adjust the corresponding weight. This method has several disadvantages:

1. Gradient guidance always needs a performance index which is differentiable, i.e. structural components such as a switch cannot be evaluated;
2. Constraint handling is difficult because of gradient search;
3. The error at the input of an ANN is hard to minimise;
4. It is a local optimisation method, i.e., no global optimum is found.

6.2 Forward Path Direct Neurocontroller Architectures

6.2.1 Network Structure

The architecture is based on that of a conventional PID controller. The reason is that a proportional input is not sufficient to deal with transient behaviours. The discrete equation of a PID-controller is given by:

$$u(k) = K\left\{e(k) + \frac{T_D}{T_0}\left[e(k) - e(k-1)\right] + \frac{T_0}{T_I}\sum_{i=0}^{k} e(k)\right\} \tag{6.1}$$

where k is the time index, K the proportional (and overall) gain, T_D the time-constant of the differentiator, T_I the time-constant of the integrator, T_O the sampling period, $u(k)$ the output of the controller and $e(k)$ the discrete error signal between the desired output and actual output of the plant.

This behaviour is mapped directly into the neurocontroller. The three inputs for the neural network just represent 'three terms' in PID. According to different requirements of problems, it could be chosen as PD and PI controller. From this point, this structure can be understood as nonlinear PID controller. In another way, the idea behind is close to fuzzy control. After training, for different combinations of error, change of error, summation of error, there should be a different control action, just like the fuzzy control table. However, in many cases, neural networks can enjoy the benefit of smoothness. As a structure of neurocontroller, three layers are chosen. The first layer is to distribute the inputs to the middle layer. The number of neurons in the middle layer is to be optimised, but contains only one neuron in the simplest case. The whole structure is shown in Figure 6.3.

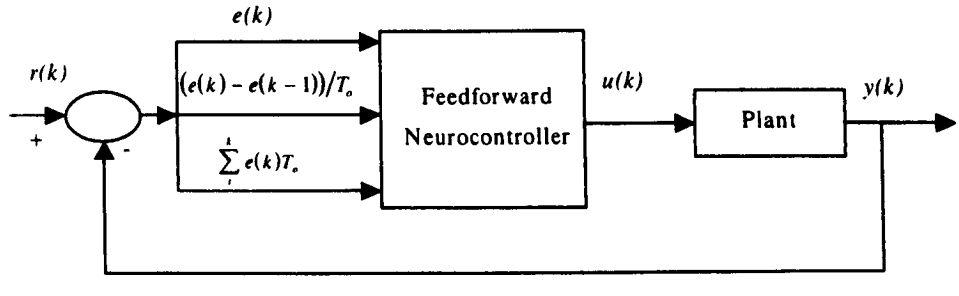

Figure 6.3 Structure of a neurocontroller embedded in feedforward path

6.2.2 *Design of Neurons*

In the simplest form, an artificial neuron can be modelled as a device (usually nonlinear) having one or more inputs. An input to an artificial neuron is either an input of the network of which the neuron is a part, the output of another neuron, or its own output. As can be seen from the Figure 6.4, an artificial neuron first multiplies each input by a factor called weight. The neuron then calculates the sum of all the weighted inputs. Finally, the neuron applies an activation function f to the weighted sum. Mathematically the artificial neuron of can be expressed as following:

$$y = f\left(\sum_{i=1}^{p} w_i x_i - b\right) \tag{6.2}$$

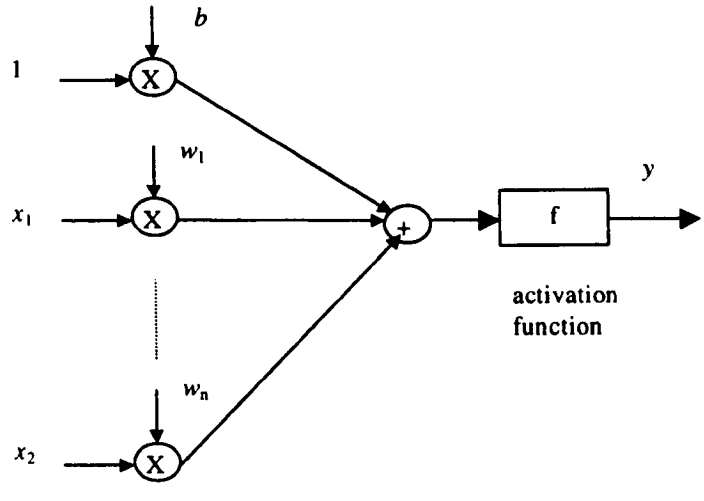

Figure 6.4 Single neuron

For PD control case, since there is a steady-state error for type 0 system, this method is used to control the systems with their own integrator. It means that when the systems approach the steady-state, it is not necessary for an input to maintain its output. Therefore, it is ideal that the output is an odd function of error and derivative of error. So if the activation function is chosen as

$$f(x) \triangleq \frac{2}{1+\exp(-\lambda x)} - 1 \tag{6.3}$$

the control action taken by the PD neurocontroller is an odd function of the inputs. It can guarantee when the error is 0 and the derivative of error is 0, no control action is taken.

To control a type 0 system, although the threshold could probably offset the steady-state error, it only works at the specific operating points. Generalisation of the neural control could be damaged by such a choice. However, if another term the integration of error is added in the input end of the neurocontroller, the problem is solved. Our tests show that it can work at different operating points. Then it becomes a PID type neurocontroller.

6.3 Evolving Direct Neurocontrollers

6.3.1 Evolutionary Selection and Training

EAs have led to the breakthrough in enabling an ANN to be deployed in the same way as conventional linear or other nonlinear controllers such as sliding mode and fuzzy controllers (Li, *et al.* 1996a). As discussed in the Chapter 3, these algorithms use a selection scheme based on *Darwin*'s survival-of-the-fittest law according to a given fitness function. In addition, they perform random perturbation and some information exchange between solutions. This enables them to reduce search time compared to exhaustive search and to find a global optimum (Vesin and Gruter, 1999; Mackay *et al.*, 1996; and Goldberg, 1989). The advantage of such EAs is that they overcome the problems with error backward propagation mentioned in Section 6.2.1.

In this work, a hybridised genetic algorithm based design environment, which is implemented in Chapter 3, has been applied. A neurocontroller can always be evolved by an EA on the following conditions:

1. The system is analysable, i.e., the performance of candidate designs can be evaluated.
2. A performance index has values with more information than simple *true-or-false* answer.

6.3.2 Parametric and Structural Design

Each solution is represented by a vector containing all the weights and threshold weights in the way, same as they appear in the network. The range of the weights could be chosen from 5 to 20, according to different problems.

The structure is optimised by using a growing mechanism. The initial structure is the minimal 4-1-1 network. After the algorithm has found an optimum, another neuron is added in the middle layer and the optimisation continues. The best solutions of the previous structure are included and new weights in the new structure are set to zero. After the best solution for this new structure is found, it is compared with the previous best one. If there is any improvement, another neuron is added and the new best optimum takes the place of the previous one. Then steps above are repeated. If there is no improvement the previous structure is retained and the process stops.

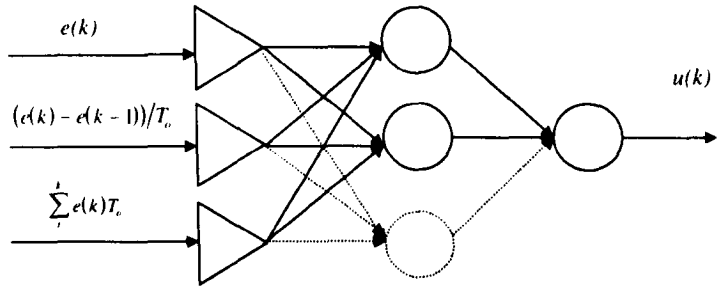

Figure 6.5 Growing a neurocontroller with architectural optimisation

6.3.3 Design Direct from Response Data

Convolution is a method to obtain the result of two signals, which are multiplied in frequency domain. The underlying rule is that any signal consists of a sum of infinite impulses and since every system has a characteristic impulse response, the output can be obtained from the summation of all these impulse responses at time t_o. Furthermore, a step is an integrated impulse and therefore, the impulse response can be obtained by differentiation. For any given input the output can be calculated as:

$$y(k) = \frac{1}{a}\sum_{i=0}^{N} x(k-i)[y_S(i) - y_S(i-1)] \tag{6.4}$$

where $y(k)$ is the output at the discrete time index k, $x(k)$ is the input and $y_s(k)$ the step response to a given step of amplitude a. (6.4) is a discretised version of the continuous convolution integral, in which $y_s(t)$ is differentiated and in addition, it is limited to N points. The larger N and the sampling period the better the approximation.

In the paper (Li *et al.*, 1996a), it has been shown that convolution provides a high fidelity model of the plant around some operating point. Furthermore, it has been successfully applied to generic linear controllers. In Li *et al.*'s work (1996a), a generic nonlinear

controller is evolved, which is to be independent from any specific plant model. This strategy is used to calculate the plant output in this chapter as well.

6.4 Applications

6.4.1 Regulation of Ship Heading

The problem of manoeuvring a ship is challenging and of considerable interest because of the complexity in obtaining an accurate dynamic model. Various external forces such as wave motion and wind effects, allied with the coupled behaviour of the navigation, steering and auto pilot systems, make the control task very difficult. In this example, the only point of interest is the design of a controller for regulating a cargo ship heading toward at a desired angle. A fuller description of the problem is given in the paper (Åström and Källström, 1976). It is also listed as IFAC benchmark problem number 89-08.

For straight-line motion the model of the ship under constant velocity is described as

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \tag{6.5}$$

$$\mathbf{y} = \mathbf{c}\mathbf{x} \tag{6.6}$$

where $\mathbf{x} \in R^3$, $\mathbf{u} \in R^1$, $\mathbf{y} \in R^1$ are given as follows:

u = rudder angle

y = heading angle of ship

x_1 = sway velocity of ship

x_2 = turning yaw rate

x_3 = heading angle of ship

and the structure of A, B and C is given by

$$\mathbf{A} = \begin{pmatrix} -0.895 & -0.286 & 0 \\ -4.367 & -0.918 & 0 \\ 0 & 1 & 0 \end{pmatrix} \quad \mathbf{B} = \begin{pmatrix} 0.108 \\ -0.918 \\ 0 \end{pmatrix} \quad \mathbf{C} = \begin{pmatrix} 0 & 0 & 1 \end{pmatrix}$$

The objective is to find a controller of the system to control and regulate the heading angle of the ship to a desired angle, such that the following constraints are satisfied:

1. No overshoot occurs in the output response of y.
2. The rudder motion is constrained:

$$|u| \leq 40^\circ \tag{6.7}$$

3. The rate of rudder motion is constrained:

$$\left|\dot{u}\right| \leq 10^\circ/s \tag{6.8}$$

First, it can be observed that this system is an unstable system. Its transfer function is

$$G(s) = \frac{-0.918s - 1.2932}{s^3 + 1.813s^2 - 4.274s} \tag{6.9}$$

From (6.9), the poles are

$$p_1 = 0.211, \quad p_2 = 0 \quad \text{and} \quad p_3 = -2.02 \tag{6.10}$$

Then it should be pointed out here that there is integration action in its transfer function, so no control action is necessary, when the system approaches the steady state. Physically, it is very understandable that when the heading direction is correct, the rudder should not be moved. So a PD based neurocontroller is designed for this ship heading regulation problem. Another characteristic in this problem is that there are strict limits not only for the controller action but also for the change of the controller action. Thus, the neurocontroller in the problem is like,

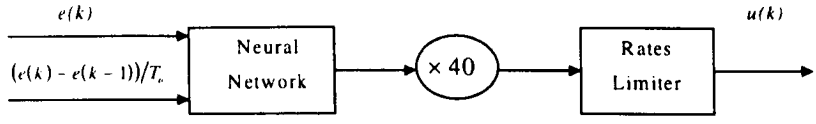

Figure 6.6 Neurocontroller designed for ship regulation

Output of the neural network is times by 40^0, which is maximum output of the controller specified in the problem. Considering rates limit $10^0/s$, the rate limiter could mathematically described as

$$u(k) = \begin{cases} u(k) & \text{if } |u(k) - u(k-1)| < 10T_0 \\ u(k-1) + 10T_0 & \text{if } u(k) - u(k-1) > 10T_0 \\ u(k-1) - 10T_0 & \text{if } u(k) - u(k-1) < -10T_0 \end{cases} \tag{6.11}$$

Since there is the requirement of no overshoot of the output, a special cost function to optimise the neurocontroller is designed for the problem,

$$J = m\int\left(|e| + k\left|\dot{e}\right|\right)tdt \qquad \text{if } e < 0, \quad m = 2 \tag{6.12}$$

Here k = 2. The condition present here is a penalty factor for overshoot. The reason behind this cost function is that the simple IATE plus the penalty of candidate controllers with overshoots fails to produce good results. It can not eliminate overshoots, and the results are the same as IATE without such penalty a factor. It seems like all slow processes have been cleansed in the early generation of EA optimisation. So the derivative part is attached in the cost function to keep the relatively slow processes to survive.

