OPTIMISATION OF TIME DOMAIN CONTROLLERS FOR SUPPLY SHIPS USING GENETIC ALGORITHMS AND GENETIC PROGRAMMING

A thesis submitted for the degree of
Doctor of Philosophy
to the Department of Electronics and Electrical Engineering
of the University of Glasgow

By

María Eva Alfaro Cid

October 2003

A Jose y mi familia

ABSTRACT

The use of genetic methods for the optimisation of propulsion and heading controllers for marine vessels is presented in this thesis. The first part of this work is a study of the optimisation, using *Genetic Algorithms*, of controller designs based on a number of different time-domain control methodologies such as *PID*, *Sliding Mode*, H_∞ and *Pole Placement*. These control methodologies are used to provide the structure for propulsion and navigation controllers for a ship. Given the variety in the number of parameters to optimise and the controller structures, the Genetic Algorithm is tested in different control optimisation problems with different search spaces. This study presents how the Genetic Algorithm solves this minimisation problem by evolving controller parameters solutions that satisfactorily perform control duties while keeping actuator usage to a minimum. A variety of genetic operators are introduced and a comparison study is conducted to find the Genetic Algorithm scheme best suited to the parameter controller optimisation problem. The performance of the four control methodologies is also compared. A variation of Genetic Algorithms, the ***Structured Genetic Algorithm***, is also used for the optimisation of the H_∞ controller. The H_∞ controller optimisation presents the difficulty that the optimisation focus is not on parameters but on transfer functions. Structured Genetic Algorithm incorporates hierarchy in the representation of solutions making it very suitable for structural optimisation. The H_∞ optimisation problem has been found to be very appropriate for comparing the performance of Genetic Algorithms versus Structured Genetic Algorithm. During the second part of this work, the use of *Genetic Programming* to optimise the controller structure is assessed. Genetic Programming is used to evolve control strategies that, given as inputs the current and desired state of the propulsion and heading dynamics, generate the commanded forces required to manoeuvre the ship. Two Genetic Programming algorithms are implemented. The only difference between them is how they generate the numerical constants needed for the solution of the problem. The first approach uses a random generation of constants while the second approach uses a combination of Genetic Programming with Genetic Algorithms. Finally, the controllers optimised using genetic methods are evaluated through computer simulations and real manoeuvrability tests in a laboratory water basin facility. The robustness of each controller is analysed through the simulation of ***environmental disturbances***. Also, optimisations in presence of disturbances are carried out so that the different controllers obtained can be compared. The particular vessels used in this study are two scale models of a supply ship called *CyberShip I* and *CyberShip II*. The results obtained illustrate the benefits of using Genetic Algorithms and Genetic Programming to optimise propulsion and navigation controllers for surface ships.

AKNOWLEDGEMENTS

First and foremost I would like to thank my supervisor Dr. Euan McGookin for encouraging me to start this research and for his help and guidance during these three years. I also would like to thank my second supervisor Prof. David Murray-Smith for his useful advice.

Thanks also to Prof. Thor I. Fossen and Prof. Asgeir J. Sørensen for making it possible for me to obtain a Marie Curie doctoral fellowship and spend some very fruitful months in Trondheim. The opportunity of using the Marine Cybernetics Laboratory has been key in my research. During my stay in Norway the help of Dr. Bjørnar Vik, Roger Skjetne, Dag Abel Sveen and Jørgen Corneliussen was invaluable.

In respect to funding I am indebted to the EPSRC for funding my registration fees and to the Department of Electronics and Electrical Engineering of the University of Glasgow for providing a maintenance grant and two travel grants to go to Norway. Also the European Union commission for funding my stay in Norway with a Marie Curie fellowship. Finally, IEEE/MTS and IEE for funding my attendance at the Oceans'01 conference in Hawaii and the UKACC'02 conference in Sheffield.

In addition I would like to mention Tom O'Hara, technician of the Department of Electronics and Electrical Engineering, always ready to help, and my colleagues David McGeoch and Alasdair Mitchell for being great office mates.

Last but not least, expresar mi agradecimiento a mi familia y a mi novio, Jose, por su apoyo y paciencia durante estos años.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1. PRELUDE

Traditionally the manoeuvring of a ship has entailed the ability of a highly experienced helmsman and the guidance measurement provided by a magnetic compass. The ship reacts to commands from the helmsman and, in turn, the helmsman relies on the readings from the magnetic compass to ensure that he is heading in the right direction. This is an example of human in the loop control. All that was to change at the beginning of the 20th century thanks to the invention the century before of the *gyroscope* in 1810 by C. A. Bohnenberger and the *electrically driven gyroscope* in 1890 by G. M. Hopkins [Fossen (2000), Skjetne (2003)]. The unreliability of magnet compasses in steel ships and the interest of the military in underwater vessels resulted in the practical application of the gyroscope as a *gyrocompass* by Dr. H. Anschutz and Elmer Sperry (patents from 1908 and 1911 respectively). Soon after this milestone, Sperry invented "Metal Mike", the first gyroscope-guided automatic steering mechanism [Fossen (2000), Skjetne (2003)]. As opposed to the open loop control that characterises the performance of a human operator, "Metal Mike" performance was based on feedback (*closed-loop control*), i.e. it regulates the rudder action by means of the difference between the desired and the current heading (measured by the gyrocompass). The operation of Sperry's steering mechanism was improved in 1922 by the work of Minorsky on position feedback control of automatic steered bodies [Minorsky (1922)]. In this work the author introduced *three-term control* or what it is now called *PID control*.

Both Sperry's and Minorsky's steering systems are *single-input single-output* (SISO) systems for controlling heading using the measurements from a gyrocompass. However, the manoeuvring capability of a ship consists of two main coupled tasks: getting the ship to sail along the desired direction (*heading control*) at the desired speed (*propulsion control*). Although traditionally it has received less attention than the heading control, the accurate control of the speed is an important issue and it affects significantly the heading performance due to the relative flow of the water over the rudder [Fossen (1994)].

Current economic and business demands require reliable manoeuvring of vessels. The issue of automation in ships had already been addressed in the 1970s [Norrbin (1970), Zuidweg (1970)]. The possibilities of reducing the number of crew as well as operating the vessels in an optimal way (minimising travel time or fuel consumption) have been and continue to be attractive features from an economic point of view. Naturally there are safety concerns: analytical studies have indicated that manual control of ships would be impossible beyond a certain size [Norrbin (1970)] and the increasing traffic densities in some areas have made some situations difficult to handle by the human operator. Besides, the offshore operations of the oil industry vessels require accurate control to be accomplished despite adverse weather conditions [Skjetne (2003)].

All this has led to further research in automatic steering systems. One way of improving the performance of traditional autopilots is the use of developments in *automatic control theory* [Dorf and Bishop (2001), Dutton *et al.* (1997)]. Automatic feedback systems have been known and used for a long time, however keystones in the development of feedback control were the invention of the steam engine governor by Watt in the 18th century and the derivation of the stability conditions of a system in the 19th century by Maxwell and Routh [Bennet (1996)]. During the first half of the 20th century the growth in knowledge and applications of electricity and the war effort resulted in the establishment of the classical control techniques [Bennet (1996)].

In general, control theory provides design strategies that allow a better understanding of the system being controlled (e.g. a vessel in marine applications) and a mechanism to regulate the way in which the system operates. In the context of ship manoeuvrability automatic control can improve the regulation of the rudder and propellers (or equivalent actuators) to achieve a better track of the desired manoeuvre at the desired speed.

As has already mentioned, traditional autopilots have been designed using PID controllers [Fossen (1994), Saelid and Jenssen (1983)]. Today, more advanced control methodologies have demonstrated their efficiency, especially when dealing with non-linear systems and noisy environments. One of the paradigms of modern control was the introduction of the time domain approaches (as opposed to the frequency domain methods used in classical control) based on the state-space equation in the 1950s [Bennet (1996)]. Their application resulted in the extensive use of state feedback compensators. The need for techniques for the computation of the feedback gain matrix resulted in the development of *Pole Placement* methods [Andry *et al.* (1983), Kautsky *et al.* (1985)]. Two major currents in modern control theory are *optimal control* [Zhou *et al.* (1996)] and *non-linear control* [Slotine and Li (1991)]. Optimal control aims to achieve not only a stable controller but also the "best" controller according to some performance index. In this context the H_∞ norm was proposed for robustness analysis of *multi-input multi-output* (MIMO) systems, leading to the development of H_∞ controllers [Glover and Doyle (1988), Zhou *et al.* (1996)]. On the other hand, non-linear control intends to provide a robust tool for improving the control of non-linear systems. One of the most successful control mechanisms developed in non-linear control has been the *Sliding Mode* control theory [Slotine and Li (1991), Utkin (1972)]. New research has been conducted in the application of these improved techniques in the field of marine control [Fossen (1994), McGookin (1997), Kallstrom *et al.* (1979), Saelid and Jenssen (1983)].

An important issue to consider while designing a control strategy is that the performance of the controllers depends on the values of the controllers' parameters. Moreover, the parameter values are connected to the system being controlled, i.e. changing the plant implies readjusting the controller's parameters. Conventionally, the designer, who attempts to find an acceptable controller solution, manually tunes these parameters by trial and error. However, this relies on the experience of the designer. If the designer is not experienced this process can become tedious and time consuming. In addition, there is no guarantee that the designed solution will perform satisfactorily as the tuning process depends on the qualitative judgement of the designer.

A solution to this problem is to use optimisation techniques that tune such parameters automatically. One of the most popular of these techniques is based on *Genetic Algorithms* [Goldberg (1989), Holland (1975)]. Genetic Algorithms (GAs) are search methods that mimic natural biological evolution. They operate on a population of potential solutions applying the Darwinian principal of *survival of the fittest* to produce better and better possible solutions to a given problem. At each generation, a new set of candidate solutions is created by the process of selecting individuals according to their level of fitness (the better the performance of the solution, the larger the fitness value is for the solution) and breeding them together using operators borrowed from natural genetics. This process results in the evolution of populations of better possible solutions to a given problem.

The advantages of GAs over traditionally parameter optimisation techniques are:

- GAs do not need gradient information of the search space (which usually is not available for the designer anyway) as opposed to classical gradient search methods
- GAs do not require any *a priori* information about the search space, i.e. there is no need for choosing an appropriate starting point since the starting points are chosen at random
- GAs are able to avoid local minima because they operate over a population of points, as opposed to conventional optimisation techniques that tackle one point at a time
- GAs perform a guided search, as opposed to enumerative schemes that set a grid over the search space and search all the points in the grid

One of the drawbacks of the method is that GAs require the encoding of solutions as a fixed-length string of genes emulating the *chromosome* structure. This chromosome-like shape makes the application of genetic operators very neat and simplifies the mathematical analysis of the method. However, it imposes a rigid way of representing the candidate solutions. Different genetic models (such as *Structured Genetic Algorithm* [Dasgupta and McGregor (1993a)] and *Genetic Programming* [Koza (1992)]) that relax the rigidity of the solutions representation have been introduced lately.

Structured Genetic Algorithm (sGA) includes a second layer in the chromosome structure that is used to activate and deactivate bits of the chromosome, defining a hierarchy. Thus, although the chromosome length is fixed as in the GA case, the activation and deactivation of parts of it modifies its effective length. The hierarchic structure makes sGA well suited to applications that require structural optimisation. It can be described as a half way approach to Genetic Programming.

Genetic Programming (GP) is based on the idea that in nature structure undergoes adaptation. Thus, the structure created over a period of time is the outcome of natural selection and sexual reproduction. GP is a structural optimisation technique (as opposed to parametric optimisation technique). The individuals in GP are represented as hierarchical structures (typically tree structures) and the size and shape of the solutions are not defined *a priori* as in GAs, but they evolve along the generations.

1.2. WORK DEVELOPED IN THIS THESIS

The first part of this work is a study of the optimisation, using GA, of controller designs based on a number of different time-domain control methodologies, i.e. *PID* [Astrom and Hagglund (1995), Dutton *et al.* (1997)], *Sliding Mode* [Slotine and Li (1991), Utkin (1972)], H_∞ [Glover and Doyle (1988), Zhou *et al.* (1996)] and *Pole Placement* [Andry *et al.* (1983), Kautsky *et al.* (1985)]. These control methodologies are used to provide the structure for propulsion controllers (for governing surge velocity) and navigation controllers (for governing heading) for a marine surface vessel. The goal of this study is to obtain controller solutions that satisfactorily perform these duties while keeping actuator usage to a minimum. The GA solves this minimisation problem by evolving controller parameters solutions that satisfy these objectives for the plant considered.

The H_∞ controller optimisation presents an added difficulty since the optimisation focus is on the weighting functions, which are not parameters but structures (transfer functions) [Glover and Doyle (1988), Zhou *et al.* (1996)]. sGA has been used for the optimisation of the H_∞. The H_∞ optimisation problem is very appropriate for comparing the performance of GA versus sGA.

Once the controller's parameters have been optimised, the controller performance is evaluated through simulations in Matlab and the robustness of each controller is analysed through the simulation of ***environmental disturbances*** (wind-generated waves) [Fossen (1994)]. Also, GA optimisations in presence of disturbances are carried out so that the different controllers obtained can be compared. The inclusion of environmental disturbances in the optimisation provides a more realistic environment for the testing of the controllers. The aim of the inclusion of waves in the optimisation is to obtain controllers that are robust against disturbances.

The particular vessels used in this study are two scale models of a supply ship called *CyberShip I* (CS1) [McGookin (1997), Strand (1999)] and *CyberShip II* (CS2) [Lindegaard and Fossen (2002)], which are the test vehicles for the Marine Cybernetics lab at the Norwegian University of Science and Technology (NTNU) in Trondheim. Supply ships are service vessels in charge of supplying the offshore oil platforms with food and equipment. They are essential for the smooth operation of the platforms. In Europe they usually work in the North Atlantic, where the probability of the sea being rough (i.e. waves higher than 2.5 m) is of the order of 20% [Fossen (1994)]. Thus, they have to be able to cope with adverse weather conditions.

Computer-generated simulations based on a non-linear hydrodynamic model of CyberShip I and CyberShip II are used in the design of the controllers and in the optimisation studies. The results obtained from this study illustrate the benefits of using GAs to optimise propulsion and navigation controllers for surface ships.

In addition, the resulting controllers have been tested in the real plant in the Marie Curie Training Site in the NTNU. The facilities in the lab include a water tank equipped with a positioning system and a wave synthesiser for the generation of environmental disturbances. The results obtained in the real implementation allow meaningful evaluation of the optimised controllers and a comparison benchmark for performance and robustness against disturbances. Also, the advantages of including disturbances in the simulation during the controller optimisation have been analysed and discussed. The real testing was conducted over two visits to the facilities in the NTNU and slight modifications were made to the system between the visits.

The second stage of this study involves the implementation of GP to the combined optimisation of parameters and structures for the controllers. In the context of this thesis GP is used to evolve control strategies that, given as inputs the current and desired state of the propulsion and heading dynamics, provide through the actuators the commanded forces that are required to manoeuvre the ship.

One of the difficulties when implementing a GP is how to generate the numerical constants that might be needed for the solution of the problem undertaken. In this research, using the findings of the GA study as a basis, two GP algorithms have been implemented to solve the supply ship control problem. The only difference in these implementations is the way of generating numerical constants. The first approach uses Koza's recommendation where the constants are randomly generated without further modification (unless they undergo mutation) [Koza (1992)]. The second approach uses a combination of GP plus GA [Howard and D'Angelo (1995)]. Before being evaluated, each individual in the population experiences a GA parametric optimisation of its numerical constants.

As in the GA case, the GP optimisation has been performed with and without disturbances. The aim of the GP optimisation is to find a controller that structurally is best suited to the surface vessel control problem. Again the addition of disturbances during the optimisation is meant to improve the robustness of the controller structure. The best GP results were tested in the real plant too.

1.3. MAIN CONTRIBUTIONS OF THIS THESIS

The main contributions of this thesis are listed below:

- Design, optimisation and comparison of the real and simulated responses of four control methodologies for the control of the propulsion and heading of a supply ship using genetic optimisation methods
- Comprehensive comparison of various Genetic Algorithms schemes for the optimisation of the controller's parameters in the ship dynamics problem
- Use of the Structured Genetic Algorithm for the structural optimisation of the weighting functions for a H_∞ controller and comparison with the results obtained with the standard Genetic Algorithm
- Assessment of the effect on the robustness against external disturbances of the controllers by the inclusion of realistic environmental noise in the optimisation process
- Implementation in Matlab and use of Genetic Programming for the structural optimisation of two controllers for the heading and propulsion dynamics of a supply ship
- Comparison of two methods for the generation of constants in the Genetic Programming optimisation, namely Random Generation and Genetic Algorithms

Part of this work has already been published in the publications listed below:

- Alfaro-Cid, E. and McGookin, E.W. (2001), "Genetic Algorithm Optimisation of Oil Tanker Control Systems", *Proceedings of the IFAC Conference on Control Applications in Marine Systems (CAMS'01)*, Glasgow (UK), pp 227-32
- Alfaro-Cid, E., McGookin, E.W. and Murray-Smith, D.J. (2001), "Genetic Algorithm Optimisation of a Supply Ship Propulsion and Navigation Systems", *Proceedings of the MTS/IEEE Oceans Conference*, Honolulu (USA), pp 2645-2652
- Alfaro-Cid, E., McGookin, E.W. and Murray-Smith, D.J. (2001), "Genetic Algorithm Optimisation of a Ship Navigation System", *Acta Polytechnica*, Vol. 41, No. 4-5, pp 13-19
- Alfaro-Cid, E. (2002), "Genetic Algorithm Optimisation of an H_∞ Controller for an Oil Platform Supply Ship", *UKACC Postgraduate Symposium on Control (UKACC'02)*, Sheffield (UK), pp 137-142
- Alfaro-Cid, E., Loo, M., Mitchell, A. and McGookin, E.W. (2003), "AUV Route Planning Using Genetic Algorithms", *1st IFAC Workshop on Guidance and Control of Underwater Vehicles (GCUV2003)*, Newport (UK), pp 95-100

1.4. OUTLINE OF THESIS

The outline of this thesis is as follows:

Chapter 2 discusses the existing literature in the areas covered in this thesis. Firstly it provides a summary of the *state of art* in the four controller structures that have been optimised using GAs: linear PID control, non-linear Sliding Mode, optimal H_∞ and state feedback Pole Placement. There is a particular emphasis in those authors who have used evolutionary techniques for the tuning of these controllers. The second part portrays the research done in genetic optimisation methods such as GAs and GP, their main features and applications. Again the emphasis is mainly on control applications.

In Chapter 3 the mathematical model used in simulations throughout the research is presented. The model describes the ship and actuator dynamics of CyberShip I and CyberShip II, a supply ship scale model developed in the Norwegian University of Science and Technology (NTNU), Trondheim. This chapter also describes the facilities of the Marine Cybernetics research lab in the Marie Curie Training Site in NTNU, where the real implementation of the controllers was conducted. This laboratory consists of a water tank equipped with a wave generator that provides a very realistic environment for controller testing. In addition the mathematical wave model used in simulations for checking the controllers' robustness to environmental disturbances is presented.

Chapter 4 describes the theory behind the four control methodologies used in the research and how the designs of the controllers for CyberShip I and CyberShip II have been carried out. *Individual Channel Analysis and Design* (ICAD) [O'Reilly and Leithead (1991)] is used for analysing the extent of decoupling within the model dynamics of the ship. ICAD performs a diagnosis of the level of coupling between the dynamics of the plant, in this case, heading and propulsion dynamics. Linear PID and non-linear Sliding Mode control have been used to control the decoupled dynamic model. The design of the Sliding Mode controller is based on McGookin (1997). On the other hand, optimal H_∞ and state feedback Pole Placement are used for the control of the MIMO plant. The H_∞ design is based on the state-space approach presented by Zhou *et al.* (1996); while for Pole Placement the robust method proposed in Kautsky *et al.* (1985) is applied.

The structure, operators and main features of Genetic Algorithms (GAs) are presented in Chapter 5. The popularity of GAs as an optimisation technique has led to thousands of publications in the field. In order to find the GA scheme better suited to the control problem exposed, a comparison study is discussed and the findings obtained are presented in the chapter. The study focuses on the performance of the genetic operators. The resulting enhanced GA scheme has been used to generate the results from Chapters 6 and 8. Finally a genetic method called Structured Genetic Algorithm (sGA) is introduced. It provides the GA with a hierarchical structure that is well suited for structural optimisation.

Chapter 6 explains how the GA optimisation and real testing of the four controller structures described in Chapter 4 have been carried out through simulations in Matlab. Each controller is optimised in two different ways: normal conditions and in the presence of environmental disturbances (wind-generated waves). All the results are tested on the actual CyberShip II in the NTNU facilities. The chapter shows the results obtained from the GA optimisation and the real testing. The performance of the controllers is discussed and comparisons are drawn from the simulated and real results obtained. The advantages and disadvantages of the inclusion of disturbances during the optimisation process are argued. In addition, the H_∞ optimisation has been also attempted using sGA. H_∞ is an excellent example for the application of sGA since the tuning of the controller implies the adjustment of some transfer functions (weighting functions). Therefore, it is not only a parameter but also structural optimisation. The results obtained from the sGA optimisation of H_∞ are included in the chapter.

Chapter 7 presents the final optimisation technique used in this thesis: Genetic Programming (GP). Firstly it describes the structure, operators and tree representation typical of GP. Secondly, using as a starting point the results obtained from the GA optimisation, a GP algorithm implemented in Matlab for the CyberShip II control problem is presented. The aim of using GP as an optimisation technique is to optimise the structure of the controller. Two separate approaches are considered to solve the problem of generating numerical constants in GP. The first approach uses just a random generator as proposed by Koza (1992). The second approach intends to improve the accuracy of the results obtained by including a GA that will optimise the numerical constants while the GP optimises the structure.

The results obtained from both these GP methodologies are shown in Chapter 8. As in the GA case all the optimisations have been performed with and without waves for comparison purposes. The best controllers obtained have been tested in the facilities in the NTNU. An examination of all the results provides insight into the role of various functions in the control strategy as well as a good comparison ground for the two separate GP proposals.

Finally the thesis ends with Chapter 9. This chapter is divided in two parts, in the first part the conclusions drawn in each of the previous chapters are summarised and the second part includes suggestions for further work.

CHAPTER 2

BACKGROUND LITERATURE SURVEY

2.1. INTRODUCTION

The research work developed in this thesis involves two main research fields: control methodologies and genetic optimisation techniques. This chapter presents a review of the most relevant literature available concerning these two areas of knowledge.

The chapter is divided into two sections. Section 2.2 deals with the origin, developments and state-of-art of the four different control methodologies studied (i.e. PID, Pole Placement, Sliding Mode and H_∞). Section 2.3 presents a survey of significant developments in the field of Genetic Algorithms, Structured Genetic Algorithms and Genetic Programming and their current applications.

2.2. CONTROL METHODOLOGIES LITERATURE REVIEW

2.2.1. PID LITERATURE REVIEW

The PID controller [Astrom and Hagglund (1995), Dutton *et al.* (1997)] was first described by Minorsky (1922). During the Second World War a tremendous interest was developed in the classical control theory and particularly the PID control of processes [Bennet (1996)].

Its simplicity and general good performance made its utilisation very widespread in industry. It has been stated that in process control applications more than 95% of the controllers are PID type [O'Dwyer (2000)]. The literature reflects the popularity of PID control with hundreds of papers tackling different aspects of it (a review by Lelic and Gajic (2000) identifies at least 333 papers on PID only in the 90s). A basic text in the field is Astrom and Hagglund (1995).

The choice of appropriate PID parameters can be achieved manually by *trial and error*, using as guidelines the transient and steady response characteristic of each of the three terms. However, this procedure is very time consuming and requires certain skill.

Numerous methods have been developed to try to simplify the tuning of PIDs. Currently, according to Lelic and Gajic (2000), the most popular tuning methods are: the *relay based tuning* [Astrom and Hagglund (1984)], various ways of *frequency domain tuning* [Hagglund and Astrom (1991)] and *optimal tuning* by means of minimising an integral performance index [Ho *et al.* (1999)]. The traditional *Ziegler-Nichols method* [Ziegler-Nichols (1942)] is still widely used in industry (see Section 4.3.1). Also, a new approach based on parameter sensitivity functions in an iterative tuning process has been developed recently at the University of Glasgow [Murray-Smith *et al.* (2003)].

The relay based tuning [Astrom and Hagglund (1984)] provides a method that estimates the critical gain and the critical frequency of a system by introducing a feedback loop with a relay. Hagglund and Astrom (1991) summarises several adaptive techniques for tuning of PIDs using frequency response as opposed to a rational transfer function of the model. This avoids model uncertainties. The techniques are based on the previous relay feedback method. Ho *et al.* (1999) describes the relationship between the integral square error performance index, the gain margin and the phase margin and gives recommendations for gain and margin specification to get more performance out of PID controllers.

2.2.2. POLE PLACEMENT LITERATURE REVIEW

Since Wonham (1967) proved that the state feedback pole placement problem has a solution if, and only if, the system (A, B) is controllable, there have been numerous publications concerning Pole Placement and its applications [Andry *et al.* (1983)].

However, it was not until the early 1980s that a series of papers appeared that tackled the multi input case of the pole assignment problem (called eigenstructure assignment) [Fahmy and O'Reilly (1982), Fletcher (1981), Porter and D'Azzo (1977)]. These approaches exploited in various ways the flexibility offered by the choice of eigenvectors. A very good review of the various methods proposed can be found in White (1995).

Among them there is the algorithm proposed by Kautsky *et al.* (1985). The algorithm takes advantage of the extra degree of freedom in the assignment problem for choosing eigenvectors that are robust in the sense of minimising the perturbations in the eigenvalues produced by variations in the system or feedback gain matrices.

Pole Placement has been used as a control method in many fields, but it is remarkable the number of applications in the aerospace field that can be found in the literature [Manness and Murray-Smith (1992), Andry *et al.* (1983)].

2.2.3. SLIDING MODE LITERATURE REVIEW

What it is now known as Sliding Mode control [Slotine and Li (1991), Utkin (1972)] started to develop in the early 1950s in the Soviet Union. The earliest formulations of the Sliding Mode theory were based in single input systems and it was further extended to multi input systems in the early 1970s [Utkin (1972), Utkin and Yang (1978)].

The excellent survey in variable structure control with Sliding Mode written by Hung *et al.* (1993) identifies 1980 as the year when research in Sliding Mode control accelerated, as a result of the recognition of its robustness properties by the scientific community.

Since then, most of the papers about Sliding Mode belong to four categories [Hung *et al.* (1993)]:

- Development of Sliding Mode controllers for different system models, such as nonlinear systems [Slotine (1984)] or discrete time systems [Wu (1997)].
- Extensions of the objectives of control from the original stabilization problem to tracking [Slotine (1984)], model following [Fossard (1993)] or state observation [Han *et al.* (2000)].
- Exploration of properties such as robustness [Mudge and Patton (1988)] and elimination of chattering [Bartolini *et al.* (1998)].
- Applications in various engineering problems. Sliding Mode controllers have been successfully used in the aerospace field [Fossard (1993), Mudge and Patton (1988), Salamci *et al.* (2000)] and in the marine field in plants such as underwater vehicles [Healey and Marco (1992), Healey and Lienard (1993), Lea *et al.* (1999), McGookin (1997)] and surface vessels [Fossen (1994), McGookin (1997)].

2.2.4. H_∞ LITERATURE REVIEW

Based on the *small gain theorem* [Zames (1966)], a new optimal control method was introduced by Zames (1981). He proposed the utility of minimizing an H_∞ norm rather than the usual L_2 quadratic norm. The H_∞ norm had been found to be appropriate for specifying both the level of plant uncertainty and the signal gain from disturbance inputs to error outputs in the controlled system. In addition, the H_∞ norm gives the maximum energy gain of the system. This is in contrast to the H_2 norm, which gives the variance of the output given white noise disturbances [Zhou *et al.* (1996)].

The robust stability consequence, derived from the small gain theorem, is the main cause of the interest in the development of H_∞ methods in the 80's. The synthesis of controllers that achieve an H_∞ norm specification became a major research focus [Grimble (1986), (1987a), (1987b)]. But the standard frequency-domain approaches to H_∞ became computationally problematic when dealing with MIMO systems.

In order to solve the MIMO H_∞ control problem [Zhou *et al.* (1996)], state-space formulations of the H_∞ optimal control problem arose. Although the first state-space approaches were published in the mid 80's, it wasn't until 1988 that a simpler and less computationally demanding model was presented [Glover and Doyle (1988)]. Thus, at this stage H_∞ control theory had reached a fairly mature state, complete with state-space formulations [Glover and Doyle (1988)] and comprehensive comparisons with the widely known H_2 (or *LQG*) control problem [Doyle *et al.* (1989), Grimble (1986)].

Given the difficulty of the design process of the H_∞ controller, various techniques have been developed to simplify it. One of the most popular is a design technique that incorporates *loop shaping* methods to obtain performance/robust stability trade-offs. Loop shaping controller design involves finding a controller that shapes the loop transfer function L so that the loop gains, $\underline{\sigma}(L)$ and $\overline{\sigma}(L)$, are kept away from the boundaries specified by the performance requirements at low frequencies and by the robustness requirements at high frequencies [Postlethwaite *et al.* (1991), Zhou *et al.* (1996)]. A particular formulation of the H_∞ loop-shaping problem using *normalised coprime factorisation* was developed by McFarlane and Glover (1992). The idea behind this approach is to guarantee overall robust stability by maximising stability robustness with respect to a particular type of unstructured stable perturbations, i.e. left/right coprime factor perturbation [Zhou *et al.* (1996), Walker and Postlethwaite (1996), Postlethwaite and Bates (1999)].

The state-space H_∞ control problem has been extensively used to solve problems in the aerospace field in aircraft flight control [Postlethwaite and Bates (1999), Voulgaris and Valavani (1991), Sweriduk *et al.* (1998)] and also in helicopter flight control [Walker and Postlethwaite (1996)].

In the marine field state-space H_∞ controllers have been successfully designed for many applications, from submarine and AUV control [Marshfield (1991), Liceaga-Castro and van der Molen (1995), Silvestre and Pascoal (1997)] to dynamic position of production platforms [Dohna and Tannuri (2001)] or surface vessels course control [Desanj *et al.* (1997), Katebi *et al.* (2001)].

A very complete literature review can be found in Doyle (1996).

2.3. GENETIC OPTIMISATION METHODS LITERATURE REVIEW

The problem of finding the optimum value (i.e. either the maximum or the minimum) of an objective function in a given search space has been of prime interest for the scientific community since the pioneering work on differential calculus by Newton and Leibniz in the 17th century [Edwards (1979)].

Ever since, much effort has been put into the development of techniques that provide a solution for the optimisation problem. The current literature identifies 3 main classes of search techniques: *calculus-based*, *enumerative* and *guided random search* techniques [Goldberg (1989), Ribeiro Filho *et al.* (1994)].

Calculus-based techniques rely on the use of the gradient of the objective function. They have been extensively studied and have a strong theoretical background. However, from a practical point of view these techniques present two main drawbacks: first, they can only be used in a restricted set of functions with well-defined slope values (i.e. they depend on the existence of derivatives) and second, they are very successful in finding a local optimum of the objective function, but they struggle to find the global optimum [Goldberg (1989)].

The concept behind *enumerative techniques* is very straightforward. An enumerative scheme searches every point of the objective function's search space, assuming it is finite. Otherwise, the search space needs to be discretised *a priori*. In spite of the simplicity of the technique, it is not very efficient. Many search spaces are just too large to search one point at a time [Goldberg (1989)].

Guided random search techniques are based on enumerative techniques but use additional information to guide the search. Two good examples of guided random search techniques are *Simulated Annealing* [Metropolis *et al.* (1953)] and *Evolutionary Algorithms* [Back (1996)]. While Simulated Annealing uses thermodynamic concepts to guide the search, Evolutionary Algorithms rely on concepts drawn from natural genetics.

The main advantage of guided random search techniques is how they balance two conflicting objectives in the optimisation process: the *exploration* of the search space and the *exploitation* of the best solution (i.e. volume-oriented or path-oriented optimisation) [Back (1996), Michalewicz (1992)]. While *volume-oriented* techniques explore the search space ignoring the most promising areas, and *path-oriented* methods exploit the best solution for improvement neglecting the exploration of the remaining search space, guided random search techniques achieve a remarkable trade-off between exploration and exploitation. This balance is extremely important to find global optima.

In Evolutionary Algorithms this balance is achieved by exploiting the closed relation between the concepts of adaptation and optimisation.

Back (1996) defines Evolutionary Algorithms (EAs) as algorithms "based on models of organic evolution, i.e. nature is the source of inspiration. They model the collective learning process of a population of individuals, each of which represents not only a search point in the space of potential solutions to a given problem, but also may be a temporal container of current knowledge about the laws of the environment. The starting population is initialized by an algorithm-dependant method, and evolves towards successively better regions of the search space by means of (more or less) randomized processes of recombination, mutation and selection. The environment delivers quality information (fitness value) for new search points, and the selection process favours those individuals of higher quality to reproduce more often than worse individuals. The recombination mechanism allows for mixing of parental information while passing it to their descendants, and mutation introduces innovation into the population."

The beginnings of EAs can be traced back to the early 1950s, when several biologists used computers for simulation of biological systems [Michalewicz (1992)]. In the late 1950s some attempts were made towards evolving computer programs and optimisation based on natural evolution, but they failed, due in part to the restrictions imposed by the computers available at that time. It was not until

1964 that the first successful results from an approach to EAs were presented by Fogel (1964). His approach was called *Evolutionary Programming* (EP).

Despite Fogel's successful results, his idea of Evolutionary Programming received little attention from the rest of the scientific community. The interest generated by other approaches to EAs and the success of these techniques in many applications led to a review of Fogel's work in the late 1980's and early 1990's.

At the beginning of the seventies, the work developed independently in the 1960's on *Evolution Strategies* (ESs) in Germany by Bienert, Rechenberg and Schwefel [Rechenberg (1973), Schwefel (1975)] and on *Genetic Algorithms* (GAs) in the United States by Holland (1975), had reached a fairly mature state.

In Evolutionary Programming and Evolution Strategies the main genetic operator is mutation. Originally, the populations were formed by a single individual, which is mutated to create a single offspring individual. After testing the performance of the offspring, the better individual of both, parent and offspring, becomes the parent of the next generation. However, mutation is not a uniform random operator as in classical EP but a normally distributed mutation with expectation zero and given variance. This way, an individual is represented by two vectors. The first vector represents a point in the search space and the second is a vector of standard deviations.

Hence, until the early 1990's, the best-known algorithm types in EAs were: Evolutionary Programming, Evolution Strategies and Genetic Algorithms. More recently, a new class of EA was developed by Koza and it was called *Genetic Programming* (GP). In Genetic Programming [Koza (1992)], Koza proposes that, instead of building an evolution program to solve a given problem, we should rather search the space of possible computer programs for the best one. Optimising structures instead of parameters.

The growing interest generated by these approaches during the last 30 years, has led to several international conferences on the topic, lots of publications and a wide range of applications. Many researchers have adapted the algorithms to their own problem, modifying the genetic operators to make them more efficient in their particular search space. These modifications have produced algorithms very different from the classical formulation, so that the boundaries among EAs become blurred [Back (1996), Michalewicz (1992)].

The most widely known type of EAs is the Genetic Algorithm [Back (1996), Goldberg (1989), Holland (1975), Michalewicz (1992)].

2.3.1. GENETIC ALGORITHMS LITERATURE REVIEW

The basic principles of Genetic Algorithms were first proposed by Holland (1975) based on the work developed by him and his colleagues in the University of Michigan during the 1960's and early 1970's. The goal of their project was defined by DeJong (1975) in his doctoral dissertation as "to understand and abstract from natural systems the mechanisms of adaptation in order to design artificial systems of comparable sophistication." The algorithm they implemented (called the *Simple Genetic Algorithm* (SGA)) is very neat and allows a fairly simple mathematical analysis, however later authors have improved the operators used in the SGA (see Chapter 5) [Back (1996), Michalewicz (1992)].

Thereafter, hundreds of papers have been published on the topic of GAs and its applications. There is much literature [Back (1996), DeJong (1975), Goldberg (1989), Michalewicz (1992)] on this subject and various international conferences on GAs have been held periodically.

GAs have been successfully implemented in a number of different fields. Goldberg (1989) presented quite an exhaustive list of applications in disciplines as diverse as social sciences, medicine or physical sciences.

In the engineering field the number of applications is also very large, from structure optimisation [Day (2001)] to job scheduling [Shaw and Fleming (2000)], system identification [Kristinsson and Dumont (1992), Tan and Li (1996), (1997)], measurement [Brooks *et al.* (1996)] or robotics [Yang and Billings (2000), Wang and Zalzala (1996b)].

For its use in systems and control engineering, GAs have been applied to a number of control problems. A very good classification of GA applications in control engineering can be found in Chipperfield and Fleming (1995).

These include: *controller parameter optimisation* for different control methodologies such as Predictive Control [Smierzchalski *et al.* (2001)], Gain Scheduling [Gray *et al.* (1997)] and PID, Sliding Mode, H_∞ and Pole Placement, as it is shown below; *controller structural optimisation* [Chowdhury and Li (1996), French *et al.* (1997), Tang *et al.* (1996)] and *multiobjective controller design* [Chipperfield and Fleming (1996), Chipperfield *et al.* (1999)].

In the field of marine applications of control, GA optimisation has been applied mainly to *controller parameter optimisation* of various marine vehicles such as surface vessels [Alfaro-Cid *et al.* (2001a), (2001b), Alfaro-Cid and McGookin (2001), McGookin (1997)], submarines [McGookin (1997)] or AUVs [Smierzchalski *et al.* (2001)] and for *route planning* [Alfaro-Cid *et al.* (2003), Alvarez and Caiti (2001), Smierzchalski (2001), Stawicki and Smierzchalski (2001)].

GA tuning of PID controllers

The advantages of GAs as optimisation techniques have also attracted many researchers in the field of PID control as proved by the numerous references in the literature. This genetic approach tries to save the designer having to manually retune the results, as usually is the case when using the above-mentioned tuning methods (especially with non-linear plants). Also, the idea of the simplicity and wide applicability of GAs (since they do not depend of the plant characteristics) was appealing to the researchers [Wang and Kwok (1994)].

The earliest papers on GA-based PID controllers were published in the early nineties. Most of this pioneering work was undertaken by Wang and Kwok (1994). Various conferences, such as the 1995 and 1997 International Conferences on Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA) held in Sheffield and Glasgow, respectively, were very prolific in GA applications to PID tuning [Ahmad *et al.* (1997), DeMoura Oliveira and Jones (1997), Jones and DeMoura Oliveira (1995), Jones *et al.* (1997), Krohling (1997), Salami and Cain (1995), Vlachos *et al.* (1997)].

Recent publications in PID tuning using GAs can be divided between those concerned with improvements in the controller tuning and those focused on the application. The former would include latest research in fuzzy PID control [Cho *et al.* (1997)], adaptive PID control [Porter and Jones (1992)], self-tuning PID [Mitsukura *et al.* (2000)], combinations of neural networks and GAs for PID tuning [Omatu and Deris (1996)] or PID genetic tuning for disturbance rejection [Krohling and Rey (2001)].

The latter group of papers show that the application fields of genetic tuned PID controllers are numerous, ranging from power electronics [Wang *et al.* (2002)] to robotics [Homaifar *et al.* (1997)].

In the field of marine control, however, the references to PID tuning using GAs are scarce [Alfaro-Cid and McGookin (2001), Alfaro-Cid *et al.* (2001a), (2001b)].

GA tuning of Pole Placement controllers

The earliest papers on genetic optimised Pole Placement controllers were not published until the mid 1990s and still they are sparse. In the Pole Placement problem the choice of what is going to be optimised with the GA is not as straightforward as in the PID gains.

GAs have been used mainly for obtaining the gains of the feedback matrix that provides the appropriate closed-loop eigenstructure [Abdel-Magid *et al.* (1997), Lam and Tam (2000), Patton and Liu (1994)], given a set of fixed or constrained eigenvalues.

Another quite common approach consists of using GAs to optimise the Q and R weighting matrices so that the closed-loop eigenvalues are located as close as possible to the desired eigenvalues of a linear quadratic (LQ) optimal system [Clarke and Davies (1997), Davies and Clarke (1995)].

Finally, only two references were found in the literature that used the approach adopted in this thesis (i.e. to optimise the pole locations instead of the feedback gains): Wang and Zalzala (1996a) used a GA to find the pole locations for a pole assignment self-tuning adaptive controller and McGookin (1997) used a GA to find the pole locations for the equivalent controller in a Sliding Mode control structure.

Again, a significant number of the papers concerning genetic tuning of Pole Placement controllers are related to aerospace applications [Clarke and Davies (1997), Davies and Clarke (1995), Patton and Liu (1994), Mengali (2003)].

GA tuning of Sliding Mode controllers

The first papers in GA-tuned Sliding Mode controllers were published in the mid 1990s [Moin *et al.* (1995), Ng *et al.* (1995)]. Sliding Mode applications of GA have not been as popular as PID ones, but still there are some publications in the area.

A large majority of the publications deal with GA optimisation of fuzzy-Sliding Mode controllers. GAs are used to optimise the membership functions in the fuzzy rule bases [Chen and Chang (1998)].

It is worth mentioning as well the contribution from researchers of the University of Glasgow, especially in marine applications [Alfaro-Cid *et al.* (2001a), Li *et al.* (1996), McGookin (2001), McGookin *et al.* (1996), (1997a), (1997b), (2000a), (2000b), (2000c)].

Other non-marine applications of genetically optimised Sliding Mode controllers would be in power systems control [Matas *et al.* (2000)] or robotics [Zhu *et al.* (2001)].

Latest research in the area includes on-line GA optimisation for adaptive Sliding Mode controllers [Lin and Chou (2003)].

GA tuning of H_∞ controllers

References to H_∞ weighting functions tuning using GAs are not as frequent as with previous controller techniques. The fact that the choice of weighting functions involves, not only the adjustment of the parameters, but also determining the weighting structure, makes it a complicated optimisation problem. However, since the late 1990s, some genetic approaches to the H_∞ controller design problem can be found in the literature.

Most authors use a loop-shaping method combined with a GA search. Given a predetermined structure for the pre and post compensators required in the loop-shaping technique, the GA is used to optimise the parameters that define that structure [Christiansson and Lennartson (1999), Dohna *et al.* (1997)]. On the other hand, Dakev *et al.* (1997) allows the GA to select the optimal structure from a set, giving a structural dimension to the search process.

Despite of the small number of papers in the area of GA-based H_∞ controllers, the range of applications is quite varied: from motor control [Chouiter *et al.* (1999)] to a magnetic levitation train [Dakev *et al.* (1997)]

In the field of ship control some research has been done [Alfaro-Cid (2002), (2003d), Donha *et al.* (1997)].

2.3.2. STRUCTURED GENETIC ALGORITHM

The Structured Genetic Algorithm (sGA) was first proposed by Dasgupta and McGregor (1991) (an updated version of the report can be found in Dasgupta and McGregor (1993a)). This genetic model proposes a new chromosome structure in the sense that chromosomes become a multi-layered structure instead of a linear one.

Thus, the central feature of sGAs is the use of hierarchies and gene redundancy. Genes in the higher level of the structure act as "control genes" that activate or deactivate sets of lower level genes, but the deactivation of a set of genes does not mean its deletion. They are kept in the chromosome and can be reactivated later on by mutation or crossover of the control genes.

Dasgupta and McGregor (1993a) argue that the main reason for the unsatisfactory performance of GAs while tackling some problems is their simplistic chromosome representation. Once an optimisation converges to a local optimum to "jump" out of it requires a multi-gene change. Unless there is a great diversity in the population traditional crossover and gene mutation cannot cause that desired effect.

With the new chromosome structure presented in sGA, the mutation of one of the high level genes produces the activation or deactivation of lower set of genes, i.e. it promotes multiple changes to occur simultaneously.

Thus, the motivations for this new genetic model are [Dasgupta and McGregor (1993a)]:

1. The hierarchic multi-level structure allow multi-gene changes that avoid premature convergence
2. Sets of deactivated genes perform as a memory
3. The model is especially well suited to a changing fitness scenario since the redundant genes may carry information that will help in the adaptation process [Dasgupta and McGregor (1992a)]

As well as creating the model, the authors developed a wide variety of applications in various fields ranging from neural networks [Dasgupta and McGregor (1992b), (1993b)] to mechanical problems [Dasgupta and McGregor (1992c)] or artificial GA-deceptive problems [Dasgupta (1994)].

However, although the intention of the authors was to provide the sGA with a mechanism for adaptation to changing environments, the multi-layered chromosome configuration in sGA makes it very suitable for optimisation of structures and in that area is where most of the later applications of sGA take place.

Mainly, sGA have been used for structural optimisation (topology and weightings) of neural networks [Chalkiadakis *et al.* (2001), Tang *et al.* (1995)]. Other relevant applications are in data classification [Hsu and Hsu (2002), Imai *et al.* (1999)] and as an engineering design aid [Parmee (1998), Rafiq and Williams (1998)].

The control applications references using sGA are scarce. Dasgupta and McGregor (1993b) have presented a control application of sGA: the design of a neurocontroller for the pole-cart system. Once more the sGA has been used for the optimisation of the neural network that controls the system, obtaining good results. And Tang *et al.* (1996) used sGA for the design of the pre compensator and post compensator in a H_∞ controller design with loop-shaping.

In the field of marine technology sGA has been used as a route planner by Ono *et al.* (1996).

2.3.3. GENETIC PROGRAMMING

The foundations of Genetic Programming (GP) were laid by Koza (1992) in his book entitled *Genetic programming: On the Programming of Computers by Means of Natural Selection*. The main point in Koza's work is to prove that: "genetic programming provides a way to search the space of possible computer programs for an individual computer program that is highly fit to solve a wide variety of problems from many different fields." He presents a GP algorithm coded in LISP and applied it to various problems related to very different research fields, from symbolic regression and optimal control to discovering of game-playing strategies, robotic planning or pattern recognition. He aims to prove empirically the efficiency of GP as a problem-solving technique.

In Koza (1994), further developments in the technique were introduced, mainly the development of *Automatic Defined Functions* (ADFs). ADFs are trees that are evolved in parallel with the main tree and can be called by it as a terminal node. Koza's aim is to develop a tool that is able to decompose a problem into subproblems, solve the subproblems and assemble the solutions to the subproblems into a solution of the overall problem, exploiting any symmetry or pattern of the problem.

Following the work by Koza, several PhD theses were published, such as Tackett (1994) and O'Reilly (1995). Both works focus on explaining why GP works and how GP could be improved, since such an extensive empirical work had already been presented by Koza.

Tackett's (1994) thesis aims to clarify the mechanisms that govern GP by presenting an analogy between GP and *Beam Search* [Rosenbloom (1987)], a well-known heuristic search method used in Artificial Intelligence. He also analyses various selection operators in the context of noisy and under sampled data and introduced what he called the *Greedy Recombination Operator*. In order to further understand the underlying mechanisms of GP two problems are constructed: a *Royal Road* and a *Deceptive* problem and an approach called *Gene Banking* is taken so that relevant statistics such as frequency of occurrence of subexpressions, time of first creation, and time of extinction are recorded.

O'Reilly's (1995) thesis aims to be a systematic analysis of GP. She studies the influence of the designer in the performance of GP while choosing the function and terminal sets and the role of hierarchy in GP. She also states a *Building Block Hypothesis* for GP and derives a *Schema Theorem*. In the second part of her thesis, O'Reilly compares the performance of GP with that of Simulated Annealing and Stochastic Iterated Hill Climbing by modifying them introducing a new mutation operator called *Hierarchical Variable Length Mutate* operator (HVL-Mutate). She repeats the comparison substituting the mutation operator by a crossover operator. Finally, a new hybrid algorithm based on the previous results is introduced

In the current literature the amount of GP papers is remarkably large and diverse, specially considering that Koza's first book was published only 10 years ago. These papers can be grouped in three categories: papers about GP mechanisms, papers about GP operators and GP applications. By far, the last group is the biggest.

In order to prove empirically the effectiveness of GP, Koza (1992) and Koza (1994) present a broad range of GP applications to a variety of fields, from optimal control to evolution of Boolean expressions including robot path planning, symbolic regression, emergent behaviour, strategy, molecular biology...

One of the areas where GP has been applied quite extensively has been the evolution of electrical hardware [Koza *et al.* (1997), Fernandez *et al.* (2002)]. GP has been used for the placement and routing of circuits.

Applications to research in molecular biology and medical screening are reported in Dracopoulos and Kent (1997). The structures being researched in molecular biology are very large and GP provides a mechanism to accelerate the analysis. It can also be used for the identification of patterns in the patients that suffer a grave condition that would help in the prevention of other cases.

The robotic community has used GP for planning and controlling robot movements [Ebner (1998), Busch *et al* (2002)], and cooperative robot team strategies [Luke

(2000a), Yanai and Iba (2002)]. Koza (1992) itself presented some examples of this application. For a review of the topic, see Mataric and Cliff (1996).

GP has also been used for prediction [Dorado *et al.* (2002), Howard and Roberts (2002)] and modelling problems [Gray *et al.* (1998), Bastian (2000), Grosman and Lewin (2002)].

Other applications include computer animation [Gilfind *et al.* (2000)], digital signal processing [Esparcia-Alcazar (1998)] and music generation [Putnam (1996)].

So far, GP has been applied to a small number of control problems. Koza (1992) included some of what he called cost-driven evolution problems (i.e., optimal control problems), such as the cart entering, broom balancing and truck backer problems. Also, Koza *et al.* (2000) optimise the structure of a controller for a three-lag plant with a five-second delay.

Dracopulos and Kent (1997) show the use of GP to discovery the control laws that allow performing attitude manoeuvres for a satellite or spacecraft. A stability proof for the GP derived controller was included in the study.

Shimooka and Fujimoto (1998) solved the inverted pendulum problem but instead of evolving equations to determine the direction of the bang-bang force as Koza (1992) had done, they look also for the magnitude of the force that applied to the cart can move it to a target position while keeping the pendulum standing.

In his master's thesis, Ng (2000) includes an implementation in Matlab of GP. It is not easy to implement a tree structure since Matlab does not provide pointers. The author has created 3 different data structures. All of them flatten the tree into a linear string of functions. The main disadvantage is that the number of shapes that can be represented this way is very restricted. Due to the linear evaluation every internal node takes the previous node as an argument and a terminal node as the second argument (see Chapter 7 for a more detailed description and comparison with the Matlab implementation of GP used in this study).

The applications that Ng (2000) solves are all control applications: a simple Lyapunov solver, a discrete Lyapunov solver, a simple optimal control problem and a model reference adaptive system. He does not optimise structures but parameters (matrices in the Lyapunov case, an input vector for the optimal problem and parameters for the adaptive system).

2.4. SUMMARY

The chapter has presented a review of some of the literature available regarding the control methodologies that have been considered in this thesis (i.e. PID, Pole Placement, Sliding Mode and H_∞) together with the optimisation techniques analysed (i.e. Genetic Algorithms, Structured Genetic Algorithm and Genetic Programming).

The existent applications of these optimisation techniques in the field of automatic control have been emphasised, especially the control of marine vessels.

This chapter only intends to be a brief review of the available literature. In the thesis there are further references to the literature when required and a more detailed discussion.

CHAPTER 3

SUPPLY SHIP APPLICATION

3.1. INTRODUCTION

Traditionally, the steering of a vessel has required the ability of a highly skilled helmsman to sail the boat along the desired path. As well as keeping the right heading the pilot has been in charge of sailing at the desired speed. Therefore, while sailing a boat there are two coupled tasks to perform: getting the ship to navigate in the desired direction (heading control) at the desired speed (propulsion control).

The invention of the gyrocompass and the gyropilot, at the beginning of the 20th century marked the start of the automatic steering of ships [Skjetne (2003)]. Many factors have led to the increase of the shipping automation. From an economic point of view automatic control allows reduced crews (i.e. reduced costs) and it also permits near optimal operation of boats (reducing actuators usage and fuel consumption) [Zuidweg (1970)]. Furthermore, the increasing traffic densities and the increasing size of some vessels (e.g. oil tankers) makes them difficult to handle manually [Norrbin (1970)]. In addition to these economic and safety issues, military operations require high standards of effectiveness.

In the last decades, in order to keep up with the demand for oil, companies have started to drill for oil offshore. Oil platform supply ships are offshore service vessels employed in carrying supplies to drilling units of sub sea oil and gas. Their cargo consists of the equipment, food and water that an offshore platform needs to keep its production going and, therefore, these vessels are essential to the operations of the offshore oil and gas industry. Such supply boats need to be robust against environmental disturbances and manoeuvrable. They have to be able to handle adverse weather conditions (especially the ships for Northern latitudes, e.g. Atlantic Canada and the North Sea) and keep a position as steady as possible while unloading operations are carried out. Their importance for the smooth operation of oil platforms imposes the need for their navigation system to be accurate and robust against environmental disturbances. This only can be achieved through automatic controllers.

The particular applications used in this research are two scale models of an oil platform supply ship called *CyberShip I* (CS1) [McGookin (1997), Strand (1999)] and *CyberShip II* (CS2) [Lindegaard and Fossen (2002)]. Both are test vehicles developed in the Department of Engineering Cybernetics at the Norwegian University of Science and Technology (NTNU) in Trondheim. CS2 is an upgraded version of CS1 and the main difference between them lies in the actuators they are equipped with and their size. CS1 is slightly smaller and lighter and is actuated by four pod thrusters two at the stern and two at the bow. On the other hand CS2 is equipped with two propellers with rudders placed at the stern and a small tunnel thruster capable to provide a sway force in the bow.

Mathematical models of CS1 [Strand (1999)] and CS2 [Lindegaard (2003)] have been used for the design of automatic controllers of the boats and their *a posteriori* optimisation using GAs. This work has been carried out through computer simulations in Matlab. CS1 was used for a preliminary comparison study to find the GA scheme best suited to the problem of the optimisation of the controllers' parameters. Then, the resulting GA was applied to the optimisation of the parameters of the controllers designed for CS2. The similarity between both models provides confidence in the suitability of the resulting GA for the optimisation of the controllers of CS2. In addition, GP structural optimisation of the control strategy for the navigation and propulsion dynamics of CS2 has been conducted. All the optimised controllers have been implemented on the actual physical model of CS2 in the Marie Curie Training Site facilities at NTNU.

The development of the mathematical models and analysis of the ship dynamics are performed in this chapter. Section 3.2 deals with the description of the physical scale model and the facilities at the Marie Curie Training Site where the trials have been performed. Section 3.3 describes the mathematical models of CS1 and CS2, outlining the differences in the ship and actuator dynamics of both vessels. Section 3.4 is dedicated to the description of the environmental disturbance model that has been applied to the plant simulation to check its robustness to external disturbances. Section 3.5 presents the manoeuvres used for the testing and optimisation of the control designs throughout this work. Finally, Section 3.6 summarises the chapter.

3.2. SCALE MODEL AND MARIE CURIE TRAINING SITE FACILITIES

The physical model used in the experimental part of this work was CS2. As has already been mentioned, CS2 is a scale model (scale 1/70th aprox.) of an oil platform supply ship and it was developed at the Department of Engineering Cybernetics at the Norwegian University of Science and Technology in Trondheim (Norway). The testing of the optimised controllers in the real model has been performed in the Marine Cybernetics Lab (MCLab) at the NTNU. The MCLab is an experimental laboratory for testing of ships and underwater vehicles and has been appointed as a Marie Curie Training Site since 2002.

The length of CS2 is of 1.255m and its mass is 23.8kg. It is actuated by means of a tunnel-thruster placed at the bow and two main propellers with rudders situated at the stern (see Figure A.2 in the Section A.3 of Appendix A). The following Figure 3.1 shows different aspects of CS2 such as the stern actuators in the right bottom corner and the general aspect of the model.

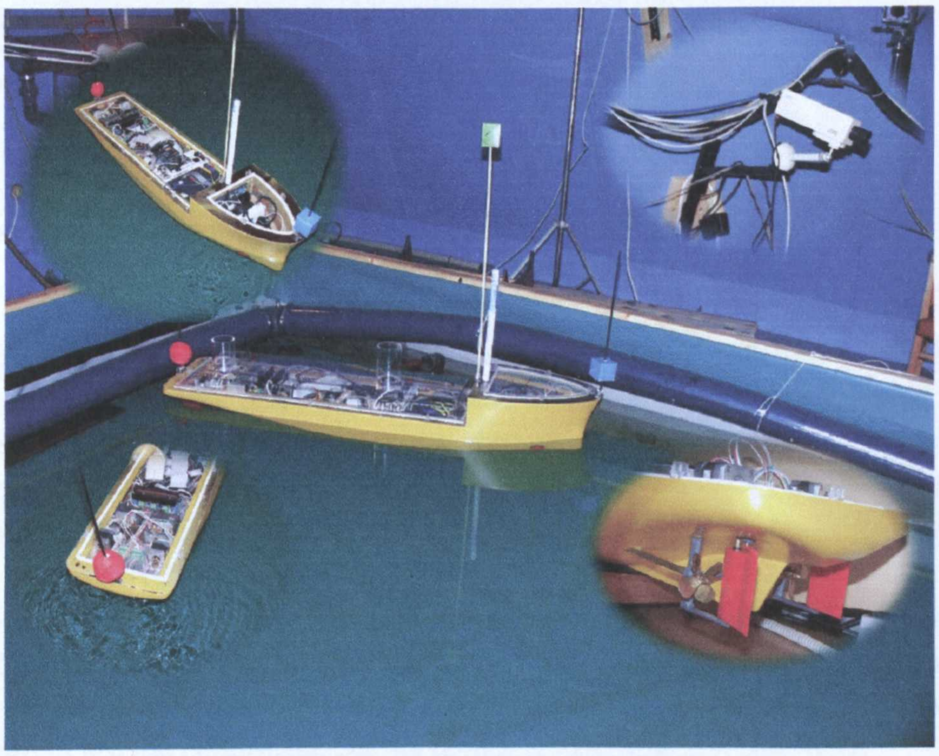


Figure 3.1: Various Aspects of CS2
Reproduced with permission of Prof. T.I. Fossen from
http://www.itk.ntnu.no/ansatte/Fossen_Thor

For position measurement purposes, CS2 is fitted with three 3-dimensional emitters that do not appear in the previous figure but that there are marked in Figure 3.2 with a yellow line. The infrared signals emitted are detected by 4 motion capture PC/cameras that provide the measurements of the (x, y) coordinates plus the heading angle to the user. They can be seen in both right and left upper corners in Figure 3.2.

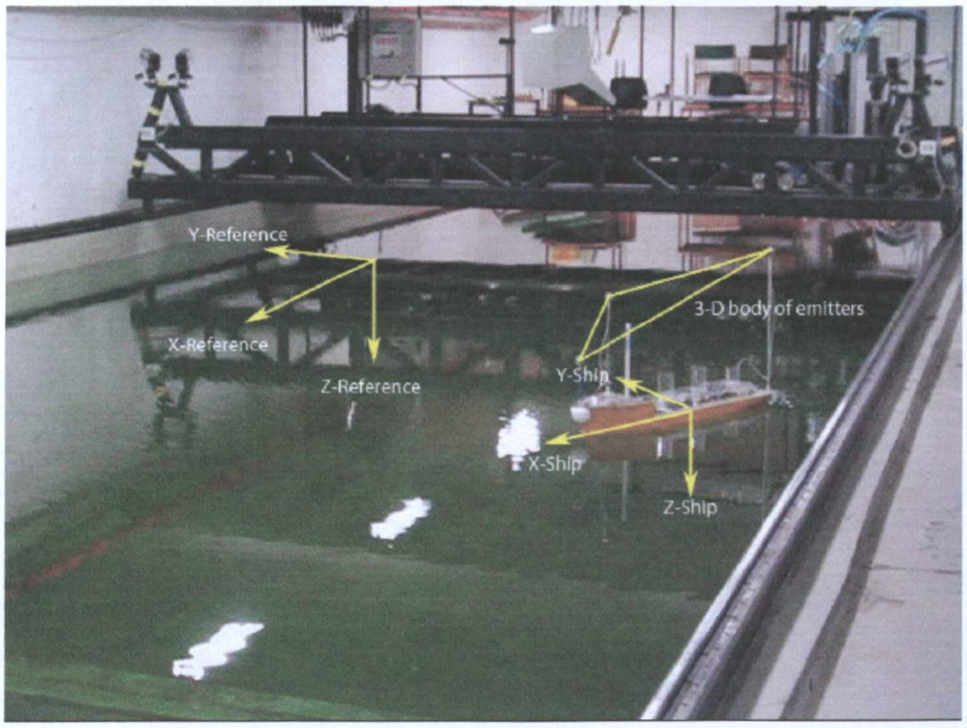


Figure 3.2: Positioning System in the MCLab

CS2 is equipped with an onboard PC (as shown in Figure 3.3) running QNX real time operating system. The control is done in real-time by an onshore PC. The connection between both PCs is made through a wireless Ethernet link and an automatic C-code generator. Matlab Simulink and Real Time Workshop are coupled with a graphical user interface in LabView for real-time presentation of the results.

The software in the onshore PC allows three types of control: manual (where the user types in the settings of the propellers and rudders via the LabView screen), using a joystick and, finally, using an automatic controller. This last option was the one used to test the performance of the optimised controllers in this study.

For more accurate information about the Marine Cybernetics Lab and CS2 refer to Lindegaard and Fossen (2002), Corneliussen (2003) and Sveen (2003).

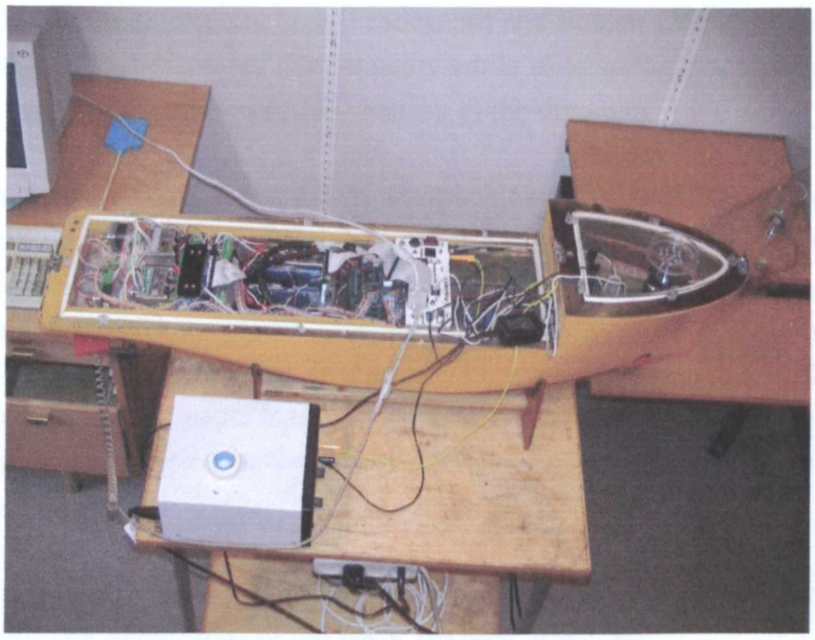


Figure 3.3: On-Board PC in CS2

Below, Figure 3.4 shows the SimuLink block diagram for the control of CS2.

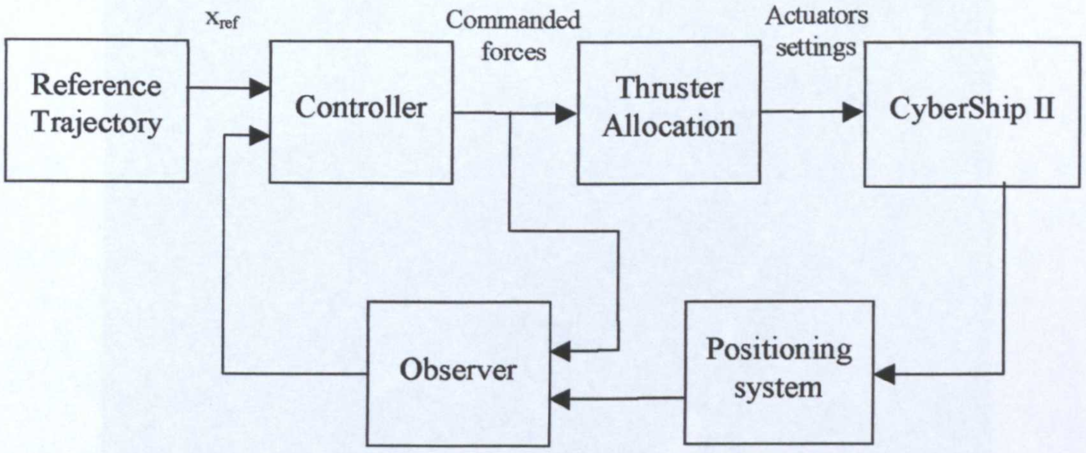


Figure 3.4: Simulink Block Diagram of CS2 Software Control

In order to create the trajectory reference, the user introduces a set of waypoints. The reference trajectory block designed by Corneliussen (2003) provides a smooth path along them using a 7th order polynomial interpolation. The reference speed is set by means of a dial in the Labview screen.

Once the controller produces the commanded forces vector, based on the state errors, these forces and moment have to be allocated to the actuators. This is done in the Thruster Allocation block. This block calculates the values for propeller speed and rudder deflection in an optimal way [Lindegaard and Fossen (2002)]. The algorithm optimises the fuel efficiency of the actuators while eliminating discontinuities in the signals that would produce excessive wearing in the thrusters.

The movement of CS2 is measured by the positioning system. The observer block estimates the state vector (consisting of the x and y coordinates, heading angle, surge and sway speed and yaw rate) using the measurements from the positioning system and the commanded forces produced by the controller [Sveen (2003)]. The observer block also includes a wave filter. In the time between the two visits to the MCLab facilities improvements were made in the observer and wave filter blocks. These improvements are reflected in the results obtained because of the reduced effect of the wave noise.

The picture below (Figure 3.5) shows the water tank in the MCLab seen from the control room. The dimensions of the basin are 40x6.45x1.5m.

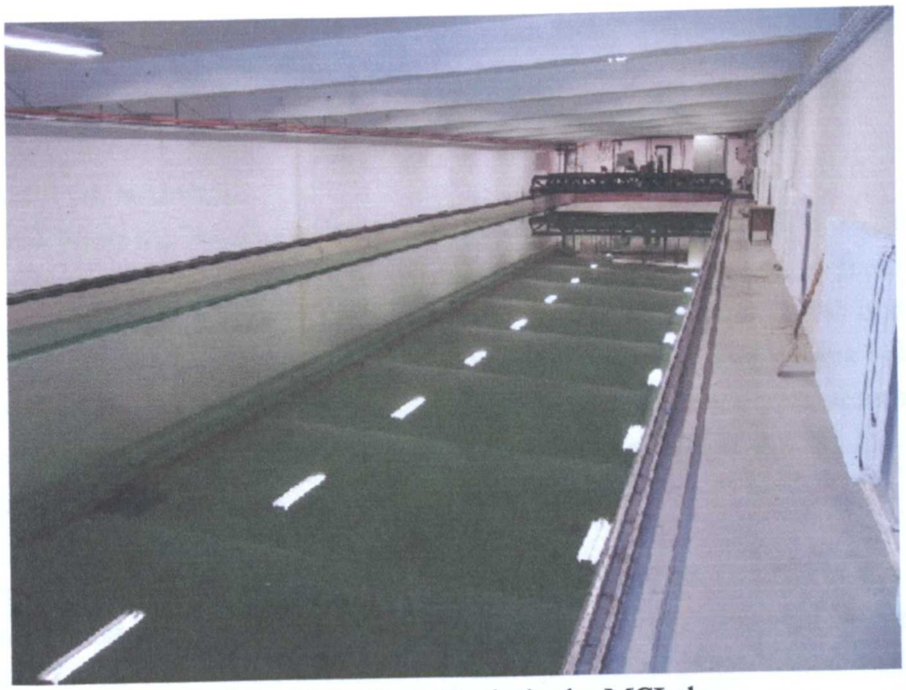


Figure 3.5: Water Basin in the MCLab
Reproduced with permission of Prof. T.I. Fossen from
http://www.itk.ntnu.no/ansatte/Fossen_Thor

The facility is also equipped with a wave generator, shown in Figure 3.6. Such a generator consists of a single flap controlled by a wave synthesiser and can produce regular and irregular waves with various spectra and wave height. The wave generator was used in the experimental trials in this study.

Figure 3.6: Wave Generator in the MCLab
Reproduced with permission of Prof. T.I. Fossen from
http://www.itk.ntnu.no/ansatte/Fossen_Thor

3.3. MATHEMATICAL MODEL

3.3.1. SHIP DYNAMICS

When analysing the motion of a vessel it is convenient to define two coordinate frames [Fossen (1994)]. The moving coordinate frame X_B Y_B Z_B is fixed to the vehicle and is called the body-fixed reference frame. The origin of the body-fixed frame is usually chosen to coincide with the centre of gravity. For marine vehicles it is usually assumed that the accelerations of a point on the surface of the Earth can be neglected. As a result of it, an earth-fixed reference frame X_E Y_E Z_E can be considered to be inertial.

Position and orientation of the vehicle should be described relative to the inertial reference frame while the linear and angular velocities of the vehicle should be expressed in the body-fixed coordinate system [Fossen (1994)]. Based on this, the general motion of the vessel can be described by the following vectors:

$$\begin{aligned} \boldsymbol{\eta} &= [x, y, \psi]^T \\ \boldsymbol{\nu} &= [u, v, r]^T \\ \boldsymbol{\tau} &= [\tau_1, \tau_2, \tau_3]^T \end{aligned} \tag{3.1}$$

Here η denotes the position and orientation vector with coordinates in the earth-fixed frame, ν denotes the linear and angular velocity vector with coordinates in the body-fixed frame and τ is used to describe the input force vector in the body-fixed frame.

A non-linear hydrodynamic model based on the kinetic and kinematic equations represents the dynamics of the vessel.

The kinetic equations are represented by the following matrix equation [Fossen (1994)]:

$$\mathbf{M}\cdot\dot{\mathbf{v}}+\mathbf{C}(\mathbf{v})\cdot\mathbf{v}+\mathbf{D}\cdot\mathbf{v}=\boldsymbol{\tau} \tag{3.2}$$

Here **M** is the mass/inertia matrix, **C** is the Coriolis matrix and **D** is the damping matrix. Also, $\nu = [u, v, r]^T$ is the body-fixed linear and angular velocity vector and $\boldsymbol{\tau} = [\tau_1, \tau_2, \tau_3]^T$ is the input force vector, given that τ_1, τ_2 and τ_3 are the forces and torque along the X, Y and Z-axis, respectively.

The kinematic equation relates the body-fixed reference frame to the earth-fixed reference frame. The vessel's path relative to the earth-fixed coordinate system is given by a velocity transformation [Fossen (1994)]:

$$\dot{\boldsymbol{\eta}}=\mathbf{J}(\boldsymbol{\eta})\cdot\mathbf{v} \tag{3.3}$$

where **J** is the Euler matrix and $\eta = [x, y, \psi]^T$ denotes the position and orientation vector with coordinates in the earth-fixed frame.

When kinetic and kinematic equations are combined together the following matrix form is produced (assuming **M** to be invertible):

$$\begin{bmatrix}\dot{\mathbf{v}}\\ \dot{\boldsymbol{\eta}}\end{bmatrix}=\begin{bmatrix}-\mathbf{M}^{-1}(\mathbf{C}(\mathbf{v})+\mathbf{D}) & \mathbf{0}\\ \mathbf{J}(\boldsymbol{\eta}) & \mathbf{0}\end{bmatrix}\cdot\begin{bmatrix}\mathbf{v}\\ \boldsymbol{\eta}\end{bmatrix}+\begin{bmatrix}\mathbf{M}^{-1}\\ \mathbf{0}\end{bmatrix}\cdot\boldsymbol{\tau} \tag{3.4}$$

This expression corresponds to a non-linear state space equation:

$$\dot{\mathbf{x}}=\mathbf{A}(\mathbf{x})\cdot\mathbf{x}+\mathbf{B}\cdot\boldsymbol{\tau} \tag{3.5}$$

Thus, the corresponding states and inputs for this model are shown in Table 3.1 [McGookin (1997)].

Table 3.1: Supply Ship Model States and Inputs

STATES		INPUTS	
u	surge velocity	τ_1	surge thrust force
v	sway velocity	τ_2	sway thrust force
r	yaw rate	τ_3	yaw thrust torque
ψ	yaw (heading) angle		
x_p	x-position on earth		
y_p	y-position on earth		

All the numerical data for the CS1 and CS2 model matrices can be found in Appendix A. Differences in the mathematical models of the ship dynamics are small. CS2 is slightly longer and heavier.

3.3.2. ACTUATORS DYNAMICS

As mentioned previously, the main differences between CS1 and CS2 lie in the actuators they are equipped with. In this section the actuators dynamics of each of them are described.

CyberShip I

In the case of CS1, four pod thrusters provide the forces and torque relative to the body-fixed axis that constitute the inputs to the vessel model. Two of them are placed at the stern, symmetric about the body-fixed X-axis, while the other two are placed at the bow, along the body-fixed X-axis [McGookin (1997)]. A diagram of the actuators layout can be found in Appendix A (Figure A.1 in Section A.2).

Each thruster is represented by the force (f_i) it produces and the angle (α_i) defining its direction. Thrusters at the stern operate always in the same direction, so α_1 and α_2 are always equal and with the same power. The amplitude operating limit values for the thrusters forces are shown in Table 3.2 [McGookin (1997)].

Table 3.2: CS1 Thrusters Operating Limits

f_i (N)	α_i (rad)
±0.9	$\pm\pi$

The three input force components (τ) to the individual forces produce by the thrusters (f_i) using trigonometric relationships [Fossen (1994), Strand (1999)]. From these relationships the maximum possible values of the surge thrust force (τ_1), sway thrust force (τ_2) and yaw thrust torque (τ_3) can be obtained.

CyberShip II

On the other hand, CS2 is equipped with 3 actuators: two propellers with rudders placed at the stern, symmetric about the body-fixed X-axis, and a small tunnel thruster capable to provide a sway force in the bow. The three propellers are controlled by fixing the number of revolutions per minute [Lindegaard and Fossen (2002)].

The amplitude operating limit values for the actuators in CS2 are shown in the following Table 3.3.

Table 3.3: CS2 Actuators Operating Limits

Stern rudders deflection ($^{\circ}$)	± 35
Bow thrusters speed (rps)	± 80
Stern propellers speed (rps)	± 35

In both CS1and CS2 the forces and torque produced by the actuators are states of the system and are included in the plant state space equation as follows:

$$\dot{\boldsymbol{\tau}} = \begin{bmatrix} -1/T_1 & 0 & 0 \\ 0 & -1/T_2 & 0 \\ 0 & 0 & -1/T_3 \end{bmatrix} \boldsymbol{\tau} + \begin{bmatrix} 1/T_1 & 0 & 0 \\ 0 & 1/T_2 & 0 \\ 0 & 0 & 1/T_3 \end{bmatrix} \boldsymbol{\tau}_{\mathbf{com}} \qquad (3.6)$$

where τ_{com} is the vector of the commanded inputs. The values T_1, T_2 and T_3 are time constants (see Appendix A for values) that allow us to restrict the rate of change of the actual forces provided for the thrusters.

Hence, both ship models have 9 states (i.e. u, surge velocity; v, sway velocity; r, yaw rate; ψ, yaw angle; x_p, x-position on earth, y_p, y-position on earth, τ_1, τ_2 and τ_3, forces and torque along the body fixed X, Y and Z-axis respectively) and 3 inputs (i.e. τ_{1com}, τ_{2com} and τ_{3com}, commanded forces and torque along the body fixed X, Y and Z-axis respectively) [McGookin (1997)].

3.4. DISTURBANCES MODEL

In order to evaluate the robustness against environmental disturbances of the controllers obtained through the GA optimisation, simulations of manoeuvres in the presence of environmental disturbances have been carried out [Fossen (1994)].

There are three main types of environmental disturbances: wind-generated waves, ocean currents and wind. However, in this research the analysis has been restricted to the disturbance considered to be the most relevant for surface vessels, i.e. wind-generated waves. In addition they can be reproduced in the MCLab for testing.

The model that has been used to simulate the wave's action on the vessel derives the forces and moments induced by a regular sea on a block-shaped ship and it is described in Zuidweg (1970). It forms a vector called $\boldsymbol{\tau}_{\mathrm{waves}}$ that is directly added to the input vector, $\boldsymbol{\tau}$, in Equation (3.4) using the principle of superposition. $\boldsymbol{\tau}_{\mathrm{waves}}$ has 3 components:

$$\begin{aligned} X_{wave}(t) &= \sum_{i=1}^{N} \rho g B L T \cos(\beta - \psi) s_i(t) \\ Y_{wave}(t) &= \sum_{i=1}^{N} -\rho g B L T \sin(\beta - \psi) s_i(t) \\ N_{wave}(t) &= \sum_{i=1}^{N} \frac{1}{24} \rho g B L (L^2 - B^2) \sin 2(\beta - \psi) s_i^2(t) \end{aligned} \qquad (3.7)$$

Here L, B and T are the length, breadth and draft of the wetted part of a ship, considering it as a parallelepiped. ρ is the density of the water, $s_i(t)$, the wave slope and $(\beta - \psi)$, the angle between the heading of the ship and the direction of the wave (in radians).

The derivation of the Equations (3.7) is based on the assumption of forces and moments being the result of the water pressure acting on the wet surface of a block-shaped ship with dimensions L, B and T. This assumption holds better for big vessel wit flat hulls such oil tankers. In our case, a block coefficient factor has been applied to the forces and torque amplitude to take into account the fact that the ship's hull is not flat.

In order to implement the above formulas, the wave slope, s_i, has to be computed. The wave slope, s_i of a long crested irregular sea propagating along the positive x-axis can be written as a sum of wave components. So, assuming that $x=0$ and that higher order terms can be neglected, the wave slope, s_i can be related to the wave frequency spectrum in the following way [Fossen (1994)]:

$$s_i(t) = A_i k_i \sin(\omega_{ei} t + \Phi_i) \tag{3.8}$$

where A_i is the wave amplitude of a wave component i, k_i is the wave number of a wave component i (i.e. $k_i = 2\pi/\lambda_i = \omega_i^2/g$) and ω_{ei} is the frequency of encounter.

The wave amplitude A_i of wave component i is related to the wave spectral density function $S(\omega_i)$ as:

$$A_i^2 = 2S(\omega_i)\Delta\omega \tag{3.9}$$

Here ω_i is the wave frequency of wave component i and $\Delta\omega$ is a constant difference between successive frequencies.

To compute $S(\omega_i)$ and thus A_i, different wave spectra can be considered. For prediction of responses of marine vehicles and offshore structures in open sea, the International Ship and Offshore Structures Congress, 2nd ISSC (1964), and the International Towing Tank Conference, 12th ITTC (1969) and 15th ITTC (1975) have recommended the use of a modified version of the Pierson-Moskowitz spectrum, that is [Fossen (1994)]:

$$S(\omega) = \frac{4\pi^3 H_s^2}{(0.710T_o)^4 \omega^5} \exp\left(\frac{-16\pi^3}{(0.710T_o)^4 \omega^4}\right) \tag{3.10}$$

Thus, if the modal period, T_o, is chosen as a function of H_s, the significant wave height, the spectrum only depends on one parameter, H_s. H_s needs to be suitably scaled so that the resulting waves are according to the dimensions of the supply ship scale model.

The following Figure 3.7 shows the modified version of the Pierson-Moskowitz spectrum for wave heights of 2 (solid line), 3 (dotted line) and 4 (dashed line) meters.

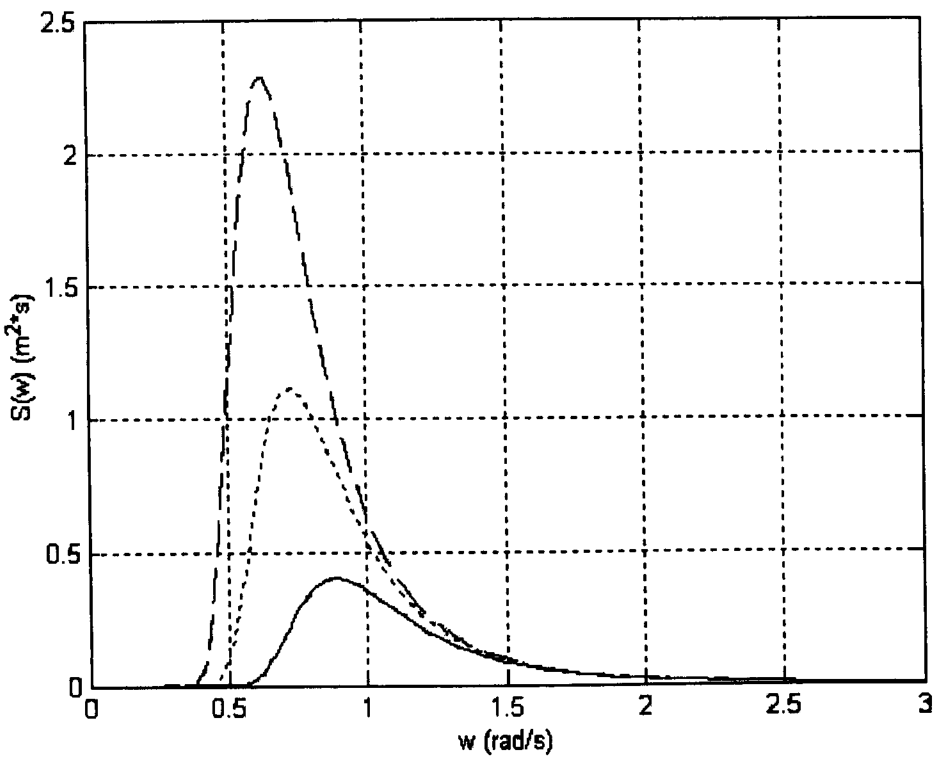


Figure 3.7: PM-Spectrum for Different Values of H_s
(solid line: 2m, dotted line: 3m, dashed line: 4m)

Therefore, the algorithm used to compute the forces and moments induced on the ship is as follows [Fossen (1994)]:

(1) The spectral density function $S(\omega)$ for the chosen wave height is divided into N intervals with length $\Delta\omega$

(2) A random frequency ω_i is chosen in each of the frequency intervals and $S(\omega_i)$ is computed.

(3) The wave amplitude (A_i) is computed for $i=1..N$ using Equation (3.9)

(4) Wave slope (s_i) is calculated by applying Equation (3.8)

(5) The forces and moments induced are calculated applying Equations (3.7). A block coefficient factor is applied to take into account the fact that the ship's hull is not flat.

The following Figure 3.8 shows the profiles of the forces and torque created over the vessel by 3m high wave disturbances (suitably scaled down in a 1/70th factor) with a constant angle of attack of 135° while the vessel keeps a straight trajectory and speeds up from 0 to 0.7 m/s.

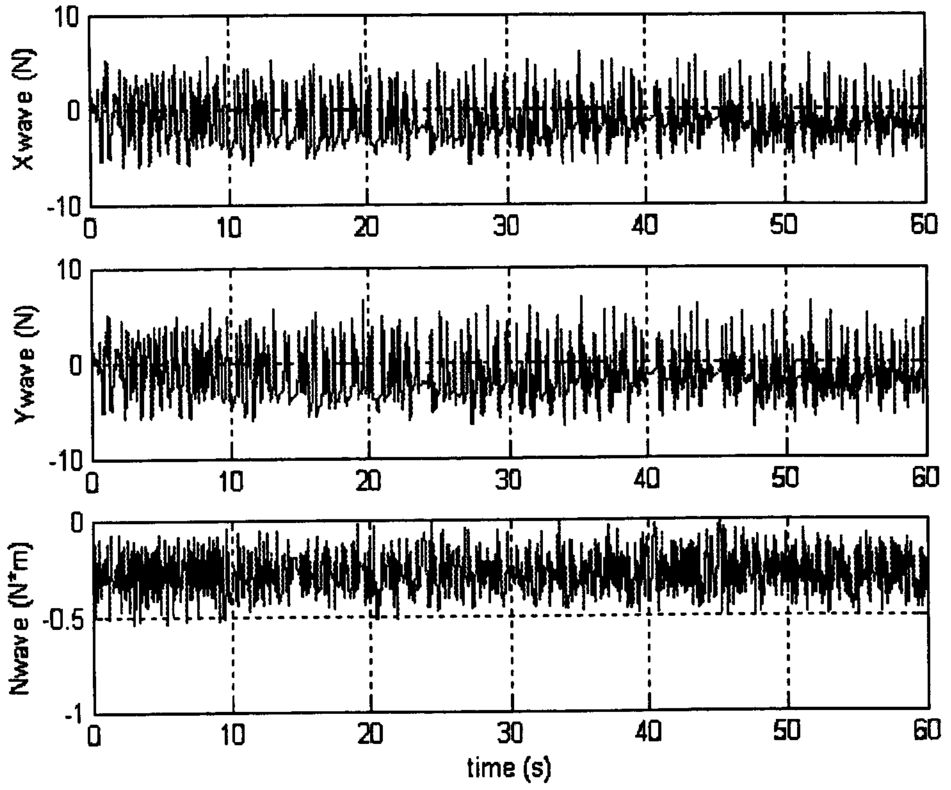


Figure 3.8: Waves Forces and Torque Generated by 3m High Waves with an Angle of Encounter of 135°

3.5. MANOEUVRES

The desired responses for propulsion and heading used throughout this study are two critically damped steps up and down as shown in the following Figure 3.9. The heading reference is a 45° double step manoeuvre. The reference for the surge speed makes the vessel accelerate up to 0.7 m/s and then decelerate back to rest. Since the model scale is 1/70th, 0.7 m/s would be equivalent to 6 m/s approximately in the full-scale vessel. These reference signals have been used for the GA and GP optimisation as well as for the manual tuning of the controllers.

The controller results have been tested in the water basin executing the zig-zag manoeuvre shown in Figure 3.10. This path is based on 5 waypoints, the starting point (7,1) and then (5,1), (3,-1), (1,1) and (-1,0). Those waypoints have been chosen to reproduce the 45° turning manoeuvres used in the simulation work. An algorithm generated the path and desired heading and surge signals based on these waypoints.

The computation method used to create the path is a 7th order polynomial interpolation [Corneliussen (2003)]. The zig-zag is tracked at a constant surge speed of 0.2m/s approximately. Given that the way of adjusting the set point for the speed is a manual dial, a great deal of accuracy is not possible.

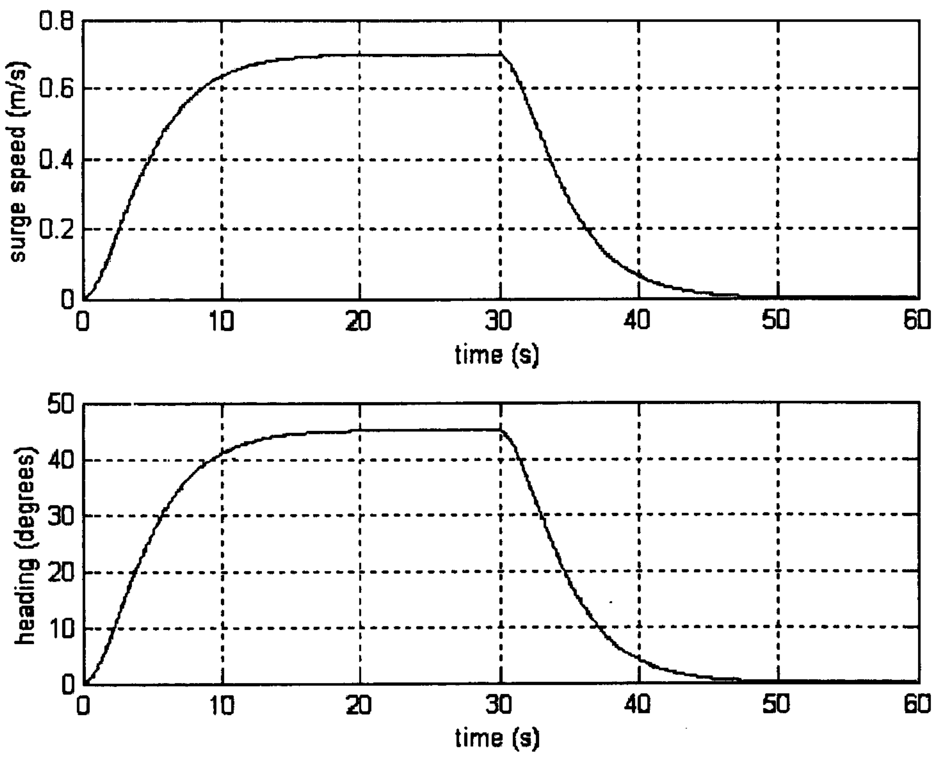


Figure 3.9: Desired Responses to Track

Also, the manoeuvre is executed while generating waves in order to study their effect. The waves synthesizer is set to generate irregular waves with a Pierson-Moscowicz spectrum [Fossen (1994)], like the one used in simulations. The significant wave height is 5 mm (scale 1/70th) and the peak period of 0.80 s. The wave height could not be set higher because the vessel model is not well isolated to handle them and there was a risk of the boat sinking. The angle of attack of the waves is 0°. This was defined by the fixed position of the flap that generated the waves in the tank.

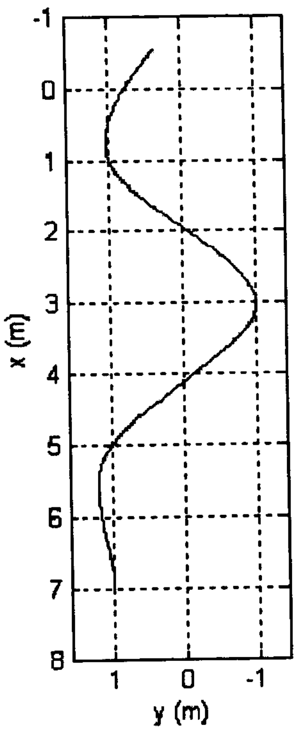


Figure 3.10: Path to Track

3.6. SUMMARY

This chapter has presented mathematical models used in the simulation part of this work. The ship and actuators dynamics of CS1 and CS2 have been described. The full model including the numerical values of both CS1 and CS2 can be found in the Appendix A.

This chapter also has included a description of the full scale vessel CS2 and the laboratory where the real trials to test the performance of the optimised controllers were conducted.

During the optimisation work disturbances have been added to the simulation to create a more realistic environment. The mathematical model of the waves used in the simulation and how they were implemented have also been described.

Finally, the last section of the chapter was dedicated to the description of the manoeuvres used in simulations as well as in the real trials throughout this work for the optimisation and testing of the performance of the control designs.

CHAPTER 4

CONTROL METHODOLOGIES FOR THE CONTROL OF A SUPPLY VESSEL

4.1. INTRODUCTION

In order to ensure the safe navigation of surface vessels their motion has to be controlled accurately. This manoeuvring control can be provided through the application of *automatic control theory* [Dorf and Bishop (2001), Dutton *et al.* (1997)]. In general, control theory provides design strategies that allow a better understanding of the system being controlled (e.g. a vessel) and a mechanism to regulate the way in which the system operates. There are various control methodologies that have their own unique structure. Despite being fundamentally different in style they perform the same task i.e. to make the system behave in a desired manner.

Since the early 1970s important research has been conducted on the subject of automatic control of marine surface vessels. Authors like Norrbin (1970), Astrom and Kallstrom (1976) or Kallstrom and Astrom (1981) have worked extensively in the identification of ship dynamics. Other literature includes papers about steering control [Kallstrom *et al.* (1979), Saelid and Jenssen (1983), Zuidweg (1970)] or dynamic positioning control [Fossen *et al.* (1996), Sorensen *et al.* (1996)]. A major text in this field is Fossen (1994).

Although the particular vessels used in this study, *Cybership I* and *II* (see Chapter 3), have also been used in other studies to investigate dynamic positioning systems [Strand (1999), Lindegaard and Fossen (2002)], in which the sway thrust force (τ_2) becomes more relevant; this research has focused in the heading and propulsion dynamics, i.e. the control system of interest reacts to heading (*course changing*) and surge velocity commands from a pilot.

In the context of this thesis, the navigation and propulsion capabilities of the test vessels are regulated using control design methodologies. Since controlling the motion of a vessel implies performing two coupled tasks, these methodologies rely on either two *single input single output* (SISO) control strategies (i.e. one controller for the propulsion subsystem, which has τ_1 as an input and the surge speed as the output, and a second controller for the heading subsystem, which has τ_3 as an input and the heading as the output), or a *multi input multi output* (MIMO) approach (i.e. the MIMO system has got three inputs, τ_1, τ_2 and τ_3, and two outputs, surge speed and heading). Prior to the design of the controllers to regulate the motion of the vessel, a study into the level of coupling between the propulsion and heading dynamics was performed through a diagnosis method called *Individual Channel Analysis and Design* (ICAD) [O'Reilly and Leithead (1991)]. ICAD is a framework for the analysis of MIMO plants. It is based on the decomposition of the system in individual channels without losing any information about the cross coupling between the channels. Hence, ICAD allows the assessment of the suitability of decoupled or MIMO controllers for a given plant.

After the ICAD analysis, this chapter presents a theoretical description of the four control methodologies that have been optimised using GAs: PID [Astrom and Hagglund (1995), Dutton *et al.* (1997)], Pole Placement [Andry *et al.* (1983), Kautsky *et al.* (1985)], Sliding Mode [Slotine and Li (1991), Utkin (1972)] and H_∞ [Glover and Doyle (1988), Zhou *et al.* (1996)]; and describes how they have been implemented for the control of the particular plant (i.e. CS1 and CS2). Each of these types of controller is very representative of a certain field of automatic control (i.e. linear control, state feedback, nonlinear control and optimal control). So they offer a good variety of control structures. Thus, given the variety in the number of parameters to optimise and the controller structures, the GA is tested in different control optimisation problems with different search spaces. Moreover, the comparison of the performance of these controllers provides us a good understanding of the advantages and drawbacks of each controller structure and a good starting point for the Genetic Programming, since the GP uses these structural bits to build (and optimise) the controller structure.

As well as describing the theoretical background of each control strategy, a description of their practical implementation for CS1 and CS2 together with the results obtained in the manual tuning of each controller for both vessels are included in this chapter. The manually tuned controllers for CS2 have also been implemented in the real plant and the results are shown.

Chapter 4 describes the control methods as follows: Section 4.2 is a study of the level of coupling of the model using ICAD as a diagnosis aid, Section 4.3 describes linear PID control, Section 4.4, Pole Placement control and, Section 4.5 and Section 4.6 describe Sliding Mode and H_∞ control respectively. Finally, Section 4.7 summarises the chapter findings.

4.2. DECOUPLED SYSTEM

In this work various controllers have been design to control the heading (ψ) and surge speed (u) of the plant. Some of them are based on a *multi input multi output* (MIMO) design, 3-inputs (τ_{1com}, τ_{2com} and τ_{3com}) 2-outputs (surge and heading), while others perform on two decoupled *single input single output* (SISO) systems. The decoupling of the system consists of the decomposition of the system into SISO subsystems where a single input governs each motion (i.e. surge speed or heading). It is a "divide and conquer" design approach. The decoupling of the system simplifies considerably the controller design and it is a widely used method for MIMO systems [Franklin *et al.* (1994), McGookin (1997)]. However, if the interactions between states of the system are too strong, SISO control can become unacceptable.

In order to analyse the level of coupling between the surge speed, sway speed and heading states, an analysis tool in the form of Individual Channel Analysis and Design (ICAD) [Leithead and O'Reilly (1992), O'Reilly and Leithead (1991)] has been used.

ICAD is a framework for the analysis and design of feedback controllers for MIMO plants. ICAD is based on the decomposition of MIMO systems into SISO individual channels with no loss of information (i.e. coupling within the system is preserved in the design of the individual channels). In the case of *two input two output* (TITO) systems, they can be decomposed into two SISO channels.

The method defines the *multivariable structure function* $\gamma(s)$. The structure function describes the interaction between channels, i.e., the cross coupling within the plant. When the magnitude of $\gamma(s)$ is much smaller than 1, channel interaction is low; otherwise channel interaction is high [Leithead and O'Reilly (1992)].

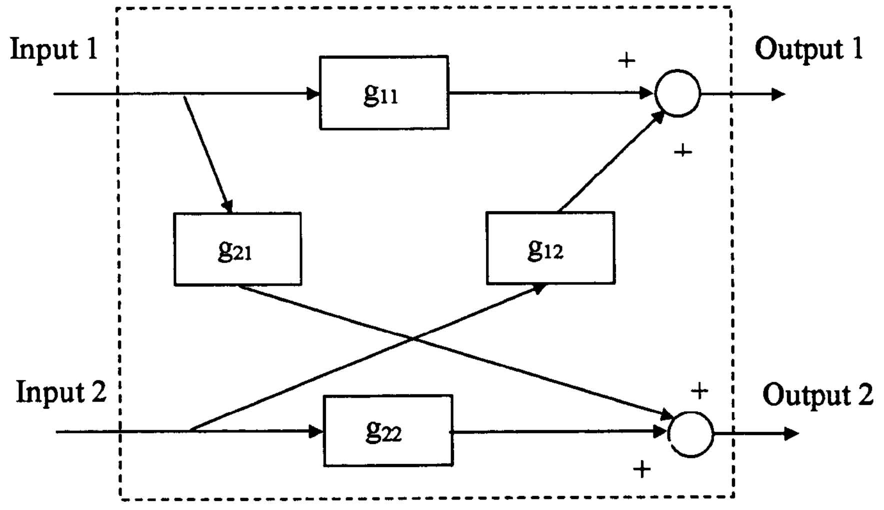


Figure 4.1: Transfer Function Expression of a TITO Plant

Figure 4.1 describes the transfer function diagram of a linear TITO system. Thus, a linear TITO plant can be represented as a 2x2 transfer function matrix as follows:

$$\mathbf{G}(s) = \begin{bmatrix} g_{11}(s) & g_{12}(s) \\ g_{21}(s) & g_{22}(s) \end{bmatrix} \qquad (4.1)$$

Then $\gamma(s)$ is defined:

$$\gamma(s) = \frac{g_{12}(s)g_{21}(s)}{g_{11}(s)g_{22}(s)} \qquad (4.2)$$

In the particular case of CS1 and CS2, three clear channels can be defined. The first one has got the force in the X-direction (τ_1) as the input and the surge speed (u) as the output (i.e. channel (u, τ_1)). The second channel has got the force in the Y-direction (τ_2) as an input and the sway speed (v) as the output (i.e. channel (v, τ_2)). Finally the third channel has got the torque (τ_3) as the input and the heading angle (ψ) as the output (i.e. (ψ, τ_3)). The coupling between the channels (u, τ_1) - (ψ, τ_3) and (v, τ_2) – (ψ, τ_3) has been analysed using $\gamma(s)$. For that purpose, the state-space equations of CS1 and CS2 (of the form shown in Equation (3.4)) have been linearised using a Taylor series expansion [Dutton *et al.* (1997)] around an operating point and truncating the Taylor series after the first partial derivatives. The transfer function matrix can be calculated just by applying [Dutton *et al.* (1997)]:

$$\mathbf{G}(s) = \mathbf{C}\cdot(s\mathbf{I} - \mathbf{A})^{-1}\cdot\mathbf{B} + \mathbf{D} \qquad (4.3)$$

Here **A** is the system matrix, **B** is the input matrix, **C** is the output matrix and **D** is the feed forward matrix of the state-space model of the system as defined in Equation (3.5) and **I** is the identity matrix. The numerical values of $g_{11}(s)$, $g_{12}(s)$, $g_{21}(s)$ and $g_{22}(s)$ of the transfer function matrix as defined in Equation (4.1) together with the value of the $\gamma(s)$ as defined in Equation (4.2) obtained while analysing the coupling between the channels (u, τ_1) - (ψ, τ_3) and (v, τ_2) – (ψ, τ_3) for CS1 and CS2 can be found in the Appendix A.