In order to control turning degree up to 90^0, the neurocontroller is trained at five different operating points. The sample time is 0.05s, and the simulation time is 50s. At 25s, a step noise imposed on the output. Figure 6.7 shows performances resulting from the trained neurocontroller. Figure 6.8 shows the neurocontroller achieved.

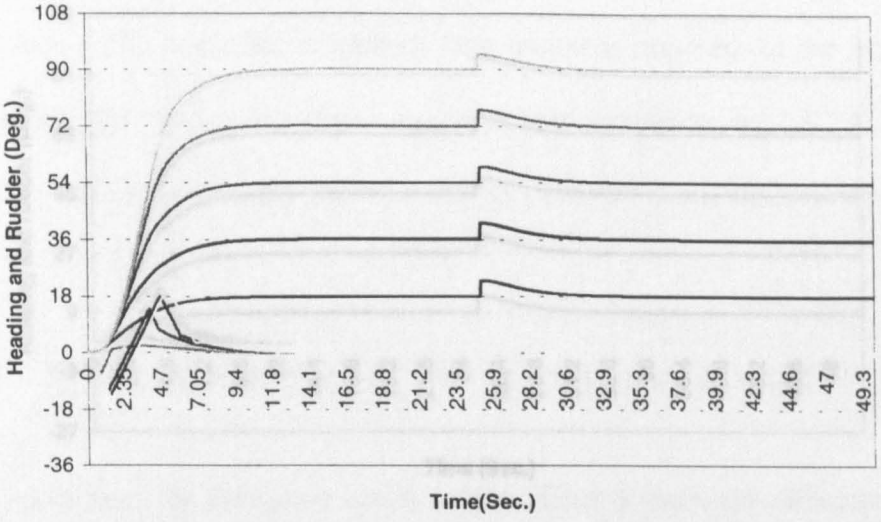

Figure 6.7 Performances of the ship regulating neurocontroller at known operating points

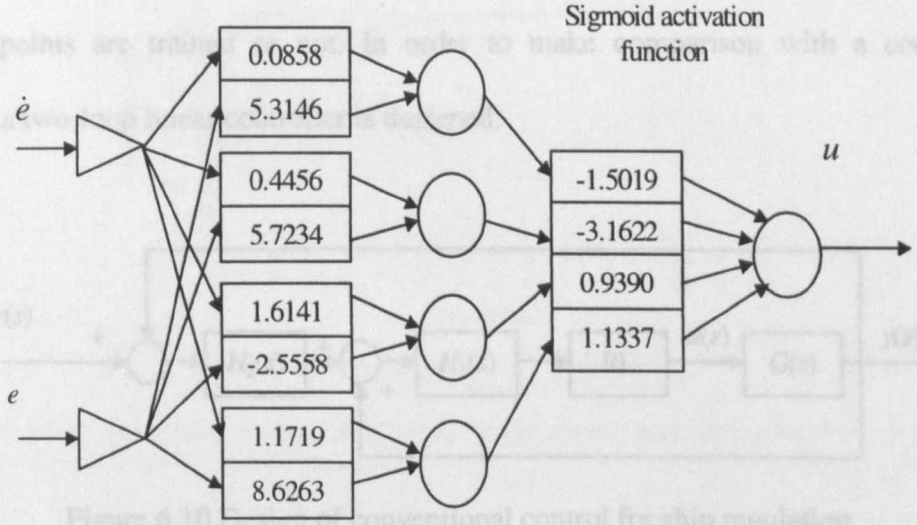

Figure 6.8 Neurocontroller achieved for ship regulation problem

In order to confirm the generality of the neurocontroller, performances of system on another five operating points, at which the neurocontroller is not trained, are tested here.

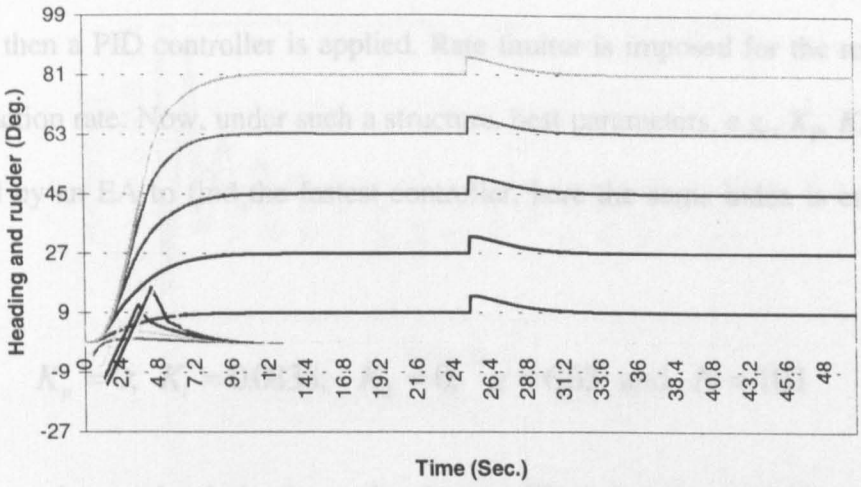

Figure 6.9 Performances of the ship regulating neurocontroller at different operating points

From Figures 6.7 and 6.9, the results of heading regulation are very good, no matter whether operating points are trained or not. In order to make comparison with a conventional controller, a two-loop linear controller is designed.

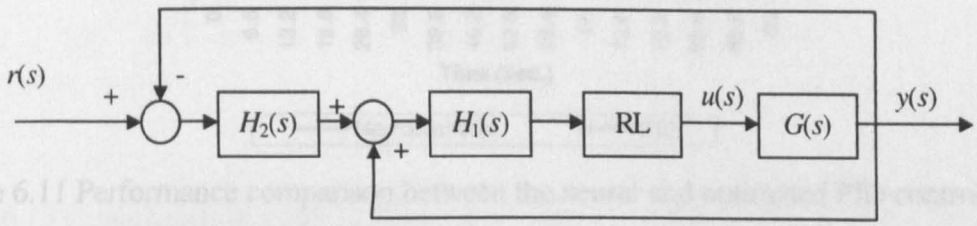

Figure 6.10 Design of conventional control for ship regulation

In Figure 6.10, $G(s)$ is (6.9). RL is a rate limiter, similar to (6.10). Others are given as follows

$$H_1(s) = g\frac{s+2}{s+10} \tag{6.13}$$

$$H_2(s) = K_p + \frac{K_i}{s} + \frac{K_d s}{1+\frac{1}{N}s} \tag{6.14}$$

Because the system is unstable, positive feedback compensation (6.13) is used to stabilise the system, and then a PID controller is applied. Rate limiter is imposed for the requirement of the control action rate. Now, under such a structure, best parameters, e.g., K_p, K_i, K_d, g and N are searched by an EA to find the fastest controller, here the same index is employed. The results are

$$K_p = 0;\ K_i = 0.0838;\ K_d = 0;\ g = 6.63 \text{ and } N = 10.1 \tag{6.15}$$

Since K_i is equal zero, the derivative action is zero. Time domain simulations are shown in Figure 6.11 to 6.13.

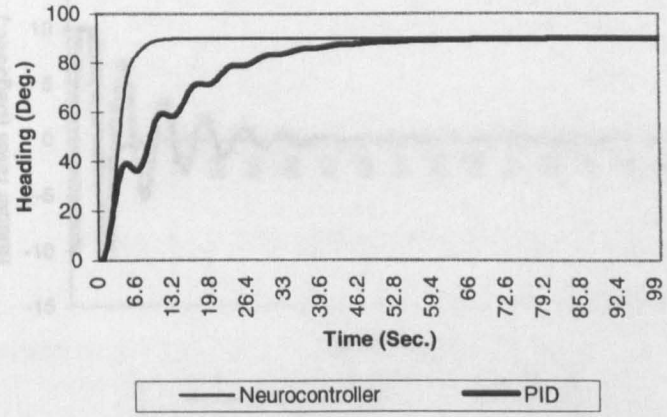

Figure 6.11 Performance comparison between the neural and optimised PID controllers

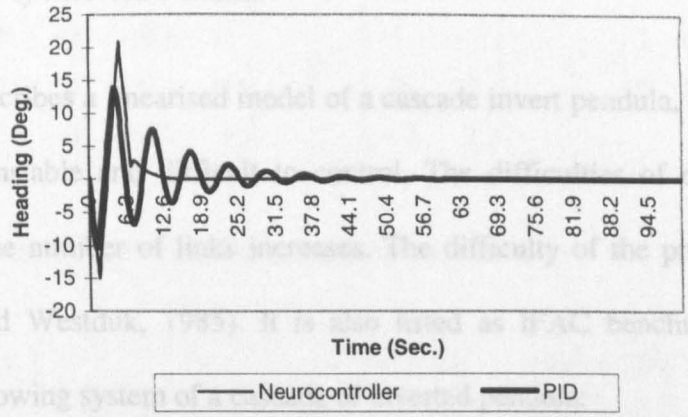

Figure 6.12 Comparison of rudder actions between the neural and optimised PID controllers

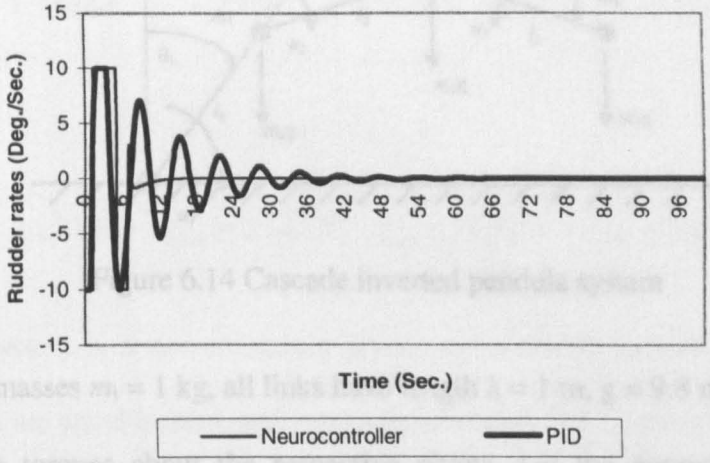

Figure 6.13 Comparison of rudder rates between the neural and optimised PID controllers

From the simulations, performance resulting from the neurocontroller is better than that of conventional PID controller apparently. Limits of rudder rate make the control more difficult, and this is the reason why there is the need to design a neurocontoller. It is expected that adding a proper prefilter can lead to a better result than the simple PID control. However, if the more improvement is needed, the prefilter could be very complex (Fossen, 1994 and Simensen, *et al.*, 1995). And the ideal filter just can operate in a specific operating point to get the optimum results.

6.4.2 *Cascade of Inverted Pendula*

This problem describes a linearised model of a cascade invert pendula, and has the feature of being 'highly' unstable and difficult to control. The difficulties of control become more pronounced as the number of links increases. The difficulty of the problem is analysed by (Kwakernaak and Westduk, 1985). It is also listed as IFAC benchmark problem 89-01. Consider the following system of a cascade of inverted pendula:

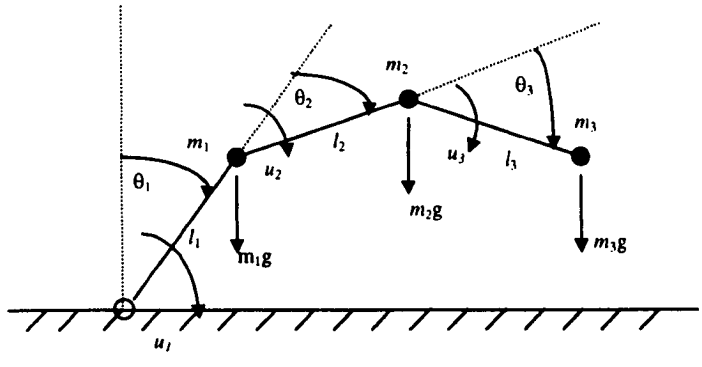

Figure 6.14 Cascade inverted pendula system

Where all point masses m_i = 1 kg, all links have length l_i = 1 m, g = 9.8 m/sec^2, and where u_1, u_2, u_3,... denote torques about the respective pivots. Let the outputs of the system be $y_i = \theta_i$, $i = 1, 2, 3...$ and the inputs to the system be u_i, i = 1, 2, 3....

It is desired to design a controller to stabilise the system so that the outputs are regulated to zero in the presence of unmeasured constant disturbances, which may be applied to the system.

In our research, a two links problem is solved. The linearised model is

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{Ax} + \mathbf{Bu} \\ \mathbf{y} &= \mathbf{Cx} \end{aligned} \tag{6.16}$$

where

$$\mathbf{A}=\begin{pmatrix} 0 & 1.0 & 0 & 0 \\ 9.8 & 0 & -9.8 & 0 \\ 0 & 0 & 0 & 1.0 \\ -9.8 & 0 & 29.4 & 0 \end{pmatrix} \quad \mathbf{B}=\begin{pmatrix} 0 & 0 \\ 1 & -2 \\ 0 & 0 \\ -2 & 5 \end{pmatrix} \quad \mathbf{C}=\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{6.17}$$

Translating it into the transfer function in frequency domain

$$\begin{pmatrix} y_1 \\ y_2 \end{pmatrix}=\frac{1}{s^4-39.2s^2+192.08}\begin{pmatrix} s^2-9.8 & -2s^2+9.8 \\ -2s^2+9.8 & 5s^2-29.4 \end{pmatrix}\begin{pmatrix} u_1 \\ u_2 \end{pmatrix} \tag{6.18}$$

The poles are

$$p_1=5.7844 \quad p_2=2.396 \quad p_3=-2.396 \quad \text{and} \quad p_4=-5.7844 \tag{6.19}$$

The zeros are

$$z_{1,2}=\pm 3.1305 \quad z_{3,4}=\pm 2.2136 \quad z_{5,6}=\pm 2.2136 \quad z_{7,8}=2.4249 \tag{6.20}$$

Therefore, the system is a non-minimum phase and unstable system. Since if the initial values of x_1 to x_4 are equal to zero, and there are not inputs, the system can maintain its state, A PD type neurocontroller is designed to regulate the system. Because the model is linearised, the maximum x_1 and x_3 to be regulated are supposed as 0.1 rad. Because the x_1 and x_3 could be positive and negative, so the neurocontroller is trained to deal with four initial states:

Case 1: $\left(x_1, \dot{x}_1, x_2, \dot{x}_2\right)$ = (0.1 0 0.1 0)

Case 2: $\left(x_1, \dot{x}_1, x_2, \dot{x}_2\right)$ = (0.1 0 −0.1 0)

Case 3: $\left(x_1, \dot{x}_1, x_2, \dot{x}_2\right)$ = (0.1 0 0 0)

Case 4: $\left(x_1, \dot{x}_1, x_2, \dot{x}_2\right)$ = (0 0 0.1 0)

Just because in our design, the neurocontroller is an odd function, the following four cases could be dealt with automatically,

$$\text{Case 5: } \left(x_1, \dot{x}_1, x_2, \dot{x}_2\right) = (-0.1,\ 0,\ -0.1,\ 0)$$

$$\text{Case 6: } \left(x_1, \dot{x}_1, x_2, \dot{x}_2\right) = (-0.1,\ 0,\ 0.1,\ 0)$$

$$\text{Case 6: } \left(x_1, \dot{x}_1, x_2, \dot{x}_2\right) = (-0.1,\ 0,\ 0,\ 0)$$

$$\text{Case 8: } \left(x_1, \dot{x}_1, x_2, \dot{x}_2\right) = (0,\ 0,\ -0.1,\ 0)$$

In order to compare the neurocontroller with conventional controllers, linear quadratic control law (Anderson and Moore, 1989) is employed as the index for neurocontroller, and maximum output of controller is supposed to 10 N.

$$J = \int \left(\mathbf{x}'\mathbf{Q}\mathbf{x} + \mathbf{u}'\mathbf{R}\mathbf{u}\right)dt \tag{6.21}$$

Where

$$\mathbf{Q} = \begin{pmatrix} 10 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad \mathbf{R} = \begin{pmatrix} 0.01 & 0 \\ 0 & 0.01 \end{pmatrix} \tag{6.22}$$

In (6.22), much weight is put on x_1 and x_3, because the values of forces are very big. The neurocontroller achieved through the evolutionary training method is in Figure 6.15.

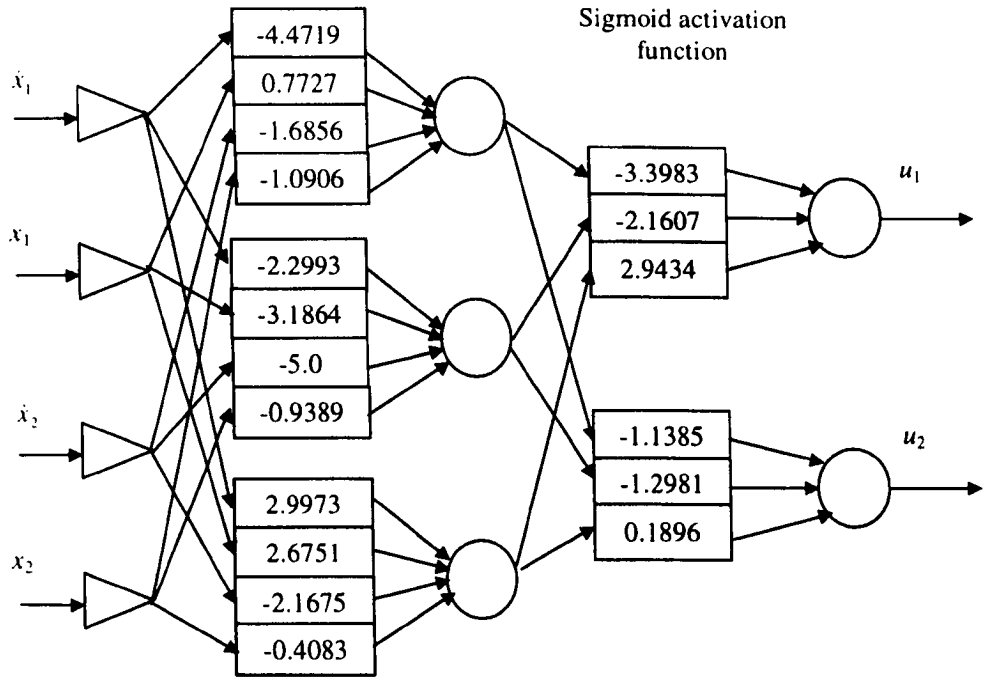

Figure 6.15 Neurocontroller for the inverted pendula with output limit to 10 N

Through solving corresponding Riccati equation, the linear controller achieved is given in (6.23). A comparison between the neurocontrol and LQR control is given in the Figures 6.16 to 6.23.

$$\mathbf{k} = \begin{pmatrix} 73.403 & 28.906 & 14.714 & 7.570 \\ 14.714 & 7.570 & 43.975 & 13.766 \end{pmatrix} \tag{6.23}$$

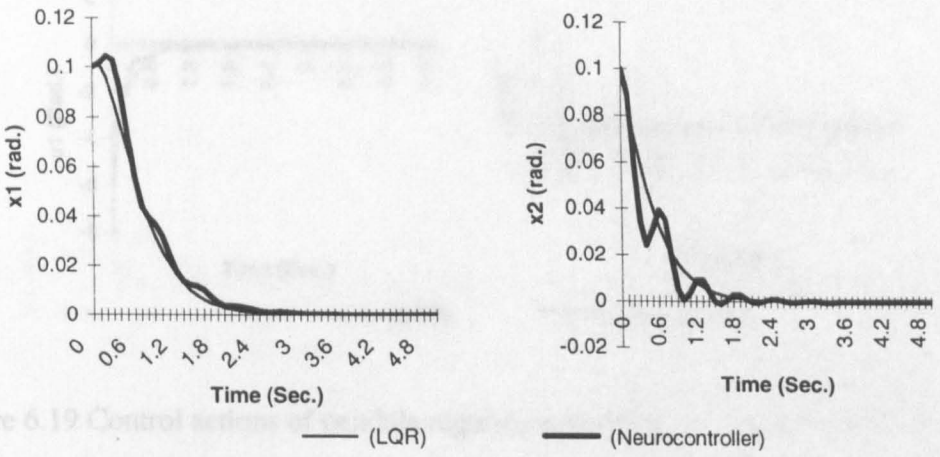

Figure 6.16 Regulations of pendula with $\left(x_1, \dot{x_1}, x_2, \dot{x_2}\right) = (0.1\ 0\ 0.1\ 0)$

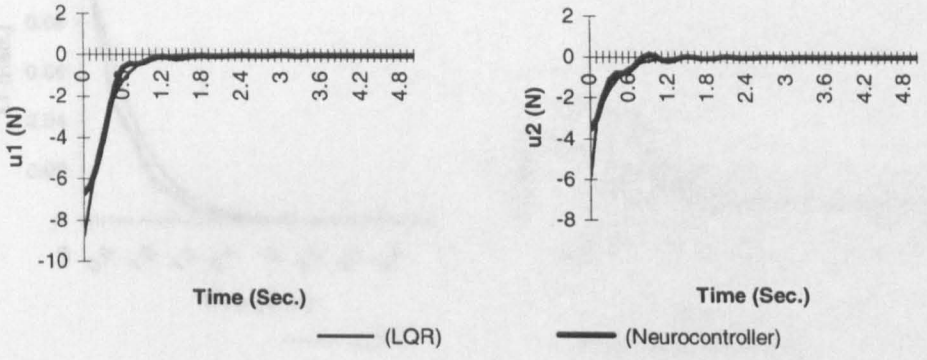

Figure 6.17 Control actions of pendula regulation with $\left(x_1, \dot{x_1}, x_2, \dot{x_2}\right) = (0.1\ 0\ 0.1\ 0)$