CyberShip I

The results of $\gamma(s)$ show that the interaction between the channels (u, τ_1) - (ψ, τ_3) is basically null, since in the linearised model $g_{12} = g_{21} = 0$. The entire cross coupling effect is lost during the linearisation of the matrices.

Figure 4.2 shows the Bode magnitude response of the multivariable structure function $\gamma(s)$ of the TITO system consisting of 2-inputs (τ_2 and τ_3) and 2-outputs (v and ψ).

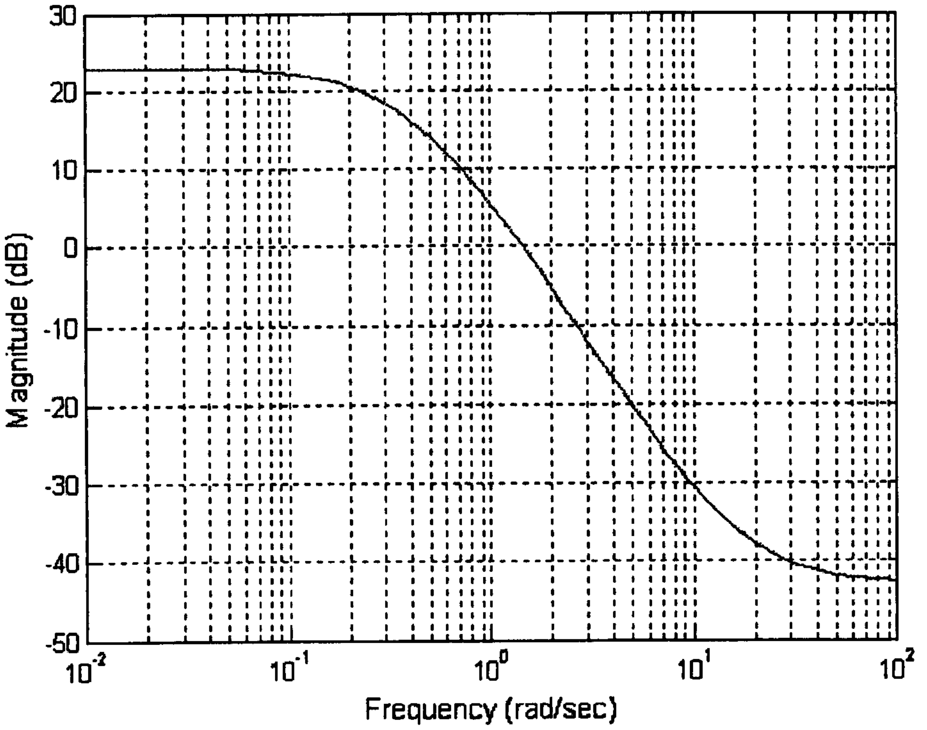


Figure 4.2: Bode Magnitude Response of $\gamma(s)$ for CS1

The magnitude response of $\gamma(s)$ for the channels (v, τ_2) – (ψ, τ_3) is well over 1. This indicates a big interaction between both channels, which has to be taken into account when decoupling the system.

CyberShip II

Expectedly, the results of the ICAD analysis for CS2 are very similar. The value of $\gamma(s)$ obtained in the study of the interaction between the (u, τ_1) and (ψ, τ_3) channels is zero due to g_{12} and g_{21} being zero in the linearised model. Once more the cross coupling effect between surge speed and heading is lost during the linearisation of the matrices.

The value of $\gamma(s)$ when analysing the interaction between the (u, τ_1) and (ψ, τ_3) channels, it has been plotted in Figure 4.3.

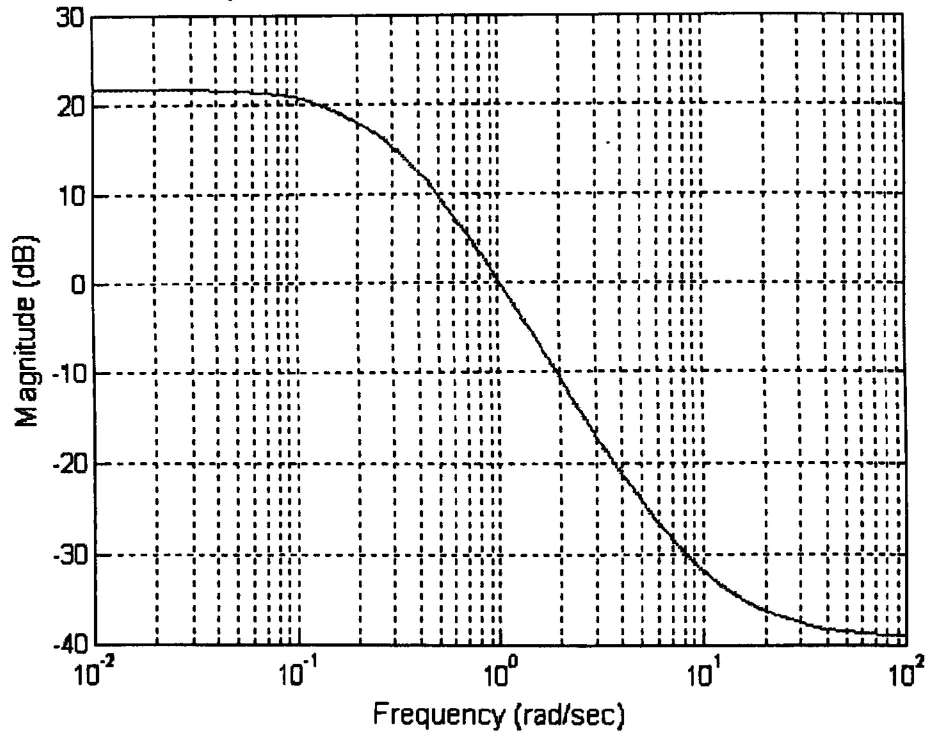


Figure 4.3: Bode Magnitude Response of $\gamma(s)$ for CS2

The result is equivalent to that of CS1 and points out the big coupling between the heading and sway speed channels.

Summarising, the ICAD analysis has highlighted the importance of the coupling between the heading and sway speed states. Regarding the interaction between surge speed and heading, the ICAD analysis shows it to be null in the linearised model, but it does not take into account the possible coupling due to the nonlinearities. Thus this result must be taken into perspective. Both models CS1 and CS2 have provided very similar results in the ICAD analysis, as it was expected.

In the next sections some controllers have been design on the decoupled system while others perform on the MIMO model. For those using the decoupled system the two SISO systems to control are as follows: since the study about the decoupling of the system shows the strong cross coupling between v (sway speed) and ψ (heading), the heading subsystem consists of 3 states ($x_H = [v,\ \psi,\ r]$) and one input ($u_H = \tau_3$). Meanwhile the propulsion subsystem consists of a single state ($x_P = u$) and one input ($u_P = \tau_1$). Hence, the state space equations that represent the decoupled system to be controlled are [Alfaro-Cid *et al.* (2001a), McGookin (1997)]:

$$\dot{u} = \mathrm{A_p} \cdot u + \mathrm{B_p} \cdot \tau_1 \tag{4.4}$$

$$\begin{bmatrix} \dot{v} \\ \dot{r} \\ \dot{\psi} \end{bmatrix} = \mathbf{A_h}(v, r, \psi) \cdot \begin{bmatrix} v \\ r \\ \psi \end{bmatrix} + \mathbf{B_h} \cdot \tau_3 \tag{4.5}$$

Here u and τ_1 are the state and input of the propulsion subsystem while v, r and ψ are the states of the heading subsystem and τ_3 the input. $\mathrm{A_p}$, $\mathbf{A_h}$, $\mathrm{B_p}$ and $\mathbf{B_h}$ are the partitions of the state-space matrices of the model (see Equation (3.5)) corresponding to the propulsion and heading subsystems respectively.

4.3. PID LINEAR CONTROL

4.3.1. THEORETICAL BACKGROUND

PID control [Astrom and Hagglund (1995), Dutton *et al.* (1997)] is a linear control methodology. The structure of PID controllers is very simple. They operate on the error signal, which is the difference between the desired output and the actual output, and generate the actuation signal that drives the plant. They have three basic terms: *proportional action*, in which the actuation signal is proportional to the error signal, *integral action*, where the actuation signal is proportional to the time integral of the error signal, and *derivative action*, where the actuation signal is proportional to the time derivative of the error signal. The basic structure of a SISO PID controller is shown in the following Figure 4.4

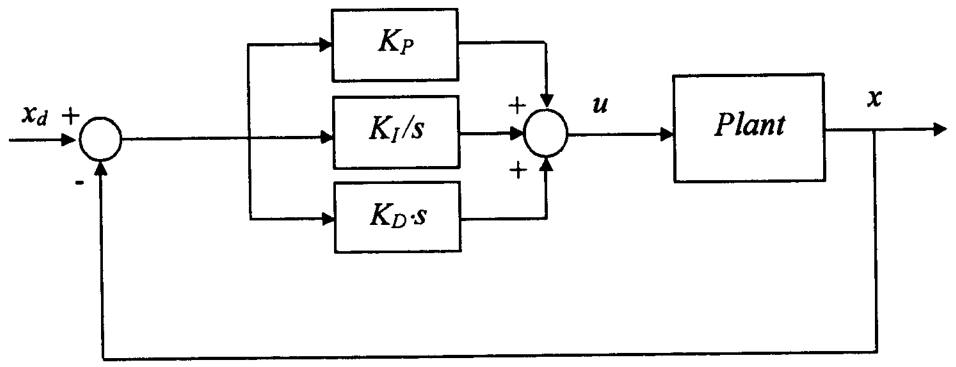


Figure 4.4: Structure of a SISO PID Controller

Here x_d is the reference signal to track, x is the state of the system to control and u is the control action generated by the PID controller. K_P, K_I and K_D are the gains of the proportional, integral and derivative terms.

The proportional term allows slight improvement in the steady-state error and in the rise time, but it can lead to oscillations. The inclusion of the integral term improves the permanent action since it usually eliminates the steady-state error, but in detriment of the speed of the response. Finally, the derivative term can be used to increase damping and reduce the oscillations, though it must be used with caution because it amplifies any existent noise in the signal [Dutton *et al.* (1997)].

These *three-term controllers* have been found to be reasonably effective and easy to implement. Consequently, the *PID* controller is the standard controller design in the process industries. The PID standard form is given by [Dutton *et al.* (1997)]:

$$u = K_P \cdot e + K_I \int e \cdot dt + K_D \cdot \frac{de}{dt} \qquad (4.6)$$

where e is the error signal, u is the control action and K_P, K_I and K_D are the proportional, integral and derivative gain, respectively.

To design a particular control loop, the three constants (K_P, K_I and K_D) have to be adjusted to arrive at acceptable performance. In order to get a first approach to an acceptable solution there are several controller design methods that can be applied. For example, classical control methods in the frequency domain [Hagglund and Astrom (1991)] or automatic methods like Ziegler-Nichols [Ziegler and Nichols (1942)], the most well-known of all tuning PID methodologies. Ziegler and Nichols (1942) recognised that the step response of most process control systems has a S-shaped curve called the *process reaction curve* [Franklin *et al.* (1994)] and can be generated experimentally or from dynamic simulation of the plant. The shape of the curve is characteristic of high-order systems, and the plant behaviour may be approximated by the following transfer function [Franklin *et al.* (1994)]:

$$\frac{Y(s)}{U(s)} = \frac{K \cdot e^{-t_d \cdot s}}{\tau \cdot s + 1} \tag{4.7}$$

which is simply a first-order system plus a transportation lag. The constants in the above equation can be determined from the unit step response of the process. Based on the time delay t_d and the slope of the reaction curve $R = K/\tau$, Ziegler and Nichols (1942) suggest the following PID controller parameters: $K_P = 1.2 / R{\cdot}L$, $K_I = K_P / 2L$ and $K_D = 0.5{\cdot}L{\cdot}K_P$. Although the method provides a first approximation the response produced is underdamped and needs further manual retuning [Dutton *et al.* (1997)].

4.3.2. CONTROLLER IMPLEMENTATION

It is standard practice in most existing marine systems to use a series of SISO PID controllers, one for each variable to control [Fossen (1994)]. Therefore, in this work, two classical PID controllers have been used to provide the structure for the propulsion controller (for governing surge or forward velocity) and the navigation controller (for governing heading, i.e. direction) [Alfaro-Cid *et al.* (2001a), (2001b)].

Using the PID structure defined in Equation (4.6), the resulting control actions are:

$$\tau_1 = K_{Pp}(u - u_d) + K_{Ip}\int(u - u_d)dt + K_{Dp}\frac{d(u - u_d)}{dt} \tag{4.8}$$

$$\tau_3 = K_{Ph}(\psi - \psi_d) + K_{Ih}\int(\psi - \psi_d)dt + K_{Dh}\frac{d(\psi - \psi_d)}{dt} \tag{4.9}$$

where τ_1 is the force in the X-direction, τ_3 is the torque, u is the surge speed, u_d is the desired surge speed, ψ is the heading, ψ_d is the desired heading and K_{Pp}, K_{Ip}, K_{Dp}, K_{Ph}, K_{Ih} and K_{Dh} the propulsion and heading PID gains.

To design each particular control loop the values of the three parameters (K_P, proportional gain, K_I, integral gain, and K_D, derivative gain) have to be adjusted to achieve acceptable control performance.

In this particular design anti-windup features have not been considered because wind-up has not been an issue in the implementation of the controller.

4.3.3. MANUAL TUNING RESULTS

CyberShip I

The results obtained for the manual tuning of the PID decoupled controllers for CS1 are shown in Table 4.1:

Table 4.1: Manually Tuned PID Results for CS1

	K_P	K_I	K_D
Propulsion	200	0.1	50
Heading	20	1.5	30

Figure 4.5 plots the simulated response obtained when tracking the manoeuvre from Figure 3.9 using the manually tuned control gains from Table 4.1.

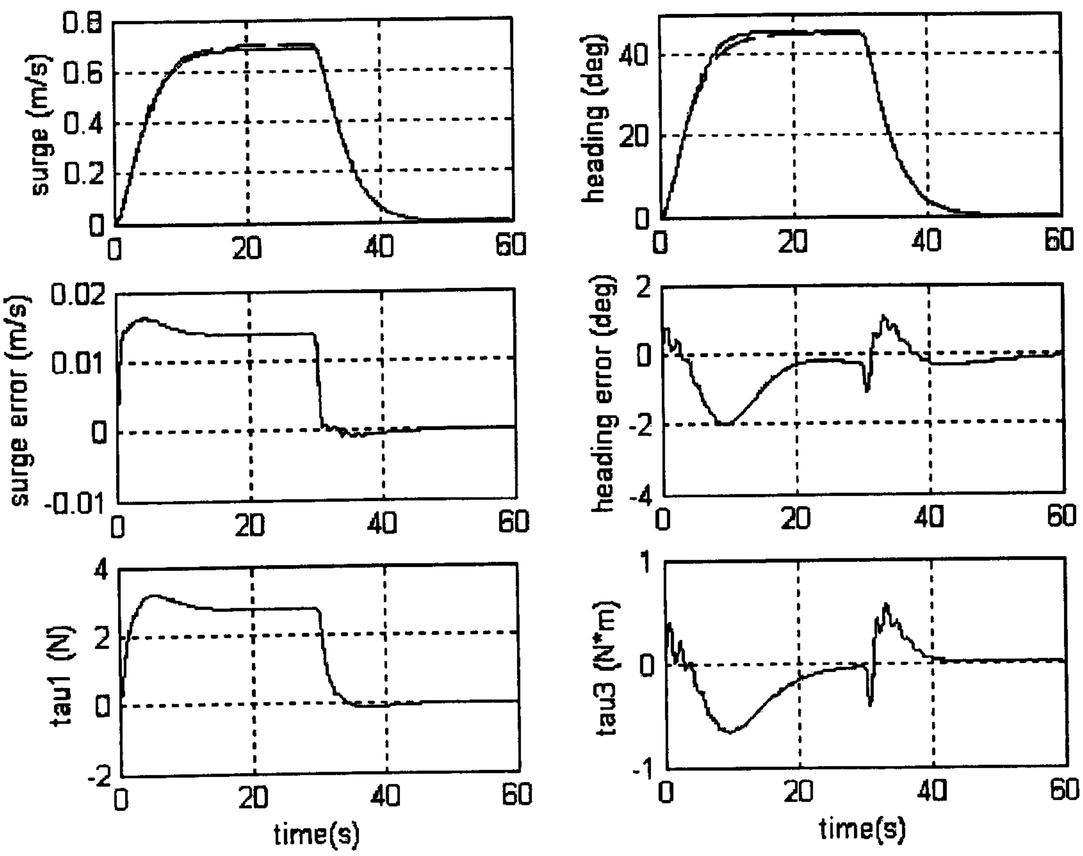


Figure 4.5: Simulated Results of the Manually Tuned PID Controller for CS1

This plot is the standard format that has been used for the presentation of the results in the thesis. It is divided into 6 subplots. In the left hand side, the results obtained for the propulsion subsystem are plotted, while the results obtained for heading are plotted in the right hand side. The subplots at the top of the figure represent the desired and measured outputs, u and ψ respectively. The desired outputs are plotted in a dashed line and the actual outputs are represented in a solid line. The subplots in the middle of the figure represent the output errors, i.e. the surge error, $u_d - u$, and the heading error, ψ_d - ψ. Finally, the subplots at the bottom of the figure depict the control signals corresponding with the propulsion and heading subsystems, i.e. τ_1 and τ_3.

As it can be seen from previous Figure 4.5 the tracking that this control strategy provides is quite good, although there is a slight steady-state error in the speed and some overshooting in the heading signal. Trying to reduce the overshooting leads to a very oscillatory heading control action, thus this choice of gains was a good compromise.

CyberShip II

Table 4.2 presents the gains obtained in the manual tuning of the PID gains for CS2. The gains values for CS1 from Table 4.1 were used as a starting point of the tuning process.

Table 4.2: Manually Tuned PID Results for CS2

	K_P	K_I	K_D
Propulsion	200	60	5
Heading	50	0.1	10

The next Figure 4.6 plots the simulated response obtained when implementing the gains from Table 4.2 in the model of CS2.

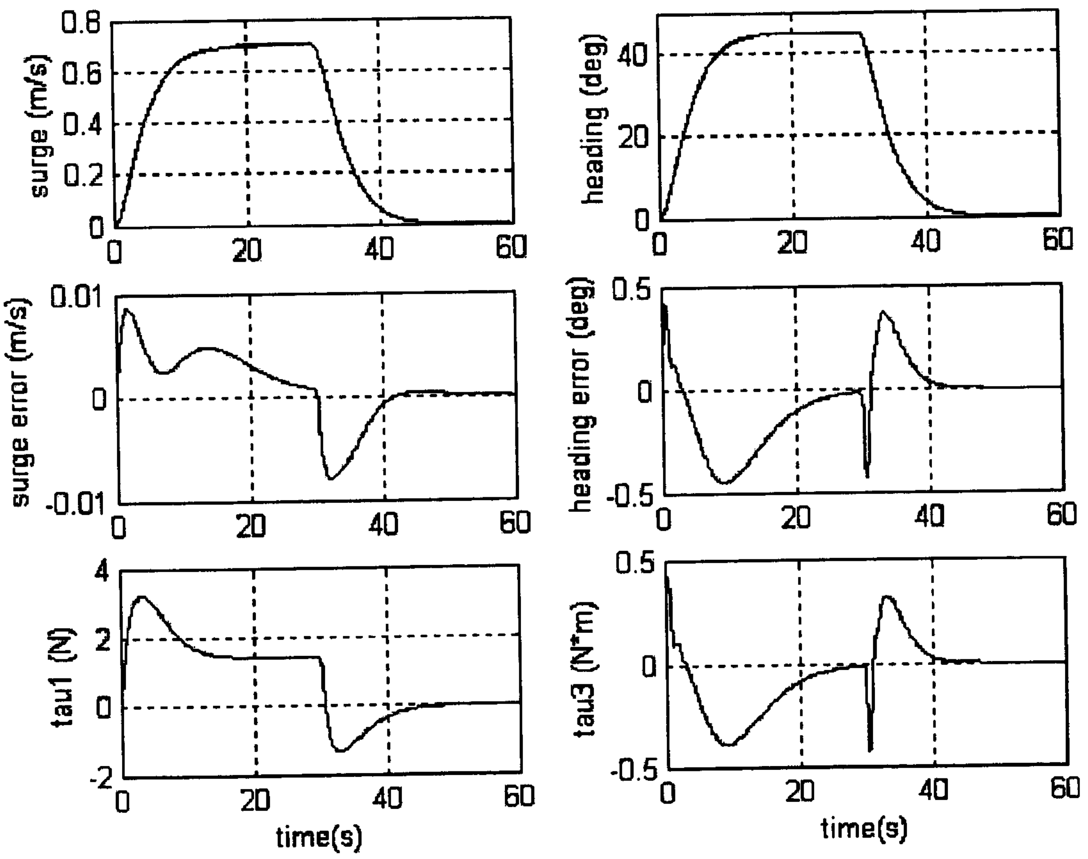


Figure 4.6: Simulated Results of the Manually Tuned PID Controller for CS2

The previous Figure 4.6 shows very good tracking of the desired responses as well as control signals free of unwanted oscillations. The control signals are smoother than in the CS1 case (see Figure 4.5).

Figure 4.7 and Figure 4.8 show the results obtained once this controller is implemented in the real plant. Figure 4.7 shows the performance of the controller when manoeuvring in calm waters. The manoeuvre to execute is a zig-zag manoeuvre as described in Figure 3.10.

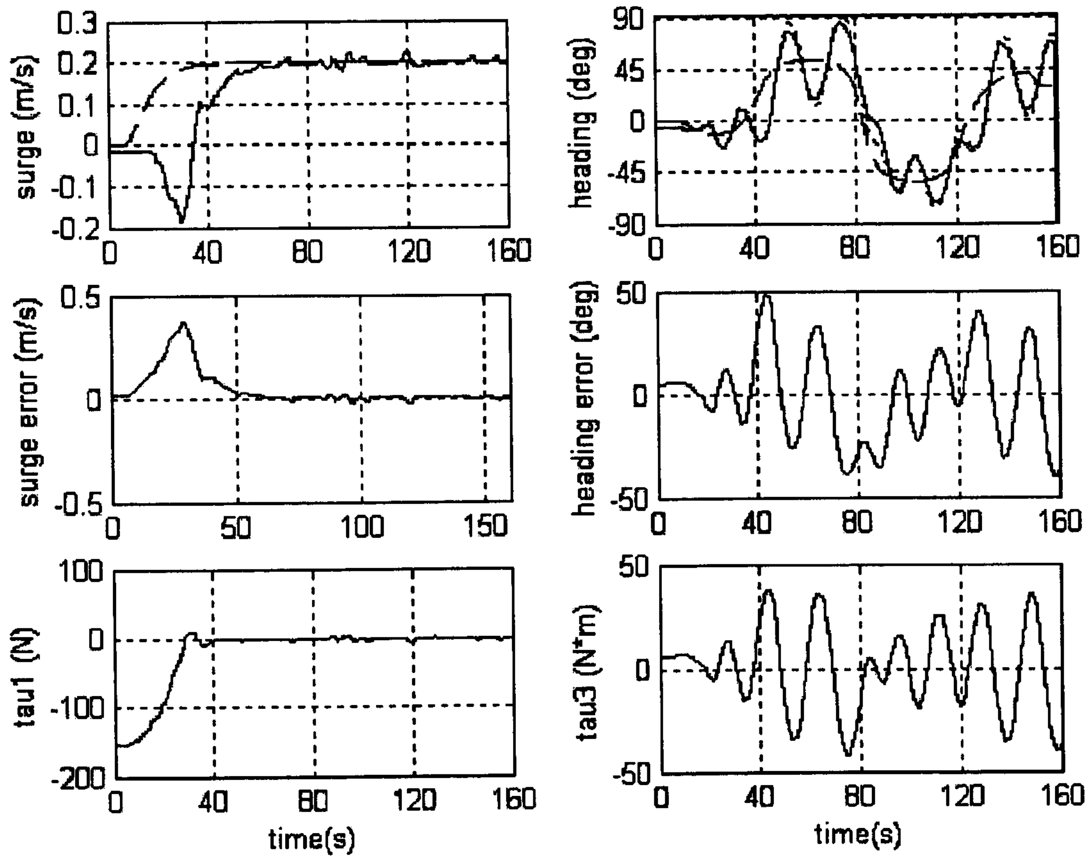


Figure 4.7: Real Results of the Manually Tuned PID Controller When Manoeuvring in Calm Waters

The performance shown in the figure is very poor. At the start of the manoeuvre the plant shows signs of instability and the performance is very oscillatory throughout the whole manoeuvre. This effect has not been observed in the simulated responses from Figure 4.6 and it raises doubts about the accuracy of the model. In order to dismiss the possibility of the poor performance being due to the change in the manoeuvre (from the double step used in the optimisation to the zig-zag used in the real testing), the performance of the controller has been simulated while tracking the zig-zag manoeuvre used in the real testing (see Figure F.1). The simulated results do not show the oscillatory response observed in Figure 4.7, therefore we can conclude that the model is not providing very precise results.

Figure 4.8 shows the performance of the controller once waves are generated in the water tank. The waves generated are as described in Section 3.5.

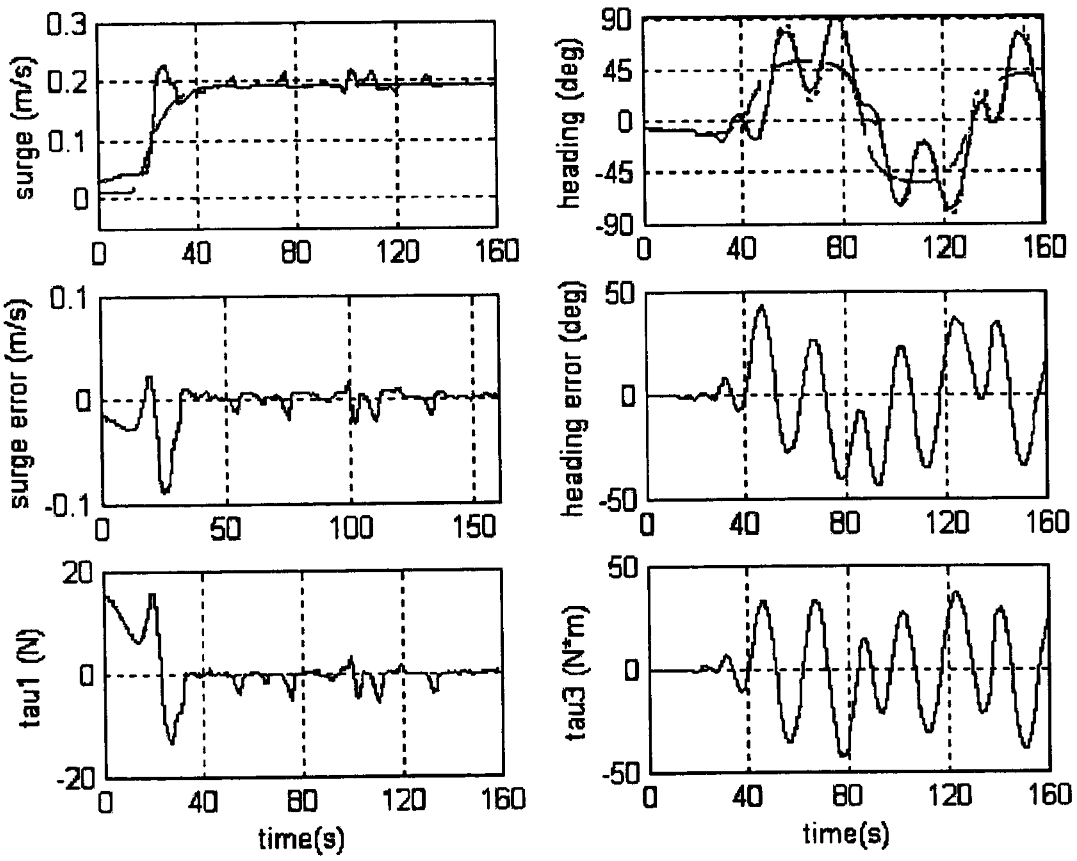


Figure 4.8: Real Results of the Manually Tuned PID Controller When Manoeuvring in the Presence of Waves

The performance is again very poor. Especially the heading tracking is very oscillatory. The effect of the waves is not very significant except for some induced rippling in the propulsion error.

4.4. POLE PLACEMENT

4.4.1. THEORETICAL BACKGROUND

Pole Placement [Andry *et al.* (1983), Kautsky *et al.* (1985)] is a control technique based on the use of a feedback gain. The following Figure 4.9 shows the control structure in state feedback control [Dutton *et al.* (1997)].

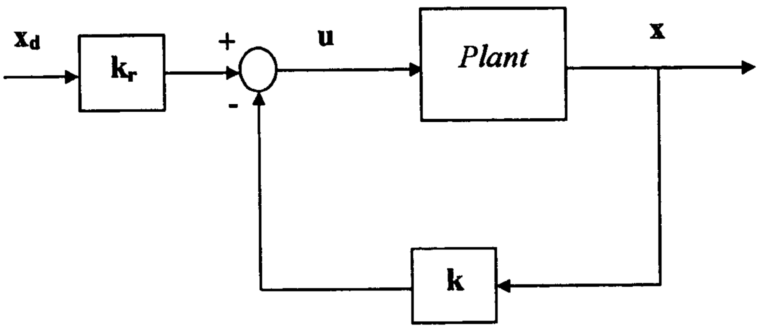


Figure 4.9: State Feedback Gain Control Structure

The desired closed-loop performance of a system is characterised by the position of the closed loop poles of the system (i.e. the eigenvalues of the closed-loop system matrix). One way of getting the system to have certain properties is by modifying the position of such closed loop poles. Thus, given the state-space equation of a MIMO system, $\dot{\mathbf{x}} = \mathbf{Ax} + \mathbf{Bu}$ (where $\mathbf{x}$ is the state vector, $\mathbf{u}$ is the input vector and $\mathbf{A}$ and $\mathbf{B}$ are real, constant matrices), this control over the poles can be accomplished by introducing a state-feedback control $\mathbf{u} = -\mathbf{k}\cdot\mathbf{x} + \mathbf{k_r}\cdot\mathbf{x_d}$, as shown in Figure 4.9 (where $\mathbf{k}$ is the feedback gain, $\mathbf{k_r}$ is the conditioning matrix for the reference and $\mathbf{x_d}$ the reference signal to track). Then, substituting in the space-state equation of the MIMO system gives:

$$\dot{\mathbf{x}} = (\mathbf{A} - \mathbf{B}\cdot\mathbf{k})\cdot\mathbf{x} + \mathbf{B}\cdot\mathbf{k_r}\cdot\mathbf{x_d} \qquad \textit{(4.10)}$$

Hence, the state feedback control problem consists of choosing $\mathbf{k}$ so that the modified closed loop system ($\mathbf{A_c} = \mathbf{A} - \mathbf{Bk}$) has the desired poles (i.e. the closed loop matrix $\mathbf{A_c}$ has the desired eigenvalues). It is basically an inverse eigenvalue problem. Instead of calculating the eigenvalues of a matrix, $\mathbf{k}$ has to be computed so that $\mathbf{A_c}$ contains a set of given eigenvalues [Kautsky *et al.* (1985)]. The problem has a solution if, and only if, the system (**A**, **B**) is controllable [Wonham (1967)]. In a SISO case the solution, if exists, is unique. However, for MIMO systems, the solution is underdetermined [Andry *et al.* (1983)]. Many authors have proposed solutions to the MIMO Pole Placement problem, exploiting in various ways the flexibility that provides the non-uniqueness of the solution [Fahmy and O'Reilly (1982), Fletcher (1981), Kautsky *et al.* (1985), Porter and D'Azzo (1977)].

In this research the robust Pole Placement method for linear state feedback proposed by Kautsky *et al.* (1985) has been used for the design of a MIMO controller for CS1 and CS2. A criterion must be chosen to restrict the degrees of freedom of the solution. Kautsky *et al.* (1985) present a numerical method to determine the solution to the problem that meets a robustness criterion, i.e. the method favours those solutions whose poles are more insensitive to perturbations in the coefficients of the matrices of the system.

Once the controllability of the matrices has been checked to determine the existence of solution, the robust pole placement method can be stated as follows [Kautsky *et al.* (1985)]:

If the eigenvectors of $\mathbf{A_c}$ are linearly independent it can be shown that the sensitivity of an eigenvalue λ_j to perturbations in **A**, **B** or **k** depends upon the magnitude of the condition number c_j:

$$c_j = \frac{\|\mathbf{y}_j\|_2 \cdot \|\mathbf{x}_j\|_2}{\left|\mathbf{y}_j^T \cdot \mathbf{x}_j\right|} \tag{4.11}$$

$\mathbf{x}_j$ and $\mathbf{y}_j$ being the right and left eigenvectors of $\mathbf{A_c}$ corresponding to the eigenvalue λ_j. A bound on the sensitivities, c_j, of the eigenvalues is given by the condition number of the matrix of right eigenvectors, **X**.

$$c_j \leq \|\mathbf{X}\|_2 \cdot \|\mathbf{X}^{-1}\|_2 \tag{4.12}$$

Therefore, the better conditioned the matrix of eigenvectors is, the less sensitive the eigenvalues are and the more robust the closed loop system becomes.

<u>Formulation of the robust state-feedback problem</u>: Given (**A**, **B**) as previously, find a real matrix **k** and a non-singular matrix **X** satisfying:

$$(\mathbf{A} - \mathbf{B} \cdot \mathbf{k})\mathbf{X} = \mathbf{X} \cdot \mathbf{\Lambda} \tag{4.13}$$

where $\Lambda = diag\{\lambda_1, \lambda_2, \lambda_3 \ldots \lambda_n\}$ is a diagonal matrix with the desired poles as elements of the diagonal, such that some measure of the conditioning of the eigenproblem is optimised [Kautsky *et al.* (1985)].

Theorem 4.1 by Kautsky *et al.* (1985) states that, given Λ and $\mathbf{X}$ non-singular, then there is a solution $\mathbf{k}$ if and only if

$$\mathbf{U}_1^T \cdot (\mathbf{A}\cdot\mathbf{X} - \mathbf{X}\cdot\Lambda) = 0 \qquad (4.14)$$

and

$$\mathbf{B} = [\mathbf{U}_0 \quad \mathbf{U}_1]\cdot\begin{bmatrix}\mathbf{Z}\\ \mathbf{0}\end{bmatrix} \qquad (4.15)$$

with $\mathbf{U} = [\mathbf{U}_0 \quad \mathbf{U}_1]$ orthogonal and $\mathbf{Z}$ non-singular. Then $\mathbf{k}$ is given by:

$$\mathbf{k} = \mathbf{Z}^{-1}\mathbf{U}_0^T\left(\mathbf{X}\cdot\Lambda\cdot\mathbf{X}^{-1} - \mathbf{A}\right) \qquad (4.16)$$

Therefore the problem now is reduced to choosing a non-singular matrix $\mathbf{X}$ so that the conditioning of the eigenproblem is minimised. To do that the following property is applied: the conditioning of an eigenproblem such as the one shown in Equation (4.13) is optimal if an only if the matrix $\mathbf{X}$ is unitary [Kautsky *et al.* (1985)]. Consequently, the eigenvectors $\mathbf{x}_j$ have to be chosen so that $\|\mathbf{x}_j\|_2 = 1$ and the vectors $\mathbf{x}_j$ are as orthogonal as possible to the space formed by the other eigenvectors (χ). As shown in the following Figure 4.10, the objective is to choose vectors $\mathbf{x}_j$ so that the angles α_j are maximised.

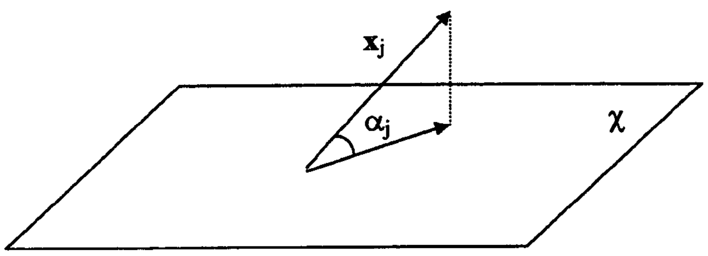


Figure 4.10: Choice of Vectors to Get Optimal Conditioning

A numerical solution to the computation of the unitary matrix of eigenvectors $\mathbf{X}$ is presented in Kautsky *et al.* (1985).

Thus, the algorithm for the calculation of the gain feedback matrix according to Kaustky *et al.* (1985) method consists mainly of 3 steps:

(1) Compute the decomposition of $\mathbf{B}$ as given by Equation (4.15)
(2) Select the vectors $\mathbf{x}_j$ that form the eigenvector matrix $\mathbf{X}$
(3) Calculate $\mathbf{k}$ as given by Equation (4.16)

4.4.2. CONTROLLER IMPLEMENTATION

The Matlab command called *place* solves the robust Pole Placement in state feedback control problem according to the method described by Kautsky *et al.* (1985). It has been used for the implementation of the MIMO Pole Placement controller in this work. Matlab also provides another command for solving the state feedback Pole Placement problem called *acker*. It is based on the formula by Ackermann (1972). Its use is not recommended since the algorithm is not numerically reliable. Moreover it can only be used for single input systems.

The state-space MIMO system that represents the system to control is:

$$\begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{r} \\ \dot{\psi} \end{bmatrix} = \mathbf{A}(u,v,r,\psi)\cdot\begin{bmatrix} u \\ v \\ r \\ \psi \end{bmatrix} + \mathbf{B}\cdot\begin{bmatrix} \tau_1 \\ \tau_2 \\ \tau_3 \end{bmatrix} \qquad (4.17)$$

In order to use the command the system matrices had to be linearised around a nominal point. As before, this was done using a Taylor method [Dutton *et al.* (1997)]. Therefore, Equation (4.17) becomes $\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\boldsymbol{\tau}$, where **A** and **B** are real and constant matrices.

In the Pole Placement implementation for reference tracking a matrix has to be used as a conditioning matrix for the reference signal. This matrix does not affect the position of the poles (see Equation 4.10). In this work it was chosen to be $\mathbf{k_r} = \mathbf{k}$ [White (1995)]. In this case the parameters to optimise are the 4 poles of the system. While choosing the set of poles it is necessary to restrict their multiplicity to 1 to meet the condition of the eigenvectors of $\mathbf{A_c}$ being linearly independent.

4.4.3. MANUAL TUNING RESULTS

CyberShip I

The following Table 4.3 presents the pole positions considered to be most suitable after the hand tuning of the Pole Placement controller for CS1

Table 4.3: Manually Tuned Pole Placement Results for CS1

pole1	*pole2*	*pole3*	*pole4*
-0.8	-4.5	-12	-15

Figure 4.11 shows the simulated results obtained when tracking the simultaneous double step manoeuvre once the Pole Placement controller presented in Table 4.3 is implemented in the model of CS1.

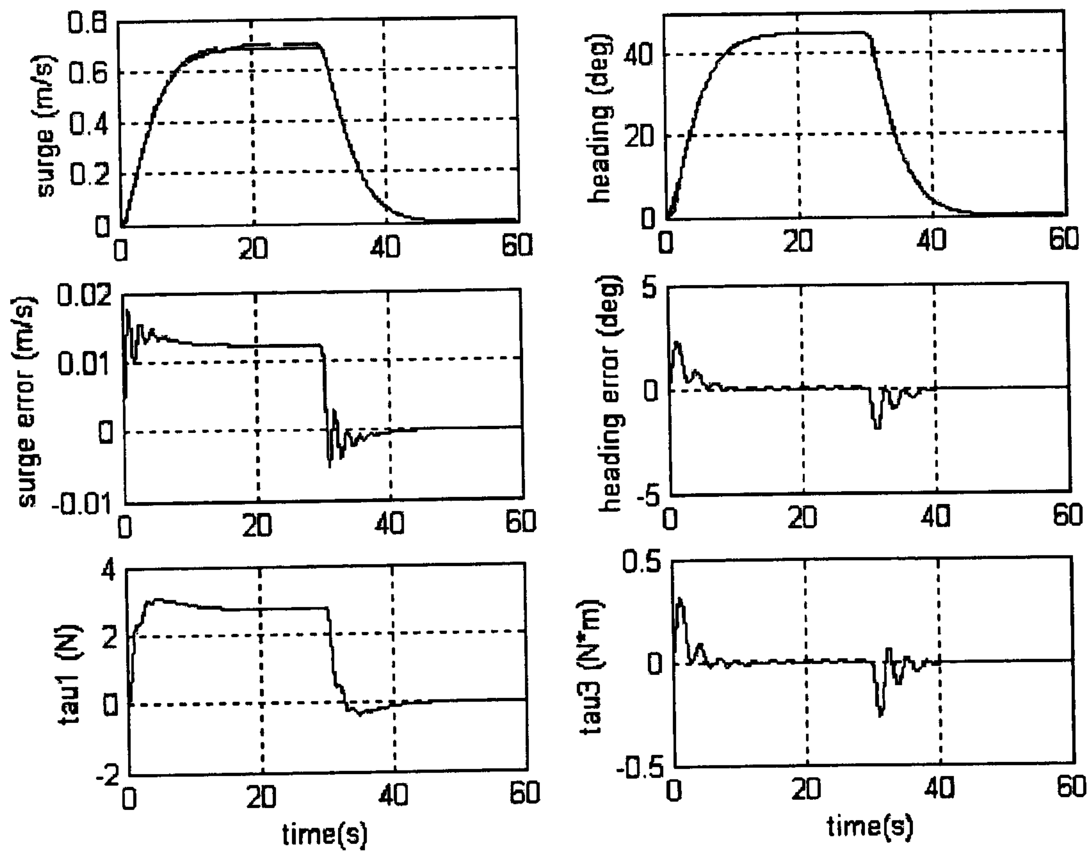


Figure 4.11: Simulated Results of the Manually Tuned Pole Placement Controller for CS1

As it can be seen from the plot, the tracking of the reference signals is good, with a slight steady-state error in the surge speed, although PID provided a closer tracking. As in Figure 4.5 the signals are a bit oscillatory, especially in the heading subsystem.

CyberShip II

When tuning the Pole Placement controller for CS2 it was found that the pole positions found in the tuning of CS1 (used as a starting point) provided a very good response, even better than for CS1. Therefore, the hand tuned results for the Pole Placement control of CS2 are the same than those of CS1 and they are shown in Table 4.4.

Table 4.4: Manually Tuned Pole Placement Results for CS2

pole1	*pole2*	*pole3*	*pole4*
-0.8	-4.5	-12	-15

The simulated performance of the Pole Placement controller shown in Table 4.4 once implemented in the model of CS2 is portrayed in Figure 4.12.

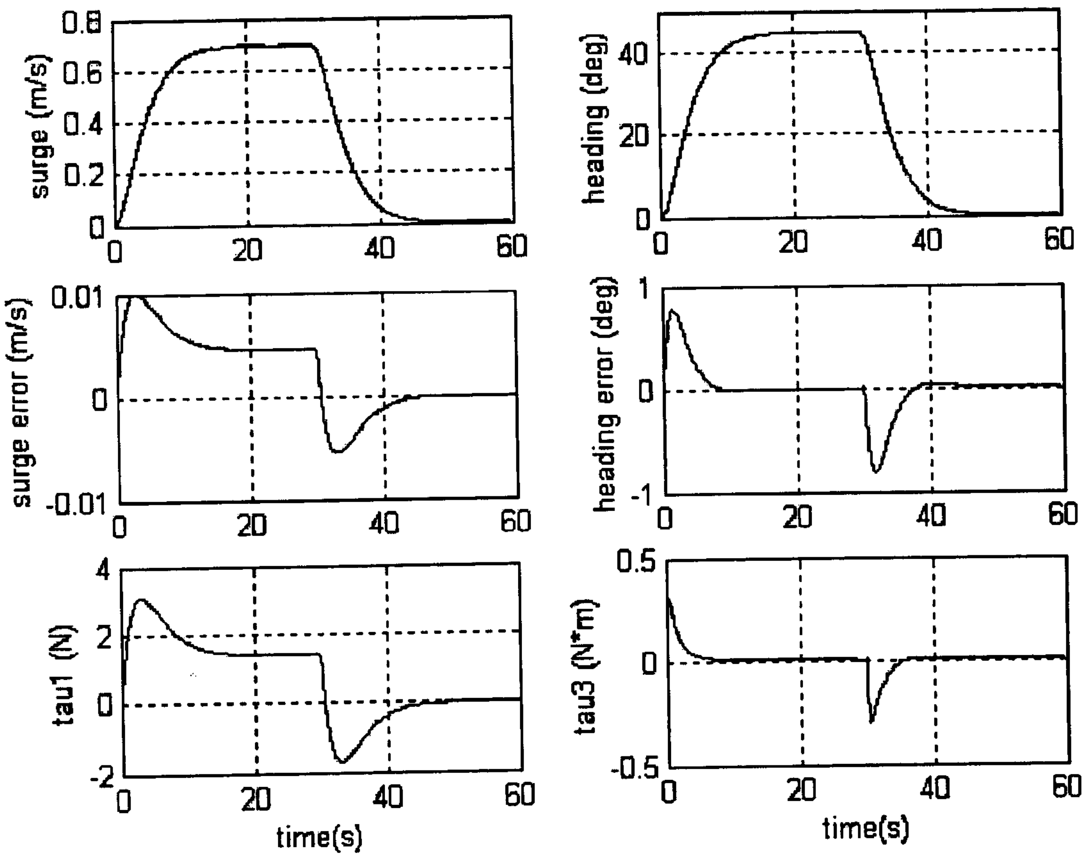


Figure 4.12: Simulated Results of the Manually Tuned Pole Placement Controller for CS2

The manoeuvring performance shown in Figure 4.12 is very good. The tracking is very good and the control signals very smooth. Once more the CS2 model has provided better simulated results. If compared with Figure 4.6 (manually tuned PID for CS2) the errors are slightly bigger, but the control effort required is smaller.

The following Figure 4.13 and 4.14 show the results obtained when the manually tuned Pole Placement controller from Table 4.4 is implemented in the real plant. Figure 4.13 shows the results obtained in calm waters.

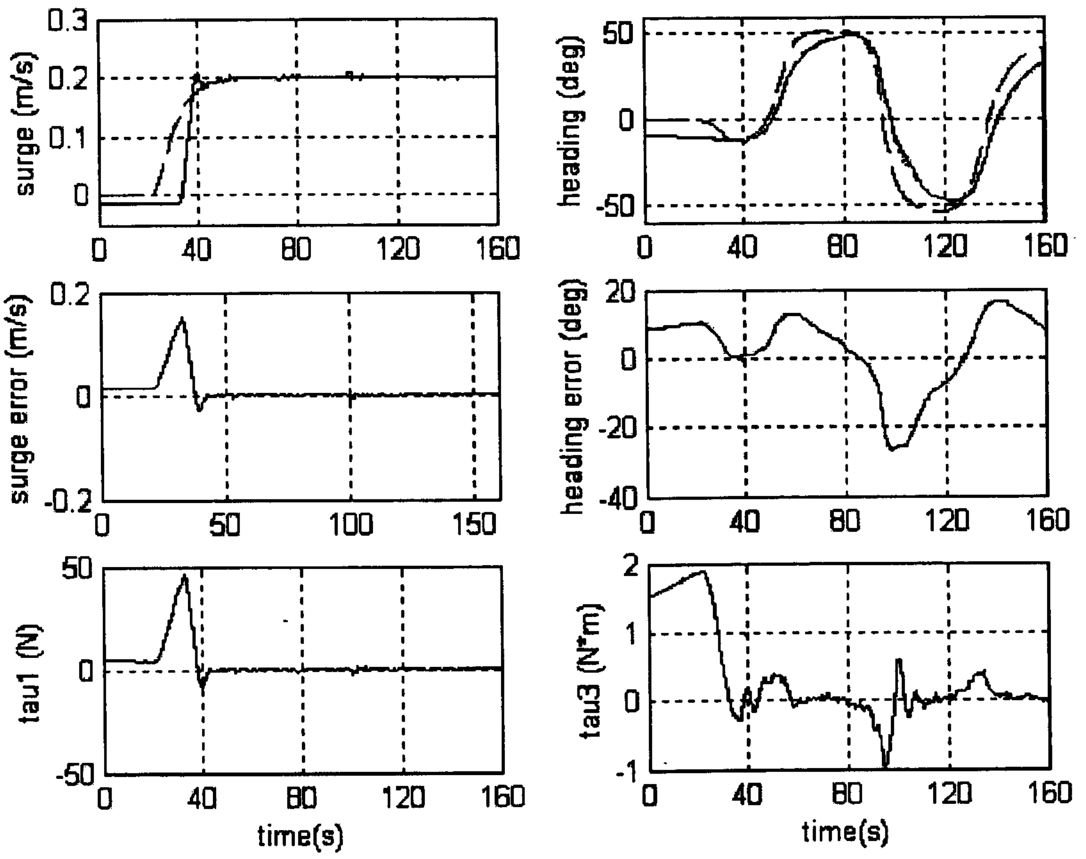


Figure 4.13: Real Results of the Manually Tuned Pole Placement Controller When Manoeuvring in Calm Waters

The results shown are much better than those obtained with the manually tuned PID controller. The responses do not present such an oscillatory behaviour and follow the desired response quite well, although the response of the system is quite slow.

As in the PID case, the figure showing the simulated performance of the Pole Placement controller tracking the zig-zag manoeuvre used in the real testing can be found in Appendix F (Figure F.2). Once more the simulated results do not reflect very accurately the actual response of the system once tested in the water basin. Although the difference is not as considerable as in the PID case.

Figure 4.14 illustrates the effect of including waves in the water tank while performing the manoeuvring tests using the manually tuned Pole Placement controller from Table 4.4.

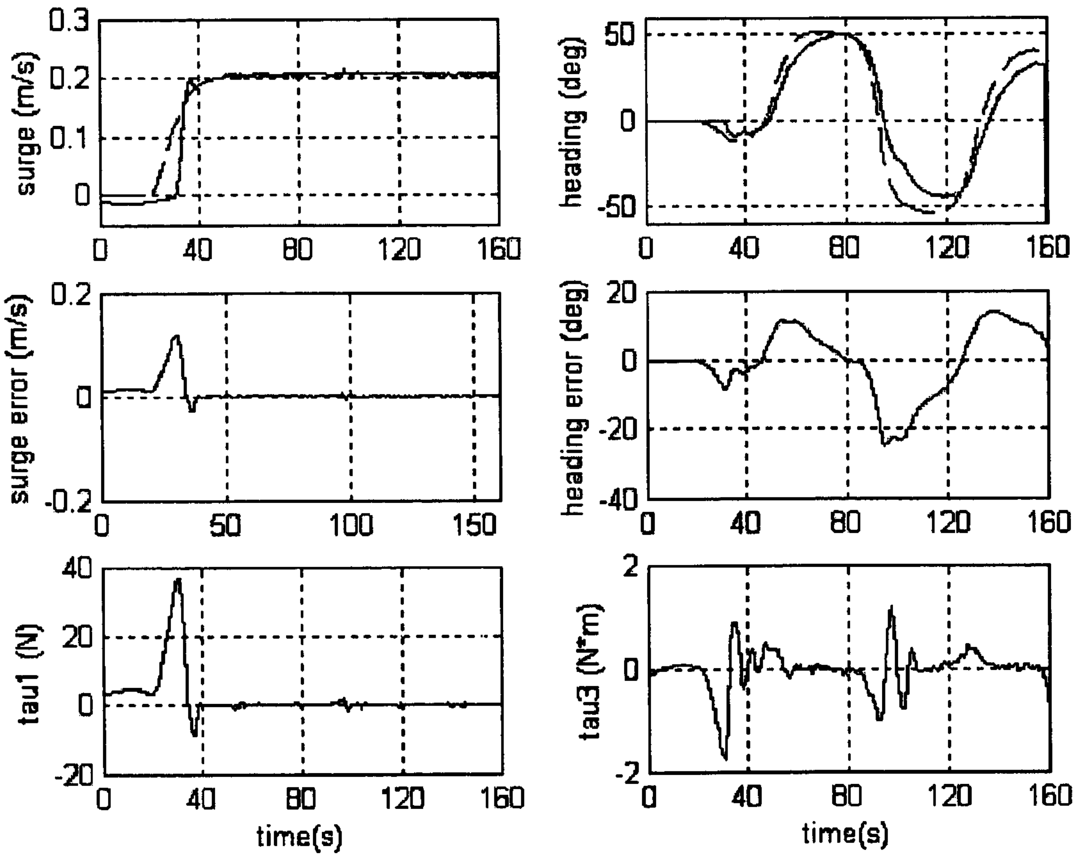


Figure 4.14: Real Results of the Manually Tuned Pole Placement Controller When Manoeuvring in the Presence of Waves

The inclusion of waves does not alter the responses significantly. These results, as all the real results of manually tuned controllers, were obtained in a second visit to the facilities and the improvements in the wave filtering block reduce the effect of the waves, especially since the wave height can not be chosen to be very high due to water getting into the hull of the boat.

4.5. NONLINEAR SLIDING MODE

4.5.1. THEORETICAL BACKGROUND

Sliding Mode control [Edwards and Spurgeon (1998), Slotine and Li (1991), Utkin (1972), Utkin and Yang (1978)], although more difficult to design than PID, is considered to be more robust and therefore more able to handle changes in the plant and external disturbances without as much performance degradation. The structure of a Sliding Mode controller is composed of a nominal part plus an additional term aimed at providing additional control effort for dealing with disturbances.

As in the previous control implementations, the control problem to be solved with a Sliding Mode controller is to get the plant response to track a specified desired response. This is achieved by comparing the actual states to be controlled ($\mathbf{x}$) with the desired states ($\mathbf{x}_d$). Sliding Mode control constructs a surface that is a function of the tracking error, $\hat{\mathbf{x}} = \mathbf{x} - \mathbf{x}_d$, called the *sliding surface* ($\sigma(\hat{\mathbf{x}})$) [Edwards and Sprugeon (1998), Healey and Lienard (1993), Healey and Marco (1992), Slotine and Li (1991), Utkin (1972), Utkin and Yang (1978)]. Then, the n-dimensional problem of solving $\hat{\mathbf{x}} = 0$, is reduced to driving the sliding surface to zero. The Sliding Mode controller provides a control input that drives the system to the sliding surface. Once the system is on the sliding surface is said to be in the *sliding mode*. Therefore, the problem of tracking $\mathbf{x}_d$ is equivalent to that of remaining on the zero sliding surface for all $t>0$ [McGookin (1997), Slotine and Li (1991)].

For Sliding Mode control the plant input has two distinct components [Edwards and Sprugeon (1998), McGookin *et al.* (2000b), Slotine and Li (1991)]: the equivalent control (u_{eq}) and the switching term (u_{sw}). The equivalent control provides the main control action, while the switching signal ensures the discontinuity of the control law across $\sigma(\hat{\mathbf{x}})$, supplying additional control to account for the presence of matched disturbances and unmodelled dynamics.

The equivalent component of the control action is usually chosen as a linear controller. In this case a feedback gain controller of the following form [Fossen (1994), McGookin (1997), Mudge and Patton (1988)] is used:

$$u_{eq} = -\mathbf{k} \cdot \mathbf{x} \tag{4.18}$$

where **k** is a feedback gain obtained from robust Pole Placement theory according to the method proposed by Kautsky *et al.* (1985) (see previous Section 4.4). This method was chosen since it minimises the sensitivity of the poles to perturbations in the coefficients of the matrices of the system.

The switching term is a non-linear term that provides the additional control action to counteract disturbances in the plant. This switching control action is designed round the sliding surface $\sigma(\hat{\mathbf{x}})$. The sliding surface has to be chosen so that as the surface value tends to zero the state error tends to zero as well, as shown in Figure 4.15.

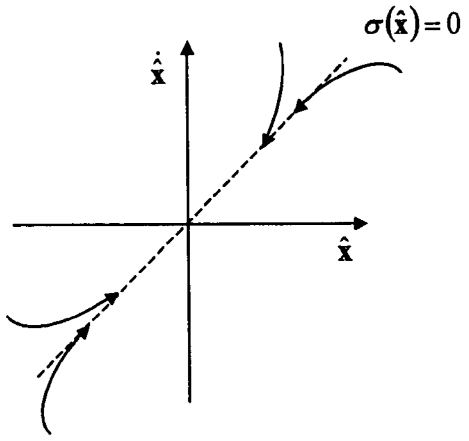


Figure 4.15: Sliding Surface in State-Space

The sliding surface used in this work is as follows [Healey and Lienard (1993), Healey and Marco (1992), McGookin (1997)]

$$\sigma(\hat{\mathbf{x}}) = \mathbf{h}^T \cdot \hat{\mathbf{x}} \tag{4.19}$$

where **h** is the right eigenvector of the desired closed loop system matrix $\mathbf{A}_c$ as defined in Equation (4.10). The following derivation to calculate the switching term from Equation (4.19) can be found in McGookin (1997).

Differentiating Equation (4.19) with respect to time gives:

$$\dot{\sigma}(\hat{\mathbf{x}}) = \mathbf{h}^T \cdot \dot{\hat{\mathbf{x}}} \tag{4.20}$$

A non-linear SISO system can be expressed in state-space form as:

$$\dot{\mathbf{x}} = \mathbf{A} \cdot \mathbf{x} + \mathbf{B} \cdot u + f(\mathbf{x}) \tag{4.21}$$

where $\mathbf{x}$ is the state vector, **A** is the system matrix, **B** is the input matrix, u is the vector of the inputs of the system and $f(\mathbf{x})$ represents the nonlinearities, unmodelled dynamics and external disturbances.

Substituting Equation (4.21) into Equation (4.20) gives:

$$\dot{\sigma}(\hat{\mathbf{x}}) = \mathbf{h}^T \cdot \left(\mathbf{A} \cdot \mathbf{x} + \mathbf{B} \cdot u + f(\mathbf{x}) - \dot{\mathbf{x}}_d\right) \tag{4.22}$$

and replacing $u = u_{eq} + u_{sw}$ the following equation is obtained:

$$\dot{\sigma}(\hat{\mathbf{x}}) = \mathbf{h}^T \cdot \left(\mathbf{A}\cdot\mathbf{x} + \mathbf{B}\cdot u_{eq} + \mathbf{B}\cdot u_{sw} + f(\mathbf{x}) - \dot{\mathbf{x}}_\mathbf{d}\right) \qquad (4.23)$$

If Equation (4.18) is used to substitute u_{eq} and $\mathbf{A_c} = \mathbf{A} - \mathbf{B}\cdot\mathbf{k}$ is applied, then Equation (4.23) becomes:

$$\begin{aligned}\dot{\sigma}(\hat{\mathbf{x}}) &= \mathbf{h}^T \cdot \left(\mathbf{A}\cdot\mathbf{x} - \mathbf{B}\cdot\mathbf{k}\cdot\mathbf{x} + \mathbf{B}\cdot u_{sw} + f(\mathbf{x}) - \dot{\mathbf{x}}_\mathbf{d}\right) \\ &= \mathbf{h}^T \cdot \left(\mathbf{A}_c\cdot\mathbf{x} + \mathbf{B}\cdot u_{sw} + f(\mathbf{x}) - \dot{\mathbf{x}}_\mathbf{d}\right)\end{aligned} \qquad (4.24)$$

Rearranging Equation (4.24) and assuming that $\mathbf{h}^T\mathbf{B}$ is not zero:

$$u_{sw} = (\mathbf{h}^T\mathbf{B})^{-1}\left(\mathbf{h}^T\dot{\mathbf{x}}_\mathrm{d} - \mathbf{h}^T\mathbf{A}_\mathbf{c}\mathbf{x} - \mathbf{h}^T f(\mathbf{x}) + \dot{\sigma}\ (\hat{\mathbf{x}})\right) \qquad (4.25)$$

If $\mathbf{h}^\mathrm{T}$ is chosen as the right eigenvector of $\mathbf{A_c}$ corresponding to a zero eigenvalue, the following holds:

$$\mathbf{h}^T\mathbf{A}_\mathbf{c} = 0 \qquad (4.26)$$

Therefore the Equation (4.25) can be simplified:

$$u_{sw} = (\mathbf{h}^T\mathbf{B})^{-1}\left(\mathbf{h}^T\dot{\mathbf{x}}_\mathrm{d} - \mathbf{h}^T f(\mathbf{x}) + \dot{\sigma}\ (\hat{\mathbf{x}})\right) \qquad (4.27)$$

The quantity $\dot{\sigma}\ (\hat{\mathbf{x}})$ is defined as [Healey and Lienard (1993), Healey and Marco (1992), McGookin (1997)]:

$$\dot{\sigma}(\hat{\mathbf{x}}) = \mathbf{h}^T \Delta f(\mathbf{x}) - \eta\cdot\mathrm{sgn}(\sigma(\hat{\mathbf{x}})) \qquad (4.28)$$

Here Δf(**x**) is the difference between the actual system deviations and the estimate made of this function, η is the switching gain and the *sign* function provides the switching action.

$$\mathrm{sgn(x)} = \begin{cases} 1 \text{ if } \mathrm{x} > 0 \\ 0 \text{ if } \mathrm{x} = 0 \\ -1 \text{ if } \mathrm{x} < 0 \end{cases} \qquad (4.29)$$

Substituting Equation (4.28) into Equation (4.27) the following is obtained:

$$\begin{aligned} u_{sw} &= (\mathbf{h}^T\mathbf{B})^{-1}\left(\mathbf{h}^T\dot{\mathbf{x}}_d - \mathbf{h}^T f(\mathbf{x}) + \mathbf{h}^T \Delta f(\mathbf{x}) - \eta\cdot\operatorname{sgn}(\sigma(\hat{\mathbf{x}}))\right) \\ &= (\mathbf{h}^T\mathbf{B})^{-1}\left(\mathbf{h}^T\dot{\mathbf{x}}_d - \mathbf{h}^T \hat{f}(\mathbf{x}) - \eta\cdot\operatorname{sgn}(\sigma(\hat{\mathbf{x}}))\right) \end{aligned} \tag{4.30}$$

where $\hat{f}(\mathbf{x})$ is the estimate of the system deviations. The term containing $\hat{f}(\mathbf{x})$ is either negligible or constant. In the latter case it can be compensated by making the switching gain sufficiently high [McGookin (1997)]. Therefore the switching term used in this research work has been [Fossen (1994), Healey and Lienard (1993), Healey and Marco (1992), McGookin (1997)]:

$$u_{sw} = (\mathbf{h}^T\mathbf{B})^{-1}\left(\mathbf{h}^T\dot{\mathbf{x}}_d - \eta\cdot\operatorname{sgn}(\sigma(\hat{\mathbf{x}}))\right) \tag{4.31}$$

and the resulting controller:

$$u = -\mathbf{k}\cdot\mathbf{x} + (\mathbf{h}^T\mathbf{B})^{-1}\left(\mathbf{h}^T\dot{\mathbf{x}}_d - \eta\cdot\operatorname{sgn}(\sigma(\hat{\mathbf{x}}))\right) \tag{4.32}$$

The additional control effort provided by the switching term, while beneficial to improve robustness, can also lead to a phenomenon called *chattering* [Edwards and Spurgeon (1998), McGookin (1997), Slotine (1984)]. The chattering is due to the inclusion of the sign function in the switching term and it can cause the control input to start oscillating around the zero sliding surface, resulting in unwanted wear and tear of the actuators.

One way to solve the problem is to smooth the switching term as the sliding surface gets closer to zero (*soft switching*). In this study this has been done by using the continuous hyperbolic tangent function instead of the discontinuous sign function. The hyperbolic tangent function has the same asymptotes as the sign function, but around the zero σ value there is a gradual transition area (called the ***boundary layer***). The width of this boundary layer is defined by the ***boundary layer thickness*** (Φ) [McGookin (1997)]. Equation (4.28) becomes:

$$\dot{\sigma}(\hat{\mathbf{x}}) = \mathbf{h}^T \Delta f(\mathbf{x}) - \eta \tanh\left(\frac{\sigma(\hat{\mathbf{x}})}{\Phi}\right) \tag{4.33}$$

The boundary layer thickness has to be large enough to counteract the large switching action. However if it is too large the switching action will be replaced by a proportional action (see Figure 4.16).

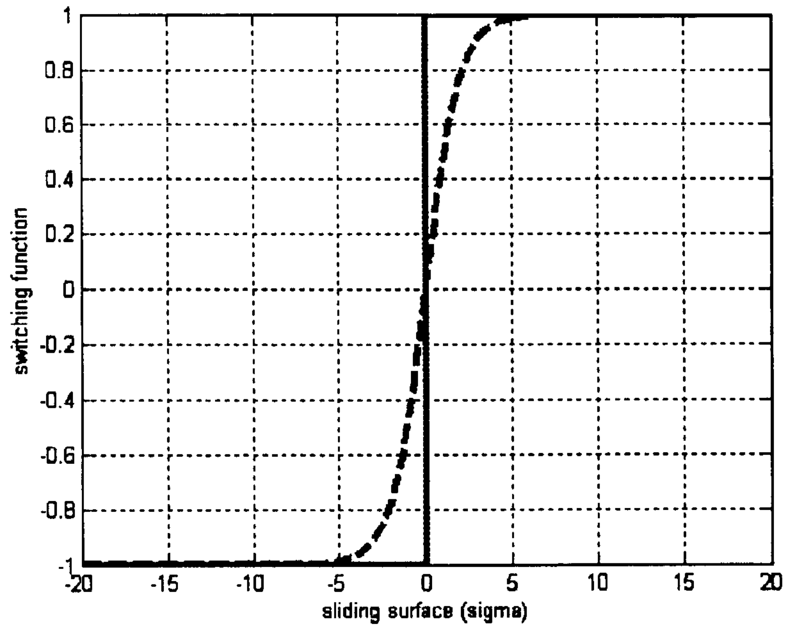


Figure 4.16: Comparison between Switching Functions with Sign Function (solid line) or Hyperbolic Tangent Function with $\Phi = 2$ (dashed line)

Hence, the final controller equation becomes:

$$u = -\mathbf{k}\cdot\mathbf{x} + (\mathbf{h}^T\mathbf{B})^{-1}\left(\mathbf{h}^T\dot{\mathbf{x}}_d - \eta\cdot\tanh\left(\frac{\sigma(\hat{\mathbf{x}})}{\Phi}\right)\right) \qquad (4.34)$$

4.5.2. CONTROLLER IMPLEMENTATION

As in the PID case, the system has been decoupled for the Sliding Mode control. This is a standard practice for controlling MIMO systems [Dutton *et al.* (1997), McGookin (1997)]. As shown in Equation (4.5), the heading subsystem consists of 3 states ($\mathbf{x}_H=[v, \psi, r]$) and one input ($u_H = \tau_3$). Meanwhile the propulsion subsystem consists of a single state ($x_P = u$) and one input ($u_P = \tau_1$). Since the method described for the design of the Sliding Mode controllers is based on SISO multi-state systems, a solution to the problem would be to include the force in the X-direction (τ_1) as a state and the commanded force (τ_{1com}) as the input of the subsystem, i.e. $x_P=[u, \tau_1]$ and $u_P = \tau_{1com}$. The relationship between both is given by Equation (3.6). The vectors of the desired states to track would then be, $x_{hd} = [v_d, r_d, \psi_d]$ and

$x_{hp}=[u_d,\ \tau_{1d}]$. However, this approach raises implementation issues such as the choice of the reference signal for τ_1. In this study the Sliding Mode controller for propulsion has been designed for the SISO single state system from Equation (4.4). If the pole for the propulsion subsystem is chosen to be small, the simplification from Equation (4.26) still holds.

For the calculation of the matrices **k** (Equation (4.18)) and **h** (Equation (4.19)), the system matrices have been linearised using a Taylor series expansion [Dutton *et al.* (1997)] around an operating point and truncating the Taylor series after the first partial derivatives. The Matlab command *place* has been used for the computation of the feedback matrix **k** like in the Pole Placement implementation.

Then, the control actions become [Alfaro-Cid *et al.* (2001a), McGookin (1997)]:

$$\tau_1 = -\mathbf{k}_p^T u + (\mathbf{h}_p^T \mathbf{b}_p)^{-1}\left(\mathbf{h}_p^T \dot{u}_d - \eta_p \tanh\left(\frac{\sigma_p(u-u_d)}{\Phi_p}\right)\right) \qquad (4.35)$$

$$\tau_3 = -\mathbf{k}_h^T \mathbf{x}_h + (\mathbf{h}_h^T \mathbf{b}_h)^{-1}\left(\mathbf{h}_h^T \dot{\mathbf{x}}_{hd} - \eta_h \tanh\left(\frac{\sigma_h(\mathbf{x}_h-\mathbf{x}_{hd})}{\Phi_h}\right)\right) \qquad (4.36)$$

Consequently, there are 4 parameters to optimise for the heading Sliding Mode controller: 2 poles in the equivalent term (the pole that is associated with the heading angle feedback is set to zero to meet Equation (4.26)) and η_h and Φ_h in the switching term. In the same way, there are 3 parameters to optimise for the propulsion control: 1 pole, η_p and Φ_p.

4.5.3. MANUAL TUNING RESULTS

CyberShip I

The following Table 4.5 presents the results obtained in the manual tuning of the gains of the propulsion and heading Sliding Mode controllers.

Table 4.5: Manually Tuned Sliding Mode Results for CS1

	pole$_1$	*pole$_2$*	η	Φ
Propulsion	0		10	1
Heading	-0.3	-0.5	5	0.8

Once the manually tuned parameters have been implemented in the controller the performance of the vessel tracking the simultaneous double step manoeuvre is shown in Figure 4.17.

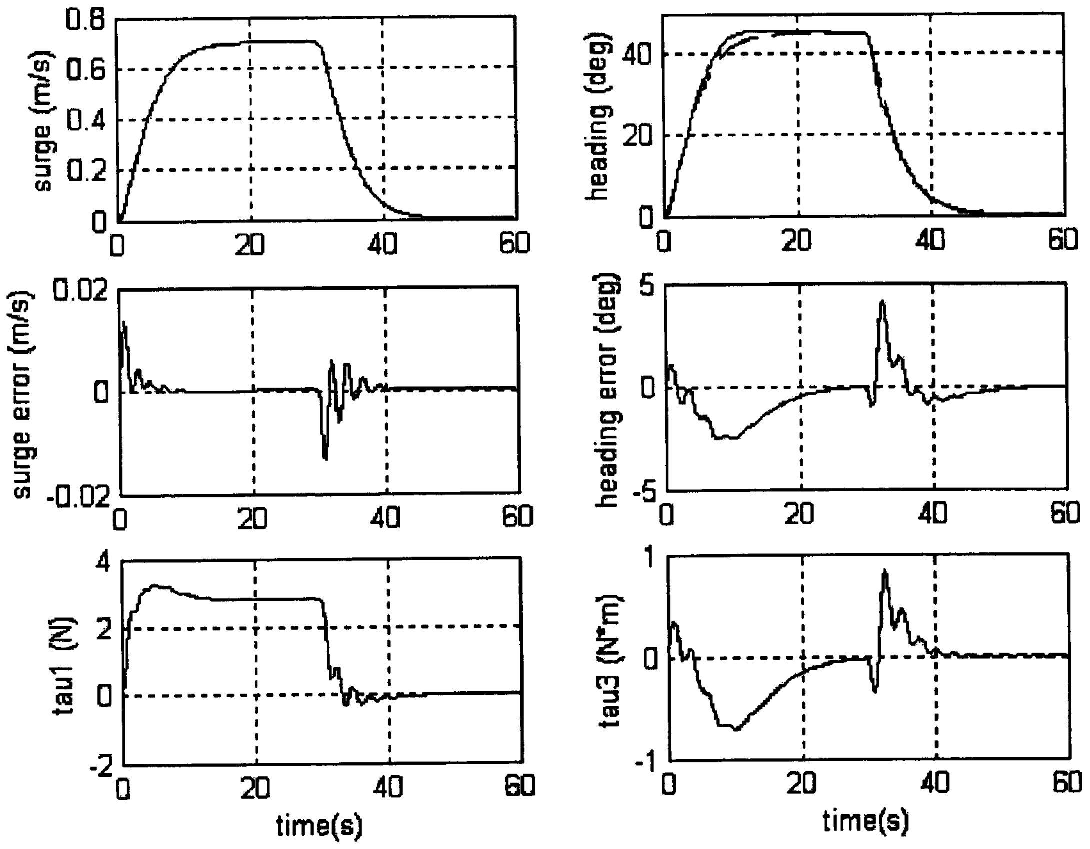


Figure 4.17: Simulated Results of the Manually Tuned Sliding Mode Controller for CS1

The heading tracking provided by the Sliding Mode controller is definitely worse than that obtained with PID (Figure 4.5) or Pole Placement (Figure 4.11), i.e. the heading error is nearly 5°, while in PID and Pole Placement it was kept below 3°.

CyberShip II

The numerical valued obtained after the manual tuning of the Sliding Mode controllers for CS2 are presented in the following Table 4.6.

Table 4.6: Manually Tuned Sliding Mode Results for CS2

	$pole_1$	$pole_2$	η	Φ
Propulsion	0		10	1
Heading	-0.3	-0.35	5	0.8

The next Figure 4.18 shows the performance of the Sliding Mode controller when tracking the simultaneous double step manoeuvre used for the tuning.

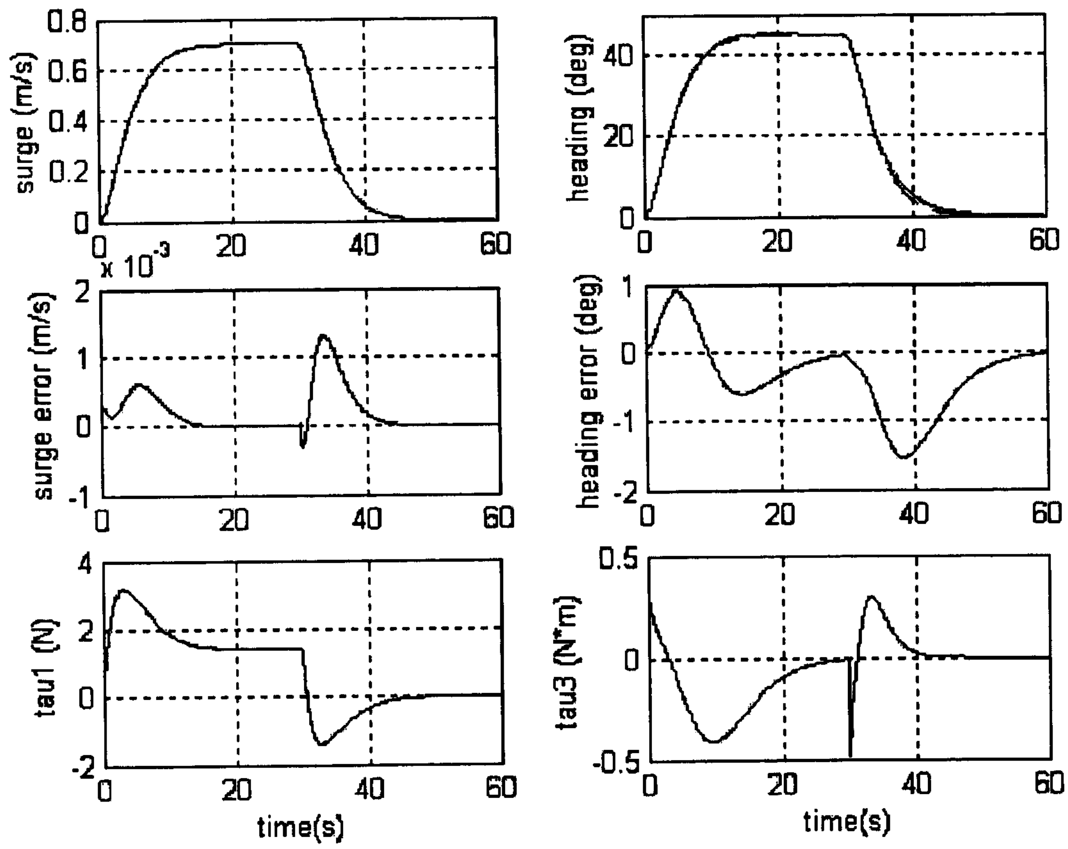


Figure 4.18: Simulated Results of the Manually Tuned Sliding Mode Controller for CS2

The tracking of the reference obtained with Sliding Mode is slightly worse than with PID (Figure 4.6) or Pole Placement (Figure 4.12). The heading error is at some points larger than 1°, while with previous controllers is kept around 0.5°. There is a slight overshooting in the heading response. The controller has proven to be quite difficult to tune. As usual the result obtained for CS2 is free of the oscillations that characterise CS1.

Figure 4.19 and 4.20 show the real results obtained when the manually tuned controller from Table 4.6 is implemented in the real plant. Figure 4.19 shows the results obtained in still waters.

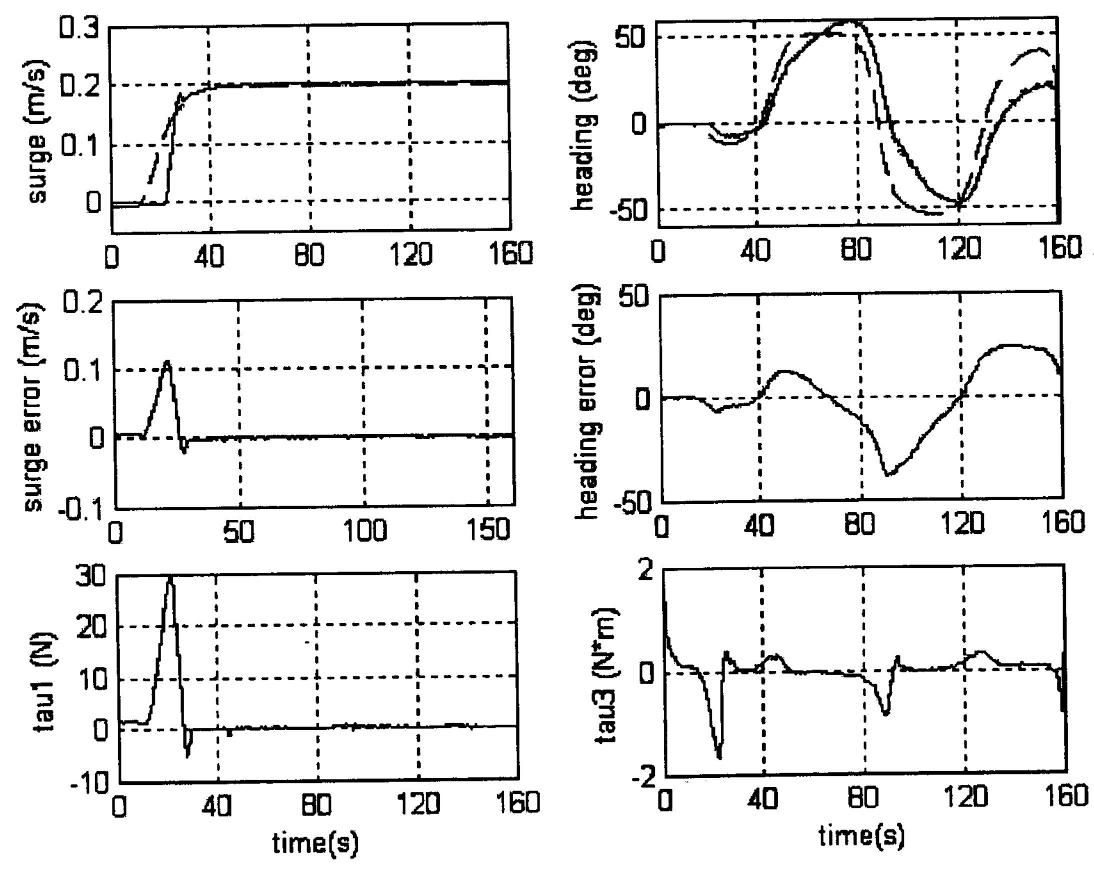


Figure 4.19: Real Results of the Manually Tuned Sliding Mode Controller When Manoeuvring in Calm Waters

The responses shown in the previous figure are free of the oscillations that characterised the PID responses. The slight overshooting that can be appreciated in the simulated responses from Figure 4.18 is reflected in the heading real responses from Figure 4.19. This overshooting induces a delay in the heading response. The system has to overcompensate for the overshooting when the following turning starts.

Figure F.3 (Appendix F) shows the simulated results obtained when tracking the zig-zag manoeuvre used in the real testing. Although there are some discrepancies with the results shown in Figure 4.19, the simulated and real results obtained with the Sliding Mode controller are closer than those obtained with PID control.

Figure 4.20 shows the manoeuvring performance of the manually tuned Sliding Mode controller when waves are generated in the water tank.

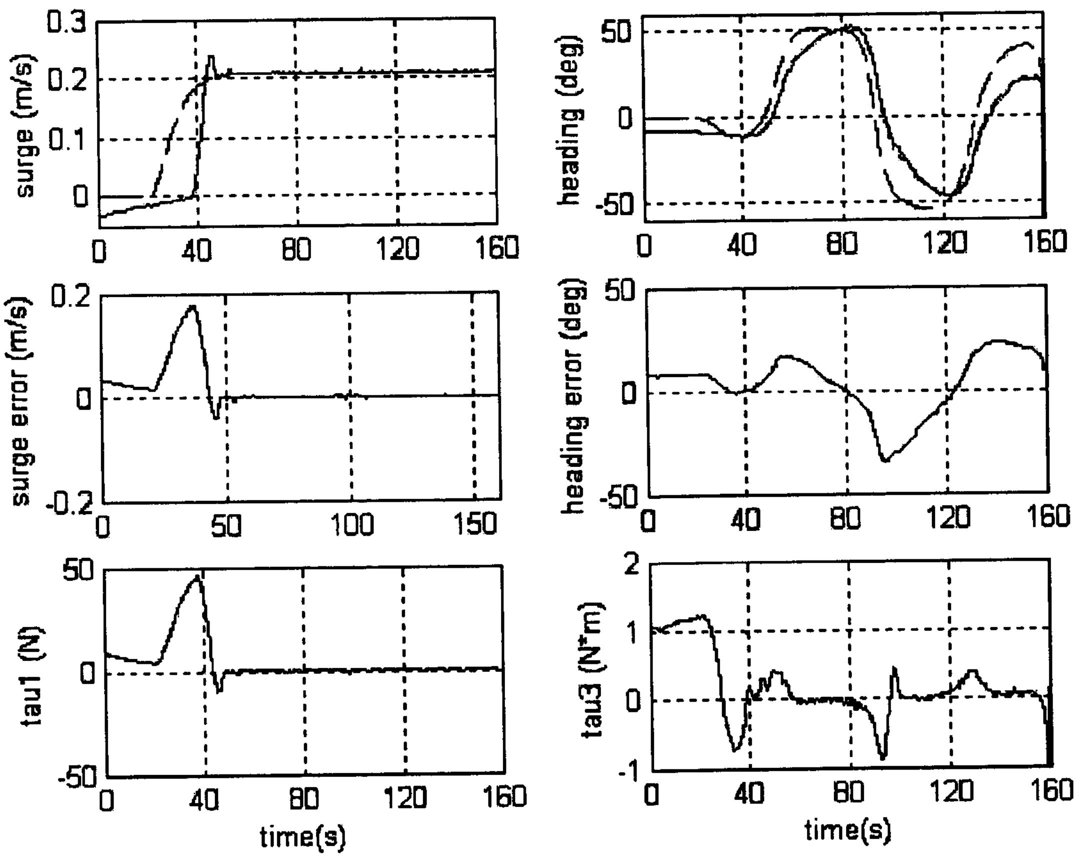


Figure 4.20: Real Results of the Manually Tuned Sliding Mode Controller When Manoeuvring in the Presence of Waves

Once more the inclusion of waves does not change the responses significantly.

4.6. H_∞ OPTIMAL CONTROL

4.6.1. THEORETICAL BACKGROUND

As with the rest of the controllers already presented, H_∞ control has been used to provide the structure for propulsion controllers (for governing surge velocity) and navigation controllers (for governing heading) for CS1 and CS2. Like Pole Placement and as opposed to PID and SM, the H_∞ controller designed is a MIMO controller (i.e. the system has not been decoupled). The H_∞ controller design used in this work is based on the state-space H_∞ structure presented by Zhou *et al.* (1996). The following section is an account of the theory behind this state-space H_∞ formulation.

H_∞ control is based on a standard feedback structure. It consists of a plant, a controller, reference **r**, commanded input **u**, sensor noise **n** and plant disturbance **d** as shown in Figure 4.21 below.

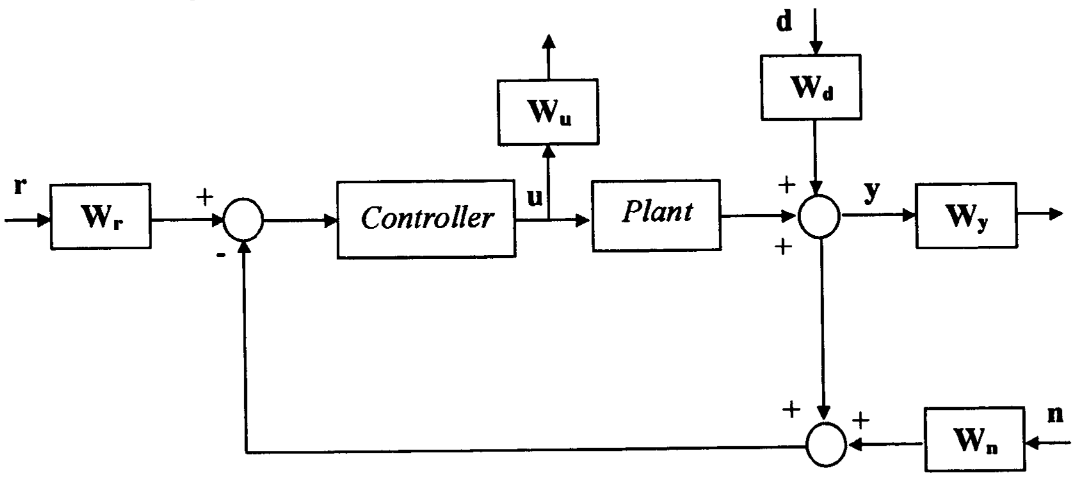


Figure 4.21: Standard Feedback Configuration with Weightings

In order to include some performance objectives in the system model, the standard feedback structure is modified by adding some weighting functions. The aim is not only to put the emphasis on some of the components but also to make components measured in different metrics comparable. Once the weighting matrices are included the feedback configuration is as shown in Figure 4.21.

Any feedback control configuration can be expressed as a *linear fractional transformation (LFT)* [Zhou *et al.* (1996)]. The following block diagram (Figure 4.22) represents a *lower linear fractional transformation, F_l(G,K)*, of the previous Figure 4.21.

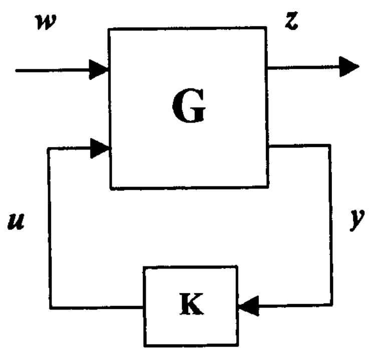


Figure 4.22: Linear Fractional Transformation Configuration

Here **w** represents the exogenous input, consisting of commands, external disturbances and sensor noise (**r**, **n** and **d** in previous Figure 4.21), **y** is the measurement available to the controller, **u** is the output from the controller, and **z** is the error signal that is desired to keep small. The transfer function matrix **G** represents not only the conventional plant to be modelled but also any weighting functions included to specify the desired performance and **K** represents the controller.

The H_∞ optimal control problem is then to design a stabilizing controller, **K**, so as to minimise the closed-loop transfer function from **w** to **z**, T_{zw}, in the H_∞ norm,

$$\|T_{zw}\|_\infty = \sup_\omega \bar{\sigma}(T_{zw}(j\omega)) \qquad (4.37)$$

$\bar{\sigma}(T_{zw}(j\omega))$ being the largest singular value of $T_{zw}(j\omega)$. Thus, the H_∞ norm is the supreme of the largest singular values of $T_{zw}(j\omega)$ over all the values of ω.

Finding an optimal H_∞ controller is often both numerically and theoretically complicated. However, in practice it is often not necessary to design an optimal controller. It is usually enough to obtain controllers that are very close, in the norm sense, to the optimal designs. These will be called *suboptimal controllers*.

<u>Suboptimal H_∞ control problem</u>: given $\gamma>0$, find all admissible controllers **K**, if there are any, such that $\|T_{zw}\|_\infty < \gamma$.

Designing a controller that reduces $\|T_{zw}\|_\infty$ results in a minimisation of the signal gain from disturbance inputs to error outputs in the controlled system. In addition, the H_∞ norm gives the maximum energy gain of the system, which is minimised as well.

In order to solve the suboptimal H_∞ state-space problem it is necessary to find a *stabilizing solution* for two *algebraic Riccati equations (ARE)* [Glover and Doyle (1988), Zhou *et al.* (1996)]. An ARE is a matrix equation in the following form:

$$\mathbf{A}^*\mathbf{X} + \mathbf{XA} + \mathbf{XRX} + \mathbf{Q} = 0 \qquad (4.38)$$

where **A**, **Q** and **R** are real nxn matrices with **Q** and **R** symmetric. $\mathbf{A}^*$ represents the complex conjugate transpose of **A**. Associated with this ARE there is a $2n$x$2n$ matrix called *Hamiltonian matrix* [Glover and Doyle (1988), Zhou *et al.* (1996)]:

$$\mathbf{H} = \begin{bmatrix} \mathbf{A} & \mathbf{R} \\ -\mathbf{Q} & -\mathbf{A}^* \end{bmatrix} \qquad (4.39)$$

The solution of the suboptimal H_∞ state-space problem requires finding a stabilizing solution for that equation. Assume **H** has no eigenvalues on the imaginary axis, i.e. the system is not marginally stable. Then it must have n eigenvalues in $Re(s)<0$ (left hand plane) and n in $Re(s)>0$ (right hand plane). Consider $X_-(H)$, the n-dimensional invariant subspace corresponding to eigenvalues $Re(s)<0$. By finding a basis for $X_-(H)$, forming a matrix with the basis vectors, and partitioning the matrix gives:

$$\mathbf{X}_-(\mathbf{H}) = \mathrm{Im}\begin{bmatrix} \mathbf{X}_1 \\ \mathbf{X}_2 \end{bmatrix} \qquad (4.40)$$

where $\mathbf{X}_1$, $\mathbf{X}_2 \in \Re^{n\times n}$. If $\mathbf{X}_1$ is non-singular, **X** can be defined as $\mathbf{X}=\mathbf{X}_2\mathbf{X}_1^{-1}$. Then $\mathbf{X}=Ric(\mathbf{H})$ is uniquely determined by **H**, (i.e. *Ric* is a function $Ric:\mathbf{H}\rightarrow\mathbf{X}$) and it is called the ***stabilizing solution***. Therefore, the domain of *Ric*, called ***dom(Ric)***, consists of Hamiltonian matrices **H** with no eigenvalues on the imaginary axis and that determine a non-singular $\mathbf{X}_1$ matrix.

The following results solve the general case of the suboptimal H_∞ control problem. Consider the system described by the block diagram of Figure 4.22. The partition of the transfer matrix G is taken to be:

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \mathbf{A}\mathbf{x}(t) + \mathbf{B}_1\mathbf{w}(t) + \mathbf{B}_2\mathbf{u}(t) \\ \mathbf{z}(t) &= \mathbf{C}_1\mathbf{x}(t) + \mathbf{D}_{11}\mathbf{w}(t) + \mathbf{D}_{12}\mathbf{u}(t) \\ \mathbf{y}(t) &= \mathbf{C}_2\mathbf{x}(t) + \mathbf{D}_{21}\mathbf{w}(t) + \mathbf{D}_{22}\mathbf{u}(t) \end{aligned} \qquad (4.41)$$

which is compatible with the dimensions $\mathbf{z}(t)\in\Re^{p1}$, $\mathbf{y}(t)\in\Re^{p2}$, $\mathbf{w}(t)\in\Re^{m1}$, $\mathbf{u}(t)\in\Re^{m2}$, and the state $\mathbf{x}(t)\in\Re^{n}$.

The following assumptions are made [Glover and Doyle (1988), Zhou *et al.* (1996)]:

Assumption 1:

($\mathbf{A}$, $\mathbf{B}_2$) is stabilizable and ($\mathbf{C}_2$, $\mathbf{A}$) is detectable

Assumption 2:

$$\mathbf{D}_{12} = \begin{bmatrix} \mathbf{0} \\ \mathbf{I} \end{bmatrix} \text{ and } \mathbf{D}_{21} = \begin{bmatrix} \mathbf{0} & \mathbf{I} \end{bmatrix}$$

Assumption 3:

$$\begin{bmatrix} \mathbf{A}-j\omega\mathbf{I} & \mathbf{B}_2 \\ \mathbf{C}_1 & \mathbf{D}_{12} \end{bmatrix} \text{ has full column rank for all } \omega$$

Assumption 4:

$$\begin{bmatrix} \mathbf{A}-j\omega\mathbf{I} & \mathbf{B}_1 \\ \mathbf{C}_2 & \mathbf{D}_{21} \end{bmatrix} \text{ has full row rank for all } \omega$$

Assumption (1) is necessary for the existence of stabilising controllers. The assumptions in (2) mean that the penalty on z includes a non-singular, normalised penalty on the control u, and that the exogenous signal weighting is normalised and non-singular. Assumptions (3) and (4) together with (1) guarantee that the two Hamiltonian matrices in the corresponding H_2 problem belong to *dom(Ric)* [Glover and Doyle (1988), Zhou *et al.* (1996)]..

Let **R** and **Ř** be defined as:

$$\mathbf{R} = \mathbf{D}_{1\bullet}^{*}\mathbf{D}_{1\bullet} - \begin{bmatrix} \gamma^2\mathbf{I}_{m1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}, \quad \text{where } \mathbf{D}_{1\bullet} = [\mathbf{D}_{11} \quad \mathbf{D}_{12}] \tag{4.42}$$

$$\breve{\mathbf{R}} = \mathbf{D}_{\bullet 1}\mathbf{D}_{\bullet 1}^{*} - \begin{bmatrix} \gamma^2\mathbf{I}_{p1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}, \quad \text{where } \mathbf{D}_{\bullet 1} = \begin{bmatrix} \mathbf{D}_{11} \\ \mathbf{D}_{21} \end{bmatrix} \tag{4.43}$$

Then the Riccati equations will be given by the following Hamiltonian matrices:

$$\mathbf{H}_\infty = \begin{bmatrix} \mathbf{A} & \mathbf{0} \\ -\mathbf{C}_1^{*}\mathbf{C}_1 & -\mathbf{A}^{*} \end{bmatrix} - \begin{bmatrix} \mathbf{B} \\ -\mathbf{C}_1^{*}\mathbf{D}_{1\bullet} \end{bmatrix} \mathbf{R}^{-1} \begin{bmatrix} \mathbf{D}_{1\bullet}^{*}\mathbf{C}_1 & \mathbf{B}^{*} \end{bmatrix} \tag{4.44}$$

$$\mathbf{J}_\infty = \begin{bmatrix} \mathbf{A}^{*} & \mathbf{0} \\ -\mathbf{B}_1\mathbf{B}_1^{*} & -\mathbf{A} \end{bmatrix} - \begin{bmatrix} \mathbf{C}^{*} \\ -\mathbf{B}_1\mathbf{D}_{\bullet 1}^{*} \end{bmatrix} \breve{\mathbf{R}}^{-1} \begin{bmatrix} \mathbf{D}_{\bullet 1}\mathbf{B}_1^{*} & \mathbf{C} \end{bmatrix} \tag{4.45}$$

Thus, $\mathbf{X}_\infty = Ric(\mathbf{H}_\infty)$ and $\mathbf{Y}_\infty = Ric(\mathbf{J}_\infty)$.

The controller **K** will be stated in terms of the solutions $\mathbf{X}_\infty$ and $\mathbf{Y}_\infty$ of the Riccati equations together with the matrices **F** and **L** defined as shown in Equations (4.46) and (4.47):

$$\mathbf{F} = -\mathbf{R}^{-1}\left[\mathbf{D}_{1\bullet}^{*}\mathbf{C}_1 + \mathbf{B}^{*}\mathbf{X}_{\infty}\right] \qquad (4.46)$$

$$\mathbf{L} = -\left[\mathbf{B}_1\mathbf{D}_{\bullet 1}^{*} + \mathbf{Y}_{\infty}\mathbf{C}^{*}\right]\cdot\breve{\mathbf{R}}^{-1} \qquad (4.47)$$

Partitions **D**, **F** and **L** are as follows:

	$\mathbf{F}_{11\infty}^{*}$	$\mathbf{F}_{12\infty}^{*}$	$\mathbf{F}_{2\infty}^{*}$
$\mathbf{L}_{11\infty}^{*}$	$\mathbf{D}_{1111}$	$\mathbf{D}_{1112}$	$\mathbf{0}$
$\mathbf{L}_{12\infty}^{*}$	$\mathbf{D}_{1121}$	$\mathbf{D}_{1112}$	$\mathbf{I}$
$\mathbf{L}_{2\infty}^{*}$	$\mathbf{0}$	$\mathbf{I}$	$\mathbf{0}$

Theorem 4.2 (Existence of a solution): Suppose **G** satisfies the Assumptions (1)-(4). There exists an admissible controller **K** such that $\|T_{zw}\|_{\infty} < \gamma$ if and only if [Glover and Doyle (1988), Zhou *et al.* (1996)]:

(a) $\gamma > \max\left(\bar{\sigma}\left[\mathbf{D}_{1111}, \mathbf{D}_{1112}, \mathbf{D}_{1111}^{*}, \mathbf{D}_{1121}^{*}\right]\right)$

(b) $\mathbf{H}_{\infty} \in dom(Ric)$ *with* $\mathbf{X}_{\infty} = Ric(\mathbf{H}_{\infty}) \geq 0$

(c) $\mathbf{J}_{\infty} \in dom(Ric)$ *with* $\mathbf{Y}_{\infty} = Ric(\mathbf{J}_{\infty}) \geq 0$

(d) $\rho\,(\mathbf{X}_{\infty}\cdot\mathbf{Y}_{\infty}) < \gamma^2$, being $\rho\,(\mathbf{X}_{\infty}\cdot\mathbf{Y}_{\infty})$ the spectral radius of $\mathbf{X}_{\infty}\cdot\mathbf{Y}_{\infty}$, (i.e. let $\{\lambda_1, \lambda_2, ..\lambda_n\}$ be the eigenvalues of $\mathbf{X}_{\infty}\cdot\mathbf{Y}_{\infty}$, then $\rho(\mathbf{X}_{\infty}\cdot\mathbf{Y}_{\infty}) = \max_{1\leq i\leq n}|\lambda_i|$)

Given that the previous conditions are satisfied, then all rational internally stabilising controllers **K**(*s*) satisfying $\|T_{zw}\|_{\infty} < \gamma$ are given by **K**=$F_l(\mathbf{M}_{\infty}, \mathbf{Q})$ for an arbitrary **Q** such that $\|\mathbf{Q}\|_{\infty} < \gamma$.