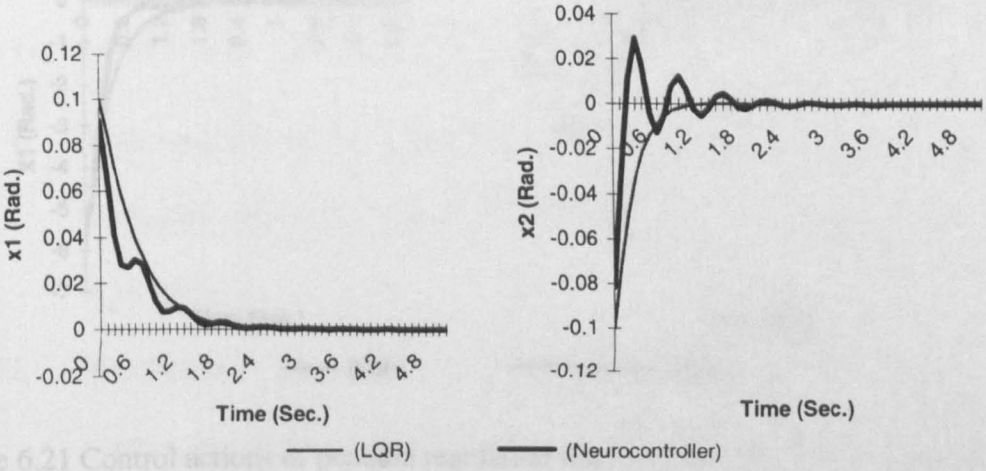

Figure 6.18 Regulations of pendula with $\left(x_1, \dot{x_1}, x_2, \dot{x_2}\right) = (0.1\ 0\ -0.1\ 0)$

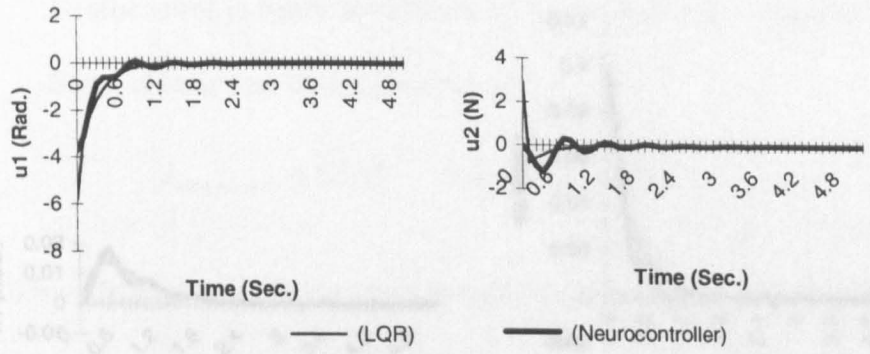

Figure 6.19 Control actions of pendula regulation with $\left(x_1, \dot{x_1}, x_2, \dot{x_2}\right) = (0.1\ 0\ -0.1\ 0)$

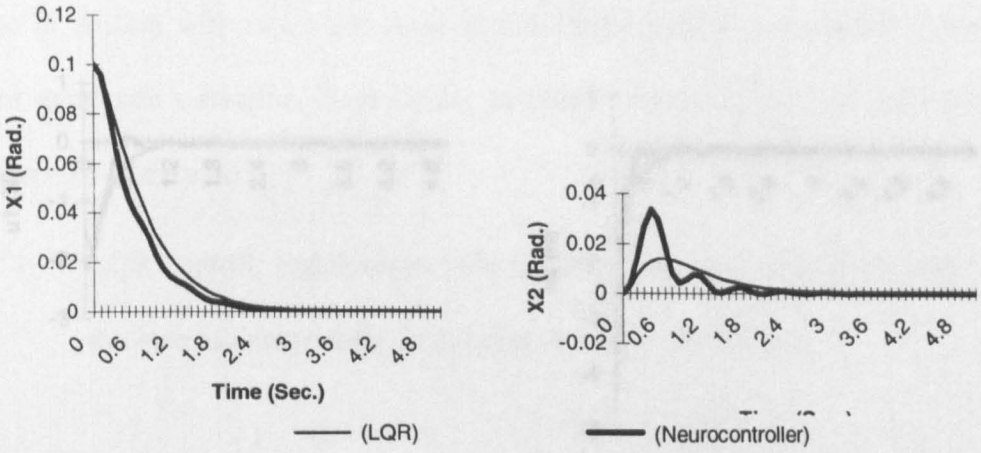

Figure 6.20 Regulations of pendula with $\left(x_1, \dot{x_1}, x_2, \dot{x_2}\right) = (0.1\ 0\ 0\ 0)$

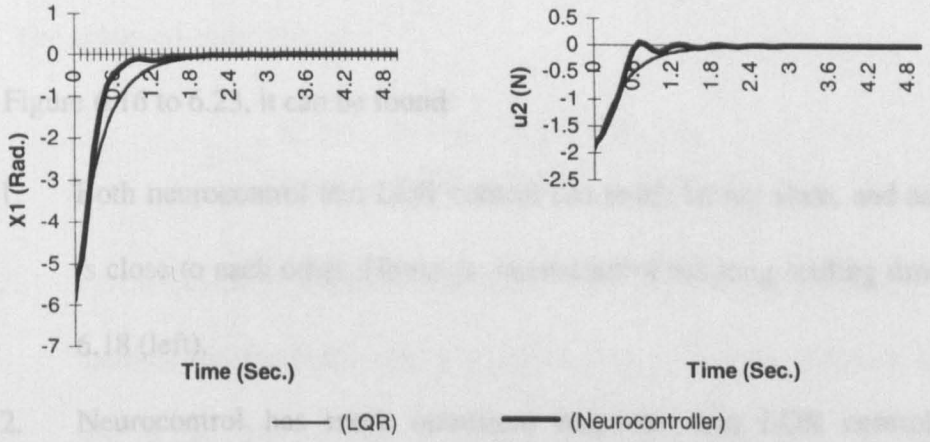

Figure 6.21 Control actions of pendula regulation with $\left(x_1, \dot{x_1}, x_2, \dot{x_2}\right) = (0.1\ 0\ 0\ 0)$

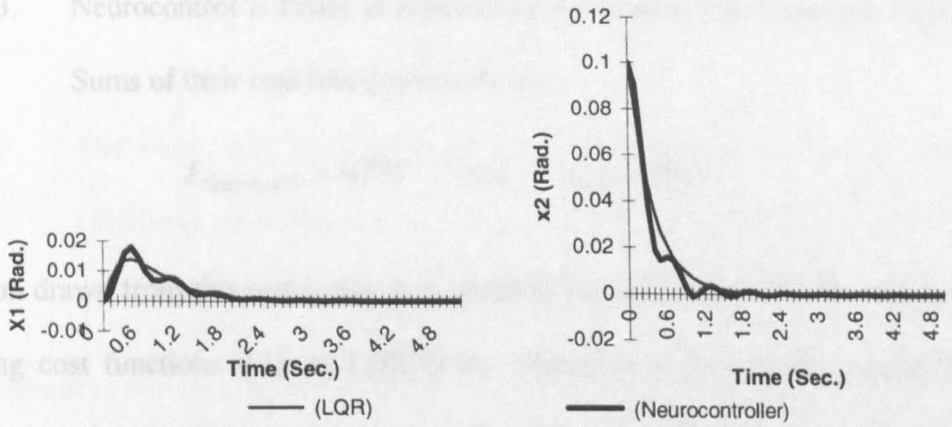

Figure 6.22 Regulation of pendula with $\left(x_1, \dot{x_1}, x_2, \dot{x_2}\right) = (0\ 0\ 0.1\ 0)$

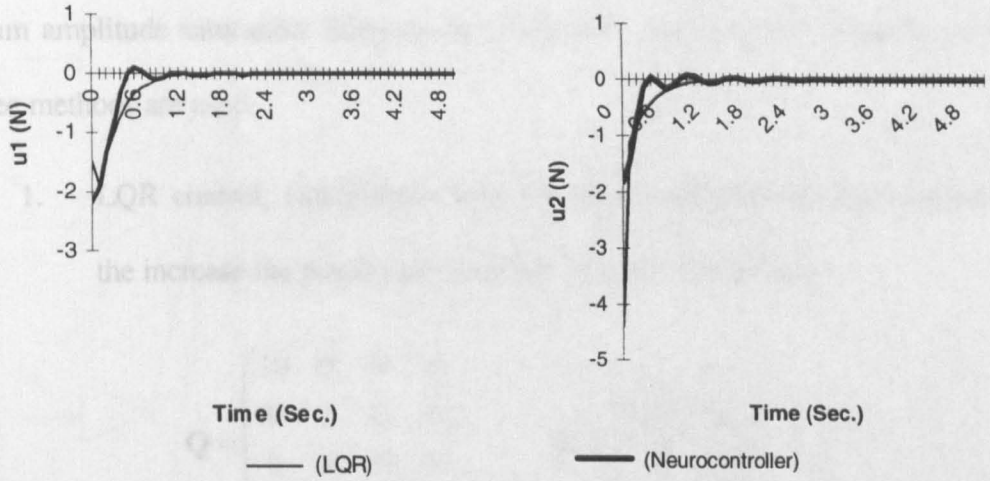

Figure 6.23 Control actions of pendula regulation with $\left(x_1, \dot{x_1}, x_2, \dot{x_2}\right) = (0\ 0\ 0.1\ 0)$

From the Figure 6.16 to 6.23, it can be found

1. Both neurocontrol and LQR control can reach steady state, and settling time is close to each other. However, neurocontrol has long settling time in Figure 6.18 (left).
2. Neurocontrol has more otransient response than LQR control. In LQR control, the response at most has one peak. In neurocontrol, the response may have many peaks.

3. Neurocontrol is better at minimising the cost function than the LQR control. Sums of their cost function results are:

$$J_{\text{Neurocontrol}} = 0.535 \quad \text{and} \quad J_{\text{LQR}} = 0.621 \tag{6.24}$$

Conclusion drawn from this test is that it is possible that the neurocontroller can be better at minimising cost functions than an LQR linear controller at the specific operating points. However, there is not apparent improvement in terms of performance in this linear model control example. Since the ship regulation example has shown that neurocontroller has the advantage in dealing with rate limit, there is a discussion about a controller under simple maximum amplitude saturation. Suppose the maximum output of the controller to be 5 N. The three methods are used:

1. LQR control, which deals with the requirement of maximum amplitude by the increase the penalty on the control action. In this case,

$$\mathbf{Q} = \begin{pmatrix} 10 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad \mathbf{R} = \begin{pmatrix} 100 & 0 \\ 0 & 100 \end{pmatrix}. \tag{6.25}$$

The achieved controller is:

$$\mathbf{k} = \begin{pmatrix} 58.803 & 24.131 & 19.598 & 9.172 \\ 19.698 & 9.172 & 19.608 & 5.787 \end{pmatrix} \tag{6.26}$$

2. Linear controller under the saturation limits optimised by an EA. Under this method, the feedback matrix is decided by minimum ITAE. In this case, it just is the sum of two outputs' ITAE, since saturation is considered in simulation. Time domain simulation considering the controllers is carried out by numerical methods, e.g. Runge-Kutta. The controller achieved as

$$\mathbf{k} = \begin{pmatrix} 59.963 & 20.965 & 18.236 & 12.016 \\ 18.824 & 7.358 & 16.697 & 6.384 \end{pmatrix} \tag{6.27}$$

3. The neurocontroller is designed the same as before, except making the controller maximum be 5 N. The neurocontroller achieved through evolutionary training is given in Figure 6.24.

The performances and control actions of the controllers are given respectively in the Figure 6.25 and Figure 6.26.

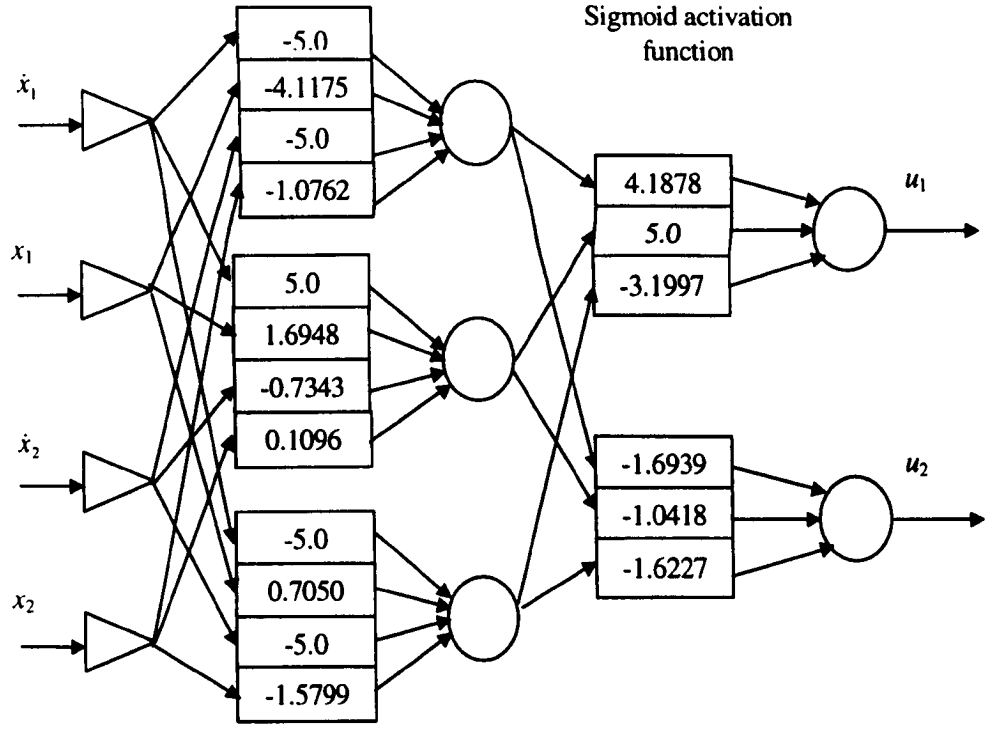

Figure 6.24 Neurocontroller for the inverted pendula with output limit to 5 N

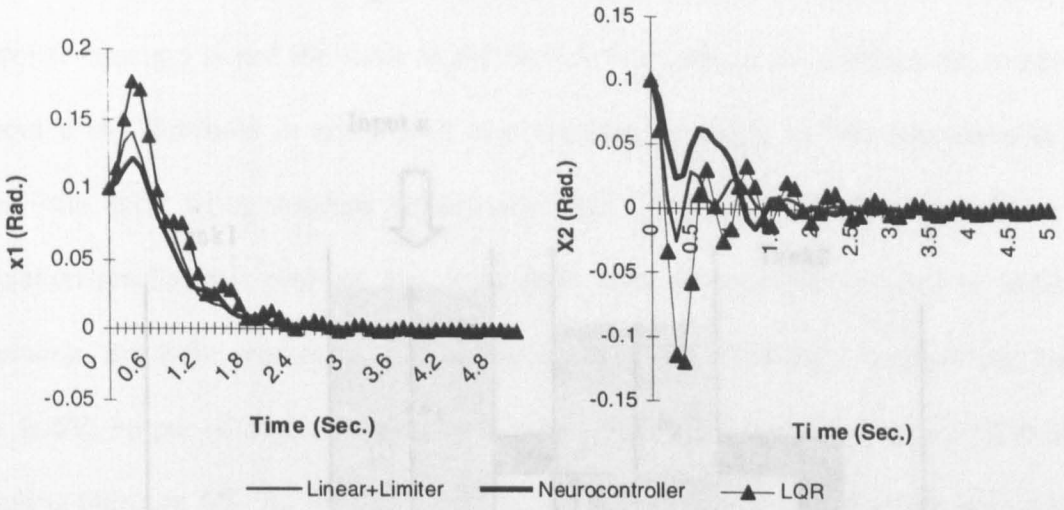

Figure 6.25 Performance results with amplitude saturation of different methods

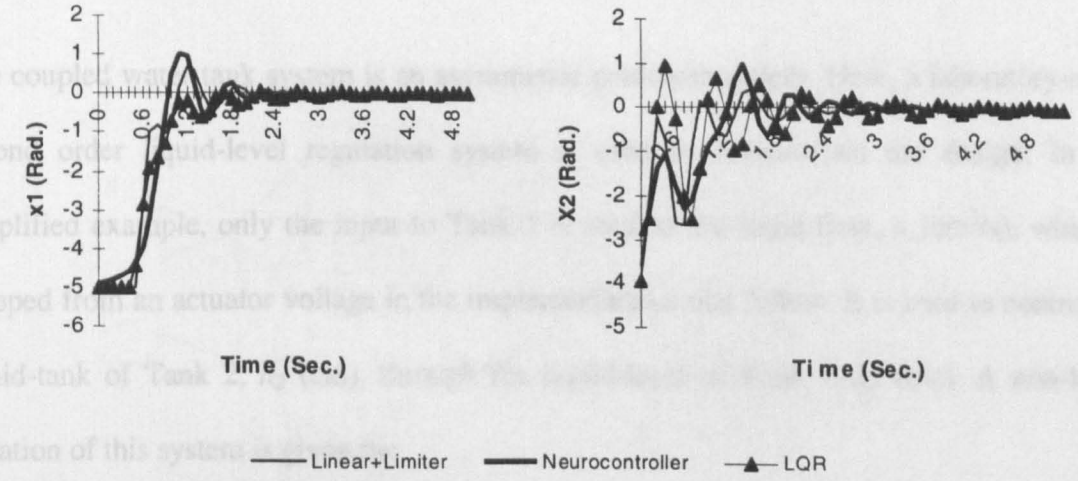

Figure 6.26 Control action results with amplitude saturation of different methods

From these simulation results, it could be found that the results from LQR are not as good as the results from the linear controller trained under limits and neurocontroller. However, it seems that neurocontroller is not better than the linear controller trained under limits.

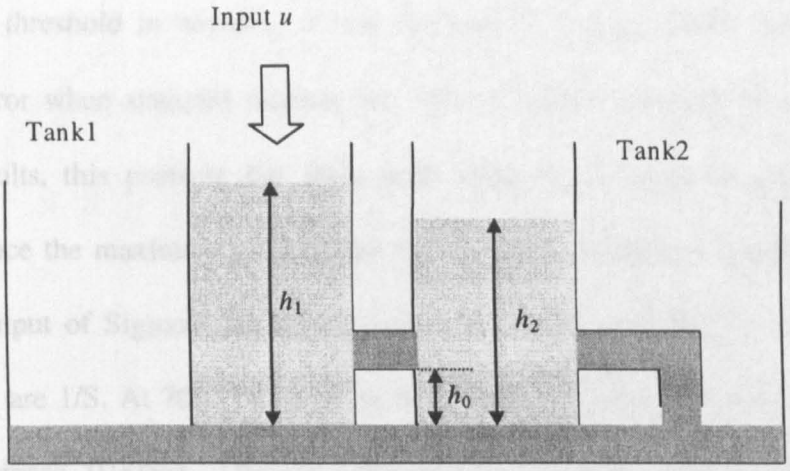

Figure 6.27 Laboratory-scale water tank demonstration system

The coupled water tank system is an asymmetric nonlinear system. Here, a laboratory-scale, second order liquid-level regulation system is used to demonstrate the design. In this simplified example, only the input to Tank 1 is used as the input flow, u (cm^3/s), which is mapped from an actuator voltage in the implementations that follow. It is used to control the liquid-tank of Tank 2, h_2 (cm), through the liquid-level of Tank 1, h_1 (cm). A non-linear equation of this system is given by:

$$\left.\begin{aligned} A\dot{h}_1 &= u - a_1 c_1 \sqrt{2g(h_1 - h_2)} \\ A\dot{h}_2 &= a_1 c_1 \sqrt{2g(h_1 - h_2)} - a_2 c_2 \sqrt{2g(h_2 - h_0)} \end{aligned}\right\} \tag{6.28}$$

where A = 100 cm^2, the cross-section area of both tanks; a_1 = 0.396 cm^2 and a_2 = 0.385cm^2, the orifice areas of Tank 1 and Tank 2, respectively; $c_1 = c_2 = 0.58$, the discharge constants; h_0 = 3 cm, the height of the orifice and of the coupling path; and g = 9.81cm/s^2, the acceleration due to gravity. At rest, there was no input for a long time. The initial conditions of h_1 and h_2 are thus the same as h_0.

Considering that control action is still needed, when the tank level approach the steady state, controller structure is just the same as Figure 6.3, is employed to approach the steady state without using threshold in neurons. It was reported by Brune (1998) that there is small steady-state error when standard neurons are used to control the tank system. From these simulation results, this problem has been dealt with. Convolution is used to realise the simulation. Since the maximum voltage that could be put to motor to pump water into the tank is 5V, output of Sigmoid function is times by 5. The sampling time is 1000 S. The sampling rates are 1/S. At 700^{TH} S, suppose that some object is put in the water tank force the water increase 0.005M. Through evolutionary training suggested early, the final neurocontroller achieved is given in Figure 6.28.