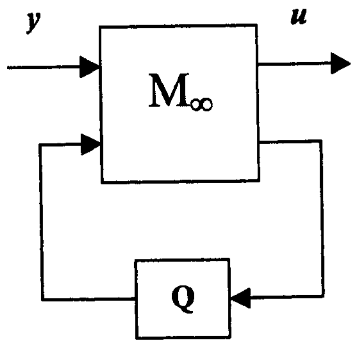


Figure 4.23: LFT Configuration of the Set of Admissible H∞ Controllers that Solve the Suboptimal Problem Parameterised Using the Matrix Q

Thus, $\mathbf{K}$ are observer-based controllers and $\mathbf{M}_\infty$ has the following state-space representation:

$$\begin{aligned}
\dot{\hat{\mathbf{x}}}(t) &= \hat{\mathbf{A}}\hat{\mathbf{x}}(t) + \hat{\mathbf{B}}_1\hat{\mathbf{w}}(t) + \hat{\mathbf{B}}_2\hat{\mathbf{u}}(t) \\
\hat{\mathbf{z}}(t) &= \hat{\mathbf{C}}_1\hat{\mathbf{x}}(t) + \hat{\mathbf{D}}_{11}\hat{\mathbf{w}}(t) + \hat{\mathbf{D}}_{12}\hat{\mathbf{u}}(t) \\
\hat{\mathbf{y}}(t) &= \hat{\mathbf{C}}_2\hat{\mathbf{x}}(t) + \hat{\mathbf{D}}_{21}\hat{\mathbf{w}}(t) + \hat{\mathbf{D}}_{22}\hat{\mathbf{u}}(t)
\end{aligned} \tag{4.48}$$

where:

$$\begin{aligned}
\hat{\mathbf{A}} &= \mathbf{A} + \mathbf{BF} + \hat{\mathbf{B}}_1\hat{\mathbf{D}}_{21}^{-1}\hat{\mathbf{C}}_2 \\
\hat{\mathbf{B}}_1 &= -\mathbf{Z}_\infty\mathbf{L}_{2\infty} + \hat{\mathbf{B}}_2\hat{\mathbf{D}}_{12}^{-1}\hat{\mathbf{D}}_{11} \\
\hat{\mathbf{B}}_2 &= \mathbf{Z}_\infty(\mathbf{B}_2 + \mathbf{L}_{12\infty})\hat{\mathbf{D}}_{12} \\
\hat{\mathbf{C}}_1 &= \mathbf{F}_{2\infty} + \hat{\mathbf{D}}_{11}\hat{\mathbf{D}}_{21}^{-1}\hat{\mathbf{C}}_2 \\
\hat{\mathbf{C}}_2 &= -\hat{\mathbf{D}}_{21}(\mathbf{C}_2 + \mathbf{F}_{12\infty}) \\
\hat{\mathbf{D}}_{11} &= -\mathbf{D}_{1121}\mathbf{D}_{1111}^*\left(\gamma^2\mathbf{I} - \mathbf{D}_{1111}\mathbf{D}_{1111}^*\right)^{-1}\mathbf{D}_{1112} - \mathbf{D}_{1122} \\
\hat{\mathbf{D}}_{12}\hat{\mathbf{D}}_{12}^* &= \mathbf{I} - \mathbf{D}_{1121}\left(\gamma^2\mathbf{I} - \mathbf{D}_{1111}^*\mathbf{D}_{1111}\right)^{-1}\mathbf{D}_{1121}^* \\
\hat{\mathbf{D}}_{21}^*\hat{\mathbf{D}}_{21} &= \mathbf{I} - \mathbf{D}_{1112}^*\left(\gamma^2\mathbf{I} - \mathbf{D}_{1111}\mathbf{D}_{1111}^*\right)^{-1}\mathbf{D}_{1112}
\end{aligned} \tag{4.49}$$

and $\mathbf{Z}_\infty = \left(\mathbf{I} - \gamma^{-2}\mathbf{Y}_\infty\mathbf{X}_\infty\right)^{-1}$.

From Figure 4.23, it can be seen that the solution for the suboptimal control problem is not unique. The admissible controllers that solve the suboptimal problem have been parameterised using the matrix $\mathbf{Q}$. If $\mathbf{Q}$ is chosen to be 0, then the resultant controller is called the *central controller* or *minimum entropy controller* [Zhou *et al.* (1996)].

4.6.2. CONTROLLER IMPLEMENTATION

The implementation process coded in Matlab consisted of the steps represented in the following flow diagram from Figure 4.24 [Alfaro-Cid (2002)]:

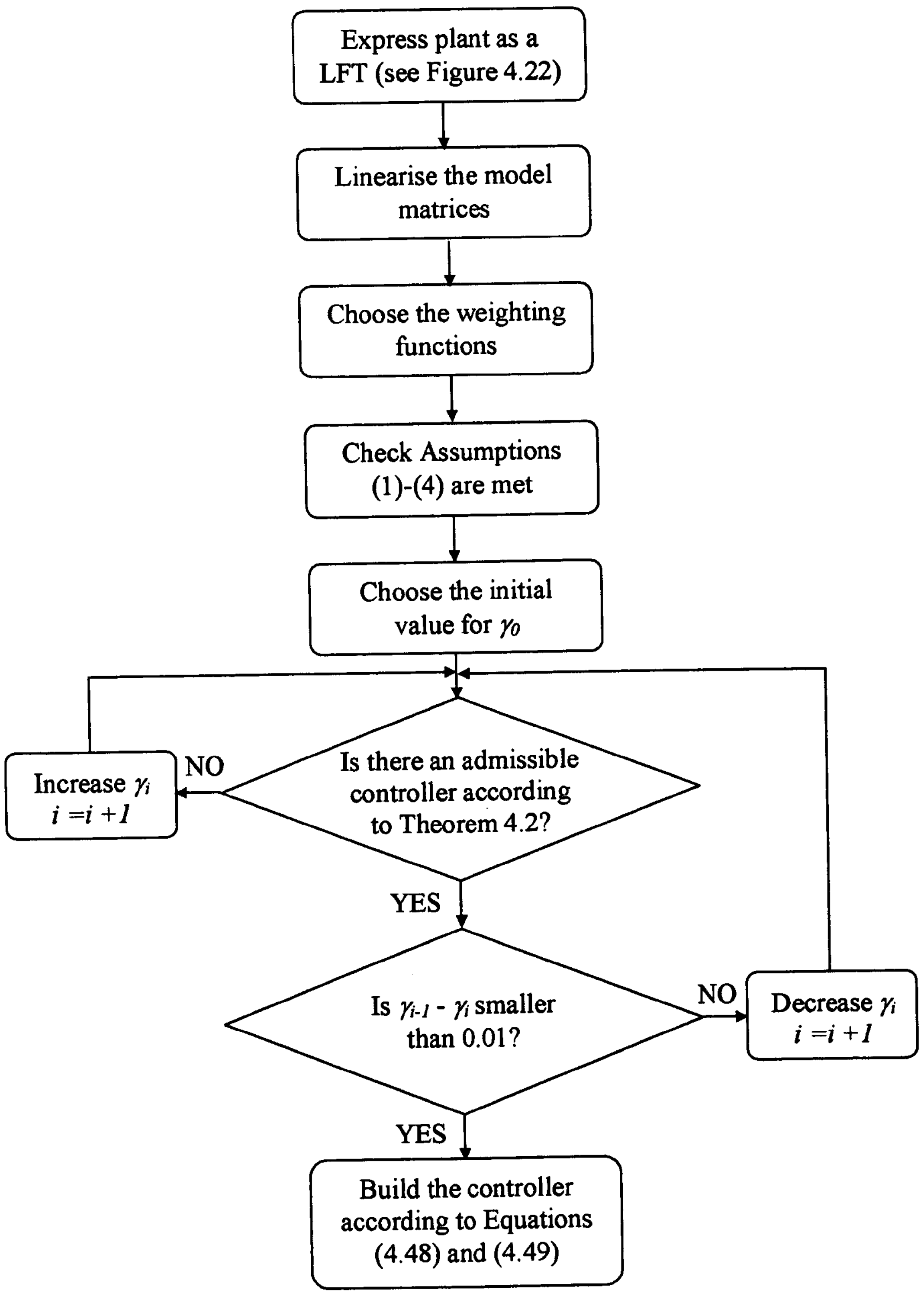

Express plant as a LFT (see Figure 4.22)
Linearise the model matrices
Choose the weighting functions
Check Assumptions (1)-(4) are met
Choose the initial value for γ_0
Is there an admissible controller according to Theorem 4.2?
NO
Increase γ_i $i = i + 1$
YES
Is $\gamma_{i-1} - \gamma_i$ smaller than 0.01?
NO
Decrease γ_i $i = i + 1$
YES
Build the controller according to Equations (4.48) and (4.49)

Figure 4.24: Flow Diagram of the H_∞ Implementation

So that the state-space model matrices of the plant meet the Assumptions (1)-(4) for the existence of an admissible controller it has been necessary to suppress some states in the mathematical model. All the states that denote the position and orientation in the earth-fixed frame (i.e. kinematic states $\eta=[x_p,\ y_p,\ \psi]^T$) have not been used because they introduce a column of zeros in the system matrices and then, the Assumptions (3) and (4) are not met.

Therefore, the choice of the state-space vectors has been as follows:

$$\begin{aligned}\mathbf{x} &= [u,\ v,\ r,\ \tau_1,\ \tau_2,\ \tau_3]^T\\ \mathbf{u} &= [\tau_{1com},\ \tau_{2com},\ \tau_{3com}]^T\\ \mathbf{y} &= [u_c-u,\ r_c-r]^T\\ \mathbf{w} &= [X_w,\ Y_w,\ N_w,\ u_c,\ r_c]\\ \mathbf{z} &= [u_c-u,\ r_c-r,\ \tau_{1com},\ \tau_{2com},\ \tau_{3com}]^T\end{aligned} \tag{4.50}$$

Here u_c and r_c are the reference signals, **x** is the state vector without the kinematic states, **u** is the input vector, **y** is the measurement available to the controller, **w** represents the exogenous input (i.e. forces created by wave disturbances and references) and **z** consists of the signals that are desired to keep small (i.e. surge speed error and yaw rate error as well as the commanded forces).

The state-space matrices follow the structure given by Equation (4.41).

$$\mathbf{A}=\begin{bmatrix}-\mathbf{M}^{-1}(\mathbf{C}(\mathbf{v})+\mathbf{D}) & \mathbf{M}^{-1}\\ \mathbf{0} & -\mathbf{I}\end{bmatrix}$$

$$\mathbf{B}_1=\begin{bmatrix}\mathbf{M}^{-1} & \mathbf{0}\\ \mathbf{0} & \mathbf{0}\end{bmatrix}\quad \mathbf{B}_2=\begin{bmatrix}\mathbf{0}\\ \mathbf{I}\end{bmatrix}$$

$$\mathbf{C}_1=\begin{bmatrix}\mathbf{C}_{11} & \mathbf{0}\\ \mathbf{0} & \mathbf{0}\end{bmatrix}\quad \mathbf{C}_2=[\mathbf{C}_{11}\quad \mathbf{0}] \tag{4.51}$$

$$\mathbf{D}_{11}=\begin{bmatrix}\mathbf{0} & \mathbf{I}\\ \mathbf{0} & \mathbf{0}\end{bmatrix}\quad \mathbf{D}_{12}=\begin{bmatrix}\mathbf{0}\\ \mathbf{I}\end{bmatrix}\quad \mathbf{D}_{21}=[\mathbf{0}\quad \mathbf{I}]\quad \mathbf{D}_{22}=[\mathbf{0}]$$

where $\mathbf{C}_{11}=\begin{bmatrix}-1 & 0 & 0\\ 0 & 0 & -1\end{bmatrix}$

Again, matrix **A** is partly nonlinear, due to the Coriolis effects. It needs to be linearised. The nonlinear part corresponds to:

$$\dot{\mathbf{v}} = -\mathbf{M}^{-1}(\mathbf{C}(\mathbf{v}) + \mathbf{D})\mathbf{v} + \mathbf{M}^{-1}\boldsymbol{\tau} \qquad (4.52)$$

It has been linearised using a Taylor series expansion [Dutton *et al.* (1997)] around an operating point and truncating the higher order partial derivatives in the Taylor series after the first partial derivatives.

In the particular case of H_∞ controllers, the parameters to tune are the weighting functions. This is an extra complication, since the weightings are not just constant values to tune but transfer functions whose structure needs to be optimised as well. Since poorly chosen weighting functions will provide a poor H_∞ controller, the choice of weighting functions must be the prime concern of the designer during the designing process.

While designing the weighting functions it is necessary to keep in mind that the total order of the controller is increased by the order of the weighting functions. High order weights will result in a high order controller. In order to keep the order of the controller as low as possible, only the elements of the **z** vector (i.e. the signal to minimised) are shaped by the weighting functions. Therefore, after including the weighting functions the resulting **z** vector becomes: $\mathbf{z} = [W_u(s)(u_c\text{-}u),\ W_r(s)(r_c\text{-}r),\ W_{t1}(s)\tau_{1com},\ W_{t2}(s)\tau_{2com},\ W_{t3}(s)\tau_{3com}]^T$. For the same reason, simple weighting function structures are considered in this study, i.e. gains, first and second order transfer functions.

The number of poles and zeros of the weighting functions determine the size of the extended state-space model. Obviously, this will lead to a substantial change in the state-space matrices of Equation (4.51). For example, a first order weighting function implies the addition of a new state to the state vector **x**, if it is a second order function two new states are needed in **x**. The addition of zeros in the weighting function does not increase the order of the controller but involves changes in the coefficients of the **A** and **B** matrices.

Given the choice of state-space vectors from Equation (4.50), the LFT structure for the plant plus the weighting functions (i.e. W_u, W_r, W_{t1}, W_{t2}, W_{t3}) plus the H_∞ controller (K) is as shown in the following Figure 4.25:

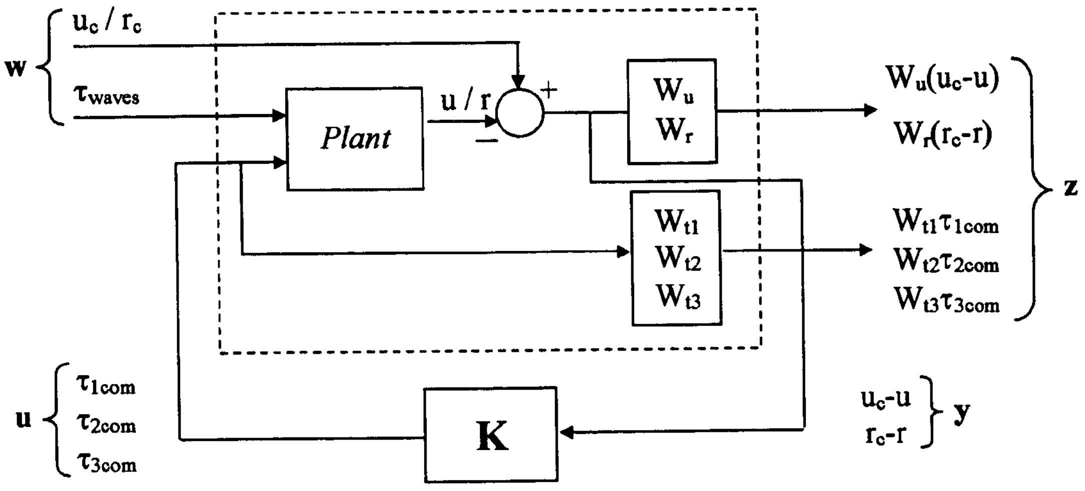


Figure 4.25: LFT Configuration of Plant + Weighting Functions + H_∞ Controller

After the required assumptions have been satisfied, it is a matter of finding, γ_{opt}, the infimum over all γ such that $\|T_{zw}\|_\infty < \gamma$, given that the resulting **K**(s) is an admissible controller. This search has been implemented by means of an iterative loop (see Figure 4.24). Starting with an arbitrary value for γ the Hamiltonian matrices, $\mathbf{H}_\infty$ and $\mathbf{J}_\infty$, derived in Equations (4.44) and (4.45) are calculated. Then, solving the associated ARE we get the values $\mathbf{X}_\infty = Ric(\mathbf{H}_\infty)$ and $\mathbf{Y}_\infty = Ric(\mathbf{J}_\infty)$. Finally, the requirements from Theorem 4.2 must be checked. If they are satisfied, the value for γ is decreased otherwise it is increased. The process is repeated until a value of γ that meets the requirements is found and the difference between this γ and the previous γ that met the requirements is smaller than a tolerance given *a priori*.

Once γ_{opt} has been found the state-space equations of the set of resulting H_∞ controllers can be calculated as it is shown in Equations (4.48) and (4.49). All the resulting controllers will satisfy $\|T_{zw}\|_\infty < \gamma$. In this study the analysis has been kept as simple as possible by choosing Q to be 0, i.e. the central controller. Therefore, the state-space matrices of the resulting controller will be given by $\hat{\mathbf{A}}$, $\hat{\mathbf{B}} = \hat{\mathbf{B}}_1$, $\hat{\mathbf{C}} = \hat{\mathbf{C}}_1$ and $\hat{\mathbf{D}} = \hat{\mathbf{D}}_{11}$ ($\hat{\mathbf{A}}$, $\hat{\mathbf{B}}_1$, $\hat{\mathbf{C}}_1$ and $\hat{\mathbf{D}}_{11}$ as defined in Equation (4.49)).

4.6.3. MANUAL TUNING RESULTS

CyberShip I

Using two second-order transfer functions with zero for W_u and W_r and simple gains for W_{t1}, W_{t2} and W_{t3}, the results obtained in the manual tuning of the H_∞ controller are shown in Equations (4.53) and Table 4.7.

$$W_u(s) = \frac{K_u' \cdot (s+\alpha_u)}{s \cdot (s+\beta_u)} \qquad (4.53)$$

$$W_r(s) = \frac{K_r' \cdot (s+\alpha_r)}{s^2}$$

$$W_{t1} = K_{t1};\ W_{t2} = K_{t2};\ W_{t2} = K_{t2}$$

Table 4.7: Manually Tuned H_∞ Results for CS1

K_u'	α_u	β_u	K_r'	α_r	K_{t1}	K_{t2}	K_{t3}
0.017	0.1	50	200	0.7	0.001	0.1	0.03

The following Figure 4.26 shows the performance of the H_∞ controller while tracking the simultaneous double step manoeuvre.

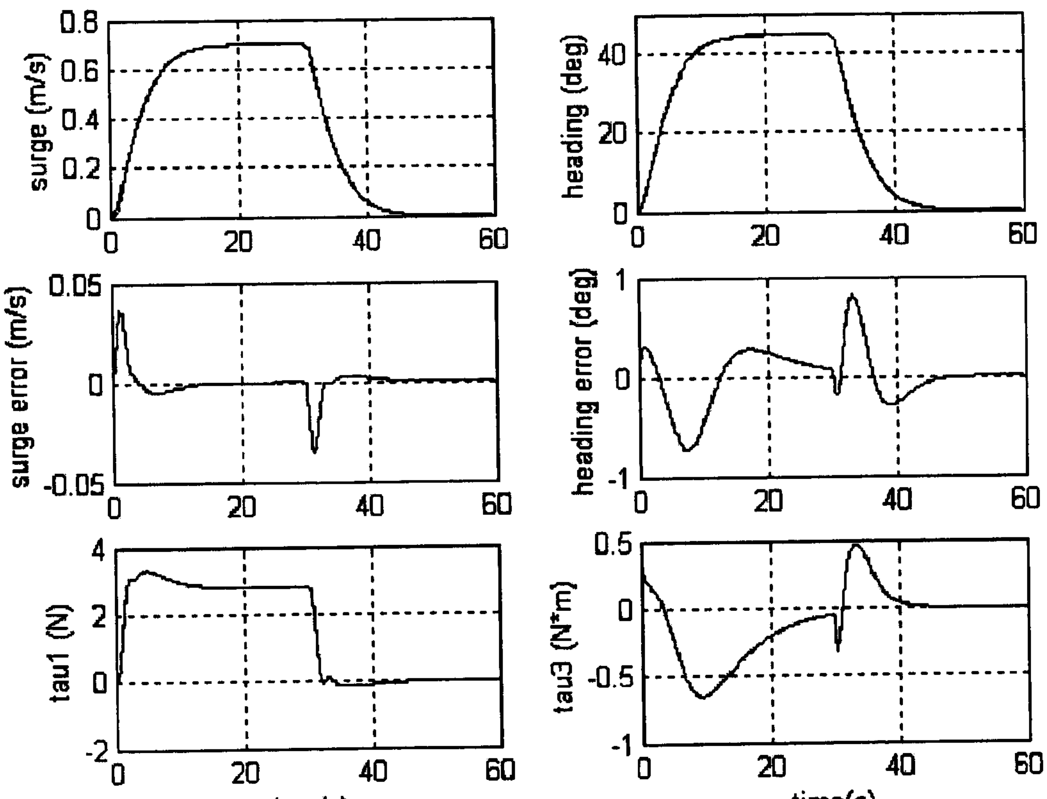


Figure 4.26: Simulated Results of the Manually Tuned H_∞ Controller for CS1

Although the tuning process was very difficult and time consuming the final response shown in Figure 4.26 is quite good. The tracking of the signals is quite accurate (especially in the heading subsystem) and the control actions very smooth as opposed to previous controllers.

CyberShip II

Once more two second-order transfer functions with zero for W_u and W_r and three gains for W_{t1}, W_{t2} and W_{t3} were used as a starting point for the manual tuning. The resulting weighting functions are shown in Equations (4.54) and Table 4.8.

$$W_u(s) = \frac{K_u^{'}}{(s+\beta_u)} \qquad (4.54)$$

$$W_r(s) = \frac{K_r^{'} \cdot (s+\alpha_r)}{s \cdot (s+\beta_r)}$$

$$W_{t1} = K_{t1};\ W_{t2} = K_{t2};\ W_{t2} = K_{t2}$$

Table 4.8: Manually Tuned H∞ Results for CS2

K_u'	β_u	K_r'	α_r	β_r	K_{t1}	K_{t2}	K_{t3}
15	0.001	0.1	1.5	0.01	0.001	0.001	0.001

Figure 4.27 plots the simulated performance of the H∞ controller.

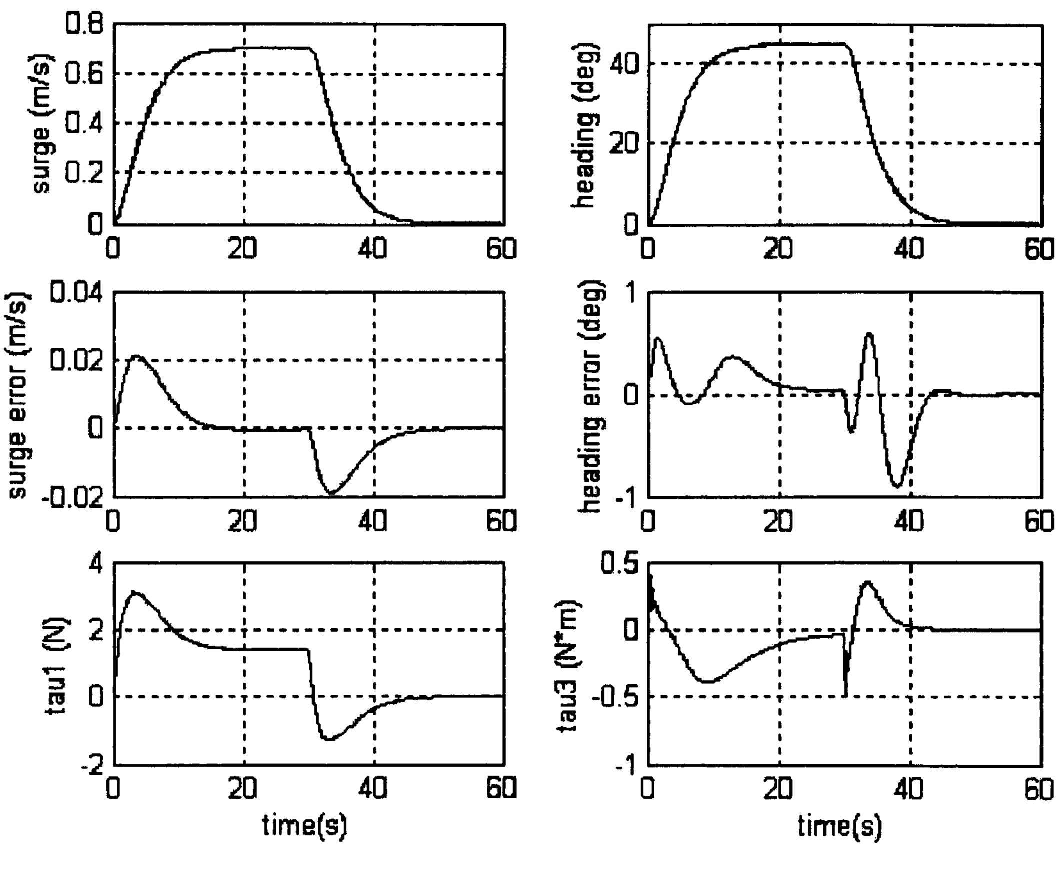


Figure 4.27: Simulated Results of the Manually Tuned H∞ Controller for CS2

Again, H_∞ was extremely difficult to tune, mainly because it is very sensitive to small changes in the weighting functions. Also the number of parameters to consider is very large, despite the efforts trying to keep a simple structure. The final results are quite good. Still, PID and Pole Placement controllers, being based on a far simpler structure, have outperformed them by producing smaller errors.

Figure 4.28 and 4.29 show the results obtained when the manually tuned H_∞ controller is implemented in the real plant. Figure 4.28 shows the responses obtained in still waters.

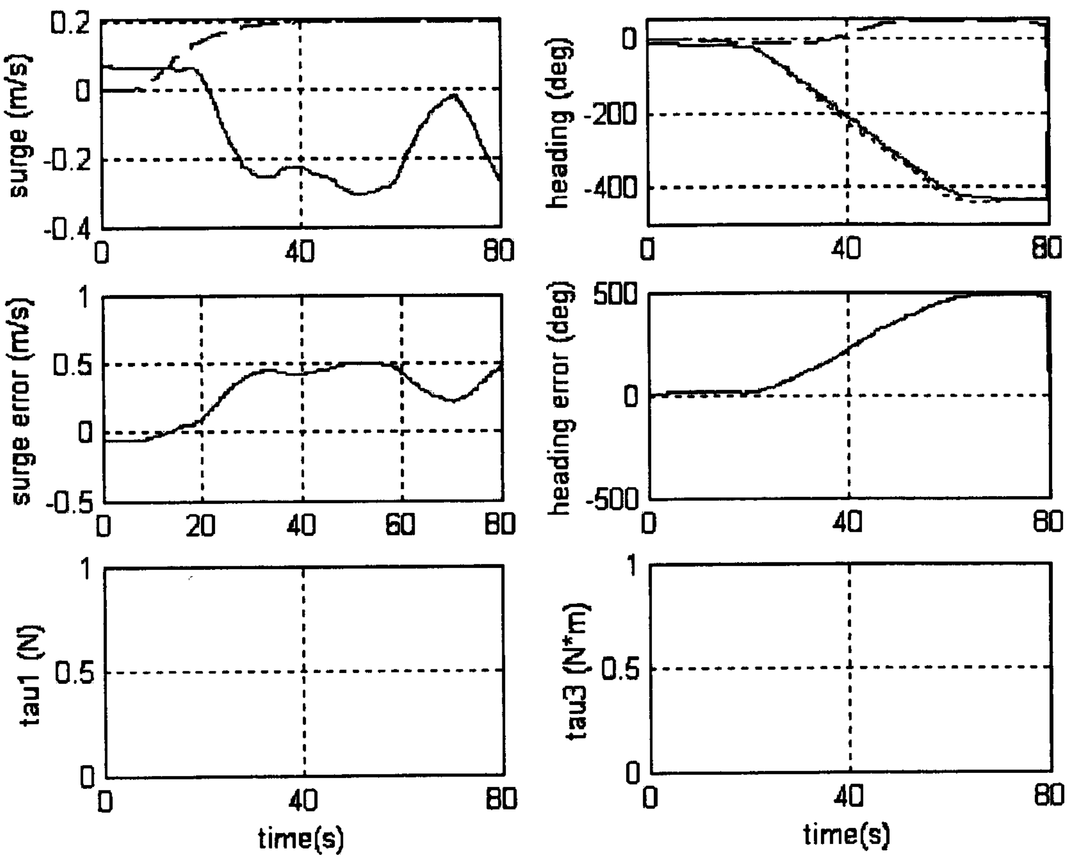


Figure 4.28: Real Results of the Manually Tuned H_∞ Controller When Manoeuvring in Calm Waters

As it can be seen in the figure the controller goes totally unstable. The commanded forces go to infinity and therefore they are not plotted in the figure. The manoeuvre had to be stopped after 80s to avoid any damage to the ship.

As in the previous controllers the performance of the H_∞ controller while tracking the zig-zag manoeuvre used in the real testing has been plotted in Figure F.4 (Appendix F). Since the simulated responses do not show any stability problem, the results raise doubts once more about the accuracy of the model.

Figure 4.29 shows the controller performance when waves are generated.

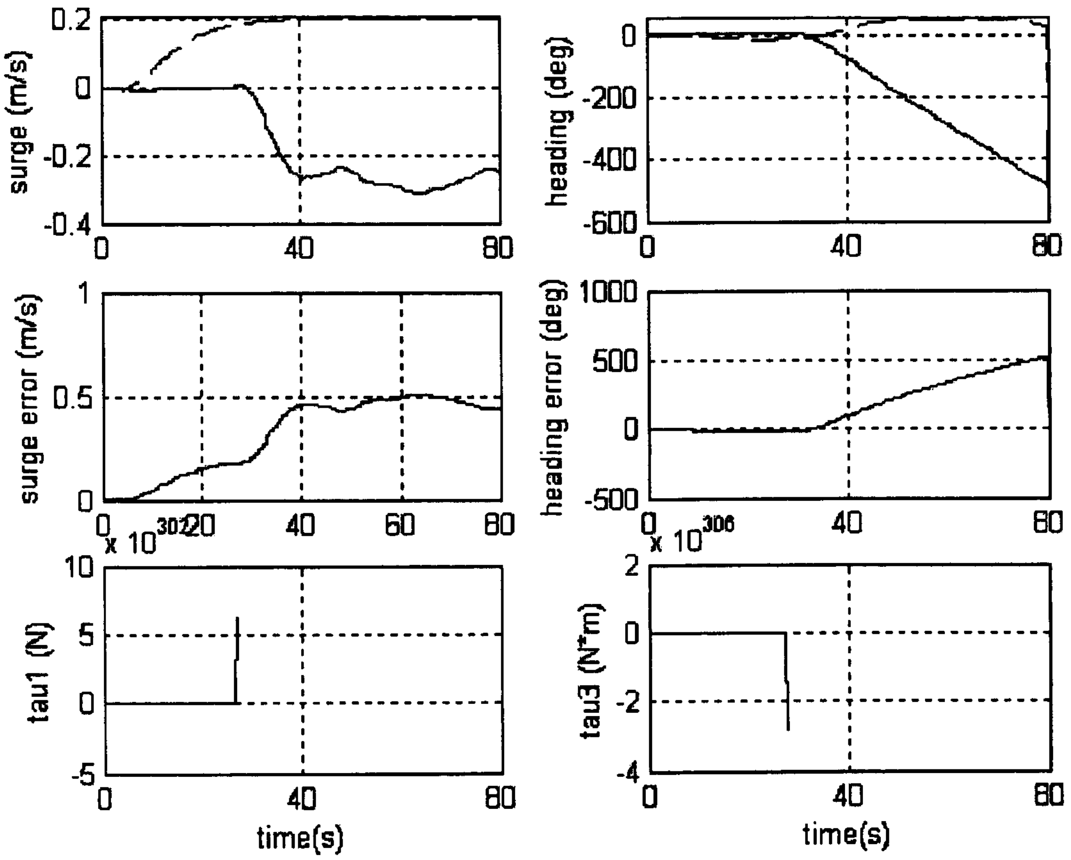


Figure 4.29: Real Results of the Manually Tuned H_∞ Controller When Manoeuvring in the Presence of Waves

As expected the test in the present of waves confirms the stability problems reflected in Figure 4.28.

4.6. SUMMARY

As a precursor to the controller designs presented in this chapter, an analysis using ICAD of the level of coupling between the propulsion and heading dynamics is provided. The analysis has recognised the high level of coupling between sway speed and heading. Regarding the interaction between surge speed and heading, ICAD has proven it to be null in the linearised model. However the coupling due to non-linearities is not considered by ICAD, which is a drawback of the method.

Also, in this chapter the different controller structures used in this study have been presented. The theoretical background of each control methodology as well as the way it has been implemented in the vessel have been introduced.

These four control structures form a very representative group of the main current trends in automatic control (i.e. classical linear control, state feedback, non-linear control and optimal control). In addition, they provide a diversity of problems to test the GA and the future building blocks for the GP structural optimisation.

The four controller designs have been implemented and manually tuned in both CS1 and CS2. CS1 has proven to be more difficult to tune, often resulting in very oscillatory behaviour.

The degree of difficulty of the tuning process varies a lot from one control structure to another. PID and Pole Placement control have been the easiest to tune, while Sliding Mode and H_∞ have been far more difficult. Both are very sensitive to small changes in the parameters, which suggests a peaky search space. In the particular case of H_∞ controllers, the parameters to tune are the weighting functions. This is an added difficulty, since the structure of the weighting functions has to be decided prior to tuning. Although in this study the H_∞ controller structure was kept as simple as possible, the number of parameters to tune was still very large. Moreover the tuning is not very intuitive; there are not well-established performance rules to refer to, as in the PID case. Just to find a set of weighting functions that result in an admissible controller is quite complex.

Some of the real results obtained for the manually tuned controllers raise doubts about the accuracy of the model, since they do not reflect the responses obtained in the simulation studies.

CHAPTER 5

GENETIC ALGORITHMS: A COMPARATIVE STUDY OF GENETIC MODELS

5.1. INTRODUCTION

The performance of automatic controllers depends on the values of the controller parameters. Conventionally, the designer, who attempts to find an acceptable controller solution, manually tunes these parameters. However, this relies on an ad hoc approach to tuning, which depends on the experience of the designer. If the designer is not experienced this process can become tedious and time consuming. In either case there is no guarantee that the designed solution will perform satisfactorily as the tuning process depends on the qualitative judgment of the designer. A solution to this problem is to use optimisation techniques that tune such parameters automatically.

Genetic Algorithms (GAs) [Back (1996), Goldberg (1989), Holland (1975), Michalewicz (1992)] are optimisation techniques that mimic the way species evolve in nature. In natural evolution many organisms evolve by means of two mechanisms: natural selection and sexual reproduction. The Darwinian theory of *survival of the fittest* describes the concept of natural selection [Darwin (1859)]. Sexual reproduction allows the offspring to inherit the features from both its parents.

GAs emulate this process by encoding the points of the search space (called individuals) in a chromosome-like shape and evolving a population of them through a number of generations using mechanisms drawn from natural evolution (i.e. *selection*, *crossover* and *mutation*). The better suited to the optimisation problem an individual is, the more chances it has to survive into the next generation. As the generations progress, this results in the prevalence in the population of stronger solution over weaker ones. Thus, the evolution process tends to near optimal solutions [Goldberg (1989), Holland (1975)].

The way the solutions evolve in a GA is determined by several factors. Among them the implementation and the probability of occurrence of the genetic operators: selection, crossover and mutation. In the literature there are numerous comparison studies that aim to identify the advantages of a certain GA model over others [Back (1996), Brooks *et al.* (1996), Dasgupta (1994)], especially in the beginnings of GA [Brindle (1981), DeJong (1975)]. However, there is not a general agreement on what defines the "best" GA. It is a search space dependent issue.

Since there is not an agreement in the literature about which would be the "ideal" GA, two comparison studies have been carried out in order to assess the dependence of the GA on these factors and to find a GA scheme well suited to the controller parameter optimisation problem. The benchmark problem used in the comparison studies has been a set of 4 controller parameter optimisation problems (i.e. PID, Sliding Mode, Pole Placement and H_∞) being the plant to control a marine vessel (CyberShip I). The main disadvantage of this approach is the long simulation time required. It means that only a restricted number of evaluations are feasible.

Although the optimisation of the controller parameters will be compared and analysed in detail in Chapter 6, this chapter will present an investigation into the "ideal" GA for this study. As an initial starting point Chapter 5 explains the structure and basic operators of the GAs, as well as the conclusions drawn from evaluating different methods of implementing these operators. Overall this will form the basis for constructing the "ideal" GA through the best genetic operator methods for the problem in hand (as defined in Chapter 4).

Also in Chapter 5 a new genetic model called *Structured Genetic Algorithm* (sGA) [Dasgupta and McGregor (1993a)] and its application to structural optimisation is described as a half way approach to the Genetic Programming technique that will be presented in Chapter 7.

Thus, Chapter 5 is divided into five main blocks. Section 5.2 illustrates the structure and basic operators of a Genetic Algorithm. Section 5.3 introduces the main assets of Structured GA (sGA). Section 5.4 presents the optimisation problems to be solved by the GA schemes and the criteria for the analysis of results. Section 5.5 presents the actual GA schemes used in the comparison study and the corresponding results obtained. Finally, Section 5.6 summarises the main points.

5.2. STRUCTURE AND OPERATORS OF A GA

A GA initiates the process of searching by randomly generating an initial population of possible solutions (suitably encoded). The performance of each solution is evaluated using a *cost function*, which is a measure of how well the performance of the solution compares with the desired response. Then, a new generation is produced according to the three main operators of the GA: *selection*, *crossover* and *mutation*. Selection determines which solutions are chosen for mating according to the principal of *survival of the fittest* (i.e. the better the performance of the solution, the more likely it is to be chosen for mating and therefore the more offspring it produces). Crossover allows an improvement in the species in terms of the evolution of new solutions that are fitter than any seen before, and mutation reintroduces values that might have been lost through selection or crossover, or creates totally new features. The cycle is performed until a termination criterion is met (for instance a predetermined number of generations). The following Figure 5.1 describes the basic structure of a GA.

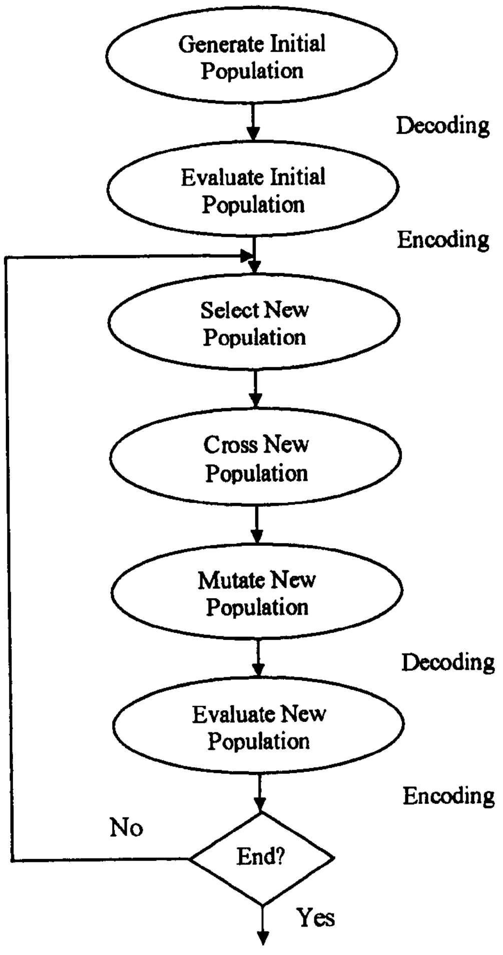


Figure 5.1: Flow Chart of a GA

Over the last three decades, considerable research has focused on improving GA performance (for a review see Srinivas and Patnaik (1994)). Although many of the approaches attempted lack a strong theoretical background, the results have proved successful and, in many cases, performed better than Holland's (1975) pioneer GA (usually called the *Simple Genetic Algorithm* (SGA)).

Below, a description of the genetic operators used in GAs is drafted. It includes a very brief report of the way each operator has been initially depicted by Holland (1975) and later developments. This is not intended to be an exhaustive study of GA operators, just a brief introduction to the mechanisms used in the comparison studies of the next section.

5.2.1. ENCODING

In order to search the space of possible solutions the GA uses a string of digits called a *chromosome* as a representation of the elements of the space of possible solutions (i.e. the possible solutions are suitably *encoded*). Traditionally, GAs have used a binary encoding [Holland (1975), DeJong (1975)]. This binary representation offers the maximum number of *schema* per bit of information of any coding. A schema is a template describing a group of chromosomes that have the same digits at certain string positions [Goldberg (1989), Holland (1975)]. It also facilitates the theoretical analysis and the operation of the genetic mechanisms. However, it has some drawbacks when applied to more complex problems. For example, in the case of problems that require a high precision solution, the length of the encoded binary string is often very big, leading to a very large search space, computer memory problems and poor performance of the GA [Michalewicz (1992), Srinivas and Patnaik (1994)].

Using the binary encoding as a starting point and aiming to reduce the length of the strings, the logical next step is to encode the solutions as an ordered string of octal, decimal or hexadecimal digits [Brindle (1981), Li *et al.* (1996), McGookin (1997)].

In this thesis, each controller parameter value is encoded as a string of five genes [McGookin (1997)]. These genes, instead of being binary bits are decimal integers included within the interval [0, 9]. This allows a wider range of possible solutions (from 0.001×10^{-2} to 9.999×10^{3}) in smaller chromosomes. The encoding mechanism is shown in Figure 5.2

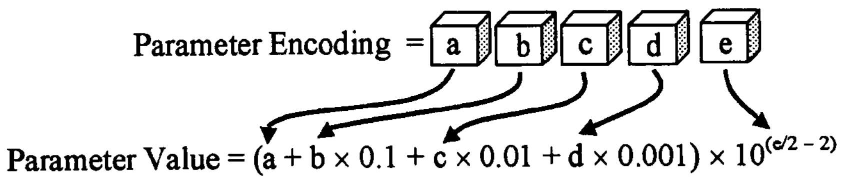

Parameter Encoding =
a
b
c
d
e
Parameter Value = (a + b × 0.1 + c × 0.01 + d × 0.001) × 10^(e/2 - 2)

Figure 5.2: Parameter Encoding Mechanism

5.2.2. COST FUNCTION

Once an initial population of chromosomes is generated at random, the chromosomes are decoded to obtain the corresponding parameters values and these are implemented in the controller. A simulation is run and the controller's performance is evaluated. This is achieved by applying an optimisation design criterion to the simulated responses obtained. For minimisation problems the optimisation design criterion is called *cost function*, whereas for maximisation problems it is called *fitness*. In this study the term used is *cost function*.

The optimisation design criterion used in this study is defined by the cost function (5.1). Since the objective of the controllers is to make the vessel track desired heading and propulsion responses with the minimum actuator effort, the cost function will have three terms for each controller [Alfaro-Cid *et al.* (2001a), (2001b)].

$$C = \sum_{i=0}^{tot}\left[\Delta\psi_i^2 + \lambda_1 \cdot \tau_{3i}^2 + \mu_1\left(\frac{\tau_{3i} - \tau_{3i-1}}{\Delta t}\right)^2 + \Delta u_i^2 + \lambda_2 \cdot \tau_{1i}^2 + \mu_2\left(\frac{\tau_{1i} - \tau_{1i-1}}{\Delta t}\right)^2\right] \tag{5.1}$$

Here, $\Delta\psi_i$ is the ith heading angle error between the desired and obtained heading, τ_{3i} is the ith yaw thrust force, Δu_i is the ith surge velocity error between the desired and obtained surge velocity and τ_{1i} is the ith surge thrust force. Therefore, the quantities $\Delta\psi$ and Δu give an indication of how well the controllers are operating by showing the tracking between the actual and the desired heading and surge velocity and the input components τ_3 and τ_1 are used to keep the actuators movement to a minimum so that they can operate well within their operating limits.

The third and sixth terms of Equation (5.1) introduce a measurement of the inputs increasing or decreasing rates [Alfaro-Cid *et al.* (2001a), (2001b)]. It reduces the oscillations in the inputs, avoiding unnecessary wear and tear of the actuators that shortens their operational lifespan. In the minimisation process these two terms will be also minimised, leading to a smoother input response.

Also, in Equation (5.1), *tot* is the total number of iterations (simulation time steps) and λ_1, λ_2, μ_1 and μ_2 are scaling factors. As the input force and torque are always larger than the output errors near the optimum, they dominate the cost values in this area. It leads to solutions that provide very small thruster effort, but very poor tracking of the desired responses. In order to avoid this, these four coefficients are introduced, so that an equally balanced trade-off between the six terms of the cost function is obtained. The numerical values for these coefficients can be found in the Table B.1 of the Appendix B.

It is a single objective, multi-aspect criterion. Each term from Equation (5.1) represents a different aspect of the optimisation problem. The weighted sum of the 6 terms results in a single objective cost function.

5.2.3. SELECTION

Once the initial population is generated at random, the chromosomes are decoded to get the corresponding parameters and these are introduced in the controllers. A simulation is run and the time responses obtained for each set of controller parameters is evaluated, using the cost function. Based on this cost function the selection procedure takes place [Goldberg (1989), Holland (1975)].

The *selection* scheme is used to draw chromosomes from the evaluated population into the next generation. In the early SGA the selection procedure used was based on *roulette wheel selection* [Goldberg (1989)]. This is a probabilistic method that consists of creating a biased roulette wheel where each chromosome has a roulette wheel slot sized in proportion to its cost. To reproduce, the roulette wheel is spun as many times as there are chromosomes in the population.

The first extensive study and comparison of several variations of the roulette wheel selection was due to DeJong (1975). In addition, in his doctoral dissertation, Brindle (1981) considered some further modifications aiming to achieve a more accurate sampling. The methods they considered relied on assigning a probability of selection to each individual of the population according to its cost function, i.e. all of them were improved versions of the roulette wheel selection basic scheme. In early documents on GA [Brindle (1981)], selection schemes based on relative fitness defined in terms of rank in the population were discarded due to the loss of variance in the population that they imply. However, in recent years has been a growing interest in selection schemes that rely on rank-dependant selection probabilities versus fitness-dependant selection probabilities. Moreover, this interest has been backed up by the good experimental results obtained with these methods [Back (1996), Brooks *et al.* (1996), McGookin (1997), Tan and Li (1996)]

Back's (1996) work identifies four types of selection procedures in Evolutionary Algorithms for single objective optimisation as the most relevant, namely *proportional selection* (or roulette wheel selection), *ranking*, *tournament selection* and *(μ,λ)-selection*, and compares their performance while optimising five different objective functions. Since the proportional selection scheme has already been described, the remaining mechanisms are discussed below.

The term *ranking* denotes a selection method that assigns selection probabilities solely on the basis of the rank of individuals cost values (relative costs), therefore ignoring absolute cost values. The main advantages of ranking over roulette wheel selection are that the selective pressure can be easily controlled in the ranking method and that ranking can speed up the search. Linear ranking assigns the selection probability of an individual in the following way:

$$p_i = \frac{1}{n}\left(\eta - 2(\eta - 1)\frac{i-1}{n-1}\right) \tag{5.2}$$

Here n is the size of the population, i the position of the individual in the ranking and η a parameter that controls the *selective pressure* of the selective scheme ($1 \leq \eta \leq 2$). Selective pressure is the probability of the best individual being selected compared to the average probability of selection of all individuals. Thus, by increasing η the probability of the best individuals being selected is increased too.

The *tournament selection* method selects a single individual by choosing a group of q individuals randomly from the population and selecting the best individual in terms of cost from this group to survive.

The *(μ,λ)-selection* method comes from the field of Evolution Strategies and Evolutionary Programming. Given a parent population size of μ, an offspring of size λ is created via crossover and mutation (being $\lambda > \mu$), then the selection procedure selects the μ best individuals out of the offspring. A variation of this method is the *(μ+λ)-selection* [Back (1996)], where the best μ individuals are selected from the union of parents and offspring (i.e. this is a form of *elitism* [DeJong (1975), Back (1996)]).

In an *elitist selection* technique the best individuals of the population are automatically selected to go to the next generation without undergoing crossover or mutation.

In his experimental investigations, Back (1996) concludes that a general ordering of selections mechanisms according to increasing selective pressure would be: proportional selection, linear ranking, tournament selection and (μ,λ)-selection. His results prove that a strong selective pressure is a desirable feature in a selection mechanism and that rank-based methods, in contrast to fitness-based methods, allow an effective control of selective pressure by just one control parameter: η, as defined in Equation (5.2), q, the tournament group size and the ratio μ / λ, the ratio between the offspring population size and the parents population size.

5.2.4. CROSSOVER

The *crossover* operator combines the features of two parents to create new solutions. One or several *crossover points* are selected at random on each parent and then, complementary fractions from the two parents are spliced together to form a new chromosome [Back (1996), DeJong (1975), McGookin (1997), Michalewicz (1992)]. In Figure 5.3 two points crossover is shown.

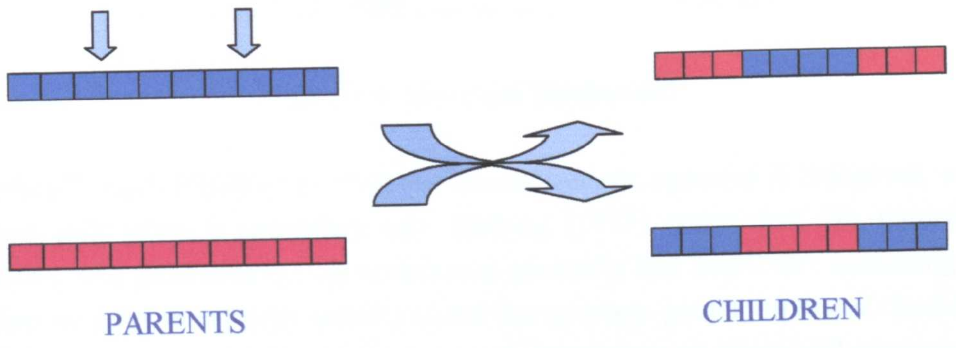


Figure 5.3: Two Points Crossover Mechanism

Variations in the crossover operator in the literature include not only modifications to the operator itself but also to its probability of occurrence.

In his thesis, DeJong (1975) studied the effect of increasing the number of crossover points in the crossover operator concluding that overall performance of the GA degrades as the number of crossover points increases. However, the selection procedures used by DeJong are considered to be very conservative (i.e. the selective pressure was small), so a very disruptive crossover mechanism does not perform satisfactorily in combination with them. Disruptive mechanisms such as multi-point crossover and high mutation rates are used in combination with high selective pressure to avoid premature convergence [McGookin (1997)].

In more recent studies, when stronger selection schemes have been used, the utility of *two-point crossover* or *multiple-point crossover* has been recognised and in fact, two-point crossover is currently a standard implementation [Alfaro-Cid and McGookin (2001), Back (1996), McGookin (1997), Michalewicz (1992)].

5.2.5. MUTATION

The *mutation* operator alters a copy of a chromosome. One or more locations are selected on the chromosome and replaced with new randomly generated values, as shown in Figure 5.4.

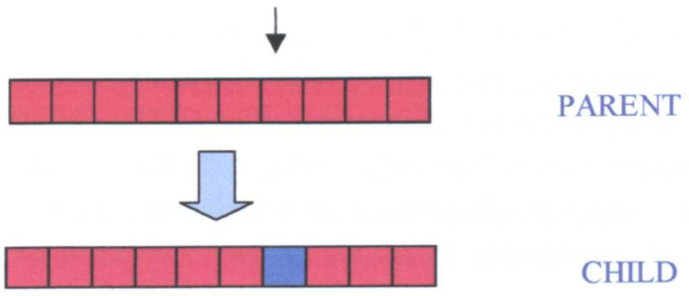


Figure 5.4: Mutation Mechanism

In Holland's work [Holland (1975)] the crucial genetic operator is crossover, while mutation only plays a secondary role. DeJong (1975) wrote that: "In nature the probability of a gene undergoing mutation is generally less that 0.001 indicating that mutation (a form of random search) is not the primary genetic operator. Rather, it should be viewed as a background operator guaranteeing no allele will permanently disappear." Therefore, traditionally mutation was used at a low rate to help ensure that all areas of the search space remain reachable providing higher variation in the chromosomes of each population. It also allows the reintroduction of features that might have been lost during the selection procedure or that have never been in the population.

Holland (1975) theorised that the mutation rate necessary to maintain variance in the population is inversely proportional to the population size. DeJong (1975) settled on a mutation rate of 1/20 the size of the population as a good compromise. Brindle (1981) stated that a high mutation rate improves performance on "difficult" functions while degrading performance on the "easy" ones. He concluded that a probability of mutation of 0.02 or higher is generally undesirable.

Other authors have proved the usefulness of time-varying mutation rates for some GA optimisation problems and hint that, given the helpful effect of high selective pressure in selection mechanisms, a combined increase of selective pressure and mutation rate can improve the search process [Back (1996), McGookin *et al.* (1997c)]. Also, reducing crossover in the latter stages (due to saturation) allows mutation to become more effective [McGookin *et al.* (1997c)].

5.3. STRUCTURED GENETIC ALGORITHMS

Structured Genetic Algorithm (sGA) [Dasgupta and McGregor (1993a)] was first proposed by Dasgupta and McGregor (1991). The sGA differs from the normal GA in the chromosome structure. In sGA the chromosome consists of 2 types of genes: control genes and coefficient genes [Dasgupta and McGregor (1993a)]. The control genes define which coefficient genes will be used in the decoding of the individual, therefore promoting a hierarchy in the chromosome structure. This hierarchy allows the sGA to be suited not only for parametric optimisation but also for structural optimisation.

In this work sGA has been used to optimise the weighting function in the H_∞ control problem. Tang *et al.* (1996) have published a similar application. The main difference between the two approaches is that Tang *et al.* (1996) used a loop-shaping technique where the sGA optimised the precompensator and postcompensator of the weighted plant. In addition, in their work each control gene decides the inclusion or not of a single pole or zero, while in our approach 2 control genes define the whole transfer function structure for optimising the weighting function in the H_∞ control implementation.

The way that sGA has been implemented is by adding 4 extra genes (the "control genes") to the GA chromosome representation for H_∞. Two of these genes specify the structure of the weighting function acting on the yaw rate error while the other two define the structure of the function weighting the surge error signal.

This representation allows 4 options: a constant gain (when the control genes are encoded as 00), a gain plus a pole (01), a gain plus a pole plus a zero (10) and a gain plus 2 poles plus a zero (11), all of them variations of the previous 2nd order transfer function plus zero. Depending on the structure chosen the number of parameters needed to define the transfer function varies. Then the control genes activate or deactivate the parameter genes according with the weighting structure reflected in them.

5.4. OPTIMISATION PROBLEMS AND RESULTS ANALYSIS CRITERIA

5.4.1. OPTIMISATION PROBLEMS

In this work, two comparison studies of GA schemes have been undertaken. In the first comparison study, the relevance of changes in the mutation and crossover probabilities and in the choice of the selection operator is assessed. The best result is then used as a benchmark for the second comparison study where the genetic operators are modified in an attempt at improving the performance of the GA.

Each GA scheme has been applied to the 4 different controller parameter optimisation problems presented in Chapter 4. The plant to control in this study is CyberShip I and the controller structures are: two decoupled PID controllers, a decoupled Sliding Mode controller for heading plus a PI controller for propulsion, a MIMO H_∞ controller and, finally, a MIMO Pole Placement controller. The difficulty of the optimisation problem varies from the quite simple problem of tuning two decoupled PID controllers to the very complicated optimisation of the weighting functions for the MIMO H_∞. As explained in Chapter 4 this selection of controllers covers the most representative areas in current control research. Thus, it allows a good comparison among the GA schemes proposed. In the following Table 5.1 the parameters to optimise for each control structure are presented.

Table 5.1: Parameters to Optimise for each Controller Configuration

PID	K_{Ph} K_{Ih} K_{Dh} K_{Pp} K_{Ip} K_{Dp}
PP	$pole_1$ $pole_2$ $pole_3$ $pole_4$
SM+PI	$pole_1$ $pole_2$ η ϕ K_{Pp} K_{Ip}
H_∞	K_u' α_u β_u γ_u K_r' α_r β_r γ_r K_{t1} K_{t2} K_{t3}

Each parameter has been encoded using the method proposed in Figure 5.2. Thus, each possible solution is represented by a chromosome with a number of genes equal to 5 genes/parameter $\times$ No of Parameters. For instance, each solution for the PID problem is encoded as a 30 genes chromosome.

Regarding the evaluation of solutions, Equation (5.1) is the cost function used. In addition to this there is a desired response that the controller must track. The desired heading and propulsion manoeuvres used in the GA optimisation of the controllers are the same for all of them: two under damped steps of 45° and 0.2 m/s for heading and propulsion control, so that the basis for comparison is consistent for this study.

The population size used is 80 and the number of generations 50 [Grefenstette (1986)]. Every GA optimisation has been run 6 times and the results have been averaged.

5.4.2. RESULTS ANALYSIS CRITERIA

The aim of a GA is to converge to a near optimal solution in the smallest possible number of generations. This fast convergence feature is only desirable in combination with a robust performance (i.e. premature convergence to a local optimum must be avoided).

Therefore, the attributes that are required for an effective GA are: quality of result (i.e. near optimal performance) and speed of convergence. In addition it is interesting to check the amount of diversity in the final population to see if there is room for improvement.

The criteria used to compare the GAs performance are:

1. Best cost overall (in any generation)
2. First generation whose best cost is only 10% bigger that the best cost of the final generation (*generation of convergence)*
3. Number of individuals in the final population whose cost is only 10% bigger that the best individual (*amount of convergence)*
4. Average Maximum Frequency (*AMF*)

AMF is a measure of the fixation of loci in the last generation (i.e. how similar the individuals of the last generation are, not regarding their cost functions or phenotypes but their genotype). AMF is a parameter proposed by Brindle (1981) for convergence comparison purposes. This is defined using the following Equation (5.3):

$$AMF = \frac{1}{L}\sum_{i=1}^{L}\max_{j=1..B}\left(h_{ij}\right) \qquad (5.3)$$

Let L be the length of a chromosome, B the number of possible alleles and h_{ij} the proportion of the chromosomes in the final population which contain the allele j in the position of the chromosome i.

Thus, criterion 1 relates to the quality of the solution, criterion 2 has to do with convergence speed and criteria 3-4, with convergence among the final population.

5.5. COMPARISON STUDY OF VARIOUS GA SCHEMES PERFORMANCE

5.5.1. FIRST COMPARISON STUDY

GA Schemes Under Study

It seems to be clear in the literature that rank-based selection methods (as opposed to fitness based selection methods such as *roulette-wheel*) are desirable because they provide a faster and better performance. The selection methods chosen to be compared in this study all come from the field of GAs, and are *ranking selection*, *tournament selection* and *elitism* combined with *roulette-wheel* (see Section 5.2). Although Back (1996) studies in depth the performance of the (μ, λ)-selection, in this work elitism is considered as this has been used more in the GA community [Brooks *et al.* (1996), McGookin (1997)].

All three selection procedures contain a control parameter that permits an effective management of their selective pressure. The following Table 5.2 shows the control parameters chosen for each selection method.

Table 5.2: Selection Methods Control Parameters

Ranking selection	$\eta = 1.8$
Tournament selection	$q = 8$
Elitism	$\rho = 10\%$

In *linear ranking* the control parameter is η, the maximum expected value, being $1 \leq \eta \leq 2$. In *tournament selection*, q represents the size of the group of individuals chosen to fight for survival, while ρ is the percentage of the population chosen to survive straight into the next generation in the *elitism* strategy. In all three cases the bigger the control parameter the higher the selective pressure associated to the selection mechanism.

The values presented in Table 5.2 for ranking and tournament selection have been chosen following the results obtained by Back (1996). Although the results vary from one function to another, $\eta = 1.8$ and $q = 8$ have a good overall performance. In the literature the number of individuals to survive to the next generation using the elitist strategy varies from a single individual [Eiben and Schoenauer (2002)] to 20% of the population [Brooks *et al.* (1996), McGookin (1997)], therefore in this study a percentage of elitism of $\rho = 10\,\%$ has been chosen.

Regarding crossover, 3 different probabilities of crossover have been considered. Commonly proposed settings for the crossover probability are $p_c = 0.6$ [DeJong (1975)] and $p_c \in [0.75, 0.95]$ [Back (1996)]. So, in this study, *two-point crossover* with $p_c = 0.6$ and $p_c = 0.8$ has been used and compared. In addition, since other approaches to Evolutionary Algorithms, namely Evolution Strategies and Evolutionary Programming, rely on mutation as the prime operator (as opposed to crossover), the possibility of a Genetic Algorithm with a zero crossover probability (i.e. $p_c = 0$) has been contemplated. The aim is to check how the performance of the GA varies when crossover is withdrawn.

Mutation has to be kept high to prevent premature convergence since the selection schemes have a very high selective pressure. The mutation probabilities chosen for comparison are $p_m = 0.05$ and $p_m = 0.1$.

Therefore, in the following Table 5.3, a record of the GAs schemes used is shown:

Table 5.3: GA Schemes Chosen for Comparison

	Selection procedure	**p_c**	**p_m**		**Selection procedure**	**p_c**	**p_m**
GA1	ranking	0	0.05	**GA10**	ranking	0.6	0.1
GA2	tournament	0	0.05	**GA11**	tournament	0.6	0.1
GA3	elitism	0	0.05	**GA12**	elitism	0.6	0.1
GA4	ranking	0	0.1	**GA13**	ranking	0.8	0.05
GA5	tournament	0	0.1	**GA14**	tournament	0.8	0.05
GA6	elitism	0	0.1	**GA15**	elitism	0.8	0.05
GA7	ranking	0.6	0.05	**GA16**	ranking	0.8	0.1
GA8	tournament	0.6	0.05	**GA17**	tournament	0.8	0.1
GA9	elitism	0.6	0.05	**GA18**	elitism	0.8	0.1

First Comparison Study Results

Criterion 1: Best Cost Analysis

Figure 5.5 shows the best cost results obtained for each GA scheme of the first study. Note that the scales in the Y-axis vary greatly from one controller to another. The dotted line represents tournament selection; the solid line represents ranking selection and the dashed line, elitism. The average cost values over 6 runs have been plotted against the different combinations of mutation and crossover probabilities shown in the X-axis. See Table B.2 in Appendix B for the numerical data.

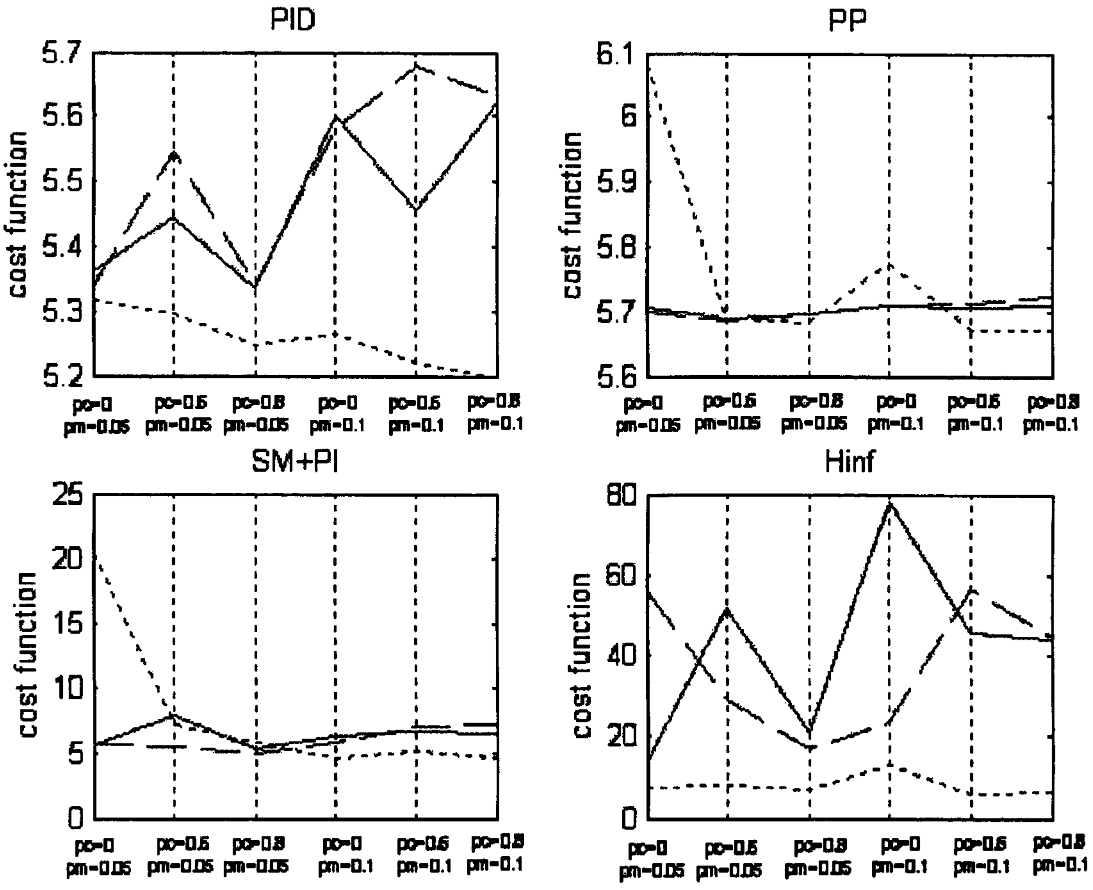


Figure 5.5: Best Cost Results for all GA Scheme for each Controller (dotted line: tournament selection, solid line: ranking selection, dashed line: elitism)

It is very notable how GA schemes using tournament selection consistently obtained better results than ranking or elitism in 19 of the 24 possible combinations of optimisation problem, crossover rate and mutation rate. Moreover, in the case of the H_∞ optimisation problem (the most difficult one), the results obtained by tournament selection are remarkably better than those obtained using elitism or ranking selection.

The best result overall has been obtained by GA17, tournament selection with p_c = 0.8 and p_m = 0.1 (i.e. high crossover and mutation rates). This GA scheme obtained the best result for PID and Pole Placement and second best for Sliding Mode and H_∞. The standard deviations values for GA17 are very small for PID and Pole Placement (σ = 0.0082 and σ = 0.0013 aprox.), indicating consistency in the runs. They are slightly higher for Sliding Mode and H_∞ (σ = 0.4655 and σ = 1.5612). This is due to the increasing difficulty of the optimisation problem to solve.

In general, it is very difficult to draw conclusions from these data about which selection method performs better: elitism or ranking selection. For the easier optimisation problems (i.e. PID and Pole Placement), the differences between the results obtained by any of them for given crossover and mutation rates are slight. While considering Sliding Mode and H_∞ optimisation problems the differences increase. Elitism seems to tend to perform better than ranking selection, especially when the mutation rate is low, although that is not true for all cases.

Focusing on the mutation probability, given an arbitrary crossover probability, tournament selection works better with higher mutation rates (i.e. p_m = 0.1). However, given a crossover probability, ranking selection and elitism perform usually better with a lower mutation rate (i.e. p_m = 0.05). This was quite unexpected; given the higher selective pressure of the elitism selection a higher mutation rate was expected to work better in order to avoid premature convergence. Anyway, as mentioned previously, the differences between the cost results obtained when varying the crossover and mutation rates for ranking selection are slight.

Regarding the importance of the crossover operator the unexpectedly good performance of p_c = 0 is significant. However, tournament selection performs generally better with a higher crossover probability. It is not possible to draw clear conclusions for the other two selection schemes since their performance varies from one optimisation problem to another. This might suggest that the crossover probability, although it plays a role, is not as important as it is believed to be in the SGA. This result is consistent with the work developed by Grensfetette (1986). He found out that for populations of 30 to 90 individuals the optimal crossover probability decreases as the population size increases.

Criterion 2: Generation of Convergence Analysis

In Figure 5.6 the generations needed for convergence (as defined in Section 5.4.2 and averaged over 6 runs) obtained for each GA scheme are shown. Again the dotted line represents tournament selection; the solid line represents ranking selection and the dashed line, elitism. See Table B.3 in Appendix B for the numerical data.

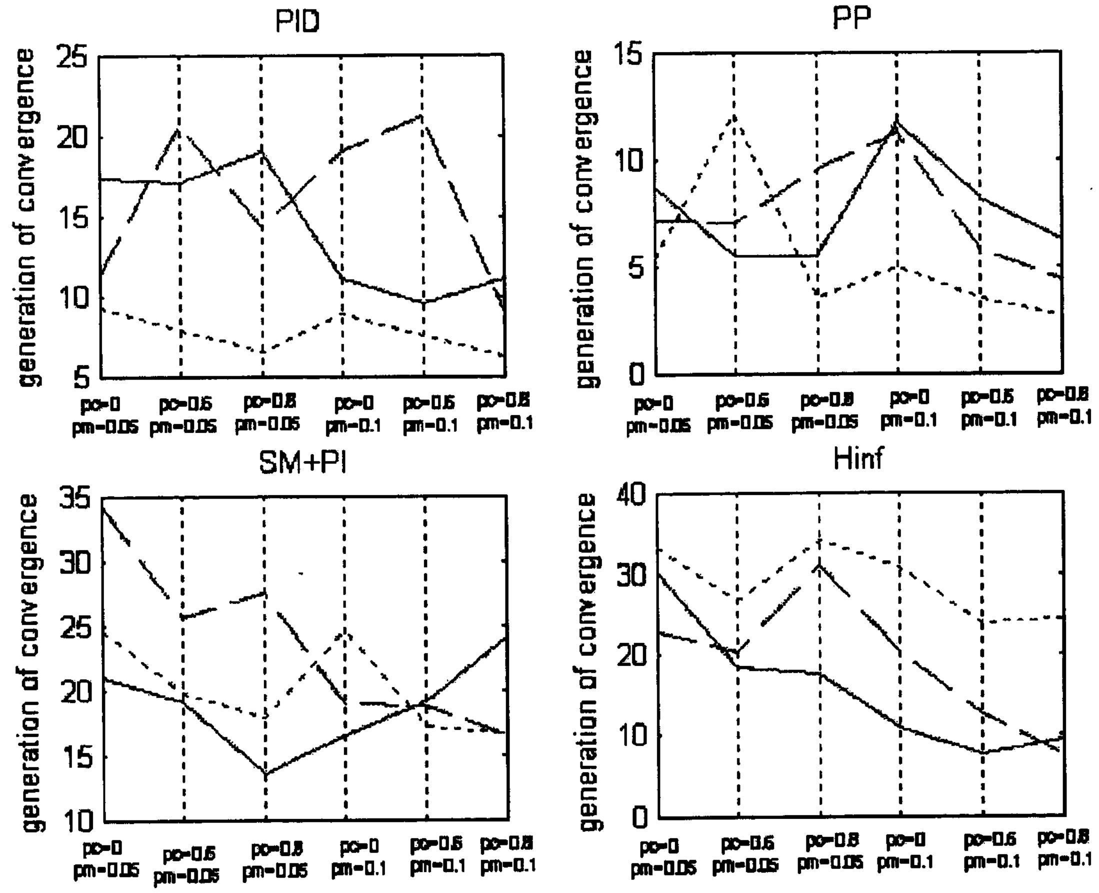


Figure 5.6: Generation of Convergence for all GA Scheme for each Controller (dotted line: tournament selection, solid line: ranking selection, dashed line: elitism)

The first aspect that stands out when checking the Y-axis scales from the previous figure is how the number of generations needed for convergence increases with the difficulty of the optimisation task.

The analysis of the generations of convergence data shows very different results for the different optimisation problems. For the PID and Pole Placement optimisation problems, those GA schemes with tournament selection have an earlier generation of convergence. In the H_∞ case the situation is totally different. The earlier generations of convergence are achieved by the GA schemes with ranking selection and elitism, while the slowest ones are the six GA schemes with tournament selection. Paradoxically, these slow GA schemes provided the best results for the complicated H_∞ optimisation, while the fast schemes obtained particularly bad results. Clearly, the later cases are incidents of *premature convergence* to a local minimum. So far in the analysis, tournament selection has proved to be fast in easy optimisation problems (such as PID and Pole Placement) while avoiding premature convergence in more difficult ones and therefore maintaining a robust performance.

Regarding the relationship between mutation and crossover rates and the generation of convergence, it can be seen that, in the GAs with tournament selection, increasing crossover and mutation rates results in an increase in the speed of convergence.

Criterion 3: Amount of Convergence in the Final Population Analysis

In the following Figure 5.7 the amount of convergence of the final population (as defined in Section 5.4.2 and averaged over 6 runs) obtained for each GA scheme are shown. See Table B.4 in Appendix B for the numerical data.

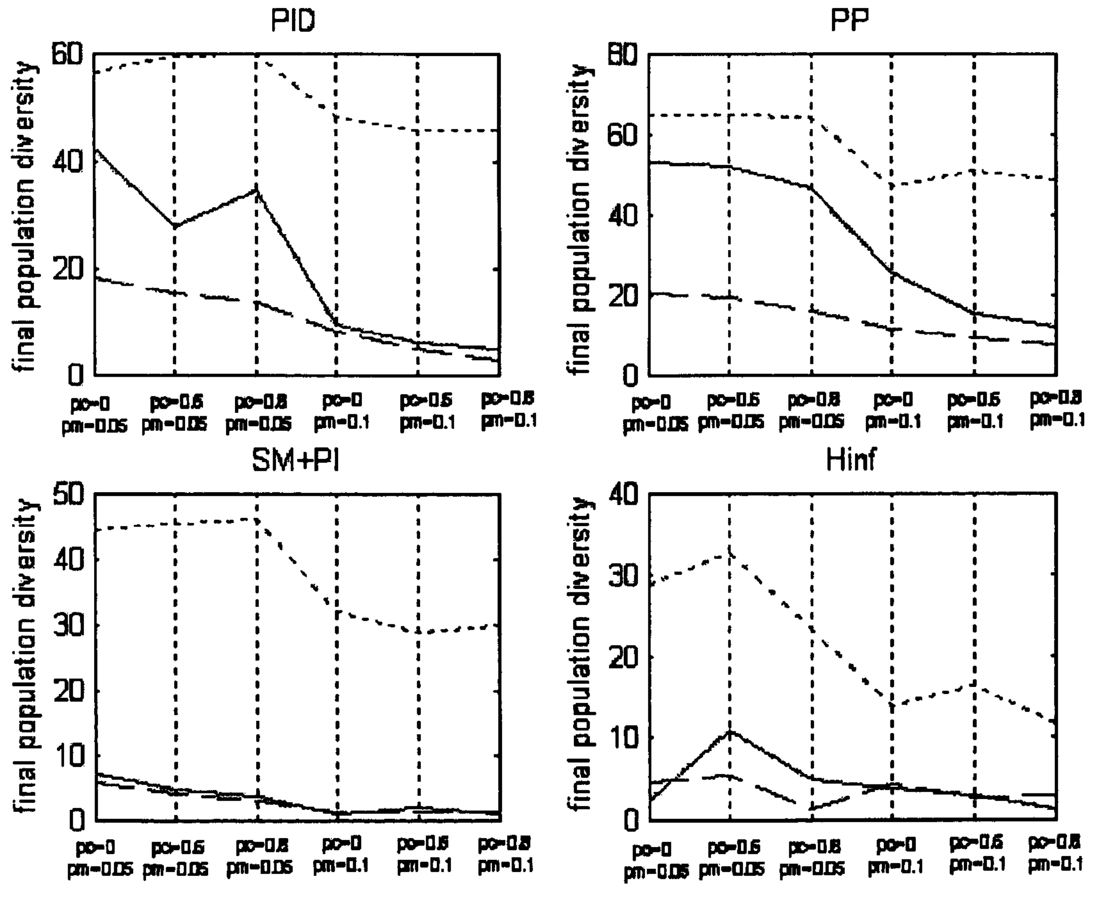


Figure 5.7: Amount of Convergence in the Final Population
(dotted line: tournament selection, solid line: ranking selection, dashed line: elitism)

The amount of convergence illustrates the saturation on higher echelons of the population with solutions with very similar cost values. Smaller convergence rates are observed for more difficult problems. This is due to the fact that the sensitivity of Sliding Mode and H_∞ controllers to small changes in the parameters causes big differences in the cost values even for similar individuals. The GA schemes with tournament selection have the highest amount of convergence. This indicates that the population is saturated with individuals alike and there is not enough diversity for much improvement.

Comparing ranking selection and elitism, it can be seen that the latter obtains smaller convergence rates throughout different combinations of crossover and mutation rates and different optimisation problems. This indicates that, although the results obtained with elitism schemes are not very good, there is room for improvement. Since it has enough diversity in the population, given more time the final solution could improve.

Focusing now on the effect of mutation and crossover probabilities, it is very apparent from the figure that, given an arbitrary crossover probability, increasing the mutation rate leads to a lower convergence rate (i.e. a greater diversity). This is to be expected since the prime role of mutation is introducing new features in the population, which obviously increases variety.

Given an arbitrary mutation rate, increasing the crossover probability also seems to reduce the convergence rate (i.e. enhances diversity), although the data are not as conclusive as in the mutation case.

Criterion 4: AMF

In Figure 5.8, the Average Maximum Frequency of loci fixation in the final population (as defined in Section 5.4.2 and averaged over 6 runs) obtained for each GA scheme are shown. See Table B.5 in Appendix B for the numerical data.

Figure 5.8 shows that the AMF values in the final populations are very similar for all the controllers. This proves that the smaller amounts of convergence in the final population for Sliding Mode and H_∞ pictured in Figure 5.7 are not due to a bigger diversity but to the fact that even individuals with a similar genotype result in quite different cost values once evaluated.

The analysis of the AMF of the final population further confirms the conclusions obtained in the previous analysis. Again the highest AMF values correspond to the

GA schemes that use tournament selection. This indicates a high proportion of loci fixation in the final population and therefore less diversity. For a second time, the results prove that ranking selection has less variety (i.e. higher AMF values) than elitism.

Once more the data corroborate that, given a certain crossover probability, the schemes with higher mutation rate show a smaller AMF (i.e. bigger diversity). In the case of varying the crossover probability while keeping a fixed mutation probability, the AMF reduces when p_c increases as seen in the previous case, although the reduction is not as significant as in the mutation case. Again, it makes sense that a higher crossover rate creates a higher diversity.

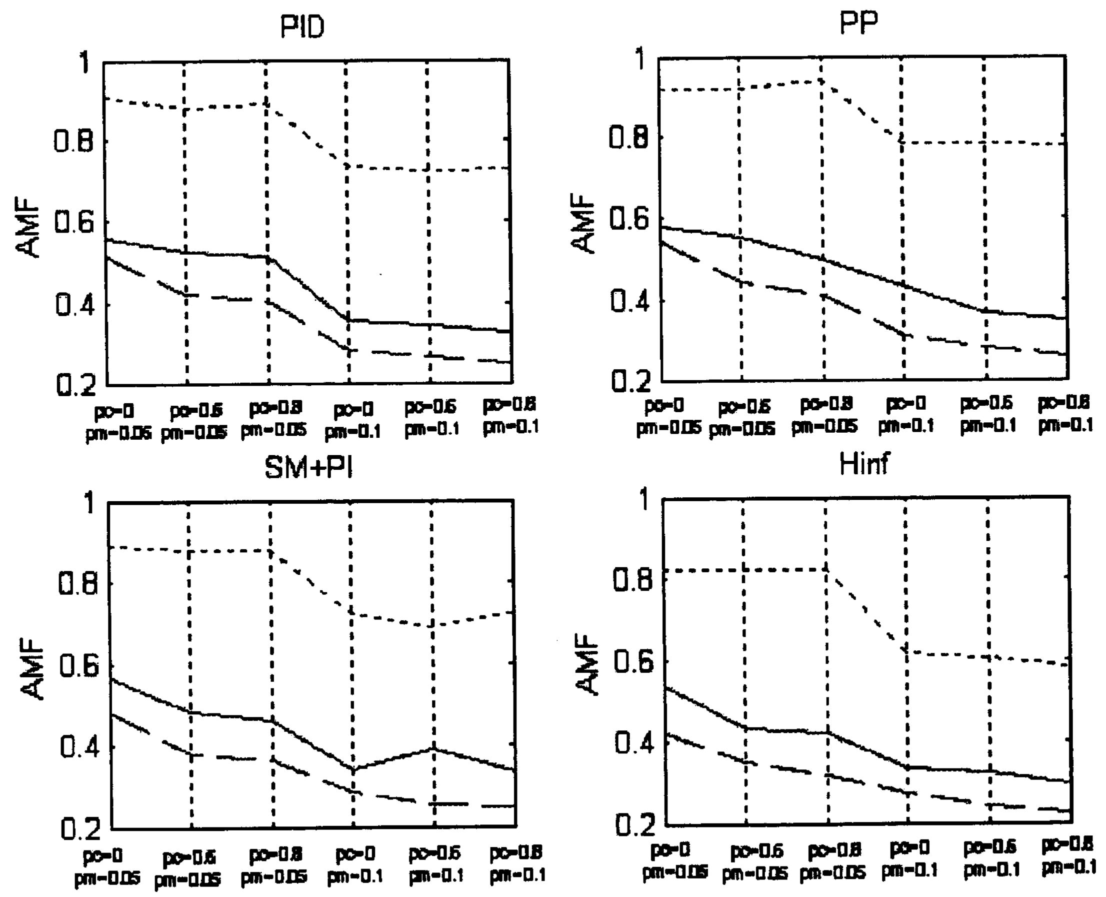


Figure 5.8: AMF in the Final Population (dotted line: tournament selection, solid line: ranking selection, dashed line: elitism)

Summary of Results from First Study

The selection method found to be the best according to the chosen set of criteria has been tournament selection. Tournament selection has provided consistently the best overall results to the control optimisation problems and has proved to be the fastest selection procedure in easy optimisation problems while avoiding premature convergence in more difficult ones and therefore maintaining a robust performance.

Regarding ranking selection and elitism the results obtained are quite good for the easier optimisation problems but not acceptable for the more complicated ones. However, elitism rates of convergence are quite small so that could indicate that, given more time, there is still enough diversity in the population for improvement.

Although the best results are obtained with higher probabilities of mutation and crossover, GA schemes with $p_c = 0$ performed reasonably well. This might suggest that crossover, although it plays a role increasing the speed of convergence, is not as important as believed to be in the SGA.

5.5.2. SECOND COMPARISON STUDY

GA Schemes Under Study

One of the main drawbacks when working with GAs is the difficulty they display in performing local search, i.e. *fine local tuning*. To improve the capabilities of GAs in this area researchers have considered different approaches.

One way to tackle the problem has been to consider variable mutation and crossover rates along the generations [Srinivas and Patnaik (1994)]. In this research line there is a very interesting method of *non-uniform mutation* proposed by Michalewicz (1992).

A totally different strategy is presented by McGookin *et al.* (1997c). It is a minimisation strategy that removes the redundancy associated with the saturation effect found in later generations. As the population size reduces, the number of crossover operations decreases and the apparent mutation rate increases, improving the search process in the latter stages of the GA optimisation.

From the different approaches considered by researchers to improve the capabilities of GAs in this area, the *non-uniform mutation* proposed by Michalewicz (1992) and the *minimisation strategy* proposed by McGookin *et al.* (1997c) have been implemented among others to assess their capabilities for improving local tuning.

Using the GA scheme that proved to work best in the previous comparison study as a benchmark, several modifications (to be exact 7) have been tested and their capabilities to enhance the performance of the GA compared.

Minimisation Technique

This minimisation technique is proposed by McGookin *et al.* (1997c). It relies on the reduction of the population size. Once a certain percentage of the individuals of the population, k, are very similar (i.e. their cost functions differ in less that a certain tolerance), a number of individuals, r, are eliminated from the population.

These r individuals are chosen so that half of them are the best $r/2$ individuals in the population and the other half are the worst $r/2$ individuals in the population. This way the cost balance of the population is maintained. In addition a minimum value for the population size is set so that once it is reached there is no further minimisation.

During this process the number of genes to be mutated is calculated from the initial population size and kept constant over the generations. Thus, the mutation rate increases with the population reduction, which compensates for the loss of features produced by the population size reduction. On the other hand, since the crossover rate is kept constant the reduction in the population size results in a reduction in the number of crossover operation taking place.

This algorithm has been implemented with two different sets of values. First, the values chosen were $k = 0.1$ and $r = 0.1$. In a second group of simulations k was kept at a value of 0.1 but r was set to 0.05 to avoid the performance degradation associated to a very small population.

Non-Uniform Mutation Technique

This technique is based on a non-uniform operator proposed by Michalewicz (1992). The idea is that when a gene is selected for mutation, the mutation jump it suffers depends on the time of the optimisation when this mutation happens. The sooner in the optimisation (i.e. the early the generation), the bigger the mutation jump. This way in the first generations exploration is encouraged while in the final generations, when convergence has occurred, small mutation jumps improve the fine local tuning.

If a gene is chosen for mutation, it will be assign one of the values shown in Equation (5.4) with a 50% probability:

$$newgene = \begin{cases} gene + \Delta(t, 9 - gene) \\ gene - \Delta(t, gene) \end{cases} \tag{5.4}$$

The function $\Delta(t,y)$ returns a value in the range $[0, y]$ such that the probability of $\Delta(t,y)$ being close to 0 increases as t increases:

$$\Delta(t,y) = y\left(1 - r^{(1-t/T)^b}\right) \tag{5.5}$$

Here r is a random number from $[0, 1]$, T is the maximal generation number and b is a system parameter determining the degree of non-uniformity.

The only difference between the non-uniform mutation technique implemented in this research and the non-uniform operator proposed by Michalewicz (1992) is the non-randomness of the parameter r.

There are genes in the coding that have a higher weight than others, i.e. a small variation in one of these genes is reflected in a big variation in the parameter to optimise once the individual is decoded. In the coding used in this research, the genes carrying a higher weight are genes number 1 and 5, while changes in genes 3 and 4 are less relevant (see Figure 5.2).

The non-uniformity of the mutation in this case consists of two effects. On one hand the mutation jump varies depending on the generation number. On the other hand it also varies depending on the position in the chromosome of the gene to be mutated.

In the implementation, the number r is not chosen at random but the choice of r is made as shown in Figure 5.9. In the initial populations higher mutation jumps are associated with the gene positions 1 and 5, resulting in big variations in the phenotype that help in the exploration of the search space. As the generations progress the effect is inverted and the biggest amounts of mutation are associated with the genes 3 and 4, resulting in closer phenotypes that improve the fine tuning.

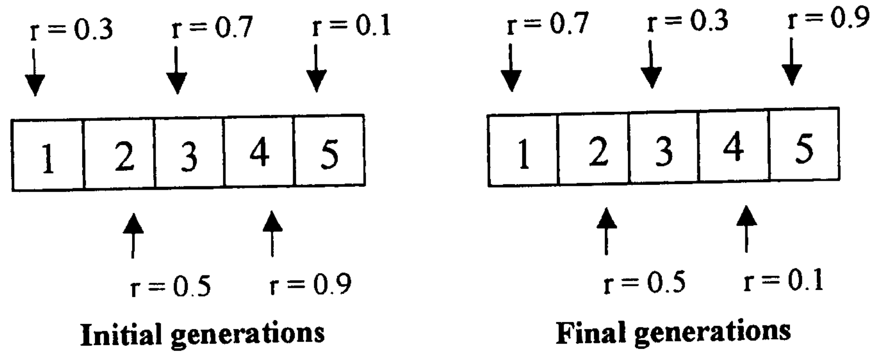


Figure 5.9: Assignation of Values of r Depending on the Gene Position

Combination Scheme

This technique is just a combination of the two previous methods: minimisation technique with $k = 0.1$ and $r = 0.05$ and non-uniform mutation. This has been done to assess the benefits of the non-uniform mutation operator in a scheme that has a very high effective mutation rate in the final generations.

Exponential Mutation

This scheme combines non-uniform mutation with an exponential probability of mutation.

It has been argued the increasing relevance of crossover in the initial generations and of mutation in the final ones. Using an exponential probability of mutation as shown in Figure 5.10 will allow testing that assertion.

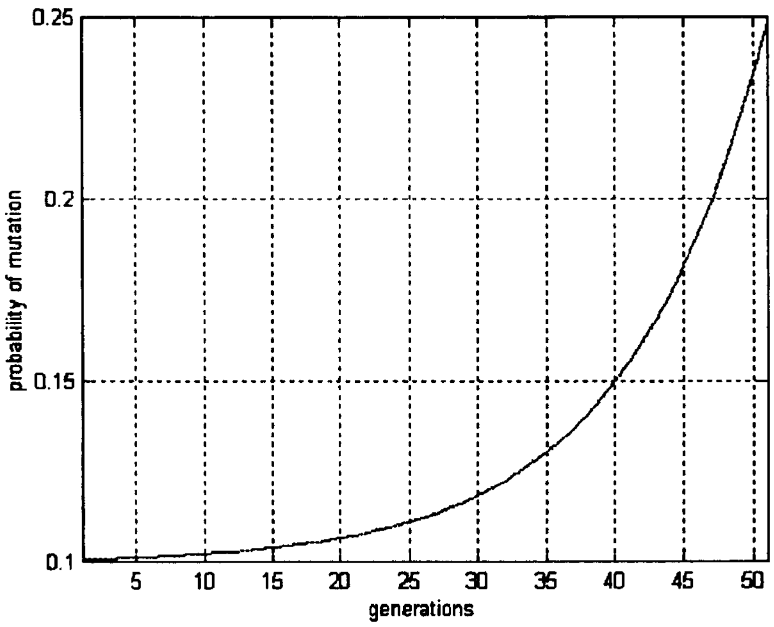


Figure 5.10: Probability of Mutation along the Generations

The reason why $p_m = 0.25$ has been chosen as the maximum value for p_m is that in the results obtained with the minimization techniques it can be observed that there is a degradation of the GA performance once the population size is smaller than 30 individuals, with high peaks produce by mutation (see Figure B.1 in Appendix B). The effective probability of mutation for that population size is $p_m = 0.2667$. Thus p_m has been kept smaller than that value.

Exponential Crossover and Mutation

This scheme combines non-uniform mutation and exponential probability of mutation with exponential probability of crossover.

Although in the comparison study the best results are obtained with high probability of crossover, the performance of the GA did not seem to be so affected by the crossover probability as it was by the mutation rate or selection scheme. The possibility of crossover playing a role speeding the convergence more than improving the optimisation results was pointed out. If so, the relevance of the crossover operator would be in the initial generations.

An exponentially decreasing probability of crossover as shown in Figure 5.11 has been used in this scheme.

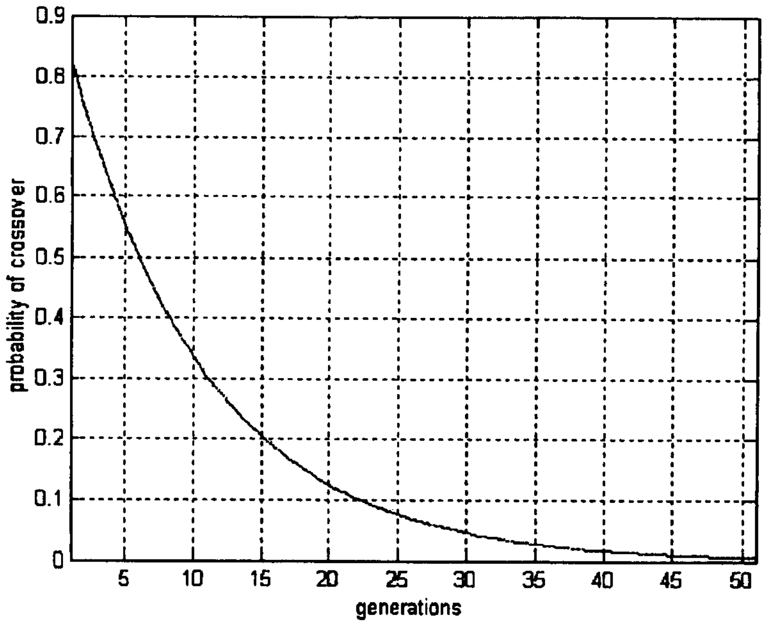


Figure 5.11: Probability of Crossover along the Generations

One Single Point Crossover

Keeping the characteristics of the previous scheme (i.e. non-uniform mutation and exponential mutation and crossover rates), the crossover operator is changed so that instead of selecting 2 points of crossover a single point of crossover is chosen.

Second Comparison Study Results

The results obtained applying these variations to the 4 optimisation problems used for the previous study are presented in this section. The results obtained with tournament selection, $p_c = 0.8$ and $p_m = 0.1$ are used as a benchmark for comparison purposes.

Criterion 1: Best Cost Analysis

In Figure 5.12 the best results (averaged over 6 runs) obtained for each GA scheme are shown. H_∞ is represented by a dash-dot line, Pole Placement is represented by a dashed line, PID is represented by a solid line and, finally, Sliding Mode plus PI is represented by a dotted line. See Table B.6 in Appendix B for numerical values.

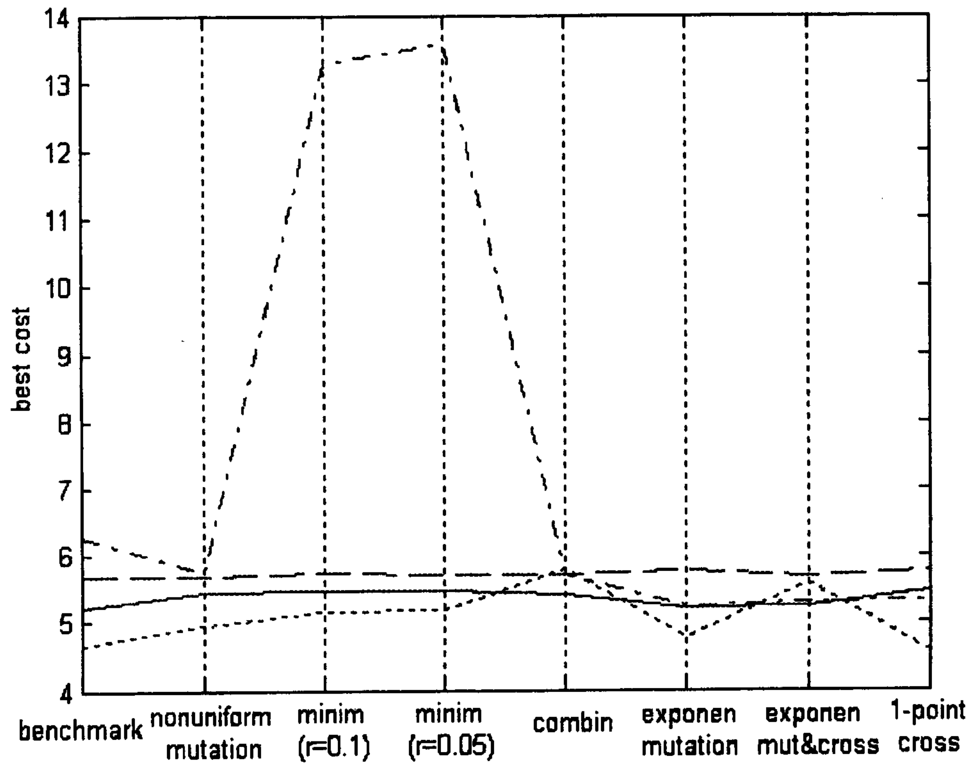


Figure 5.12: Best Costs Results for all GA Schemes for each Controller (dash-dot line: H_∞, dashed line: Pole Placement, solid line: PID, dotted line: Sliding Mode+PI)

It can be seen from Figure 5.12 that varying the GA scheme used in the optimisation of the controllers' parameters does not affect the results much, except for H_∞.

The H_∞ controller results are considerably influenced. All the GA schemes that include non-uniform mutation (i.e. non-uniform mutation, combination scheme, exponential mutation, exponential mutation and crossover, and 1-point crossover) outperform the benchmark result, while those schemes using exclusively minimisation techniques give worse results. This is to be expected since minimisation is a terminating scheme and thus reduces saturation.

In the PID and Pole Placement cases the results obtained with all the GAs are very similar. There are not significant differences with the benchmark.

The best cost values obtained for the SM+PI problem belong to a wider range (from 4.6 to 5.8) than those of PID or Pole Placement. The GAs with exponential mutation and 1-point crossover obtain the best results, although only this latter outperformed the benchmark. This time exponential mutation and crossover performed quite poorly.

Criterion 2: Generation of Convergence Analysis

In Figure 5.13 the results of generation of convergence (averaged over 6 runs) for each GA scheme are shown. Again, H_∞ is represented by a dash-dot line, Pole Placement is represented by a dashed line, PID is represented by a solid line and, finally, Sliding Mode plus PI is represented by a dotted line.

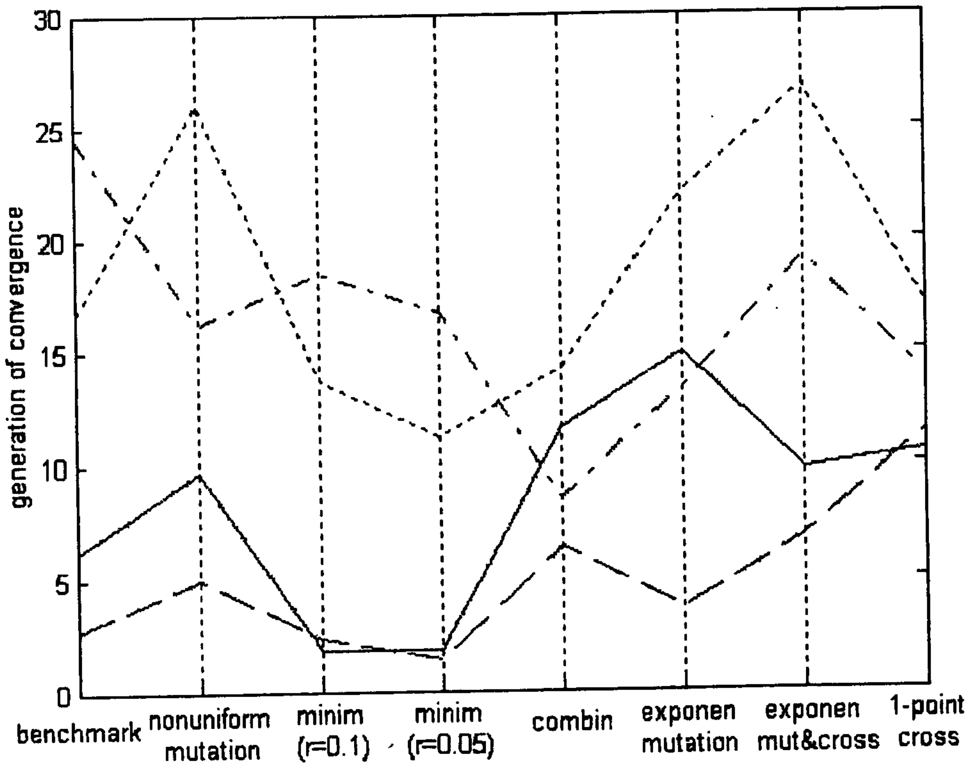


Figure 5.13: Generation of Convergence Results for all GA Schemes (dash-dot line: H_∞, dashed line: Pole Placement, solid line: PID, dotted line: Sliding Mode+PI)

It can be observed that the earliest generations of convergence are produce by the GA schemes with minimization (i.e. minimization with r=0.1, minimization with r=0.05 and combination scheme). However, this algorithm produces worse cost results so represents a case of premature convergence.

It is difficult to draw conclusions from the results obtained with the other GA schemes since they are not very consistent and the variations are not very significant.

<u>Criteria 3 and 4: Amount of Convergence in the Final Population and AMF Analysis</u>
In Figure 5.14 and Figure 5.15, the results of the amount of convergence and AMF (averaged over 6 runs) in the final population obtained for each GA scheme are shown. Again, H_∞ is represented by a dash-dot line, Pole Placement is represented by a dashed line, PID is represented by a solid line and, finally, SM + PI is represented by a dotted line. See Table B.8 in Appendix B for numerical values.