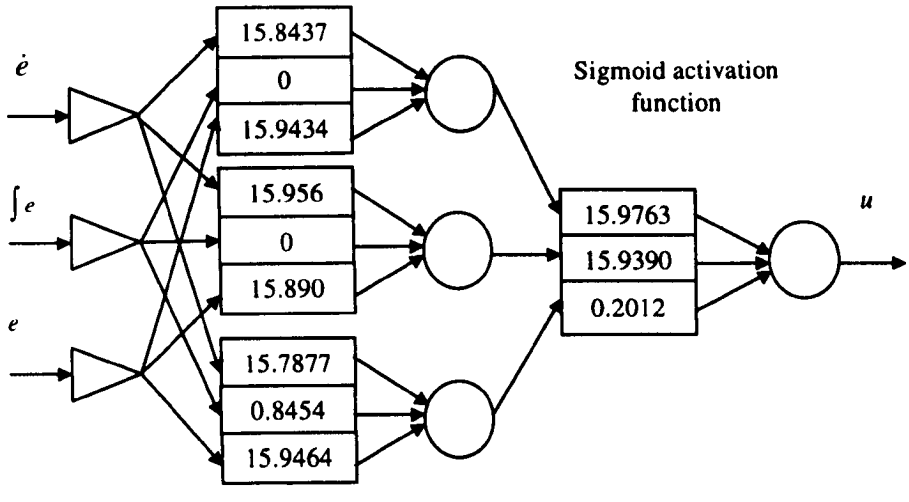

Figure 6.28 Neurocontroller for coupled water tank system

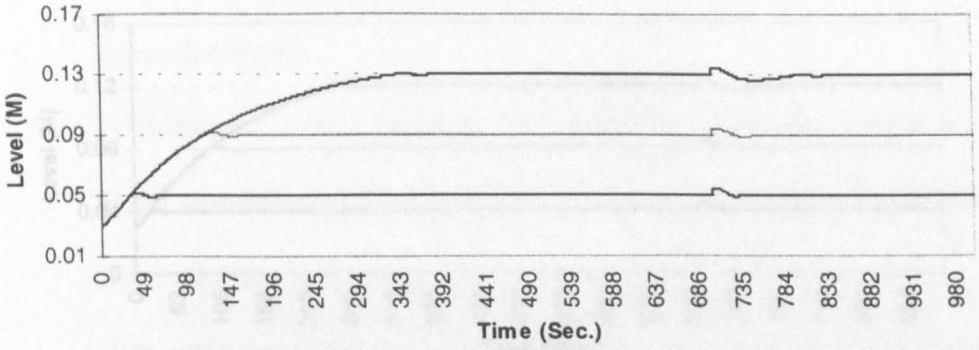

Figure 6.29 Control of water tank system by the neurocontroller at trained operating points

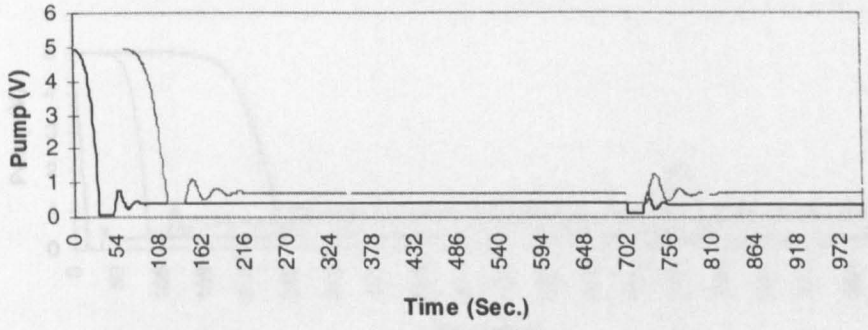

Figure 6.30 Control action of the neurocontroller at trained operating points

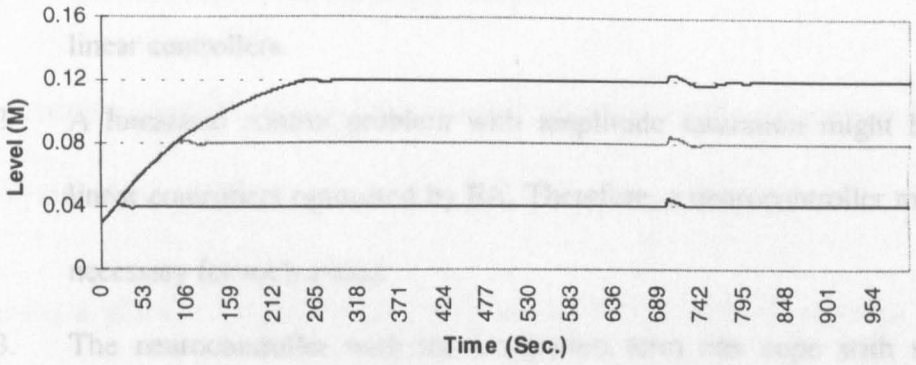

Figure 6.31 Performances of the neurocontroller at untrained operating points

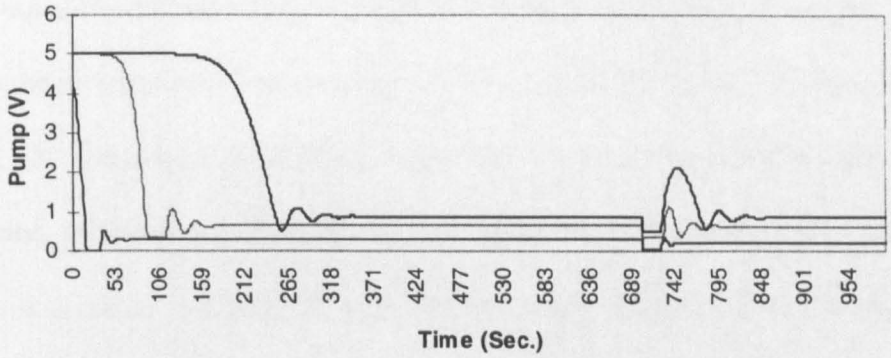

Figure 6.32 Control action of the neurocontroller at untrained operating points

6.5 Summary

In this chapter, the basic PID control structure is extended to nonlinear control architecture in form of neural networks. Special type neurons are studied in the neurocontrollers. EA training for optimisation of the structure and weights of the neurocontroller is proposed. Growth training method is used to optimise the neurocontroller structure and lead to the simplest. In the meantime, the neurocontroller is designed directly from plant step response data without a model, where convolution is used. Through the ship regulation, inverted pendula regulation and twin tank problems, it is found:

1. The neurocontroller has advantages in dealing with rate limits, compared with linear controllers.
2. A linearised control problem with amplitude saturation might be dealt by linear controllers optimised by EA. Therefore, a neurocontroller might not be necessary for such a case.
3. The neurocontroller with the integration term can cope with steady-state errors at different operating points.

Chapter 7 Conclusion and Further Work

7.1 Structuring and Global Optimisation Evolutionary Environment for Control Systems

In this thesis, a globally optimal evolutionary methodology and environment for control system structuring and design automation has been developed, which requires no design indices to be differentiable. So the indices can be designed to satisfy control requirements without being differentiable. This is based on a hybridised GA search engine, whose local tuning has been enhanced tremendously by the incorporation of HC, SA and Simplex techniques. A Lamarkian inheritance technique has also been developed to improve crossover and mutation operations in GAs. Benchmark tests have shown that this novel hybrid GA is accurate and reliable. Based on this search engine and optimisation core, the linear and nonlinear control system design automation suite has been developed in a Java based platform-independent format, which is readily available for design and design collaboration over corporate Intranets and the Internet.

To apply automation suite to control design, the indices to describe performance and robust stability have been investigated for practical control system designs. A thorough study on the merits and deficiencies of existing optimal control indices has been carried out, and ITAE is found to be the most acceptable one. Based on the findings, new indices are proposed, which can approach different damping ratios. Hybridised indices combining time and frequency domain measurement and accommodating practical constraints have been developed and applied to extending optimal control beyond its current capabilities. After finishing the development of this optimisation environment and investigation of design indices, the environment is used to optimise different type controllers: PID, QFT and neural network.

7.2 Application to PID Control System Design Automation and Batch Optimisation

PID controllers are very popular in industrial control. Although many PID tuning rules that have been developed during the past half century. PID controllers are not always tuned properly in many applications. Because set point following has been solved by Zhuang and Atherton (1993), Majhi and Atherton (1999); Atherton (2000), it has been chosen to concentrate on load disturbance rejection. The load disturbance problem has now been addressed using the EA based design automation tool. In particular, systematic and batch optimisation of PID controllers to meet practical requirements has been achieved. A novel cost function has been designed to take disturbance rejection, stability in terms of gain and phase margins and other specifications into account in the same time. The results has shown that

1. The derivative action plays an important role in improving load disturbance rejection while maintaining or improving stability margins;
2. Compared with Åström's (Levine, 1996), Ho's (1995) and Zhuang and Atherton's methods, the performance achieved in this thesis is much better not only in load disturbance rejection but in stability margins (See Section 4.4.2, Section 4.4.3 and Section 4.4.4).

7.3 Extension to GA Optimisation to Loop Shaping in QFT Design

The robustness issues experienced in PID control are addressed by extending the PID structure to a free form transfer function. This is realised in the form of QFT control. Loop shaping is the most challenging part in designing QFT controllers. For this, optimisation and 'automatic design' techniques have recently been investigated and developed to unleash the

power of QFT. The major disadvantage of existing approaches is, however, their inability in globally solving the optimisation problem of QFT design, which is often multi-modal and multi-dimensional. These analytical and convex or linear programming based techniques often impose unrealistic or unpractical assumptions and have often lead to very conservative designs.

In this thesis, a novel index that takes advantages of the EA based design automation tool has been developed to include stability, load disturbance rejection and reduction of 'high frequency gain'. This has not been achieved using existing methods. A corresponding prefilter can also be systematically designed if tracking is one of the specifications. The design results have shown that

(i) Controllers achieved by the design automation suite can offer a lower order and a lower 'high frequency gain' than results published elsewhere to date (Chait *et al.*, 1999; Borghesani *et al.*, 1995).

(ii) The designed controller combined with the corresponding prefilter performs satisfactorily in time domain (Figure 5.13).

7.4 Extension to Enabling Forward Path Neural Control

The 'three term' linear PID controller is not the best controller for robust or nonlinear applications. For nonlinear plants, the PID structuring and design strategy has been extended to a nonlinear format. This is implemented as a building block based on neural network. The automation design environment has been employed to design and optimise neurocontrollers. The design results have shown that

(i) The design suite make direct training of feedforward neurocontroller possible;

(ii) The neurocontroller can be designed directly from plant step response data without a model, where convolution is used;

(iii) Growth training method can optimise the structure and lead to the simplest structure;

(iv) If a rate limiter is required in a practical control loop, the designed neurocontroller outperforms an optimised linear controller;

(v) The neurocontroller can cope with amplitude limit better than the linear controllers achieved from LQR;

(vi) Steady-state errors at different operating points of a nonlinear double tank system can be cancelled by one trained neurocontroller.

7.5 Future Perspective

7.5.1 Evolutionary Environment

A user-friendly interface can been added to the evolutionary environment developed in this thesis. So users can flexibly change the parameters in the evolutionary process, e.g., perturbation range of mutation, number of generation to stop, etc. To evaluate the efficiency of the evolutionary environment further, other benchmark problems should be tried, e.g., multi-objective problem to be tested.

7.5.2 PID Control Design

In this thesis, design automation has been applied to batch tuning of PID controllers for first order plus dead time (FOPDT) plants. Although many systems are modelled this way in control engineering practice, more accurate models exist. Ho *et al.* (1995) used a second order plus dead time (SOPDT) to derive tuning rules. But zero-pole cancellation is used to transform an SOPDT into an FOPDT. It is demonstrated feasible for the PI tuning rules with gain and phase margins specification to be used for an FOPDT. However, the cancellation is

not the best choice to achieve load disturbance rejection. So the suggestion for future work is to achieve better results for *SOPDT by using hybridised GA*.

7.5.3 QFT Control

As pointed out in Chapter 5, in order to improve evolution time, the subroutine can be used to evaluate the design can be programmed in Java. Alternative methods, such as H_∞, can be tried and compared with designs by QFT, because the results from QFT design are not good enough in terms of order of controller and high frequency.

Three cases of the hybridised GA application to QFT loop shaping are just the solution of single-input-single-output (SISO) problems. Multi-input-multi-output (MIMO) problems could be more challenging than SISO problems (Horowitz 1992), because there are n^2 scalar loop transmissions in $L = PG$ and n^2 uncertain (but generally correlated) plant functions. It is known that multi-loop is a better choice to achieve better results (Levine 1996). Because MIMO and multi-loop need more parameters of a controller to be optimised, the hybrid GA may help human designers even more in those applications. So further work suggested here is to deal with MIMO QFT loop shaping using hybridised GA.

7.5.4 Neurocontroller Design

Applications show that neurocontrollers may outperform optimised linear controllers in the case of linear plants with rate limit and for nonlinear plants. Further investigation may be carried out on plants with other hard nonlinearities, such as Coulomb friction, dead-zones, backlash, and hysteresis (Slotine and Li, 1991). In this thesis, the linear model plus saturation is used for the inverted pendulum problem in Section 6.4.2. It is just acceptable when offset is under 0.1 rad. In order to describe the situation with bigger offset a nonlinear model plus

saturation is more accurate. Since neural networks can approximate nonlinear systems, it can be tried to control the big offset in the nonlinear model plus saturation.

References

Anderson, B. D. O., and Moore, J. B. (1989). *Optimal Control-Linear Quadratic Methods*, Prentice-Hall International Editions, NJ, USA.

Åström, K. J. (1991). Assessment of achievable performance of simple feedback loops, International Journal of Adaptive Control and Signal Processing, Vol. 5, 13-19.

Åström, K. J., and Hägglund, T. (1995). *PID Controllers: Theory, Design and Tuning*, Instrument Society of America.

Åström, K. J., and Källström, C. G. (1976). Identification of ship steering dynamics. *Automatica*, Vol. 12, 9-22.

Åström, K. J., Hägglund, T., Hang, C. C., and Ho, W. K. (1993). Automatic tuning and adaptation for PID controllers-A survey, *Control Engineering Practice*, Vol. 1. No. 4, 699-714.

Atherton, D. P. (2000). Relay autotuning: a use of old ideas in a new setting, Transactions of the Institute of Measurement and Control, Vol. 22, No. 1, 103-122.

Baker, H. A. (1995). Open environments and object-oriented methods for computer-aided control system design, *Control Eng. Practice*, Vol. 3, No. 3, 347-356.

Ballance, D. J., and Gawthrop, P. J. (1991) Control systems design via a quantitative feedback theory approach, *Proceedings of the IEE conference "Control' 91"*, Heriot-Watt University, Edinburgh, U.K., Vol. 1, 476–480.

Ballance, D. J., and Gawthrop, P. J. (1991). Control systems design via a quantitative feedback theory approach. In Proceedings of the IEE conference "*Control' 91*", Vol. 1, Heriot-Watt University, Edinburgh, U.K., 476-480

Battiti, R., and Tecchiolli, G. (1993). *Local Search with Memory: Benchmarking RTS*, Technical Report, Universita di Trento, Gruppo Collegato di Trento, Italy.

Beccerra, V. M. (2000). Book reviews (*Autotuning of PID controllers: relay feedback approach*, Cheng-Ching Yu, Springer, Berlin, 1999, 226 pages), *Automatica*, Vol. 36, No. 11.

Borghesani, C., Chait, Y., and Yaniv, O. (1995). *Quantitative Feedback Theory Toolbox User Manual.* The Math Work Inc..

Brune, T., Chew, K. G., Feng, W., *et al.* (1998). Evolving Generic Feedforward Path Nonlinear Controllers from Open-loop Plant Response Data, Asia and Pacific Conference for Control and Measurement. Dunhuang, China, 217-222.

Bryant, G. F., and Halikias, G. D. (1995). Optimal loop-shaping for systems with large parameter uncertainty via linear programming, *Int. J. Control*, Vol. 62, No. 3, 557–568.

Burns, R. S. (1995). The use of artificial neural networks for the intelligent optimal guidance control of surface ships, IEEE Journal of Oceanic Engineering, Vol. 20, No. 1, 65-72.

Caponetto, R., Fortuna, L., Muscato, G., and Xibilia, M. G. (1994). Genetic algorithms for controller order reduction, *Proc. 1st IEEE Conf. Evolutionary Computation*, IEEE World Congress on Computational Intelligence, Orlando, FL, Vol. 2, 724-729.

Chait, Y. (1997). QFT loop shaping and minimisation of the high-frequency gain via convex optimisation, *Proceedings of the Symposium on Quantitative Feedback Theory and other Frequency Domain Methods and Applications*, Glasgow, Scotland, 13-28.

Chait, Y., Chen, Q., Hollot, C. V. (1999). Automatic loop-shaping of QFT controllers via linear programming. *Journal of Dynamic Systems, Measurement and Control*, Vol. 121, 351-357.

Chait, Y., Hollot, C. V., Mehta, P, and Zheng, Y. (1996). Active noise control in ducts using non-adaptive feedforward/feedback control, *Proceedings - National Conference on Noise Control Engineering*, Vol. 1, 379-384.

Chen, W., Ballance, D. J. (1999). Plant template generation for uncertain plants in Quantitative Feedback Theory, *Journal of Dynamic Systems, Measurement and Control*, Vol. 115, 51-59.

Chen, W., Ballance, D. J., and Li, Y. (1998). Automatic loop-shaping in QFT using genetic algorithms, *Proceedings of 3rd Asia-Pacific Conference on Control and Measurement*, 63–67.

Chen, W., Ballance, D., Feng, W., and Li, Y. (1999). Genetic Algorithm Enabled Computer-Automated Design of QFT Control Systems, *Proceeding of the IEEE International Symposium on Computer-Aided Control System Design*, Hawai'i, USA, 492-497.

Chipperfield, A. J., and Fleming, P. J. (1995). Gas turbine engine controller design using multi-objective genetic algorithms, *Proc. First IEE/IEEE Int. Conf. on GAs in Eng. Syst.: Innovations and Appl.*, Univ. of Sheffield, 214-219.

Chowdhury, M. (1999). *Evolutionary and reinforcement fuzzy control*, Ph.D. Thesis, Dept. of Electronics and Electrical Engineering, University of Glasgow.

Cluett, W. R., and Wang, L. (1991). Modelling and robust controller design using step response data, *J. Chemical Eng. Science*, Vol. 56, 2065-2077.

Dahlin, E. B. (1968). Designing and tuning digital controllers, *Instruments and Control Systems*, Vol. 42, 56-60.

Dakev, N. V., Whidborne, J. F., and Chipperfield, A. J. (1995). H_∞ design of an EMS control system for a maglev vehicle using evolutionary algorithms, *First Int. Conf. on GAs in Engi. Syst.: Innovations and Application*, Sheffield, UK, 226-231.

Dorf, R. C. (1992). *Modern Control Systems*, Addison-Wesley Publishing Company.

Feng, W., and Li, Y. (1999). Performance Indices in Evolutionary CACSD Automation with Application to Batch PID Generation, *Proceeding of the IEEE International Symposium on Computer-Aided Control System Design*, Hawai'i, USA, 486-491.

Feng, W., Brune, T., and Chan, L. (1998). Benchmarks for Testing Evolutionary Algorithms, *Asia and Pacific Conference for Control and Measurement*, Dunhuang, China, 134-139.

Feng, W., Li, Y., Chen, W. and Ballance, D. J.: Automating QFT Control System Design for Industrial Applications, in preparation.

Feng, W., and Li, Y.: Batch PID Tuning for Optimal Load Disturbance Rejection, Sensitivity and Stability Margins, in preparation.

Feng, W., and Li, Y.: Feedforward-Path Nonlinear PD Controller Optimisation with Practical Constraints, in preparation.

Fleming, P. J., and Fonseca, C. M. (1993). Genetic algorithms in control systems engineering, *Proc. 12th IFAC Triennial World Congress*, Sydney, Australia, Vol. 2, 383-390.

Flexible Intelligence Group, L. L. C. (1995). *FlexTool (GA) User Manual*, Tuscaloosa, AL.

Fogel, D. B. (1995). *Evolutionary Computation*, IEEE Press, Piscataway, NJ.

Fossen, T. I. (1994). *Guidance and Control of Ocean Vehicles*, John Wiley & Sons.

Gille, J.C., Pelegrin, M. J., and Decaulne, (1957). *Feedback Control Systems*, Mcgraw-Hill, Inc., New York, 710-717.

Goh, S. J., Gu, D. W., and Man, K. F. (1996). Multi-layer genetic algorithms in robust control system design, *Int. Conf. on Control'96, Special Session on Evolutionary Algorithms for Control Engineering*, University of Exeter, UK, 699-704.

Goldberg, D. E. (1989a). *Genetic Algorithms in Searching, Optimisation and Machine Learning*. Addison-Wesley, Reading, MA.

Goldberg, D. E., and Richardson, J. (1987). Genetic algorithms with sharing for multimodal function optimisation, GAs and their Applications: *Proc. of the Second Int. Conf. on GAs*, 41-49.

Goldberg, D. E., *et al.* (1989b). Messy genetic algorithms: motivation, analysis and first results, *Complex Systems*, Vol. 3, 493-530.

Gorez, R. (1997). A survey of PID auto-tuning methods. *Journal A*, Vol 38, No. 1, 3-10.

Graham, D., and Lathrop, R. C., (1953). The synthesis of optimum response: Criteria and standard forms, Part 2, *Trans AIEE*, Vol. 72, 273-288.

Gray, G. J., Li, Y., Murray-Smith, D. J., and Sharman, K. C. (1995). Specification of a control system fitness function using constraints for genetic algorithm based design methods, *Proc. First IEE/IEEE Int. Conf. on GA in Eng. Syst.: Innovations and Appl.*, Sheffield, 530-535.

Grefenstette, J. J. (1986). Optimisation of control parameters for genetic algorithm, *IEEE Trans. on Systems, Man and Cybernetics*, Vol. 16, No. 1, 122-128.

Guan, X., and MacCallum, K. J. (1996). Adopting a minimum commitment principle for computer aided geometric design systems, *Artificial Intelligence in Design*, Gero, J. S. and Sudweeks, F., eds, Kluwer Academic Publishers, 623-639.

Hang, C. C., Åström, K. J., and Ho, W. K. (1991). Refinements of the Ziegler Nichols tuning formula, *IEE Proceedings, Part D*, Vol. 138, No. 2, 111-118.

Highham, J. D. (1968). 'Single-term' control of first- and second-order processes with dead time, *Control*, February, 2-6.

Ho, W. K., Hang, C. C., and Cao, L. S. (1995). Tuning of PID controllers based on gain and phase margin specifications, *Automatica*, Vol. 31, No. 3, 497-502.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*, The University of Michigan Press, Ann Arbor.

Homaifar, A., and McCormick, E. (1992). Full design of fuzzy controllers using genetic algorithms, *Proc. SPIE Conf. on Neural and Stochastic Methods in Image and Signal Processing*, Vol. 1766, 393-404.

Horowitz, I. M. (1973). Optimum loop transfer functionin single-loop minimum-phase feedback systems. *Int. J. Control*, Vol. 18, 97–113.

Horowitz, I. M. (1992). *Quantitative Feedback Design Theory (QFT)*, volume 1. QFT Publications, 4470 Grinnel Ave., Boulder, Colorado 80303, USA.

Horowitz, I. M., and Gera, A. (1980). Optimisation of the loop transfer function, *Int. J. Control*, Vol. 31, 389–398.

Hrycej, T. (1994). Neural network control of test benches, *Proceedings ICARCV '94-Third International Conference on Automation, Robotics and Computer Vision*, Singapore.

Hrycej, T. (1997). *Neurocontrol towards an industrial control methodology*, published by John Wiley & Sons, Inc..

Hunt, K. J. (1992). Polynomial LQG and H_∞ controller synthesis, A genetic algorithm solution, *Proc. 31st IEEE Conf. on Decision and Control*, Tucson, AZ, Vol. 4, No. 865, 3604-3609.