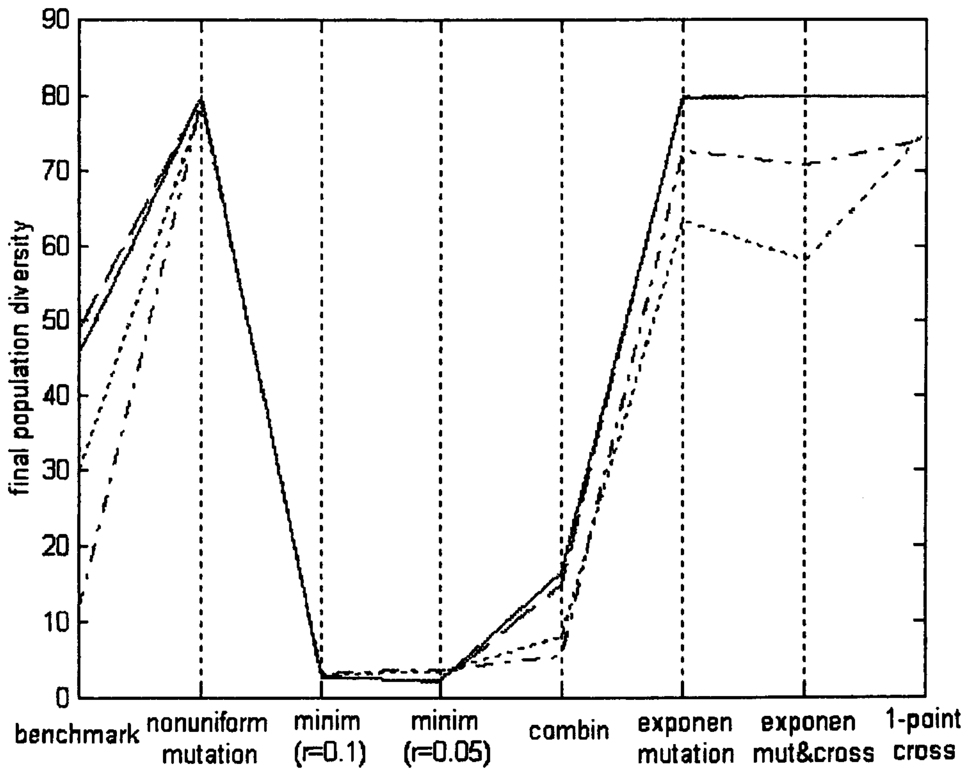


Figure 5.14: Amount of Convergence in the Final Population (dash-dot line: H_∞, dashed line: Pole Placement, solid line: PID, dotted line: Sliding Mode+PI)

The schemes that get smaller convergences are those that utilise the minimisation technique. This is to be expected since the technique eliminates similar individuals to increase diversity.

The inclusion of non-uniform mutation leads to higher amounts of convergence in the final population. This can be assessed comparing the results of non-uniform mutation, exponential mutation, exponential mutation and crossover and 1-point crossover with the benchmark results. This is to be expected since non-uniform mutation favours smaller mutation jumps in the final generations.

The results of AMF in the final population shown in Figure 5.15 corroborate previous conclusions.

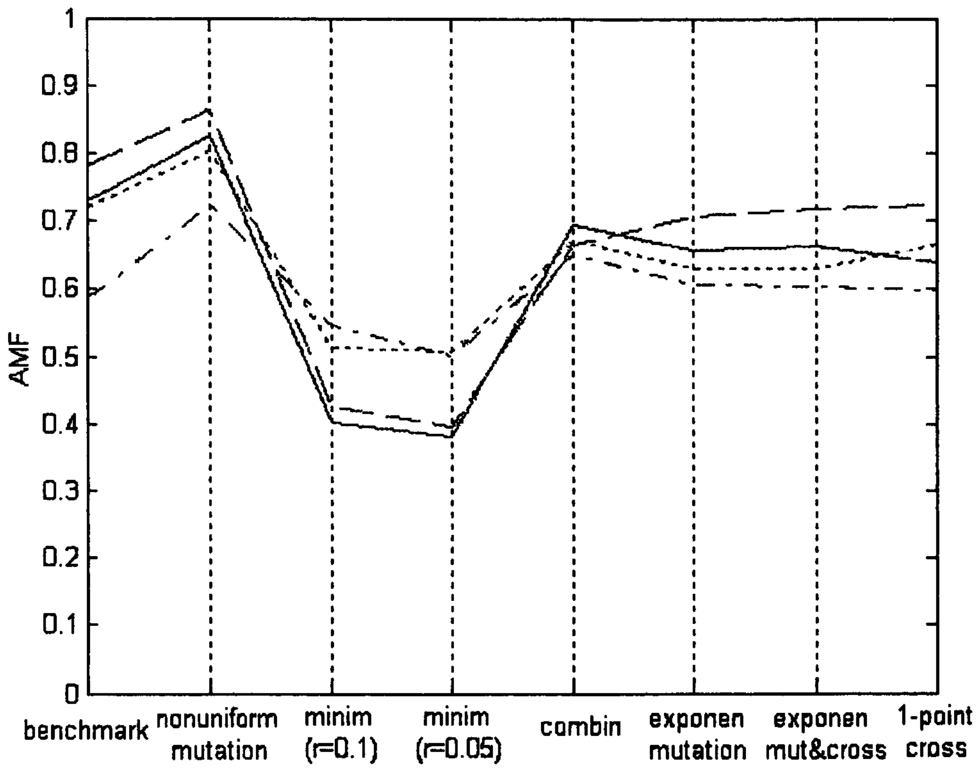


Figure 5.15: AMF in the Final Population (dash-dot line: H_∞, dashed line: Pole Placement, solid line: PID, dotted line: Sliding Mode+PI)

Summary of Results from Second Study

Minimisation techniques have proved not to be very efficient with this kind of optimisation problem. They have the worse average results in nearly every controller. In combination with non-uniform mutation they work better, but still not very well. On the other hand they evaluate fewer solutions, which reduces greatly the optimisation times.

Inclusion of non-uniform mutation improves the results obtained for H_∞ remarkably. However, the results obtained for the other controllers with just non-uniform mutation are worse than those used as a benchmark.

Once an exponential probability of mutation is combined with the non-uniform mutation operator the results improve. This scheme provides the best results for the H_∞ optimisation and for the rest of the problems the results are comparable with the benchmark (a bit worse for Pole Placement).

Modifications in the crossover operator such as the inclusion of exponential crossover probability and 1-point crossover do not improve the general performance of the GA.

This non-uniform-exponential mutation scheme would be used from now onwards as the standard GA. The reasons for choosing this particular scheme are the good results provided consistently throughout the four optimisation problems and the particularly good result for H_∞, the most difficult optimisation problem of them all. Also, the convergence rate data show good final convergence and fast speed of convergence to the near-optimal solution.

5.6. SUMMARY

Since the performance of GAs is quite problem domain dependant, two comparison studies have been carried out to find the genetic model better suited to problems of controller parameter optimisation.

From the results presented we can conclude that in this study tournament selection performs better than ranking or elitism. Also, high rates of mutation and crossover improve the results.

Regarding the new techniques introduced in the second comparison study, the only controller that clearly benefited from them is H_∞. The performance of the optimised solutions for the other controllers is not very much affected by variations in the genetic operators. However, the inclusion of non-uniform mutation together with an exponential mutation probability meant an overall improvement, especially for H_∞.

Thus, from the remainder of this study, unless otherwise stated, the GA operators used are tournament selection, 2-point crossover with a probability of 0.8 and non-uniform mutation (Section 5.5.2) with an exponential probability (Figure 5.10).

The chapter has also introduced the sGA technique as a half way technique between GAs and GP. The sGA includes 2 types of genes in the chromosome structure: the coefficient genes normally used in GAs plus some control genes. The control genes define a hierarchic structure well suited not only for parametric optimisation but also for structural optimisation. In the next chapter the results of the application of SGA for the optimisation of the weighting function in the H_∞ control problem are presented.

CHAPTER 6

GENETIC ALGORITHMS: A CONTROL APPLICATION FOR A MARINE VESSEL

6.1. INTRODUCTION

The GA scheme found to be best in Chapter 5 has been used to optimise the parameters that define the structure of the controllers presented in Chapter 4: PID [Astrom and Hagglund (1995), Dutton *et al.* (1997)], Pole Placement [Andry *et al.* (1983), Kautsky *et al.* (1985)], Sliding Mode [Slotine and Li (1991), Utkin (1972)] and H_∞ [Glover and Doyle (1988), Zhou *et al.* (1996)].

These control methodologies are used to provide the structure for propulsion controllers (for governing surge velocity) and navigation controllers (for governing heading) for CyberShip II. The goal of this study is to obtain controller solutions that satisfactorily perform these duties while keeping actuator usage to a minimum. The GA solves this minimisation problem by evolving controller parameter solutions that satisfy these objectives.

Also, GA optimisations in the presence of simulated environmental disturbances (wind-generated waves) [Fossen (1994)] are carried out. The objective is to see if by including a realistic noisy environment in the simulation the GA can improve the robustness of the controller.

Subsequently, all the optimised controllers have been implemented in the real plant and tested in the water basin of the Marine Cybernetics Lab (MCLab) of the Norwegian University of Science and Technology (NTNU) in Trondheim. Similar trials manoeuvres to those in the simulations were used in that part of the study.

The results obtained from this study illustrate the benefits of using GAs to optimise propulsion and navigation controllers for surface ships.

In addition, the H_∞ controller has been optimised using not only a GA but also the sGA genetic model [Dasgupta and McGregor (1993a)] presented in Section 5.3. The fact that the weighting functions to tune in the H_∞ problem are not only parameters but transfer functions (i.e. structures) make this optimisation problem very suitable for comparing the performance of GA versus sGA.

Chapter 6 is structured in the following way: Section 6.2 describes the method used in the study and the testing conditions in the real plant. Section 6.3 presents the simulated and real results obtained using PID control. Section 6.4 presents the simulated and real results obtained using Pole Placement methodologies. Section 6.5 and 6.6 present the results obtained using Sliding Mode and H_∞ respectively. Finally, in Section 6.7 conclusions are drawn.

6.2. DESCRIPTION OF THE METHOD

Genetic Algorithms have been used to optimise the parameters of the controller structures presented in Chapter 4. The plant to control is the heading and propulsion dynamics of CS2.

The GA model used has been the one found to be best suited to controller parameter optimisation in Section 5.5. The characteristics of the model are: tournament selection (with tournament size equal to 8); exponential and non-uniform mutation (as defined in Section 5.5) and double point crossover with a probability of 0.8. The population size was 80 and the number of generations 50. Every GA has been run 15 times.

The inclusion of waves in the simulations creates a more realistic environment and allows the potential of the GA to produce more robust controllers to be analysed. Therefore, every controller structure has been optimised in two different ways: with and without the inclusion of waves in the simulation.

The model used for the addition of simulated waves has been presented in Section 3.4. The simulated waves had a significant height of 3 meters (0.0429 m after scaling), which corresponds to a sea state code of 5 (rough sea). Waves of this height happen the North Atlantic with a probability of 15.44% [Fossen (1994)]. The initial angle of encounter between the waves and the vessel has been chosen to be 135° for this study.

The evaluation of the individuals of the population is done through the simulation of a simultaneous double step manoeuvre while varying the surge speed (see Figure 3.9). After the simulation, the performance of the controller is assessed using the cost function of Equation (5.1).

In addition, H_∞ is also optimised using the sGA model presented in Section 5.3. The genetic operators used are the same as for the GA optimisation (i.e. tournament selection, two-point crossover and non-uniform mutation), but four control genes are added to the chromosome representation for structural optimisation purposes. They define the structure of the weighting functions (i.e. a gain, a first order transfer function plus gain, a first order transfer function plus gain and one zero or a second order transfer function plus gain and one zero).

Finally all the optimised results were tested in the water basin of the MCLab in NTNU. The path tracking manoeuvre used for the trials is plotted in Figure 3.10. The testing in the real plant was also performed while generating waves as described in Section 3.5.

The structure of the next sections is as follows: the results of the optimisation of each controller are divided into two categories, those obtained without including waves in the optimisation and those obtained including waves in the optimisation. Then, for each category the best result and the averaged results obtained in the various GA runs are discussed. Finally, graphs showing the best controller responses simulated and real performance are included.

6.3. PID

6.3.1. OPTIMISATION PROCEDURE

The first optimisation task is that of optimising two decoupled PID controllers, one for heading control and the second for propulsion control.

The mission of the GA algorithm is to optimise the PID gains summarized in Table 6.1 according to the cost function from (5.1). All of them are real, positive values.

Table 6.1: Parameters to Optimise for PID Control

Propulsion	K_{Pp}	K_{Ip}	K_{Dp}
Heading	K_{Ph}	K_{Ih}	K_{Dh}

6.3.2. OPTIMISATION WITHOUT WAVES: SIMULATED AND REAL RESULTS

Once the GA optimisation of the gains presented in Table 6.1 was run 15 times, the best result obtained converged to the PID gains shown in Table 6.2 in 7 generations (see Figure C.1 from Appendix C) and provided a cost value of 1.2 according to the cost function from Equation (5.1). This is the resulting controller used in the real trials.

Table 6.2: Best PID Results - Optimisation Without Waves

	K_P	K_I	K_D
Propulsion	508.8	0.0579	9
Heading	16.8	9.09	9.4

If averaging the gains results obtained in each of the 15 runs, the average and standard deviation results are as shown in Table 6.3:

Table 6.3: Average and Standard Deviation PID Results
Optimisation Without Waves

	K_P		K_I		K_D	
	Avg	StDev	Avg	StDev	Avg	StDev
Propulsion	484.2	48.7483	3.2768	5.7124	9.1715	0.4429
Heading	17.0371	5.8375	7.5789	3.8062	9.628	0.3546

The standard deviation values show a consistent convergence to gain values of the same order of magnitude. The convergence of the GA to solutions that included derivative terms with a value of around 9 was especially regular (standard deviations smaller than 0.5). A large standard deviation in the gain values obtained in the GA runs would indicate that variations in those parameters do not degrade the cost value associated with that solution significantly.

The next Figure 6.1 plots the simulated performance of the controller in Table 6.2 when tracking a simultaneous double step manoeuvre (reference responses by dashed line).

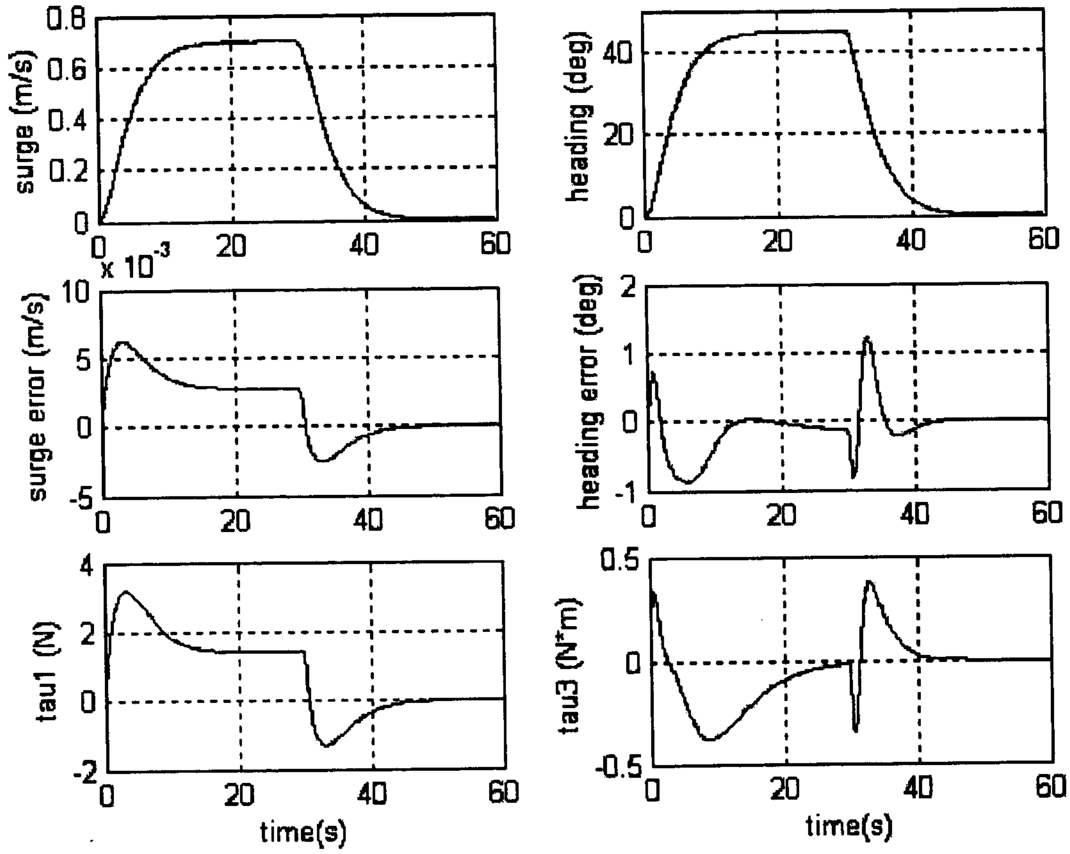


Figure 6.1: Simulated Results of the PID Controller Optimised Without Waves

As can be seen in Figure 6.1, the tracking of the desired responses is very good both in propulsion and heading (the error signals not exceeding 7 mm/s and 1.5°respectively) and the actuators usage is free from rippling. There is a slightly steady-state error in the surge response (around 3mm/s) due to the small integral term in the PID controller for propulsion. The heading control shows two spikes that correspond with the beginning of the turn. They are due to the high gain of the system that reacts very quickly to the change and then it needs to overcompensate.

The following Figures 6.2 and 6.3 show the results obtained when the optimised PID controller was implemented in the real plant. Figure 6.2 shows the manoeuvring performance in calm waters. Again, the reference signals are marked by the dashed line.

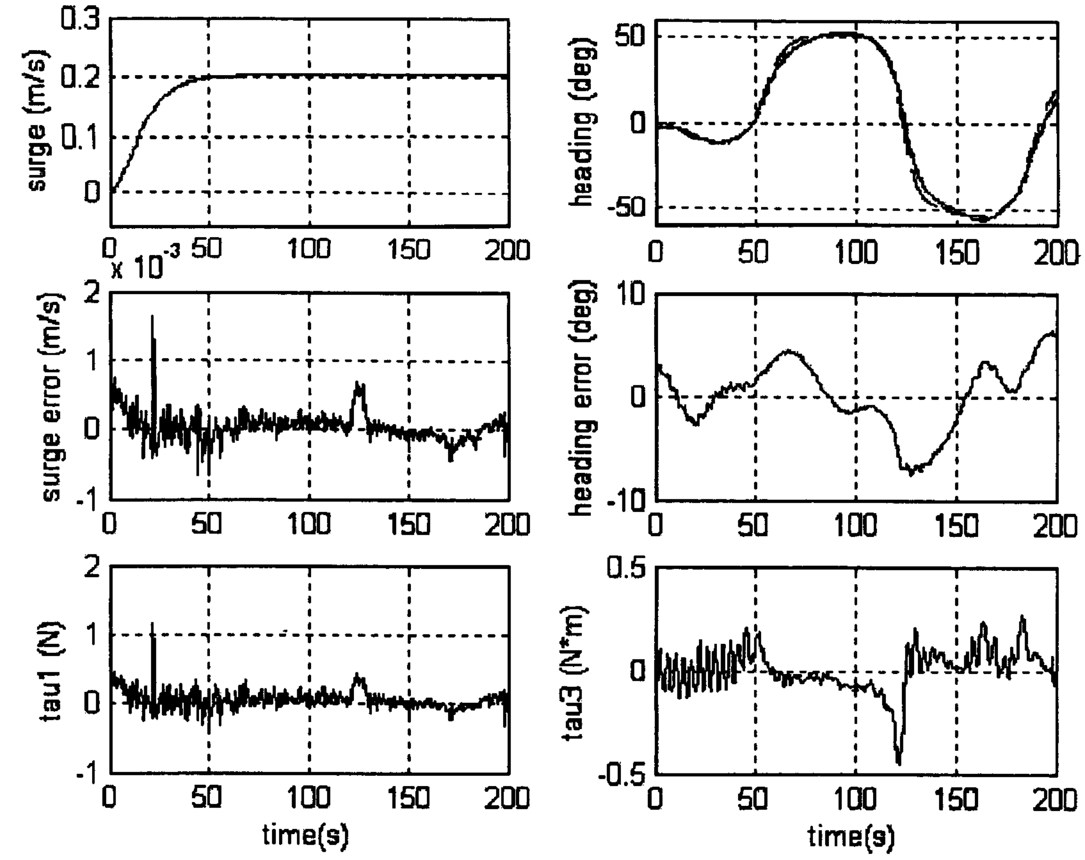


Figure 6.2: Real Results of the PID Controller Optimised Without Waves When Manoeuvring in Calm Waters

In Figure 6.2 it can be seen that the tracking of the desired responses is fairly good, especially for the surge speed (error smaller than 2 mm/s). However, the actuator signals contain lots of high frequency components that can be observed as well in the error plots. This indicates high gain control where the controller tracks the noise in the signals. Also, the heading tracking deteriorates in the second half of the zig-zag. This is probably due to the increased difficulty of a second turning just when the boat was recovering from the first. When comparing this figure with that of the manually tuned PID (see Figure 4.7) it can be observed that the signals have more high frequency components. This is due to the fact that the results shown in this chapter were obtained in a first visit to the facilities while the manually tuned results belong to a second sets of results obtained with an improved system. Even though the responses are noisier, the GA-optimised PID controller shows a far better performance, with a more accurate tracking and no signs of instability.

The following Figure 6.3 shows the manoeuvring performance when waves are generated in the water basin. The desired responses for heading and propulsion are represented in dashed lines.

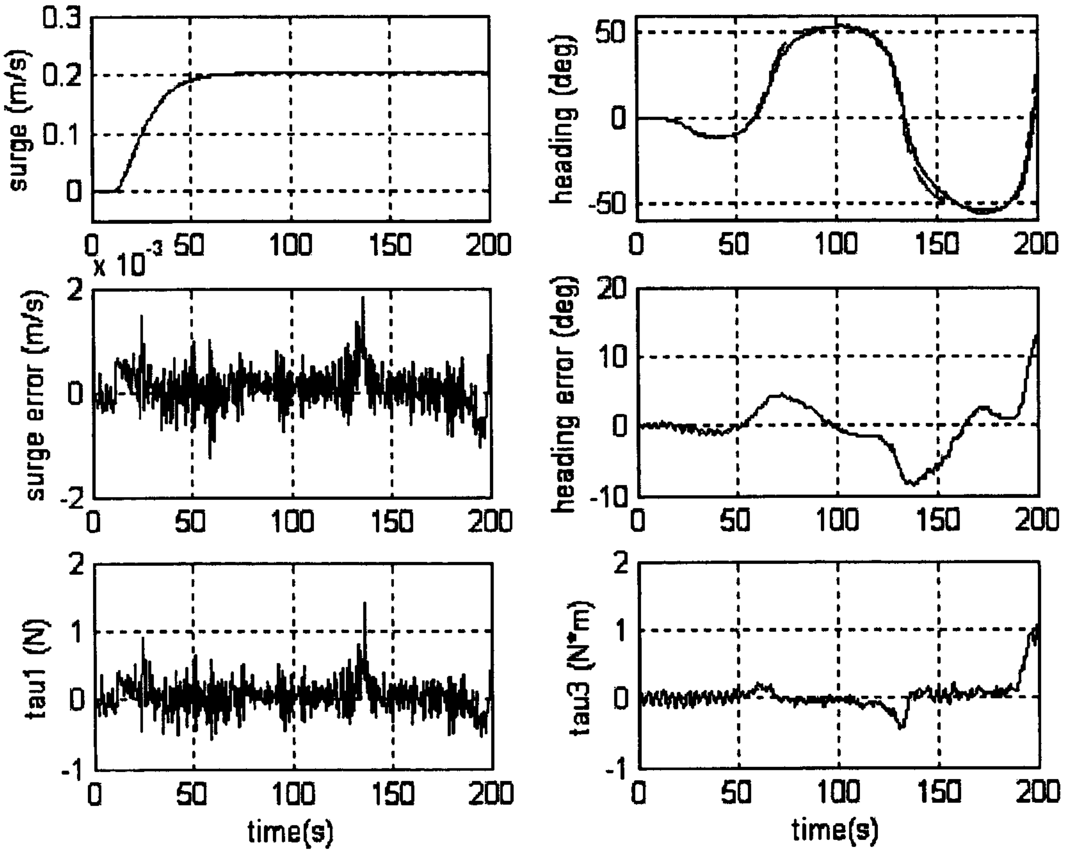


Figure 6.3: Real Results of the PID Controller Optimised Without Waves When Manoeuvring in the Presence of Waves

As can be seen, the inclusion of waves during the execution of the manoeuvre does not degrade the tracking performance much but induces more noise in the signals. The surge error is still smaller than 2 mm/s and the heading error is kept under 10° until the final stage of the manoeuvre. The higher peaks in the heading error and τ_3 that can be observed at the very end of the manoeuvre are due to the vessel moving out of the area covered by the positioning system and the consequent loss of control.

6.3.3. OPTIMISATION WITH WAVES: SIMULATED AND REAL RESULTS

When the GA optimisation was run including waves in the evaluation manoeuvre, the optimisation converged in 12 generations (as shown in Figure C.2 from Appendix C) and the best result of all provided a cost value of 4.3 with the PID gains shown in Table 6.4, which have been used afterwards for the real implementation in CS2.

Table 6.4: Best PID Results – Optimisation With Waves

	K_P	K_I	K_D
Propulsion	6.829	42.73	0.00399
Heading	21.45	0.7996	3.571

If averaging the gains results obtained in each run, the average values and standard deviation are:

Table 6.5: Average and Standard Deviation PID Results
Optimisation With Waves

	K_P		K_I		K_D	
	Avg	StDev	Avg	StDev	Avg	StDev
Propulsion	9.4775	3.3563	27.6211	17.2137	0.0046	0.0043
Heading	16.0988	4.5726	1.2689	0.8157	3.7489	0.3898

These PID gains are fundamentally different from those showed in Tables 6.2 and 6.3, especially those of the propulsion PID. In the heading PID controller the integral and derivative terms are slightly reduced, while the proportional term remains similar. On the other hand, the proportional and derivative terms in the propulsion control are significantly reduced (2 and 3 orders of magnitude respectively) and the integral term is very much increased.

The reduction of the derivative term in both PIDs is understandable since the derivative term will amplify any noise in the signals and we have included noise in the simulations. Moreover, the reduction of the derivative term in the propulsion controller has conditioned the gain reduction in the proportional term. A high gain will lead to instability. Oscillations can be compensated using a derivative term, but in this case the noisy environment imposes a small derivative term and therefore a smaller gain. In turn, the integral term has been increased to reduce the error, given the need of a small proportional gain.

Figure 6.4 plots the simulated controllers' performance when tracking a simultaneous double step manoeuvre (reference in dashed line).

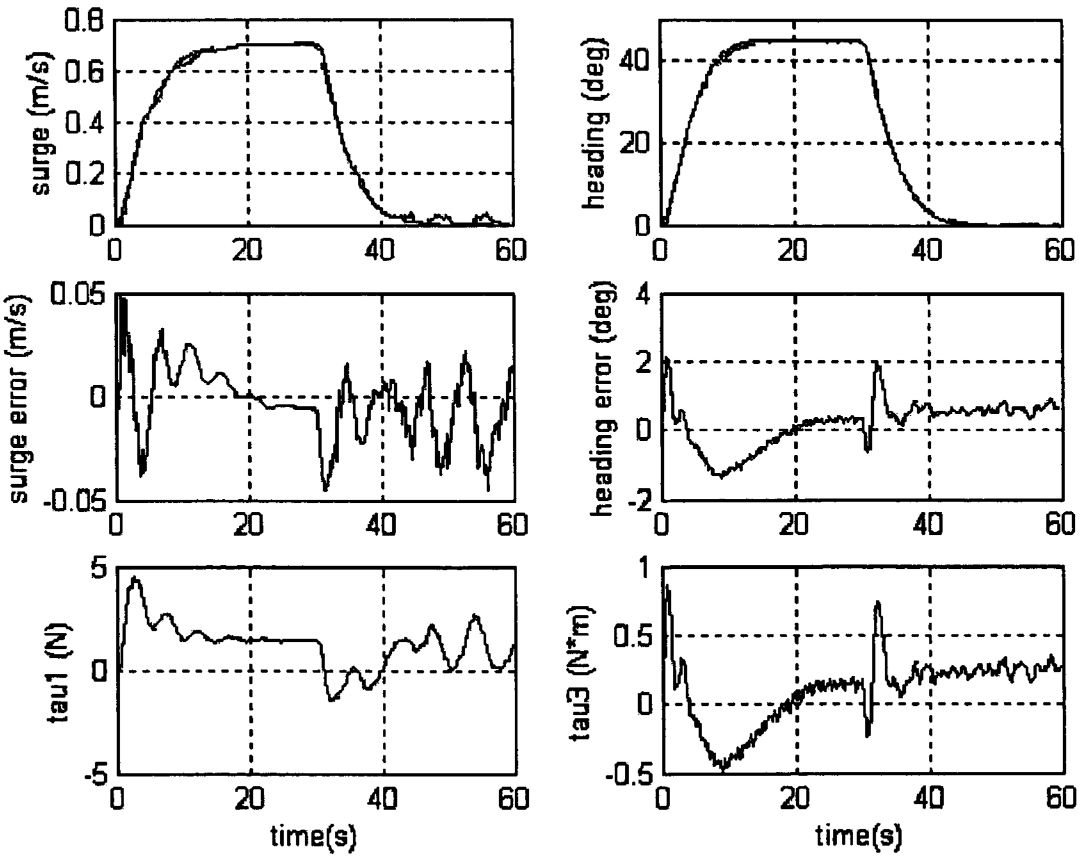


Figure 6.4: Simulated Results of the PID Controller Optimised With Waves

This shows that the PID controllers optimised with waves deal quite well with the high frequency chattering. Tracking performance is slightly degraded but still reasonable (surge and heading errors smaller than 0.05 m/s and 2°, respectively). The tracking degradation at the end of the surge signal is due to the slowing down of the speed that reduces the capability of the vessel in compensating for the effect of the waves.

When implementing the PID controllers with the parameters shown in Table 6.4 in the real plant there is a significant stability problem caused by the large integral gain and small derivative gain in the propulsion PID. Figure F.5 in Appendix F shows that the simulated response of the PID controllers from Table 6.4 when tracking the zig-zag manoeuvre is good. Therefore the model is not representing accurately the performance of the controllers.

Since it is not possible to stabilise the plant with the values obtained by the GA the propulsion controller needed to be retuned. The new PID controllers only differ from those of Table 6.4 in the integral and derivative terms of the propulsion PID. They are presented in Table 6.6 and they are used in the real testing performance shown in Figures 6.5 and 6.6.

Table 6.6: Retuned PID Gains

	K_P	K_I	K_D
Propulsion	6.829	0.01	0.01
Heading	21.45	0.7996	3.571

The results obtained in the MCLab while testing these controllers are shown in the next Figures. Figure 6.5 plots the results where waves have not been generated in the water tank.

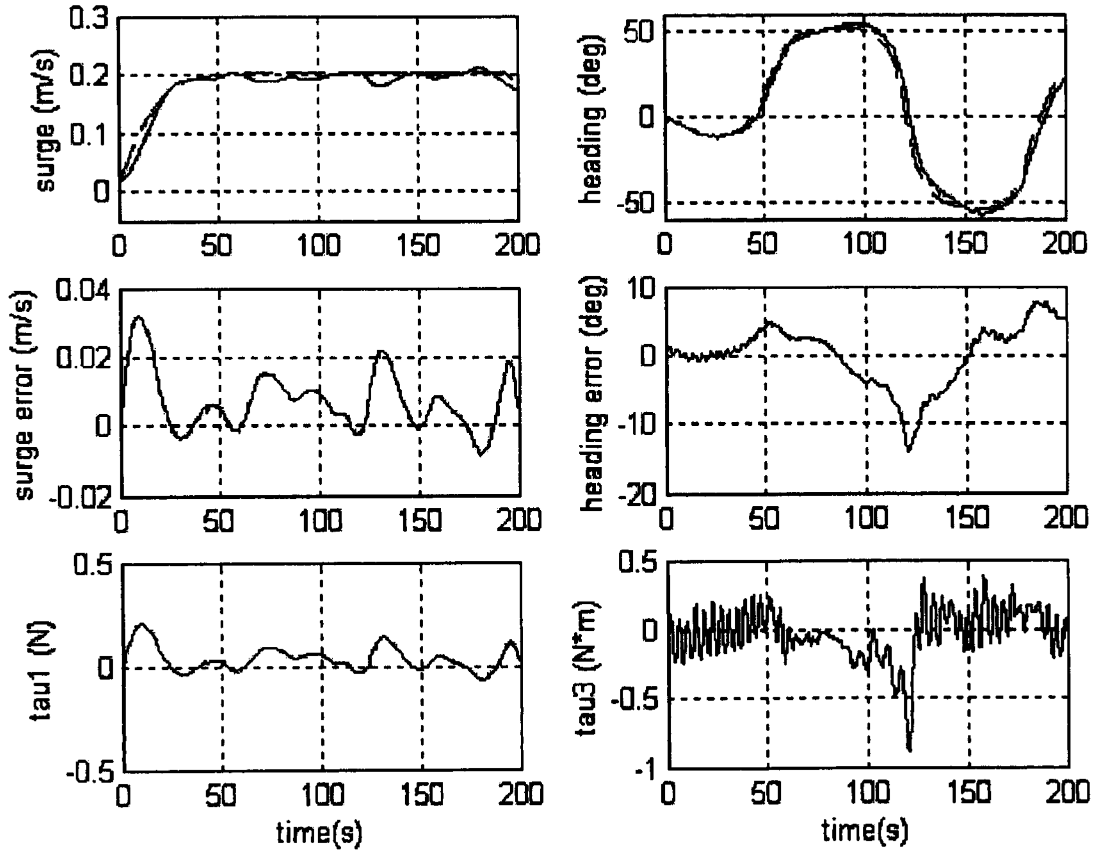


Figure 6.5: Real Results of the PID Controller Optimised With Waves When Manoeuvring in Calm Waters

The tracking performance of these controllers is quite degraded in comparison with previous PID tuning (Figure 6.2), especially in the surge speed signal, which is very oscillatory. However, these controller solutions reduce chattering in the actuators more than those controllers optimised without noise.

Figure 6.6 shows the effect of waves on the controllers' performance.

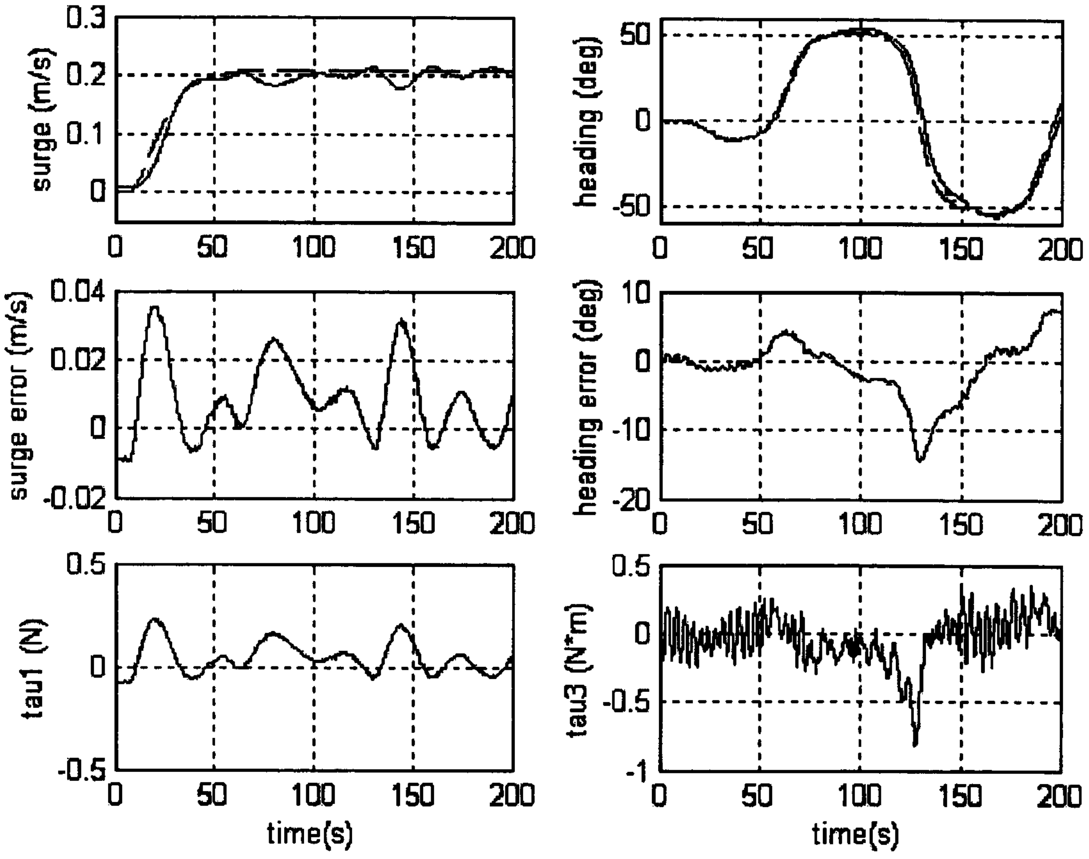


Figure 6.6: Real Results of the PID Controller Optimised With Waves When Manoeuvring in the Presence of Waves

The inclusion of waves degrades the tracking performance slightly. When comparing the performance in the real trials of the PID optimised without waves from Table 6.2 (shown in Figures 6.2 and 6.3) with the performance of the PID controller from Table 6.6, the conclusion that can be drawn is that the PID controller optimised with waves included in the simulation exhibits worse tracking performance, especially for propulsion, but less noisy signals. Thus, the inclusion of waves in the optimisation results in a reduction of the chattering in the actuators, not in a more robust performance against disturbances.

6.3.4. SUMMARY OF PID RESULTS

A decoupled PID controller has been optimised using GA in two different ways, i.e. with and without waves in the simulation. The GA converged to significantly different controller gains in each case. Both work well in simulation.

The controller optimised without waves did not need any retuning when implemented in the real plant and it demonstrated very good tracking characteristics, even in the presence of waves. However, the actuator signals were very noisy, especially τ_1.

The GA results when including waves in the evaluation process show evidence of instability so the integral and derivative terms of the propulsion PID needed to be retuned. This raises doubts about the accuracy of the mathematical model, since the simulated results of Figure 6.4 and F.5 were good and did not exhibit any sign of instability.

Once the retuned controller was implemented in the real plant, the tracking performance was poorer than the previous one but the chattering in the actuators signals was significantly reduced.

6.4. POLE PLACEMENT

6.4.1. OPTIMISATION PROCEDURE

In the Pole Placement (PP) optimisation problem the parameters to be optimised are the four poles of the MIMO system.

Table 6.7: Parameters to Optimise for PP Control

PP	*pole1*	*pole2*	*pole3*	*pole4*

The poles can be: 4 real poles or 2 real poles plus a complex conjugate pair. The real part of the poles of a system needs to be negative for the system to be stable. Therefore, only the magnitude of the poles has been encoded in the GA and the polarity has been added in the decoding process.

6.4.2. OPTIMISATION WITHOUT WAVES: SIMULATED AND REAL RESULTS

The GA converged in 4 generations (see Figure C.3 from Appendix C) to a cost value of 0.96, with the poles from Table 6.8.

Table 6.8: Best PP Results - Optimisation Without Waves

pole1	*pole2*	*pole3*	*pole4*
-15.09	-2.168	-0.9043+1.82j	-0.9043-1.82j

Averaging the results obtained through different runs is more complicated in the PP problem since some of the runs converged to real poles and others to 2 real poles plus a complex conjugate pair. Separating real and complex poles in two convergence subgroups the average and standard deviation values are as shown in Table 6.9:

Table 6.9: Average and Standard Deviation PP Results
Optimisation Without Waves

Real solutions (4 real poles)							
pole1		*pole2*		*pole3*		*pole4*	
Avg	StDev	Avg	StDev	Avg	StDev	Avg	StDev
-9.9985	0.0005	-4.67	0.33	-2.1895	0.2905	-1.2499	0.25

Complex solutions (2 real poles + 1 complex conjugate pair)							
real part		*imaginary part*		*pole3*		*pole4*	
Avg	StDev	Avg	StDev	Avg	StDev	Avg	StDev
-1.1216	0.283	1.4984	0.6063	-11.564	2.351	-2.99	2.063

From the 15 total runs, 2 converged to 4 real poles while 13 converged to a pair of complex conjugate poles plus 2 real poles. The complex conjugate pairs from Tables 6.8 and 6.9 are slower poles than the real ones (i.e. closer to the imaginary axis). Therefore they are dominating the system response. This means that the GA has favoured pole positions that provide responses of second order type. Since the desired response in the evaluation manoeuvre of the optimisation process is a second order response, this was to be expected. Even in the case of 4 real poles, the two slower ones are quite close, showing as well a tendency to 2nd order type of responses.

The next Figure 6.7 shows the simulated performance of the PP controller when tracking the simultaneous double step manoeuvre used in the optimisation runs (the desired response is represented in a dashed line).

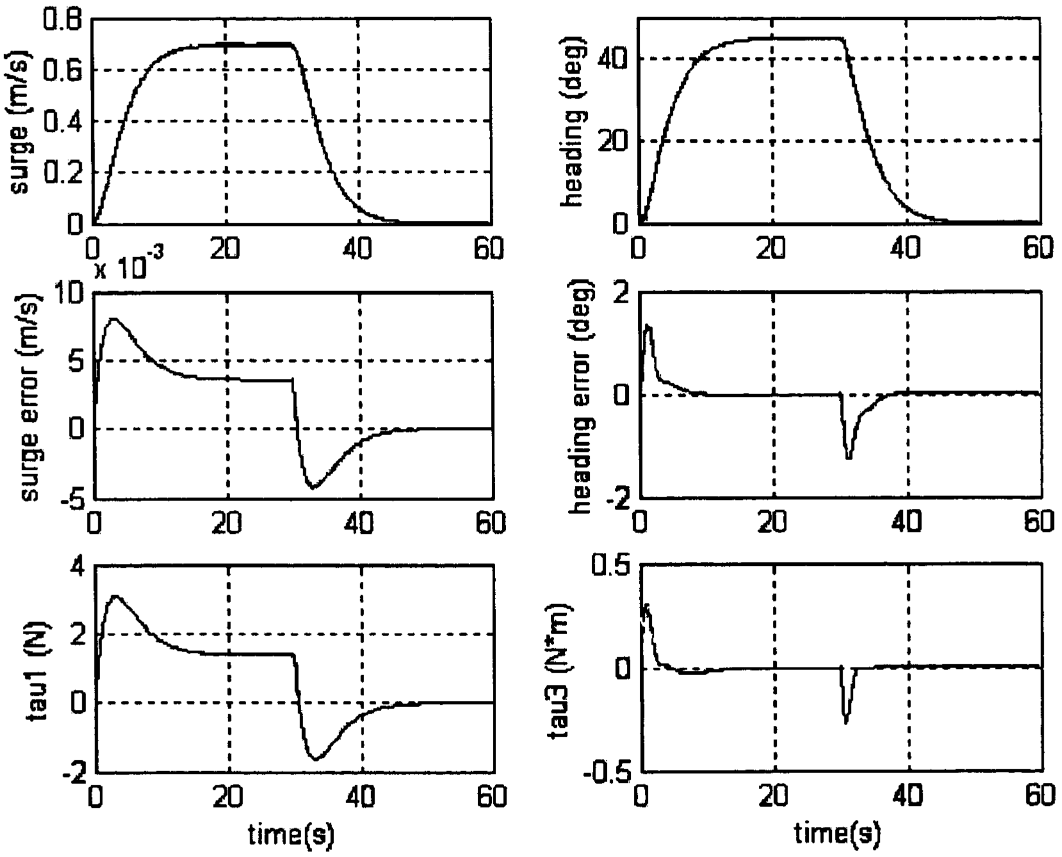


Figure 6.7: Simulated Results of the PP Controller Optimised Without Waves

The tracking of the desired response is once more very good, although the surge and heading errors are slightly worse than those obtained with the PID controller optimised without waves (see Figure 6.1). On the other hand the actuator usage is reduced with respect to PID, especially in τ_3. The reaction of the controller is significantly slower than in PID, which means that the gain is lower. The heading control signal does not show peaks as in PID control, but as a consequence the heading error is larger. It is the usual trade-off between tracking performance and control effort.

The following Figures 6.8 and 6.9 present the findings of the implementation of the PP controller in the real plant. Figure 6.8 shows the results obtained while tracking the desired propulsion and heading signals in calm waters.

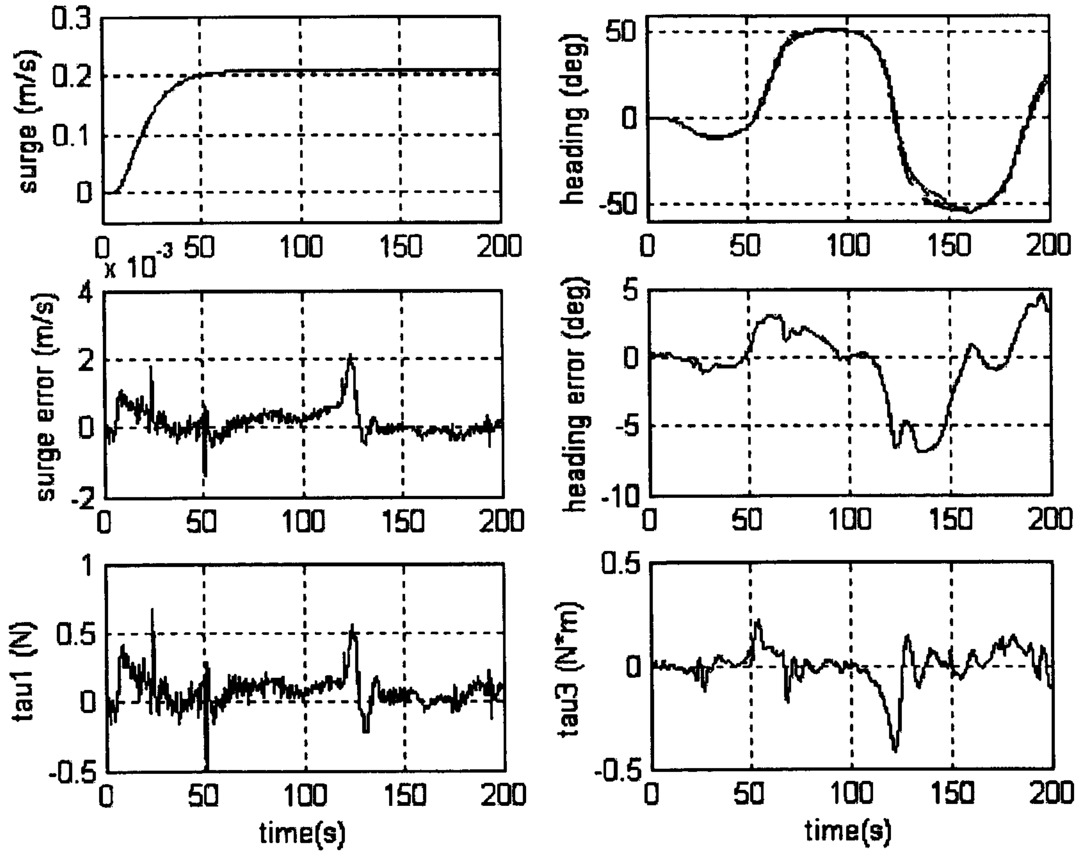


Figure 6.8: Real Results of the PP Controller Optimised Without Waves When Manoeuvring in Calm Waters

From the above graph it can be seen that the PP controller tracks both reference signals well. Yet again, the commanded surge force τ_1 is quite noisy. In this case the chattering in the heading control signal is better than for PID. The oscillations are due to the controller trying to compensate for the slower reaction to the noise due to the low gain of the controller. The controller produces too much momentum that results in oscillations. When comparing this figure when the manually tuned PP controller responses (see Figure 4.13) it can be observed that the tracking is more accurate and the actuator usage smaller despite of the manually tuned response having been obtained with an improved system.

Figure 6.9 shows the effect produced in the vessel dynamics by the generation of waves in the water tank while executing the zig-zag manoeuvre.

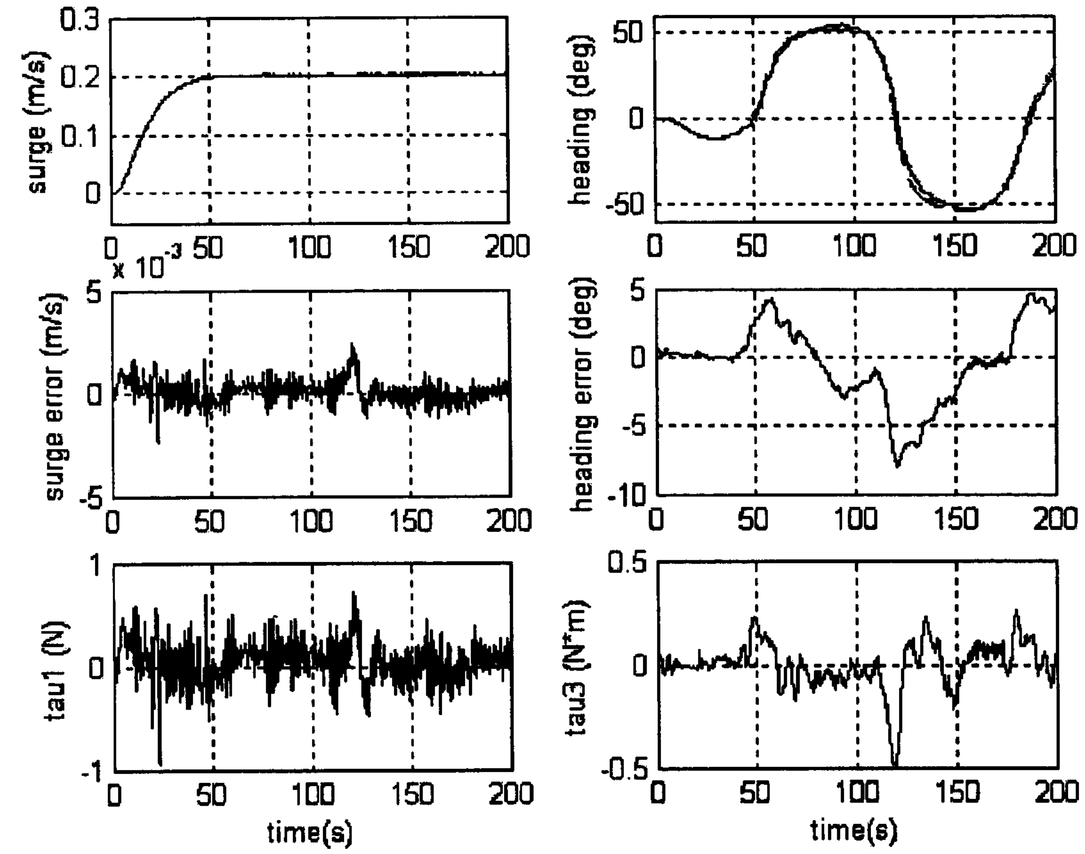


Figure 6.9: Real Results of the PP Controller Optimised Without Waves When Manoeuvring in the Presence of Waves

The inclusion of waves, as it can be observed from the figure, does not degrade tracking but induces high frequency components.

6.4.3. OPTIMISATION WITH WAVES: SIMULATED AND REAL RESULTS

The GA converges to a cost value of 10.6 in 16 generations (see Figure C.4 from Appendix C), with the following poles:

Table 6.10: Best PP Results – Optimisation With Waves

pole1	*pole2*	*pole3*	*pole4*
-6.224	-0.997	-0.451+2.308j	-0.451-2.308j

Again, averaging the results obtained through 15 runs is more complicated in the PP problem since some of the runs converged to real poles and others to complex ones.

Separating real and complex poles into two convergence subgroups provides the following average and standard deviation values:

Table 6.11: Average and Standard Deviation PP Results Optimisation With Waves

Real solutions (4 real poles)							
pole1		*pole2*		*pole3*		*pole4*	
Avg	StDev	Avg	StDev	Avg	StDev	Avg	StDev
-9.2805	0.249	-2.712	0.4764.	-1.1876	0.3097	-0.9805	0.0523

Complex solutions (2 real poles + 1 complex conjugate pair)							
real part		*imaginary part*		*pole3*		*pole4*	
Avg	StDev	Avg	StDev	Avg	StDev	Avg	StDev
-0.7542	0.378	1.963	0.4209	-6.7174	1.0718	-1.0403	0.0856

From the 15 total runs, 8 converged to 4 real poles while 7 converged to a complex pair conjugate plus 2 real poles. Thus, the inclusion of waves has increased the number of times the GA has converged to real poles. Two dominant real poles produce an increase damping in the response if compared with two dominant complex poles. This results in a smaller gain in the system. It can also be observed that the poles have been chosen closer to the imaginary axis than when the optimisation was run without waves. This means that the control effort will be reduced, since the fast poles require more control input than the slower ones. As in the PID case, the inclusion of waves in the optimisation has led to controller solutions that reduce the control effort in the actuators.

The next Figure 6.10 shows the simulated performance of the PP controller from Table 6.10 when tracking the simultaneous double step manoeuvre for propulsion and heading used in the GA optimisation process. The reference signals are represented in a dashed line.

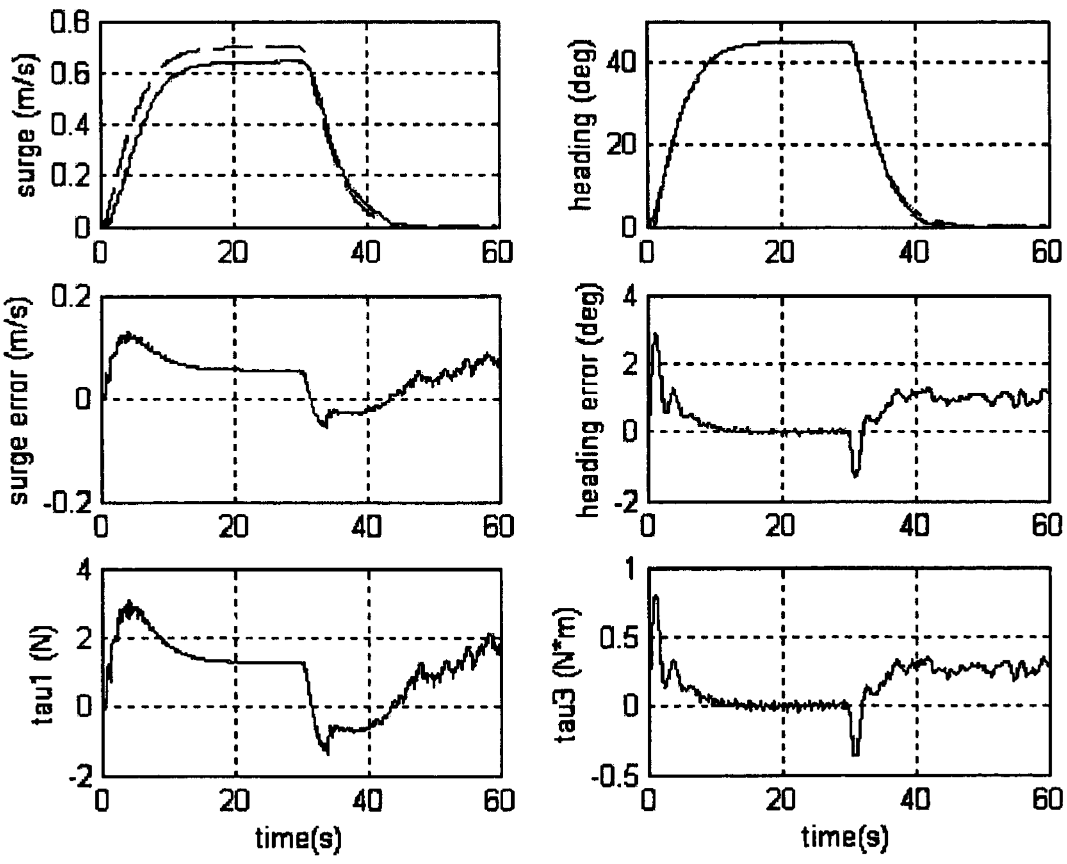


Figure 6.10: Simulated Results of the PP Controller Optimised With Waves

As for the PID the PP optimised with waves reduces control effort in detriment of surge speed tracking. In this case the propulsion tracking is really poor with a steady-state error of more than 0.05m/s.

The following Figures 6.11 and 6.12 will present the results obtained while implementing the optimised PP controller into the real plant. Figure 6.11 shows the vessel performing the zig-zag manoeuvre without waves being generated in the water tank. The reference signals are once more represented in a dashed line.

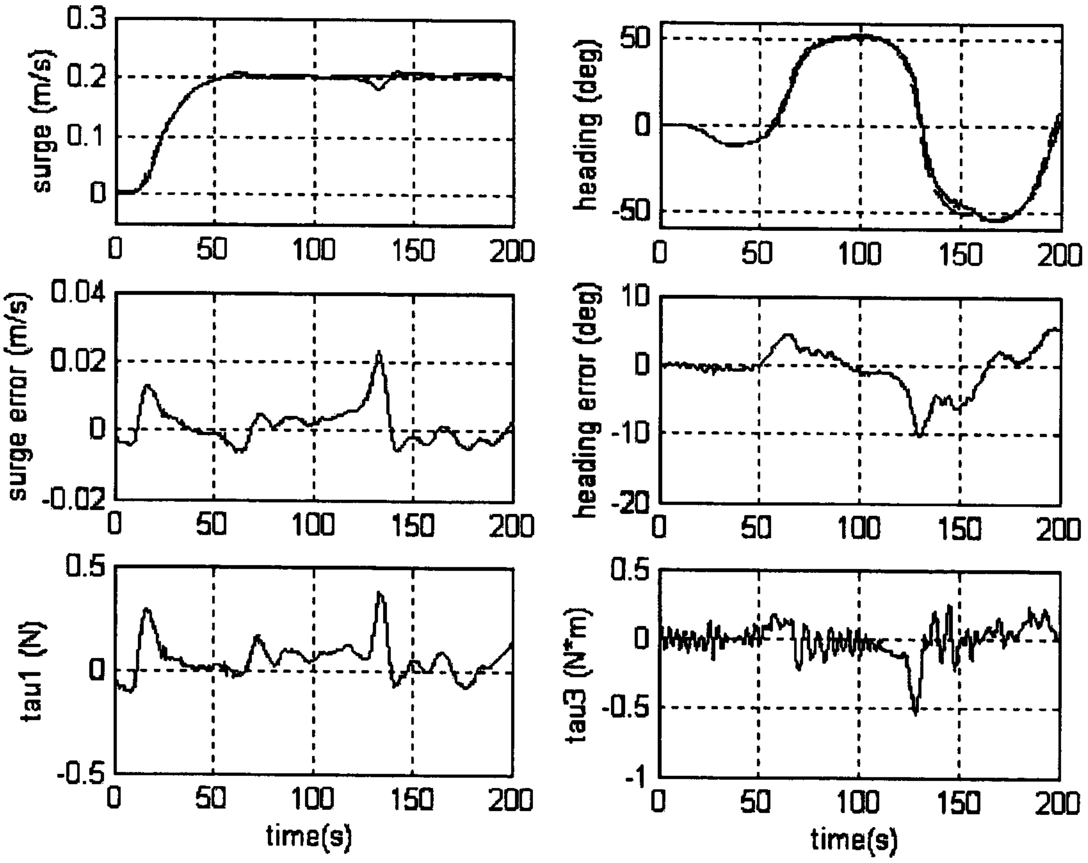


Figure 6.11: Real Results of the PP Controller Optimised With Waves When Manoeuvring in Calm Waters

The usual trade-off between fewer oscillations in τ_1 or better surge speed tracking is reflected again in Figure 6.11. As in previous cases the controller optimised with disturbances exhibits fewer oscillations, but surge and heading error are worse than in Figure 6.8 (PP optimised without waves manoeuvring in calm waters). However, given the bad propulsion tracking obtained in the optimisation studies (Figure 6.10), the real propulsion response is better than expected.

In Figure 6.12 the same manoeuvre is shown but this time it is performed by CS2 in the presence of waves in the water tank.

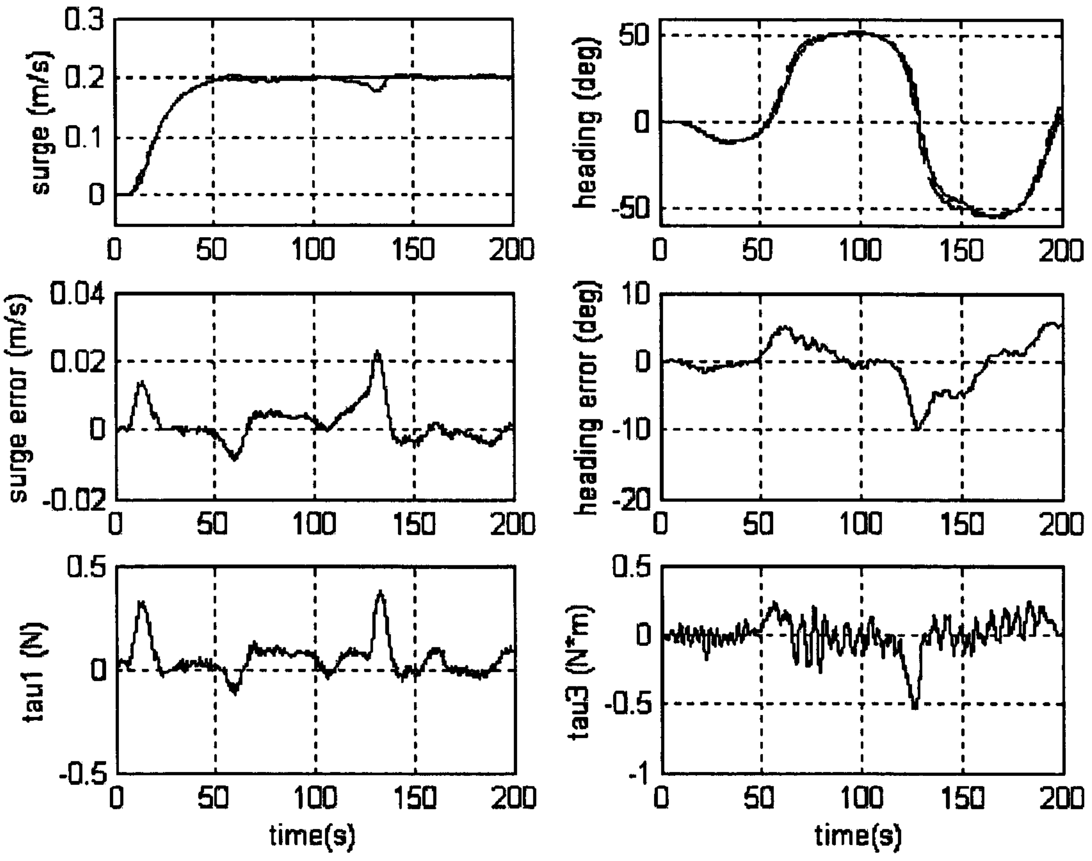


Figure 6.12: Real Results of the PP Controller Optimised With Waves When Manoeuvring in the Presence of Waves

Figure 6.12 shows how the inclusion of waves increases the chattering in the control signals but it does not degrade the tracking performance significantly with respect to the performance without waves. When comparing to the results obtained in the real testing with waves with the PP controller optimised without waves (Figure 6.9), the same conclusion can be drawn once more, the chattering in the actuator signals has been reduced at the expense of tracking performance.

6.4.4. SUMMARY OF POLE PLACEMENT RESULTS

PP controller results are consistent with the conclusions drawn from the PID controller results: the inclusion of waves in the GA evaluation leads to lower control gains while it degrades tracking. The results are comparable with those obtained with PID.

It is worth mentioning that both PP optimised controllers (with and without waves) were implemented in the real plant without any need for further tuning.

6.5. SLIDING MODE

6.5.1. OPTIMISATION PROCEDURE

Two different configurations based on the Sliding Mode controller presented in Section 4.5 have been optimised using GA: a Sliding Mode controller for heading and PI controller for propulsion (SM+PI) (Section 6.5.2) and two decoupled Sliding Mode (SM) controllers (Section 6.5.3).

The parameters to optimise for minimising the cost function from Equation (5.1) are two poles ($pole_{h1}$ and $pole_{h2}$), the switching gain (η_h) and the boundary layer thickness (Φ_h) for heading. For propulsion, either a proportional (K_{Pp}) and integral gain (K_{Ip}) for the SM+PI configuration or one pole ($pole_{p1}$), the switching gain (η_p) and the boundary layer thickness (Φ_p) for the SM configuration.

Table 6.12: Parameters to Optimise for Sliding Mode Control

SM+PI	$pole_{h1}$	$pole_{h2}$	η_h	Φ_h	K_{Pp}	K_{Ip}	
SM	$pole_{h1}$	$pole_{h2}$	η_h	Φ_h	$pole_{p1}$	η_p	Φ_p

The pair of poles of the equivalent term for the heading subsystem can be either real poles or a complex conjugate pair. As before, only the magnitude of the poles has been optimised, the sign was added in the decoding operator to ensure that they were in the left hand side of the s-plane. The switching gain has to be positive to ensure stability robustness [McGookin (1997)].

6.5.2. SLIDING MODE + PI CONTROLLERS

6.5.2.1. Optimisation Without Waves: Simulated and Real Results

The GA optimisation of the SM+PI parameters shown in Table 6.12 has converged in 4 generations (see Figure C.5 in Appendix C). The best result obtained provides a cost value of 4.5 with the following gains:

Table 6.13: Best SM+PI Results - Optimisation Without Waves

	K_p	K_{Ip}
Propulsion	354.4	23.9

	$pole_{h1}$	$pole_{h2}$	η	Φ
Heading	-1.293	-0.2594	0.9076	0.465

After averaging the parameters obtained through the 15 runs, the resulting average and standard deviation values are shown in Table 6.14. Since the switching gain (η) and boundary layer thickness (Φ) have converged to very different values in the separate runs, the ratio values (η / Φ) have also been included in Table 6.14.

Table 6.14: Average and Standard Deviation SM+PI Results Optimisation Without Waves

	K_{Pp}		K_{Ip}	
	Avg	StDev	Avg	StDev
Propulsion	237.0373	130.41	33.4018	35.293

	$pole_{h1}$		$pole_{h2}$	
	Avg	StDev	Avg	StDev
Heading	-1.034	0.5847	-0.2514	0.0884

	η		Φ		η / Φ	
	Avg	StDev	Avg	StDev	Avg	StDev
Heading	1360.5471	3363.0	675.3331	1675.3254	2.4679	1.4399

Although the poles converge to very similar values, that is not the case of the switching gain η and boundary layer width Φ. However, if instead of comparing the values of η and Φ, we take into account the ratio between them, there is a clear convergence. This can be due to the SM controller operating not in the switching range of the hyperbolic tangent function but in the boundary layer [McGookin (1997)]. If so, Equation (4.33) becomes:

$$\dot{\sigma}(\hat{\mathbf{x}}) \approx -\eta \cdot \frac{\sigma(\hat{\mathbf{x}})}{\Phi} \qquad (6.1)$$

and therefore, the ratio η/Φ drives the performance of the controller, not the individual values for η and Φ.

The PI propulsion control optimised gains are quite different from those of the PID decoupled controller. When comparing Tables 6.3 and 6.14 it can be observed that the average proportional gain for SM+PI is smaller (though of the same order of magnitude and with a large standard deviation) and the average integral gain is significantly bigger. The lack of derivative term restricts the size of the proportional term since it cannot be used to compensate for the oscillations induced by a large proportional gain. The GA has probably tried to compensate for the damping produced by a smaller proportional gain by increasing the integral term and, consequently, improving the steady-state error.

Next Figure 6.13 shows the simulated performance of the decoupled SM+PI controllers when tracking a simultaneous double step manoeuvre.

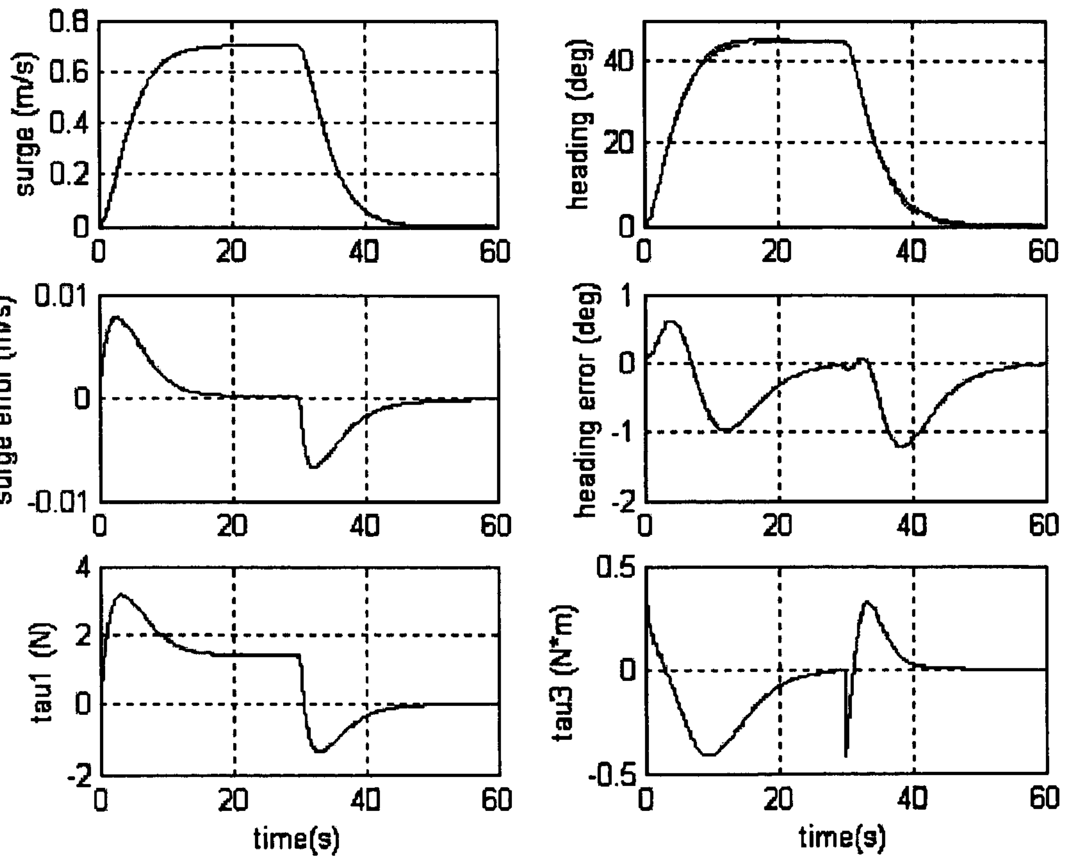


Figure 6.13: Simulated Results of the SM+PI Controller Optimised Without Waves

The tracking of the responses is very good. The surge error shows higher peaks while increasing or decreasing the speed (i.e. the response from the system is quite slow). This slowness is due to the large integral action that degrades the transient response of the system. On the other hand, the steady-state error of the response is zero, an improvement with respect to the PID and PP controllers (see Figures 6.1 and 6.7). The heading signal presents a slight overshoot and the actuators signals show two large spikes (similar to those obtained with the PID optimised without waves). This is caused by a high gain controller. The fact that the switching term is operating in the boundary layer is also increasing the proportional gain of the SM controller.

As usual the best result obtained in the optimisation runs has been tested in the real plant and the results obtained are presented in Figures 6.14 and 6.15. Figure 6.14 shows the tracking performance of the SM+PI controller once implemented in the real plant in calm waters.

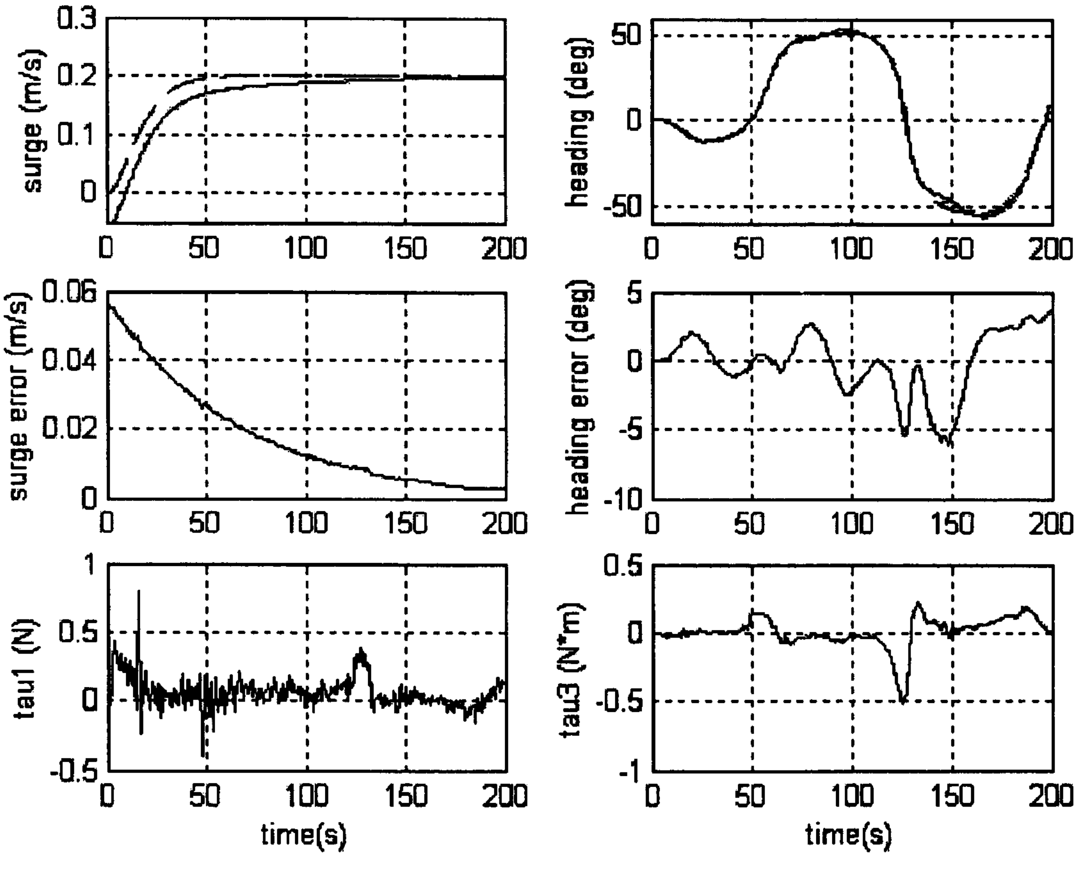


Figure 6.15: Real Results of the SM+PI Controller Optimised Without Waves When Manoeuvring in Calm Waters

SM+PI presents, like PID, a very noisy surge commanded force τ_1 and it has some problems coping with the 2nd half of the heading zig-zag when increasing the speed. Also, as predicted from the simulation results, the surge speed response is very slow due to the high integral gain of the PI. The results plotted in Figure 6.15 must be regarded with care because the dynamic positioning manoeuvre to drive CS2 to the starting point of the path was not finished by the time the reference signals were applied. This has caused an initial error of 0.05m/s in the surge.

Next, Figure 6.15 demonstrates the effect of inducing waves while the execution of the manoeuvre.

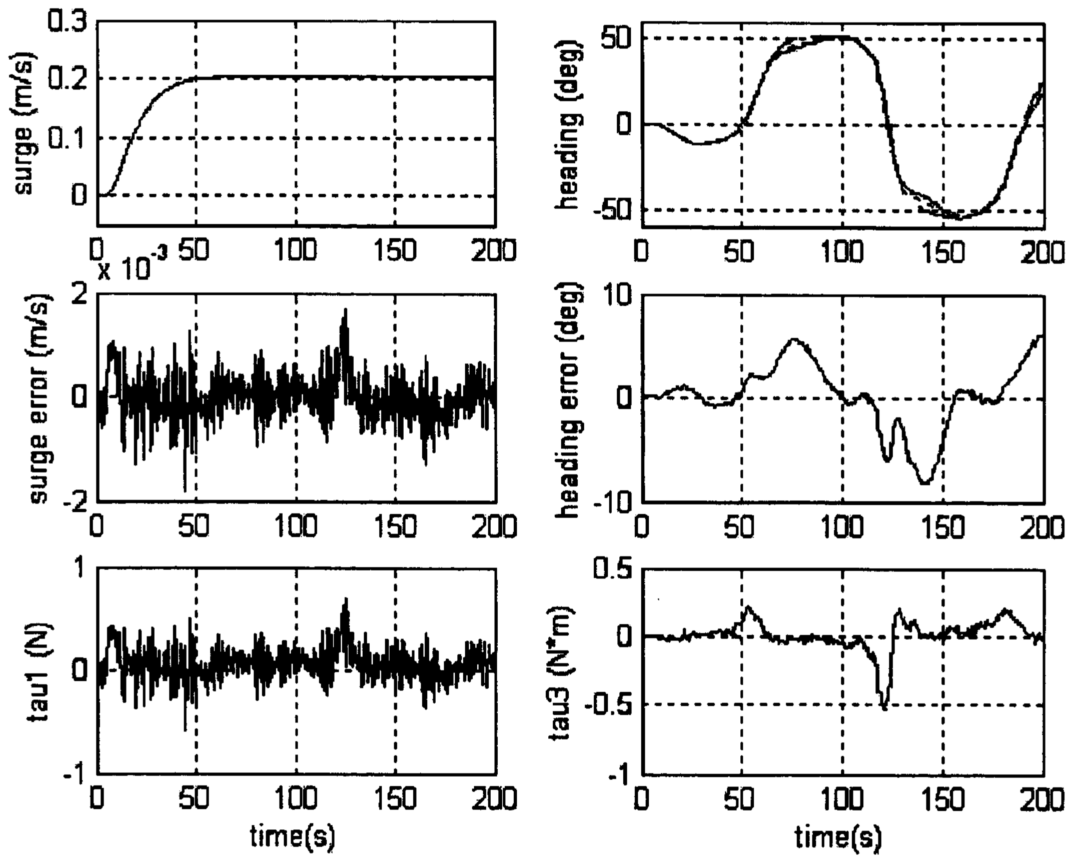


Figure 6.16: Real Results of the SM+PI Controller Optimised Without Waves When Manoeuvring in the Presence of Waves

When including waves the heading is not affected significantly and surge speed tracking improves. As expected the waves induce high frequency components in the propulsion signals, but not so much in the heading. This results from high gain control where the controller tracks the noise.

6.5.2.2. Optimisation With Waves: Simulated and Real Results

In the case of the SM+PI configuration, the GA optimisation with waves has converged in 15 generations (see Figure C.6 from Appendix C) and the best result obtained provides a cost value of 10.5 with the parameters shown in Table 6.15.

Table 6.15: Best SM+PI Results – Optimisation With Waves

	K_{Pp}	K_{Ip}
Propulsion	9.3	27

	$pole_{h1}$	$pole_{h2}$	η	Φ
Heading	-0.9929	-0.4079	99.9	50.46

After averaging the parameters obtained through the various runs we get:

Table 6.16: Average and Standard Deviation SM+PI Results
Optimisation With Waves

	K_{Pp}		K_{Ip}	
	Avg	StDev	Avg	StDev
Propulsion	9.8985	1.0486	30.1907	2.684

	$pole_{h1}$		$pole_{h2}$	
	Avg	StDev	Avg	StDev
Heading	-0.7473	0.165	-0.3117	0.2094

	η		Φ		η / Φ	
	Avg	StDev	Avg	Avg	StDev	Avg
Heading	1795.2278	3301.7165	878.3902	1617.6097	2.0095	0.1834

The heading controller resulting from the optimisation with waves only varies slightly. The poles get closer and the ratio η/Φ reduces a little. By reducing the ratio η/Φ the gain of the controller is effectively reduced. Increasing the closeness of the poles results in a fall in the damping (i.e. it yields to a response more similar to a 2nd order response, while separating the poles results in a 1st order type of response).

On the other hand the propulsion PI controller is affected by the inclusion of waves in the optimisation process. The propulsion gain is reduced by two orders of magnitude and the averaged results shown in Table 6.16 are very similar to those presented in Table 6.5 of the PID controller for propulsion optimised with waves. As for PID and PP the GA optimisation with waves results in controllers with smaller gain.

Next, Figure 6.16 shows the simulated performance of the decoupled SM+PI controllers when tracking a simultaneous double step manoeuvre.

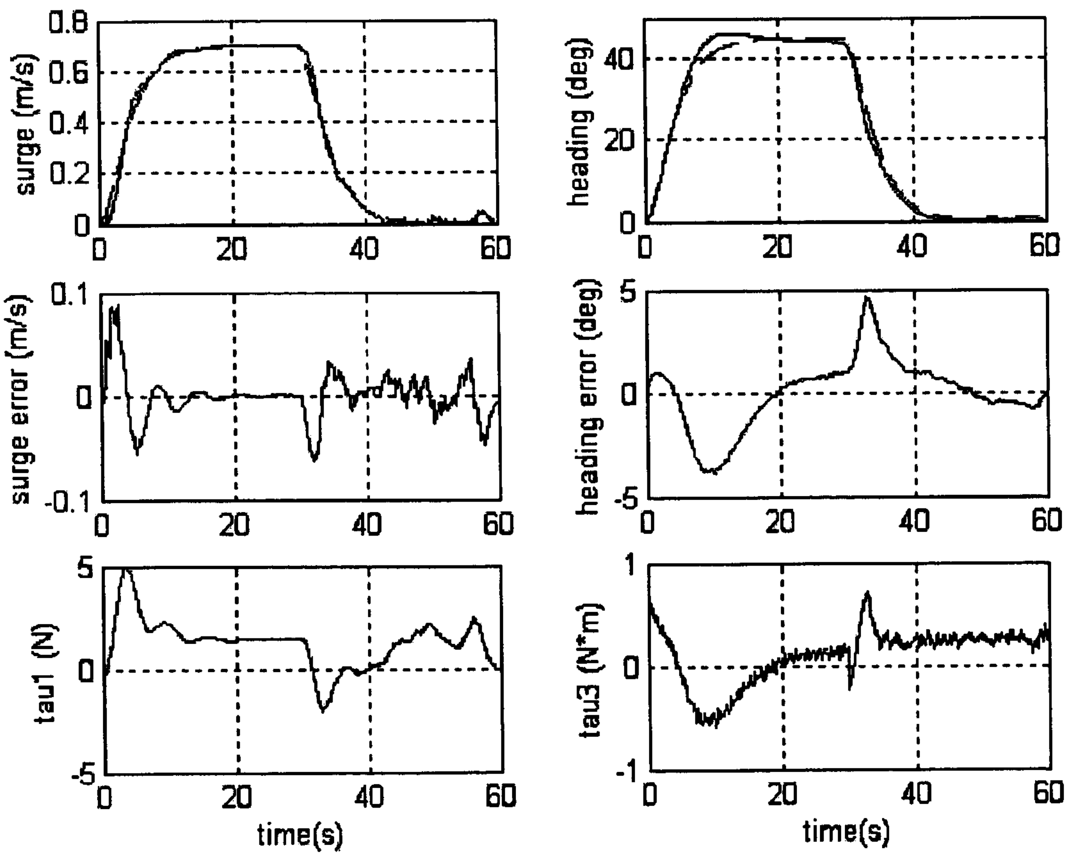


Figure 6.16: Simulated Results of the SM+PI Controller Optimised With Waves

The propulsion signals are a bit worse than those of the PID (Figure 6.4). The heading tracking error is bigger than for PID or PP. There is quite a large overshoot in the heading response due to the effect of moving the poles together, creating a more underdamped response.

The main problem that presented with this controller structure is that the PI propulsion controller had to be retuned since it was impossible to stabilise the plant using the values obtained in the GA optimisation. The initial peak in the commanded propulsion force is so high that the boat would traverse the water tank before stabilising. This is hardly surprising since the PI gains are very similar to those of the PID for propulsion, which have already given stability problems. As before the simulated responses of the model while tracking the zig-zag manoeuvre used in the real testing (see Figure F.6) do not reflect any stability problem. The GA has resorted to a high integral gain to improve the tracking in presence of noise and, although in simulation the outcome is satisfactory, the unmodelled dynamics of the system turn the system unstable when the controller is implemented in the real plant.

Therefore, the results shown in Figures 6.17 and 6.18 are obtained after retuning the PI propulsion controller to the following values:

Table 6.17: Retuned SM+PI Gains

	K_p	K_{Ip}
Propulsion	20	1

	$pole_1$	$pole_2$	η	Φ
Heading	-0.9929	-0.4079	99.9	50.46

Figure 6.17 shows the performance of the retuned controller implemented in the real plant when there are no waves generated.

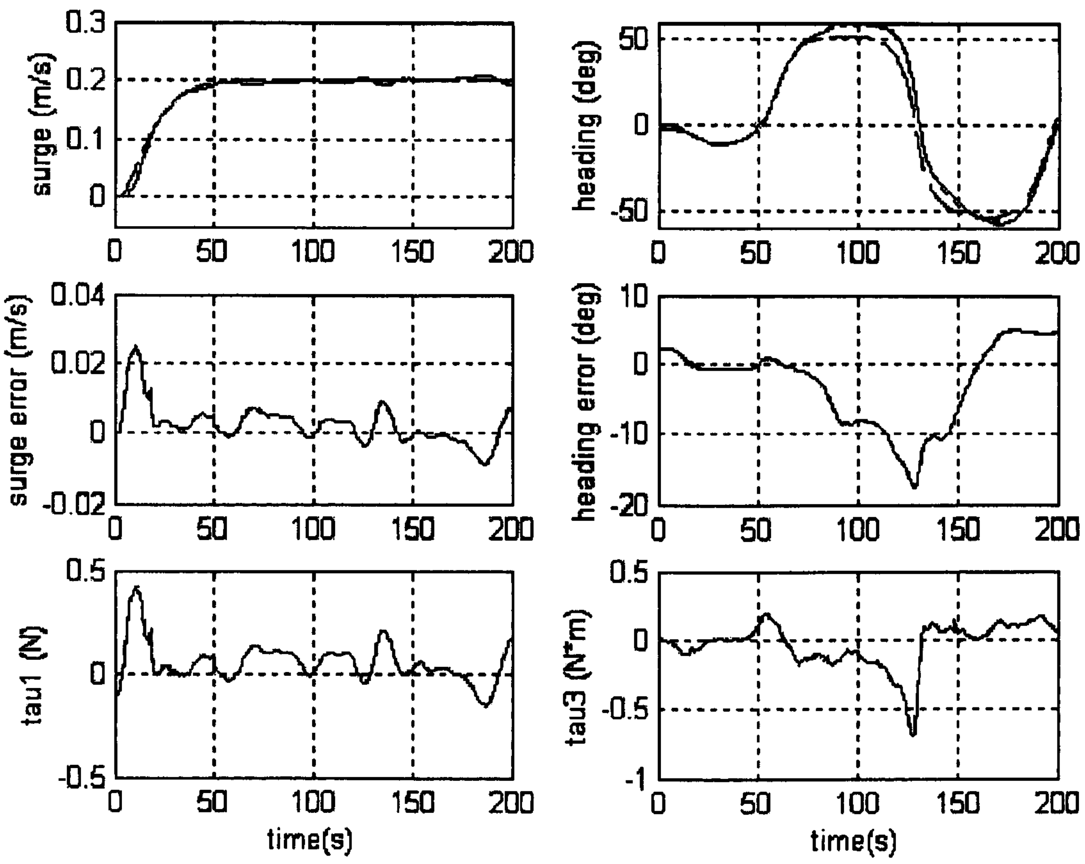


Figure 6.17: Real Results of the SM+PI Controller Optimised With Waves When Manoeuvring in Calm Waters

Although the new PI gains provide a stable result, the surge speed is still quite oscillatory.

The heading tracking is poorer than in the previous PP and PID results. The overshooting observed in the simulation results can be clearly appreciated in the real trials as well.

Figure 6.18 shows the performance of the retuned controller when implemented in the real plant with waves.

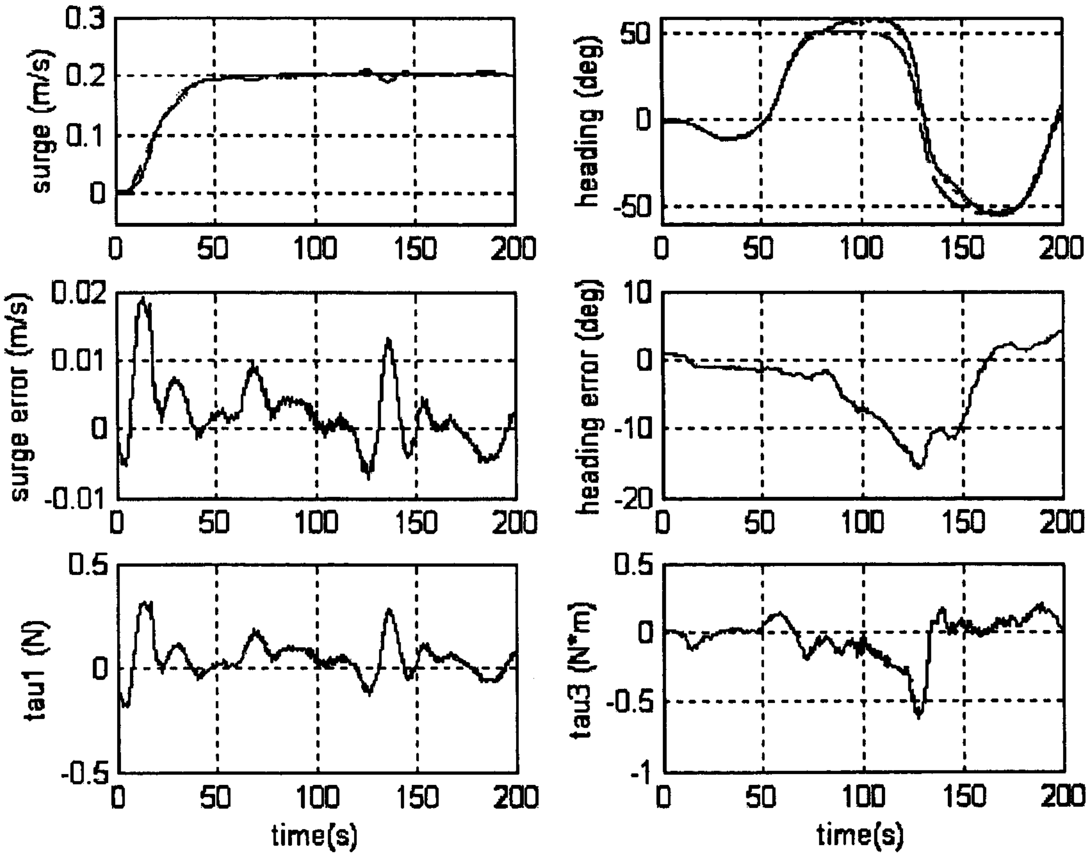


Figure 6.18: Real Results of the SM+PI Controller Optimised With Waves When Manoeuvring in the Presence of Waves

Results very similar to those plotted in Figure 6.17: degraded heading tracking and oscillatory surge speed response. Some wave-induced rippling, especially in τ_1.

6.5.3. DECOUPLED SLIDING MODE CONTROLLERS

6.5.3.1. Optimisation Without Waves: Simulated and Real Results

For the two decoupled Sliding Mode configuration, the best result converges to a cost value of 4.5, with the parameters from Table 6.18. The GA has converged to the near-optimal solution in 14 generations as shown in Figure C.7 from Appendix C.

Table 6.18: Best SM Results - Optimisation Without Waves

	$pole_1$	$pole_2$	η	Φ
Propulsion	-0.01029		0.7912	0.1701
Heading	-1.4	-0.256	3.8	1.991

When averaging the gain results obtained through different runs, the average and standard deviation values are:

Table 6.19: Average and Standard Deviation SM Results
Optimisation Without Waves

	$pole_1$		$pole_2$	
	Avg	StDev	Avg	StDev
Propulsion	-0.0127	0.0055		
Heading	-1.0779	0.3518	-0.2862	0.0359

	η		Φ		η / Φ	
	Avg	StDev	Avg	StDev	Avg	StDev
Propulsion	1507.8665	2676.1996	310.5498	565.7521	5.1758	0.7075
Heading	415.4362	574.0593	201.0921	277.1211	2.0742	0.2335

The GA optimisation for the heading controller has converged to similar values that for the SM+PI case. The propulsion controller converges to a pole close to zero, and the same observation made for the SM+PI controller can be made here: there is not convergence to similar values of the switching gain (η) or boundary layer thickness (Φ) but to the ratio of both values, indicating that the SM controllers are operating in the boundary layer.

Figure 6.19 shows the simulated performance of the SM controllers when tracking the simultaneous double step manoeuvre used in the GA optimisation.

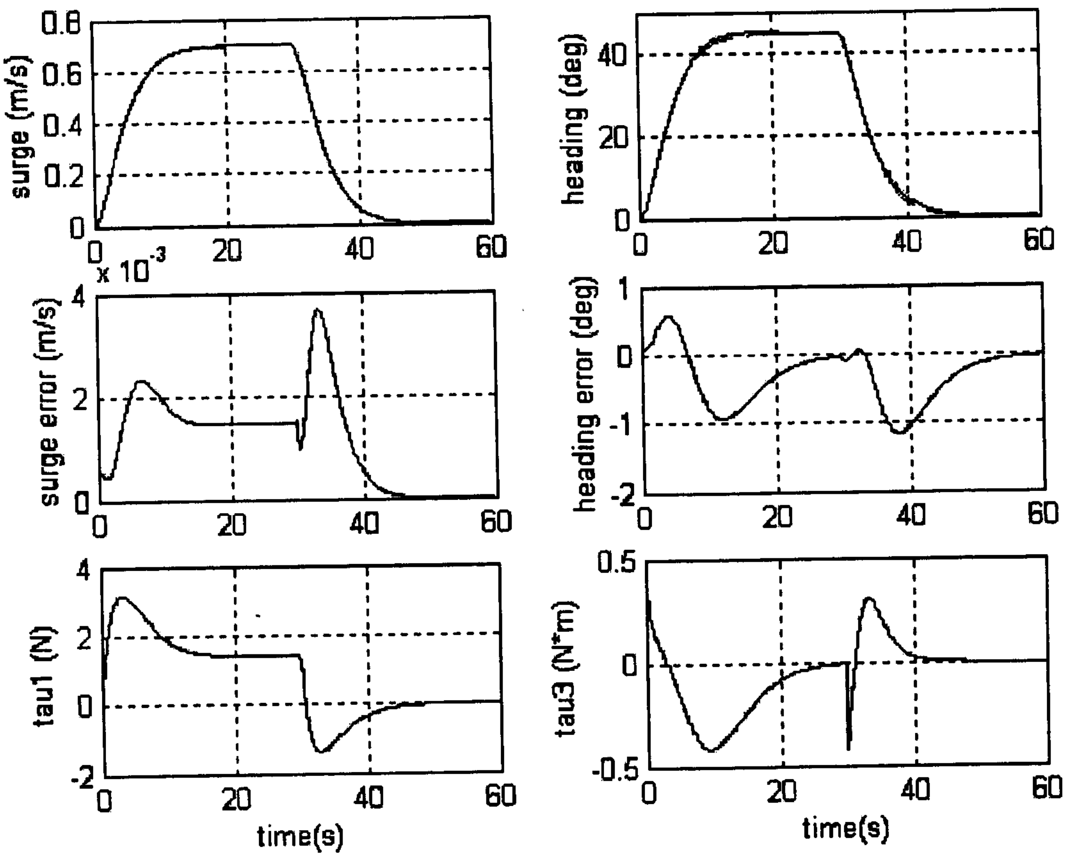


Figure 6.19: Simulated Results of the SM Controller Optimised Without Waves

The heading results are equivalent to those shown in Figure 6.15 (SM+PI optimised without waves). Both controllers present a slight overshooting in the heading responses, due to SM being a high gain controller. The propulsion tracking is the best obtained so far in simulation, keeping the surge error always under 4 mm/s, although it shows a small steady state error that the PI controller from the SM+PI configuration has been able to eliminate.

The results obtained while implementing the optimised SM controller in the real plant are plotted in the next graphs. In Figure 6.20 the testing manoeuvre is performed without waves and in Figure 6.21 the effect of generated waves in the water tank is analysed. The reference signals are represented as a dashed line in both figures.

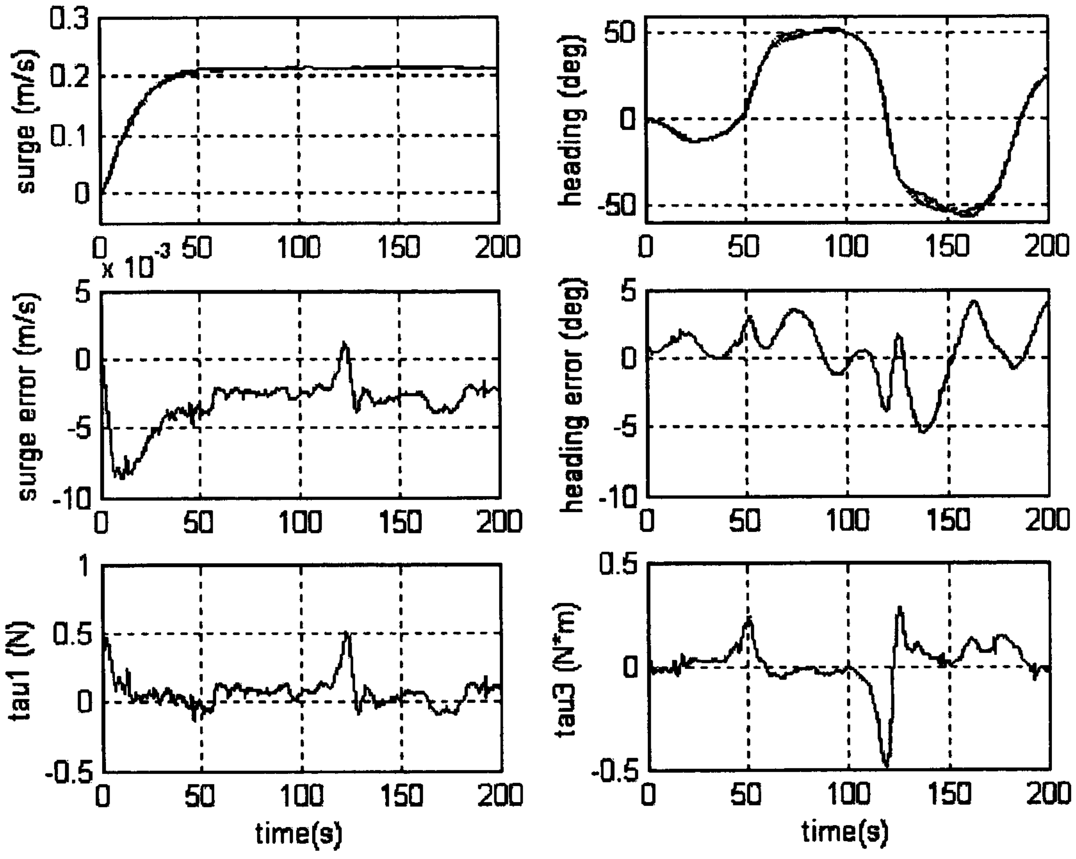


Figure 6.20: Real Results of the SM Controller Optimised Without Waves When Manoeuvring in Calm Waters

Again the tracking is good, although the heading tracking degrades in the second half of the zig-zag as in previous results. When comparing SM versus PID and PP it can be seen that the control signals obtained with SM are less noisy than in the PID case. As it was to be expected, SM provides a heading tracking very similar to the one obtained with the SM+PI controller. Also, the heading control provided by the SM controller is better than that provided by PP or PID. The heading error is smaller and, although the peaks in the control signal are slightly larger in magnitude for SM, the noise is so much reduced that the control effort for SM is smaller. Regarding the propulsion subsystem, the SM surge error is slightly larger that the error in PID or PP control, but again much less noisy. When compared with the manually tuned SM (see Figure 4.19) it can be observed how much better the responses are even though they were obtained with the suboptimal system.

Figure 6.21 shows the results obtained by CS2 when performing the same manoeuvre but in the presence of waves in the water tank.