Hwang, S. H., and Chang, S. H. (1987). A theoretical examination of closed-loop properties and tuning methods of single-loop PI controllers, *Chemical Engineering Science*, No. 42, 2395-2415.

Ichikawa, Y., and Sawa, T. (1992). Neural network applications for direct feedback controllers, *IEEE Trans. Neural Networks*, Vol. 3, 224-231.

Keating, M. S., Pachter, M., and Houpis, C. H. (1997). Fault tolerant flight control system: QFT design, *International Journal of Robust and Nonlinear Control*, Vol.7, No.6, 551-559.

Kirkpatrick, S. (1984). The method of simulated annealing, *Journal of Statistical Physics*, Vol. 34, 975-986.

Koza, J. R., Benett, F. H., Andre, D., Keane, M. A., and Dunlap, F. (1997). Automated synthesis of analog electrical circuits by means of genetic programming, *IEEE Transactions on Evolutionary Computation*, Vol. 1, No. 2, 109-128.

Krishnakumar, K. and Satyadas, A. (1996). Evolving multiple fuzzy models and its application to an aircraft control problem, *Genetic Algorithms in Engineering and Computer Science*, Winter *et al.*, (Eds), John Wiley & Sons Ltd.

Kwakernaak, H., and Westduk, H. (1985). Regulability of a multiple inverted pendulum system, *Control-Theory and Advanced Technology*, Vol. 1, No. 1, 1-9.

Kwong, S., Ng, A. C. L. and Man, K. F. (1995). Improving local search in genetic algorithms for numerical global optimisation using modified GRID-point search technique, *First Int. Conf. on GAs in Engi. System: Innovation and Application*, Sheffield, UK, 419-423.

Levine, W. S. (1996). *Control Handbook*, CRC Press/IEEE Press.

Li, Y. (1995). Modern information technology for control systems design and implementation, *Proc. 2nd Asia Pacific Conference on Control and Measurement*, Chongqing, China, 12-19.

Li, Y. (1999). *Neural and Evolutionary Computing IV*, Lecture Notes, Dept. of Electronics and Electrical Engineering, University of Glasgow.

Li, Y., and Häußler, A. (1996). Artificial evolution of neural networks and its application to feedback control, *Artificial Intelligence in Engineering*, Vol. 10, No. 2, 143–152,

Li, Y., Feng, W., Tan, K. C., *et al.* (1998). PIDeasy™ and automated generation of optimal PID controllers, *Asia and Pacific Conference for Control and Measurement.* Dunhuang, China, 1998, 29-34.

Li, Y., Ng, K. C., Murray-Smith, D. J., Gray, G. J., and Sharman, K. C. (1996a). Genetic algorithm automated approach to design of sliding mode control systems, *Int. J. Control*, Vol. 63, No. 4, 721-739.

Li, Y., Tan, K. C., and Gong, M. R. (1997). Global structure evolution and local parameter learning for control system model reductions, *Evolutionary Algorithms in Engineering Applications*, D. Dasgupta and Z. Michalewicz (Eds.), Springer Verlag.

Li, Y., Tan, K. C., and Marionneau, C. (1996b). Direct design of uniform LTI controllers from plant I/O data using a parallel evolutionary algorithm, *Proc. UKACC International Conference Control'96*, Exeter, U.K., 680-686.

Li, Y., Tan, K. C., Ng, K. C., and Murray-Smith, D. J. (1995). Performance based linear control system design by genetic evolution with simulated annealing, *Proc. 34th IEEE CDC*, New Orleans, 731-736.

Majhi, S., Atherton, D. P. (1999). Autotuning and controller design for processes with small time delays, IEE Proceedings: Control Theory and Applications, Vol. 146, No. 5, 415-425.

McKay, B., Lennox, B., Willis, M., Barton, G. W., and Montague (1996). Extruder modelling: a comparison of Two Paradigms, *Proc. UKACC International Conference Control'96*, Exeter, U.K., 734-739.

Mei, S., Huang, Z., and Fang, (1998). Neural network controller based on genetic algorithm, *Proceedings of the IEEE International Conference on Intelligent Processing Systems*, Wuhan, China.

Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller A., and Teller, E. (1953). Metropolis Algorithm, *Journal of Chemical Physics*, Vol. 21, 1087-1092.

Michalewicz, Z. (1994). *Genetic Algorithms + Data Structure = Evolutionary Programs*, Springer-Verlag, Berlin, 2nd Editon.

Murdock, T. M., Schimitendorf, W. E., and Forrest, S. (1991). Use of a genetic algorithm to analyze robust stability problems, *Proc. of the 1991 American Control Conference*, Massachussets, Vol. 1, 886-889.

Nelder, J. A., and Mead, R. (1965). Downhill simplex method, *Computer Journal*, Vol.7, 391-398.

Newton, Jr, G. C., Gould, L. A., and Kaiser, J. F. (1957). *Analytical Design of Linear Feedback Controls*, John Wiley & Sons.

Ng, K. C. (1995). *Switching Control Systems and Their Design Automation via Genetic Algorithms*, PhD. Thesis, Dept. of Electronics and Electrical Engineering, University of Glasgow.

Ng, K. C., and Li, Y. (1994). Design of sophisticated fuzzy logic controllers using genetic algorithms. *Proc. 3rd IEEE Int. Conf. Fuzzy Syst., IEEE Word Congr. Comput. Intell.*, Orlando, FL, Vol. 3, 1708-1712.

Ostolaza, J. X., and GarciaSanz, M. (1998). QFT-robust control of a wastewater treatment plant, *IEEE Conference on Control Applications – Proceedings*, Vol. 1, 21-25.

Patten, R. J., Chen, J., and Liu, G. P. (1995). Robust fault detection of dynamic systems via genetic algorithms, *First Int. Conf. on GAs in Eng. Syst.: Innovations and Applications*, Sheffield, UK, 511-516.

Patten, R. L., and Liu, G. P. (1994). Robust control design via eigenstructure assignment genetic algorithm and gradient-based optimisation, *IEE Proc. Control Theory Application*, 202-208.

Pemberton, T. J. (1972). PID: The logical control algorithm, *Control Engineering*, May, 66-67.

Prechelt, L. (1995). *Some Notes on Neural Learning Algorithm Benchmarking*. Technical Report, Fakultat fur Informatik, University of Karlsruhe, Germany.

Press, W.H., Teukolsky, S.A., Vetterling, W.T., and Flannery, W. V. (1994). Minimisation or Maximisation of Functions, *Numerical Recipes in C*, Cambridge University, 394-397.

Psaltis, D., Sidris, A., and Yamamura, A. (1988). A multi-layered neural network controller, *IEEE Contr. Syst. Mag.*, Vol. 8, 17-21.

Renders, J. M., and Bersini, H. (1994). Hybridizing genetic algorithms with hill-climbing methods for global optimisation: two possible ways, *Proc. 1st IEEE Int. Conf. Evolutionary Computing,* IEEE World Cong. Comp. Intel, Orlando, 312-317.

Rogers, E. & Li, Y. (eds) (1993). *Parallel Processing in a Control System Environment,* Prentice-Hall International, London, Part II: Architectures for Intelligent Control, 107-206.

Schultz, A. C., Grefenstette, J., and Jong, K. D. (1997). Learning to break things: adaptive test of intelligent controllers, *Handbook of Evolutionary Computation*, Institute of Physics Publishing and Oxford University Press.

Seborg, D. E., Edgar, T. F., and Mellichamp, D. A. (1989). *Process Dynamics and Control*, Wiley, New York, USA.

Simensen, R., and Murray-Smith, D. J. (1995). Ship steering control by neural networks using feedback linearization control laws, *Proceeding of IFAC/IMACS International Workshop on Artificial Intelligence in Real Time Control*, Bled, Slovenia, 269-274.

Sirag, D. and Weisser, P. (1987). Toward a unified thermo-dynamic genetic operator, *Proc. of the 2nd international Conf. on Genetic Algorithms*, 116-122.

Slotine, J. E., and Li, W. (1991). *Applied Nonlinear Control*, Prentic-Hall Internationl, Inc.

Smith, C. L., and Murrill, P. W. (1966). A more precise method for tuning controllers, *ISA Journal*, May, 217-219.

Smith, O. J. M. (1957). Close control of loops with dead time. *Chemical Engineering Process*, Vol. 53, 217-219.

Spears, W. and DeJong, K. (1991). An analysis of multi-point crossover, *Foundations of Genetic Algorithms*, San Mateo, 301-315.

Srinivas, M. and Patnaik, L. M. (1994). Genetic algorithms: A survey, *IEEE Computer*, Vol. 27, No. 6, 17-27.

Tan, K. C. (1997). *Evolutionary Methods for Modelling and Control of Linear and Nonlinear Systems*, PhD. Thesis, Dept. of Electronics and Electrical Engineering, University of Glasgow.

Tan, K. C., Lee, T. H., Khor, E. F. (1999). Control system design automation with robust tracking thumbprint performance using multi-objective evolutionary algorithm, *Proceeding of the IEEE International Symposium on Computer-Aided Control System Design*, Hawai'i, USA, 497-502.

Tan, K. C., Li, Y., Murray-Smith, D. J. and Sharman, K. C. (1995). System identification and linearisation using genetic algorithm with simulated annealing, *First Int. Conference on GAs in Engi. Syst.: Innovations and Application*, Sheffield, UK, 164-169.

The MathWorks, Inc. (1997). MATLAB Reference Guide.

Thompson, D. F., and Nwokah, O. D. I. (1994) Analytical loop shaping methods in quantitative feedback theory. *Journal of Dynamic Systems, Measurement, and Control*, Vol. 116, 169–177.

Unar, M. A. (1999). *Ship Steering Control Using Feedforward Neural Networks, PhD. Thesis*, Dept. of Electronics and Electrical Engineering, University of Glasgow.

Usaska, K., and Kawamata, M. (1999). Synthesis of low coefficient sensitivity digital filters using Genetic Programming, *IEEE International Symposium on Circuits and Systems*, Vol. 3, III-307-III-310.

Vesin, J., Gruter, R. (1999) Model selection using a simplex reproduction genetic algorithm, *Signal Processing*, Vol. 78, No. 3, 321-327

Voda, A. A., Landau, I. D. (1995). A method for the auto-calibration of PID controller, *Automatica*, Vol. 31, No. 1, 41-53.

Wu, S. F., Grimble, M. J., and Breslin, S. G. (1998). Introduction to Quantitative Feedback Theory for lateral robust flight control systems design, *Control Engineering Practice*, Vol. 6, No. 7, 805-828.

Xu, D. J.; Daley, M. L. (1995). Design of optimal digital filter using a parallel genetic algorithm, *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, Vol. 42, No. 10, 673-675.

Zhang, L. J. (1995). *A Study on Fuzzy Neural Network Technology & its Applications in Traffic Control of ATM Networks*, PhD. Thesis, University of Tsinghua, China.

Zhou, K., Doyle, J. C. and Glover, K. (1996). *Robust and Optimal Control*, Prentice-Hall, Upper Saddle River.

Zhuang, M., and Atherton, D. P. (1993). Automatic tuning of optimum PID controllers, *IEE Proceeding-D*, Vol. 140, No. 3, 216-223.

Ziegler, J.G., and Nichols, N. B. (1942). Optimum settings for automatic controllers, *Trans. ASME*, Vol. 64, 759-768.

Zurada, J. M. (1992). *Introduction to Artificial Neural Systems*, West Publishing Company, MN, USA.