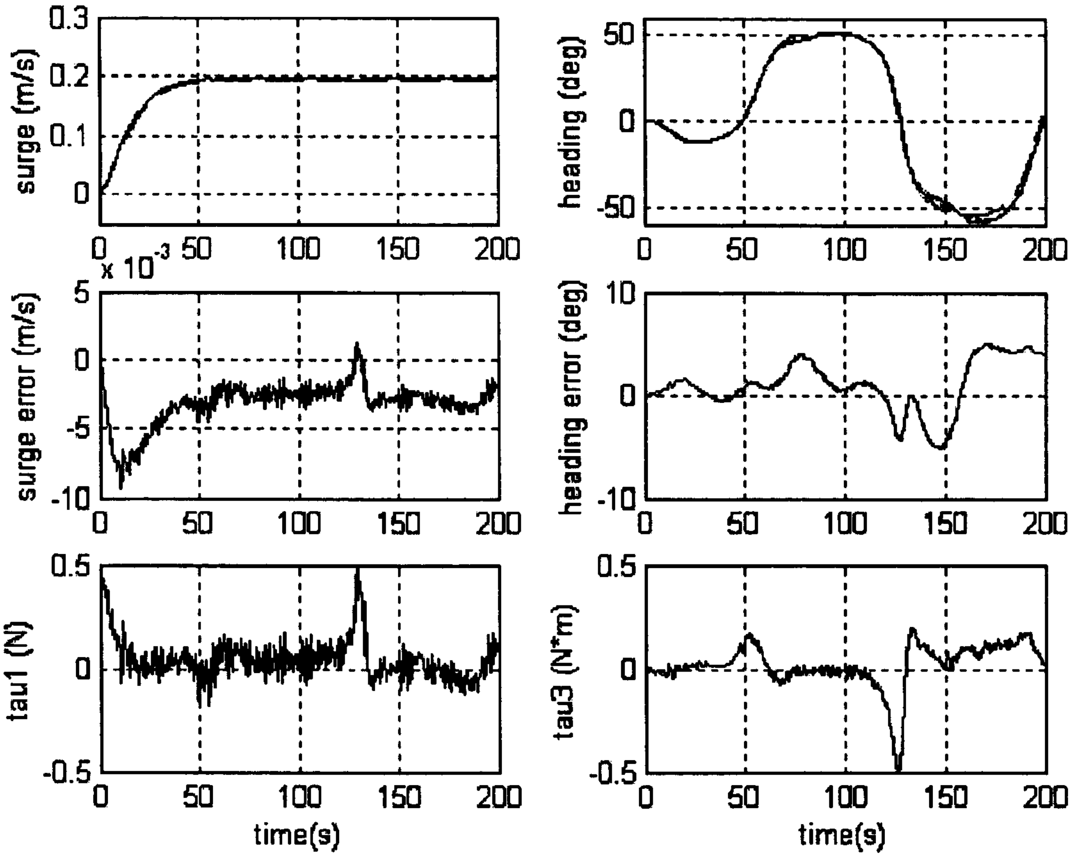


Figure 6.21: Real Results of the SM Controller Optimised Without Waves When Manoeuvring in the Presence of Waves

It can be observed from the figure that the inclusion of waves does not degrade the tracking significantly though, it induces high frequency components in the signals, especially in the propulsion subsystem.

6.5.3.2. Optimisation With Waves: Simulated and Real Results

For the two decoupled Sliding Mode configuration, The GA has converged in 9 generations (see Figure C.8 from Appendix C). The best result has got a cost value assigned of 11.5, with the following parameters values:

Table 6.20: Best SM Results - Optimisation With Waves

	$pole_1$	$pole_2$	η	Φ
Propulsion	-0.00124		762.5	864
Heading	-0.945	-0.4324	9.05	4.305

When averaging the gain results obtained through 15 runs, the average and standard deviation values are:

Table 6.21: Average and Standard Deviation SM Results Optimisation With Waves

	$pole_1$		$pole_2$	
	Avg	StDev	Avg	StDev
Propulsion	-0.0028	0.0028		
Heading	-0.9739	0.2939	-0.4663	0.0620

	η		Φ		η / Φ	
	Avg	StDev	Avg	StDev	Avg	StDev
Propulsion	1822.2413	2386.4264	2384.4492	3271.2975	0.7912	0.0948
Heading	1483.8938	3114.3927	879.2401	1944.4034	1.8496	0.2873

When comparing these optimised results with those of Tables 6.18 and 6.19 we can see that including waves in the simulation makes the GA converge to a similar heading controller with somewhat closer poles and analogous η/Φ ratio. As in the SM+PI case, closer poles lead to a more underdamped response.

On the other hand, the GA search converges to a quite different propulsion controller where the pole is closer to zero and the η/Φ ratio has been reduced by one order of magnitude. By placing the propulsion pole closer to zero the GA is trying to achieve an integral action that will improve the tracking. Moreover, reducing the ratio η/Φ will reduce the gain of the controller, given that SM is operating in the boundary layer.

Below, Figure 6.22 shows the simulated performance of the decoupled SM controllers when tracking the simultaneous double step manoeuvre for surge and heading used in the GA optimisation with waves. The desired responses have been represented in a dashed line.

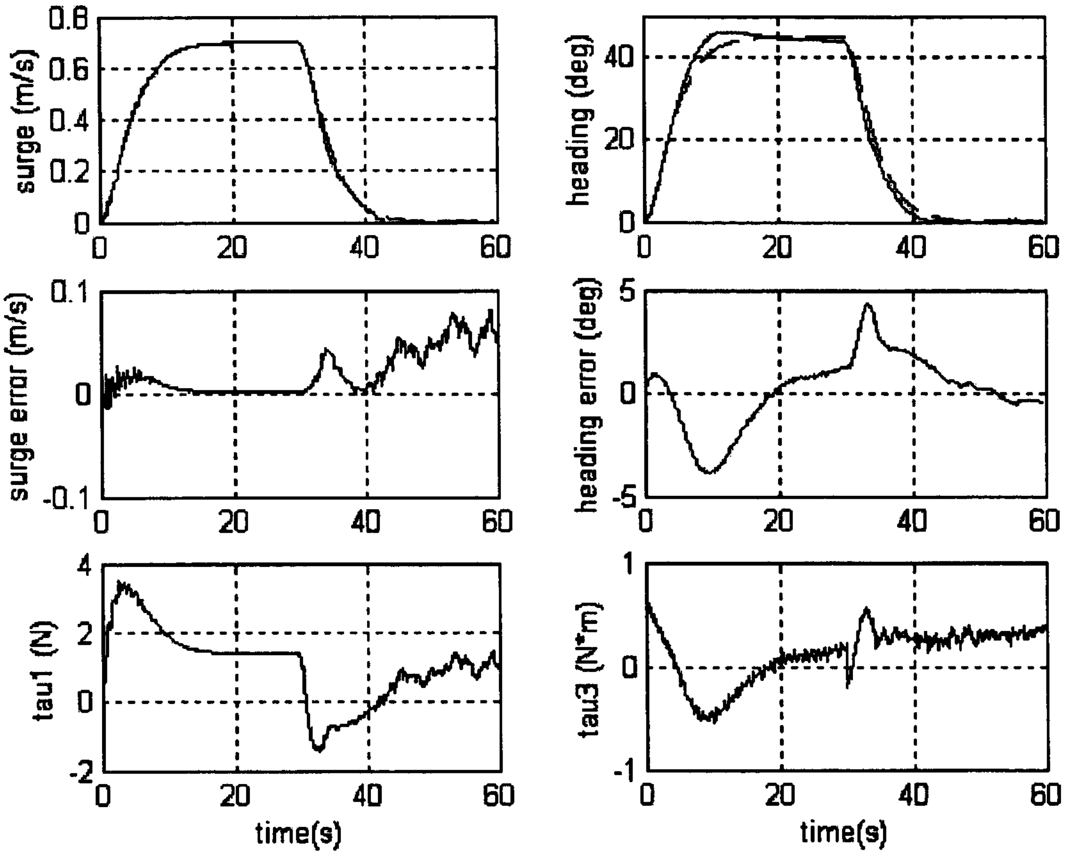


Figure 6.22: Simulated Results of the SM Controller Optimised With Waves

As expected the performance of the SM controller for heading is equivalent to that of the SM+PI configuration (Figure 6.12). The SM propulsion control is better than that of SM+PI but at the expense of a larger control effort in τ_l.

The above figure shows very good surge tracking during the first half of the simultaneous double step (when the encounter angle between the heading and the direction of the waves is 90°) though the tracking is worse at the very beginning and in the second half of the manoeuvre (when the encounter angle between the heading and the direction of the waves is 135°). It seems that the SM propulsion controller handles better lateral waves than approaching ones. The degradation of the performance at the end of the surge response is due to the slow surge speed that diminishes the effectiveness of the actuators for compensating the waves effect.

Next Figures 6.23 and 6.24 illustrate the results obtained in the real tests in the MCLab when the SM controllers had been implemented in the real plant. Figure 6.23 shows the execution of the manoeuvre in calm waters.

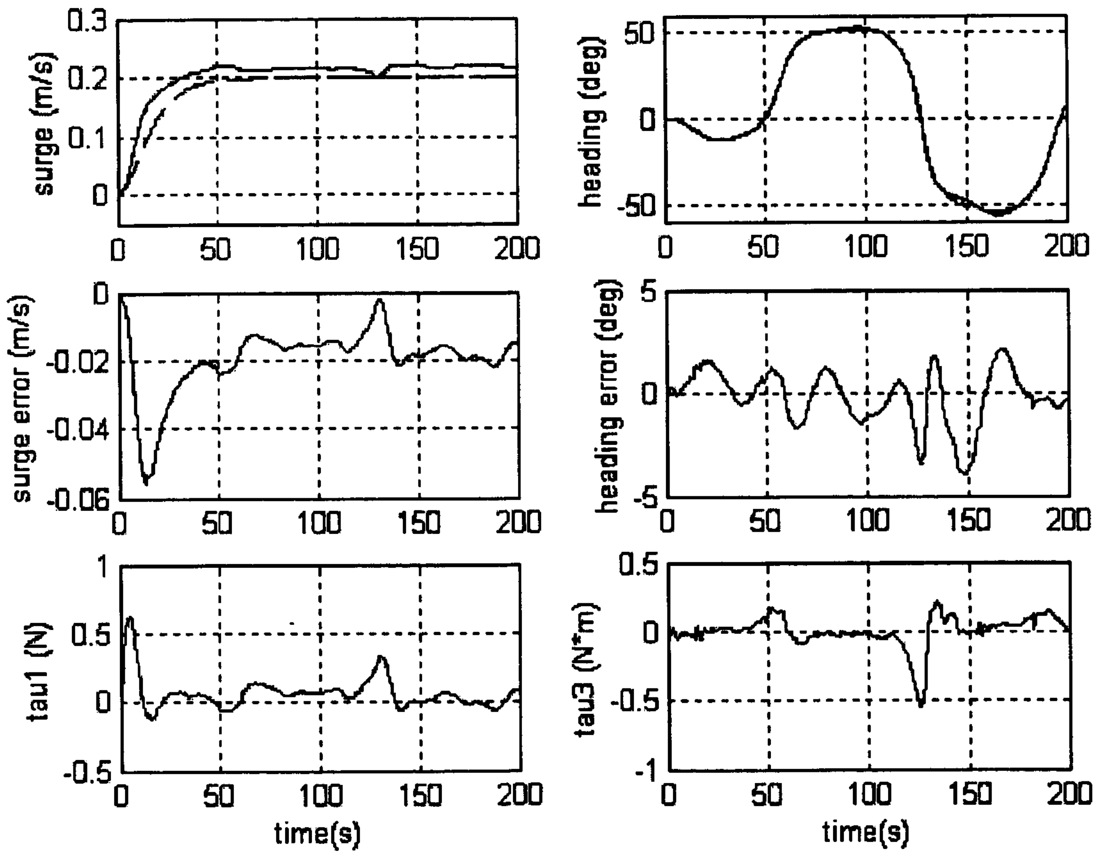


Figure 6.23: Real Results of the SM Controller Optimised With Waves When Manoeuvring in Calm Waters

The set of plots shows that this tuning solution for the SM controller provides very good heading tracking (much better than PID, PP or SM+PI and even better than the previous SM optimised without waves), but quite poor propulsion control. There is a big steady-state error in surge speed and the signal is quite oscillatory. This figure further confirms the reduction in noise in the signals when using SM as opposed to PID or PP.

The following Figure 6.24 illustrates the results obtained when the zig-zag manoeuvre was performed in the presence of waves.

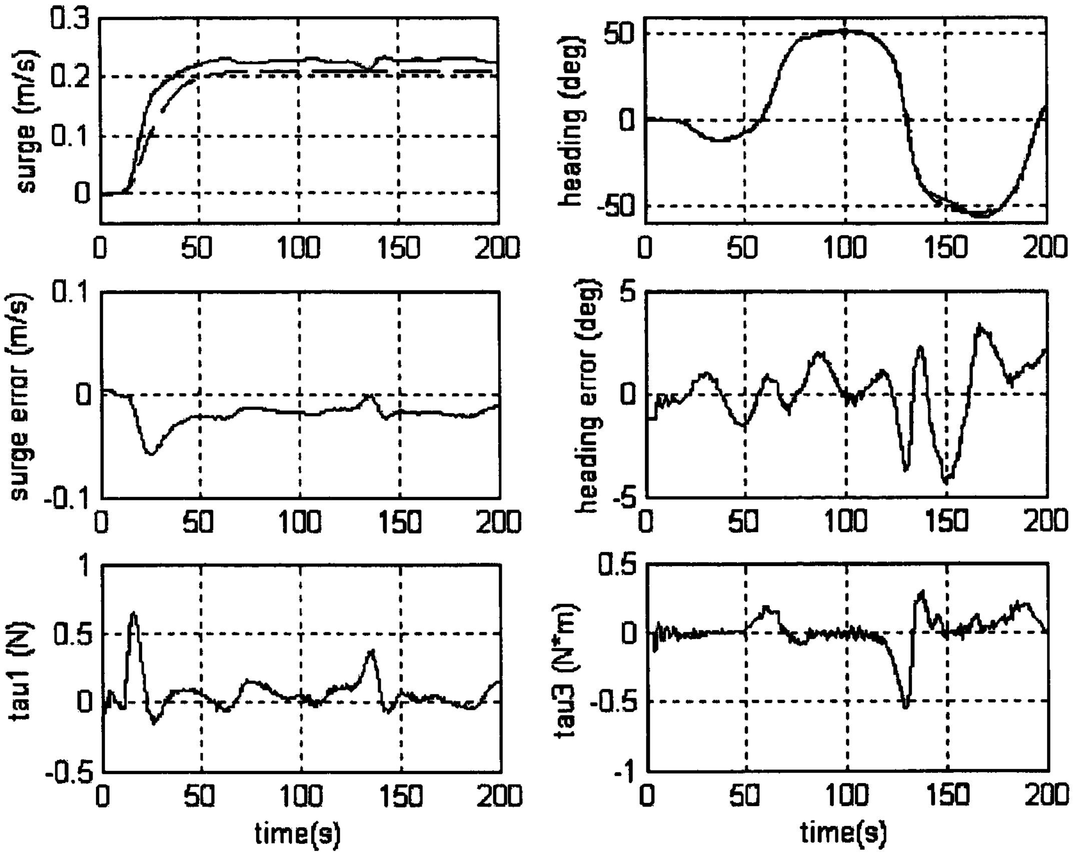


Figure 6.24: Real Results of the SM Controller Optimised With Waves When Manoeuvring in the Presence of Waves

When including waves, they do not affect the heading and propulsion tracking and the commanded forces signals are definitely less influenced than those of the SM controller optimised without disturbances.

Once more, the controller optimised with waves provides smoother commanded signals that the one optimised without waves, while the propulsion tracking degrades.

6.5.4. SUMMARY OF SLIDING MODE RESULTS

The previous section has analysed the results obtained in the implementation of two controller structures based on non-linear Sliding Mode techniques.

One of the structures consisted of two decoupled SM controllers for heading and propulsion. As usual, it has been optimised with and without waves.

The results obtained in the GA optimisation without waves are very good. Tracking is fine (heading tracking is improved compared to PID and PP) and the controller signals are smoother than for PID and PP (especially τ_l).

When optimised in the presence of waves the resulting heading SM parameters do not vary a lot but the propulsion SM parameters do. As in the PID and PP case the SM control obtained focuses on reducing the switching in the actuator by reducing the control effort and that leads to a poorer surge speed tracking. Hence, there is a trade off between the actuator effort and the tracking error.

As in the PP case no retuning whatsoever was needed for the real implementation of the SM controller.

The other structure analysed consisted of a SM controller for heading and a PI for propulsion. This configuration exhibits some of the characteristics of SM and PID previously mentioned. These are discussed below.

When optimised without waves the heading performance is good, equivalent to that of the decoupled SM control. However, τ_l shows a very noisy operation like in the PID case.

The most relevant aspect of this SM+PI tuning when compared with previous SM and PID control structures is the slow propulsion response. This is due to a large integral gain and small proportional term. The latter is caused by the omission of a derivative term, which would compensate for overshoots and oscillations in the response. By including a derivative the proportional gain could be increased and the resulting transient response quickened. Since there is no derivative term the size of the proportional gain has to be limited to ensure a sufficiently damped response.

The SM+PI controller optimised with waves has the same stability problem that the PID controller has (therefore the PI controller for propulsion needed to be retuned). This is hardly surprising since the optimisation of both propulsion PID and PI controllers converged to similar results. Both converged to high integral actions that worked well in simulation but produced instability in presence of the real plant unmodelled dynamics. This raises doubts about the accuracy of the model used since that effect was not appreciated in the simulation results.

The retuned controller performance is consistent with previous observations of operation of controllers optimised with waves: reduced ripples in τ_1 to the detriment of accurate speed tracking.

The overall performance of the decouple SM controller is better than the performance of SM+PI.

6.6. H$_\infty$ OPTIMAL CONTROL

6.6.1. OPTIMISATION PROCEDURE

H$_\infty$ is the most complicated optimisation problem of those tackled, for the large number of parameters to optimise and the difficulty of finding an admissible controller. The optimisation has been performed using two GA models: the GA used for the rest of the controllers plus the Structured Genetic Algorithm (sGA) proposed in Section 5.4.

The generic GA assumes a predefined 2nd order transfer function with a single zero structure for the weighting functions of the error signals. This way the GA can choose the structure of the transfer function, e.g. if a first order weighting function is more appropriate the GA can choose α to be equal to β and thus the zero cancels one of the poles. Thus, the weighting functions of the error signals are as follows:

$$W_u(s) = \frac{K_u'(s+\alpha_u)}{(s+\beta_u)(s+\gamma_u)} \tag{6.2}$$

$$W_r(s) = \frac{K_r'(s+\alpha_r)}{(s+\beta_r)(s+\gamma_r)} \tag{6.3}$$

To avoid a very high order controller, the weightings for τ_{1com}, τ_{2com} and τ_{3com} (i.e. W_{t1}, W_{t2} and W_{t3}) have been kept constant. Thus, the vector to be minimised will be $z = [W_u(s)\cdot(u_c-u),\ W_r(s)\cdot(r_c-r),\ K_{t1}\cdot\tau_{1com},\ K_{t2}\cdot\tau_{2com},\ K_{t3}\cdot\tau_{3com}]^T$. The GA optimises the eleven parameters shown in Table 6.22 that define this predetermined structure.

Table 6.22: Parameters to Optimise for H$_\infty$ Control Using GA

K_u'	α_u	β_u	γ_u	K_r'	α_r	β_r	γ_r	K_{t1}	K_{t2}	K_{t3}

All the values are encoded to be positive to ensure stability and minimum-phase characteristics.

On the other hand, the sGA optimises not only the parameters of the weighting functions but also the structure. This is done by means of the multilayered chromosome structure characteristic of sGA, where some control genes can activate and deactivate sets of parameter genes. Thus, the sGA optimises the parameters shown in the previous Table 6.22, and also the control genes (w_u and w_r) that define the structure of the weighting functions of the error signals (see Table 6.23). The weighting functions for the actuator signals have been preset to be just gains as in the normal GA optimisation. Hence, the 13 parameters to optimise by the sGA are as follows:

Table 6.23: Parameters to Optimise for H_∞ Control Using sGA

K_u'	α_u	β_u	γ_u	K_r'	α_r	β_r	γ_r	K_{t1}	K_{t2}	K_{t3}	w_u	w_r

As for the GA optimisation all the parameters are encoded to be positive values to ensure stability and minimum-phase characteristics.

The control genes w_u and w_r define the structure to be used for $W_u(s)$ and $W_r(s)$ as shown in Table 6.24, being i equal to u or r:

Table 6.24: Structures Represented by the Control Genes

w_i	$W_i(s)$
0	K_i'
1	$\frac{K_i'}{s+\beta_i}$
2	$\frac{K_i'\cdot(s+\alpha_i)}{s+\beta_i}$
3	$\frac{K_i'\cdot(s+\alpha_i)}{(s+\beta_i)\cdot(s+\gamma_i)}$

6.6.2. GA PARAMETRIC OPTIMISATION

6.6.2.1. Optimisation Without Waves: Simulated and Real Results

The best result of the 15 runs of the GA provided a cost value of 1.6 with the H_∞ gains shown in Table 6.25. The GA has converged in 32 generations (see Figure C.9 in Appendix C). It is remarkable how the generation of convergence is far later than in any of the previous GA optimisations, showing the difficulty of the problem.

Table 6.25: Best H_∞ Results – GA Optimisation Without Waves

K_u'	α_u	β_u	γ_u	K_r'	α_r	β_r	γ_r
0.9276	98.92	0.2171	0.01999	0.2901	5.63	0.05868	0.06149

K_{t1}	K_{t2}	K_{t3}
0.03675	560.7	0.0532

After averaging the parameters obtained through the 15 runs yields:

Table 6.26: Average and Standard Deviation H_∞ Results
GA Optimisation Without Waves

K_u'		α_u		β_u		γ_u	
Avg	StDev	Avg	StDev	Avg	StDev	Avg	StDev
134.773	349.178	1095.75	2441.71	25.5077	34.5249	9.3520	26.1045

K_r'		α_r		β_r		γ_r	
Avg	StDev	Avg	StDev	Avg	StDev	Avg	StDev
202.316	738.750	64.6526	223.283	7.5127	23.5635	6.1168	15.6352

K_{t1}		K_{t2}		K_{t3}	
Avg	StDev	Avg	StDev	Avg	StDev
3.4517	12.8174	1019.2729	1627.9833	0.0478	0.0208

Looking at the standard deviation values from Table 6.26 it can be seen that the results obtained in the various GA runs are very different. This can be caused by a very flat search space where large variations in the genotype do not alter the cost function much (i.e. a plateau). Also the difficulty in the optimisation problem might be causing the GA to converge to non-optimal solutions. If an individual results in a non-admissible controller it is assigned a very large cost value (i.e. a penalty). Therefore if that occurs often it can be misleading for the GA, which cannot discriminate between all these individuals with the same cost.

The next Figure 6.25 shows the performance of the H_∞ controller when using the weighting functions from Table 6.25.

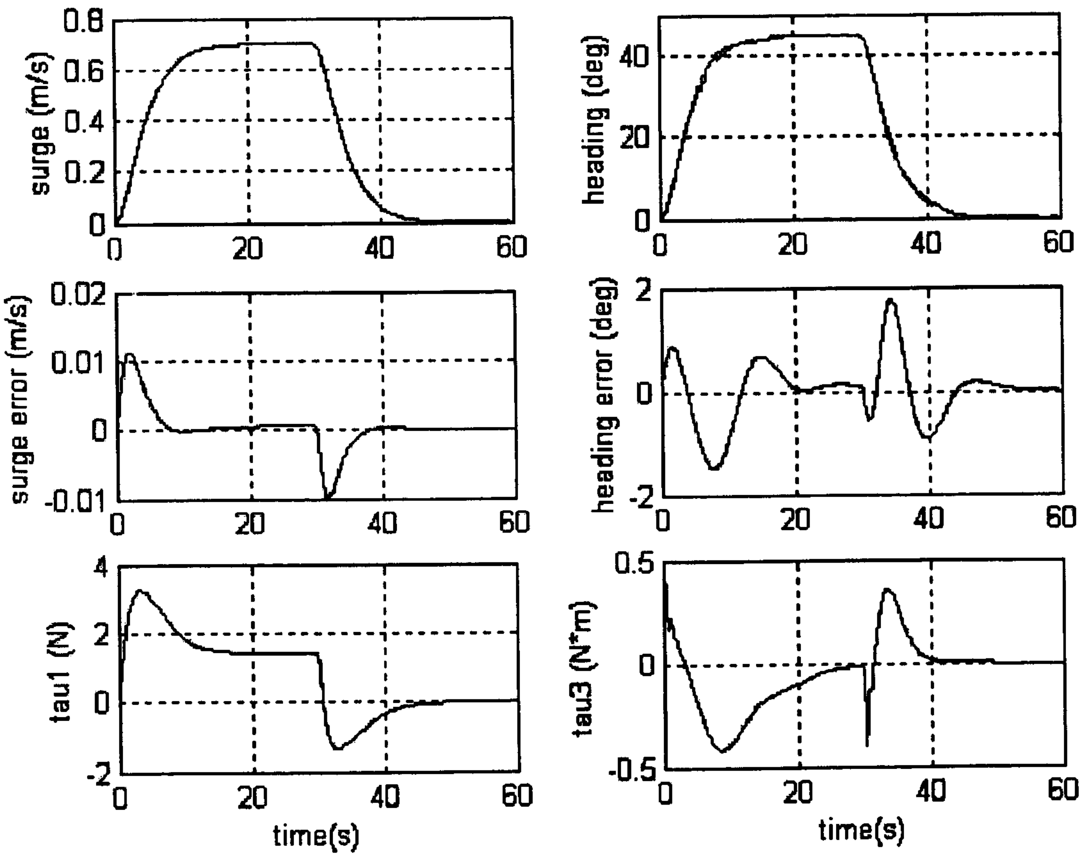


Figure 6.25: Simulated Results of the H_∞ Controller GA Optimised Without Waves

The simulated results from the previous graph are good. The heading tracking is slightly worse than in previous controllers but the overall performance is very satisfactory.

The responses obtained once this controller was implemented in the real plant and tested in calm waters are shown in Figure 6.26.

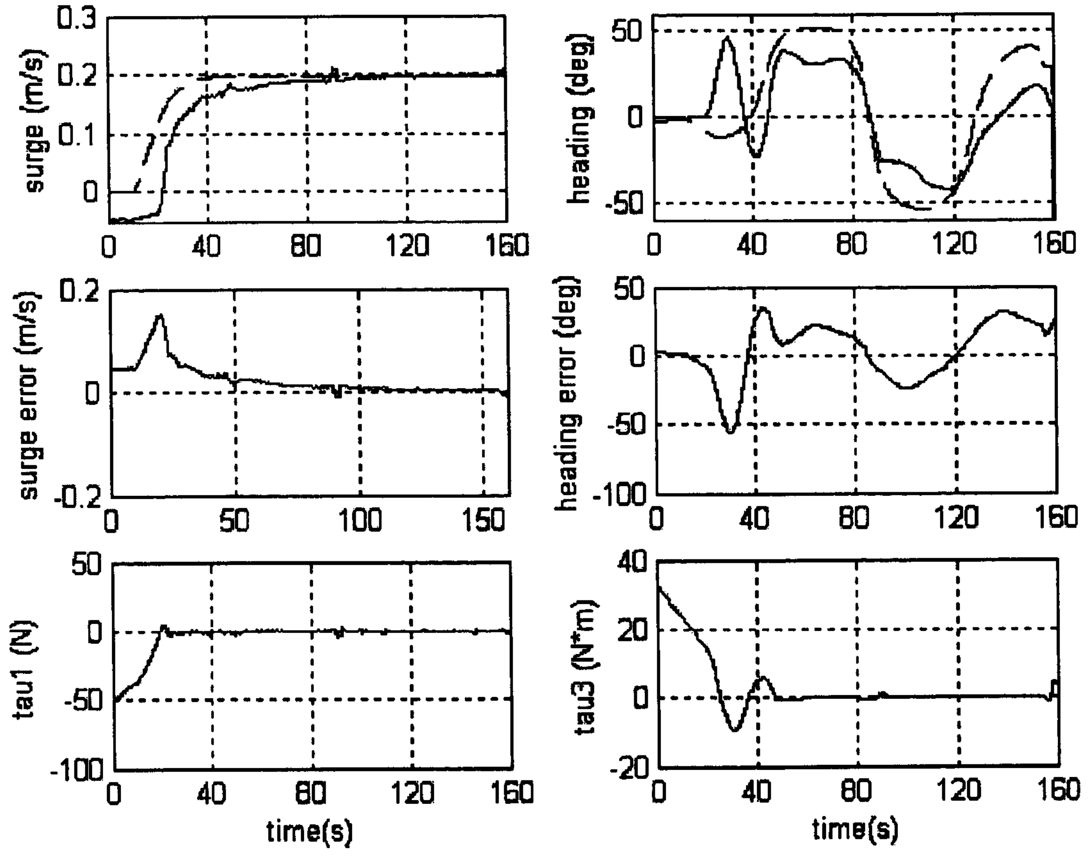


Figure 6.26: Real Results of the H_∞ Controller GA Optimised Without Waves When Manoeuvring in Calm Waters

The responses shown in the figure are very unsatisfactory. After an initial stage of instability the ship manages to follow the desired responses, although the tracking is very poor (especially in the heading response) and the actuator usage is excessive. tThe results obtained are still better that those of the manually tuned H_∞, which are totally unstable (see Figure 4.28). Figure F.7 in Appendix F shows the simulated responses obtained when the reference signals used in the real testing are used in simulation. As in previous cases the differences are extremely significant. The real results for H_∞ were obtained using the improved facilities; therefore they are free of the noise that characterised the responses of previous controllers.

Figure 6.27 shows the responses obtained using the optimised H_∞ controller from Table 6.25) when the manoeuvre was conducted in the presence of waves in the water tank. It can be observed that the inclusion of waves does not affect significantly a very poor performance.

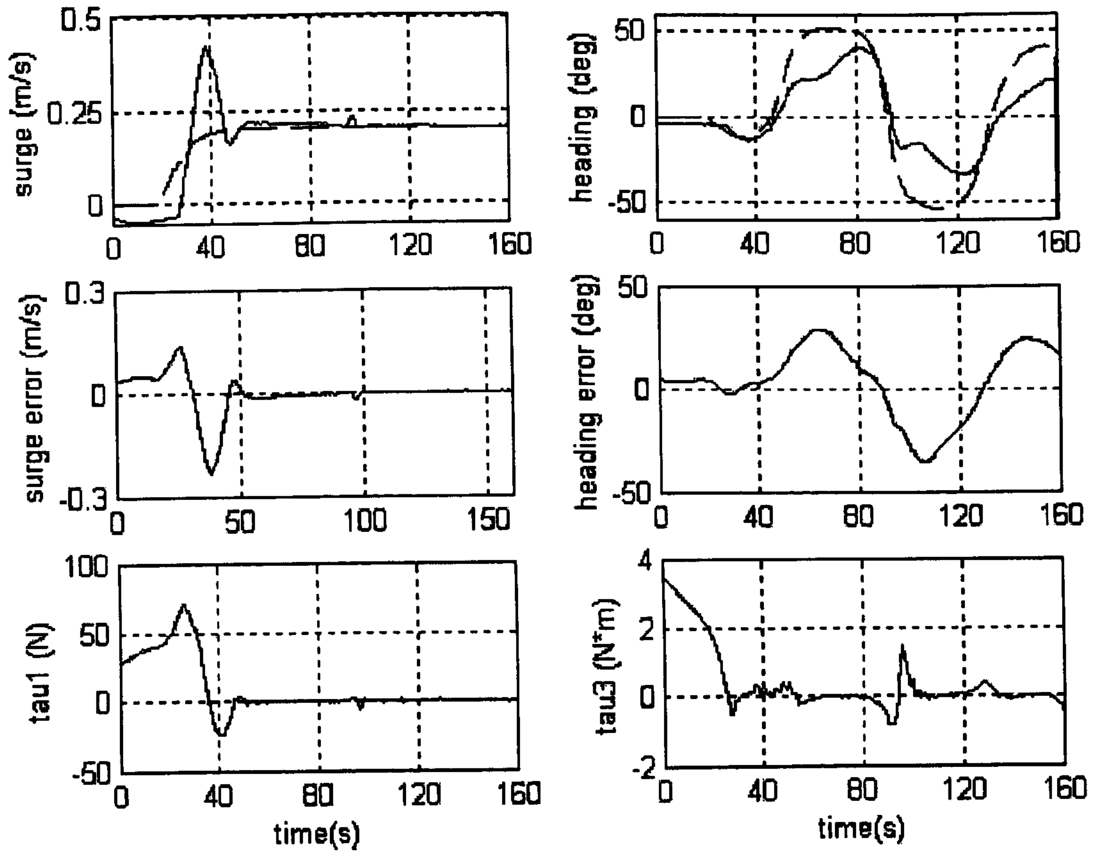


Figure 6.27: Real Results of the H∞ Controller GA Optimised Without Waves When Manoeuvring in the Presence of Waves

6.6.2.2. Optimisation With Waves: Simulated and Real Results

The GA with waves has converged in the last generation (see Figure C.10 from Appendix C) to a cost value of 33.4 with the following H∞ gains:

Table 6.27: Best H∞ Results - GA Optimisation With Waves

K_u'	α_u	β_u	γ_u	K_r'	α_r	β_r	γ_r
3.465	8.215	0.06798	16.97	0.058	0.6639	0.0587	0.01419

K_{t1}	K_{t2}	K_{t3}
0.00698	0.4741	0.0133

Averaging the parameters obtained through the 15 runs gives the values from Table 6.28. As in the optimisation without waves the tables show disparity among the results from various GA runs.

Table 6.28: Average and Standard Deviation H_∞ Results
GA Optimisation With Waves

K_u'		α_u		β_u		γ_u	
Avg	StDev	Avg	StDev	Avg	StDev	Avg	StDev
61.3889	185.364	1010.02	2667.48	16.2042	30.7471	18.5001	29.97

K_r'		α_r		β_r		γ_r	
Avg	StDev	Avg	StDev	Avg	StDev	Avg	StDev
0.3167	0.4563	14.2571	40.9211	12.1895	30.8155	0.0433	0.0576

K_{t1}		K_{t2}		K_{t3}	
Avg	StDev	Avg	StDev	Avg	StDev
0.0286	0.0324	1928.6634	3039.3280	0.0321	0.0333

Figure 6.28 illustrates the simulation results of the H_∞ controller from Table 6.27.

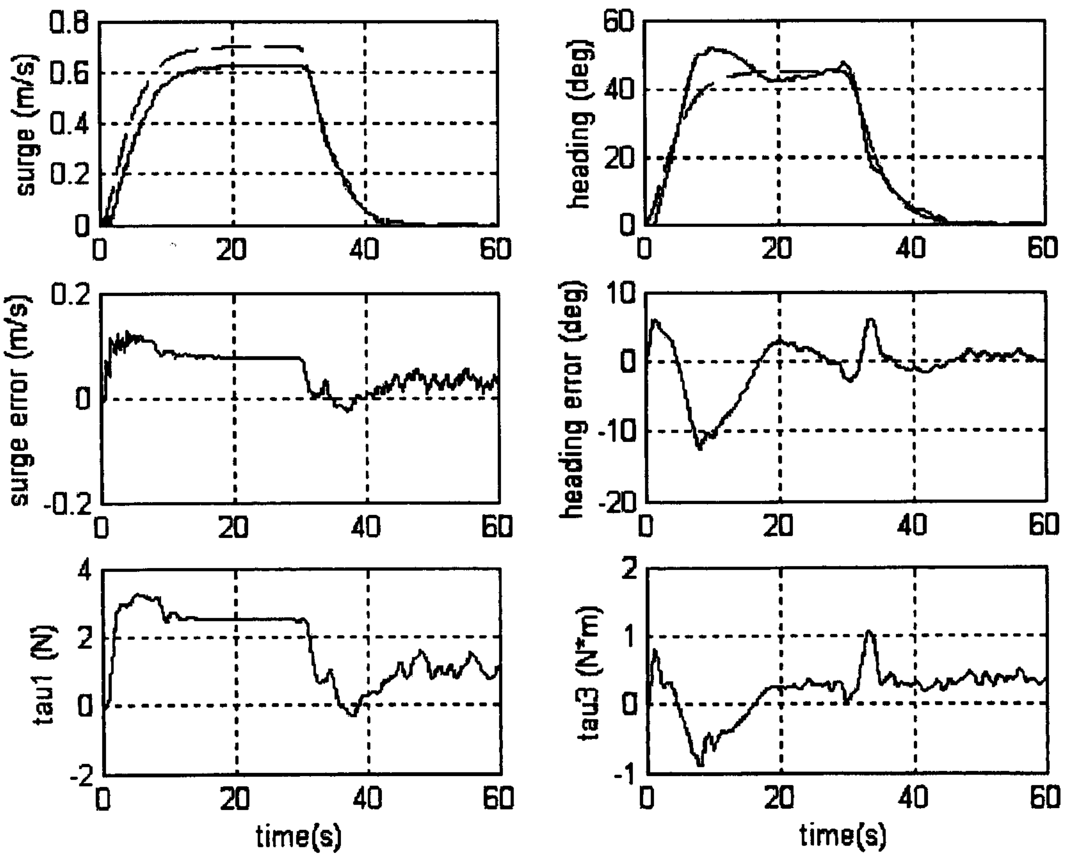


Figure 6.28: Simulated Results of the H_∞ Controller GA Optimised With Waves

The plots illustrate the degradation in surge speed tracking previously observed in the optimisations with waves. As in Figure 6.25 the heading tracking is worse than for other controllers.

Figures 6.29 and 6.30 show the responses obtained when the H_∞ controller has been implemented in the real plant. Figure 6.29 presents the responses obtained with the H_∞ controller from Table 6.27 when manoeuvring in calm waters.

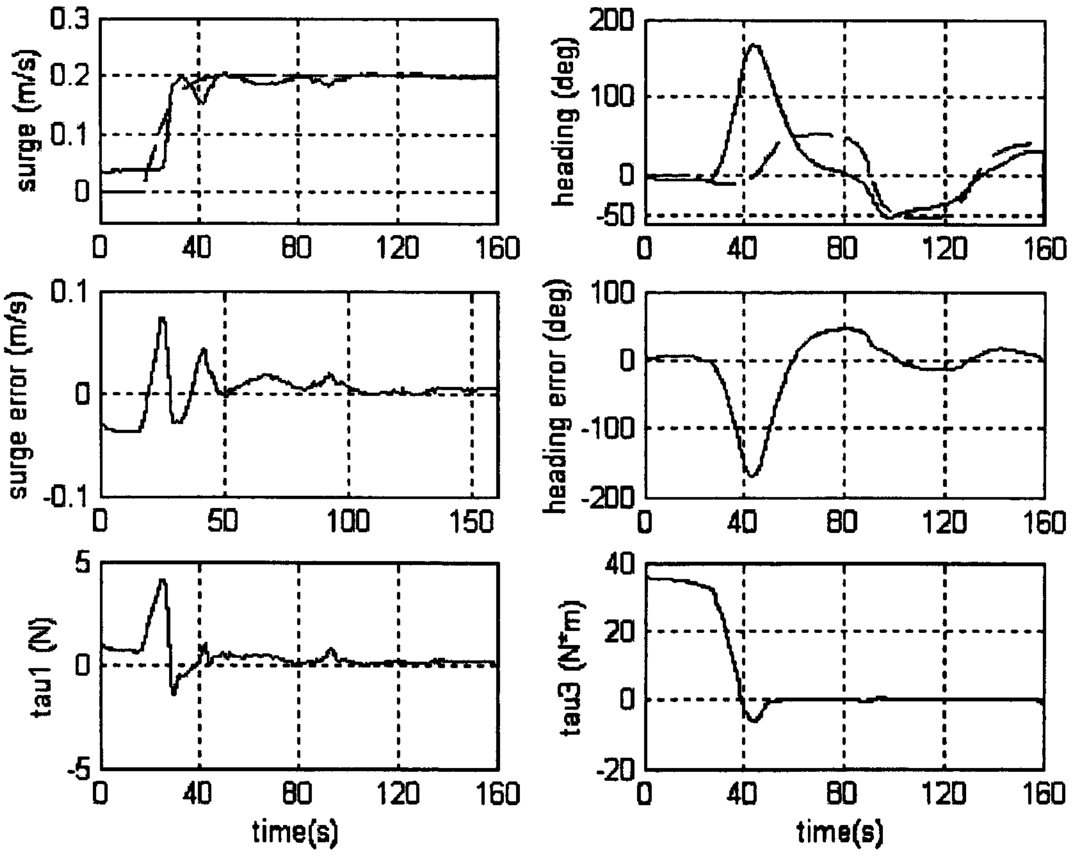


Figure 6.29: Real Results of the H_∞ Controller GA Optimised With Waves When Manoeuvring in Calm Waters

Once more the responses obtained with the H_∞ controller are very poor. In this case the heading performance is especially bad with an initial overshooting that makes the boat spin before starting to follow the desired response. Figure F.8 in Appendix F shows the simulated responses to be expected when tracking the zig-zag manoeuvre. The discrepancies are very significant.

Figure 6.30 illustrates the effect that has on the system the inclusion of waves while performing the manoeuvre. The response is so poor that adding waves does not degrade it significantly.

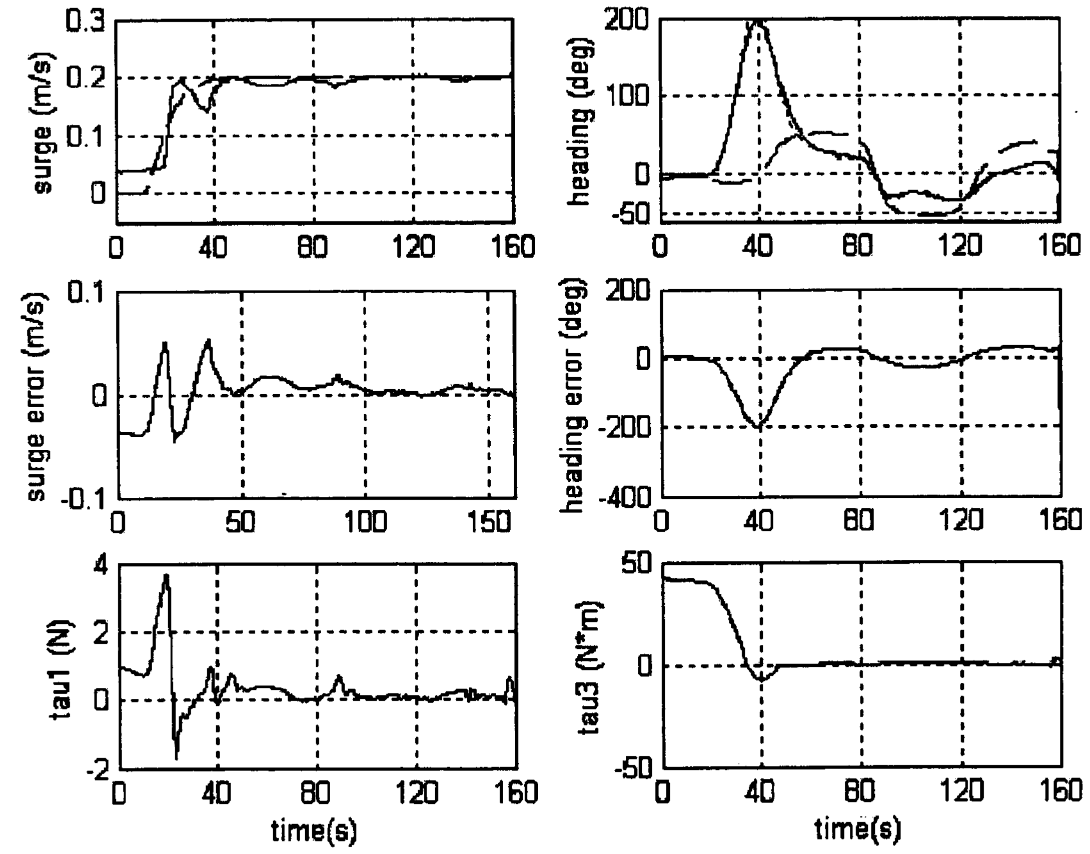


Figure 6.30: Real Results of the H_∞ Controller GA Optimised With Waves When Manoeuvring in the Presence of Waves

6.6.3. sGA STRUCTURAL OPTIMISATION

6.6.3.1. Optimisation Without Waves: Simulated and Real Results

Once run 15 times, the GA that provided the best result has converged in 24 generations (see Figure C.11 in Appendix C) to a cost value of 1.6:

Table 6.29: Best H_∞ Results - sGA Optimisation Without Waves

K_u'	α_u	β_u	γ_u	K_r'	α_r	β_r	γ_r
450	0.09	0.02749	0.09789	0.969	0.9972	0.00002	0.07061

K_{t1}	K_{t2}	K_{t3}	w_u	w_r
0.01701	0.557	0.03378	1	3

The values obtained for the control genes mean that for the surge speed error the weighting function is a first-order transfer function with a gain equal to K_u' and a pole at $-\beta_u$ (the values α_u and γ_u are deactivated), while the weighting function for the yaw rate error is a second-order transfer function (with a gain of K_r' and poles at $-\beta_r$ and $-\gamma_r$) with a zero at $-\alpha_r$.

After averaging the parameters obtained through the 15 runs the following values are obtained:

Table 6.30: Average and Standard Deviation H_∞ Results
sGA Optimisation Without Waves

K_u'		α_u		β_u		γ_u	
Avg	StDev	Avg	StDev	Avg	StDev	Avg	StDev
55.8139	1174.19	1174.19	2513.53	0.03875	0.05767	1472	3295.18

K_r'		α_r		β_r		γ_r	
Avg	StDev	Avg	StDev	Avg	StDev	Avg	StDev
1.25536	2.0921	632.75	2251.98	46.7332	174.753	726.782	1769.78

K_{t1}		K_{t2}		K_{t3}	
Avg	StDev	Avg	StDev	Avg	StDev
0.01715	0.02737	846.856	2296.759	0.029275	0.0434

The averaged results shown in Table 6.29 are not very significant since the control genes converged to different values, resulting in different weighting structures. Some of these parameters are deactivated in the controller and therefore might have a totally irrelevant value. The general tendency was to have a first order transfer function as the weighting for the surge speed error ($W_u(s)$), i.e. $w_u = 1$, and a second order with one zero weighting function for the yaw rate error ($W_r(s)$), i.e. $w_r = 3$.

Next Figure 6.31 shows the simulated results obtained with the controller from Table 6.29 while tracking a simultaneous double step.

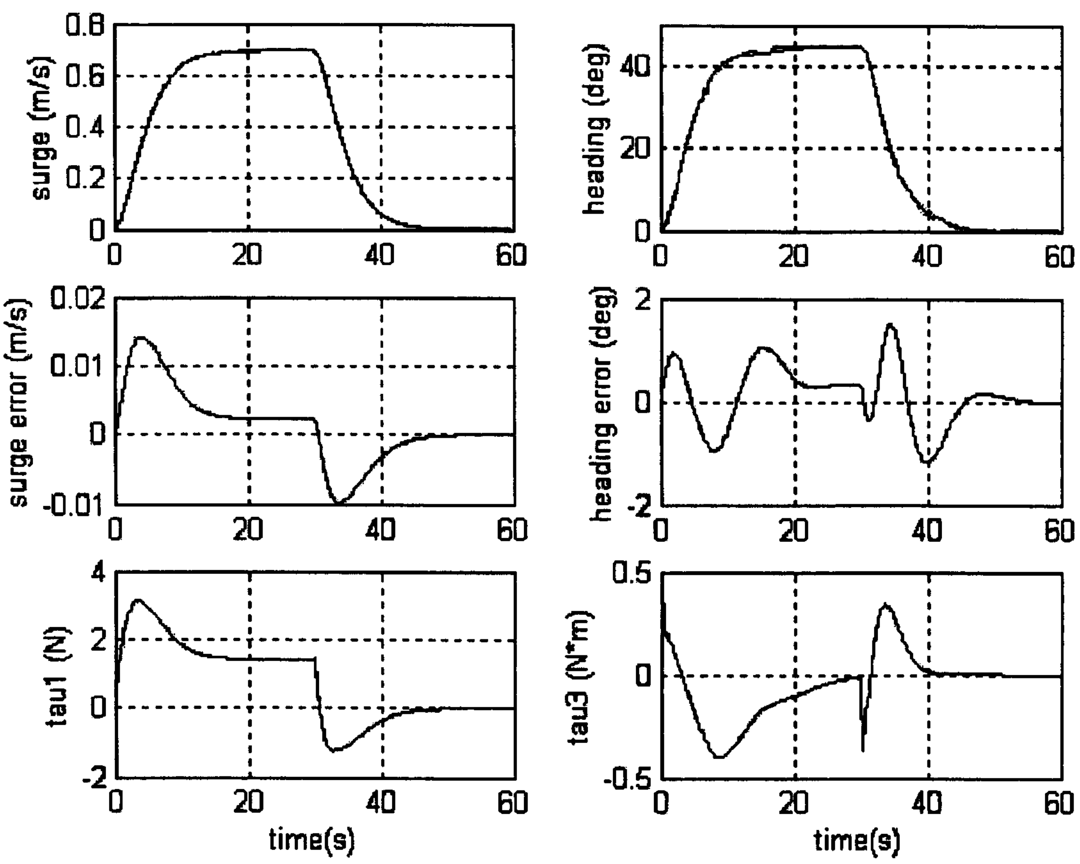


Figure 6.31: Simulated Results of the H_∞ Controller sGA Optimised Without Waves

If comparing the results obtained with GA and sGA when optimising the H_∞ controller just by visual inspection of Figures 6.25 and 6.31, they are practically the same. Taking into account the cost function of the solutions, sGA provided a equivalent cost value (1.5653 versus 1.5807) using a more simple weighting structure.

Figures 6.32 and 6.33 show the responses obtained when the H_∞ controller sGA optimised without waves is implemented in the real plant. Figure 6.32 illustrates the performance of the controller when manoeuvring in calm waters.

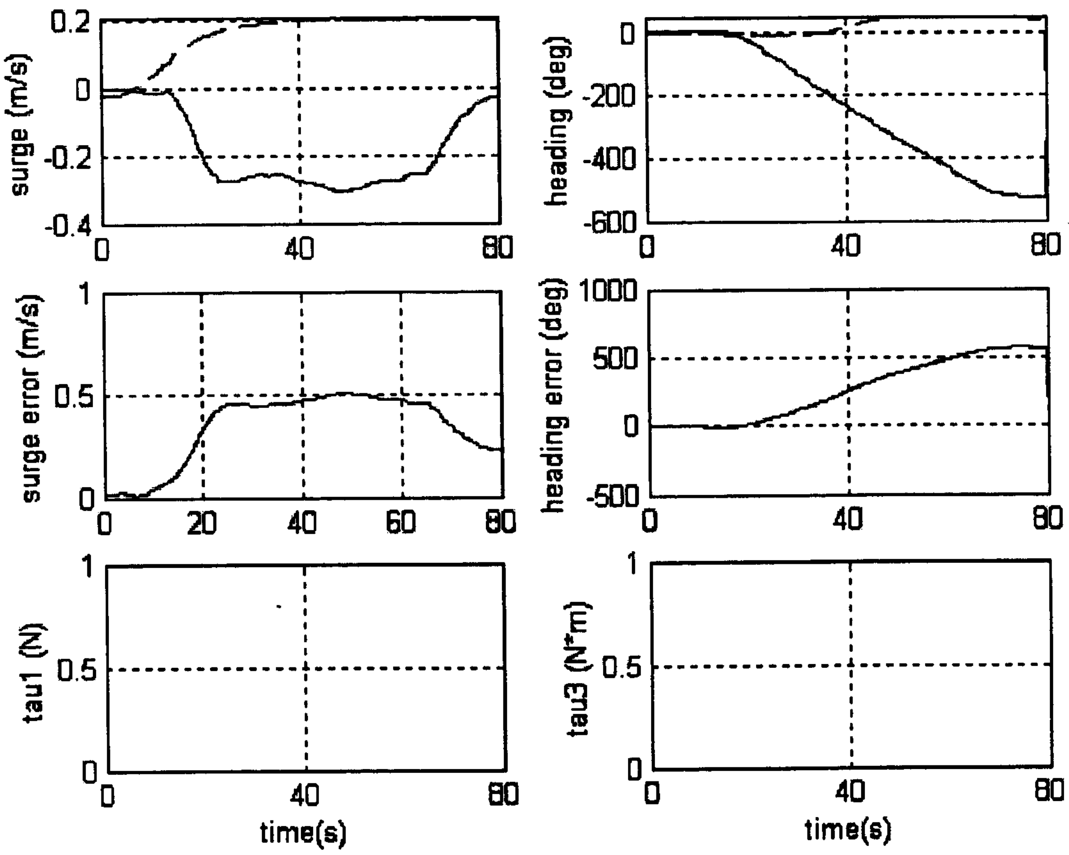


Figure 6.32: Real Results of the H_∞ Controller sGA Optimised Without Waves When Manoeuvring in Calm Waters

As it can be seen in the picture the system goes unstable when the H_∞ controller optimised using sGA is implemented. The commanded forces go to infinity and therefore they are not plotted in the figure. This has been also the case when the manually tuned H_∞ controller was implemented. The manoeuvre has been stopped after 80s to avoid damages to the ship. The mathematical model did not provide satisfactory results in this case since the simulations of the double step manoeuvre (Figure 6.31) and the zig-zag manoeuvre (Figure F.9 in Appendix F) did not reflect the stability problem experienced when using this methodology.

Figure 6.33 present the results obtained when the H_∞ controller was tested in the presence of waves. As expected the results shown in Figure 6.33 further confirm the instability of the controller.

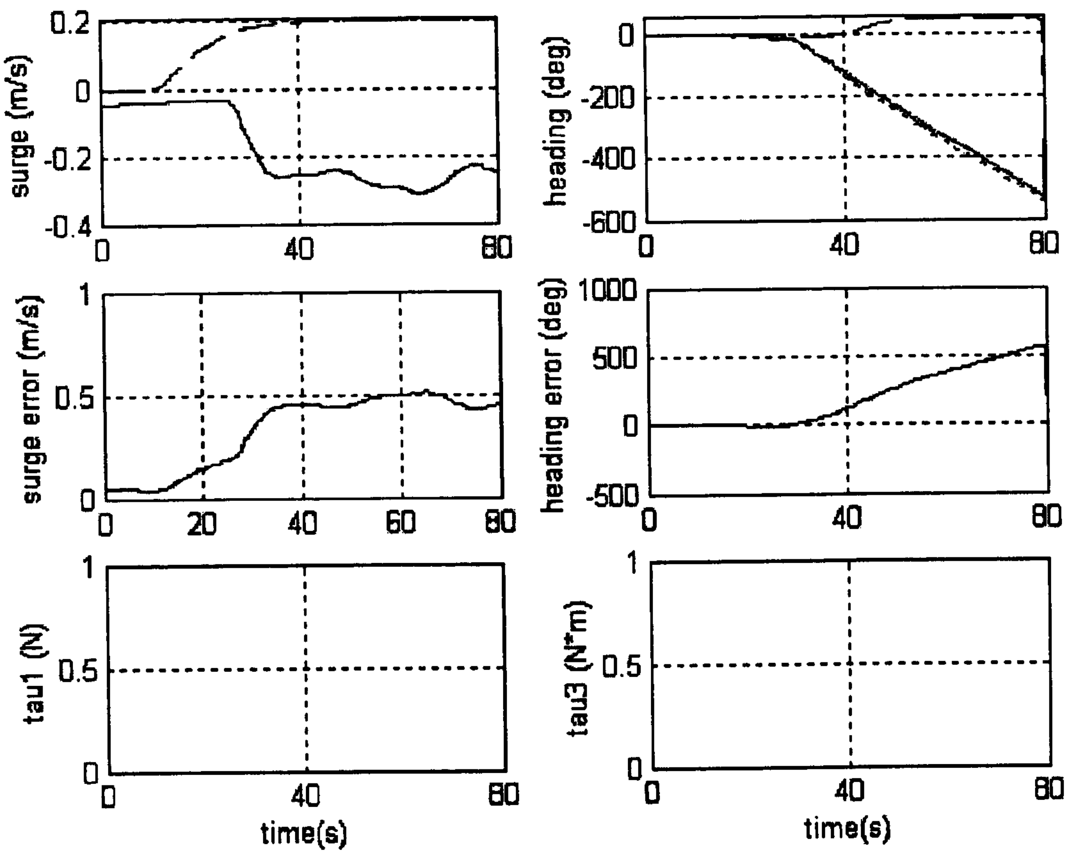


Figure 6.33: Real Results of the H_∞ Controller sGA Optimised Without Waves When Manoeuvring in the Presence of Waves

One of the reasons for this poor stability performance of the H_∞ controller could be related to numerical factors of the stability conditions. One of the conditions that has to be met so that the H_∞ controller is internally stabilising is that the matrices X_2 and Y_2 (solutions to the AREs defined by the Hamiltonian matrices H_∞ and J_∞) are invertible (see Equations (4.40), (4.44) and (4.45)). By definition a matrix is non-invertible if its determinant is zero. This creates a computational problem: how small a number has to be to be considered zero? It has been found out that the value of that tolerance is key to the H_∞ design, in the sense that variations in those numerical values lead to completely different results. Simulations have been run varying such tolerance in the range from 10^{-3} to 10^{-20}. The results show that making the tolerance bigger (i.e. imposing a more strict stability condition) degrades controller performance (i.e. the resulting cost values are higher). On the other hand, making the tolerance smaller improves the resulting cost values but it gives stability problems once the controllers are implemented in the real plant.

6.6.3.2. Optimisation With Waves: Simulated and Real Results

The best result of all runs of the GA with waves provides a cost value of 11.7 with the H_∞ gains shown in Table 6.31. The GA has converged in 41 generations as it can be observed in Figure C.12 in Appendix C.

Table 6.31: Best H_∞ Results - sGA Optimisation With Waves

K_u'	α_u	β_u	γ_u	K_r'	α_r	β_r	γ_r
0.09952	3009	0.007	26.6	0.09431	3.67	0.0308	6560

K_{t1}	K_{t2}	K_{t3}	w_u	w_r
0.00398	5509	0.0402	1	2

Since the control genes are set to 1 for propulsion and 2 for heading, the weighting function for the surge error is a first-order transfer function (gain equal to K_u' and pole at $-\beta_u$). The weighting function for the heading is a first-order transfer function (gain equal to K_r' and pole at $-\beta_r$) with one zero at $-\alpha_r$. The values γ_u, α_u and γ_r are deactivated.

The sGA has converged to solution that provides a smaller cost value that the one obtained with GA (11.7 versus 33.4) with a lower order controller. The inclusion of waves imposes bigger changes in the propulsion subsystem than in the navigation.

After averaging the parameters obtained the values in Table 6.32 are obtained:

Table 6.32: Average and Standard Deviation H_∞ Results
sGA Optimisation With Waves

K_u'		α_u		β_u		γ_u	
Avg	StDev	Avg	StDev	Avg	StDev	Avg	StDev
89.3352	183.918	1198.12	2403.58	0.05219	0.08321	3024.50	3730.71

K_r'		α_r		β_r		γ_r	
Avg	StDev	Avg	StDev	Avg	StDev	Avg	StDev
0.2166	0.2631	225.347	285.412	173.15	345.552	470.289	1631.24

K_{t1}		K_{t2}		K_{t3}	
Avg	StDev	Avg	StDev	Avg	StDev
0.018665	0.01995	1705.825	2731.34	0.04386	0.05934

The following Figure 6.34 presents the results obtained when simulating the solution from Table 6.31.

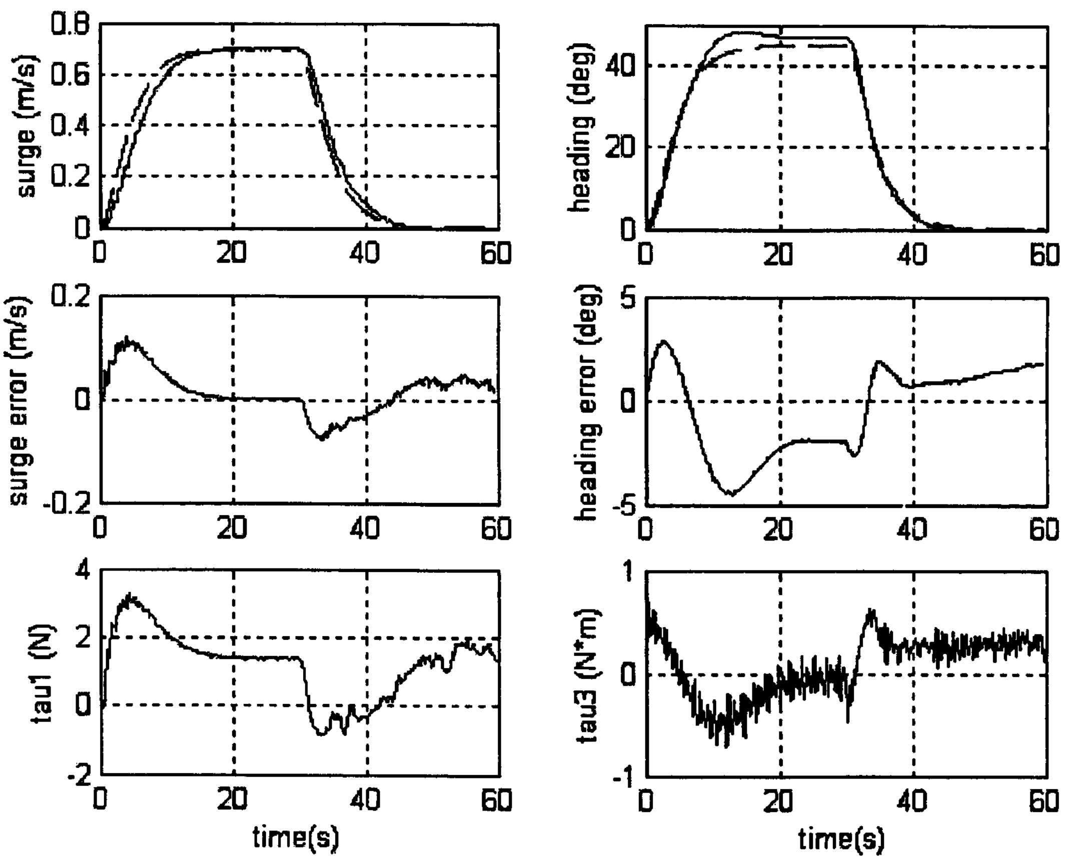


Figure 6.34: Simulated Results of the H_∞ Controller sGA Optimised With Waves

Comparing Figures 6.28 and 6.34 it can be concluded that the performance of the H_∞ solution obtained using sGA when disturbances are included in the optimisation is better than solution obtained with the standard GA in the optimisation with waves. The propulsion tracking is more accurate and the heading tracking less oscillatory.

Figure 6.35 and 6.36 show the responses obtained when the optimised H_∞ controller from Table 6.31 is implemented and the zig-zag manoeuvre performed in the real plant. Figure 6.35 shows the results obtained when the controller is tested in calm waters.

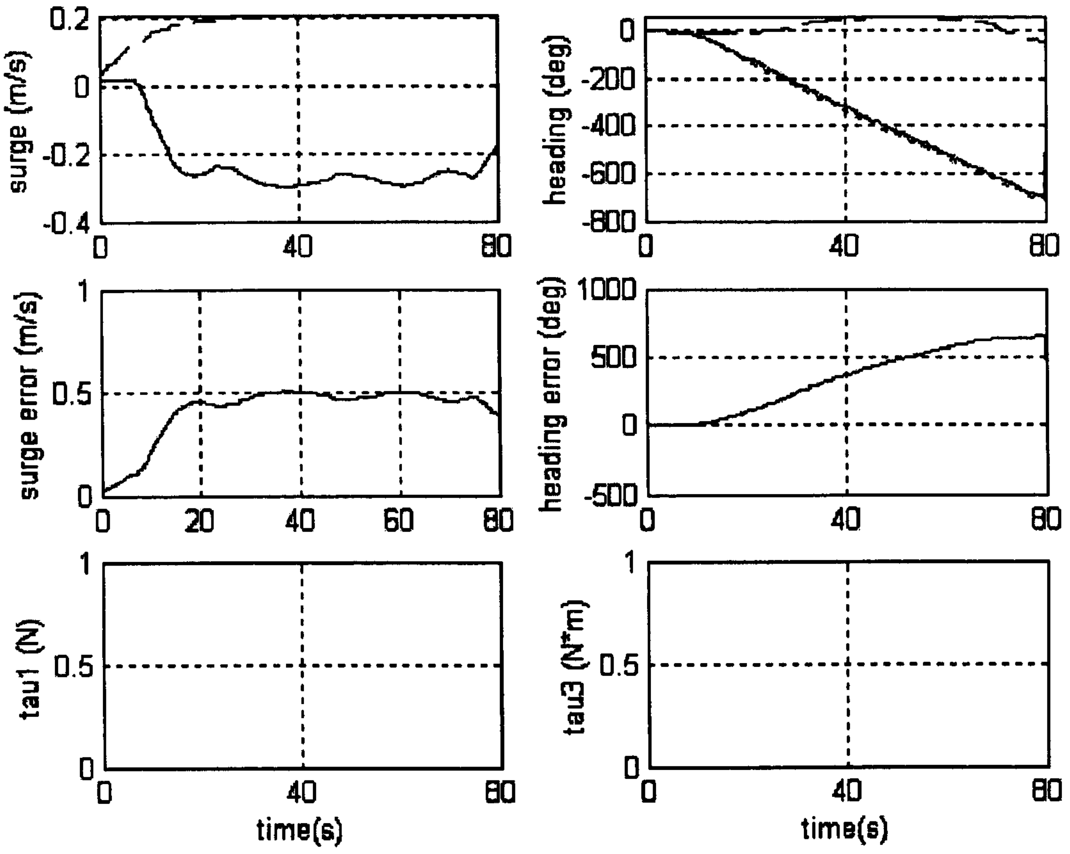


Figure 6.35: Real Results of the H_∞ Controller sGA Optimised With Waves When Manoeuvring in Calm Waters

The figure reflects the same stability problem that has already been encountered in the manually tuned H_∞ controller and in the H_∞ controller optimised without waves using sGA. The commanded forces go to infinity and the boat starts spinning. The manoeuvre had to be stopped after 80s in order not to damage the boat. As in previous cases the simulated results obtained when using the same references than in the real testing (see Figure F.10 in Appendix F) have not reflected the stability problem at all.

The responses shown in Figure 6.36 correspond to the testing of the sGA-optimised H_∞ Controller real implementation once waves are generated in the water tank. As expected the figure further confirms the instability problem already encountered in Figure 6.35.

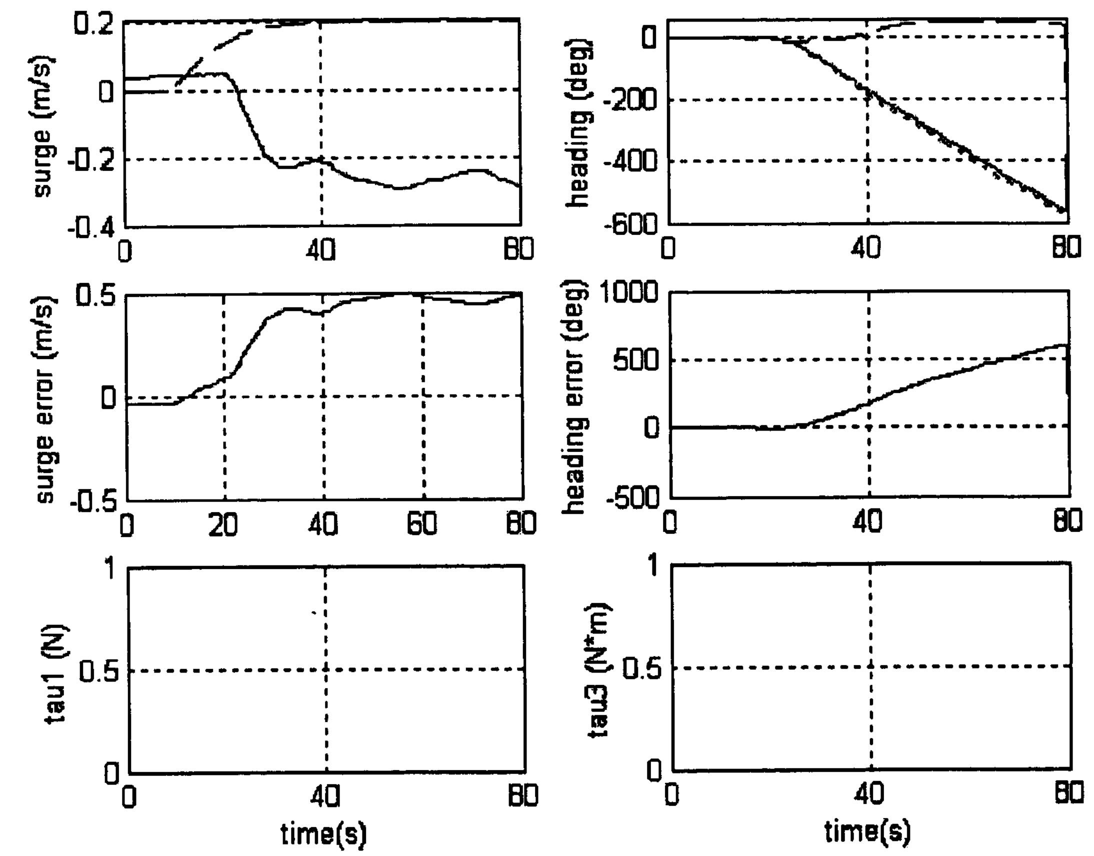


Figure 6.36: Real Results of the H_∞ Controller sGA Optimised With Waves When Manoeuvring in the Presence of Waves

6.6.4. SUMMARY OF H_∞ RESULTS

The main problem encountered in the optimisation of H_∞ has been the disappointing performance of the controllers once they have been implemented in the real plant. The reason could lie in numerical factors related to the stability conditions imposed while designing the controller. The simulation results are, though, quite good.

When comparing the simulated performance of the controller solutions obtained with GA versus those obtained with sGA, it seems quite clear the advantage of using sGA for this kind of optimisation problem since it provides better tracking with more simple weighting functions (i.e., lower order in the controller). However, once implemented in the real plant the controllers optimised using sGA perform worse than those optimised using GA, i.e. they become totally unstable.

Once more the results obtained in the optimisation with waves show a degradation of surge speed tracking.

6.7. SUMMARY

Table 6.33 summarises the cost values obtained in the optimisation with and without waves of the control procedures used in the chapter.

Table 6.33: Summary of Cost Values

	PID	PP	SM+PI	SM	H_∞ (GA)	H_∞ (sGA)
w/o waves	1.2	0.96	4.5	4.5	1.6	1.6
with waves	4.3	10.6	10.5	11.5	33.4	11.77

As it can be seen from the table the best results obtained in the simulation correspond to PID and Pole Placement controllers. However, once implemented in the real plant, the performance of the controllers varies significantly. The accuracy of the mathematical model raises doubts since the stability problems encountered in the H_∞ controllers and the PID and SM+PI optimised with waves controllers are not reflected at all in the simulations and, therefore, the optimisation process could not do anything to correct for these effects. This emphasises the importance of an accurate model. With the exception of these stability problems, all the controllers have performed well when implemented in the real plant without any need for further retuning. Tracking of heading was generally good and propulsion results were reasonably good. Overall, SM optimised without waves provided a very consistent good performance, despite of being outperformed by other controllers in the simulations as it can be seen from Table 6.33.

When the controllers obtained in the GA optimisation without waves are tested in the presence of waves the tracking performance does not degrade significantly, the main effect of the waves is the noise induced in the commanded forces (especially τ_l).

Regarding the optimisation carried out with waves, the objective was to ensure that the GA gives a more robust controller (i.e. an improved performance in presence of environmental disturbances). However, if comparing the performance of the controllers optimised with and without waves, it can be observed that those controllers optimised with waves do not provide a better performance but a reduction in control effort. They are also characterised by a very poor propulsion control (even leading to instability in two cases, SM+PI and PID). The GA concentrates on reducing the ripple in τ_l (the main effect of the waves) to improve the cost function value and that degrades the performance of the propulsion controllers. On the other hand, the navigation subsystem is less sensitive to the effect of waves. Consequently, the resulting heading controller parameters are less modified by the GA when introducing waves in the optimisation. This can be due to the sluggish propulsion dynamics. The controller tries to react to the noise but due to the slow dynamics, overcompensates and ends up producing all these oscillations.

As a result, the advantage of including noise in the simulation lies in obtaining controller solutions with smoother control signals that are able to reduce wearing and tear of actuators more than robust performance against external disturbances.

The proposed sGA model has performed well as a structure optimisation technique in simulation. It has provided better results than those obtained with GA while reducing the order of the weighting functions and, as a result, the order of the controller. The sGA results prove that the method can be of assistance when identifying appropriate weighting functions orders. However, the real performance of the sGA-optimised controllers has been very poor.

CHAPTER 7

GENETIC PROGRAMMING: AN IMPLEMENTATION IN MATLAB

7.1. INTRODUCTION

The standard GA conditions the search space before the optimisation starts and it is an artificial representation of the solutions because it involves a fixed-length string representation. The operators of GAs defined by Holland (1975) and DeJong (1975) have the advantage of neatness and easy implementation, but parsimony and tidiness are human concepts, nature's approaches to evolution are untidy and disorganised. In the GA community there have been various attempts of providing more flexibility to the GA by changing the representation of solutions and allowing a dynamical evolution of the size and shape of solutions. Researchers have looked for more flexible encodings for the representation of solutions such as *Messy Genetic Algorithms* (mGA) [Goldberg *et al.* (1989)]. The individuals in mGA are encoded as variable length strings and the genetic operators are altered according to this representation. The *Structured Genetic Algorithm* (sGA) from Section 5.3 is another example of variation in the representation of the solutions. Although the length of the chromosome is fixed, the control genes can activate and deactivate sets of genes so that the effective length of the chromosome varies among the individuals and along the generations. However, the most recognised of all these attempts to give more flexibility to the representation of solutions in genetic optimisation is Koza's (1992) creation, *Genetic Programming*.

When introducing *Genetic Programming (GP)* in 1992, Koza recalled that the main point in his work was to prove that: "genetic programming provides a way to search the space of possible computer programs for an individual computer program that is highly fit to solve a wide variety of problems from many different fields."

Computer programs are complex and hierarchical structures created by man that have proven to be able to solve a large number of problems in many different fields. Koza (1992) uses the term to describe any hierarchical solution from a mathematical identity to a game strategy. In this context a tool is sought that allows the solution of problems without specifying *a priori* the size, shape and structure of such a solution, i.e. the issue is to find an instrument for searching the space of possible computer programs to find a near optimal one.

Hence, GP is an evolutionary search technique based in the idea that, in nature, structures suffer adaptation to the environment. The structure created over a period of time is the outcome of natural selection and sexual reproduction.

In the control problem being tackled here the optimised computer program will provide a control strategy. Using the states of the system to control as an input, the outcome of the program will be the actuator's commanded signal. Therefore, the optimisation problem of finding a near-optimal controller is taken a step forward. By using GP the structure of the whole controller will be optimised.

Two of the main flaws encountered by GP researchers have been the *code bloat* (or tendency of GP solutions to grow quickly in size) [Koza (1992), Luke (2000c)] and how to generate the numerical constants necessary for the solution of most GP problems [Koza (1992), Esparcia-Alcazar (1998), Fernandez and Evett (1998)].

The code bloat problem has received lots of attention from the GP community [Angeline (1994), Koza (1992), Langdon and Poli (1998), Luke (2000a), (2000c), Nordin *et al.* (1996), Soule and Foster (1998), Tackett (1994)]. It is a serious problem that causes waste of computer resources and slows down the optimisation. This chapter includes a review of the most common theories about code growth and the techniques to control it.

In order to deal with the issue of the generation of numerical constants, two kinds of GP algorithms have been implemented. The first one chooses the constants necessary to create the controller structure by random generation (GP+RG) [Koza (1992)]. The second GP algorithm includes a GAs technique for the optimisation of such constants (GP+GA) [Howard and D'Angelo (1995)].

Chapter 7 includes a description of GP from a theoretical point of view together with a description of how the two GP algorithms (i.e. GP+RG and GP+GA) have been implemented in Matlab for the solution of the control problem of the heading and propulsion dynamics of CyberShip II. The results obtained with the GP optimisation are shown in Chapter 8.

The outline of Chapter 7 is as follows. Section 7.2 describes GP general structure, the representation of individuals and GP main operators. Section 7.3 presents the implementation in Matlab of the GP for the CS2 control problem. It illustrates both approaches for the creation of numerical constants: random generation and GAs optimisation. Finally, Section 7.4 summarises the chapter.

7.2. GENETIC PROGRAMMING OPTIMISATION

7.2.1. GP TREE REPRESENTATION

The main difference between GAs and GP rests in the individuals being evolved. With classic GAs the individuals that are evolved are represented by one-dimensional strings. Since the aim of GP is to search the space of computer programs, the individuals being evolved are hierarchical structures with no predefined size or shape [Koza (1992)].

Prior to the optimisation process the user has to define which functions and terminals are relevant for the problem to solve. This choice defines the search space for the problem in question. In order to obtain a solution for the problem the GP requires that all the functions and terminals needed to express the solution are included in the set of terminals and the set of functions. This is called the *sufficiency property* [Koza (1992)]: the set of terminals and the set of functions have to comprehend the terminals and functions necessary to express a solution to the problem. However, the search space size grows exponentially with the number of terminals and functions. Therefore the addition of numerous extraneous functions and terminals, in general, degrades the performance. Thus, the user must be very careful in the selection of the elements in the terminal and function sets to balance the trade-off between sufficiency and performance degradation.

Traditionally, the structure being evolved in GP has a tree shape [Koza (1992)]. This tree is formed by internal and external nodes. Once a set of functions and a set of terminals are defined, the internal nodes of the tree are occupied by functions while the terminals take the external nodes. This tree structure is a hierarchic structure that replaces the role of the parenthesis in normal algebraic notation.

For instance, the expression $y\ (x+2)$, would be represented as shown in Figure 7.1, where the set of functions would be $F=\{+, *\}$ and the set of terminals $T=\{x, y, 2\}$.

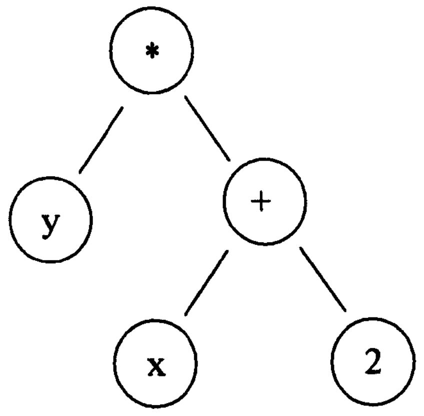


Figure 7.1: Example of Tree Representation

Another important issue to take into account when choosing the functions and terminals is the *closure property* [Koza (1992)]. The closure property says that each function has to be able to take as an argument any value and data type returned from the evaluation of any function or from the terminal set. For example, the division function has to be implemented in a way that can take a zero as a denominator without returning an error. Usually this is done by setting the division operator to return a fixed 0 or 1 value when the denominator is zero.

7.2.2. GP STRUCTURE

The flow chart of GP is identical to that of GAs. Initially, a population of trees is created at random. Each individual in the population is evaluated using a cost function. The individuals that performed better in the evaluation process have more possibilities of being selected for the new population than the rest. The individuals of this new population typically show better performance than those of the previous one, since the best individuals have a better chance of being selected for reproduction. Once the new population is created, the individuals are subject to the genetic operators of crossover and mutation with a certain probability. The loop is run until a certain termination criterion is met [Holland (1975), Koza(1992)].

The following Figure 7.2 shows the typical flow chart of a GP:

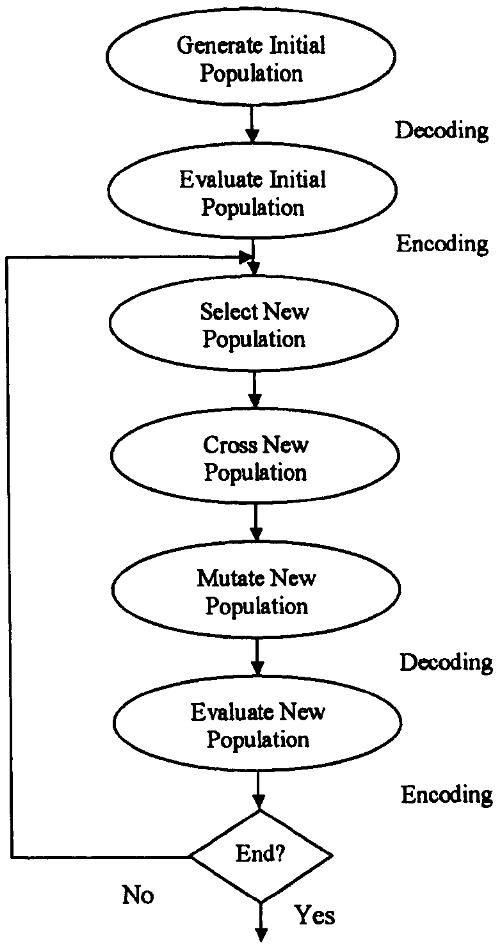


Figure 7.2: Flow Chart of a GP

7.2.3. GP INITIALISATION

The initialisation process is more complex than for GAs due to the increased complexity of the individuals' representation.

In his pioneering work Koza (1992) presented three methods of initialising the population: *full method, grow method* and *ramped half-half.*

The *full method* involves creating trees with a fixed length between the root and every terminal node.

The *grow method* involves creating trees with a specified maximum length between the root and every terminal node. Thus, the shapes of the trees in the population show a bigger diversity.

The *ramped half-half method* creates an equal number of trees for every length value from 2 until the maximum. For each length value half of them are created using the full method and the other half using the grow method. Koza (1992) used the ramped half-half in his experiments because of the wide variety of shapes it creates.

Luke (2000b) argues that the classical initialisation algorithm (used as well for the randomly created subtrees in the mutation operator) leads to very large tree sizes. Unless very careful choice is made of the probability of selection of an internal node over a terminal node and the number of arguments the functions require, the expected size of a tree tends to infinity. Luke (2000b) presented two new creation algorithms for GP that allow a better control of the tree size. The advantages of these algorithms over the generally used grow method are: user control over the tree size; user-defined probabilities of occurrence of functions and terminals in the trees; and very low computational complexity. From my point of view the main criticism of this method is that when the user is given so much choice the search is biased towards the direction the user is pointing, which might not be the right direction. Again there is a trade-off between flexibility and domain knowledge.

7.2.4. GP SELECTION

As with any evolutionary technique, the selection operator is used to determine which individuals are going to pass to the next generation. GP algorithm evaluates every solution tree and assigns a cost according to its performance. The selection procedure favours the selection of those individuals that obtained a better cost.

Selection methods used in the GP literature are equivalent to those of GAs. The most popular methods are roulette-wheel selection (used by Koza (1992) in his examples) and tournament selection [Chellapilla (1997), Luke (2000a)].

Tackett (1994) makes a comparison study of selection methods in Chapter 4 of his thesis. He studies the effect of the selection operator in the context of noisy and under-sampled data. Using tournament selection as a basis, he examines the differences in performance due to distributed selection and steady-state selection. The final conclusion of this study is that the differences in performance between the selection mechanisms studied are minor.

7.2.5. GP CROSSOVER

The most popular form of crossover is what has been called *subtree crossover* [Koza (1992)]. Two individuals are selected from the population. A crossover node is chosen at random in each parent and the whole subtrees rooted at those nodes are swapped, as shown in Figure 7.3. Typically, the offspring generated will have different shapes and sizes than their parents. As opposed to GAs, the two offspring created by crossing two identical parents usually are different, unless both crossover points are the same. This feature of GP provides a bigger diversity in the population.

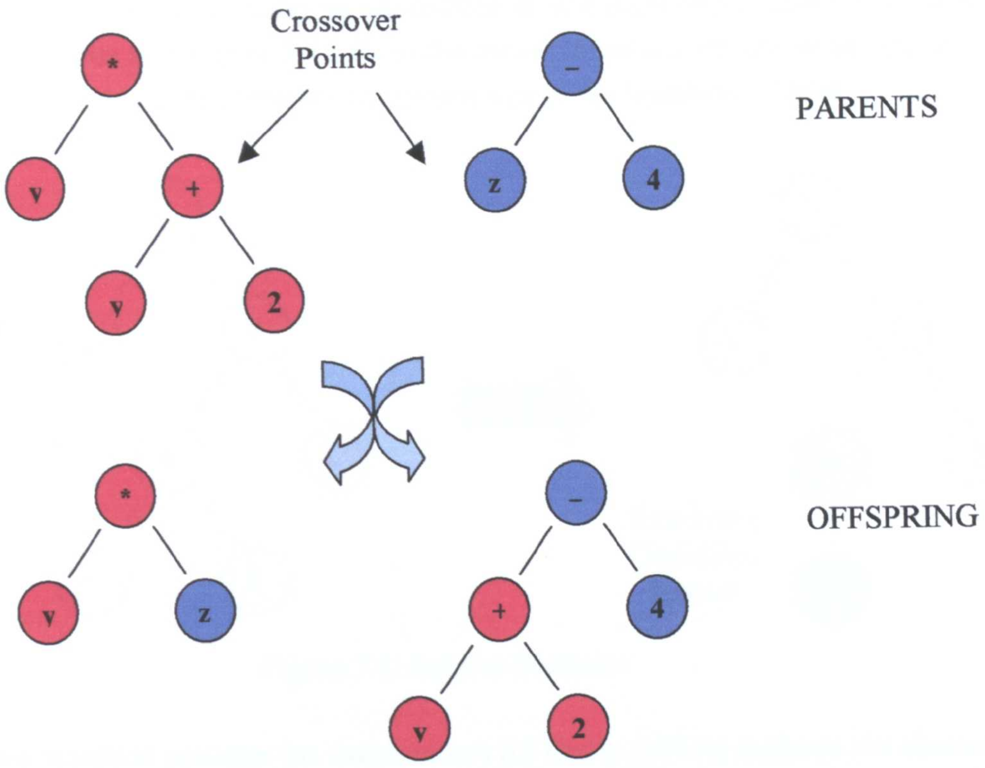


Figure 7.3: Subtree Crossover

Other variants of the subtree crossover include a *non-leaf selection bias* [Koza (1992)] or the *greedy recombination operator* [Tackett (1994)]. The non-leaf selection bias ensures that once a tree has been selected for crossover, the probability of a terminal node being chosen as a crossover point is 10%, as opposed to a 90% probability of choosing an internal node as the crossover point.

The novelty of the greedy recombination operator is that every couple of parents produces *n* pairs of offspring, instead of a single pair. Then all the individuals in the offspring are evaluated and only the pair that obtained the best cost of all is chosen as offspring while the rest are rejected. To reduce computational cost the evaluation process used to evaluate the offspring is normally shorter than the one applied to the population. However, for many control problems evaluation just takes too long to implement this kind of operator.

Typically, the probability of crossover is between 80-95% [Koza (1992)].

7.2.6. GP MUTATION

The tree structure of GP solutions allows a variety of mutation operators.

The standard mutation operator is *subtree mutation* [Koza (1992)]. It selects a mutation point at random, removes the subtree at that point and inserts a randomly generated subtree (see Figure 7.4). As in the crossover case a maximum permissible depth is chosen that determines the maximum size of the mutation subtree.

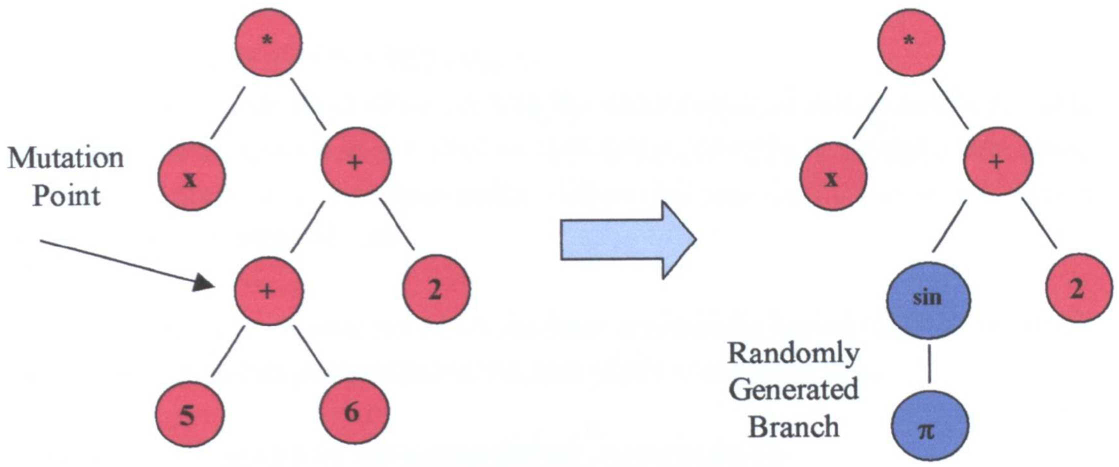


Figure 7.4: Subtree Mutation

The *point mutation* operator (or *substitution*) [O'Reilly (1995)] replaces the chosen node with another function of the function set with the same number of arguments if the chosen node is an internal node or with a terminal of the terminal set otherwise (see Figure 7.5).

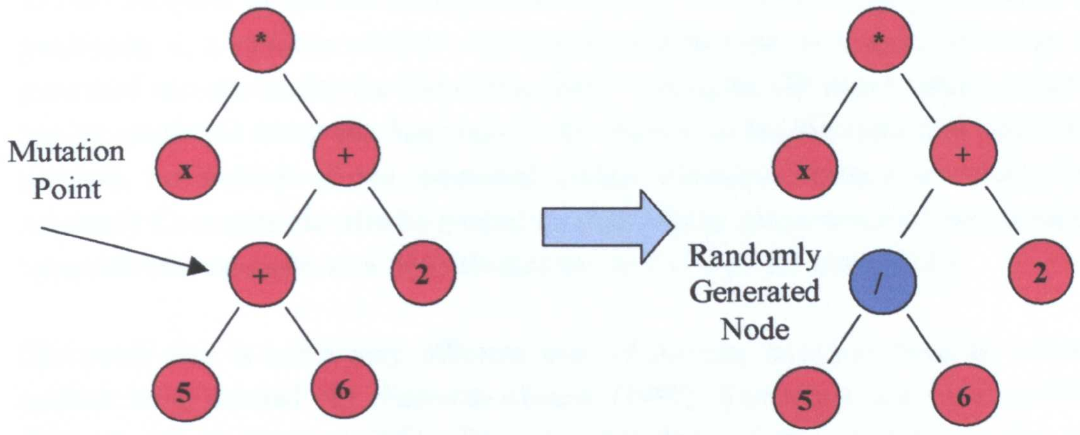


Figure 7.5: Point Mutation

Other mutation operators are those presented by Chellapilla (1997) and Angeline (1997). They consist of several suboperations applied sequentially. These suboperations are variations of the subtree mutation and the point mutation. Some of these variations (under different names) include: replace the tree whose root is the chosen node by its largest subtree; or replace the chosen node with another function or terminal of the function and terminal sets using the tree whose root is the chosen node to act as the first argument; or swap the order of two arguments of a node randomly chosen; or perturb those terminal that are a numerical constant.

7.2.7. GP TERMINATION CRITERION

Mainly there are two ways of terminating the optimisation. If the optimal cost value is known, the simulation is run until an individual succeeds in reaching that value, but that requires *a priori* knowledge. Otherwise the simulation is run for a predetermined number of runs.

Once the simulation is over the result obtained is either the best so far if a record has been kept of previous generations or the best of the final generation.

7.2.8. GP GENERATION OF NUMERICAL CONSTANTS

The choice of the numerical constants necessary in the solution of the problem tackled with GP has generated much discussion [Koza (1992), Esparcia-Alcazar (1998), Fernandez and Evett (1998), Howard and D'Angelo (1995)].

Koza (1992) proposed the inclusion in the terminal set of what he called an ephemeral random constant: "whenever the ephemeral random constant R is chosen for any endpoint of the tree during the creation of the initial random population in generation 0, a random number of a specified data type in a specified range is generated and attached to the tree at that point." Along the GP search these constants can be combined using the functions in the internal nodes to create new ones (for example, the addition of two ephemeral random constants creates a new numerical constant). Constants can also be created by arithmetical combination of non-numeric terminals (for example, a variable divided by itself creates the constant 1).

Obviously this is not a very efficient way of creating new constants as various authors have pointed out [Esparcia-Alcazar (1998), Fernandez and Evett (1998), Howard and D'Angelo (1995)]. The main drawback of this approach is that the number of constants depends totally on the initialisation of the trees. Since Koza (1992) considered a zero probability of mutation, numerical constants can be eliminated but not created. Also, the initial value these constants take cannot be altered along the optimisation by any operator.

Various other approaches can be found in the literature that address the problem. Fernandez and Evett (1998) use a variation of Koza's random generation. In order to avoid the problem of not being able to vary the numerical constants, the method they propose is a numeric mutation that is applied to part of the population. All the numeric terminals are altered by a value depending of the cost of the best individual in the population. The fitter the individual, the smaller the range of change.

In her thesis dissertation Esparcia-Alcazar (1998) presents the concept of *node gains*. A node gain is a numerical parameter that multiplies the output value of a node. Every tree is associated with a vector of parameters with as many elements as nodes are in the tree. This approach simplifies the problem of the placement of the numerical constants and their possible elimination from the tree. However, the computational cost and complexity of the representation is substantially increased. Also, it does not avoid the need for numerical constants in the terminal set for functions such as addition or subtraction (for example, for the identification of the function $f(x) = x + 4$, it is necessary to include a terminal with the value 4, there is no gain value multiplied by x that would provide a satisfactory result).

Other authors have included a secondary optimisation method for the tuning of the numeric parameters in the GP tree. For instance, Gray *et al.* (1998) included a combined Nelder-Simplex and Simulated Annealing method, and Bastian (2000) used a downhill simplex. Howard and D'Angelo (1995) combined a GP with a GA. The authors associate a GA-like fixed-length chromosome that represents the numerical values of the solution, although they may or may not be present in the tree. The fixed-length chromosome is evolved together with the tree and it is submitted to crossover with other chromosomes and mutation. The main problem that this approach has is that the fixed-length of the chromosome determines the maximum number of numerical constants that can be found in the tree. This requires *a priori* knowledge of the solution. Also, if the chromosome is made to be very long just to account for any additional constant that can be necessary, the length of the chromosome will hamper the correct evolution of solutions and it will also increase the computational cost.

7.2.9. SIZE PROBLEM IN GP

One of the main problems encountered by GP researchers is that of *code bloat* [Koza (1992), Luke (2000c)]. Luke (2000a) defines it as: "the tendency of candidate program solutions to grow in size independent of any corresponding increase in quality" (i.e. the size of trees grows out of control but many of the subtrees do not actually contribute to the final solution). This flaw causes waste of computer resources evaluating huge trees, difficulties in the understanding of the final solutions and spoils the convergence by hampering the modification of trees in a meaningful way.

A variety of approaches to tackle this problem appear in the literature, see Bhattacharya and Nath (2001) and Luke (2000c) for a review of the topic.

The most popular method to date for controlling bloat is the imposition of size limits proposed by Koza (1992): depth limitation for the generation of individuals in the initial population, depth limitation in the generation of subtrees for subtree mutation and restriction of the crossover and mutation operators so that children larger than the permissible size are not included in the population.

The second most common method consists of the inclusion of structural complexity in the cost function [Koza (1992), Shimooka and Fujimoto (1998)]. The problem is that when parsimony is included in the cost function, considerably more individuals must be process in order to find a good solution [Koza (1992)].

Finally, other authors either modify existing genetic operators or create new ones. Koza (1992) introduces an *editing* operator that periodically would simplify the trees, eliminating the subtrees that do not add anything to the final solution (for examples, subtrees that are multiplied by zero). He did not use it in his algorithm because the test did not show conclusively any benefit for its inclusion (drawbacks were not proved either). Luke (2000b) presents two methods for the initialisation of trees that allow control over the expected tree size (see Section 7.2.3). Ekart (2000) introduces a mutation operator that modifies the structure but does not alter the cost of a tree (i.e. it performs the algebraic simplification of the tree expression), in a similar way to Koza's editing operator. Van Belle and Ackley (2002) create a special operator called *uniform subtree mutation* to prevent bloating. It consists of the application of standard subtree mutation a binomially distributed number of times. This binomial distribution is based on the size of the tree. This mutation operator increases the probabilities of undergoing mutation of the nodes in larger trees.

Many of the theories explaining this bloat phenomenon are based on the concept of *introns*, areas of code that can be removed without altering the evaluation of the solution, i.e. they are redundant [Angeline (1994)]. The main theories linking code bloat with introns are: *hitchhiking, defence against crossover* and *removal bias.*

The *hitchhiking* theory is the brainchild of Tackett (1994). In this work the author proved that random selection in conjunction with random crossover does not cause code growth and therefore it is concluded that fitness is the cause of the increasing of the size. The preservation of important bits of code by the crossover operator leads to introns or hitchhikers attached to these important bits being propagated along the generations with them.

Defence against crossover theories [Nordin *et al.* (1996)] go a step further. They argue that the role played by the introns is that of increasing the number of nodes of the tree, making it more difficult to destroy with crossover.

Removal bias is similar to defence against crossover but it adds a bias towards removing small subtrees (smaller than the introns) for preservation purposes. Thus there is a tendency of removing small subtrees but there is not such a bias in the size of the replacing subtrees leading consequently to bigger trees [Soule and Foster (1998)].

The main code growth theory not based on introns is the *diffusion* theory [Langdon and Poli (1998)]. This theory relies on the idea that in the search space there are more big tree structures than small ones. Thus during the search process the GP will tend to find bigger trees.

Lately, Luke (2000c) has studied bloat quite extensively. In his paper he presents experimental evidence against the claim that it is the crossover between introns that causes the bloat problem, concluding that: "tree growth is the cause of inviable code growth", not the consequence. The mechanism he proposes to explain code growth is a generalisation of the removal bias theory. Usually there is an inverse relationship between the depth of a node and its influence on the cost value generated in the tree evaluation. Therefore, in very fit individuals there is a bias towards choosing deeper crossover points that will not damage the evaluation of the tree. Such a bias has two consequences, on one hand it can promote larger parents and, on the other hand, removed subtrees rooted in a deep node are more likely to be small, but the inserted subtrees have no such size bias.

7.3. GP IMPLEMENTATION IN MATLAB FOR A CONTROL PROBLEM

7.3.1. GP TREE REPRESENTATION IN MATLAB

GP algorithms were originally coded in LISP. Koza (1992) explains some of the advantages of using LISP language for GP (such as the common form for programs in LISP S-expressions, being equivalent to a tree structure), although he points out that GP is not inherent to LISP, it can be coded in any other language chosen. One of the most popular options has been C++ [Fraser (1994), Singleton (1994)]. C++ provides the programmer with pointers that facilitate the encoding of trees and, also, C++ is fast.

Ng (2000) includes an implementation in Matlab of GP. The author describes three different data structures, which, due to Matlab not accommodating pointers, represent the candidate trees as linear strings of functions. In *Data Structure I*, every function is represented by 4 string cells: two arguments, the function and the result of the evaluation. *Data Structure II* shortens the tree to 3 string cells (two arguments and the function) by using a temporary variable for storing the result during the evaluation process. *Data Structure III* lifts the structure one level. A matrix structure is used where every row is a branch of a tree.

The main disadvantage of all three structures is that the number of shapes that can be represented is very restricted. Due to the linear evaluation every internal node takes the previous node as an argument and a terminal node as the second argument. Thus, the resulting tree structure is as shown in Figure 7.6.

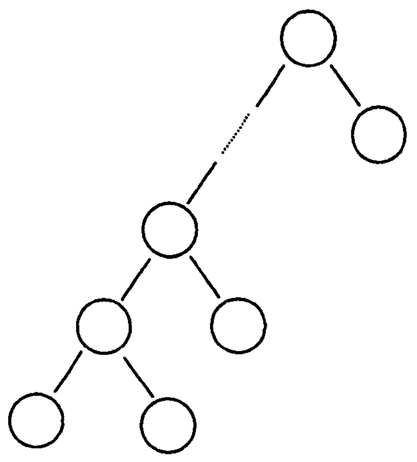


Figure 7.6: Ng (2000) Resulting Tree Structure

The structure used in this thesis differs from Ng's approach and allows non-linear tree structures like those proposed by Koza (1992).

The whole population is stored in a cell array, every cell storing one individual. The tree structure is represented by a matrix whose size is: number of internal nodes x 5. the classical GA vector structure is given a second dimension and becomes a matrix. In the GA case the length of the vector is fixed, while in GP the number of rows in the matrix represents the number of internal nodes and it is evolved along the GP generations. Every internal node is thus encoded as a 1x5 vector:

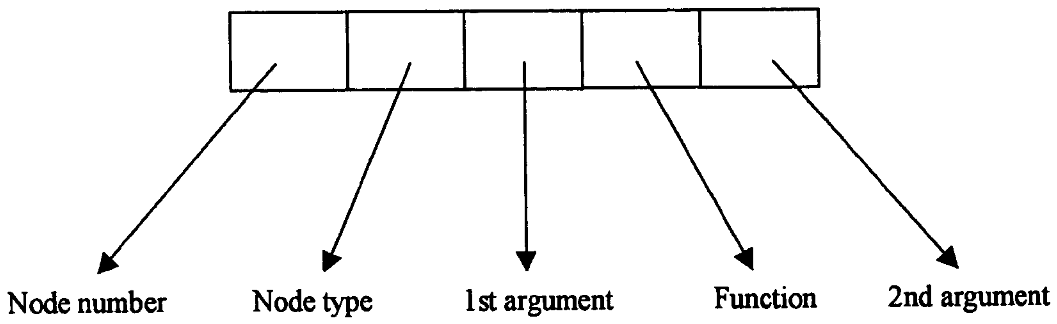
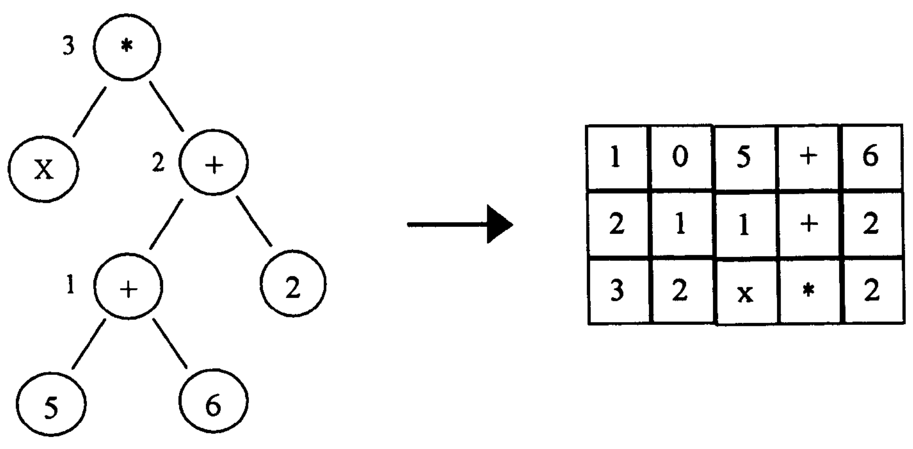


Figure 7.7: Internal Node Matlab Representation

As seen in Figure 7.7, the first element in the node vector is the node number. The tree structure needs to be flattened so every internal node is assigned a number. The nodes of the tree are counted from left to right, upwards. A node is not counted until all the nodes of the subtrees rooted in it are counted (see Figure 7.8). The second element in the Matlab vector representation defines the kind of node it is. Since Matlab does not provide pointers this element distinguishes if the arguments of the function are terminal nodes or internal nodes (for example, a value of 0 indicates that both arguments are terminal nodes and a value of 1/2 indicates that the 1st/2nd argument is an internal node). The three last elements provide the arguments of the functions (functions chosen have either 1 or 2 arguments) in columns 3 and 5 and the function itself in column 4. If an argument is a terminal, it is included in the correspondent position; otherwise the numeric value refers to the number of the internal node that is rooted there.

The main disadvantage of this representation is that subtree crossover and mutation in terminal nodes are quite complicated to implement.



1	0	5	+	6
2	1	1	+	2
3	2	x	*	2

Figure 7.8: Matrix Representation of a Tree

7.3.2. GP OPERATORS USED IN THE IMPLEMENTATION

This section summarises the initialisation, selection, crossover and mutation operators used in this GP implementation. See Appendix E for the code listings.

The initialisation method used here is the grow method [Koza (1992)]. It creates trees with a minimum of 5 nodes and a maximum of 15. The maximum depth of a tree (meaning by depth the number of levels of hierarchy) created in the initialisation process is 4. This insures a good diversity in the initial population. All the functions have the same probability of being chosen when creating a tree.

In order to choose the selection method as well as the crossover and mutation probabilities, a comparison study equivalent to that of GAs has been conducted. The same genetic schemes from Table 5.3 have been used, i.e. all the possible combinations of 3 selection methods: elitism, ranking and tournament selection, 3 crossover probabilities: 0, 0.6 and 0.8 and 2 mutation probabilities: 0.05 and 0.1. Each GP optimisation has been run 6 runs and the results have been averaged.

Best Cost Analysis

The following Figure 7.9 shows the averaged best results obtained. Only the results from ranking (solid line) and tournament selection (dotted line) are shown because the results obtained with elitism are well out of the Y-axis scale.

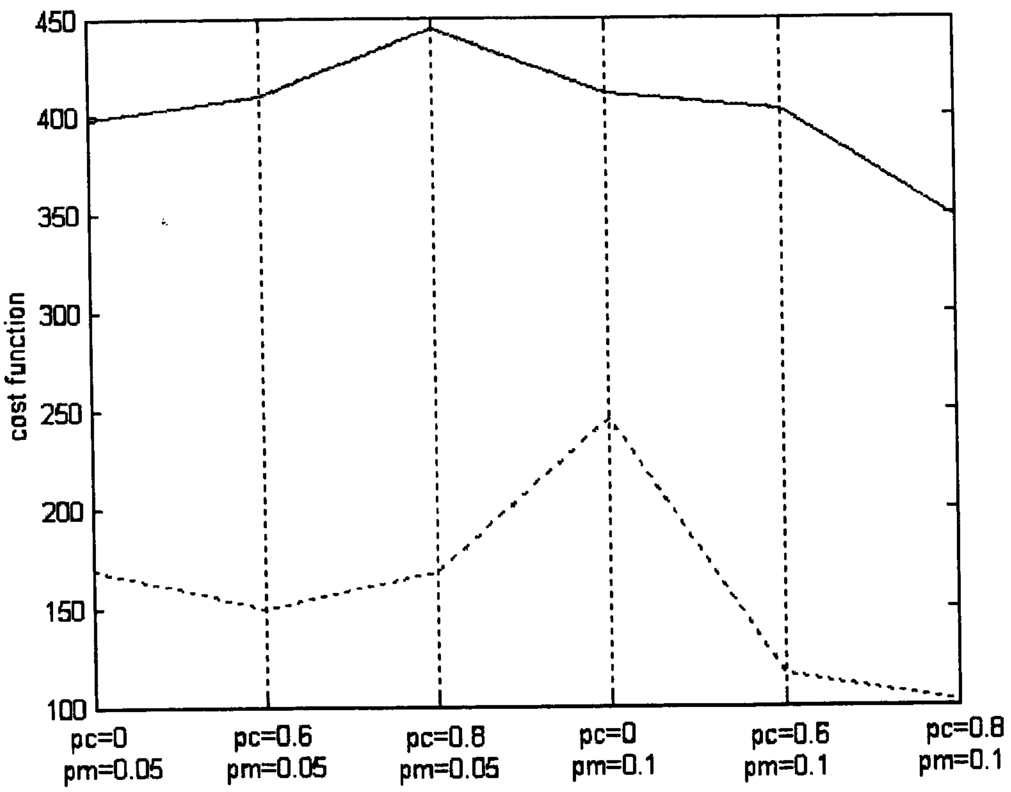


Figure 7.9: Best Cost Results for all GP Scheme
(dotted line: tournament selection, solid line: ranking selection)

It can be observed from the figure that tournament selection provided the best results for each combination of crossover and mutation probability. Also, the best overall result was obtained with tournament selection, a crossover probability of 0.8 and a mutation probability of 0.1. This result is consistent with the results obtained in the comparison studies of GA schemes presented in Chapter 5.

<u>Speed of Convergence Analysis</u>

In the following Figure 7.10 the generations needed for convergence (as defined in Section 5.4.2 and averaged over 6 runs) obtained for each GP scheme are plotted. As previously, the dotted line represents tournament selection; the solid line represents ranking selection and the dashed line, elitism.

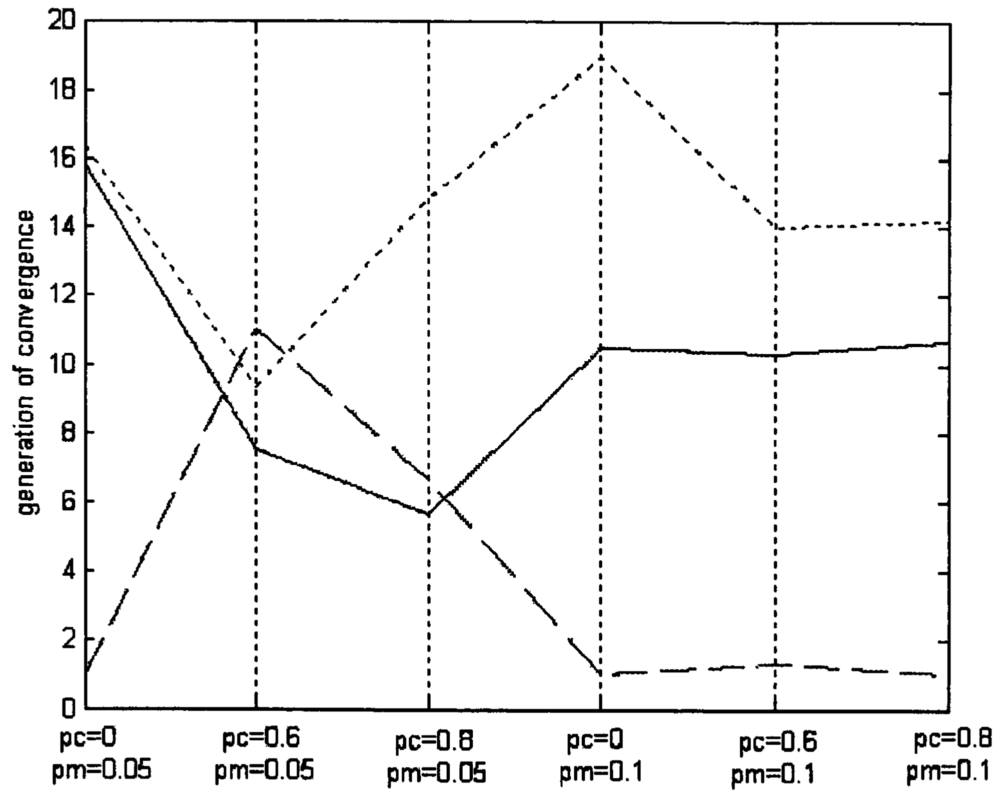


Figure 7.10: Generation of Convergence for all GP Scheme
(dotted line: tournament selection, solid line: ranking selection, dashed line: elitism)

The fastest convergence has been achieved by elitism, but it also converged to the worst results. For tournament and ranking selection increasing mutation generally slows down convergence and increasing crossover generally speeds up convergence, although the variations are not very significant.

Amount of Convergence in the Final Population Analysis

Finally, Figure 7.11 shows the amount of convergence of the final population (as defined in Section 5.4.2 and averaged over 6 runs) obtained for each scheme. Again the dotted line represents tournament selection; the solid line represents ranking selection and the dashed line, elitism.

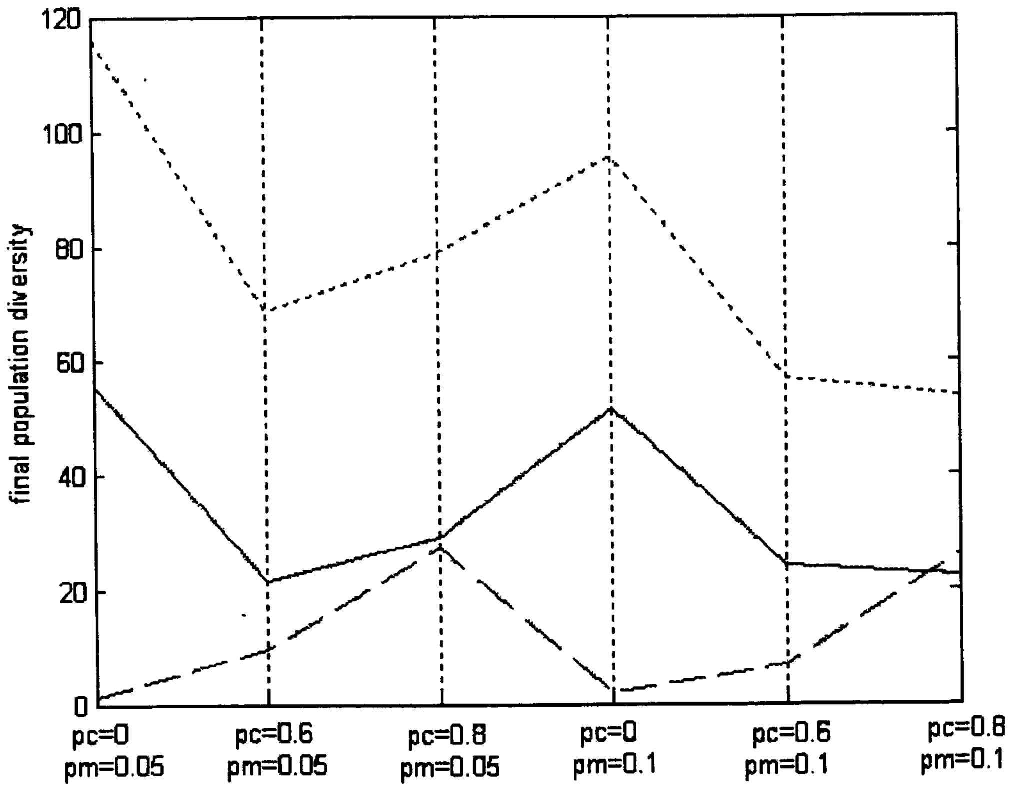


Figure 7.11: Amount of Convergence in the Final Population (dotted line: tournament selection, solid line: ranking selection, dashed line: elitism)

The selection method with a larger amount of convergence in the final population is tournament selection. This proves the saturation on higher echelons of the population with solutions with very similar cost values. Therefore there is not enough diversity for further improvement. On the other hand, elitism shows very small amount of convergence in the final population. This indicates that, although the results obtained with elitism schemes are quite poor, there is room for improvement.

Thus, given the results obtained in the comparison study, the operators used for this study have been tournament selection, crossover probability of $p_c = 0.8$ and probability of mutation of $p_m = 0.1$.

Two different mutation operators are included and they happen with a probability of 0.5 each. The first type of mutation operator implemented is subtree mutation [Koza (1992)]. In subtree mutation a node is chosen at random and whole subtree rooted in the node is replaced by a randomly created one (see Figure 7.4). The second mutation operator is point mutation [O'Reilly (1995)]. A node is chosen at random and replaced randomly, always respecting the kind of node it is, i.e., terminal node, 2-arguments function or 1-argument function (see Figure 7.5).

In order to avoid the problem of over-sized trees that are very difficult to analyse and waste computer resources, the subtree crossover and mutation operators include a limitation in size. Therefore, if the crossover or mutation operation implies that the size of the resulting tree is over the maximum size allowed (30 internal nodes), the operation does not take place. Also, they only take place in internal nodes due to the difficulties in the implementation in terminal nodes mentioned in Section 7.3.1. Point mutation has been included to counteract the effect of subtree crossover and mutation only affecting internal nodes.

It is important to consider that, given that evaluation of the individuals is the most time demanding part of a GP run, a considerable amount of computer time can be saved by not computing the cost for an individual that appears in the present generation as a result of selection (without crossover or mutation) from the previous generation (i.e. unchanged individuals).

7.3.3. GP APPLICATION TO THE CONTROL PROBLEM

GP has been applied to the search of a controller structure for the control of the heading and propulsion dynamics of CyberShip II. The objective of the control is the same as in the GA optimisation (i.e. to obtain a good tracking of the desired response and minimise the use of the actuators). GP does it by evolving tree structures along the generations. Therefore there is no need for choosing *a priori* the size or the shape of the solutions. They are dynamically evolved.

Every solution to the control problem stated will consist of two independent trees: one for heading control and other for propulsion control (i.e. a decoupled controller). Given the good results obtained with decoupled controllers in the GA problem (Sections 6.3 and 6.5) and the fact that it simplifies the GP structure, this two-tree structure is chosen as the most appropriate.

For comparison purposes, the cost function and manoeuvres are the same as those used for the GA optimisation of controllers (see Equation (5.1) from Section 5.2.2 and Figure 3.9 from Section 3.5). The best resulting controllers have been also tested in the water basin facilities in NTNU.

7.3.4. TERMINAL AND FUNCTION SET

The terminal and function sets have been chosen taking into account the results obtained in previous chapters with the GA optimisation. The terminal set consists of 4 terms: error, state, reference and one numerical constant. Thus, the propulsion terminal set consists of the surge error ($\varepsilon_p = u_d - u$), surge (u) and desired surge (u_d) plus one numerical constant. The heading terminal set consists of the heading error ($\varepsilon_h = \psi_d - \psi$), heading ($\psi$) and desired heading ($\psi_d$) plus one numerical constant. The terminal sets are shown in Table 7.1.

Table 7.1: Terminal Sets for Propulsion and Heading

Propulsion	Heading
surge error (ε_p)	*heading error (ε_h)*
u	ψ
u_d	ψ_d
R	R

The probability of generating a numerical constant is three times bigger than the probability of choosing any of the other terminals (i.e. probability of 0.5).

The function set is formed by eleven functions that are related to the controllers described previously, 5 two-arguments functions, 5 one-argument functions and one function that has two arguments when use for heading and only one when used for propulsion. They can be seen in the next Table 7.2.

Table 7.2: Function Set

Two-argument functions	One-argument functions	One/Two-argument functions
$\mathrm{arg1} \cdot \mathrm{arg2}$	$\int \mathrm{arg}\, dt$	place (-arg) place (0, -arg1, -arg2)
$\mathrm{arg1} + \mathrm{arg2}$	$\frac{d(\mathrm{arg})}{dt}$	
$\mathrm{arg1} - \mathrm{arg2}$	$\mathrm{arg} \cdot sign(h'(x - x_d))$	
$\mathrm{arg1} / \mathrm{arg2}$	$\sin(\mathrm{arg})$	
$\mathrm{arg1} \cdot \tanh\left(\frac{h'(x - x_d)}{\mathrm{arg2}}\right)$	$\exp(\mathrm{arg})$	

The four basic arithmetic operations {+, -, *, /} are routinely included in most GP algorithms. The *integral* and *derivative* functions were included to account for a *PID* type of structure (see Section 4.3). The *hyperbolic tangent* and *sign* functions will allow the construction of *switching terms* equivalents, similar to Sliding Mode control (see Section 4.5). The *place command* was included because of the good results obtained in the GA optimisation of Pole Placement (see Section 4.4). In addition, the *sine* and *exponential* functions will give more versatility to the algorithm. Although many more functions could have been added, it was not advisable, since the performance of the GP algorithm degrades with the addition of numerous functions [Koza (1992)].

In the hyperbolic tangent and sign formulas from Table 7.2, $x = [u]$ and $x_d = [u_d]$ for the propulsion control tree, while $x = [v,\ r,\ \psi]^T$ and $x_d = [v_d,\ r_d,\ \psi_d]^T$ for heading. The **h** matrix is the right eigenvector associated to a zero pole for the desired closed-loop system matrix as defined in Equation (4.19) and calculated based on the best solution found in the GA optimization of the decoupled Sliding Mode controller (see Table 6.18). Both functions use one of the arguments as a gain and in that sense they are similar to the node gain method presented in Esparcia-Alcazar (1998).

The place command returns the value $-\mathbf{k}\cdot\mathbf{x}$, where $\mathbf{x}$ is as defined before and $\mathbf{k}$ is the feedback matrix obtained by executing *place (0, -arg1, -arg2)* in the heading control tree or *place (-arg)* in the propulsion control. Since the other functions in the function set only produce real values, the poles to be assigned are always real numbers.

Besides, in order to ensure that the closure property is met, some of the functions have some "protection mechanism", to avoid situations where the solution is not defined (for example, division by zero). Thus, the division function is encoded so that if the denominator is 0, the result of the division is set to 1. Also, the hyperbolic tangent function returns *arg1* when *arg2* is 0. The most likely function to give problems is the *place* command. It has been set to return 0 if there is any error message activated (for example if the poles are too close).

7.3.5. TWO WAYS OF GENERATING NUMERICAL CONSTANTS FOR GP

The generation of numerical constants has been approached in two different ways:

Random Generation of the Numerical Constants

The first GP (GP+RG) algorithm uses Koza's (1992) suggestion of simply introducing a random constant *R* in the set of terminals so that every single time this terminal is chosen a random number is generated and associated with that terminal node. The GP should be able to generate other constants needed by using arithmetic operations to create them. This is a very simple approach but it may raise questions about the accuracy of the result.

As opposed to Koza's GP that does not use mutation, in this work point mutation has been included as an operator. This enables the GP to modify the terminal values. Thus, a numerical constant can change its value and a terminal occupied by a variable can be mutated into a numerical constant.

The random constants have been generated using the *rand* command and multiplying by 100, so that the range covered is [0, 99.99]. This range has been chosen by inspection of the results typically obtained in the GA parameter optimisation.

Each GP scheme was run 20 times. The population size used was 120 and the number of generations was 30. Therefore the population size is larger than for GA and the number of generations smaller. The recommendation in the GP literature is to use more and shorter runs with larger populations mainly to avoid the bloat problem [Van Belle and Ackley (2001), Dracopoulos and Kent (1997)].

GAs Optimisation of the Numerical Constants

The second GP algorithm tested uses a GA as a parametric optimisation technique combined with the GP. The aim is that the GP+GA algorithm provides a better parameter adjustment and therefore, hopefully, better results.

The GP+GA hybrid method used in this study is fundamentally different from the GA-P mechanism presented by Howard and D'Angelo (1995). Instead of associating a GA chromosome with a GP tree and evolving them together, GP+GA combines a GP evolution process with a GA learning process, i.e. every time a tree is evaluated a mini-GA is run prior to the evaluation to optimise the values of the numerical constants present in that tree. Thus, the individual is able to acquire a better cost value and, consequently, has better chances of being selected for reproduction. The big advantage over the Howard and D'Angelo (1995) technique is that the maximum number of numerical constants in the tree does not need to be fixed and only those constants that are actually in the tree are encoded, reducing substantially the size of the chromosomes to evolve.

The GP+GA algorithm has been coded so that the total number of tree evaluations is the same as in the GP+RG case, providing a good basis for comparison. The number of trees in the population for the single GP is 120 and the number of generations is 31, so the total number of evaluations is 3720. In order to get the same number of evaluations for the GP+GA optimisation, the GP will have a population of 31 individuals and it will be run for 7 generations. Each GA will have a population of 5 individuals and it is run for 3 generations.

Moreover, this choice seems very appropriate because the ratio between the number of individuals in the population and the number of generations is 3.8710 for the GP+RG case and practically the same for the GP in the GP+GA algorithm, i.e. 3.8750. The GA ratios are also similar: 1.5686 for the GA used in previous chapters and 1.6667 for the GA in the GP+GA optimisation.

The GA used has been the one found to be best in Chapter 5, i.e., tournament selection, non-uniform exponential mutation and probability of crossover of 0.8. Each parameter is encoded using 3 genes: tens, units and decimals. This way the range covered is equivalent to that of the random numbers, from 0 to 99.9.

7.4. SUMMARY

This chapter has illustrated the general structure, tree representation and operators of GP, together with a practical implementation of how to use GP to solve a control problem. It has emphasised the different representation of the candidate solutions between GAs and GP. The tree structures typically used in GP provides an extra degree of freedom to the evolution process, allowing the evolution of the size and shape of the solutions, as opposed to the fixed-length of the GA chromosomes that needs to be determined prior to the optimisation process.

Regarding the genetic operators, it has been shown that the selection procedure is equivalent to that of GAs, although the mutation and crossover operators have been modified in order to be adapted to the new structure. The tree shape allows for more variety of the genetic operators, especially mutation.

It has been shown that the generation of numerical constants is an important issue in GP optimisation. Two alternative methods for this problem have been presented: one is based in random generation while the second method is more complex, being a combination of GP and GA. Both methods have been used to obtain the results presented in the next chapter.

The problem of uncontrolled code growth, its causes and consequences as well as techniques to avoid it have also been discussed. Since the size of the solutions is an important issue in GP in the next chapter the average tree sizes found in the final population of the GP optimisation runs are illustrated and discussed.

CHAPTER 8

GENETIC PROGRAMMING: A CONTROL APPLICATION FOR A MARINE VESSEL

8.1. INTRODUCTION

This chapter presents the results obtained when using a GP implementation to optimise the structure of controllers for the problem of the control of the heading and propulsion dynamics of CyberShip II.

The two Matlab implementations introduced in Chapter 7 (i.e. GP+RG and GP+GA) have been used to search for two structures that solve the control problem near-optimally (i.e. one for propulsion, one for heading). Each implementation has been optimised with and without waves in the simulation and the best resulting controllers have been tested in the real plant. This study allows the comparison of the performance of both GP schemes and further assesses the relevance of including waves in the evaluation process, already discussed in the results obtained with the GA optimisation in Chapter 6.

The examination of the results has focused in four different aspects of the optimisation. First of all, the best cost values obtained in each run are analysed. In addition, the results are tested using a *validation function*. A validation function is basically a cost function used after the optimisation in off-line tests to validate the result obtained with the GP. Off-line tests are used extensively throughout the GP literature [O'Reilly (1995), Tackett (1994)]. There are problems that traditionally have received much attention from GP researchers such as the 6 and 11 bit Boolean multiplexer [Koza (1992), O'Reilly (1995)] that allow the inclusion of all possible test cases in the evaluation of the candidate solutions. However that is not the case for most problems. Regarding the evaluation of solutions Koza (1992) stated that: "Typically, each computer program in the population is run over a number of different fitness cases so that its fitness is measured as a sum or an average over a variety of representative different situations". In some cases time limitations do not permit the evaluation of various fitness cases so an off-line validation function can be used for evaluating the generality of a solution that performs well during the GP optimisation.

Secondly, the analysis has focused on the structure of the best solutions of each run. All the equations that represent the best tree structure obtained in each run can be found in Appendix D together with figures illustrating their manoeuvring performance. It has been found in this study that the structures can be placed in groups that exhibit similar forms. The most common groups are presented and discussed in this chapter. The real and simulated responses of the overall best structure when tracking the optimisation and validation manoeuvres are plotted.

The third aspect taken into consideration is the size of the resulting trees from each run and those of the final population. In Chapter 7 the consequence of the uncontrolled growth of code [Koza (1992), Luke (2000c)] has been explained, as well as the techniques used to avoid the problem and the most relevant theories about its cause. In this chapter the effect that this phenomenon has in our implementation is discussed.

Finally, the investigation focuses on the percentage of occurrence of the functions from the function set in the best solutions. This study will hopefully allow conclusions to be drawn about why the GP favours some functions over others.

This chapter is structured as follows: Section 8.2 deals with the evaluation manoeuvre and cost function used in the GP optimisation together with the method used for the validation of the GP results. Section 8.3 shows the results obtained using GP+RG to optimise the control strategy, with and without the inclusion of waves in the optimisation, Section 8.4 presents the results obtained with the combined algorithm GP+GA, again with and without the inclusion of waves in the optimisation, and in Section 8.5 conclusions are drawn from these results.

8.2. GP RESULTS VALIDATION

For comparison purposes, the manoeuvre used for the GP optimisation has been the same simultaneous double step manoeuvre used for GAs (see Figure 3.9). Each GP scheme (i.e. GP+RG and GP+GA) has been run 20 times, with and without disturbances.

The best results found have been validated after the GP optimisation. The reason for this validation test is to verify that the resulting tree is actually performing a control task, not merely generating a signal shaped in the right way for this manoeuvre but totally wrong for any other. This has not been necessary in the GA optimisation because it is based on a control structure already established.

The manoeuvre used for the validation test is shown in Figure 8.1. It consists of two turning circle manoeuvres linked together, first to port and then to starboard. The resulting trajectory is an ∞ shape. The reason for choosing this specific manoeuvre is that the Maritime Safety Committee, in its Resolution MSC.137(76) [Anonymous (2002)] on standards for ship manoeuvrability, recommends the turning point manoeuvre together with the zig-zag manoeuvre as the most appropriate for ship performance testing.

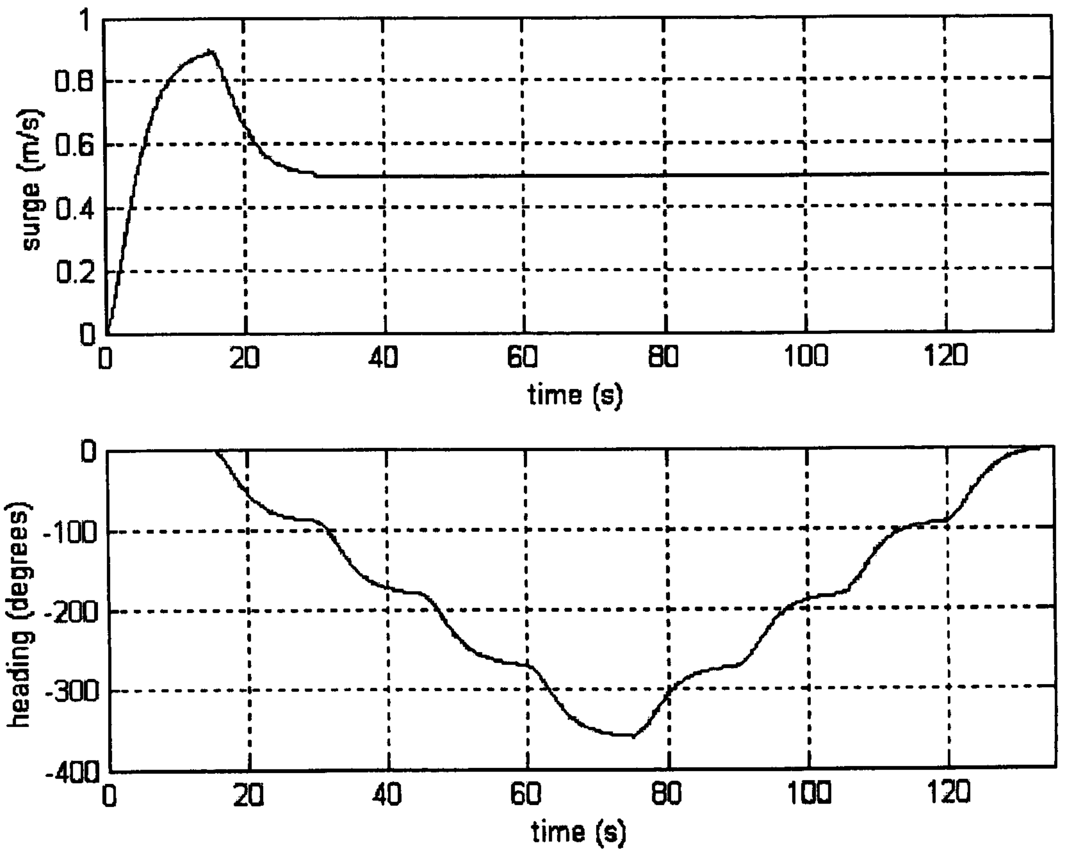


Figure 8.1: Desired Responses for the Validation Test

8.3. GP WITH RANDOM GENERATION RESULTS

8.3.1. OPTIMISATION WITHOUT WAVES: SIMULATED AND REAL RESULTS

Analysis of the Best Costs

The following Table 8.1 presents the cost values obtained after the GP+RG optimisation using the simultaneous double step manoeuvre without waves and the posterior validation test with the double turning point manoeuvre. The figures with the performance of each controller can be found in Appendix D (Section D.1) .

Table 8.1: Best GP+RG Results
Optimisation Without Waves

Run No.	Double Step	Turning Point
1	2.44	91.80
2	4.14	243.60
3	28.63	314.54
4	200.73	552.00
5	3.04	13981.99
6	13.20	3824.18
7	1.92	3431.07
8	152.54	3544.74
9	4.75	325.34
10	237.84	865.02
11	162.56	1230.30
12	167.86	867.61
13	421.25	5104.16
14	540.99	53151.16
15	540.99	53124.88
16	541.07	116868.37
17	15.86	196863.84
18	2.13	4339635.54
19	2.53	$2.74 \cdot 10^{9}$
20	1.99	$5.67 \cdot 10^{12}$

The best overall result is obtained in run 1. It is not the best result achieved in the optimisation but the result in the validation test is extremely good. The amount of convergence in the final population of this run has been 32 individuals and the generation of convergence has been the 10th generation. Results from runs 2 and 9 have been also consistently good for both manoeuvres. On the other hand, the results from runs 18, 19 and 20 are extremely bad. This corroborates the importance of the validation test since those three controllers converged to a very good cost value during the optimisation.

Analysis of the Structure of the Results

Looking at the best controller structures, the results show similar patterns and these can be categorised into six similar groupings.

1st Group

The first group structure usually consists of a hyperbolic tangent function providing heading control and a proportional term for propulsion control. To this group belong the best results obtained, such as those of runs 1, 2 and 3. Results from runs 4 and 5 have a similar structure and they have also been included in this group, although the cost values they have obtained are worse. Analysing the range of values that perform as argument of the hyperbolic tangent it is easy to see that the function is not reaching the saturation limits, therefore it is acting as a proportional term [McGookin (1997)]. For example, the result obtained in run 1 is as follows:

$$\tau_{1com} = 88.4 \cdot \varepsilon_p \tag{8.1}$$

$$\begin{aligned}\tau_{3com} &= -68.2 \cdot \tanh\left(\frac{\mathbf{h}_h' \cdot (\mathbf{x}_h - \mathbf{x}_{hd})}{7.6}\right) + \\ &+ 0.1332 \cdot \left(-68.2 \cdot \tanh\left(\frac{\mathbf{h}_h' \cdot (\mathbf{x}_h - \mathbf{x}_{hd})}{-68.2}\right) - \sin(\psi)\right) + 11.7 \cdot \tanh\left(\frac{\mathbf{h}_h' \cdot (\mathbf{x}_h - \mathbf{x}_{hd})}{\sin(91.5)}\right)\end{aligned} \tag{8.2}$$

Equation (8.1) is just a proportional action, while the heading control expression seems to be very complicated. However, once it is analysed it can be simplified just by applying:

$$\tanh\left(\frac{\mathbf{h}_h' \cdot (\mathbf{x}_h - \mathbf{x}_{hd})}{7.6}\right) = 0.132 \cdot \mathbf{h}_h' \cdot (\mathbf{x}_h - \mathbf{x}_{hd}) \tag{8.3}$$

$$\tanh\left(\frac{\mathbf{h}_h' \cdot (\mathbf{x}_h - \mathbf{x}_{hd})}{-68.2}\right) = -0.0148 \cdot \mathbf{h}_h' \cdot (\mathbf{x}_h - \mathbf{x}_{hd}) \tag{8.4}$$

$$\tanh\left(\frac{\mathbf{h}_h' \cdot (\mathbf{x}_h - \mathbf{x}_{hd})}{\sin(91.5)}\right) = -2.63 \cdot \mathbf{h}_h' \cdot (\mathbf{x}_h - \mathbf{x}_{hd}) \tag{8.5}$$

This results in a proportional controller plus a sine term, as shown in Equation (8.6). If the heading of the ship is smaller than 0.5 rad (30° approx.) the sine can be approximated by the heading signal.

$$\tau_{3com} = -38.868 \cdot \mathbf{h}_{\mathrm{h}}' \cdot (\mathbf{x}_{\mathrm{h}} - \mathbf{x}_{\mathrm{hd}}) - 0.1332 \cdot \sin(\psi) \qquad (8.6)$$

2nd Group

The second group of results is characterised by the use of the P and PD type of controllers. To this group belong structures such as those of runs 6, 7, 8 or 9. The resulting controller from run number 9 is as follows:

$$\tau_{1com} = 29 \cdot \varepsilon_p + \sin(u_d) \qquad (8.7)$$

$$\tau_{3com} = 12 \cdot \varepsilon_h + \frac{d\varepsilon_h}{dt} \qquad (8.8)$$

The propulsion controller consists of a proportional term plus a sine term. For small angles the sine is approximately equal to the angle. Therefore, at low speed (under 0.5m/s), $sin(u_d) \approx u_d$. Since the gain of the proportional term is quite small, especially if compared with the proportional propulsion controller obtained in run 16 with a gain of 88.4 (as shown in Equation (8.1)), the way the GP found of increasing the control effort is by adding a type of open-loop step, instead of augmenting the gain. Obviously, this is not a very good control practice.

3rd Group

Some other results, such as runs number 10, 11 and 12, have converged to results where the main control action is provided by a trigonometric function. For example, the structure of the resulting controller from run number 10 is as follows:

$$\tau_{1com} = \sin(\varepsilon_p - 5.6432) \qquad (8.9)$$

$$\tau_{3com} = \sin(\varepsilon_h) \qquad (8.10)$$

Once more, for small angles $sin(\alpha) \approx \alpha$ and consequently the heading control strategy is basically proportional control.

4th Group

The fourth group is formed by all those controllers that set one of the commanded signals to zero. This is a kind of "damage limitation" technique: a zero control action provides a better cost than an unstable one. Results from runs 13 and 14 belong to this group. Run number 14 has converged to a zero action in the heading control and a proportional action for propulsion; on the other hand number 13 has a proportional controller for heading and basically zero control for propulsion.

5th Group

The fifth group comprises the results from runs number 15 and 16. Both use a pole placement technique in the heading control and a PI for propulsion. As an example the structure of the controller obtained in run 15 is shown:

$$\tau_{1com} = 302.5 \cdot \varepsilon_p + 49.2 \cdot \int \varepsilon_p \cdot dt + 2 \cdot \varepsilon_p^{\ 2} \tag{8.11}$$

$$\begin{gathered} \tau_{3com} = -\mathbf{k} \cdot \mathbf{x_h} = 0 \\ \mathbf{k} \equiv place(0, \quad -\psi \quad -70.8) \end{gathered} \tag{8.12}$$

For small errors the term of the square error in Equation (8.11) is very small so the control action is effectively a PI controller. The pole placement technique gives very bad results, which is understandable as there is no reference signal included in the heading controller structure and therefore the commanded heading signal remains at zero. Initially the heading dynamics are set to zero and consequently the outcome of the product $-\mathbf{k} \cdot \mathbf{x_h}$ is zero. Since there is no control action the heading is never modified and, consequently, the heading control action remains null. Also, to place one of the poles depending of the value of one of the states causes instability once the state changes its polarity.

6th Group

In addition, some other results, although they do not share a common structure, belong to the group of results that after performing very successfully in the simultaneous double step manoeuvre have obtained really poor results in the validation test. Clear examples are the results from runs 17, 18, 19 and 20. For instance, after getting the second best result in the optimisation process, the controller from run number 20 obtains the worst result by far in the validation process. The controller structure is very complicated, as can be seen from the following equations:

$$\tau_{1com} = 344 \cdot \varepsilon_p + 3 \cdot \frac{d\varepsilon_p}{dt} + 34.5 \cdot u \cdot \tanh\left(\frac{h_p' \cdot (u - u_d)}{u_d}\right) \tag{8.13}$$

$$\tau_{3com} = \left(-\mathbf{k}_1 \cdot \mathbf{x}_h \cdot \tanh\left(\frac{\mathbf{h}_h' \cdot (\mathbf{x}_h - \mathbf{x}_{hd})}{-68.2} \right) - 31.2 \right) \cdot \tanh\left(\frac{\mathbf{h}_h' \cdot (\mathbf{x}_h - \mathbf{x}_{hd})}{0.9511} \right)$$

$$\begin{gathered} \mathbf{k}_1 \equiv place\left(0,\ \ 94.6 \cdot sign\left(\mathbf{h}_h' \cdot (\mathbf{x}_h - \mathbf{x}_{hd})\right),\ \ \mathbf{k}_2 \cdot \mathbf{x}_h\right) \\ \mathbf{k}_2 \equiv place\left(0,\ \ 94.6 \cdot sign\left(\mathbf{h}_h' \cdot (\mathbf{x}_h - \mathbf{x}_{hd})\right),\ \ \mathbf{k}_3 \cdot \mathbf{x}_h + \psi_d\right) \\ \mathbf{k}_3 \equiv place\left(0,\ \ -0.9511,\ \ \mathbf{k}_4 \cdot \mathbf{x}_h\right) \\ \mathbf{k}_4 \equiv place\left(0,\ \ -0.9511,\ \ -d\psi/dt\right) \end{gathered} \qquad (8.14)$$

Just by inspection it can be seen that the control structure for heading does not make sense. It includes four pole placement commands and the way that the poles are defined does not ensure their position in the left side of the s-plane.

Next Figures 8.2 and 8.3 show the performance of the controller considered to be best obtained from run number 1 (see Equations (8.1) and (8.6)) when tracking the evaluation and the validation manoeuvres.

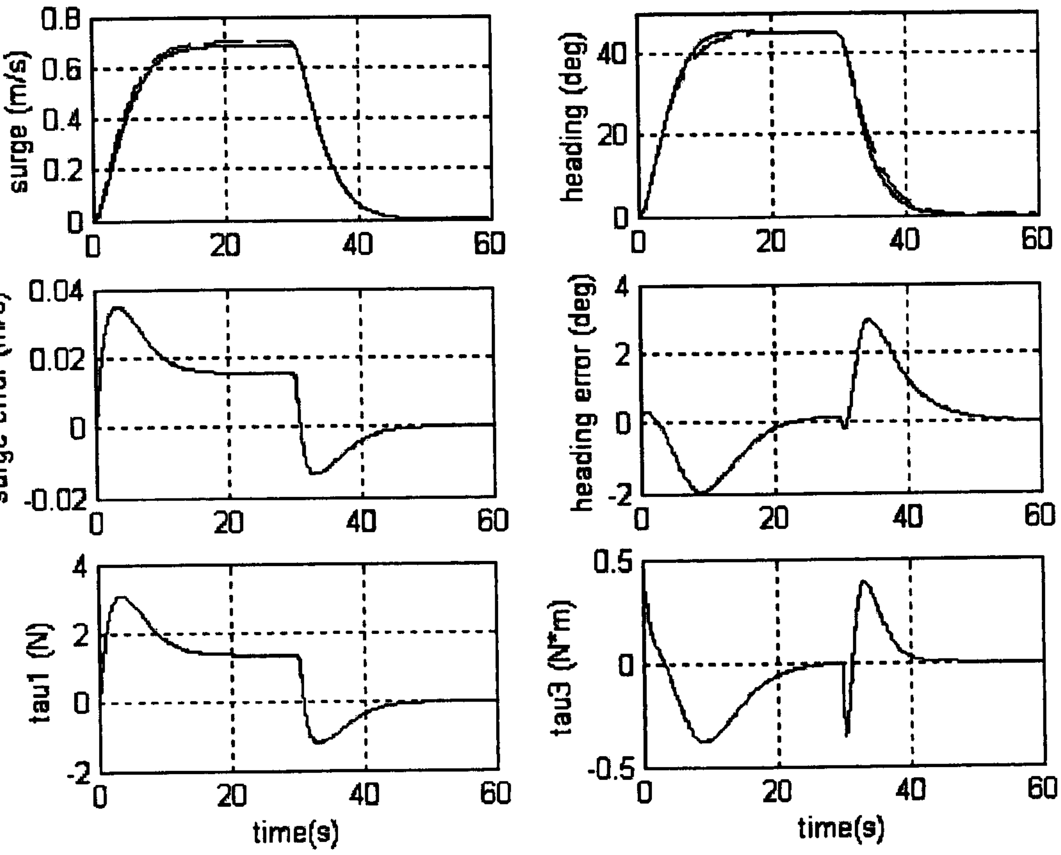


Figure 8.2: Simulated Results of the GP+RG Optimisation Without Waves

The performance of the controller for the simultaneous double step manoeuvre is quite good. There is a slight steady-state error in the surge response caused by the lack of an integral term in the propulsion control (see Equation (8.1)). The heading control is also good apart from a slight overshooting caused by the high gain.

The performance shown in the previous figure is very similar to that of the decoupled PID optimised without waves but with a larger steady-state error in the propulsion subsystem caused by the lack of integral term and the small proportional gain imposed by the lack of derivative term.

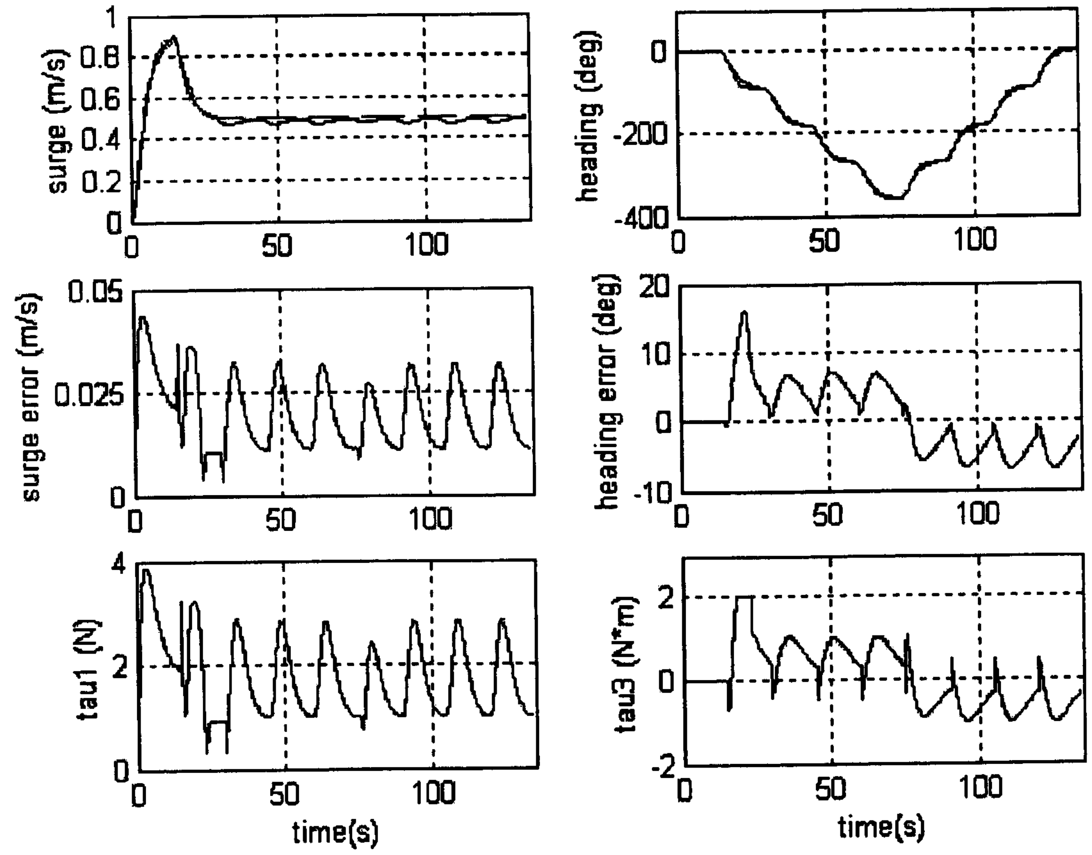


Figure 8.3: Validation Test for the Result of the GP+RG Optimisation Without Waves

Given that the double turning point manoeuvre is very demanding, both controllers perform quite well. It can be seen that the tracking of the desired responses is quite good. The surge speed signal has some oscillations due to the coupling of the system, which are induced by the constant turning motion caused by the heading control system. Also, the heading manoeuvre is tracked well, although the same slight overshooting noticed in the simultaneous double step manoeuvre can be perceived. It can be seen in the τ_3 force that the actuator's amplitude limits has restricted the commanded signal to 2 N·m.

Figure 8.4 and 8.5 show the responses obtained when the controllers from Equations (8.1) and (8.6) have been implemented in the real plant. Figure 8.4 illustrates the results obtained when manoeuvring in calm waters.

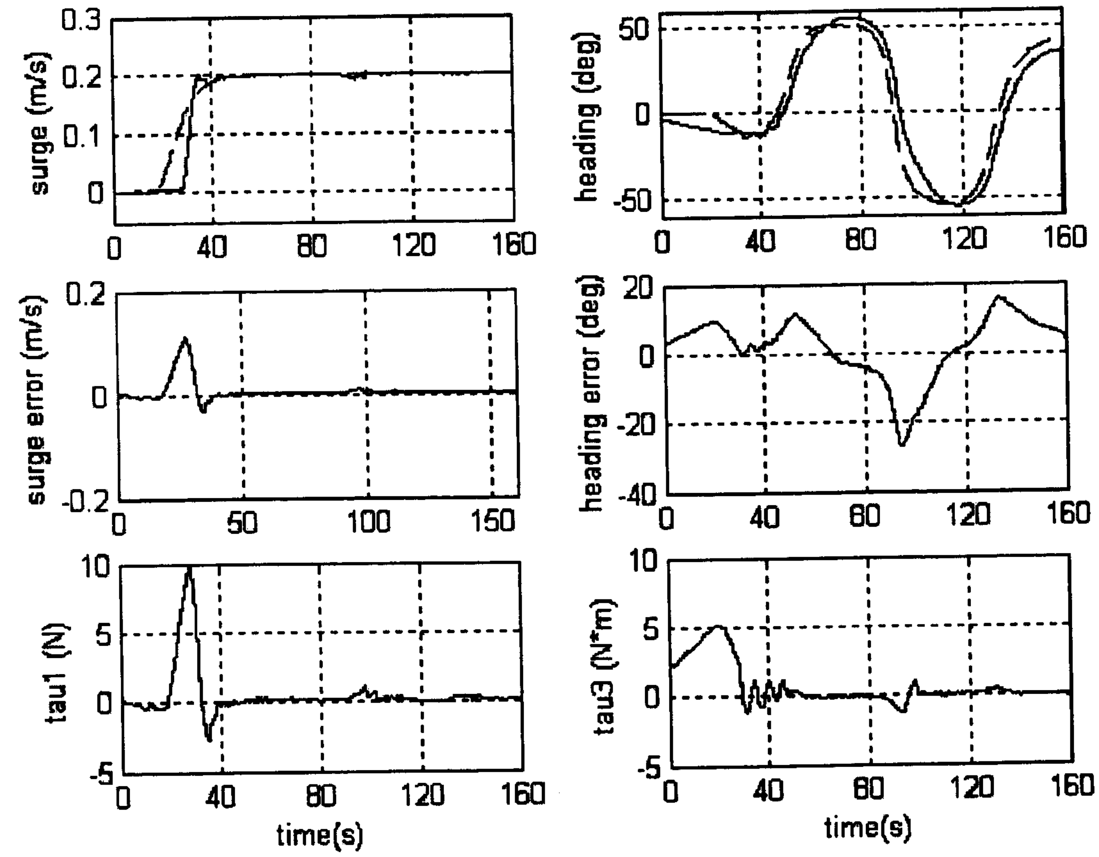


Figure 8.4: Real Results of the GP+RG Optimisation Without Waves When Manoeuvring in Calm Waters

The responses obtained are quite good. The tracking is quite accurate, although the same overshooting that has been observed in the simulated responses for heading can be also appreciated in the real responses. A delay introduced by the system can be observed in the responses. This induces very high initial control signals until the ship starts the manoeuvre.

Figure 8.5 shows the effect of waves in the performance of the controllers obtained in the GP+RG optimisation without waves. As previously the waves generated are those presented in Section 3.5.

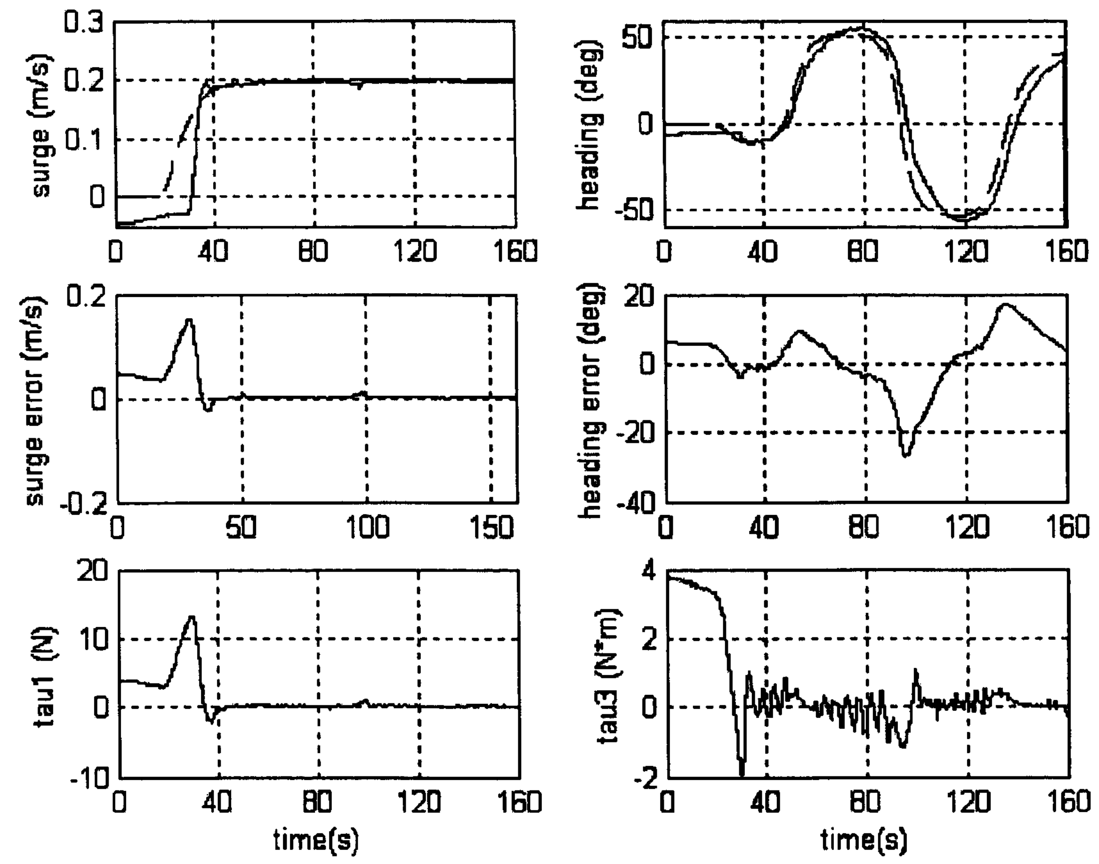


Figure 8.5: Real Results of the GP+RG Optimisation Without Waves When Manoeuvring in the Presence of Waves

Since all the real results in this chapter have been obtained with an improved system, the effect of the waves is not very significant, although some wave-induced high frequency components can be observed in the heading control action. As in Figure 8.4, a delay can be observed in the responses of the boat.

Analysis of the Sizes of the Trees

Next, in Table 8.2 the average size of the trees in the final population together with the size of the best tree and their optimisation cost values are presented. Tree size is defined as the number of internal nodes in a tree. In order to clarify the posterior discussion the results are listed in ascending order starting with the best cost.

From Table 8.2 it seems that those results based on a big heading tree are associated with better cost values, while bigger propulsion trees produce worse costs. The best result is obtained with a small propulsion tree and a large heading tree. Nevertheless, there are some exceptions so it cannot be considered a conclusive rule.

It is interesting to notice that the best tree size and the average size of the final population for each run are always very close. This would indicate a convergence towards trees of the same size during the optimisation.

Table 8.2: Size of Trees in the GP+RG Final Populations
Optimisation Without Waves

Cost value	Run No.	Propulsion		Heading	
		Best Tree	Avg. Pop.	Best Tree	Avg. Pop.
1.92	18	1	1.03	28	21.88
1.99	20	16	16.17	25	20.99
2.13	18	17	8.52	29	20.78
2.44	1	3	3.05	28	17.75
2.53	19	11	7.44	7	5.01
3.04	5	6	4.37	5	5.87
4.14	2	11	15.83	2	2.05
4.75	9	30	25.25	13	12.46
13.20	6	1	1.04	29	24.75
15.86	17	9	12.91	2	2.05
28.63	3	29	26.42	6	5.99
152.54	8	11	9.17	2	2.00
162.56	11	28	23.86	2	2.04
167.86	12	1	1.08	1	1.06
200.73	4	1	1.04	3	3.05
237.84	10	2	2.05	1	1.05
421.25	13	30	25.87	2	2.06
540.99	16	29	26.72	1	1.74
540.99	14	29	26.53	2	1.08
541.07	15	30	28.17	1	2.92

The bloat problem has not been a real issue while running these optimisations. Only 5 heading trees (i.e. 25%) and 7 propulsion trees (35%) have reached the maximum size allowed and in the analysis of the structure the emergence of introns has been rare. According to the most recent theory about code bloat by Luke (2000c), the bias towards deeper crossover points (less likely to destroy the individual) is causing this bloat effect. If this is the case, the matrix representation chosen in Matlab could be the reason for the lack of bloat in this case. Since every row in a tree represents an internal node, subtree crossover and mutation only act over internal nodes, never terminals. This implies that the minimum size for a removed subtree is already 2 or 3 nodes (1 internal node plus 1 or 2 terminal node depending on the number of arguments the function has). All the functions have the same probability of being chosen to randomly generate the new subtree in subtree mutation. Approximately half of the functions in the set terminal require one argument and the other half need two. Since the randomly generated mutation subtrees have a depth of 3, approximately half of the new subtrees to be inserted in subtree mutation have 3 or 4 nodes, and the other half between 5 and 7. Therefore, even if the deepest mutation point available is chosen for mutation (worst-case scenario), 50% of the times the subtree mutation operator only implies an increase in size of maximum 1 node or no increase at all. Moreover 50% of the mutations are just point mutation which does not increase the size of the tree. Thus, it is very unlikely that the subtree mutation will lead to a very quick increase in tree size. Regarding the subtree crossover the results are more uncertain, but the fact that the deepest node available for crossover is an internal node reduces the chances of a large increment in size.

This absence of bloat has caused that the execution times are not as long as expected for a GP optimisation. The execution time in the absence of bloat is proportional to the evaluation time for the desired manoeuvre. In the optimisation problem considered in this work the execution time for the GP has been of around 6 hours.

Analysis of the Percentage of Occurrence of Functions

In Figure 8.6 the percentage of final solutions that use a certain function from the function set is shown.

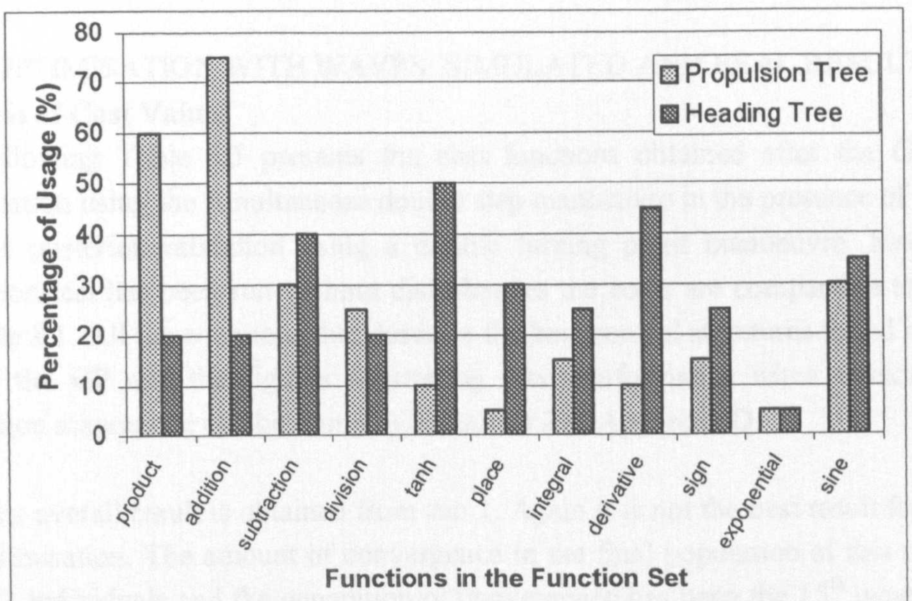


Figure 8.6: Percentage of Usage of Functions in the GP+RG Best Results Optimisation Without Waves

By looking at the data from the propulsion trees in Figure 8.6 it can be seen that by far the most used functions are the arithmetic functions addition and multiplication. This is consistent with the predominant usage of proportional terms reflected in the previous structural analysis of the solutions.

In contrast, the most predominant function in the heading control trees is the hyperbolic tangent. Associated with a high percentage of hyperbolic tangents there must always be a high proportion of minus functions since the sign of the hyperbolic tangent needs to be reversed to provide the appropriate control signal (or the function operated on by the hyperbolic tangent is negated).

Around a third of both heading and propulsion resulting trees employ the sine function. Sometimes it is applied to generate new constants but very often can be neglected since for small angles $sin(\alpha) \approx \alpha$.

It is important to remark that, although the study of the frequency of occurrence of functions in the final results can lead to interesting conclusions, the figures can be quite misleading. For example, the high percentage of derivative and integral terms for the heading trees shown in the previous Figure 8.6 is not a true reflection of the actual results. Very often both terms cancel each other out, thus, although there are various resulting heading controllers that do include a derivative action, all the integral terms but one are effectively cancelled.

8.3.2. OPTIMISATION WITH WAVES: SIMULATED AND REAL RESULTS

Analysis of Cost Values

The following Table 8.3 presents the cost functions obtained after the GP+RG optimisation using the simultaneous double step manoeuvre in the presence of waves and the posterior validation using a double turning point manoeuvre. Since the validation test has been run without disturbances the costs are comparable to those of Table 8.1. All the equations that describe the best control structures found in each run of the GP and the figures illustrating their performance when tracking the evaluation manoeuvre can be found in Section D.2 in Appendix D.

The best overall result is obtained from run 1. Again it is not the best result from the GP optimisation. The amount of convergence in the final population of this run has been 43 individuals and the generation of convergence has been the 15th generation. Run number 5 provided the best result from the GP runs and the validation result is reasonably good. Other controllers that have performed well in both optimisation and validation test are those of runs 2 and 4. As with the optimisation without disturbances, some of the good results from the GP optimisation failed the validation test (e.g. results from runs 8, 19 and 20).

Table 8.3: Best GP+RG Results
Optimisation With Waves

Run No.	Double Step	Turning Point
1	12.90	79.78
2	79.15	213.40
3	123.78	360.49
4	18.09	194.95
5	7.63	289.88
6	51.55	646.29
7	120.11	423.66
8	12.34	593943.52
9	370.40	949.33
10	328.79	843.70
11	285.32	841.22
12	53.37	534.41
13	132.25	445.48
14	114.01	705.51
15	275.93	954.56
16	95.92	150968.05
17	327.72	53316.41
18	351.56	53180.82
19	30.52	735485.71
20	47.73	1501055.75

Analysis of the Structure of the Results

The most representative structures of the best trees obtained through the various runs of the GP+RG optimisation are discussed below. They can be approximately grouped into five categories.

1st Group

The first group comprises those structures whose heading and propulsion controller are based on a hyperbolic tangent. Results from runs 1, 2 and 3 belong to this group and all of them provided reasonably good results. As an example the structure of the controller obtained in run 1 is shown:

$$\tau_{1com} = -203.5368 \cdot \tanh\left(\frac{h_p' \cdot (u - u_d)}{11.8223}\right) + 2 \cdot \sin(u_d) \tag{8.15}$$

$$\tau_{3com} = 14.752 \cdot \tanh\left(\frac{\mathbf{h}_h' \cdot (\mathbf{x}_h - \mathbf{x}_{hd})}{-0.7931}\right) \tag{8.16}$$

Again the hyperbolic tangents are used as proportional controllers. Both equations could be expressed as:

$$\tau_{1com} \approx -17.5 \cdot h_p' \cdot (u - u_d) + 2 \cdot \sin(u_d) \tag{8.17}$$

$$\tau_{3com} \approx -19 \cdot \mathbf{h}_h' \cdot (\mathbf{x}_h - \mathbf{x}_{hd}) \tag{8.18}$$

The propulsion controller is similar to that obtained in the GP+RG optimisation without waves and shown in Equation (8.7). The sine function of the desired surge speed increases the control effort by acting as a form of feedforward control (i.e. provides the same control effort regardless of the current state of the boat).

2nd Group

The second group also uses hyperbolic tangents for the heading control but the propulsion control varies, using mainly some kind of P/PI type of controller. Results from runs 4, 5, 6, 7 and 8 belong to this group. The controllers obtained in run number 5 are shown as the most representative of the group:

$$\tau_{1com} = 11.1 \cdot \varepsilon_p + 33.3 \cdot \int \varepsilon_p \cdot dt \tag{8.19}$$

$$\tau_{3com} = -1208.4 \cdot \tanh\left(\frac{\mathbf{h}_h' \cdot (\mathbf{x}_h - \mathbf{x}_{hd})}{82.6268}\right) \tag{8.20}$$

Again the hyperbolic tangent is performing as a proportional controller. The equivalent control action is similar to that of Equation (8.18):

$$\tau_{3com} \approx -14.5 \cdot \mathbf{h}_h' \cdot (\mathbf{x}_h - \mathbf{x}_{hd}) \tag{8.21}$$

3rd Group

A mixture of PID type configuration together with sine and exponential functions forms the third group of results. Examples of these structures are the results from runs 9 to 16. A good example is the control strategy obtained in run 9:

$$\tau_{1com} = \sin(\exp(u_d)) \tag{8.22}$$

$$\tau_{3com} = \sin(\sin(\psi_d - \psi)) \tag{8.23}$$

The propulsion control is again based on an open loop control, no feedback is present. For small angles *sin*(α) ≈ α and also, for very small values, *exp*(α) ≈ α+*1*. Therefore at low desired speeds and for small heading errors, the above equations are equivalent to:

$$\tau_{1com} \approx \sin(u_d + 1) \tag{8.24}$$

$$\tau_{3com} \approx \psi_d - \psi \tag{8.25}$$

4th Group

The fourth group consists of two results (runs number 17 and 18) that use a Pole Placement technique for the heading control and some function of the error and u_d for the propulsion control. Both structures performed quite badly in the optimisation and validation test. The structure of the result from run number 3 is as follows:

$$\tau_{1com} = \varepsilon_p + u_d \tag{8.26}$$

$$\begin{gathered} \tau_{3com} = -\mathbf{k} \cdot \mathbf{x}_h \\ \mathbf{k} \equiv place(0 \quad -30.4 \cdot \varepsilon_h - 6 \cdot \psi \quad -\sin(-30.4 \cdot \varepsilon_h - 6 \cdot \psi)) \end{gathered} \tag{8.27}$$

Once more the result is a Pole Placement technique that lacks of reference signal and uses variables for setting the pole positions, which can lead to stability problems.

5th Group

The final group contains a set of two results (runs 19 and 20) with heading control that is based on several nested sine functions of a hyperbolic tangent and the propulsion controller is a proportional term. Both structures work well in the optimisation but they give very poor performance in the validation test. The result from run 20 is shown as an example:

$$\tau_{1com} = 71.13 \cdot \varepsilon_p \tag{8.28}$$

$$\begin{gathered} \tau_{3com} = \sin\left(\sin\left(\sin\left(\sin\left(\sin\left(\sin\left(3156.55 \cdot sign(z) \cdot \tanh\left(\frac{z}{67.5236}\right)\right)\right)\right)\right)\right)\right) \\ z = \mathbf{h}_h' \cdot (\mathbf{x}_h - \mathbf{x}_{hd}) \end{gathered} \tag{8.29}$$

Next Figures 8.7 and 8.8 illustrate the performance of the controllers from run number 1 (see Equations (8.17) and (8.18)) when tracking the simultaneous double step manoeuvre used for the GP optimisation and the double turning point manoeuvre used for the validation test. These controllers have provided the best overall performance.

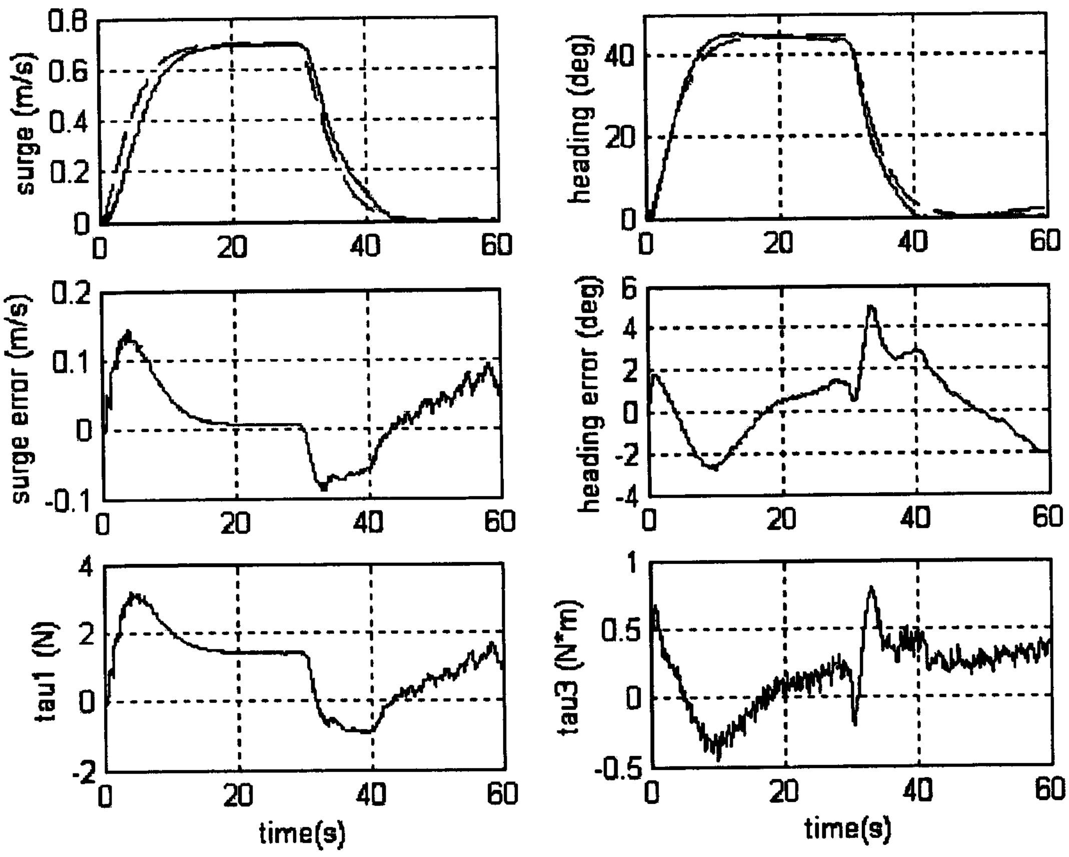


Figure 8.7: Simulated Results of the GP+RG Optimisation With Waves

As in the GA examples, it can be seen that the inclusion of waves affects the propulsion control more than the heading control, although both cope quite well. The surge speed signal is slightly slow in the response. Comparing the results obtained using the GP with and without waves (see Equations (8.17) and (8.1)) it can be observed that although the propulsion control in both controllers is proportional, the proportional gain for the controller optimised with waves is much smaller (17.5 versus 88.4), which explains the slow transient response. The heading control is not significantly affected by the disturbances, apart from the ripple in the actuators force caused by the disturbances. Again, if compared with the heading controller obtained in the GP optimisation without waves (Equations (8.18) and (8.3)), both are proportional controllers but the inclusion of waves has led to a smaller gain (38.9

versus 19). This is linked to the previous conclusion in Chapter 6 of the inclusion of disturbances in the optimisation effectively reducing the actuators usage by reducing the controller gain.

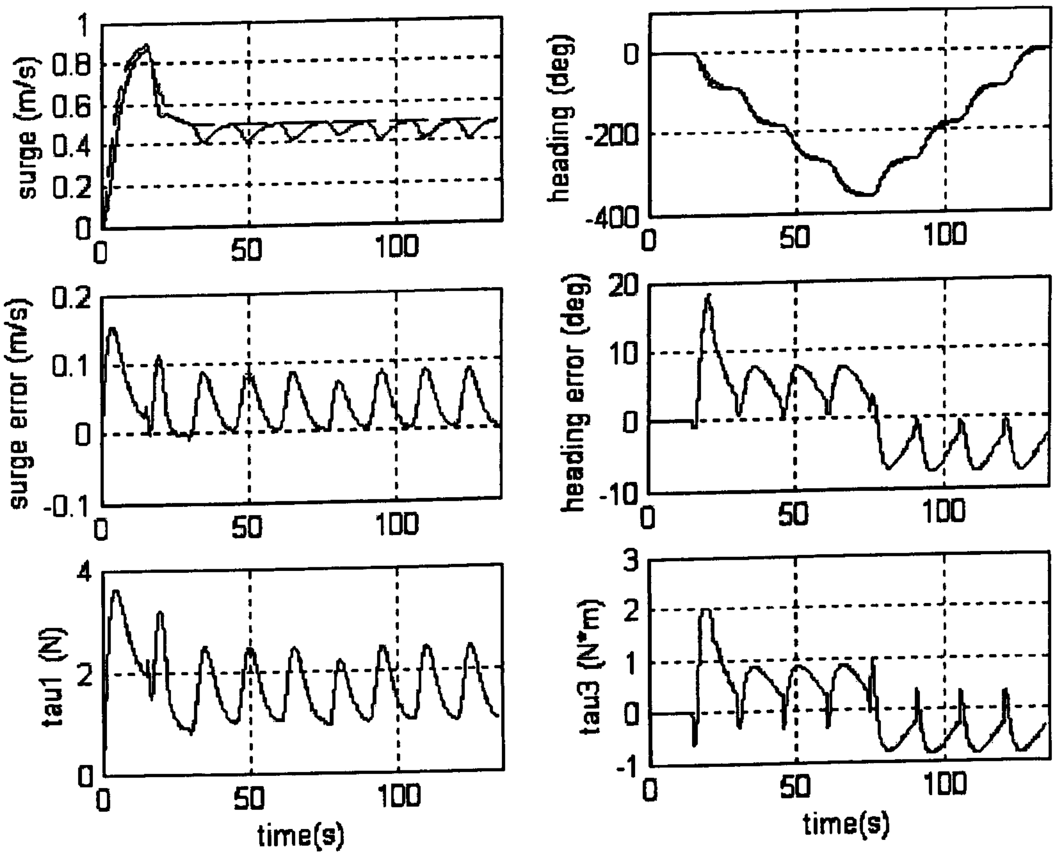


Figure 8.8: Validation Test for the Result of the GP+RG Optimisation With Waves

Figure 8.8 shows a similar result to that of the optimisation without waves (see Figure 8.3). Again the most remarkable feature is all the oscillation that the continuous change in the heading desired signal induces. The surge error is larger than in the result from the optimisation without waves and the speed transient response slower. However the control effort is also smaller and consequently the final cost value obtained by this controller is marginally better than the value obtained by the result from the optimisation without waves (79.78 versus 91.80).

Figures 8.9 and 8.10 show the results obtained when the controllers represented by Equations (8.17) and (8.18) are implemented in the real plant. Figure 8.9 shows the responses obtained when the manoeuvring has been performed in calm waters.

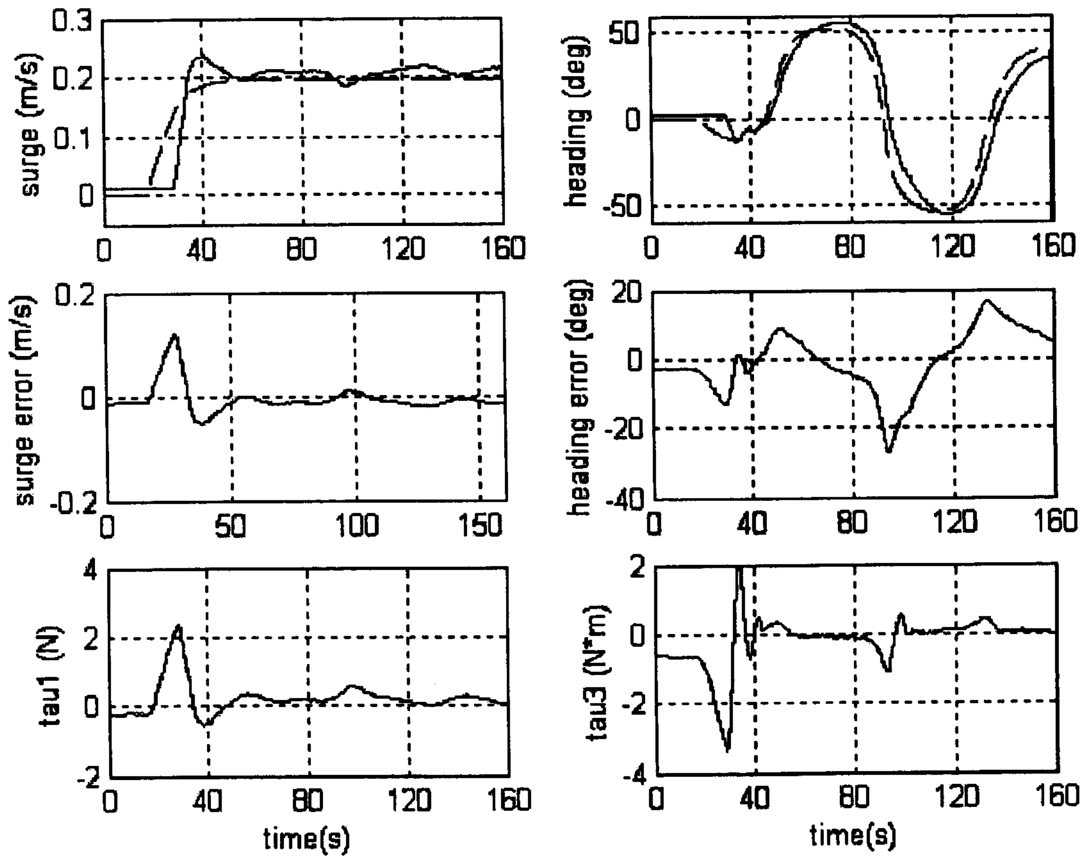


Figure 8.9: Real Results of the GP+RG Optimisation With Waves When Manoeuvring in Calm Waters

The real responses obtained are quite satisfactory. The tracking is quite good, especially for the heading response, although with a slight overshooting. When comparing with the real responses obtained for the GP+RG optimisation without waves (see Figure 8.4) it can be observed that, although the surge response is worse the surge control action has been reduced, as is usually the case when including disturbances in the optimisation.

Figure 8.10 shows the responses obtained with the controllers from Equations (8.17) and (8.18) when the same zig-zag manoeuvre has been performed in the presence of waves in the water tank.

As it can be seen from the figure, the main effect of including waves can be observed in the disturbance-induced ripples in the heading control action.

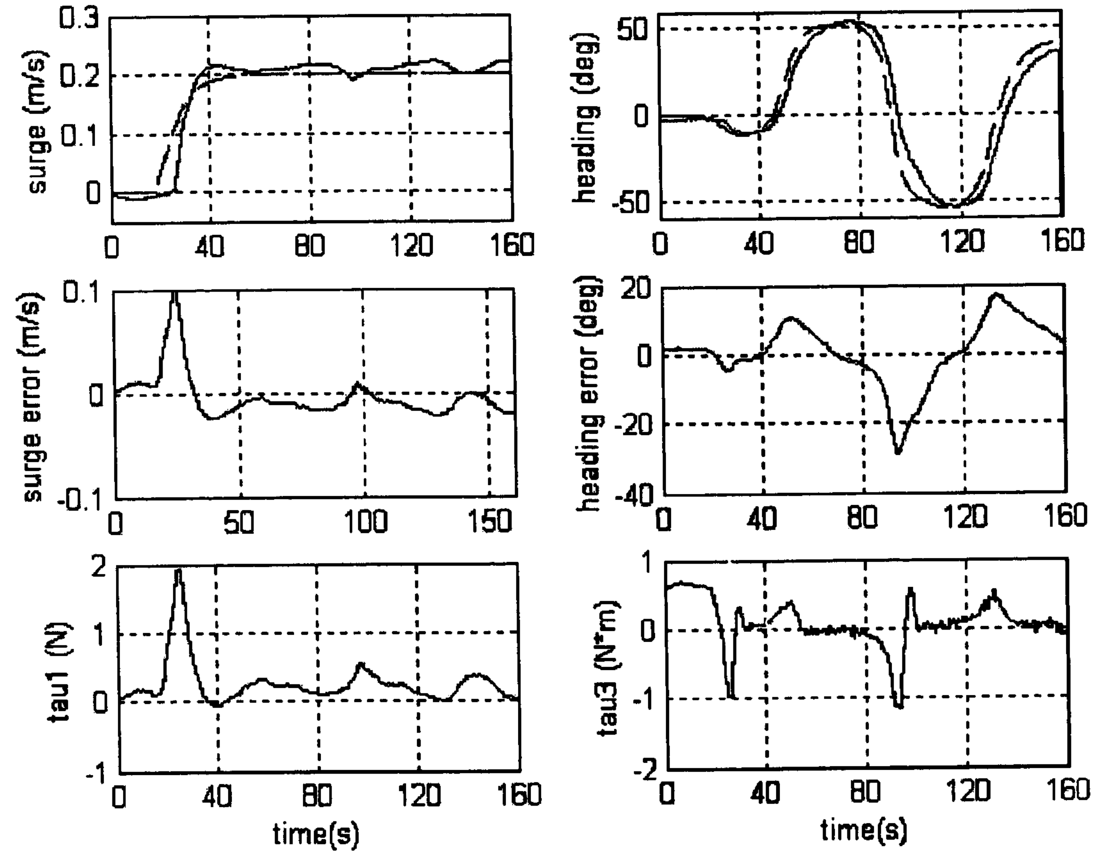


Figure 8.10: Real Results of the GP+RG Optimisation With Waves When Manoeuvring in the Presence of Waves

Analysis of the Sizes of the Trees

In Table 8.4 the sizes of the best trees (defined as the number of internal nodes of the tree) obtained in each run for heading and propulsion control are presented. The average size of the final population and the cost value obtained during the GP+RG optimisations are shown as well:

Once more there seems to be a tendency for larger heading trees that achieve better results. However the figures from Table 8.4 are less conclusive than those obtained with the GP+RG optimisation without waves shown in Table 8.2.

Again size bloat is not an issue with these optimisation runs. Only 4 propulsion trees (20%) and 4 heading trees (20%) have grown to a number of internal nodes close to the limit (30 internal nodes).

Table 8.4: Size of Trees in the GP+RG Final Populations
Optimisation With Waves

Cost value	Run No.	Propulsion		Heading	
		Best Tree	Avg. Pop.	Best Tree	Avg. Pop.
7.63	5	8	7.88	3	3.06
12.34	8	2	2.07	26	23.46
12.90	1	30	27.63	14	17.23
18.09	4	28	25.58	30	21.65
30.52	19	1	5.57	30	23.78
47.73	20	3	2.65	10	9.42
51.55	6	2	3.83	23	19.69
53.37	12	29	26.42	15	13.06
79.15	2	7	6.84	5	5.00
95.92	16	14	18.80	2	2.03
114.01	14	26	15.68	28	25.62
120.11	7	5	4.97	16	19.34
123.78	3	12	9.89	3	3.18
132.25	13	4	5.66	5	5.13
275.93	15	17	14.32	3	3.04
285.32	11	2	2.06	3	3.02
327.72	17	3	1.19	16	15.60
328.79	10	14	13.81	1	1.02
351.56	18	2	3.15	6	5.97
370.40	9	17	17.10	3	2.97

Analysis of the Percentage of Occurrence of Functions

In Figure 8.11 the percentage of best solutions that use a certain function from the function set is reported. Looking at the percentage of function usage for the propulsion trees and comparing them with the figures obtained in the GP+RG optimisation without waves (see Figure 8.6), it can be observed that although the arithmetic functions still maintain very high percentage of usage, there is an increase in the application of the hyperbolic tangent and especially the exponential function. The reason for the use of the hyperbolic tangent could be to take advantage of its properties as a switching term to counteract the disturbances. Yet, the choice of arguments for the hyperbolic tangent function points to the hyperbolic tangent operating around its proportional area, not the switching one. Therefore, it appears that the GP+RG has favoured the hyperbolic tangent as an easier way of creating a proportional control, as opposed to the complication of having to use various arithmetic operators.

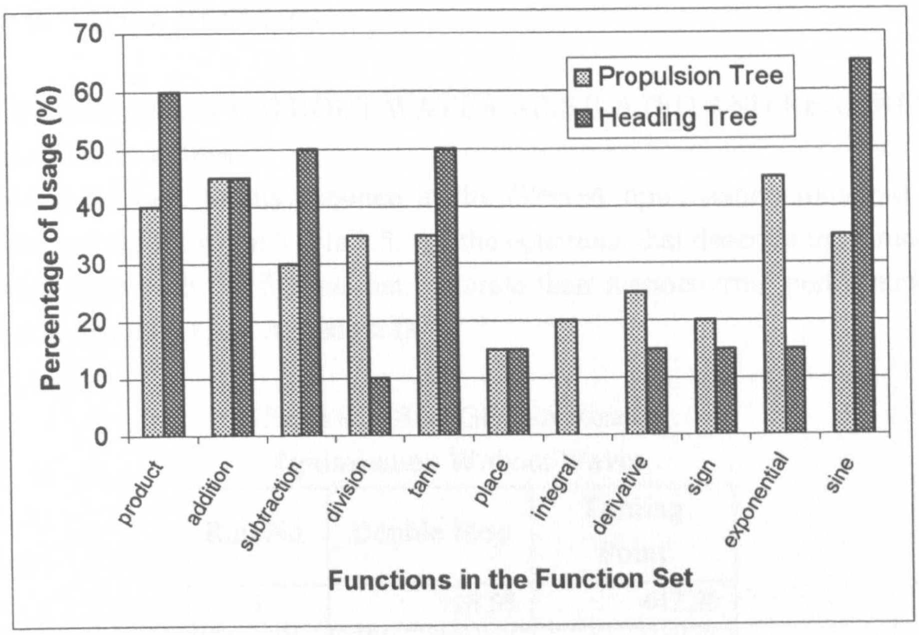


Figure 8.11: Percentage of Usage of Functions in the GP+RG Best Results Optimisation With Waves

The frequency of use of the exponential function is quite unexpected. If applied to the wrong argument it can lead to very large figures that would create problems for the controller. However, when applied to very small numbers (smaller than 0.4) $exp(\alpha)\approx\alpha+1$. Thus, the exponential function is basically adding an offset to the signal. This offset, although it would be very damaging for heading control, it is not so detrimental for the propulsion control, given the shape of the τ_{1com} signal. It can be seen from Figure 8.11 that the usage of the exponential function for the heading trees is reduced to a third of the percentage for propulsion trees, which is consistent with the previous argument.

Regarding the resulting heading trees once more, the most frequently used functions are the hyperbolic tangent and sine functions. It is also noticeable that there is a reduction in utilisation of integral and derivative terms when compared with the figures obtained for the GP+RG optimisation without waves (see Figure 8.6). As has already been mentioned, it is only the derivative function that has a significant effect on the control action since the integral terms are effectively cancelled in the evaluation of the tree. Hence the value of zero for integral terms in Figure 8.11 is more realistic than the previous results shown in Figure 8.6. On the other hand the reduction of derivative terms can be due to their detrimental noise amplification effects.

8.4. GP WITH GA RESULTS

8.4.1. OPTIMISATION WITHOUT WAVES: SIMULATED AND REAL RESULTS

Analysis of Cost Values

The cost of the best results obtained in the GP+GA optimisation runs and in the validation test are shown in Table 8.5. All the equations that describe the structure of these controllers with the figures that illustrate their manoeuvring performance can be found in Section D.3 in Appendix D.

Table 8.5: Best GP+GA Results
Optimisation Without Waves

Run No.	Double Step	Turning Point
1	126.55	447.20
2	282.61	921.02
3	195.99	1327.95
4	242.60	1026.84
5	304.71	888.67
6	293.68	1222.99
7	234.37	539.93
8	251.33	374.52
9	288.72	1226.77
10	754.28	53310.61
11	164.31	3943.42
12	372.85	1913.37
13	384.95	13818.24
14	428.94	1303.87
15	669.74	53316.41
16	541.04	53103.63
17	165.67	39279.66
18	560.23	10570.73
19	541.01	53246.64
20	541.04	57029.55

As can be seen from the results from Table 8.5 the responses are quite poor. The best cost value obtained after the GP+GA optimisation is 126.55 (run 1), nearly a hundred times bigger than the best result from the GP+RG optimisation without waves, where the run number 3 provided a cost of 1.92. However, when comparing the worst results obtained from both methods the cost values are not so different (540.99 obtained with the GP+RG scheme versus 754.28 obtained with the GP+GA). The controllers' performance in the validation test is also quite mediocre in the sense of the cost values obtained. In run number 1 the amount of convergence in the final population has been 21 individuals and the generation of convergence has been the 5th generation.

Analysis of the Structure of the Results

The general structures that have been obtained using the GP+GA approach are quite similar to those using GP+RG. In this case they can be separated into 7 groups.

1st Group

Results from runs 1, 2, 3 and 4 constitute the first group. They are characterised by using a hyperbolic tangent for the heading control and some arithmetic combination of the propulsion error, surge speed or desired surge speed for the propulsion. The controller found in run number 1 is shown as an example:

$$\tau_{1com} = \varepsilon_p + u_d \tag{8.30}$$

$$\begin{aligned}\tau_{3com} &= \varepsilon_h \cdot \tanh\left(\frac{\mathbf{h}_h' \cdot (\mathbf{x}_h - \mathbf{x}_{hd})}{\psi_d}\right) - 81.5 \cdot \tanh\left(\frac{\mathbf{h}_h' \cdot (\mathbf{x}_h - \mathbf{x}_{hd})}{3.8}\right) \approx \\ &\approx -81.5 \cdot \tanh\left(\frac{\mathbf{h}_h' \cdot (\mathbf{x}_h - \mathbf{x}_{hd})}{3.8}\right)\end{aligned} \tag{8.31}$$

As in previous cases the hyperbolic tangent is not acting as a switching term but as a proportional term. Therefore the action is equivalent to:

$$\tau_{3com} \approx -21 \cdot \mathbf{h}_h' \cdot (\mathbf{x}_h - \mathbf{x}_{hd}) \tag{8.32}$$

2nd Group

The second group is formed by controllers that use trigonometric functions for the heading control. The propulsion control uses a time function. The results from runs number 5 and 6 belong to this group. The resulting control strategy from run number 6 can be expressed as follows:

$$\tau_{1com} = \frac{96.3}{\int 20.4 \cdot dt} \tag{8.33}$$

$$\tau_{3com} = \sin(\sin(26.7) \cdot \varepsilon_h) \tag{8.34}$$

The propulsion control action is inversely proportional to the time so it decreases along the manoeuvre. Obviously this is not a very good control policy. Since $sin(26.7)=1$, the heading control is proportional to the error if the error is kept small.

3rd Group

The results from runs number 7 to 10 have in common a constant or practically constant control action for propulsion. The heading action varies among them: the results from run number 7 and 9 have two nested sine functions providing the heading control, while the result from run number 8 has a proportional term and the result from run number 9 a zero heading control action. The control structures obtained in run number 7 are as follows:

$$\tau_{1com} = \sin(\sin(1)) \approx 0.7456 \qquad (8.35)$$

$$\tau_{3com} = \sin\left(\sin\left(\frac{d(35.7 \cdot \varepsilon_h)}{dt}\right)\right) \qquad (8.36)$$

4th Group

The fourth group consists of results that rely on a hyperbolic tangent for propulsion control and some kind of proportional or derivative control for heading. Results from runs 11 to 13 belong to this group. The result from run 11 is the one that achieves better results so it is given below to demonstrate this type of structure:

$$\tau_{1com} = \exp(u_d) \cdot sign\left(h_p'(u - u_d)\right) \cdot \tanh\left(\frac{h_p'(u - u_d)}{d(\exp(u_d))/dt}\right) \qquad (8.37)$$

$$\tau_{3com} = 8.4 \cdot \varepsilon_h \qquad (8.38)$$

For the first time the hyperbolic tangent is used as a switching term, due to the small value of the derivative term in the denominator. The hyperbolic tangent term is multiplied by the sign function so effectively the sign of the derivative term determines the sign of the control action (since the exponential is always positive).

5th Group

All the structures that have a zero control action in either heading or propulsion, such as the results from runs 14, 15 or 16, belong to the fifth group. As mentioned previously this is a kind of "damage limitation" technique: a zero control action results in a better cost value than an unstable action.

6th Group

The results from runs 17 and 18 use the hyperbolic tangent function for both heading and propulsion and they form the sixth group. Their performance is poor because of the very low gain propulsion control action.

The equations that represent the control action from the trees obtained in the run number 18 of the GP optimisation are shown as an example of controller using two hyperbolic tangent functions:

$$\tau_{1com} = (1.4 + u)\cdot\tanh\left(\frac{h_p^{'}\cdot(u - u_d)}{74.8}\right) \tag{8.39}$$

$$\tau_{3com} = \varepsilon_h \cdot sign\left(\mathbf{h}_\mathbf{h}^{'}\cdot(\mathbf{x}_\mathbf{h} - \mathbf{x}_\mathbf{hd})\right)\cdot\tanh\left(\frac{\mathbf{h}_\mathbf{h}^{'}\cdot(\mathbf{x}_\mathbf{h} - \mathbf{x}_\mathbf{hd})}{\exp(-\mathbf{k}\cdot\mathbf{x}_\mathbf{h})}\right) \tag{8.40}$$
$$\mathbf{k} \equiv place(0, \quad -24.8, \quad -\psi_d)$$

Given that the exponential function is always positive, the product of the sign multiplied by the hyperbolic function is equivalent to the absolute value of the hyperbolic function. The high peaks in the exponential term imply that in this case the hyperbolic function acts some times as a proportional term and other times as a switching term. The hyperbolic tangent in the propulsion control acts as a proportional term, although the proportional gain is very small. The main problem in the propulsion case is that the sign has the wrong polarity, i.e. as the desired speed increases, the commanded control signal decreases. Therefore, the controllers presented by Equations (8.39) and (8.40) can be expressed equally as:

$$\tau_{1com} = 0.013\cdot(1.4 + u)\cdot h_p^{'}\cdot(u - u_d) \tag{8.41}$$

$$\tau_{3com} = \varepsilon_h \cdot\left|\tanh\left(\frac{\mathbf{h}_\mathbf{h}^{'}\cdot(\mathbf{x}_\mathbf{h} - \mathbf{x}_\mathbf{hd})}{\exp(-\mathbf{k}\cdot\mathbf{x}_\mathbf{h})}\right)\right| \tag{8.42}$$
$$\mathbf{k} \equiv place(0, \quad -24.8, \quad -\psi_d)$$

7th Group

Finally, the results from runs 19 and 20 are characterised by a Pole Placement control method for the heading plus a hyperbolic tangent for the propulsion. Both controllers exhibit extremely poor performance characteristics. As an example the structure of the controllers obtained in run number 20 are shown.

$$\tau_{1com} = (93.3 + u_d) \cdot \tanh\left(\frac{h'_p \cdot (u - u_d)}{17.9 + u_d} \right) \qquad (8.43)$$

$$\tau_{3com} = -\mathbf{k} \cdot \mathbf{x}_h$$
$$\mathbf{k} \equiv place\left(0, \quad -\exp(\psi), \quad -\exp(\psi) \cdot sign\left(\mathbf{h}'_h \cdot (\mathbf{x}_h - \mathbf{x}_{hd})\right)\right) \qquad (8.44)$$

Again, the heading action lacks any reference quantity and it also has a sign function associated with the location of one of the poles. The pole will be successively stable and unstable depending on the polarity of the error.

The performance of the results from run number 1 (see Equations (8.30) and (8.32)) in the optimisation process (simultaneous double step manoeuvre) is shown in Figure 8.12.

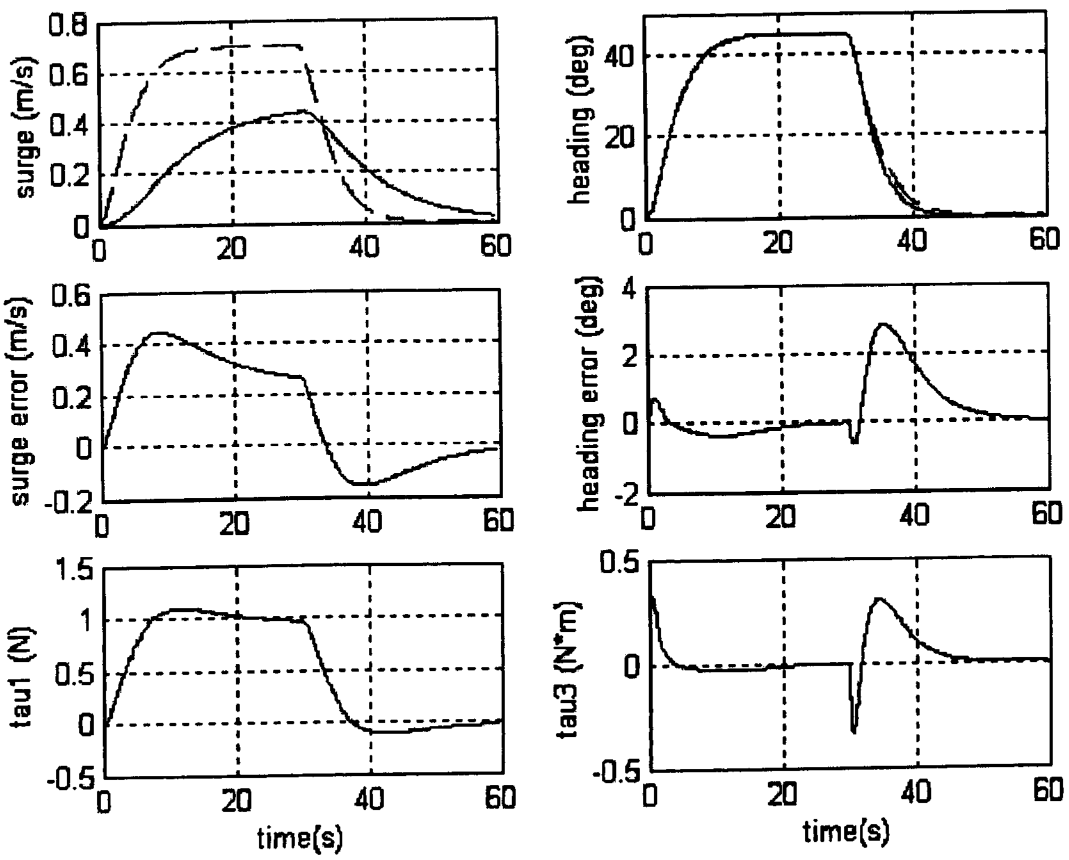


Figure 8.12: Simulated Results of the GP+GA Optimisation Without Waves

Figure 8.12 shows that although the heading control action is very good, the final cost value is degraded because of the very poor performance of the propulsion controller. The heading controller is effectively equivalent to the result obtained by the GP+RG optimisation with waves (see Equation 8.18). The propulsion control is equal to the control error plus the desired surge speed. The way the GP found of increasing the control action is to add the desired surge speed as in an open loop control. Although the surge response has got the right shape there is not enough gain.

These controllers result in the poorer tracking performance found so far in simulation either with GP+RG or GAs optimisation.

Figure 8.13 shows the simulated responses of the vessel when tracking the double turning point manoeuvre (validation test).

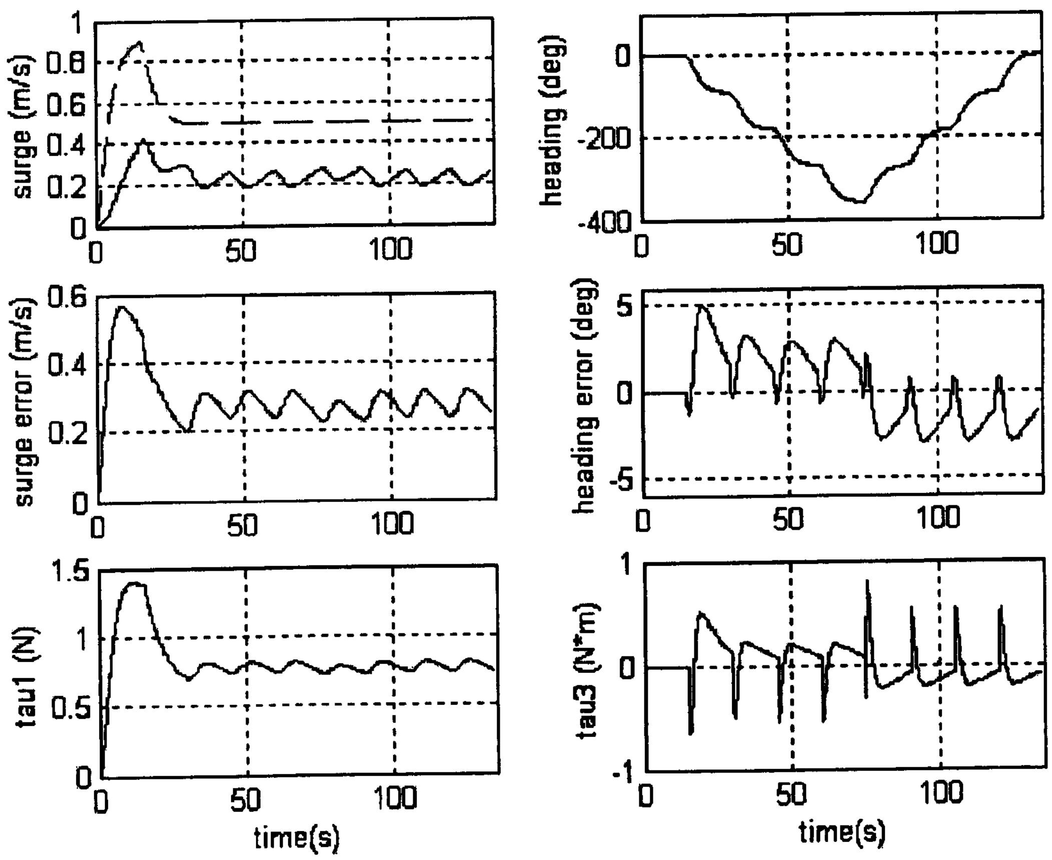


Figure 8.13: Validation Test for the Result of the GP+GA Optimisation Without Waves

Again Figure 8.13 illustrates the bad performance of the propulsion controller as opposed to the satisfactory performance of the heading controller. The propulsion commanded force is too small to keep the desired speed. It would need a higher gain. The heading response is very good and requires very little additional control effort, even though the manoeuvre is accomplish for a speed profile that is very much reduced.

Figures 8.14 and 8.15 show the responses obtained when these controllers have been implemented in the real plant. Figure 8.14 shows the results obtained when the manoeuvre was performed in calm waters.

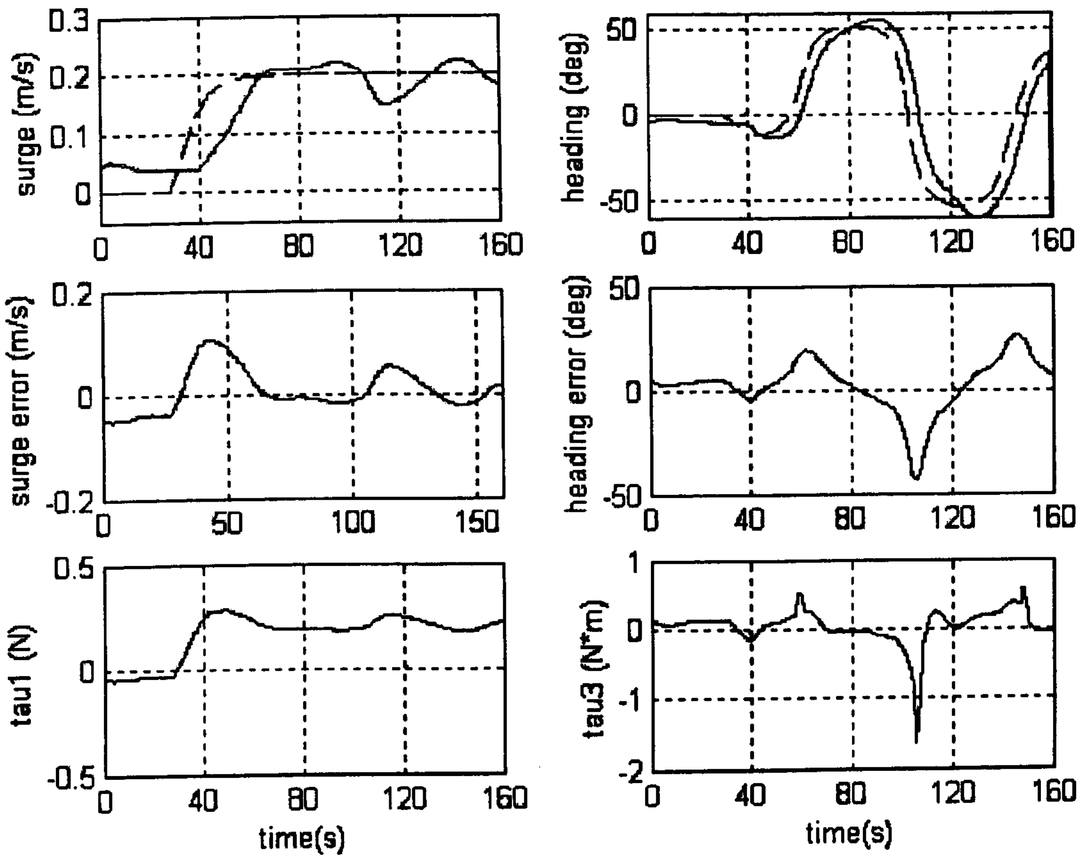


Figure 8.14: Real Results of the GP+GA Optimisation Without Waves When Manoeuvring in Calm Waters

As in the simulated results the heading controller performance is quite good while the propulsion controller performs very poorly. The gain for the propulsion control is too low at the beginning of the manoeuvre and the addition of the desired surge speed to the proportional control (see Equation (8.30)) does not allow an accurate steady state response.

Figure 8.15 shows the results obtained when the manoeuvre was performed in the presence of waves.

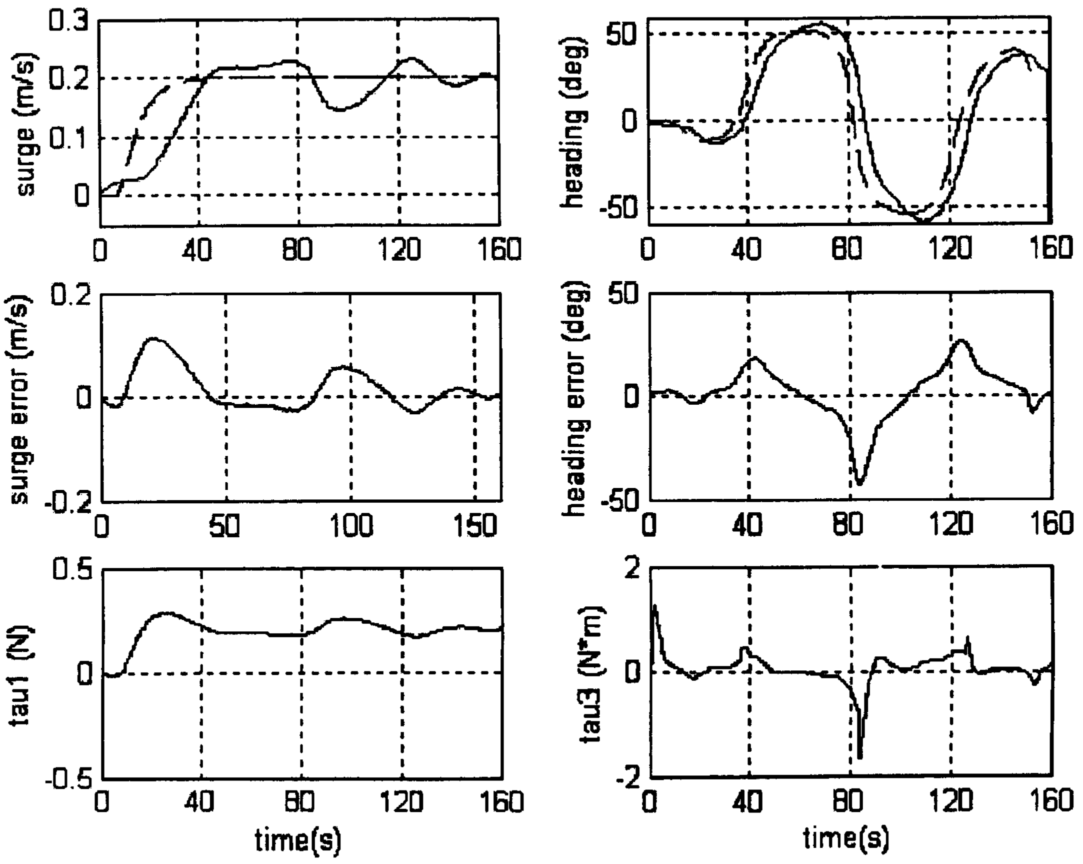


Figure 8.15: Real Results of the GP+GA Optimisation Without Waves When Manoeuvring in the Presence of Waves

Since these results have been obtained using an improved wave filter in the facilities the effect of the waves does not modify the responses significantly.

Analysis of the Sizes of the Trees

Table 8.6 presents the tree sizes (defined as the number of internal nodes) of the best trees obtained in each run as well as the average size of the trees in the final population and the cost value assigned to each result during the evaluation procedure.

In Table 8.6 it can be observed that the size of the trees is generally very small. None are close to the maximum size stipulated as a limit. This is due to the small number of generations in the structural optimisation since a considerable part of the evaluation process is consumed by the GA optimisation.

Table 8.6: Size of Trees in the GP+GA Final Populations
Optimisation Without Waves

Cost value	Run No.	Propulsion		Heading	
		Best Tree	Avg. Pop.	Best Tree	Avg. Pop.
126.55	1	1	1.10	3	6.65
164.31	11	5	7.23	1	1.10
165.67	17	13	7.32	5	5.23
195.99	3	2	1.48	5	5.52
234.37	7	4	7.97	4	3.00
242.60	4	5	5.94	8	7.74
251.33	8	1	9.58	1	5.26
282.61	2	2	4.00	5	6.61
288.72	9	3	3.35	6	5.94
293.68	6	2	4.29	3	2.97
304.71	5	3	3.74	1	2.03
372.85	12	9	7.23	3	2.06
384.95	13	5	5.29	2	2.03
428.94	14	2	3.23	3	3.00
541.01	19	7	6.87	5	3.68
541.04	16	6	10.32	14	11.03
541.04	20	4	4.23	4	4.45
560.23	18	2	3.00	4	5.16
669.74	15	1	1.00	4	1.29
754.28	10	1	1.06	1	1.03

Analysis of the Percentage of Occurrence of Functions

The subsequent statistics (Figure 8.16) reflect the percentage of usage of every function in the function set.

When looking at the percentage of occurrence of functions in Figure 8.16 and comparing those figures with the results obtained in the GP+RG optimisations (see Figures 8.6 and 8.11) it can be seen that the average percentage of occurrence of functions is lower and the percentages are more uniformly distributed. This is due to the smaller number of generations that the GP in the GP+GA scheme is run. All the functions are randomly initialised with the same probability and the structural optimisation has not been run for many generations so the resulting trees are smaller and the functions are more uniformly distributed. Thus, the percentages of occurrence of functions in the propulsion trees are quite unremarkable. Each function appears in between the 15% and the 40% of the trees (i.e. between 3 and 8 of the 20 resulting trees). The percentages of occurrence in the heading trees are also quite small. Again the most regularly used functions are the hyperbolic tangent and sine functions, as in previous heading trees.

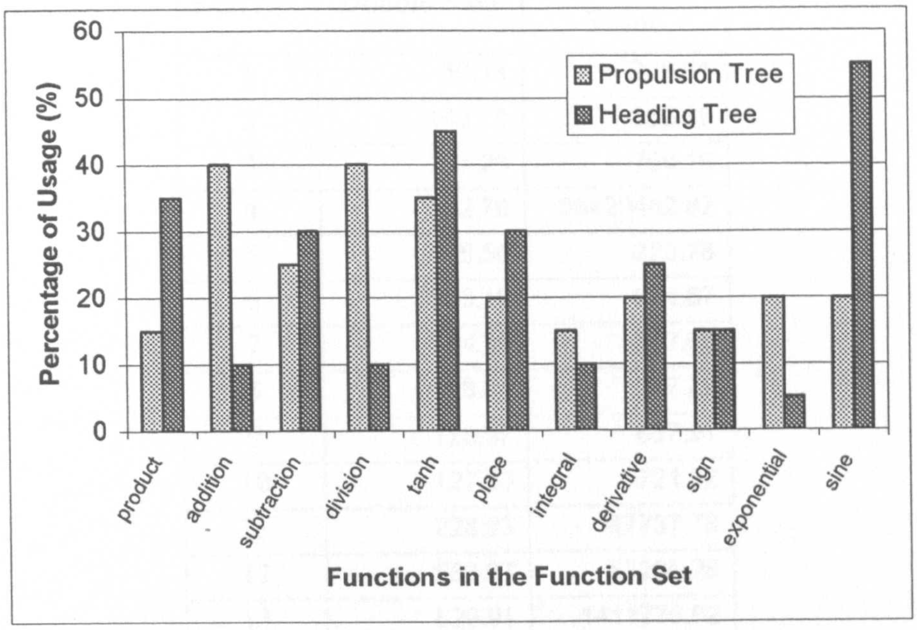


Figure 8.16: Percentage of Usage of Functions in the GP+GA Best Results Optimisation Without Waves

8.4.2. OPTIMISATION WITH WAVES: SIMULATED AND REAL RESULTS

Analysis of Cost Values

The best cost values obtained in the runs of the GP+GA optimisation with waves are shown in Table 8.7. The results from the validation test are also included. All the equations and figures that illustrate the structure and performance of these controllers can be found in Section D.4 in Appendix D.

Again the GP+GA scheme has converged to quite poor results. The convergence cost values are comparable on average with those obtained in the GP+GA optimisation without disturbances (Table 8.7). The best overall cost has been achieved from run number 1 and it is better that the best result from Table 8.5. In this run the amount of convergence in the final population has been 1 individual and the generation of convergence has been the 8th generation. Also, results from runs 2, 5 and 8 are reasonably good compared with the rest. This time the cost results obtained in the validation test and those obtained in the GP optimisation are more consistent. None of the best results in Table 8.7 fails the validation test.

Table 8.7: Best GP+GA Results
Optimisation With Waves

Run No.	Double Step	Turning Point
1	50.13	239.91
2	143.78	332.39
3	267.23	750.16
4	262.70	38420482.82
5	115.50	323.28
6	353.10	834.67
7	394.20	837.06
8	118.84	347.29
9	128.37	687.91
10	127.65	721.92
11	228.93	47737.78
12	953.27	53225.26
13	629.91	1411779.62
14	229.21	640.17
15	319.63	871.85
16	330.65	66907.90
17	530.52	1487.08
18	561.61	54399.76
19	908.00	54393.05
20	747.22	55726.58

Analysis of the Structure of the Results

Regarding their structure, the resulting control strategies could be classified into 5 categories.

1st Group

The first group comprises four of the best results obtained: results from run number 1, 2, 3 and 4. They are characterised by the use of a hyperbolic tangent function for the heading control and some arithmetic combination of the propulsion error, the desired surge speed and the actual surge speed for the propulsion. Since the best overall result has been obtained from run number 1, its structure is shown as an example of its class:

$$\tau_{1com} = 93.6 \cdot \varepsilon_p \tag{8.45}$$

$$\tau_{3com} = -57.4927 \cdot \tanh\left(\frac{\mathbf{h}_h' \cdot (\mathbf{x}_h - \mathbf{x}_{hd})}{4.0842}\right) \qquad (8.46)$$

As usual the hyperbolic tangent for the heading control acts as a proportional controller and it is equivalent to:

$$\tau_{3com} \approx -14 \cdot \mathbf{h}_h' \cdot (\mathbf{x}_h - \mathbf{x}_{hd}) \qquad (8.47)$$

2nd Group

The results obtained in runs number 5, 6 and 7 form the second group of results. They basically rely on control actions that are either proportional to the error or to the desired signal. Since the controllers that performs best are obtained from number 5, these are represented below:

$$\tau_{1com} = 2.1534 \cdot u_d \qquad (8.48)$$

$$\tau_{3com} = 2 \cdot \exp(\sin(70.2)) \cdot (2 \cdot \varepsilon_h + \sin(3.1)) \approx 7.1482 \cdot \varepsilon_h + 0.1487 \qquad (8.49)$$

The propulsion control acts as an open loop command-following type of controller; while the heading control consists of a proportional term plus a constant. This constant sets an offset that degrades the controller's performance.

3rd Group

The third group is the most numerous. All of them have a constant commanded propulsion force equal to 1 and some kind of proportional heading control action. Results from runs 8, 9, 10, 11, 12 and 13 belong to this group. Some of these results, such as those of run number 8, 9 and 10 perform satisfactorily despite the constant propulsion force. It is again a "damage limitation" issue. A constant action performs better than an unstable one. The control strategy obtained in run number 9 is presented as an example of the typical structure of this group:

$$\tau_{1com} = 1 \qquad (8.50)$$

$$\tau_{3com} = \frac{\sin(21)}{\sin(21.9)} \cdot \varepsilon_h \approx 9.1918 \cdot \varepsilon_h \qquad (8.51)$$

4th Group

The fourth set of results consists of the results from runs 14, 15 and 16. In all of them the control action is proportional to a sine function, either for the propulsion control (run number 14), for the heading control (run number 16) or for both (run number 15). In order to illustrate this kind of structure, the controllers obtained in run number 14 are shown:

$$\tau_{1com} = \sin(7.7 - u_d) \tag{8.52}$$

$$\tau_{3com} = \frac{\varepsilon_h \cdot (67.2 - \varepsilon_h)}{\psi_d + 29.3} \tag{8.53}$$

5th Group

The fifth group consists of two controllers that have a hyperbolic tangent function to provide the propulsion control and either a proportional term (result from run number 17) or another hyperbolic tangent function (result from run number 18) for the heading. Both controllers provide similar results in the GP optimisation. However, the controllers obtained in run number 17 give a reasonable result in the validation test while the other failed totally due mainly to the wrong polarity in the propulsion control. The structures of the resulting controllers from run number 17 are as follows:

$$\tau_{1com} = 0.0431 \cdot \varepsilon_p \cdot sign\left(h_p' \cdot (u - u_d)\right) \cdot \tanh\left(\frac{h_p' \cdot (u - u_d)}{\varepsilon_p / 40.8}\right) \tag{8.54}$$

$$\tau_{3com} = 9.9 \cdot \varepsilon_h \tag{8.55}$$

The propulsion control is constructed from the product of the error signal multiplied by a sign function and a hyperbolic tangent. Naturally, the effect of the sign function would be to cancel the polarity of the hyperbolic tangent function. However, the error term in the denominator of the hyperbolic tangent expression prevents a true cancellation as the equation stands. By rearranging Equation (8.52) the following relationship can be obtained:

$$\begin{aligned}\tau_{1com} &= 0.0431 \cdot \varepsilon_p \cdot sign(\varepsilon_p) \cdot \left|\tanh\left(\frac{h_p' \cdot (u - u_d)}{\varepsilon_p / 40.8}\right)\right| = \\ &= 0.0431 \cdot |\varepsilon_p| \cdot \left|\tanh\left(\frac{h_p' \cdot (u - u_d)}{\varepsilon_p / 40.8}\right)\right|\end{aligned} \tag{8.56}$$

Again the hyperbolic tangent is effectively performing as a proportional gain, therefore:

$$\tau_{1com} \approx 0.043 \cdot \left|\varepsilon_p\right| \cdot \left|\frac{h'_p \cdot (u - u_d)}{\varepsilon_p}\right| = 0.043 \cdot \left|h'_p \cdot (u - u_d)\right| \qquad (8.57)$$

Finally, two of the results (runs number 19 and 20) could not be assigned to any of the previous groups. The result from run number 19 sets to zero both control actions, and the controllers from run number 20 have a constant commanded surge force, while the heading control is provided by Pole Placement control.

The performance of the controllers considered to be best from run number 1 (see Equations (8.45) and (8.47)) is shown in Figures 8.11 and 8.12. Both figures illustrate the performance of the heading and propulsion controller when tracking the simultaneous double step and double turning point manoeuvres, respectively. The reference signals are represented in dashed line.

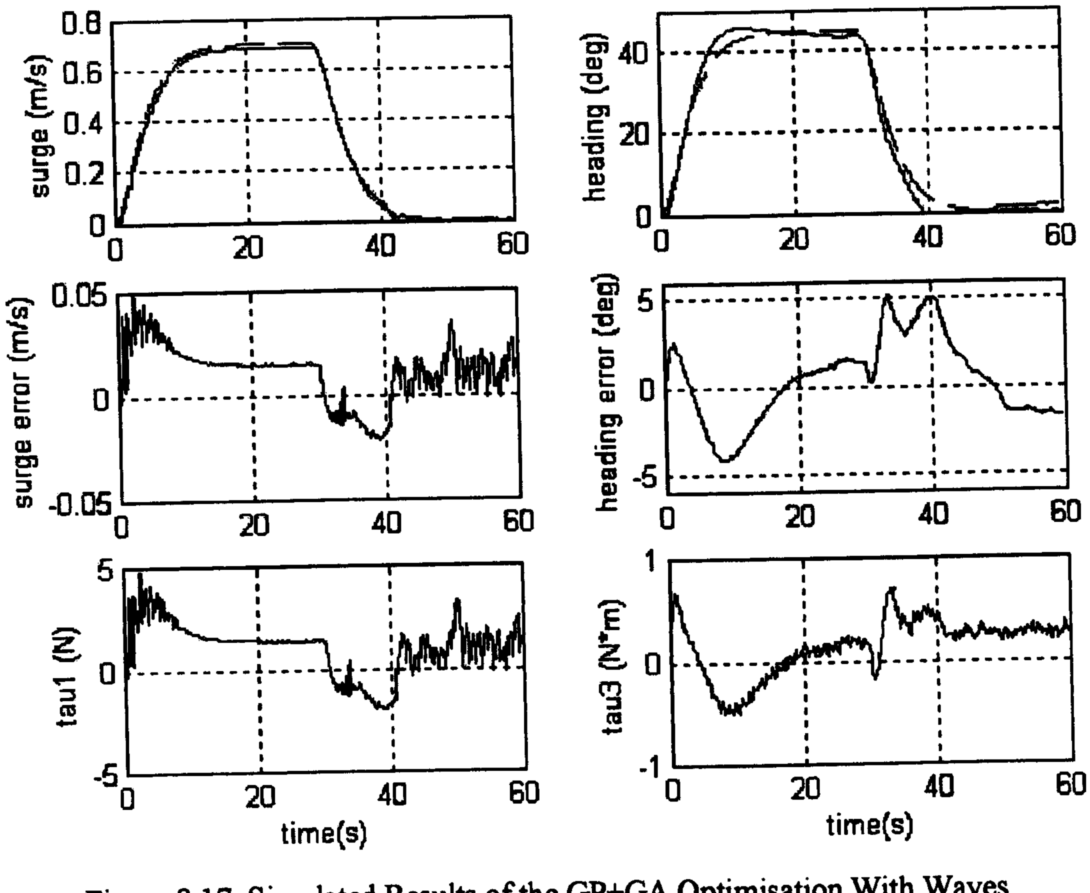


Figure 8.17: Simulated Results of the GP+GA Optimisation With Waves

It can be seen from Figure 8.17 that the performance of these controllers is better than the performance of the controllers obtained in the optimisation without waves (see Figure 8.14). The desired speed tracking is very good and the heading as well, although this response does exhibit some overshooting. The propulsion control is again a proportional controller with a gain very similar to that of the propulsion control obtained in the GP+RG optimisation without waves (see Equation 8.1). The heading controller is a proportional control whose gain is slightly smaller than those of the controllers obtained in the GP+RG optimisation with waves (see Equation 8.18) and GP+GA optimisation without waves (see Equation 8.32). The effect of the waves is mainly reflected in the rippling of the signals. While comparing this figure with the performance of the controller obtained in the GP+RG optimisation with waves (see Figure 8.7) it can be observed that, although this controller achieves a faster propulsion transient response (due to a higher gain), the control effort and the rippling in the signals is worse.

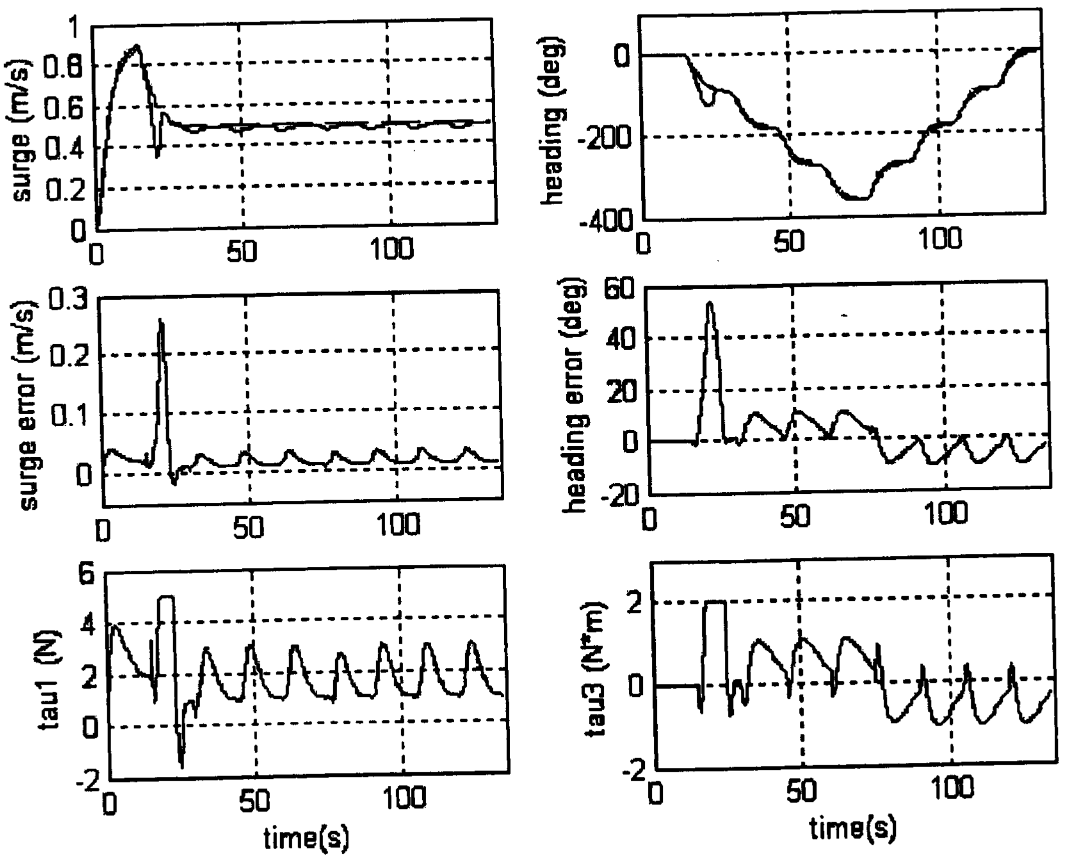


Figure 8.18: Validation Test for the Result of the GP+GA Optimisation With Waves

Again the performance of this controller is better that that of the one obtained in the optimisation without waves. Both the heading and speed tracking are reasonable. The control effort is quite high so it reaches the actuator limits. The saturation of the actuators due to the high gain causes two peaks in the error signals.

Figures 8.19 and 8.20 show the results obtained when the controllers from Equations (8.45) and (8.47) are implemented in the real plant. In Figure 8.19 the performance of the controllers operating in still waters is shown.

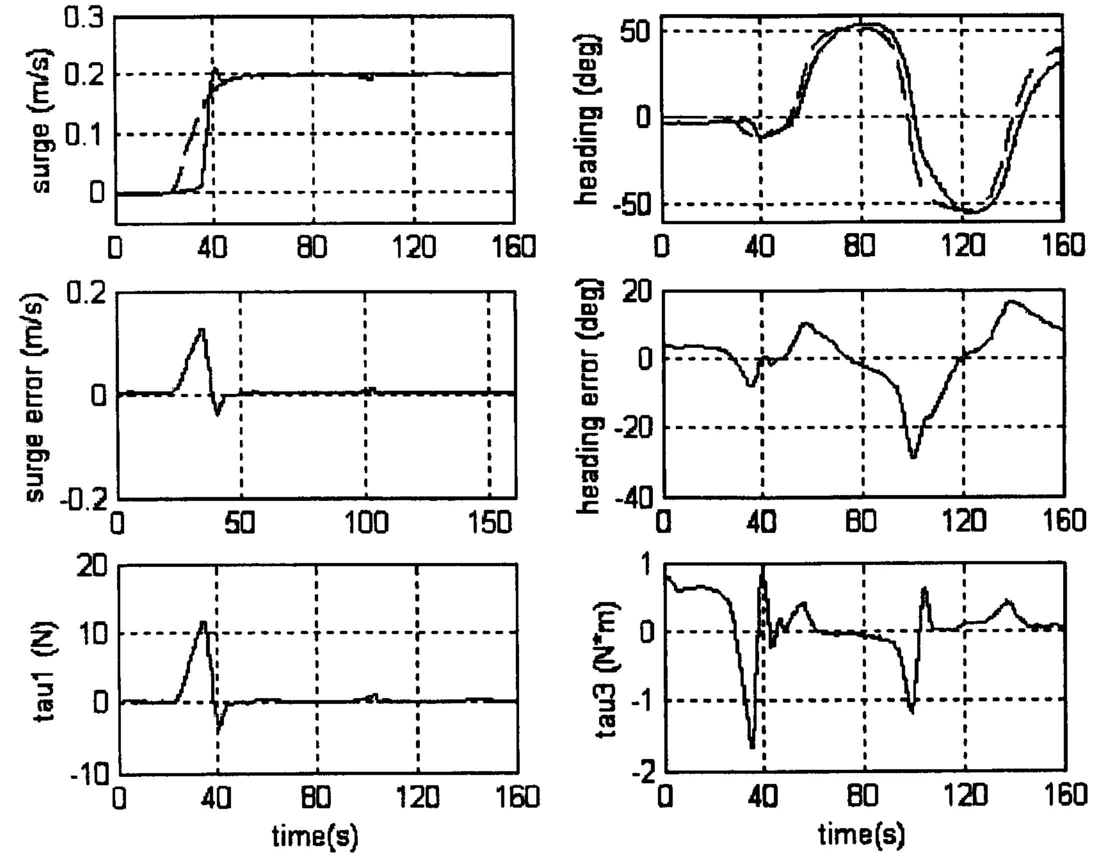


Figure 8.19: Real Results of the GP+GA Optimisation With Waves When Manoeuvring in Calm Waters

The real performance of the controllers shown in the previous figure presents similar characteristics to the simulated responses from Figure 8.17, mainly the overshooting in the heading response and the slight steady-state error in the surge response.

Figure 8.20 shows that the inclusion of waves in the water tank does not affect significantly the performance of the heading and propulsion controllers obtained in the GP+GA optimisation with waves.

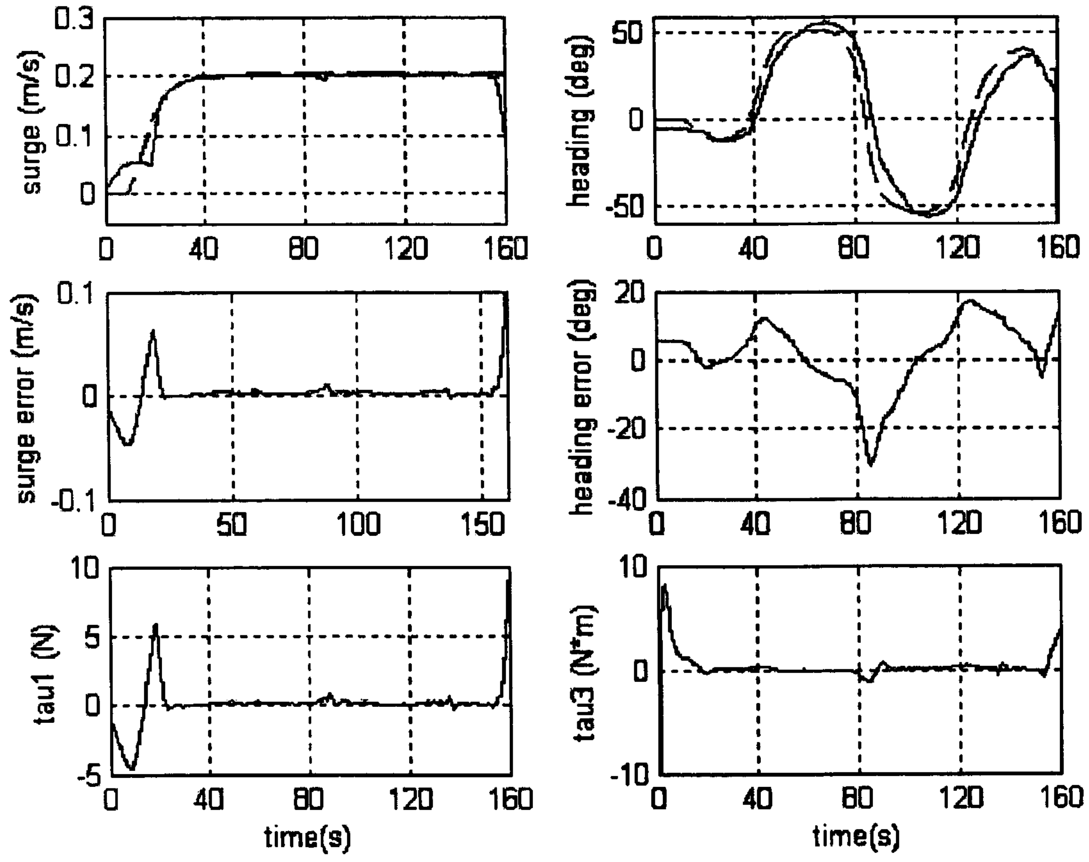


Figure 8.20: Real Results of the GP+GA Optimisation With Waves When Manoeuvring in the Presence of Waves

Analysis of the Size of the Trees

In order to analyse the tree size aspect of the optimisation Table 8.8 includes the sizes of the heading and propulsion trees found to be best in the GP+GA optimisation. The average size of the final population of each run is also included together with the cost values assigned to every solution during the evaluation procedure.

Yet again, the small number of GP generations run in the GP+GA scheme has led to very small resulting trees. Still, it can be noticed that the best results use bigger trees for heading than for propulsion.

Table 8.8: Size of Trees in the GP+GA Final Populations Optimisation With Waves

Cost value	Run No.	Propulsion		Heading	
		Best Tree	Avg. Population	Best Tree	Avg. Population
50.13	16	1	1.06	5	3.71
115.50	10	2	6.90	9	7.26
118.84	20	3	2.97	5	4.94
127.65	14	5	7.35	7	4.90
128.37	11	1	1.16	4	3.42
143.78	5	2	2.23	12	11.84
228.93	13	5	2.74	6	8.65
229.21	4	2	4.55	5	4.61
262.70	3	1	1.13	9	5.45
267.23	9	2	2.00	2	2.16
319.63	2	2	2.52	1	1.00
330.65	6	1	1.52	3	2.77
353.10	8	1	1.06	1	1.06
394.20	15	4	4.00	2	2.00
530.52	18	7	7.94	1	3.48
561.61	19	1	1.16	4	4.16
629.91	12	3	11.03	3	2.23
747.22	7	5	4.29	1	1.13
908.00	1	5	4.39	6	5.87
953.27	17	5	5.26	9	5.45

Analysis of the Percentage of Occurrence of Functions

Finally, in Figure 8.21 the percentage of usage of each function from the function set can be seen. It has been calculated considering only the best heading and propulsion trees obtained in each run.

Once more the GP+GA optimisation produces smaller percentages of occurrence of functions in the resulting trees as a consequence of the smaller size of the trees and the lesser degree of structural development.

Among the propulsion trees the most frequent functions are the arithmetic functions and the place command. As in the case of the integral term for heading control, the place commands are cancelled during the evaluation of the tree and they do not influence the final control action.

As usual, the most regular function occurrence for the heading trees is that of the sine and hyperbolic tangent functions.

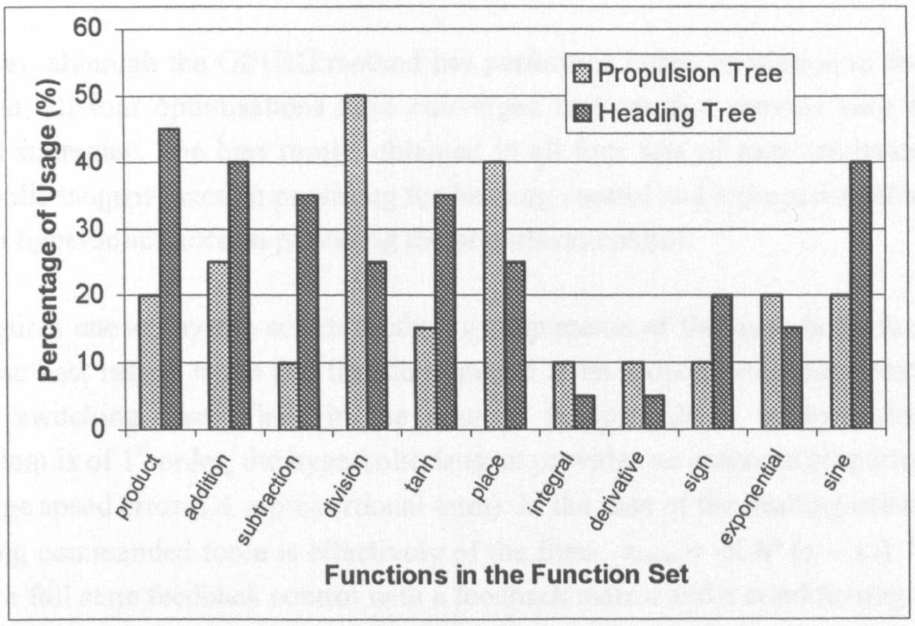


Figure 8.21: Percentage of Usage of Functions in the GP+GA Best Results Optimisation With Waves

8.4. SUMMARY

Table 8.9 summarises the cost values obtained in the optimisation with and without waves of the best structures obtained with GP+RG and GP+GA:.

Table 8.9: Summary of Cost Values

	GP+RG	GP+GA
w/o waves	2.44	126.55
with waves	12.90	50.13

As it can be seen in the table the results obtained in the GP optimisation study are quite satisfactory. The best cost values reached with the GP+RG scheme outperform the cost values obtained in the GA optimisation of control strategies such as Sliding Mode control or H_∞ in the optimisation without waves and are equivalent to the costs obtained by Sliding Mode or Pole Placement in the optimisation with waves. The manoeuvring performance of these controllers illustrated in the figures also proves their adequacy.

Conversely, the results obtained with the GP+GA implementation are very poor in comparison. The use of evaluation in the GA optimisation to improve the parameter tuning does not pay off.

However, although the GP+RG method has performed better in relation to the costs obtained, all four optimisations have converged to trees that provide very similar control strategies. The best results obtained in all four sets of runs are based on a hyperbolic tangent function providing the heading control and a proportional term or another hyperbolic function providing the propulsion control.

The figures chosen by the search method as arguments of the hyperbolic functions for these best results make this function operate in its proportional range instead of in the switching area. Thus, in the case of the propulsion control, since the subsystem is of 1st order, the hyperbolic tangent provides an outcome proportional to the surge speed error (i.e. a proportional term). In the case of the heading control, the resulting commanded force is effectively of the form: $\tau_{3com} \approx -\mathbf{K}\cdot\mathbf{h}'\cdot(\mathbf{x} - \mathbf{x}_d)$. This is in fact a full state feedback control with a feedback matrix and a conditioning matrix for the state reference equal to $\mathbf{K}\cdot\mathbf{h}'$.

Regarding the effect of the inclusion of waves in the evaluation of the candidate solutions, the results from the GP+RG optimisation are consistent with the conclusion drawn in Chapter 6. The inclusion of waves leads to controllers that reduce the control gain more than improve the robustness against external disturbances. The results obtained in the GP+GA optimisation are so poor (especially those from the optimisation without waves) that they do not allow clear conclusions.

On the subject of the size of the resulting trees the size figures shown are very encouraging since growth of trees to an extreme size is not an issue in this study. In the case of the GP+GA runs this has been expected since many of the evaluations are dedicated to GA optimisation that does not affect the tree size. However, the small size of the resulting trees in the GP+RG optimisation was not expected. Only around 20% of the trees are close to the maximum size limit. One of the theories to explain bloat states that the reason for the code bloat in GP is a bias towards deeper subtree crossover and subtree mutation points. Hence, the way the tree structure has been defined in this study, by not allowing subtree mutation or subtree crossover to act over terminal nodes can be viewed as hampering the tree growth through balancing the bias.

As a final point, the study of occurrence of functions in the resulting trees indicates a preference for the use of hyperbolic tangents and sine functions (apart from the four basic arithmetic operators) over other functions for both propulsion and heading. This kind of study is not very conclusive since the occurrence of a function in a tree does not ensure its contribution to the result, but it helps in the analysis, especially if they are consistent with those of the structural analysis.

CHAPTER 9

CONCLUSIONS AND FURTHER WORK

9.1. SUMMARY OF THE CONCLUSIONS

The overall conclusion to be drawn from this work is that in the best cases genetic optimisation techniques have performed very satisfactorily when they have been used to optimise the control strategy for the propulsion and heading dynamics of a supply vessel model. They have converged to controller solutions that provide a good tracking of the desired responses while minimising actuator usage.

In this study, GAs optimisation has been based on existing time domain control structures such as PID, Pole Placement, Sliding Mode and H_∞. GAs provides an efficient way of tuning the parameters that define these structures. However, the rigid solution encoding in GAs make them unfriendly to use as optimiser of hierarchic structures. The tree representation typical of GP provides the flexibility necessary to conduct the hierarchic search. In GP, the size and shape of the solution trees (randomly initialised), is allowed to evolve along the generations.

Table 9.1 presents the cost values obtained in the different optimised control structures:

Table 9.1: Summary of Cost Values

	PID	PP	SM+PI	SM	H_∞ (GA)	H_∞ (sGA)	GP+RG	GP+GA
w/o waves	1.2	0.96	4.5	4.5	1.6	1.6	2.44	126.55
with waves	4.3	10.6	10.5	11.5	33.4	11.77	12.90	50.13

From the table it can be seen that the best cost values for the simulated results have been obtained with PID and Pole Placement controllers. However, the real tests sometimes give a significantly different result compared with that expected by the simulation results. Thus, overall, SM optimised without waves provided a very consistent good performance, despite of being outperformed by other controllers in the simulations. In addition, the results obtained in the GP optimisation study are quite satisfactory. The best cost values reached with the GP+RG scheme outperform the cost values obtained in the GA optimisation of control strategies such as Sliding Mode control or H_∞ in the optimisation without waves and are equivalent to the costs obtained by Sliding Mode or Pole Placement in the optimisation with waves. The results obtained in the real testing of the GP+RG controller structure optimised without waves are also consistent with the good performance shown in the simulations.

The conclusions derived from the analysis of the various genetic models are presented below

9.1.1. CONTROLLERS

In Chapter 4 the design and implementation of four types of controllers for the heading and propulsions dynamics of the supply ship has been presented. As an introduction to the design ICAD was used for the analysis of the level of decoupling of the system. ICAD illustrated the strong coupling between the sway and heading dynamics. It also determined that the coupling between the surge and heading dynamics is null once the system is linearised, but it did not provide a result in terms of the level of coupling due to the non-linearities in the model. This is an important limitation of this approach.

Two of the control designs, namely PID and Sliding Mode, were based on two decoupled subsystems, while Pole Placement and H_∞ were based on the MIMO system. The MIMO controllers did not show a clear improvement of performance over the decoupled ones. Therefore, it indicates that the level of coupling is not big enough to degrade the performance of decoupled controllers, i.e. they are well-suited to the dynamics of the system.

The manual tuning of the controllers in Chapter 4 has shown the inherent difficulties associated with this process. Sliding Mode and H_∞ have proven to be especially demanding. Even for easier tuning problems, such as PID and Pole Placement, the process is very tedious. H_∞ control has the added difficulty that the elements of the controller subjected to tuning (i.e. the weighting functions) are not parameters but transfer functions. Therefore the structure of the transfer functions has to be chosen *a priori*. Also, the tuning is not very intuitive, since the transfer functions are not related to specific characteristics of the controller, as opposed to PID that has clear performance rules associated with each gain.

9.1.2. GENETIC ALGORITHMS

The results obtained with GAs (see Chapter 6) have illustrated the advantages of using an optimisation technique for the purpose of the adjustment of the controller's parameters. One of the key aspects of GAs is that they do not require any *a priori* knowledge of the search space; therefore they are very easily applied to the tuning of different controllers. A change in the controller to be optimised only involves minor modifications to the encoding of the solutions due to the different number of parameters to optimise in each structure.

Various GA schemes were analysed and their performance compared in Chapter 5 in order to find a GA scheme that was well suited to the controller optimisation problem for ship navigation. In the study, tournament selection with high mutation and crossover rates outperformed other selection techniques such as ranking or elitism. In addition, the inclusion of the non-uniform mutation operator was found to be beneficial, especially for the H_∞ optimisation problem.

All the resulting controllers from the GA optimisation have been tested in the real plant and the results have been shown in Chapter 6. The performance of the controllers once implemented in CS2 has been satisfactory. Overall, the Sliding Mode optimised without waves provided a consistently good performance.

Some controllers (namely the PID and PI controllers for propulsion in the decoupled PID control and the Sliding Mode plus PI configuration optimised with waves, and the manually tuned and optimised using sGA H_∞ controller) have shown stability problems when employed to control the real plant. In the simulation studies there was no evidence of this problem so the GA was misled by the good cost values obtained by these controllers in the evaluation process. This emphasises the importance of accurate models for controller design and optimisation. However, it is important to remark that other control structures, namely Sliding Mode and Pole Placement, were also subjected to these inaccuracies in the model and their performance was not affected (i.e. they show robustness against model uncertainties). This was to be expected since Sliding Mode is considered to be a robust controller and Pole Placement was design using a robust eigenstructure assignment method.

When the controllers optimised without waves are tested in the presence of waves, the responses become noisy, although the tracking performance is not so much affected. This high frequency action is especially manifest in the propulsion control signal (τ_l). On the other hand, the inclusion of environmental disturbances in the optimisation process did not result in the desired robustness against disturbances of the controllers. The results obtained in the real trials conducted in the water tank allow us to conclude that the effect of including waves in the optimisation is not an increase of the robustness of the controllers against external disturbances but a reduction in the control effort. This is especially true for the propulsion signals. Since the addition of waves causes this very noisy τ_l signal, the GA tries to compensate for it by reducing the gain of the controller. This leads to a degraded surge tracking, but it effectively reduces the oscillations. Thus, the inclusion of disturbances represents good practice when looking for smooth actuator usage or non-reactive low gain control.

Regarding the consistency of the GA convergence to certain type of solutions, the various runs of the GA optimisation led to similar controller solutions for all the controllers except H_∞ (as can be seen in the small standard deviation values obtained when these solutions were averaged in Chapter 6). The diversity of solutions for the GA optimisation of H_∞ provided further proof of the difficulty of this optimisation problem. This can also be notice in the generation of convergence of the optimisations. While the other controllers found solutions closer to the best one within the first ten generations, the GA did not converge to the final solution for H_∞ until the very last generations.

The Structured Genetic Algorithm (sGA) performed very well in the optimisation of the H_∞ weighting functions. The resulting cost values were smaller or equivalent to those obtained with the GA optimisation, but with the added advantage of reducing the order of the weighting functions and, as a result, the order of the controller. The sGA results show that the method can be of assistance when identifying appropriate weighting functions orders. However, once the controllers were implemented in the real plant they proved to have stability problems.

9.1.3. GENETIC PROGRAMMING

As described in Chapter 7, GP provides structural solutions for controllers. In Chapter 8 the GP provided controller structures for heading and propulsion dynamics that performed satisfactorily. Some of the cost values obtained outperformed or equalled those obtained with well-established control structures such as Sliding Mode or H_∞.

The degree of *a priori* knowledge needed to use GP has been found to be less than GAs, since the size and shape of the solutions are not determined by the user but are dynamically evolved along the generations.

An important issue when working with GP is how to generate the numerical constants needed to find a solution for most problems. Evidence has been given in Chapter 8 that the GP scheme with random generation of constants provides better results than the combination of GP with a parametric optimisation technique such as GAs when using the same number of evaluations of candidate solutions. The investment of evaluations in the GA optimisation at the expense of the GP optimisation in order to improve the parameter tuning does not pay off.

Regarding the inclusion of waves in the optimisation process in GP, the results corroborate the conclusions from Chapter 6: the inclusion of waves results in controllers with reduced control effort.

All four GP optimisations (i.e. GP with random generation with and without waves and GP in combination with GAs with and without waves) have converged to very similar control structures. The propulsion control is always provided by a proportional term. This can be built either as a gain multiplying the surge error signal or as a hyperbolic tangent of the surge error signal acting in its proportional range. The heading control always relies on a hyperbolic tangent acting in its proportional range. Therefore, it is equivalent to a state feedback control matrix (**k**) acting as well as the conditioning matrix for the reference signal (i.e. $\tau_{3com} = -\mathbf{k} \cdot (\mathbf{x_h} - \mathbf{x_{hd}})$).

Finally, the GP optimisation has not shown the size problems usually reported as one of the main flaws of the method. This is thought to be due to the way the tree structure has been implemented in Matlab in this study. By not allowing subtree mutation or subtree crossover to act over terminal nodes, the genetic operators are restricting the tree growth through balancing the bias towards deeper subtree crossover and subtree mutation points existing in the genetic operators.

9.2. FURTHER WORK

The work presented in this thesis can be extended in several directions. Some possible areas of extension are given below.

9.2.1. CONTROL APPLICATIONS

From the point of view of control, a continuation of this work could be the study of the effect in the optimisation of other disturbances such as wind or ocean currents. Currently, wind and current generation facilities are under construction in the MCLab, this would allow real testing of the effect of these disturbances.

Also, current work on CyberShip II is based on dynamic position controllers [Lindegaard (2003)], and using GA for optimisation of the controllers would be an interesting option.

9.2.2. GENETIC ALGORITHMS

Regarding the evaluation of the controller solution in the GP and GA optimisations the use of multi-objective GA (MOGA) [Fonseca and Fleming (1994), (1998a), (1998b)] could be contemplated. The cost function considered in this study had 6 terms, reflecting the objectives of the control strategy: good tracking, minimisation of actuator effort and minimisation of oscillations in the actuators for both propulsion and heading control. The optimised solution is a trade-off between these objectives. MOGA provides a solution that consists of a family of solutions (the *Pareto optimal set*) [Fonseca and Fleming (1994)]. Any improvement in one objective is at the expense of the other.

Some work has been done in the use of GAs as an on-line optimisation technique for route planning of AUVs [Alfaro-Cid *et al.* (2003)]. Given the long optimisation time usually required for GA optimisations, the number of evaluations has to be reduced for on-line applications. For an effective use of a GA with a small number of solution evaluations the size of the search space has to be restricted. The reduction of the search space can be done easily in the optimisation of the controller parameters. Every time, the GA is run in conjunction with the manoeuvre of the vessel, and only a search space around the current parameter values is considered. The increasing computer capabilities allow the use of GAs on-line as an adaptive type of controller. The on-line application of GAs would be especially interesting for supply ships, as it would allow a smooth transition from course-changing operation to dynamic positioning.

9.2.3. GENETIC PROGRAMMING

Regarding the GP performance, although the decoupled controllers have proven to be very efficient for the control of this particular plant, the GP implementation could be upgraded to a MIMO controller having a single tree with three roots for τ_{1com}, τ_{2com} and τ_{3com}. This would provide a higher level of system integration. In the same way the PID and Sliding Mode controllers optimised with GA could also be designed as MIMO controllers.

The non-uniform mutation operator, which has worked so well in GA, could be adapted for the GP algorithm. A bias could be included so that the mutation varies depending on the depth of the node affected (since the deeper a node is the less relevant for the final solution) and the evolution stage (i.e. the number of generation).

The limited amount of bloat encountered in the GP optimisations encourages further analysis of the tree representation and operators used to get a better understanding of the bloat dynamics. A deeper analysis of the crossover operator is required together with a study of the evolution of the size in combination with the evolution of the fitness and the levels of convergence among the trees of the population.

Regarding the selection of numerical constants it would be interesting to use other parametric optimisation methods such as Simulated Annealing or hill-climbing to see if they provide better results than GAs or random generation of constants.

REFERENCES

Abdel-Magid, Y.L., Bettayeb, M. and Selim, S.Z. (1997), "Power System Output Feedback Stabilizer Design via Genetic Algorithms", *Proceedings of the IEE International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA '97)*, Glasgow (UK), pp 56-62

Ackermann, J. (1972), "Der Entwurf Linearer Regelungssysteme im Zustandsraum", *Regelungstech Prozess-Datenverarb*, Vol. 7, pp 297-300

Ahmad, M., Zhang, L. and Readle, J.C. (1997), "On-Line Genetic Algorithm Tuning of a PI Controller for a Heating System", *Proceedings of the IEE International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA '97)*, Glasgow (UK), pp 510-515

Alfaro-Cid, E. (2002), "Genetic Algorithm Optimisation of an H_∞ Controller for an Oil Platform Supply Ship", *UKACC Postgraduate Symposium on Control (UKACC '02)*, Sheffield (UK), pp 137-142

Alfaro-Cid, E. and McGookin, E.W. (2001), "Genetic Algorithm Optimisation of Oil Tanker Control Systems", *Proceedings of the IFAC Conference on Control Applications in Marine Systems (CAMS '01)*, Glasgow (UK), pp 227-32

Alfaro-Cid, E., McGookin, E.W. and Murray-Smith, D.J. (2001a), "Genetic Algorithm Optimisation of a Supply Ship Propulsion and Navigation Systems", *Proceedings of the MTS/IEEE Oceans Conference*, Honolulu (USA), pp 2645-2652

Alfaro-Cid, E., McGookin, E.W. and Murray-Smith, D.J. (2001b), "Genetic Algorithm Optimisation of a Ship Navigation System", *Acta Polytechnica*, Vol. 41, No. 4-5, pp 13-19

Alfaro-Cid, E., Loo, M., Mitchell, A. and McGookin, E.W. (2003), "AUV Route Planning Using Genetic Algorithms", *1st IFAC Workshop on Guidance and Control of Underwater Vehicles (GCUV2003)*, Newport (UK), pp 95-100

Alvarez, A. and Caiti, A. (2001), "A Genetic Algorithm for Autonomous Underwater Vehicle Route Planning in Ocean Environments with Complex Space-Time Variability", *Proceedings of the IFAC Conference on Control Applications in Marine Systems (CAMS '01)*, Glasgow (UK), pp 237-242

Andry, A.N., Shapiro, E.Y. and Chung, J.C. (1983) "Eigenstructure Assignment for Linear Systems", *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 19, No. 5, pp 711-729

Angeline, P.J. (1994), "Genetic Programming and Emergent Intelligence", *Advances in Genetic Programming I*, MIT Press, Chapter 4, pp 75-97

Angeline, P.J. (1997), "Comparing Subtree Crossover with Macromutation", *EP97 Lecture Notes in Computer Science*, Vol. 1213, pp 101-110

Anonymous (2002), "Standards for Ships Manoeuvrability", *Resolution MSC.137(76)*

Astrom, K. and Hagglund, T. (1984), "Automatic Tuning of Simple Regulators with Specifications on Phase and Amplitude Margins", *Automatica*, Vol. 20, No 5, pp 645-651

Astrom, K. and Hagglund, T. (1995), *PID Controllers: Theory Design and Tuning*, Instrument Society of America, Research Triangle Park, North Carolina (USA)

Astrom, K.J. and Kallstrom, C.G. (1976), "Identification of Ship Steering Dynamics", *Automatica*, Vol. 12, pp 9-22

Back, T. (1996), *Evolutionary Algorithms in Theory and Practice*, Oxford University Press, New York

Bartolini, G., Ferrara, A. and Usai, E. (1998), "Chattering Avoidance by Second-Order Sliding Mode Control", *IEEE Transactions on Automatic Control*, Vol. 43, No. 2, pp 241-246

Bastian, A. (2000), "Identifying Fuzzy Models Utilizing Genetic Programming", *Fuzzy Sets and Systems*, Vol. 113, pp 335-350

Bennet, S. (1996), "A Brief History of Automatic Control", *IEEE Control Systems Magazine*, Vol. 16, No. 3, pp 17-25

Bhattacharya, M. and Nath, B. (2001), "Genetic Programming: A Review of Some Concerns", *ICCS2001 Lecture Notes in Computer Science*, Vol. 2074, pp 1031-1040

Brindle, A. (1981), *Genetic Algorithms for Function Optimization*, PhD Thesis, University of Alberta, Edmonton (Canada)

Brooks, R.R., Iyengar, S.S. and Chen, J. (1996), "Automatic Correlation and Calibration of Noisy Sensor Readings using Elite Genetic Algorithms", *Artificial Intelligence*, Vol. 84, pp 339-354

Busch, J., Ziegler, J., Aue, C., Ross, A., Sawitzki, D. and Banzhaf, W. (2002) "Automatic Generation of Control Programs for Walking Robots Using Genetic Programming", *EuroGP2002 Lecture Notes in Computer Science*, Vol. 2278, pp 258-267

Chalkiadakis, I., Rovithakis, G. and Zervakis, M. (2001), "A Structural Genetic Algorithm to Optimise High Order Neural Network Architecture", *Proceedings of the European Symposium on Artificial Neural Networks*, Bruges (Belgium), pp 185-192

Chellapilla, K. (1997), "Evolving Computer Programs Without Subtree Crossover", *IEEE Transactions on Evolutionary Computation*, Vol. 1, No. 3, pp 209-216

Chen, C.L. and Chang M.H. (1998), "Optimal Design of Fuzzy Sliding-Mode Control: A Comparative Study", *Fuzzy Sets and Systems*, Vol. 93, No. 1, pp 37-48

Chipperfield, A. and Fleming P. (1995), "Genetic Algorithms in Control Systems Engineering", *Control and Computers*, Vol. 23, No. 3, pp 88-94

Chipperfield, A. and Fleming P. (1996), "Multiobjective Gas Turbine Engine Controller Design Using Genetic Algorithms", *IEEE Transactions on Industrial Electronics*, Vol. 43, No 5, pp 583-587

Chipperfield, A., Fonseca, C.M., Betteridge, H.C. and Fleming, P.J. (1999), "Design of a Wide Envelope Controller for a STOVL Gas Turbine Engine", *Proceedings of the IFAC 14th Triennial World Congress*, Beijing (China), pp 479-484

Cho, H.J., Cho, K.B. and Wang, B.H. (1997), "Fuzzy-PID Hybrid Control: Automatic Rule Generation Using Genetic Algorithms", *Fuzzy Sets and Systems*, Vol. 92, No. 3, pp 305-316

Chouiter, D.R., Clerc, G., Auriol, P. and Retif, J.M. (1999), "On the Robust Control of an Induction Machine: A Complete Design and Realization", *European Physical Journal of Applied Physics*, Vol. 6, No. 1, pp 61-70

Chowdhury, M.M.M. and Li, Y. (1996), "Messy Genetic Algorithm Based New Learning Method for Structurally Optimised Neurofuzzy Controllers", *Proceedings of the IEEE International Conference on Industrial Technology*, Shangai (China), pp 274-278

Christiansson, A. and Lennartson, B. (1999), "Weight Selection for H_∞ Control using Genetic Algorithms", *Proceedings of the IFAC 14th Triennial World Congress*, Beijing (China), pp 25-30

Clarke, T. and Davies, R. (1997), "Robust Eigenstructure Assignment Using the Genetic Algorithm and Constrained Feedback", *Proceedings of the Institution of Mechanical Engineers*, Part I, Vol. 211, pp 53-61

Corneliussen, J. (2003), *Implementation of a Guidance System for CyberShip II*, Master Thesis, Norwegian University of Science and Technology, Trondheim, (Norway)

Dakev, N.V., Whidborne, J.F., Chipperfield, A. and Fleming P. (1997), "Evolutionary H_∞ Design of an Electromagnetic Suspension Control System for a Maglev Vehicle", *Proceedings of the Institution of Mechanical Engineers*, Part I, Vol. 211, pp 345-355

Darwin, C. (1859), *On the Origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life*, John Murray, London

Dasgupta, D. (1994), "Handling Deceptive Problems Using a Different Genetic Search", *Proceedings of the IEEE International Conference on Computational Intelligence*, Florida (USA), pp 807-811

Dasgupta, D. and McGregor D.R. (1991), "A Structured Genetic Algorithm: The Model and the First Results", *Technical Report No. IKBS-2-91.ps.Z*, Dept. of Computer Science, University of Strathclyde (UK)

Dasgupta, D. and McGregor D.R. (1992a), "Nonstationary Function Optimisation Using the Structured Genetic Algorithm", *Proceedings of the International Conference on Parallel Problem Solving from Nature (PPSN'92)*, Brussels (Belgium), pp 145-154

Dasgupta, D. and McGregor D.R. (1992b), "Designing Application-Specific Neural Networks Using the Structured Genetic Algorithm", *Proceedings of the International Workshop on Combinations of Genetic Algorithms and Neural Networks (COGANN'92)*, Baltimore (USA), pp 87-96

Dasgupta, D. and McGregor D.R. (1992c), "Engineering Optimizations Using the Structured Genetic Algorithm", *Proceedings of the European Conference in Artificial Intelligence (ECAI'92)*, Vienna (Austria), pp 608-609

Dasgupta, D. and McGregor D.R. (1993a), "sGA: A Structured Genetic Algorithm", *Technical Report No. IKBS-11-93.ps.Z*, Dept. of Computer Science, University of Strathclyde (UK)

Dasgupta, D. and McGregor D.R. (1993b), "Genetically Designing Neuro-Controllers for a Dynamic System", *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, Nagoya (Japan), pp 2951-2955

Davies, R. and Clarke, T. (1995), "Parallel Implementation of a Genetic Algorithm", *Control Engineering Practice*, Vol. 3, No. 1, pp 11-19

Day, S. (2001), "Design of Production-Friendly High-Speed Ship Hullforms Using Evolutionary Algorithms", *Proceedings of the Second International Conference on Advance Engineering Design*, Glasgow (UK), pp 464-469

DeJong, K. A. (1975), *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*, PhD Thesis, University of Michigan (USA)

DeMoura Oliveira, P.B. and Jones, A.H. (1997), "Robust Co-Evolutionary Design of SISO Smith Predictor PID Controllers", *Proceedings of the IEE International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA'97)*, Glasgow (UK), pp 504-509

Desanj, D.S., Grimble, M.J. and Katebi, M.R. (1997), "State-Space Adaptive H_∞ Controller with Applications to a Ship Control System", *Transactions of the Institute of Measurement and Control*, Vol. 19, No 3, pp 139-153

Donha, D.C. and Tannuri, E.A. (2001), "Non Linear Semi-Submersible Positioning System Design Using an H_∞ Controller", *Proceedings of the IFAC Conference on Control Application in Marine Systems (CAMS'01)*, Glasgow (UK), pp 95-100

Dohna, D.C., Desanj, D.S. and Katebi, M.R. (1997), "Automatic Weight Selection for H_∞ Control", *Proceedings of the IEE International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA '97)*, Glasgow (UK), pp 50-55

Dorado, J., Rabuñal, J.R., Puertas, J., Santos, A. and Rivero, D. (2002), "Prediction and Modelling of the Flow of a Typical Urban Basin through Genetic Programming", *EvoWorkshops 2002 Lecture Notes in Computer Science*, Vol. 2279, pp 190-201

Dorf, R.C. and Bishop, R.H. (2001), *Modern Control Systems*, Prentice Hall, New Jersey (USA)

Doyle, J. (1996), "Robust and Optimal Control", *Proceedings of the 35th IEEE Conference on Decision and Control*, Kobe (Japan), Vol. 4, pp 1595-1598

Doyle, J.C., Glover, K., Khargonekar, P.P. and Francis, B.A. (1989), "State-Space Solution to Standard H_2 and H_∞ Control Problems", *IEEE Transactions on Automatic Control*, Vol. 34, No. 8, pp 831-847

Dracopoulos, D.C. and Kent, S. (1997), "Genetic Programming for Prediction and Control", *Neural Computing and Applications*, Vol. 6, No. 4, pp 214-228

Dutton, K., Thompson, S. and Barraclough, B. (1997), *The Art of Control Engineering*, Addison-Wesley, Harlow

Ebner, M. (1998), "Evolution of a Control Architecture for a Mobile Robot", *ICES'98 Lecture Notes in Computer Science*, Vol. 1478, pp 303-310

Edwards, C.H. (1979), *The Historical Development of the Calculus*, Springer-Verlag

Edwards, C. and Spurgeon, S.K. (1998), *Sliding Mode Control: theory and applications*, Taylor and Francis Ltd, London (UK)

Eiben, A.E. and Schoenauer, M. (2002), "Evolutionary Computing", *Information Processing Letters*, Vol. 82, No. 1, pp 1-6

Ekart, A. (2000), "Shorter Fitness Preserving Genetic Programs", *EA99 Lecture Notes in Computer Science*, Vol. 1829, pp 73-83

Esparcia-Alcazar, A.I. (1998), *Genetic Programming for Adaptive Digital Signal Processing*, PhD Thesis, University of Glasgow, UK

Fahmy, M.M. and O'Reilly, J. (1982), "On Eigenstructure Assignment in Linear Multivariable Systems", *IEEE Transactions on Automatic Control*, Vol. 27, No. 3, pp 690-693

Fernandez, T. and Evett, M. (1998), "Numeric Mutation as an Improvement to Symbolic Regression in Genetic Programming", *EP98 Lecture Notes in Computer Science*, Vol. 1447, pp 251-260

Fernandez, F., Sanchez, J. M. and Tomassini, M. (2002), "Placing and Routing Circuits on FPGAs by Means of Parallel and Distributed Genetic Programming", *ICES 2001 Lecture Notes in Computer Science*, Vol. 2210, pp 204-215

Fletcher, L.R. (1981), "On Pole Placement in Linear Multivariable Systems with Direct Feedthrough. I. Theoretical Considerations", *International Journal of Control*, Vol. 33, No. 4, pp 739-749

Fogel, L.J. (1964), *On the Organization of Intellect*, PhD Thesis, University of California (USA)

Fonseca, C.M. and Fleming, P.J. (1994), "Multiobjective Optimal Controller Design with Genetic Algorithms", *Proceedings of the International Conference on Control*, Coventry (UK), pp 745-749

Fonseca, C.M. and Fleming, P.J. (1998a), "Multiobjective Optimization and Multiple Constraint Handling with Evolutionary Algorithms – Part I: A Unified Formulation", *IEEE Transactions on Systems, Man and Cybernetics. Part A: Systems and Humans*, Vol. 28, No. 1, pp 38-47

Fonseca, C.M. and Fleming, P.J. (1998b), "Multiobjective Optimization and Multiple Constraint Handling with Evolutionary Algorithms – Part II: Application Example", *IEEE Transactions on Systems, Man and Cybernetics. Part A: Systems and Humans*, Vol. 28, No. 1, pp 26-37

Fossard, A.J. (1993), "Helicopter Control Law based on Sliding Mode with Model Following", *International Journal of Control*, Vol. 57, No. 5, pp 1221-1235

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

Fossen, T.I. (2000), "Recent Developments in Ship Control Systems Design", *World Superyacht Review*, Sterling Publications Limited, London, pp 115-116

Fossen, T.I., Sagatun, S.I. and Sorensen, A.J. (1996), "Identification of Dynamically Positioned Ships", *Control Engineering Practice*, Vol. 4, No. 3, pp 369-376

Franklin, G.F., Powell, J.D. and Emami-Naeini, A. (1994), *Feedback Control of Dynamic Systems*, Addison-Wesley, USA

Fraser, A.P. (1994), "Genetic Programming in C++", *Technical Report 040*, University of Salford (UK)

French, I.G., Cox, C.S. and Ho, C.K.S. (1997), "Genetic Algorithm in Model Structure and Controller Structure Identification", *Proceedings of the Institution of Mechanical Engineers*, Part I, Vol. 211, pp 333-343

Gilfind, A., Gigante, M. and Al-Qaimari, G. (2000), "Evolving Performance Control Systems for Digital Puppetry", *The Journal of Visualization and Computer Animation*, Vol. 11, pp 169-183

Glover, K. and Doyle, J. (1988), "State-Space Formulae for All Stabilizing Controllers that Satisfy an H_∞ Norm Bound and Relations to Risk Sensitivity", *Systems and Control Letters*, Vol. 11, pp 167-172

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

Goldberg, D.E., Korb, B. and Deb, K. (1989), "Messy Genetic Algorithms: Motivation Analysis and First Results", *Complex Systems*, Vol. 3, pp 493-530

Gray, G.J., Li, Y., Murray-Smith, D.J., Ronco, E. and Sharman, K.C. (1997), "The Application of Genetic Algorithms to Gain-Scheduling Controller Analysis and Design", *Proceedings of the IEE International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA '97)*, Glasgow (UK), pp 364-368

Gray, G.J., Murray-Smith, D.J., Li, Y., Sharman, K.C. and Weinbrenner, T. (1998), "Nonlinear Model Structure Identification Using Genetic Programming", *Control Engineering Practice*, Vol. 6, pp 1341-1352

Grenfenstette, J.J. (1986), "Optimization of Control Parameters for Genetic Algorithms", *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 16, No. 1, pp 122-128

Grimble, M.J. (1986), "Optimal H_∞ Robustness and the Relationship to LQG Design Problems", *International Journal of Control*, Vol. 43, No. 2, pp 351-372

Grimble, M.J. (1987a), "H_∞ Robust Controller for Self-Tuning Control Applications. Part 1: Controller Design", *International Journal of Control*, Vol. 46, No. 4, pp 1429-1444

Grimble, M.J. (1987b), "H_∞ Robust Controller for Self-Tuning Control Applications. Part 2: Self-Tuning and Robustness", *International Journal of Control*, Vol. 46, No. 5, pp 1819-1840

Grosman, B. and Lewin, D. R. (2002), "Automated Nonlinear Model Predictive Control Using Genetic Programming", *Computers and Chemical Engineering*, Vol. 26, pp 631-640

Hagglund, T. and Astrom, K.J. (1991), "Industrial Adaptive Controllers Based on Frequency Response Techniques", *Automatica*, Vol. 27, No. 4, pp 599-609

Han, Y.S., Kim, Y.S. and Okuma, S. (2000), "The Position Control of Induction Motors Using a Binary Disturbance Observer", *Advanced Robotics*, Vol. 14, No. 2, pp 119-134

Healey, A.J. and Lienard, D. (1993), "Multivariable Sliding Mode Control for Autonomous Diving and Steering of Unmanned Underwater Vehicles", *IEEE Journal of Oceanic Engineering*, Vol. 18, No. 3, pp 327-339

Healey, A.J. and Marco, D.B. (1992), "Slow Speed Flight Control of Autonomous Underwater Vehicles: Experimental Results with NPS AUV II", *Proceedings of the Second International Offshore and Polar Engineering Conference*, San Francisco (USA), pp 523-532

Ho, W.K., Lim, K.W., Hang, C.C. and Ni, L.Y. (1999), "Getting more Phase Margin and Performance out of PID Controllers", *Automatica*, Vol. 35, pp 1579-1585

Holland, J. H. (1975), *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor

Homaifar, A., Bikdash, M. and Gopalan, V. (1997), "Design Using Genetic Algorithms of Hierarchical Hybrid Fuzzy-PID Controllers of Two-Link Robotic Arms", *Journal of Robotic Systems*, Vol. 14, No. 6, pp 449-463

Howard, L.M. and D'Angelo, D.J. (1995), "The GA-P: A Genetic Algorithm and Genetic Programming Hybrid", *IEEE Expert*, Vol. 10, No. 3, pp 11-15

Howard, D. and Roberts, S.C. (2002), "The Prediction of Journey Times on Motorways Using Genetic Programming", *EvoWorkshops 2002 Lecture Notes in Computer Science*, Vol. 2279, pp 210-221

Hsu, W.W. and Hsu, C.C. (2002), "GEC: An Evolutionary Approach for Evolving Classifiers", *PAKDD'2002 Lecture Notes in Artificial Intelligence*, Vol. 2336, pp 450-455

Hung, J.Y, Gao, W. and Hung J.C. (1993), "Variable Structure Control: A Survey", *IEEE Transaction on Industrial Electronics*, Vol. 40, No. 1, pp 2-22

Imai, K., Kamiura, N. and Hata, Y. (1999), "An Unsupervised Clustering with Evolutionary Strategy to Estimate the Cluster Number", *Proceedings of the International Conference on Computational Intelligence, Theory and Applications*, Dortmund (Germany), pp 99-107

Jones, A.H. and DeMoura Oliveira, P.B. (1995), "Genetic Auto-Tuning of PID Controllers", *Proceedings of the IEE/IEEE International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA '95)*, Sheffield (UK), pp 141-145

Jones, A.H., Lin Y.C., DeMoura Oliveira, P.B. and Kenway, S.B. (1997), "Auto-Tuning of Dual Mode Controllers Using Genetic Algorithms", *Proceedings of the IEE International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA '97)*, Glasgow (UK), pp 516-521

Kallstrom, C.G. and Astrom, K.J. (1981), "Experiences of System Identification Applied to Ship Steering", *Automatica*, Vol. 17, No. 1, pp 187-198

Kallstrom, C.G., Astrom, K.J., Thorell, N.E., Eriksson, J. and Sten, L. (1979), "Adaptive Autopilots for Tankers", *Automatica*, Vol. 15, pp 241-254

Katebi, M.R., Yamamoto, I., Matsuura, M., Grimble, M.J., Hirayama, H. and Okamoto, N. (2001), "Robust Dynamic Ship Positioning Control System Design and Applications", *International Journal of Robust and Nonlinear Control*, Vol. 11, No. 13, pp 1257-1284

Kaustky, J., Nichols, N.K. and Van Dooren, P. (1985), "Robust pole assignment in linear state feedback", *International Journal of Control*, Vol. 41, pp 1129-1155

Koza, J.R. (1992), *Genetic programming: On the Programming of Computers by Means of Natural Selection*, MIT Press, Cambridge, MA

Koza, J.R. (1994), *Genetic Programming II: Automatic Discovery of Reusable Programs*, MIT Press, Cambridge, MA

Koza, J.R., Andre, D., Bennet III, F.H. and Keane, M.A. (1997), "Design of a High-Gain Operational Amplifier and Other Circuits by Means of Genetic Programming", *GP97 Lecture Notes in Computer Science*, Vol. 1213, pp 125-135

Koza, J.R., Keane, M.A., Yu, J., Mydlowec, W. and Bennet III, F.H. (2000), "Automatic Synthesis of Both the Topology and Parameters for a Controller for a Three-Lag Plant with a Five-Second Delay Using Genetic Programming", *EvoWorkshops 2000 Lecture Notes in Computer Science*, Vol. 1803, pp 168-177

Kristinsson, K. and Dumont, G.A. (1992), "System Identification and Control using Genetic Algorithms", *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 22, No. 5, pp 1033-1046

Krohling, R. (1997), "Design of a PID Controller for Disturbance Rejection: A Genetic Optimization Approach", *Proceedings of the IEE International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA '97)*, Glasgow (UK), pp 498-503

Krohling, R. and Rey, J.P. (2001), "Design of Optimal Disturbance Rejection PID Controllers Using Genetic Algorithms", *IEEE Transactions on Evolutionary Computation*, Vol. 5, No. 1, pp 78-82

Lam, J. and Tam, H.K. (2000), "Robust Output Feedback Pole Assignment Using Genetic Algorithm", *Proceedings of the Institution of Mechanical Engineers*, Part I, Vol. 214, pp 327-334

Langdon, W.B. and Poli, R. (1998), "Fitness Causes Bloat", *Proceedings of the 1st European Workshop on Genetic Programming (EuroGP '98)*, Paris (France), pp 37-48

Lea, R.K., Allen, R. and Merry, S.L. (1999), "A Comparative Study of Control Techniques for an Underwater Flight Vehicle", *International Journal of System Science*, Vol. 30, No. 9, pp 947-964

Leithead, W.E. and O'Reilly, J. (1992), "Performance Issues in the Individual Channel Design of 2-input 2-output Systems. Part 2. Robustness Issues", *International Journal of Control*, Vol. 55, No. 1, pp 3-47

Lelic, M. and Gajic, Z. (2000), "A Reference Guide to PID Controllers in the Nineties", *Proceedings of the IFAC Workshop on Digital Control: Past, Present and Future (PID2000)*, Terrasa (Spain), pp 73-82

Li, Y., Ng, K.C., Murray-Smith, D.J., Gray, G.J. and Sharman, K.C. (1996), "Genetic Algorithm Automated Approach to the Design of Sliding Mode Control Systems", *International Journal of Control*, Vol. 63, No. 4, pp 721-739

Liceaga-Castro, E. and van der Molen, G. (1995), "Submarine H_∞ Depth Control Under Wave Disturbances", *IEEE Transactions on Control Systems Technology*, Vol. 3, No. 3, pp 338-345

Lin, F.J. and Chou, W.D. (2003), "An Induction Motor Servo drive Using Sliding-Mode Controller with Genetic Algorithm", *Electric Power Systems Research*, Vol. 64, No. 2, pp 93-108

Lindegaard, K.P (2003), *Acceleration Feedback in Dynamic Positioning Systems*, PhD Thesis, Norwegian University of Science and Technology, Trondheim, (Norway)

Lindegaard, K.P. and Fossen, T.I. (2002), "Fuel Efficient Rudder and Propeller Control Allocation for Marine Craft: Experiments with a Model Ship", *To appear in IEEE Transactions on Control Systems Technology*

Luke, S. (2000a), *Issues in Scaling Genetic Programming: Breeding Strategies, Tree Generation, and Code Bloat*, PhD Thesis, University of Maryland (USA)

Luke, S. (2000b), "Two Fast Tree-Creation Algorithms for Genetic Programming", *IEEE Transactions on Evolutionary Computation*, Vol. 4, No. 3, pp 274-283

Luke, S. (2000c), "Code Growth Is Not Caused by Introns", *Late Breaking Papers at the 2000 Genetic and Evolutionary Computation Conference (GECCO-2000)*, pp 228-235

Manness, M.A. and Murray-Smith, D.J. (1992), "Aspects of Multivariable Flight Control Law Design for Helicopters Using Eigenstructure Assignment", *Journal of the American Helicopter Society*, Vol. 1, pp 18-32

Marshfield, W.B. (1991), "Submarine Periscope-Depth Depth-Keeping Using an H_∞ Controller together with Sea-Noise Reduction Notch-Filters", *Transactions of the Institute of Measurement and Control*, Vol. 13, No. 5, pp 233-240

Mataric, M. and Cliff, D. (1996), "Challenges in Evolving Controllers for Physical Robots", *Robotics and Autonomous Systems*, Vol. 19, pp 67-83

Matas, J., Garcia de Vicuna, L., Lopez, O., Castilla, M. and Lopez, M. (2000), "Sliding Mode Control Parameters Optimization Using Genetic Algorithms", *Proceedings of the 9th International Conference and Exhibition on Power Electronics and Motion Control*, Kosice (Slovakia), Vol. 3, pp 96-101

McFarlane, D.C. and Glover, K. (1992), "A Loop-Shaping Design Procedure Using H_∞ Synthesis", *IEEE Transactions on Automatic Control*, Vol. 37, No. 6, pp 759-769

McGookin, E. W. (1997), *Optimization of Sliding Mode Controllers for Marine Applications: A Study of Methods and Implementation Issues*, PhD Thesis, University of Glasgow, UK

McGookin, E.W. (2001), "AUV Sliding Mode Autopilot Optimisation Using Genetic Algorithms", *Proceedings of the IFAC Conference on Control Applications in Marine Systems (CAMS'01)*, Glasgow (UK), pp 317-322

McGookin, E.W., Murray-Smith, D.J. and Li, Y. (1996), "Submarine Sliding Mode Controller Optimisation Using Genetic Algorithms", *Proceedings of the UKACC International Conference on Control*, pp 424-429

McGookin, E.W., Murray-Smith, D.J. and Li, Y. (1997a), "Parameter Optimisation of a Nonlinear Tanker Control System Using Genetic Algorithms", *Proceedings of the IEE International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA '97)*, Glasgow (UK), pp 37-42

McGookin, E.W., Murray-Smith, D.J., Li, Y. and Fossen, T.I. (1997b), "Non-Linear Tanker Control System Parameter Optimisation Using Genetic Algorithms", *Proceedings of the MTS/IEEE Oceans Conference*, Vol. 1, pp 17-22

McGookin, E.W., Murray-Smith, D.J. and Li, Y. (1997c), "A Population Minimisation Process for Genetic Algorithms and its Application to Controller Optimisation", *Proceedings of the IEE International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA '97)*, Glasgow (UK), pp 79-84

McGookin, E.W., Murray-Smith, D.J., Li, Y. and Fossen, T.I. (2000a), "Ship Steering Control System Optimisation Using Genetic Algorithms", *Control Engineering Practice*, Vol. 8, No. 4, pp 429-443

McGookin, E.W., Murray-Smith, D.J., Li, Y. and Fossen, T.I. (2000b), "The Optimisation of a Tanker Autopilot Control System Using Genetic Algorithms", *Transactions of the Institute of Measurement and Control*, Vol. 22, No. 2, pp. 141-178

McGookin, E.W., Murray-Smith, D.J., Li, Y. and Fossen, T.I. (2000c), "The Use of Genetic Algorithms for Nonlinear Controller Design", *Acta Polytechnica*, Vol. 40, No. 3, pp 14-21

Mengali, G. (2003), "Role of Eigenvectors in Aircraft Dynamics Optimization", *Journal of Guidance, Control and Dynamics*, Vol. 26, No. 2, pp 340-346

Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H. and Teller, E. (1953), "Equation of State Calculation Using Fast Computing Machines", *Journal of Chemical Physics*, Vol. 21, pp 1087-1092

Michalewicz, Z. (1992), *Genetic Algorithms + Data Structure = Evolution Programs*, Springer-Verlag, New York

Minorsky, N. (1922), "Directional Stability of Automatic Steered Bodies", *Journal American Society of Naval Engineers*, Vol. 34, No. 2, pp 280-309

Mitsukura, Y., Yamamoto, T. and Kaneda, M. (2000), "A Design of PID Controllers Using a Genetic Algorithm", *Transactions of the Society of Instrument and Control Engineers*, Vol. 36, No. 1, pp 75-81

Moin, N.H., Zinober, A.S.I. and Harley, P.J. (1995), "Sliding Mode Control Design Using Genetic Algorithms", *Proceedings of the IEE/IEEE International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA '95)*, Sheffield (UK), pp 238-244

Mudge, S.K. and Patton, R.J. (1988), "Enhanced Assessment of Robustness for an Aircraft's Sliding Mode Controller", *Journal of Guidance, Control and Dynamics*, Vol. 11, No. 6, pp 500-507

Murray-Smith, D.J., Kocijan, J. and Gong, M. (2003), "A Signal Convolution Method for Estimation of Controller Parameter Sensitivity Functions for Tuning of Feedback Control Systems by an Iterative Process", *Control Engineering Practice*, Vol. 11, pp 1087-1094

Ng, K.L. (2000), *Genetic Programming in Control Theory: On Evolving Programs and Solutions to Control Problems*, Master Thesis, Lund Institute of Technology (Sweden)

Ng, K.C., Li, Y., Murray-Smith, D.J. and Sharman, K.C. (1995), "Genetic Algorithms Applied to Fuzzy Sliding mode Control Design", *Proceedings of the IEE/IEEE International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA '95)*, Sheffield (UK), pp 220-225

Nordin, P., Francone, F. and Banzhaf, W. (1996), "Explicitely Defined Introns and Destructive Crossover in Genetic Programming", *Advances in Genetic Programming II*, MIT Press, Chapter 6, pp 111-134

Norrbin, N.H. (1970), "Theory and Observations on the Use of a Mathematical model for Ship Maneuvering in Deep and Confined Waters", *Proceedings of the 8th Symposium on Naval Hydrodynamics*, Pasadena (USA), pp 807-904

O'Dwyer, A. (2000), "PID Compensation of Time Delayed Processes: A Survey", *Proceedings of the Irish Signals and Systems Conference*, Dublin (Ireland), pp 5-12

Omatu, S. and Deris, S. (1996), "Stabilization of Inverted Pendulum by the Genetic Algorithm", *Proceedings of the IEEE International Conference on Evolutionary Computation*, Nagoya (Japan), pp 700-705

Ono, O., Kobayashi, B. and Kato, H. (1996), "Optimal Dynamic motion Planning of Autonomous Vehicles by a Structured Genetic Algorithm", *Proceedings of the 13th IFAC World Congress*, Vol. Q: Automotive, Marine, Autonomous Vehicles, San Francisco (USA), pp 435-440

O'Reilly, U. (1995), *An Analysis of Genetic Programming*, PhD Thesis, Carleton University (Canada)

O'Reilly, J. and Leithead, W.E. (1991), "Multivariable Control by Individual Channel Design", *International Journal of Control*, Vol. 54, No. 1, pp 1-46

Parmee, I.C. (1998), "Evolutionary and Adaptive Strategies for Efficient Search Across Whole System Engineering Design", *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, Vol. 12, pp 431-445

Patton, R.J. and Liu, G.P. (1994), "Robust Control Design Via Eigenstructure Assignment, Genetic Algorithms and Gradient-Based Optimisation", *Proceedings of the Institution of Electrical Engineers*, Part D, Vol. 141, No. 3, pp 202-208

Porter, B. and D'Azzo, J.J. (1977), "Algorithm for the Synthesis of State Feedback Regulators by Entire Eigenstructure Assignment", *Electronics Letters*, Vol. 13, pp 230-231

Porter, B. and Jones, A.H. (1992), "Genetic Tuning of Digital PID Controllers", *Electronic Letters*, Vol. 28, No 9, pp 843-844

Postlethwaite, I. and Bates, D.G. (1999), "Robust Integrated Flight and Propulsion Controller for the Harrier Aircraft", *Journal of Guidance, Control and Dynamics*, Vol. 22, No. 2, pp 286-290

Postlethwaite, I., Lin, J.L. and Gu, D.W. (1991), "A Loop-Shaping Approach to Robust Performance for SISO Systems", *Transactions of the Institute of Measurement and Control*, Vol. 13, No. 5, pp 262-268

Putnam, J. (1996), "A Grammar Based Genetic Programming Technique Applied to Music Generation", *Evolutionary Programming V*, MIT Press, Cambridge, MA, pp 363-368

Rafiq, M.Y. and Williams, C. (1998), "An Investigation into the Integration of Neural Networks with the Structured Genetic Algorithm to Aid conceptual Design", *Artificial Intelligence in Structural Engineering. Information, Technology for Design, Collaboration, Maintenance and Monitoring*, Lausanne (Switzerland), pp 295-307

Rechenberg, I. (1973), *Optimierung Technischer Systeme nach Prinzipien der Biologischen Evolution*, Frommann-Holzboog, Stuttgart (Germany)

Ribeiro Filho, J.L., Treleaven, P.C. and Alippi, C. (1994), "Genetic-Algorithm Programming Environments", *Computer*, Vol. 27, No. 6, pp 28-43

Rosenbloom, P. (1987), "Best First Search", *Encyclopedia of Artificial Intelligence*, Vol. 2, Wiley, New York (USA)

Saelid, S. and Jenssen, N.A. (1983), "Adaptive Ship Autopilot with Wave Filter", *Modeling, Identification and Control*, Vol. 4, No. 1, pp 33-45

Salamci, M.U., Ozgoren, M.K. and Banks, S.P. (2000), "Sliding Mode Control with Optimal Sliding Surfaces for Missile Autopilot Design", *Journal of Guidance, Control and Dynamics*, Vol. 23, No. 4, pp 719-727

Salami, M. and Cain, G. (1995), "An Adaptive PID Controller based on Genetic Algorithm Processor", *Proceedings of the IEE/IEEE International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA '95)*, Sheffield (UK), pp 88-94

Schwefel, H.P. (1975), *Evolutionsstrategie und Numerische Optimierung*, PhD Thesis, Technische Universitat Berlin (Germany)

Shaw, K.J. and Fleming, P.J. (2000), "Genetic Algorithms for Scheduling: Incorporation of User Preferences", *Transactions of the Institute of Measurement and Control*, Vol. 22, No. 2, pp 195-210

Shimooka, H. and Fujimoto, Y. (1998), "Generating Equations with Genetic Programming for Control of a Movable Inverted Pendulum", *SEAL98 Lecture Notes in Computer Science*, Vol. 1585, pp 179-186

Silvestre, C. and Pascoal, A. (1997), "Control of an AUV in the Vertical and Horizontal Planes: System Design and Test at Sea", *Transactions of the Institute of Measurement and Control*, Vol. 19, No. 3, pp 126-138

Singleton, A. (1994), "Genetic Programming with C++", *BYTE Magazine*, No. February

Skjetne, R. (2003), "Ship Maneuvering: the Past, the Present and the Future", *Sea Technology*, No. March, pp 33-37

Slotine, J.J.E. (1984), "Sliding Controller Design for Non-Linear Systems", *International Journal of Control*, Vol. 40, No. 2, pp 421-434

Slotine, J.J.E. and Li, W. (1991), *Applied Nonlinear Control*, Prentice-Hall International Inc., New Jersey (USA)

Smierzchalski, R. (2001), "On-Line Trajectory Planning in Collision Situations at Sea by Evolutionary Computation-Experiments", *Proceedings of the IFAC Conference on Control Applications in Marine Systems (CAMS'01)*, Glasgow (UK), pp 407-412

Smierzchalski, R., Kwiesielewicz, M., Szymanski, M. and Sutton, R. (2001), "Predictive Control and Dynamic Planning of an Autonomous Underwater Vehicle", *Proceedings of the IFAC Conference on Control Applications in Marine Systems (CAMS'01)*, Glasgow (UK), pp 161-166

Sorensen, A.J., Sagatun, S.I. and Fossen, T.I. (1996), "Design of a Dynamic Positioning System Using Model-Based Control", *Control Engineering Practice*, Vol. 4, No. 3, pp 359-368

Soule, T. and Foster, J.A. (1998), "Removal Bias: A New Cause of Code Growth in Tree Based Evolutionary Programming", *Proceedings of the IEEE International Conference on Evolutionary Computation*, Anchorage (USA), pp 781-786

Srinivas, M. and Patnaik, L.M. (1994), "Genetic Algorithms: A Survey", *Computer*, Vol. 27, No. 6, pp 17-26

Stawicki, K. and Smierzchalski, R. (2001), "Methods of Optimal Ship Routing for Weather Perturbations", *Proceedings of the IFAC Conference on Control Applications in Marine Systems (CAMS'01)*, Glasgow (UK), pp 101-106

Strand, J.P. (1999), *Nonlinear Position Control Systems Design for Marine Vessels*, PhD Thesis, Norwegian University of Science and Technology, Trondheim, (Norway)

Sveen, D.A. (2003), *Robust and Adaptive Tracking Control of Surface vessel for Synchronization with an ROV: Practical Implementation on CyberShip II*, Master Thesis, Norwegian University of Science and Technology, Trondheim, (Norway)

Sweriduk, G.D., Menon, P.K. and Steinberg, M.L. (1998), "Robust Command Augmentation System Design Using Genetic Methods", *Proceeding of the AIAA Guidance, Navigation and Control Conference*, Boston (USA)

Tackett, W.A. (1994), *Recombination, Selection and the Genetic Construction of Computer Programs*, PhD Thesis, University of Southern California (USA)

Tan, K.C. and Li, Y. (1996), "L_∞ Identification and Model Reduction Using a Learning Genetic Algorithm", *Proceedings of the UKACC International Conference in Control*, Exeter (UK), pp 1125-1130

Tan, K.C. and Li, Y. (1997), "Evolutionary System Identification in the Time Domain", *Proceedings of the Institution of Mechanical Engineers*, Part I, Vol. 221, pp 319-323

Tang, K.S., Chan, C.Y., Man K.F. and Kwong, S. (1995), "Genetic Structure for NN Topology and Weights Optimization", *Proceedings of the IEE/IEEE International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA '95)*, Sheffield (UK), pp 250-255

Tang, K.S., Man K.F. and Gu, D.W. (1996), "Structured Genetic Algorithm for Robust H_∞ Control Systems Design", *IEEE Transactions on Industrial Electronics*, Vol. 43, No. 5, pp 575-582

Utkin, V.I. (1972), "Equations of the Slipping Regime in Discontinuous Systems II", *Automation and Remote Control*, Vol. 2, pp 211-219

Utkin, V.I. and Yang, K.D. (1978), "Methods for Constructing Discontinuity Planes in Multidimensional Variable Structure Systems", *Automation and Remote Control*, Vol. 39, pp 1466-1470

Van Belle, T. and Ackley, D.H. (2002), "Uniform Subtree Mutation", *Euro GP 2002 Lecture Notes in Computer Science*, Vol. 2278, pp 152-161

Vlachos, C., Evans, J.T. and Williams, D. (1997), "PI Controller Tuning for Multivariable Processes Using Genetic Algorithms", *Proceedings of the IEE International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA '97)*, Glasgow (UK), pp 43-49

Voulgaris, P. and Valavani, L. (1991), "High Performance Linear-Quadratic and H_∞ Designs for a Supermaneuverable Aircraft", *Journal of Guidance and Control*, Vol. 14, No. 1, pp 157-165

Walker, D.J. and Postlethwaite, I. (1996), "Advance Helicopter Flight Control Using 2-Degree-of-Freedom H_∞ optimisation", *Journal of Guidance, Control and Dynamics*, Vol. 19, No. 2, pp 461-468

Wang, P and Kwok, D.P. (1994), "Optimal Design of PID Process Controllers based on Genetic Algorithms", *Control Engineering Practice*, Vol. 2, No. 4, pp 641-648

Wang, Q. and Zalzala, A.M.S. (1996a), "Genetic-Based Pole-Placement for Self-Tuning PID", *IEE Colloquium on Adaptive Controllers in Practice*, London (UK), Vol. 3, pp 1-3

Wang, Q. and Zalzala, A.M.S. (1996b), "Transputer Based Genetic Algorithm Motion Control for PUMA Robot", *Mechatronics*, Vol. 6, No. 3, pp 349-365

Wang Y.P., Watson, N.R. and Chong, H.H. (2002), "Modified Genetic Algorithm Approach to Design of an Optimal PID Controller for AC-DC Transmission Systems", *International Journal of Electrical Power and Energy Systems*, Vol. 24, No. 1, pp 59-69

White, B.A. (1995), " Eigenstructure Assignment: A Survey", *Proceedings of the Institution of Mechanical Engineers*, Vol. 209, pp 1-11

Wonham, W.M. (1967), "On Pole Assignment in Multi-Input Controllable Linear Systems", *IEEE Transactions on Automatic Control*, Vol.12, pp 660-665

Wu, S.T. (1997), "On Digital High-Gain and Sliding Mode Control", *International Journal of Control*, Vol. 66, No. 1, pp 65-83

Yanai, H. and Iba, H. (2002), "Multi-Agent Robot Learning of Genetic Programming: Solving an Escape Problem", *ICES 2001 Lecture Notes in Computer Science*, Vol. 2210, pp 192-203

Yang, Y. and Billings, S.A. (2000), "Neighborhood Detection and Rule Selection from Cellular Automata Patterns", *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, Vol. 30, No. 6, pp 840-847

Zames, G. (1966), "On the Input-Output Stability of Nonlinear Time-Varying Feedback Systems, Parts I and II", *IEEE Transactions on Automatic Control*, Vol. AC-11, pp 228-465

Zames, G. (1981), "Feedback and Optimal Sensitivity: Model Reference Transformations, Multiplicative Seminorms, and Approximate Inverses", *IEEE Transactions on Automatic Control*, Vol. AC-26, No. 2, pp 301-320

Zhou K., Doyle J.C. and Glover, K. (1996), *Robust and Optimal Control*, Prentice Hall, New Jersey

Zhu, Y., Qiu, L. and Tani, L. (2001), "Simultaneous Optimization of a Two-Link Flexible Robot Arm", *Journal of Robotics Systems*, Vol. 18, No. 1, pp 29-38

Ziegler, J.G. and Nichols, N.B. (1942), "Optimum Settings for Automatic Controllers", *Transactions of the ASME*, Vol. 64, pp 759-768

Zuidweg, J.K. (1970), *Automatic Guidance of Ships as a Control Problem*, PhD Thesis, Delft University of Technology (The Netherlands)

APPENDIX A

CS1 AND CS2 MATHEMATICAL MODELS

A.1. MODELS MATRICES FOR EQUATIONS (3.2) AND (3.3)

Mass matrix:

$$\mathbf{M} = \begin{pmatrix} m - X_{\dot{u}} & 0 & 0 \\ 0 & m - X_{\dot{v}} & m \cdot x_G - Y_{\dot{r}} \\ 0 & m \cdot x_G - N_{\dot{v}} & I_z - N_{\dot{r}} \end{pmatrix}$$

Coriolis matrix:

$$\mathbf{C}(\nu) = \begin{pmatrix} 0 & 0 & -(m - Y_{\dot{v}}) \cdot v - (m \cdot x_G - Y_{\dot{r}}) \cdot r \\ 0 & 0 & (m - X_{\dot{u}}) \cdot u \\ (m - Y_{\dot{v}}) \cdot v + (m \cdot x_G - Y_{\dot{r}}) \cdot r & -(m - X_{\dot{u}}) \cdot u & 0 \end{pmatrix}$$

Damping matrix:

$$\mathbf{D} = \begin{pmatrix} -X_u & 0 & 0 \\ 0 & -Y_v & -Y_r \\ 0 & -N_v & -N_r \end{pmatrix}$$

Euler matrix:

$$\mathbf{J} = \begin{pmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

A.2. CYBERSHIP I MATHEMATICAL MODEL

Model parameters:

Table A.1: Model Parameters for CS1

Main data	Hydrodynamic added mass terms	Damping terms
$m = 17.6$ (kg)	$X_{\dot{u}} = -1.4$ (kg)	$X_u = -4$ (kg/s)
$I_z = 1.8$ (kg·m^2)	$Y_{\dot{v}} = -17.6$ (kg)	$Y_v = -6$ (kg/s)
$x_G = -0.04$ (m)	$Y_{\dot{r}} = 0$	$Y_r = 0$
$Length = 1.19$ (m)	$N_{\dot{v}} = 0$	$N_v = 0$
	$N_{\dot{r}} = -0.18$ (kg·m^2)	$N_r = -1$ (kg·m^2/s)

Thruster configuration with respect to the centre of gravity:

Table A.2: Truster Configuration for CS1

	x (m)	y (m)
Port aft thruster	-0.493	-0.065
Starboard aft thruster	-0.493	0.065
Bow thruster	0.34	0
Bow thruster	0.36	0

Time constants:

Table A.3: Time Constants for CS1

$T_1 = 1$ (s)	$T_2 = 1$ (s)	$T_3 = 1$ (s)

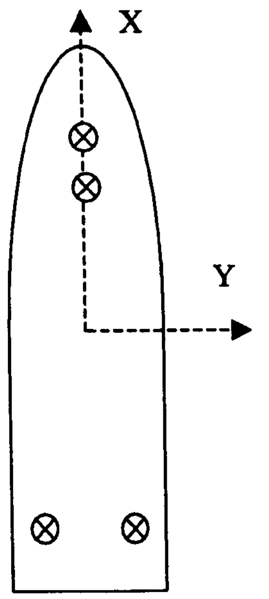


Figure A.1: Schematic Diagram of the Actuator's Layout for CS1

Transfer function matrix for the surge speed – heading channels

$$g_{11}(s)=\frac{0.0526}{s+0.2104};\ g_{12}(s)=0;\ g_{21}(s)=0;\ g_{22}(s)=\frac{0.5087\cdot s+0.2057}{s^3+0.6811\cdot s^2-2.2474\cdot s}$$

$$\gamma(s)=0$$

Transfer function matrix for the sway speed – heading channels

$$g_{11}(s)=\frac{0.0286\cdot s+0.0073}{s^2+0.6811\cdot s-2.1342};\ g_{12}(s)=\frac{0.0102\cdot s-0.1921}{s^2+0.6811\cdot s-2.1342}$$

$$g_{21}(s)=\frac{0.0102\cdot s+0.1697}{s^3+0.6811\cdot s^2-2.1342\cdot s};\ g_{22}(s)=\frac{0.5087\cdot s+0.3129}{s^3+0.6811\cdot s^2-2.1342\cdot s}$$

$$\gamma(s)=\frac{1.04\cdot 10^{-4}s^2-2.2848\cdot 10^{-4}s-0.0326}{0.0145\cdot s^2+0.0127\cdot s+0.0023}$$

A.3. CYBERSHIP II MATHEMATICAL MODEL

Model parameters:

Table A.4: Model Parameters for CS2

Main data	Hydrodynamic added mass terms	Damping terms
$m = 23.8$ (kg)	$X_{\dot{u}} = -2$ (kg)	$X_u = -2$ (kg/s)
$I_z = 1.74$ (kg·m^2)	$Y_{\dot{v}} = -10$ (kg)	$Y_v = -7$ (kg/s)
$x_G = 0.0425$ (m)	$Y_{\dot{r}} = 0$	$Y_r = -0.1$ (kg·m/s)
$Length = 1.255$ (m)	$N_{\dot{v}} = 0$	$N_v = -0.1$ (kg·m/s)
	$N_{\dot{r}} = -1$ (kg·m^2)	$N_r = -0.5$ (kg·m^2/s)

Thruster configuration with respect to the centre of gravity:

Table A.5: Thruster Configuration for CS2

	x (m)	y (m)
Port aft rudder	-0.54	-0.075
Starboard aft rudder	-0.54	0.075
Bow tunnel thruster	0.465	0

Time constants:

Table A.6: Time Constants for CS2

$T_1 = 0.03$ (s)	$T_2 = 0.03$ (s)	$T_3 = 0.03$ (s)

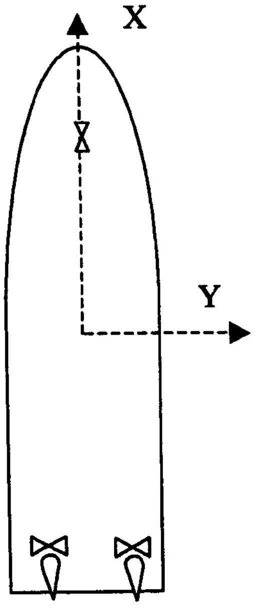


Figure A.2: Schematic Diagram of the Actuator's Layout for CS2

Transfer function matrix for the surge speed – heading channels

$$g_{11}(s)=\frac{0.0388}{s+0.0776};\ g_{12}(s)=0;\ g_{21}(s)=0;\ g_{22}(s)=\frac{0.3663\cdot s+0.0758}{s^3+0.3893\cdot s^2-1.0294\cdot s}$$

$$\gamma(s)=0$$

Transfer function matrix for the sway speed – heading channels

$$g_{11}(s)=\frac{0.0299\cdot s+0.0131}{s^2+0.3893\cdot s-1.0294};\ g_{12}(s)=\frac{-0.011\cdot s-0.1967}{s^2+0.3893\cdot s-1.0294}$$

$$g_{21}(s)=\frac{-0.011\cdot s-0.0617}{s^3+0.3893\cdot s^2-1.0294\cdot s};\ g_{12}(s)=\frac{0.3663\cdot s+0.0758}{s^3+0.3893\cdot s^2-1.0294\cdot s}$$

$$\gamma(s)=\frac{0.3663\cdot s+0.0758}{s^3-0.3893\cdot s^2-1.0294\cdot s}$$

APPENDIX B

GENETIC ALGORITHM COMPARISON STUDIES COMPLEMENTARY DATA

B.1. COST FUNCTION WEIGHTING PARAMETERS

Table B.1: Cost Function Weighting Parameters

	λ_1	μ_1	λ_2	μ_2
CS1	10^{-3}	10^{-2}	10^{-3}	10^{-2}
CS2	10^{-3}	10^{-2}	10^{-4}	10^{-3}

B.2. FIRST COMPARISON STUDY COMPLEMENTARY DATA

Table B.2: Best Costs Results

	PID		SM+PI		H_∞		PP	
	Mean	StnDev	Mean	StnDev	Mean	StnDev	Mean	StnDev
GA1	5.360	0.0893	5.566	0.7192	14.183	7.7925	5.705	0.0372
GA2	5.318	0.1432	20.386	28.6607	7.349	2.2586	6.081	0.4733
GA3	5.337	0.0811	5.67	1.3058	55.989	68.7262	5.699	0.0362
GA4	5.602	0.1109	6.343	2.5953	78.278	53.8847	5.71	0.0343
GA5	5.265	0.0739	4.504	0.1024	12.857	15.7294	5.774	0.2061
GA6	5.580	0.1480	5.823	0.9515	23.24	12.5208	5.709	0.0164
GA7	5.444	0.1507	7.836	4.0628	51.366	50.6942	5.689	0.0211
GA8	5.297	0.1053	7.206	3.2198	8.244	6.3528	5.69	0.0256
GA9	5.544	0.20371	5.446	1.3678	28.866	29.4222	5.684	0.0101
GA10	5.455	0.1727	6.583	2.4846	45.541	19.9756	5.706	0.0175
GA11	5.219	0.0190	5.034	0.8517	5.909	0.8575	5.672	0.0013
GA12	5.679	0.2417	6.921	2.5473	56.485	28.1896	5.713	0.0337
GA13	5.333	0.0847	5.333	0.968	20.957	17.1209	5.695	0.0228
GA14	5.246	0.0780	5.775	1.1317	6.678	1.8828	5.681	0.0141
GA15	5.340	0.0836	4.987	0.4134	16.726	5.8706	5.697	0.0137
GA16	5.622	0.1850	6.455	2.2365	43.604	14.7987	5.709	0.0313
GA17	5.200	0.0082	4.643	0.4655	6.259	1.5612	5.672	0.0013
GA18	5.633	0.2790	7.098	3.282	44.239	19.3803	5.725	0.0385

Table B.3: Generation of Convergence Results

	PID		SM+PI		H_∞		PP	
	Mean	StnDev	Mean	StnDev	Mean	StnDev	Mean	StnDev
GA1	17.333	6.774	21	8.505	30.167	13.359	8.667	4.679
GA2	9.167	2.544	24.5	12.285	33.333	11.629	5.333	1.599
GA3	11.333	4.15	34.167	7.84	22.833	12.402	7.167	4.74
GA4	11	9.309	16.5	14.338	11	9.018	11.667	7.157
GA5	8.833	3.532	24.5	11.529	30.833	5.177	5	1.915
GA6	19	11.416	18.833	14.404	20.167	18.225	11.167	5.145
GA7	17	9.764	19	13.976	18.333	9.049	5.5	3.253
GA8	7.833	3.976	19.667	12.12	26.5	9.962	12.167	13.018
GA9	20.667	4.11	25.667	14.091	20.333	16.967	7	3.606
GA10	9.5	5.284	19	9.781	7.667	5.935	8.167	3.236
GA11	7.5	3.731	17.167	8.591	23.833	4.776	3.5	2.217
GA12	21.167	9.923	18.667	12.802	12.5	10.259	5.833	3.184
GA13	19	10.263	13.5	4.992	17.5	7.869	5.5	2.5
GA14	6.5	1.708	17.833	9.634	34	11.832	3.5	0.764
GA15	14.167	6.962	27.5	8.539	31	12.329	9.5	5.909
GA16	11	4.655	24	12.329	9.5	12.633	6.242	2.911
GA17	6.167	1.344	16.667	10.546	24.5	5.47	2.639	1.384
GA18	8.833	2.853	16.5	10.626	7.667	6.6	4.333	4.497

Table B.4: Amount of Convergence in the Final Population

	PID		SM+PI		H_∞		PP	
	Mean	StnDev	Mean	StnDev	Mean	StnDev	Mean	StnDev
GA1	42.167	4.81	7.167	4.598	2.333	1.599	52.833	4.14
GA2	56.333	1.491	44.167	8.194	28.833	12.746	64.667	0.745
GA3	18.333	2.357	5.667	2.867	4.5	4.349	20.5	5.156
GA4	9.167	4.67	1	0	3.833	2.267	25.333	3.197
GA5	48	6.608	32	4.546	13.833	7.581	46.667	3.682
GA6	8.167	4.059	1.167	0.373	4.167	3.578	11.167	1.213
GA7	27.667	10.143	4.5	5.649	11	14.036	51.667	1.972
GA8	59.5	4.924	45.333	3.496	32.667	13.597	64.667	3.543
GA9	15.167	2.034	4	2.828	5.333	5.85	19.167	2.967
GA10	6.167	2.794	2	1	2.833	1.462	15.167	9.634
GA11	45.5	4.958	28.833	7.381	16.5	3.948	50.833	5.08
GA12	4.833	1.951	1.167	0.373	2.5	1.384	9.167	2.794
GA13	34.333	10.562	3.667	4.23	4.833	6.593	46.167	10.383
GA14	59.667	3.35	46	2.769	23.333	12.658	64	3.916
GA15	13.5	1.708	3	1.915	1.167	0.373	15.667	1.795
GA16	4.667	2.867	1	0	1.333	0.471	12.167	5.698
GA17	45.5	5.679	30	5.627	11.833	5.398	48.833	2.409
GA18	2.833	1.462	1.167	0.373	3	1.732	7.667	3.197

Table B.5: AMF in the Final Population

	PID		SM+PI		H_∞		PP	
	Mean	StnDev	Mean	StnDev	Mean	StnDev	Mean	StnDev
GA1	0.56	0.0442	0.567	0.0611	0.541	0.0578	0.583	0.061
GA2	0.91	0.0257	0.889	0.0203	0.822	0.0479	0.922	0.0206
GA3	0.513	0.0562	0.484	0.0513	0.421	0.0334	0.542	0.0413
GA4	0.354	0.0275	0.342	0.0203	0.334	0.0303	0.433	0.0308
GA5	0.738	0.0299	0.722	0.0441	0.619	0.0745	0.786	0.0356
GA6	0.281	0.0163	0.288	0.0281	0.274	0.0133	0.308	0.0278
GA7	0.524	0.047	0.482	0.0298	0.432	0.0306	0.554	0.0388
GA8	0.879	0.0249	0.875	0.0228	0.818	0.0225	0.917	0.0134
GA9	0.417	0.0167	0.38	0.0362	0.35	0.0136	0.441	0.0413
GA10	0.342	0.0113	0.391	0.0427	0.325	0.0253	0.365	0.029
GA11	0.724	0.0369	0.691	0.039	0.607	0.0346	0.787	0.0277
GA12	0.263	0.0182	0.252	0.0194	0.241	0.0412	0.278	0.0195
GA13	0.515	0.0228	0.461	0.0309	0.423	0.0355	0.498	0.0394
GA14	0.892	0.0342	0.878	0.0268	0.821	0.0245	0.94	0.0122
GA15	0.404	0.0251	0.362	0.0403	0.32	0.0258	0.408	0.037
GA16	0.326	0.0262	0.336	0.017	0.297	0.0179	0.349	0.014
GA17	0.731	0.0239	0.721	0.0234	0.585	0.0377	0.783	0.0295
GA18	0.248	0.0215	0.247	0.0161	0.225	0.0074	0.26	0.013

B.3. SECOND COMPARISON STUDY COMPLEMENTARY DATA

Table B.6: Best Costs Results

	PID		SM+PI		PP		Hinf	
	Mean	StDev	Mean	StDev	Mean	StDev	Mean	StDev
Benchmark	5.2005	0.0082	4.6433	0.4655	5.6721	0.0013	6.2591	1.5612
Non-uniform mutation	5.4502	0.4187	4.9621	0.6272	5.6738	0.0298	5.7478	0.8446
Minimisation (r=0.1)	5.4630	0.0879	5.1434	0.6733	5.7445	0.0210	13.275	17.151
Minimisation (r=0.05)	5.4838	0.1266	5.1964	1.0138	5.7054	0.0274	13.566	15.386
Combination	5.4293	0.0648	5.7902	1.4243	5.6988	0.0141	5.7197	0.3363
Exponential mutation	5.2086	0.0123	4.7732	0.6765	5.7659	0.2090	5.2519	0.0760
Exponential mut & cross	5.2518	0.0646	5.5803	0.7004	5.6705	0.0014	5.2929	0.0326
1-point crossover	5.4680	0.4194	4.5764	0.1999	5.7731	0.2063	5.3205	0.0762

Table B.7: Generation of Convergence Results

	Generation of convergence			
	PID	SM+PI	PP	Hinf
Benchmark	6.167	5.000	16.667	24.500
Non-uniform mutation	9.667	2.333	26.000	16.167
Minimisation (r=0.1)	1.833	1.333	13.667	18.500
Minimisation (r=0.05)	1.833	6.333	11.167	16.667
Combination	11.500	3.667	14.167	8.500
Exponential mutation	14.833	6.833	22.000	13.333
Exponential mut & cross	9.667	11.333	26.833	19.167
1-point crossover	10.500	5.000	17.000	13.833

Table B.8: Amount of Convergence and AMF in the Final Population

	AMF				Amount of Convergence			
	PID	SMPI	PP	Hinf	PID	SMPI	PP	Hinf
Benchmark	0.731	0.721	0.783	0.585	45.500	30.000	48.833	11.833
Non-uniform mutation	0.827	0.801	0.863	0.723	80.000	78.833	80.000	79.000
Minimisation (r=0.1)	0.401	0.513	0.426	0.546	2.500	3.000	2.667	3.167
Minimisation (r=0.05)	0.381	0.508	0.397	0.502	2.333	3.167	2.000	3.667
Combination	0.694	0.670	0.665	0.650	16.333	8.000	14.833	5.500
Exponential mutation	0.656	0.632	0.705	0.606	79.667	63.333	80.000	72.500
Exponential mut & cross	0.661	0.631	0.718	0.605	80.000	58.000	80.000	70.667
1-point crossover	0.639	0.667	0.725	0.598	80.000	75.167	80.000	74.000

Figure B.1 presents the performance degradation associated with very small populations and high effective mutation rate. The population sizes and best cost values have been average over 6 runs.

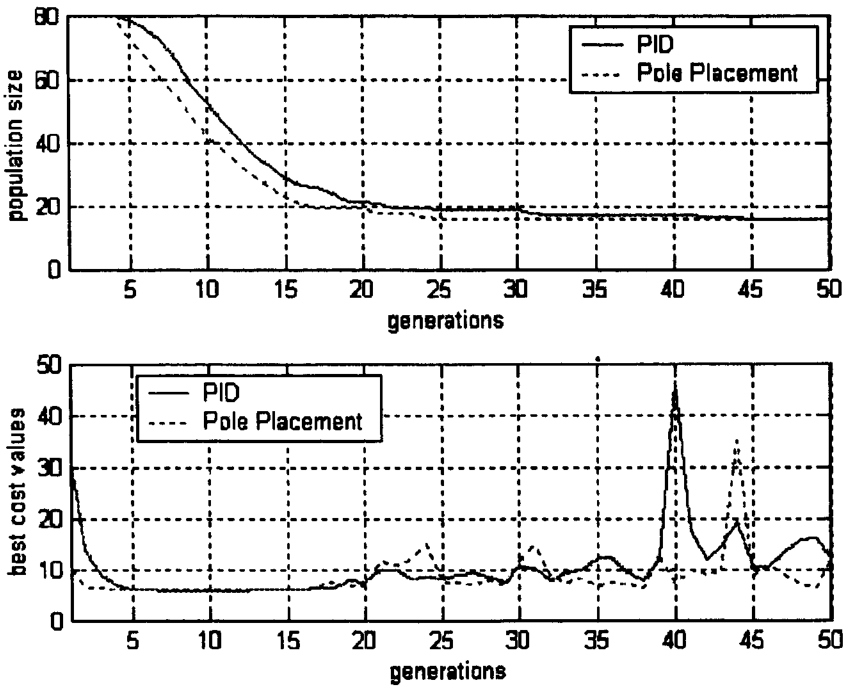


Figure B.1: Cost Degradation in Terms of Population Reduction

APPENDIX C

GENETIC ALGORITHM CONVERGENCE RATES

C.1. GA CONVERGENCE RATES

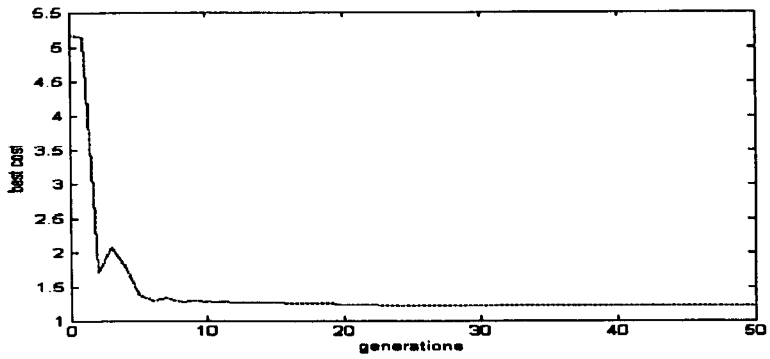


Figure C.1: GA Convergence for the PID Optimisation Without Waves

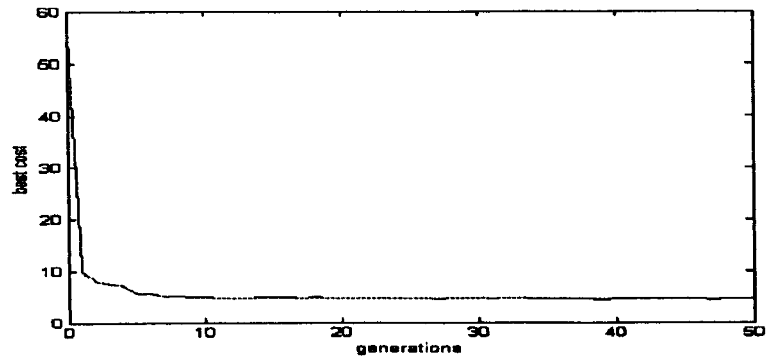


Figure C.2: GA Convergence for the PID Optimisation With Waves

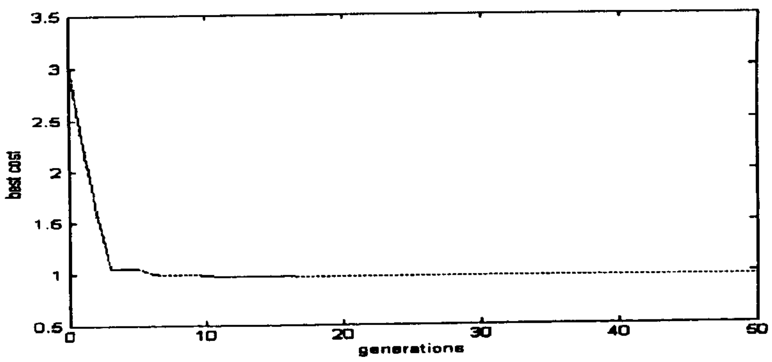


Figure C.3: GA Convergence for the PP Optimisation Without Waves

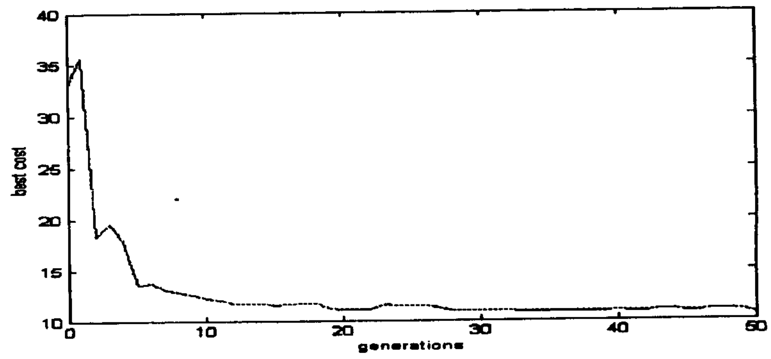


Figure C.4: GA Convergence for the PP Optimisation With Waves

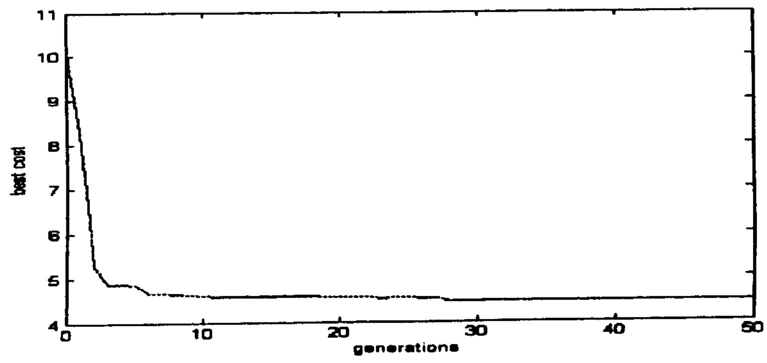


Figure C.5: GA Convergence for the SM+PI Optimisation Without Waves

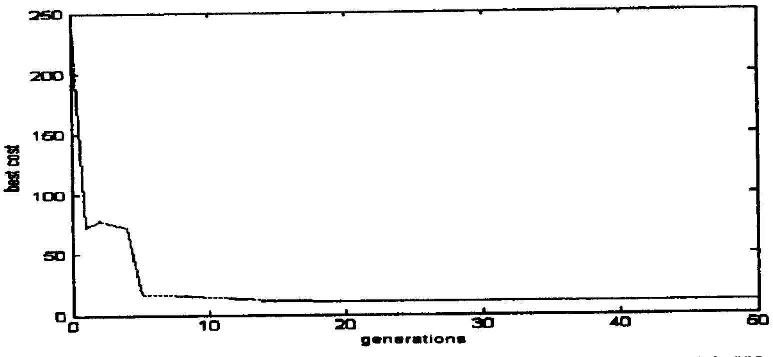


Figure C.6: GA Convergence for the SM+PI Optimisation With Waves

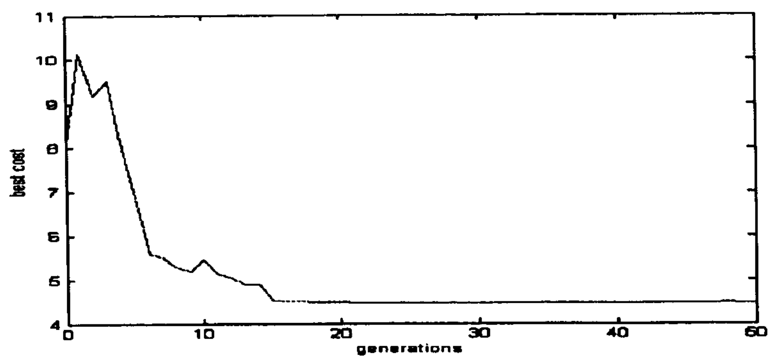


Figure C.7: GA Convergence for the SM Optimisation Without Waves

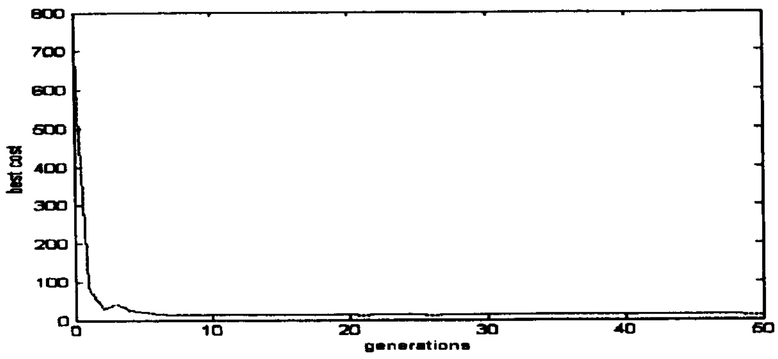


Figure C.8: GA Convergence for the SM Optimisation With Waves

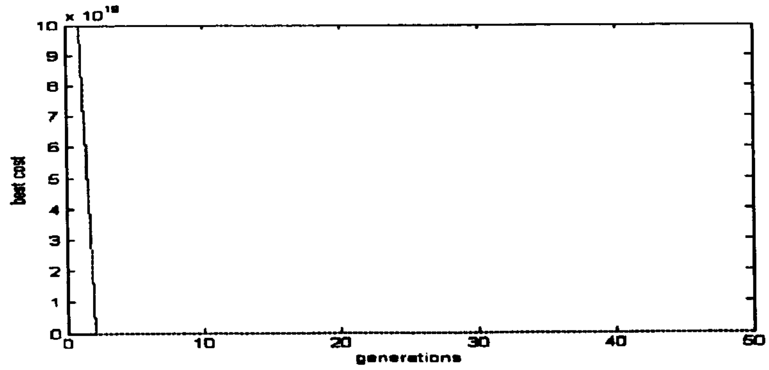


Figure C.9: GA Convergence for the H_∞ Optimisation Without Waves

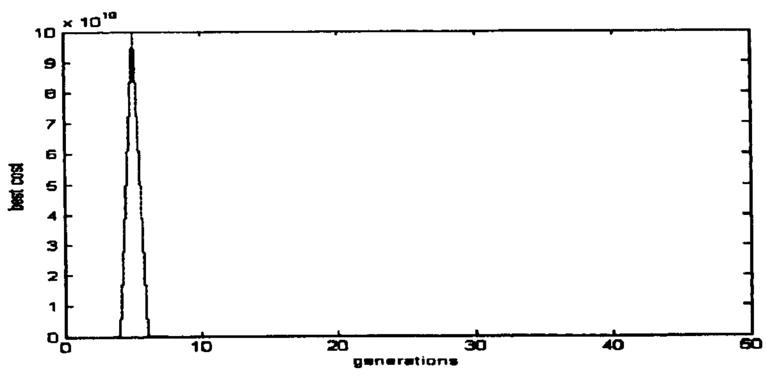


Figure C.10: GA Convergence for the H_∞ Optimisation With Waves

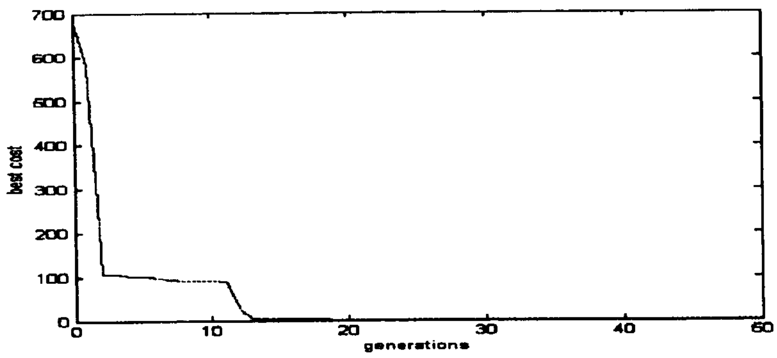


Figure C.11: sGA Convergence for the H_∞ Optimisation Without Waves

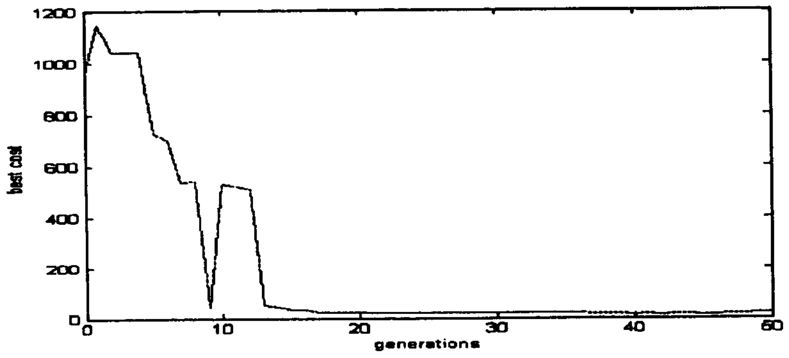


Figure C.12: sGA Convergence for the H_∞ Optimisation With Waves

APPENDIX D

GENETIC PROGRAMMING BEST RESULTS

D.1. RESULTS OBTAINED IN THE GP+RG OPTIMISATION WITHOUT WAVES

Run No. 2

$$\tau_{1com} = 90.656 \cdot \varepsilon_p$$

$$\tau_{3com} = \varepsilon_h - 54.0658 \cdot \tanh\left(\frac{\mathbf{h}_h^{'} \cdot (\mathbf{x}_h - \mathbf{x}_{hd})}{7.0042}\right)$$

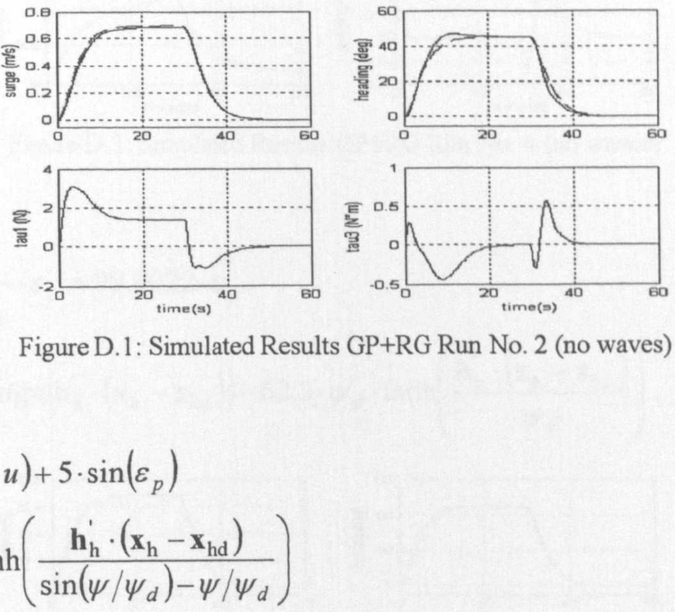


Figure D.1: Simulated Results GP+RG Run No. 2 (no waves)

Run No. 3

$$\tau_{1com} = 7 \cdot (u_d - u) + 5 \cdot \sin(\varepsilon_p)$$

$$\tau_{3com} = \frac{\psi}{\psi_d} \cdot \tanh\left(\frac{\mathbf{h}_h^{'} \cdot (\mathbf{x}_h - \mathbf{x}_{hd})}{\sin(\psi/\psi_d) - \psi/\psi_d}\right)$$

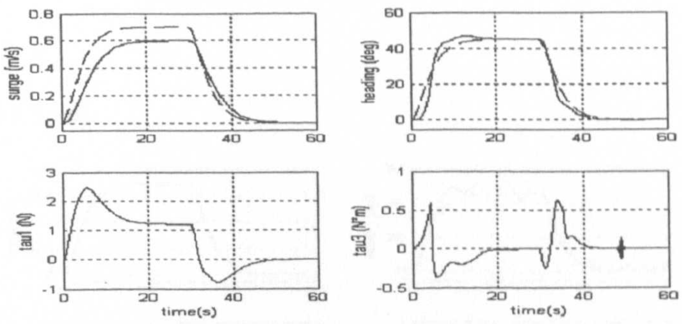


Figure D.2: Simulated Results GP+RG Run No. 3 (no waves)

Run No. 4

$$\tau_{1com} = u + u_d$$

$$\tau_{3com} = 345.8157 \cdot \tanh\left(\frac{\mathbf{h}_h' \cdot (\mathbf{x}_h - \mathbf{x}_{hd})}{-55.4707} \right)$$

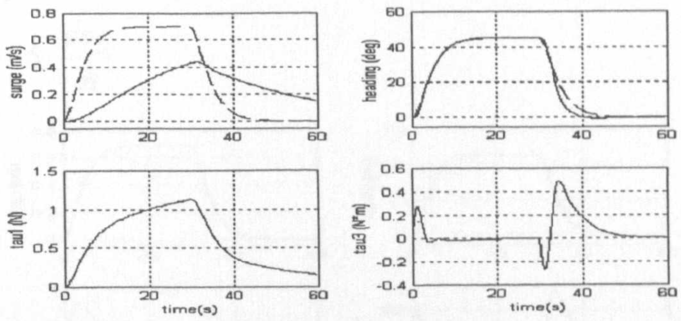


Figure D.3: Simulated Results GP+RG Run No. 4 (no waves)

Run No. 5

$$\tau_{1com} = \frac{7153}{71.6 + u_d} \cdot \varepsilon_p \approx 99.9022 \cdot \varepsilon_p$$

$$\tau_{3com} = \sin(\psi) \cdot sign(\mathbf{h}_h' \cdot (\mathbf{x}_h - \mathbf{x}_{hd})) - 62.3 \cdot \psi_d \cdot \tanh\left(\frac{\mathbf{h}_h' \cdot (\mathbf{x}_h - \mathbf{x}_{hd})}{\psi_d} \right)$$

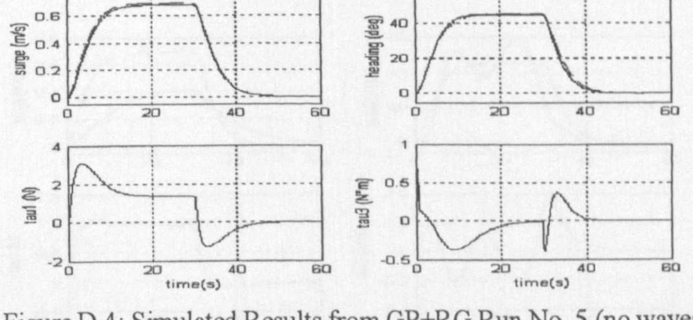


Figure D.4: Simulated Results from GP+RG Run No. 5 (no waves)

Run No. 6

$\tau_{1com} = 85.9718 \cdot \varepsilon_p$

$\tau_{3com} = 0.1833 + 5 \cdot \varepsilon_h$

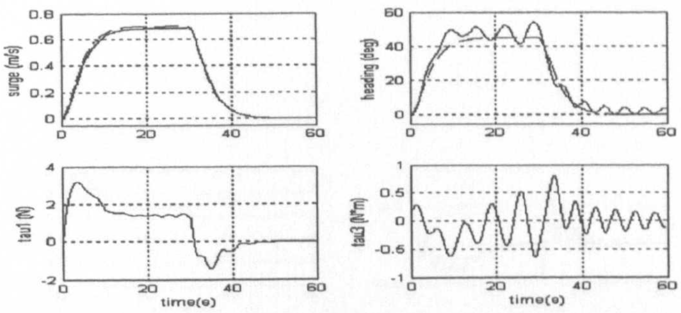


Figure D.5: Simulated Results from GP+RG Run No. 6 (no waves)

Run No. 7

$\tau_{1com} = 72.2 \cdot \varepsilon_p$

$\tau_{3com} = 17 \cdot \varepsilon_h + 5 \cdot \dfrac{d\varepsilon_h}{dt}$

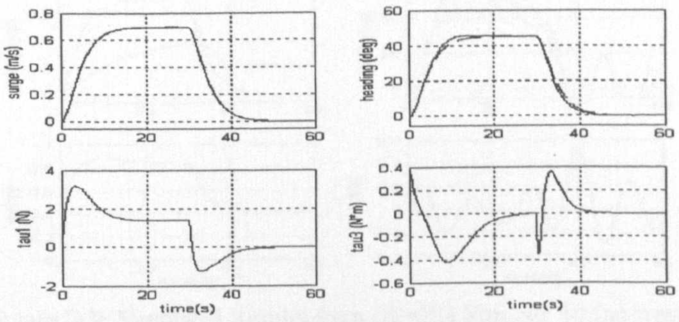


Figure D.6: Simulated Results from GP+RG Run No. 7 (no waves)

Run No. 8

$$\tau_{1com} = \varepsilon_p + 5 \cdot \frac{d\varepsilon_p}{dt} + \frac{d(d\varepsilon_p)}{dt^2} + u_d + \frac{du_d}{dt} + \frac{d(du)}{dt^2} \approx \varepsilon_p + 5 \cdot \frac{d\varepsilon_p}{dt} + u_d + \frac{du_d}{dt}$$

$\tau_{3com} = \varepsilon_h$

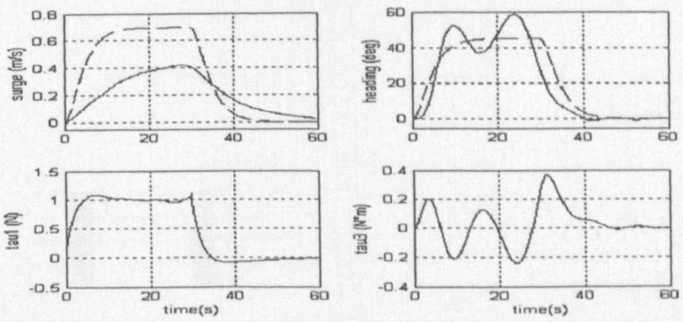


Figure D.7: Simulated Results from GP+RG Run No. 8 (no waves)

Run No. 9

$$\tau_{1com} = 29 \cdot \varepsilon_p + \sin(u_d)$$

$$\tau_{3com} = 12 \cdot \varepsilon_h + d\varepsilon_h/dt$$

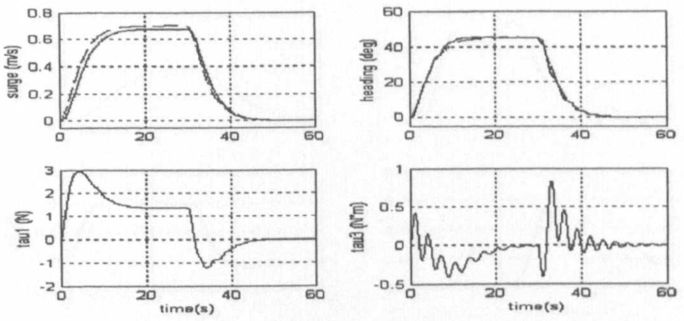


Figure D.8: Simulated Results from GP+RG Run No. 9 (no waves)

Run No. 10

$$\tau_{1com} = \sin(\varepsilon_p - 5.6432)$$

$$\tau_{3com} = \sin(\varepsilon_h)$$

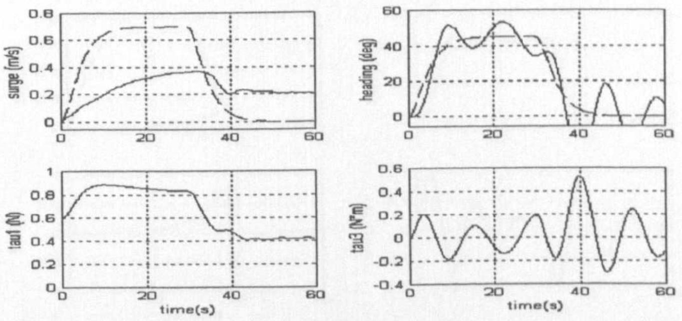


Figure D.9: Simulated Results from GP+RG Run No. 10 (no waves)

Run No. 11

$$\tau_{1com} = \sin(\sin(\sin(76.3) + \sin(\sin(\sin(\sin(\sin(\sin(y)))))))) + y$$

$$y = \sin(\sin(\sin(96.3 \cdot \varepsilon_h \cdot sign(h_p'(u - u_d)))))$$

$$\tau_{3com} = \varepsilon_h$$

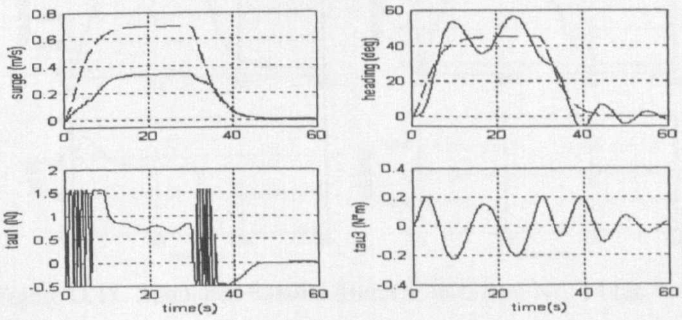


Figure D.10: Simulated Results from GP+RG Run No. 11 (no waves)

Run No. 12

$\tau_{1com} = \varepsilon_p + u_d$

$\tau_{3com} = \sin(\varepsilon_h)$

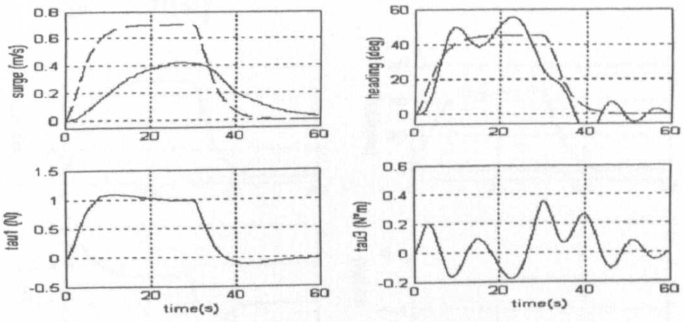


Figure D.11: Simulated Results from GP+RG Run No. 12 (no waves)

Run No. 13

$\tau_{1com} = -0.13 \cdot u + \int 0.065 \cdot u \, dt - 6 \cdot u \cdot sign(h_u^{'}(u - u_d))$

$\tau_{3com} = \varepsilon_h$

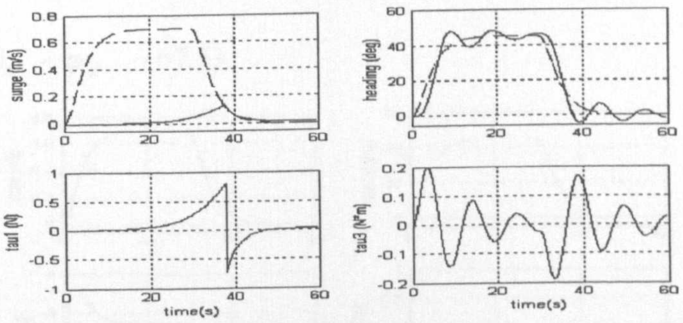


Figure D.12: Simulated Results from GP+RG Run No. 13 (no waves)

Run No. 14

$\tau_{1com} = 387.8184 \cdot \varepsilon_p + 0.0172 \cdot u_d \cdot (u_d + \varepsilon_p) \approx 387.8184 \cdot \varepsilon_p$

$\tau_{3com} = 0$

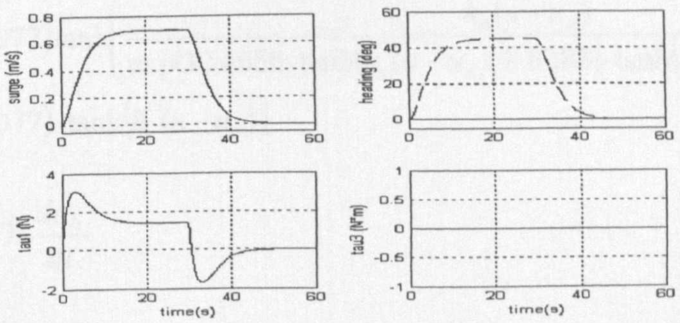


Figure D.13: Simulated Results from GP+RG Run No. 14 (no waves)

Run No. 15

$$\tau_{1com} = 302.5 \cdot \varepsilon_p + 49.2 \cdot \int \varepsilon_p \cdot dt + 2 \cdot \varepsilon_p{}^2 \approx 302.5 \cdot \varepsilon_p + 49.2 \cdot \int \varepsilon_p \cdot dt$$

$$\tau_{3com} = -\mathbf{k} \cdot \mathbf{x_h}$$

$$\mathbf{k} \equiv place(0, \quad -\psi \quad -70.8)$$

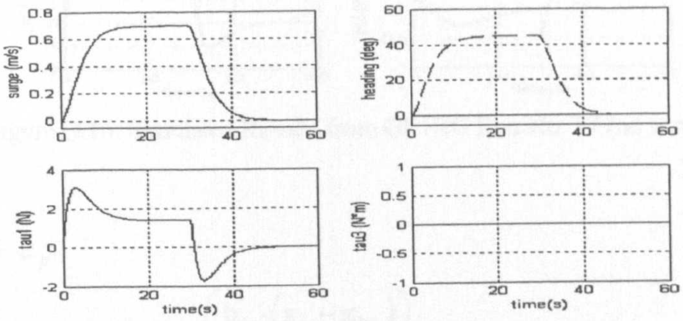


Figure D.14: Simulated Results from GP+RG Run No. 15 (no waves)

Run No. 16

$$\tau_{1com} = \left(2 \cdot \int 5.536 \cdot dt + 28.06\right) \cdot \varepsilon_p + \int \left(\int 5.536 \cdot dt + 16.9881\right) \cdot \varepsilon_p \cdot dt$$

$$\tau_{3com} = -\mathbf{k} \cdot \mathbf{x_h}$$

$$\mathbf{k} \equiv place(0, \quad -\psi_d \quad -67.5)$$

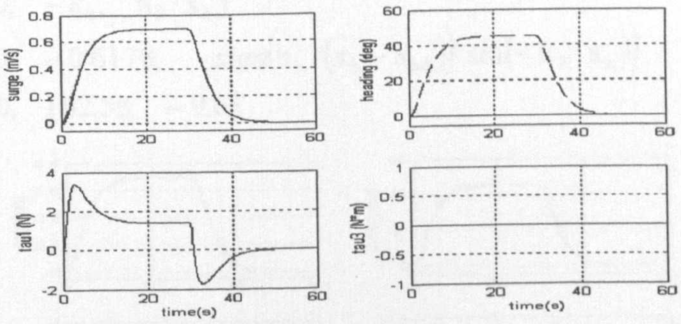


Figure D.15: Simulated Results from GP+RG Run No. 16 (no waves)

Run No. 17

$$\tau_{1com} = (u - 24.077)\tanh\left(\frac{h_u'(u-u_d)}{\exp\left(30.6658 \cdot \tanh\left(h_u'(u-u_d)/9.8288\right) \cdot \tanh\left(h_u'(u-u_d)\right)/(u+u_d)\right)}\right) =$$

$$= (u - 24.077) \cdot \tanh\left(h_u'(u-u_d)\right)$$

$$\tau_{3com} = 39.8207 \cdot \frac{d\varepsilon_h}{dt}$$

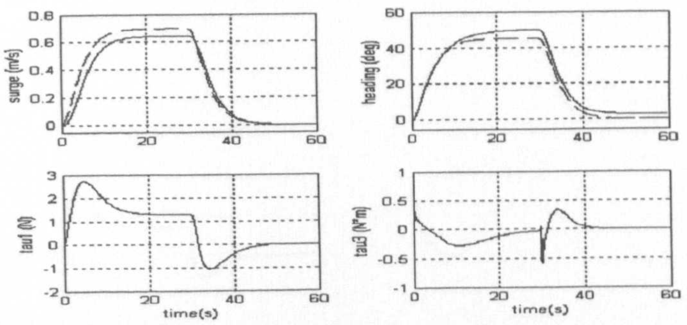


Figure D.16: Simulated Results from GP+RG Run No. 17 (no waves)

Run No. 18

$$\tau_{1com} = 424.83 \cdot \varepsilon_p$$

$$\tau_{3com} = (-\mathbf{k}_1 \cdot \mathbf{x}_h - 326.4) \cdot \tanh\left(\frac{\mathbf{h}_h^{'} \cdot (\mathbf{x}_h - \mathbf{x}_{hd})}{9.6}\right)$$

$$\mathbf{k}_1 \equiv place(0, \quad -\varepsilon_h, \quad \mathbf{k}_2 \cdot \mathbf{x}_h) \wedge \mathbf{k}_2 \equiv place(0, \quad \mathbf{k}_3 \cdot \mathbf{x}_h + 81.6, \quad -9.6)$$

$$\mathbf{k}_3 \equiv place(0, \quad -\varepsilon_h, \quad \mathbf{k}_4 \cdot \mathbf{x}_h)$$

$$\mathbf{k}_4 \equiv place\left(0, \quad (\mathbf{k}_5 \cdot \mathbf{x}_h + 244.8) \cdot \tanh\left(\mathbf{h}_h^{'} \cdot (\mathbf{x}_h - \mathbf{x}_{hd})/9.6\right) + 81.6, \quad -9.6\right)$$

$$\mathbf{k}_5 \equiv place(0, \quad -\varepsilon_h, \quad \mathbf{k}_6 \cdot \mathbf{x}_h + 81.6) \wedge \mathbf{k}_6 \equiv place(0, \quad -0.6178, \quad \mathbf{k}_7 \cdot \mathbf{x}_h)$$

$$\mathbf{k}_7 \equiv place(0, \quad -\varepsilon_h, \quad \mathbf{k}_8 \cdot \mathbf{x}_h)$$

$$\mathbf{k}_8 \equiv place\left(0, \quad -0.6178, \quad -sign\left(\mathbf{h}_h^{'} \cdot (\mathbf{x}_h - \mathbf{x}_{hd})\right) \cdot \sin(-\mathbf{k}_9 \cdot \mathbf{x}_h)\right)$$

$$\mathbf{k}_9 \equiv place(0, \quad 162.58, \quad -9.6)$$

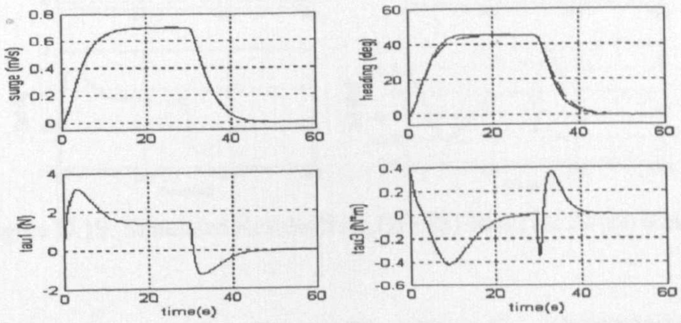


Figure D.17: Simulated Results from GP+RG Run No18 (no waves)

Run No. 19

$$\tau_{1com} = 425.5 \cdot \varepsilon_p + \sin(u_d)$$

$$\tau_{3com} = \varepsilon_h \cdot \frac{\int 87.1 \cdot dt}{\exp(\varepsilon_h)} \cdot sign\left(\mathbf{h}_h^{'} \cdot (\mathbf{x}_h - \mathbf{x}_{hd})\right) \cdot \tanh\left(\frac{\mathbf{h}_h^{'} \cdot (\mathbf{x}_h - \mathbf{x}_{hd})}{\exp(\varepsilon_h)}\right)$$

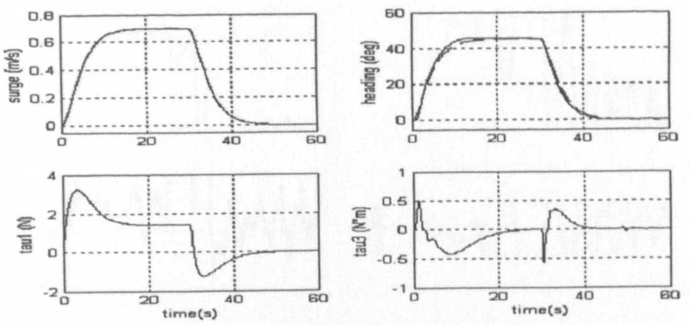


Figure D.18: Simulated Results from GP+RG Run No. 19 (no waves)

Run No. 20

$$\tau_{1com} = 344\cdot\varepsilon_p + 3\cdot\frac{d\varepsilon_p}{dt} + 34.5\cdot u\cdot\tanh\left(\frac{h_p^{'}\cdot(u-u_d)}{u_d}\right)$$

$$\tau_{3com} = \left(-\mathbf{k}_1\cdot\mathbf{x}_\mathrm{h}\cdot\tanh\left(\frac{\mathbf{h}_\mathrm{h}^{'}\cdot(\mathbf{x}_\mathrm{h}-\mathbf{x}_\mathrm{hd})}{-68.2}\right)-31.2\right)\cdot\tanh\left(\frac{\mathbf{h}_\mathrm{h}^{'}\cdot(\mathbf{x}_\mathrm{h}-\mathbf{x}_\mathrm{hd})}{0.9511}\right)$$

$$\mathbf{k}_1 \equiv place\left(0,\quad 94.6\cdot sign\left(\mathbf{h}_\mathrm{h}^{'}\cdot(\mathbf{x}_\mathrm{h}-\mathbf{x}_\mathrm{hd})\right),\quad \mathbf{k}_2\cdot\mathbf{x}_\mathrm{h}\right)$$
$$\mathbf{k}_2 \equiv place\left(0,\quad 94.6\cdot sign\left(\mathbf{h}_\mathrm{h}^{'}\cdot(\mathbf{x}_\mathrm{h}-\mathbf{x}_\mathrm{hd})\right),\quad \mathbf{k}_3\cdot\mathbf{x}_\mathrm{h}+\psi_d\right)$$
$$\mathbf{k}_3 \equiv place\left(0,\quad -0.9511,\quad \mathbf{k}_4\cdot\mathbf{x}_\mathrm{h}\right)\wedge\mathbf{k}_4 \equiv place\left(0,\quad -0.9511,\quad -d\psi/dt\right)$$

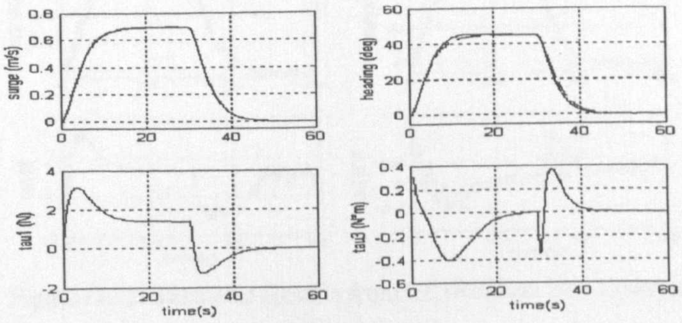


Figure D.19: Simulated Results from GP+RG Run No. 20 (no waves)

D.2. RESULTS OBTAINED IN THE GP+RG OPTIMISATION WITH WAVES

Run No. 2

$$\tau_{1com} = \exp\left(\sin\left(-114.97\cdot\tanh\left(\frac{h_p^{'}\cdot(u-u_d)}{86.2037}\right)\right)\right) \approx -114.97\cdot\tanh\left(\frac{h_p^{'}\cdot(u-u_d)}{86.2037}\right)+1$$

$$\tau_{3com} = 2340.9\cdot\tanh\left(\frac{\mathbf{h}_\mathrm{h}^{'}\cdot(\mathbf{x}_\mathrm{h}-\mathbf{x}_\mathrm{hd})}{-70.3959}\right)$$

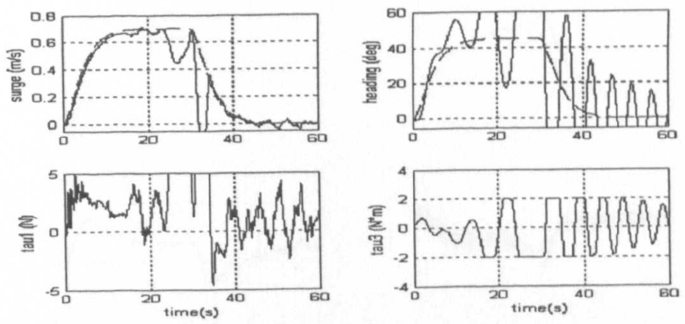


Figure D.20: Simulated Results from GP+RG Run No. 2 (waves)

Run No. 3

$$\tau_{1com} = \sin\left(94.9 \cdot \tanh\left(\frac{h_p^{'} \cdot (u - u_d)}{z_1}\right)\right)$$

$$z_i = 94.9 \cdot \tanh\left(\frac{h_p^{'} \cdot (u - u_d)}{z_{i+1}}\right) \forall i \in [1,9] \ldots z_{10} = 94.9 \cdot \tanh\left(\frac{h_p^{'} \cdot (u - u_d)}{\exp(u)}\right)$$

$$\tau_{3com} = -846.7032 \cdot \tanh\left(\frac{\mathbf{h}_{\mathrm{h}}^{'} \cdot (\mathbf{x}_{\mathrm{h}} - \mathbf{x}_{\mathrm{hd}})}{67.5236}\right)$$

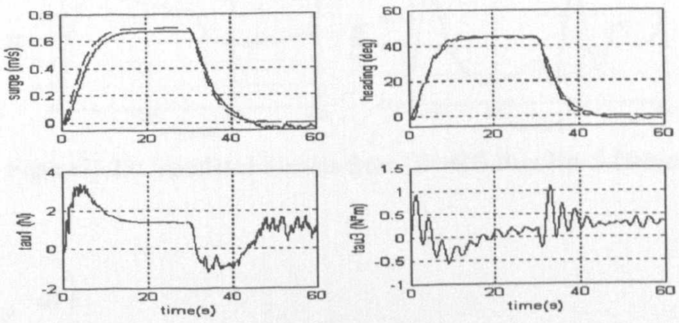


Figure D.21: Simulated Results from GP+RG Run No. 3 (waves)

Run No. 4

$$\tau_{1com} = 8 \cdot \varepsilon_p + \exp\left(\int \varepsilon_p \cdot dt\right)$$

$$\tau_{3com} = -4144.9 \cdot \tanh\left(\frac{\mathbf{h}_{\mathrm{h}}^{'} \cdot (\mathbf{x}_{\mathrm{h}} - \mathbf{x}_{\mathrm{hd}})}{-1891.6 \cdot \tanh\left(\mathbf{h}_{\mathrm{h}}^{'} \cdot (\mathbf{x}_{\mathrm{h}} - \mathbf{x}_{\mathrm{hd}}) / \exp(\psi)\right) - 4144.9}\right) - 2163.8$$

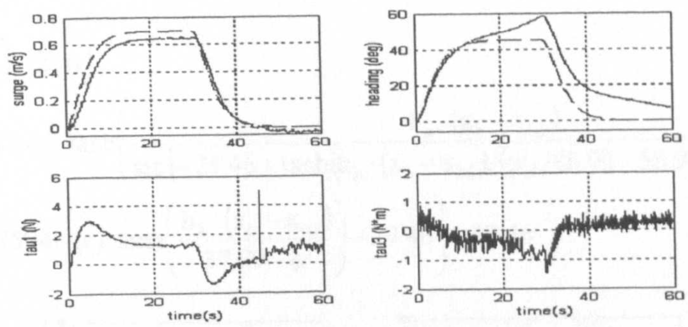


Figure D.22: Simulated Results from GP+RG Run No. 4 (waves)

Run No. 5

$$\tau_{1com} = 11.1 \cdot \varepsilon_p + 33.3 \cdot \int \varepsilon_p \cdot dt$$

$$\tau_{3com} = -1208.4 \cdot \tanh\left(\frac{\mathbf{h}_h^{'} \cdot (\mathbf{x}_h - \mathbf{x}_{hd})}{82.6268} \right)$$

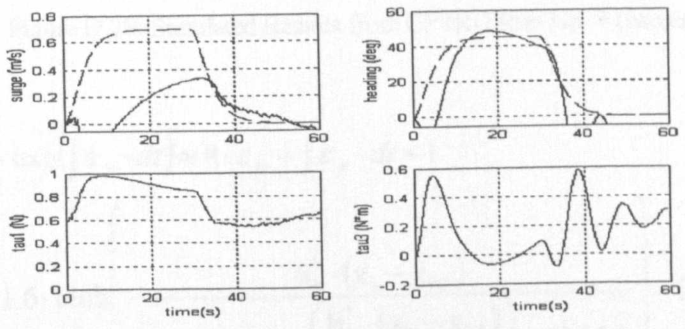


Figure D.23: Simulated Results from GP+RG Run No. 5 (waves)

Run No. 6

$$\tau_{1com} = \exp\left(\int \varepsilon_p \cdot dt\right)$$

$$\tau_{3com} = (\sin((\exp((\mathbf{k} \cdot \mathbf{x}_h + \exp(\sin(\varepsilon_h)) - \sin(\varepsilon_h) + 96.7) \cdot z) - 96.7) \cdot z) - 386.8) \cdot z \approx$$
$$\approx ((((\mathbf{k} \cdot \mathbf{x}_h + 97.7) \cdot z) - 95.7) \cdot z - 386.8) \cdot z \approx -386.8 \cdot z$$
$$z = \tanh\left(\mathbf{h}_h^{'} \cdot (\mathbf{x}_h - \mathbf{x}_{hd})/37\right) \wedge \mathbf{k} \equiv place(0 \quad -93.5 \quad -3.2)$$

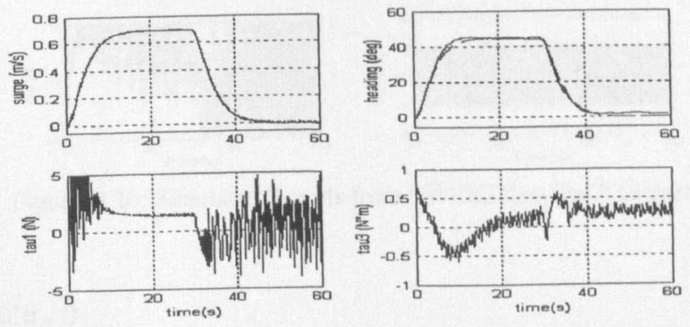


Figure D.24: Simulated Results from GP+RG Run No. 6 (waves)

Run No. 7

$$\tau_{1com} = \exp(\varepsilon_p) - 2 \cdot u^2$$

$$\tau_{3com} = (67.3594 - \psi) \cdot \tanh\left(\frac{\mathbf{h}_h' \cdot (\mathbf{x}_h - \mathbf{x}_{hd})}{\sin\left(-29.46 + \tanh\left(\mathbf{h}_h' \cdot (\mathbf{x}_h - \mathbf{x}_{hd}) / (\psi_d / 86.95 - 58.93)\right) \cdot z\right) - 29.46} \right)$$

$$z = \sin\left((67.3594 - \psi) \cdot \tanh\left(\frac{\mathbf{h}_h' \cdot (\mathbf{x}_h - \mathbf{x}_{hd})}{37.89 - \psi} \right) - 29.46 \right) - 29.46$$

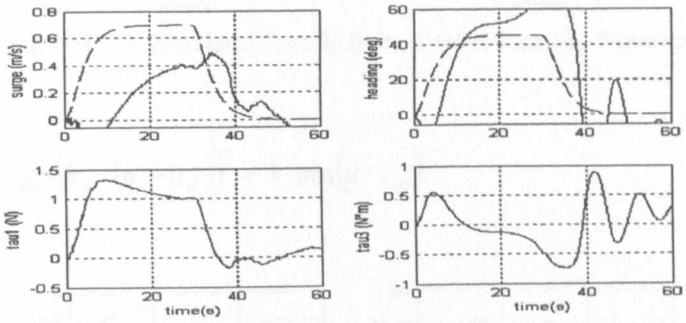


Figure D.25: Simulated Results from GP+RG Run No. 7 (waves)

Run No. 8

$$\tau_{1com} = 8 \cdot \varepsilon_p + \exp\left(\int \varepsilon_p \cdot dt\right) \approx 8 \cdot \varepsilon_p + \int \varepsilon_p \cdot dt + 1$$

$$\tau_{3com} = \left(-1891.6 \cdot \tanh\left(\frac{\mathbf{h}_h' \cdot (\mathbf{x}_h - \mathbf{x}_{hd})}{-4144.4 \cdot \tanh\left(\frac{\mathbf{h}_h' \cdot (\mathbf{x}_h - \mathbf{x}_{hd})}{\exp(\psi)} \right) - 4144.9} \right) - 2163.8 \right) \cdot$$

$$\cdot \tanh\left(\frac{\mathbf{h}_h' \cdot (\mathbf{x}_h - \mathbf{x}_{hd})}{182.7348} \right) \approx -2163.6 \cdot \tanh\left(\frac{\mathbf{h}_h' \cdot (\mathbf{x}_h - \mathbf{x}_{hd})}{182.7348} \right)$$

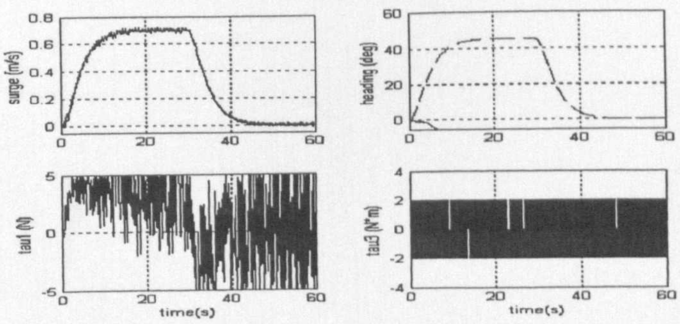


Figure D.26: Simulated Results from GP+RG Run No. 8 (waves)

Run No. 9

$$\tau_{1com} = \sin(\exp(u_d))$$

$$\tau_{3com} = \sin(\sin(\psi_d - \psi))$$

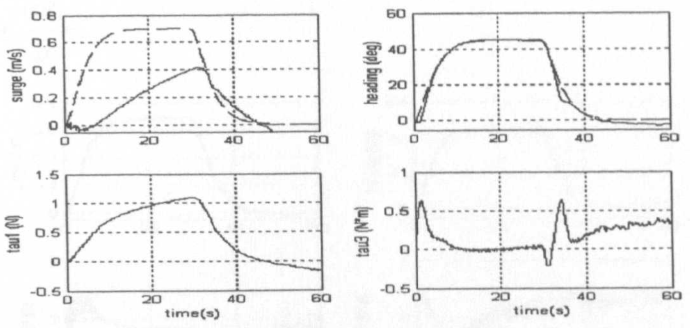


Figure D.27: Simulated Results from GP+RG Run No. 9 (waves)

Run No. 10

$$\tau_{1com} = 2 \cdot u_d \cdot sign\left(h_p^{'} \cdot (u - u_d)\right)^2 - 3 \cdot \sin\left(u \cdot \varepsilon_p\right)$$

$$\tau_{3com} = \psi_d - \psi$$

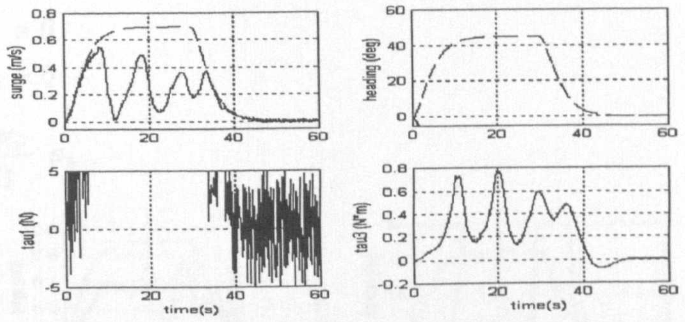


Figure D.28: Simulated Results from GP+RG Run No. 10 (waves)

Run No. 11

$$\tau_{1com} = \varepsilon_p$$

$$\tau_{3com} = \sin\left(\sin\left(6.8782 \cdot \varepsilon_h\right)\right)$$

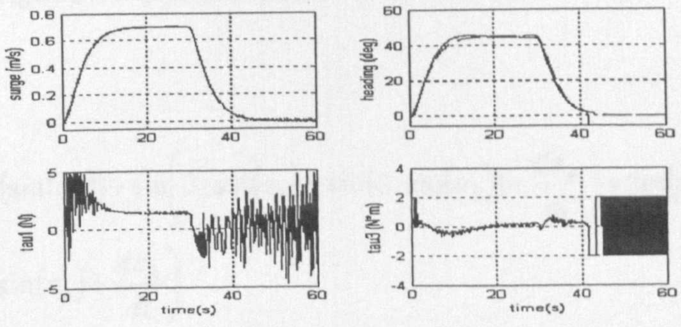


Figure D.29: Simulated Results from GP+RG Run No. 11 (waves)

Run No. 12

$$\tau_{1com} = 1 + 11 \cdot \frac{du_d}{dt}$$

$$\tau_{3com} = 14 \cdot \varepsilon_h + \frac{d\varepsilon_h}{dt}$$

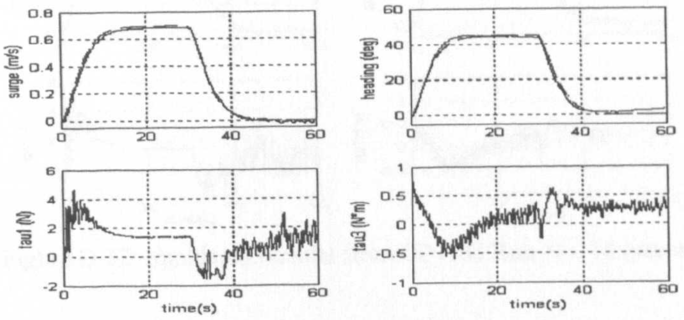


Figure D.30: Simulated Results from GP+RG Run No. 12 (waves)

Run No. 13

$$\tau_{1com} = 2 \cdot u_d + C$$

$$C = \begin{cases} 0 & u_d \neq 0 \\ 1 & u_d = 0 \end{cases}$$

$$\tau_{3com} = \frac{10.7 + \psi_d}{2.1971} \cdot \varepsilon_h$$

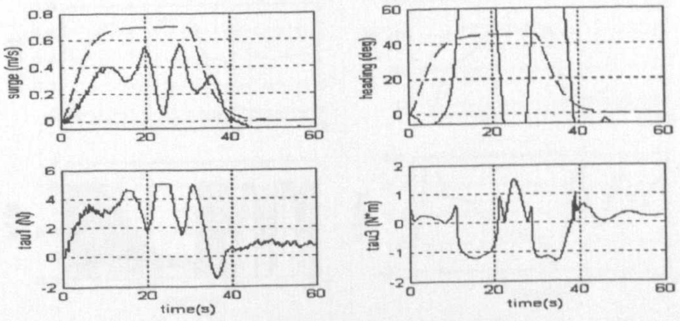


Figure D.31: Simulated Results from GP+RG Run No. 13 (waves)

Run No. 14

$$\tau_{1com} = 1$$

$$\tau_{3com} = \sin\left(\sin(\sin(\varepsilon_h)) + \sin\left(3 \cdot \sin(\varepsilon_h) + \sin(2 \cdot \sin(\varepsilon_h)) + \frac{d\varepsilon_h}{dt}\right) + \sin(\varepsilon_h)\right) +$$

$$+ \sin\left(3 \cdot \sin(\varepsilon_h) + \frac{d\varepsilon_h}{dt}\right)$$

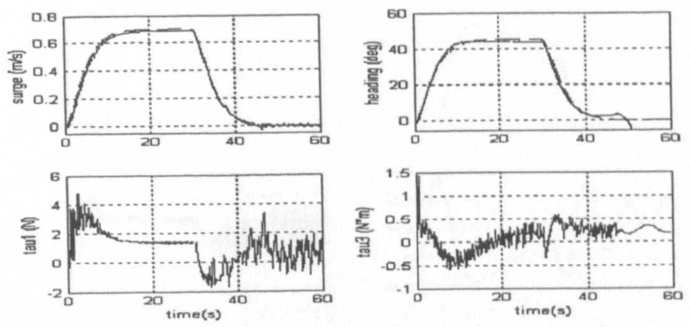


Figure D.32: Simulated Results from GP+RG Run No. 14 (waves)

Run No. 15

$$\tau_{1com} = \exp\left(\frac{u}{68.72} - s\cdot\left(\frac{u}{68.72} - s\cdot(-u - s\cdot\sin(\sin(z)))\right)\right)$$

$$z = -1.0146\cdot u\cdot\tanh\left(\frac{h_p'(u-u_d)}{u/68.72}\right)$$

$$s = sign\left(h_p'(u-u_d)\right)$$

$$\tau_{3com} = \sin(\sin(\psi_d - \psi))$$

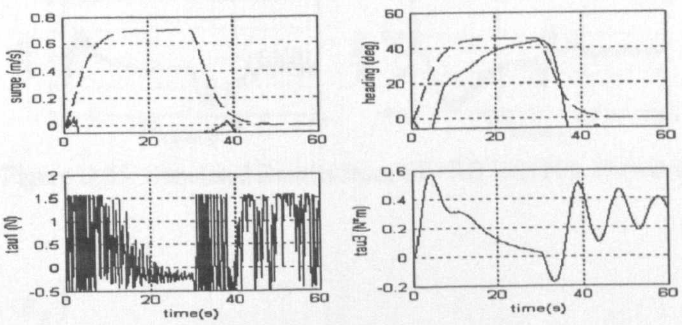


Figure D.33: Simulated Results from GP+RG Run No. 15 (waves)

Run No. 16

$$\tau_{1com} = \int \frac{\exp(u+u_d)}{26.35\cdot t^2\cdot\tanh\left(h_p'(u-u_d)/\exp(u+u_d)\right)}\cdot dt$$

$$\tau_{3com} = 7.9025\cdot\sin(\varepsilon_h)$$

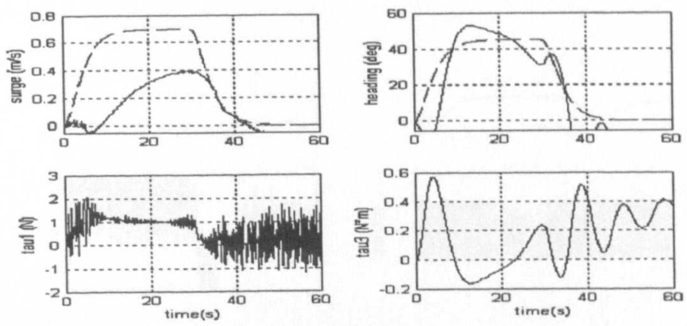


Figure D.34: Simulated Results from GP+RG Run No. 16 (waves)

Run No. 17

$$\tau_{1com} = \varepsilon_p + u_d$$

$$\tau_{3com} = -\mathbf{k}\cdot\mathbf{x}_h$$

$$\mathbf{k} \equiv place\left(0 \quad -30.4\cdot\varepsilon_h - 6\cdot\psi \quad -\sin(30.4\cdot\varepsilon_h + 6\cdot\psi)\right)$$

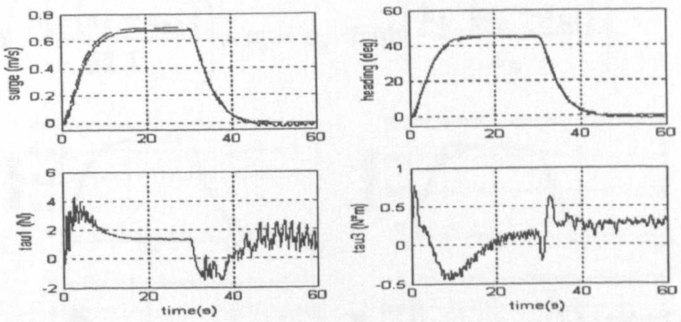


Figure D.35: Simulated Results from GP+RG Run No. 17 (waves)

Run No. 18

$$\tau_{1com} = \exp(u_d\cdot\varepsilon_p)$$

$$\tau_{3com} = -\mathbf{k}_1\cdot\mathbf{x}_h$$

$$\mathbf{k}_1 \equiv place\left(0 \quad \mathbf{k}_2\cdot\mathbf{x}_h \quad -73.5\cdot\tanh\left(\frac{\mathbf{h}_h'(\mathbf{x}_h-\mathbf{x}_{hd})}{25.4}\right)\right)$$

$$\mathbf{k}_2 \equiv place\left(0 \quad -73.5\cdot\psi_d\cdot\tanh\left(\frac{\mathbf{h}_h'(\mathbf{x}_h-\mathbf{x}_{hd})}{25.4}\right) \quad -15.3\right)$$

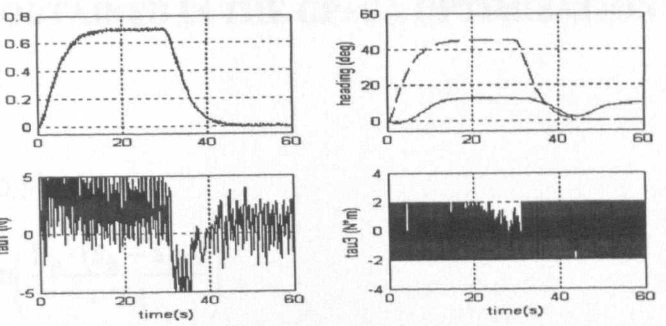


Figure D.36: Simulated Results from GP+RG Run No. 18 (waves)

Run No. 19

$$\tau_{1com} = 56.7 \cdot \varepsilon_p$$

$$\tau_{3com} = \sin(\sin(\sin(\sin(\sin(\sin(\sin(\sin(z_1 \cdot z_2 - 2z_3) - 3z_3)) - 2z_3))) - z_3) - 2z_3)$$

$$z_1 = sign\left(\mathbf{h}_h^{'} \cdot (\mathbf{x}_h - \mathbf{x}_{hd})\right)$$

$$z_2 = \varepsilon_h \cdot \tanh\left(\frac{\mathbf{h}_h^{'} \cdot (\mathbf{x}_h - \mathbf{x}_{hd})}{32.1}\right) \wedge z_3 = \varepsilon_h \cdot \tanh\left(\frac{\mathbf{h}_h^{'} \cdot (\mathbf{x}_h - \mathbf{x}_{hd})}{\varepsilon_h}\right)$$

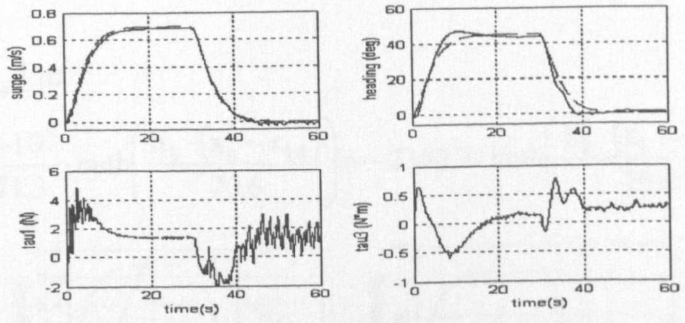


Figure D.37: Simulated Results from GP+RG Run No. 19 (waves)

Run No. 20

$$\tau_{1com} = 71.13 \cdot \varepsilon_p$$

$$\tau_{3com} = \sin\left(\sin\left(\sin\left(\sin\left(\sin\left(\sin\left(3156.55 \cdot s \cdot \tanh\left(\mathbf{h}_h^{'} \cdot (\mathbf{x}_h - \mathbf{x}_{hd}\right)/67.5\right)\right)\right)\right)\right)\right)$$

$$s = sign\left(\mathbf{h}_h^{'} \cdot (\mathbf{x}_h - \mathbf{x}_{hd})\right)$$

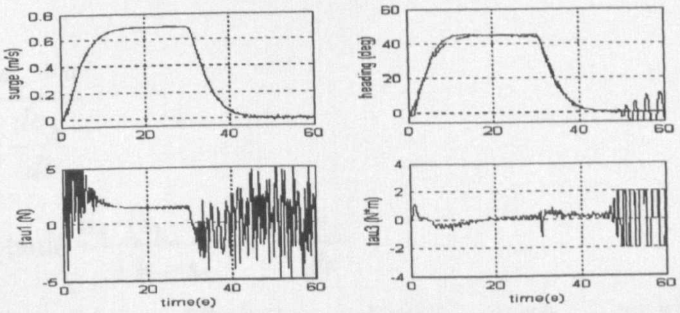


Figure D.38: Simulated Results from GP+RG Run No. 20 (waves)

D.3. RESULTS OBTAINED IN THE GP+GA OPTIMISATION WITHOUT WAVES

Run No. 2

$$\tau_{1com} = u_d \cdot (u + 0.5)$$

$$\tau_{3com} = 57.8 \cdot \tanh\left(\frac{\mathbf{h}_h^{'} \cdot (\mathbf{x}_h - \mathbf{x}_{hd})}{-7.1}\right)$$

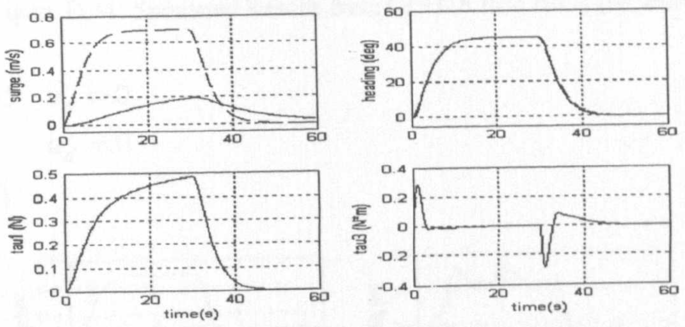


Figure D.39: Simulated Results from GP+GA Run No. 2 (no waves)

Run No. 3

$$\tau_{1com} = \int (\varepsilon_p \cdot u_d) \cdot dt$$

$$\tau_{3com} = \frac{1.6988 \cdot 10^5}{\psi_d - 71.3} \cdot \tanh\left(\frac{\mathbf{h}_h^{'} \cdot (\mathbf{x}_h - \mathbf{x}_{hd})}{70.6}\right) \approx -2382.7 \cdot \tanh\left(\frac{\mathbf{h}_h^{'} \cdot (\mathbf{x}_h - \mathbf{x}_{hd})}{70.6}\right)$$

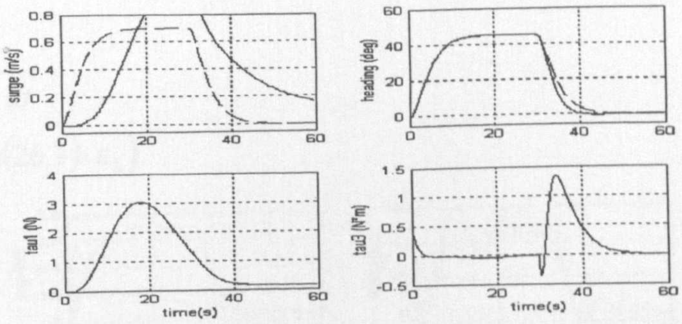


Figure D.40: Simulated Results from GP+GA Run No. 3 (no waves)

Run No. 4

$$\tau_{1com} = \frac{53.9 + \varepsilon_p + u}{68.1 + \frac{d\varepsilon_p}{dt}}$$

$$\tau_{3com} = \mathbf{k}_1 \cdot \mathbf{x}_h \cdot \tanh\left(\frac{\mathbf{h}_h^{'} \cdot (\mathbf{x}_h - \mathbf{x}_{hd})}{-\mathbf{k}_2 \cdot \mathbf{x}_h}\right) - \frac{d\varepsilon_h}{dt}$$

$$\mathbf{k}_1 \equiv place(0, \quad -84.6, \quad -87.4) \wedge \mathbf{k}_2 \equiv place(0, \quad -64.9, \quad -26.4)$$

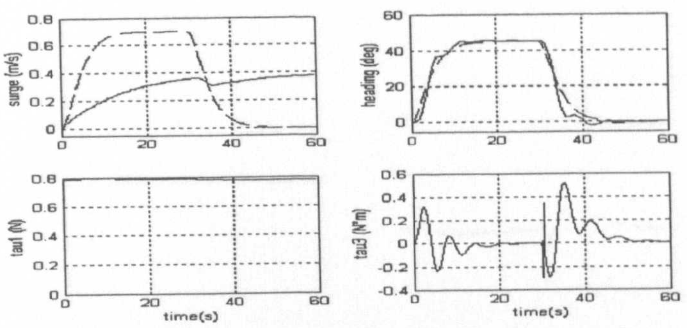


Figure D.41: Simulated Results from GP+GA Run No. 4 (no waves)

Run No. 5

$$\tau_{1com} = \begin{cases} 0.6965 & u_d \neq 0 \\ 1 & u_d = 0 \end{cases}$$

$$\tau_{3com} = \sin(\varepsilon_h)$$

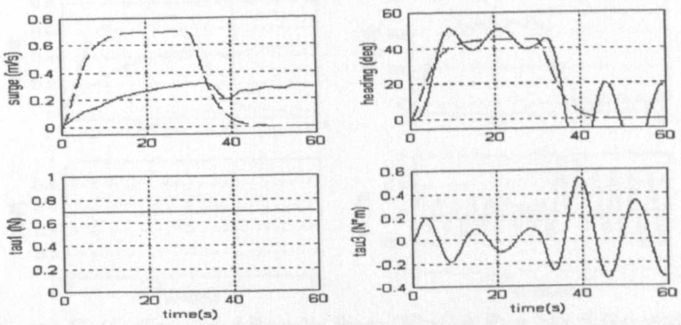


Figure D.42: Simulated Results from GP+GA Run No. 5 (no waves)

Run No. 6

$$\tau_{1com} = \frac{96.3}{\int 20.4 \cdot dt}$$

$$\tau_{3com} = \sin(\sin(26.7) \cdot \varepsilon_h)$$

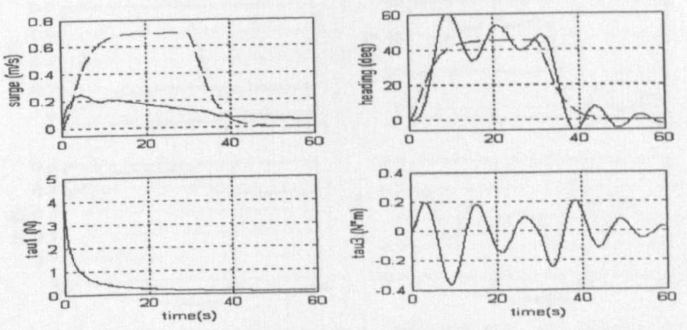


Figure D.43: Simulated Results from GP+GA Run No. 6 (no waves)

Run No. 7

$$\tau_{1com} = \sin(\sin(1)) \approx 0.7456$$

$$\tau_{3com} = \sin(\sin(d(35.7 \cdot \varepsilon_h)/dt))$$

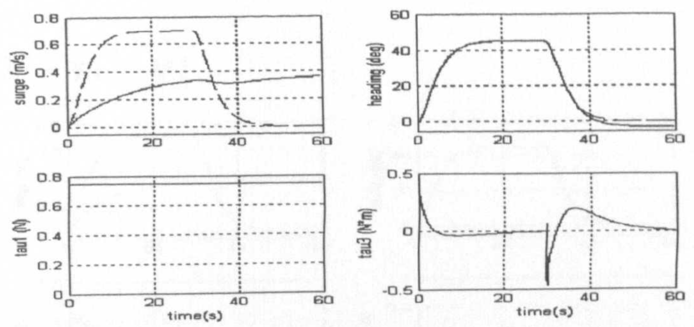


Figure D.44: Simulated Results from GP+GA Run No. 7 (no waves)

Run No. 8

$\tau_{1com} = \varepsilon_p / \varepsilon_p = 1$

$\tau_{3com} = 7 \cdot \varepsilon_h$

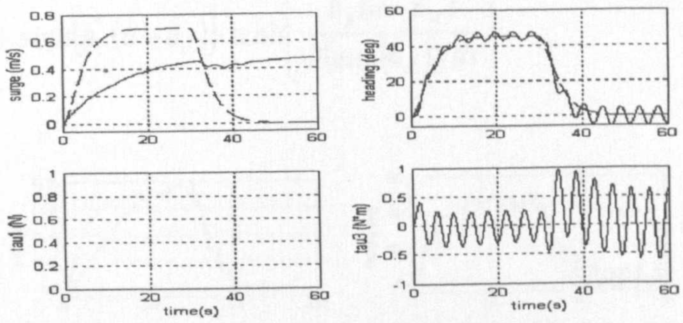


Figure D.45: Simulated Results from GP+GA Run No. 8 (no waves)

Run No. 9

$\tau_{1com} = \exp(\sin(4.7 - u_d))$

$$\tau_{3com} = \psi_d \cdot \tanh\left(\frac{\mathbf{h}_\mathrm{h}' \cdot (\mathbf{x}_\mathrm{h} - \mathbf{x}_\mathrm{hd})}{-20.3}\right) - \sin(\sin(18.8 - \varepsilon_h)) \approx -\sin(\sin(18.8 - \varepsilon_h))$$

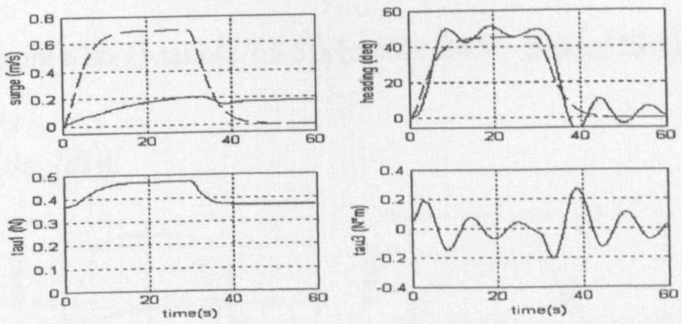


Figure D.39: Simulated Results from GP+GA Run No. 2 (no waves)

Run No. 10

$\tau_{1com} = 1.1 - u$

$$\tau_{3com} = -\mathbf{k}\cdot\mathbf{x}_{\mathrm{h}}$$

$$\mathbf{k} \equiv place(0, \quad -\psi, \quad -\psi)$$

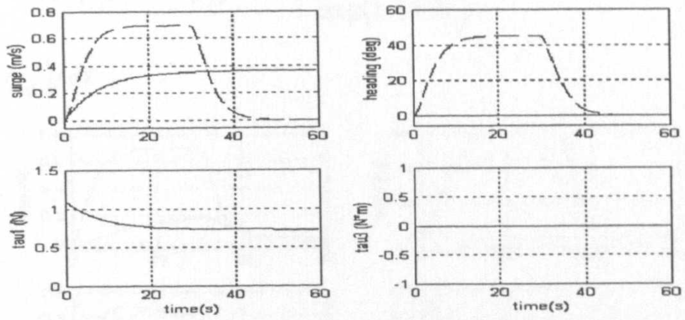


Figure D.47: Simulated Results from GP+GA Run No. 10 (no waves)

Run No. 11

$$\tau_{1com} = \exp(u_d)\cdot sign\left(h_p^{'}(u-u_d)\right)\cdot \tanh\left(\frac{h_p^{'}(u-u_d)}{d(\exp(u_d))/dt}\right)$$

$$\tau_{3com} = 8.4\cdot\varepsilon_h$$

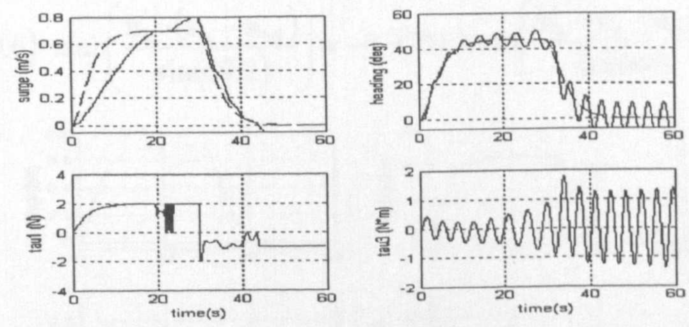


Figure D.48: Simulated Results from GP+GA Run No. 11 (no waves)

Run No. 12

$$\tau_{1com} = 1.12\cdot\left(\left(\tanh(z/86.1)\cdot\tanh(z/65.5)\right)/\left(\tanh(z/24.9)\cdot\tanh(z/76.4)\right)\right)$$

$$z = h_p^{'}(u-u_d)$$

$$\tau_{3com} = \sin(\sin(d\varepsilon_h/dt))$$

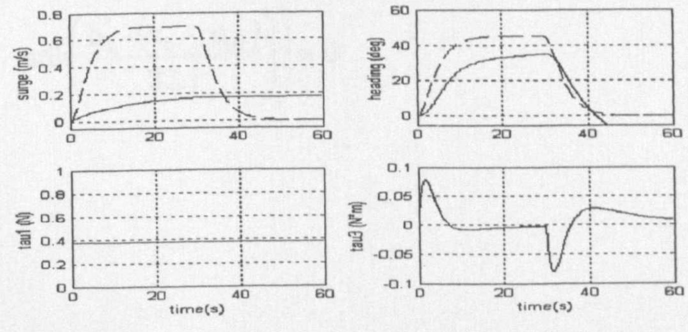


Figure D.49: Simulated Results from GP+GA Run No. 12 (no waves)

Run No. 13

$$\tau_{1com} = 2.3985 \cdot sign\left(h_p'(u-u_d)\right) \cdot \tanh\left(\frac{h_p'(u-u_d)}{\exp(1.119)}\right)$$

$$\tau_{3com} = d(\varepsilon_h + \psi_d)/dt$$

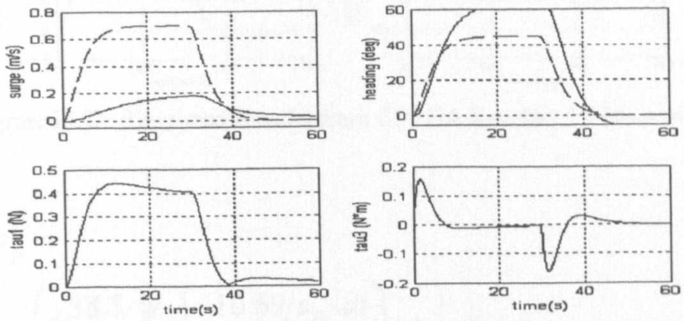


Figure D.50: Simulated Results from GP+GA Run No. 13 (no waves)

Run No. 14

$$\tau_{1com} = 0$$

$$\tau_{3com} = \sin(29.6) \cdot \tanh\left(\frac{\mathbf{h}_h' \cdot (\mathbf{x}_h - \mathbf{x}_{hd})}{\sin(69.2)}\right) \approx -0.9701 \cdot \tanh\left(\frac{\mathbf{h}_h' \cdot (\mathbf{x}_h - \mathbf{x}_{hd})}{0.0849}\right)$$

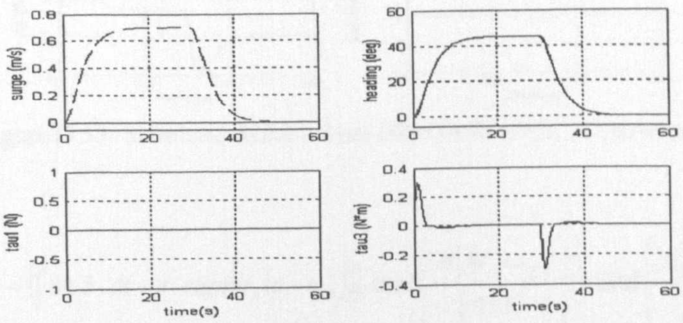


Figure D.51: Simulated Results from GP+GA Run No. 14 (no waves)

Run No. 15

$$\tau_{1com} = \varepsilon_p + u_d$$

$$\tau_{3com} = \sin\left(\psi \cdot \tanh\left(\frac{\mathbf{h}_h' \cdot (\mathbf{x}_h - \mathbf{x}_{hd})}{82.1}\right)\right) \approx 0$$

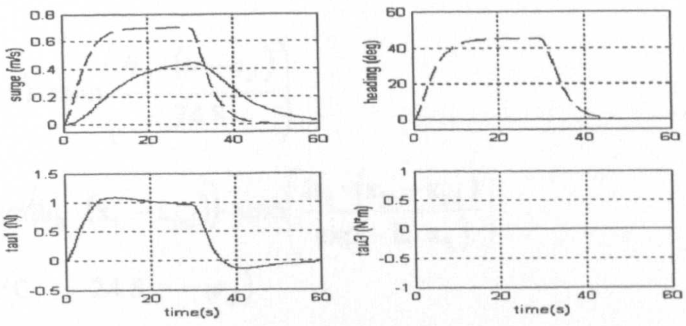


Figure D.52: Simulated Results from GP+GA Run No. 15 (no waves)

Run No. 16

$$\tau_{1com} = 197.5 \cdot \varepsilon_p$$

$$\tau_{3com} = \int \psi \cdot \left(\int \psi \cdot \left(\int \frac{38.5 \cdot \psi \cdot \int -10.69/\varepsilon_h \cdot dt}{\varepsilon_h} \right) \cdot dt \right) \cdot dt \approx 0$$

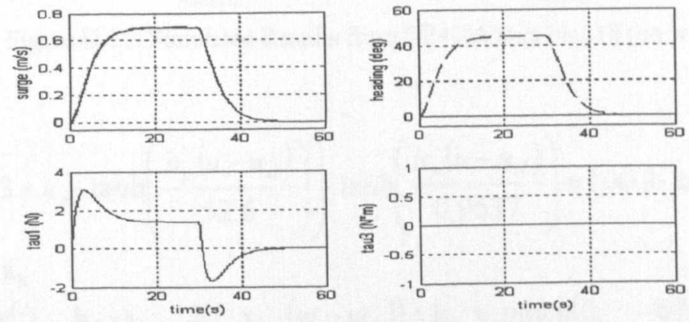


Figure D.53: Simulated Results from GP+GA Run No. 16 (no waves)

Run No. 17

$$\tau_{1com} = \left(\int 90.1 \cdot dt - \left(\int 84.5 \cdot dt - u \cdot sign\left(h_p'(u-u_d)\right)\right) \cdot \tanh\left(\frac{h_p'(u-u_d)}{-\int 89.6 \cdot dt} \right) \right) \cdot \tanh\left(\frac{h_p'(u-u_d)}{-\int 44.1 \cdot dt} \right)$$

$$\tau_{3com} = \sin(\psi + 92.3) \cdot \tanh\left(\frac{\mathbf{h}_h' \cdot (\mathbf{x}_h - \mathbf{x}_{hd})}{\sin(\psi + 57.2)} \right)$$

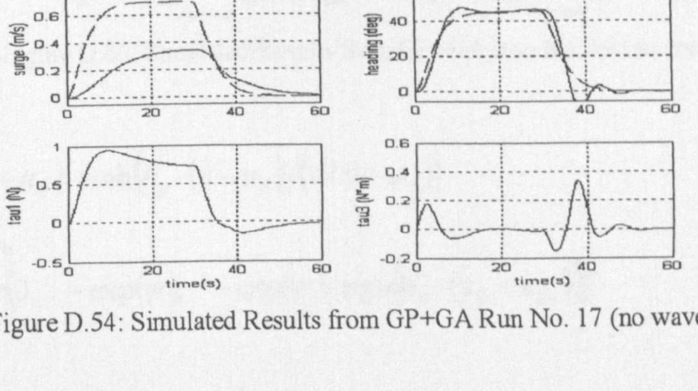


Figure D.54: Simulated Results from GP+GA Run No. 17 (no waves)

Run No. 18

$$\tau_{1com} = (1.4 + u) \cdot \tanh\left(\frac{h_p' \cdot (u - u_d)}{74.8}\right)$$

$$\tau_{3com} = \varepsilon_h \cdot sign\left(\mathbf{h}_h' \cdot (\mathbf{x}_h - \mathbf{x}_{hd})\right) \cdot \tanh\left(\frac{\mathbf{h}_h' \cdot (\mathbf{x}_h - \mathbf{x}_{hd})}{\exp(-\mathbf{k} \cdot \mathbf{x}_h)}\right)$$

$$\mathbf{k} \equiv place(0, \quad -24.8, \quad -\psi_d)$$

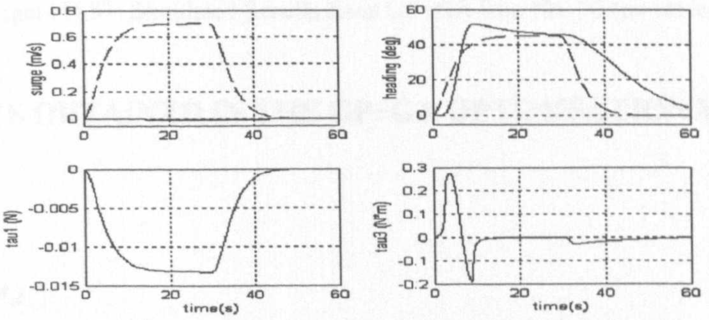


Figure D.55: Simulated Results from GP+GA Run No. 18 (no waves)

Run No. 19

$$\tau_{1com} = \left(100.3 + u_d \cdot \tanh\left(\frac{h_p'(u - u_d)}{52.6}\right)\right) \cdot \tanh\left(\frac{h_p'(u - u_d)}{0.9537}\right) \approx 100.3 \cdot \tanh\left(\frac{h_p' \cdot (u - u_d)}{0.9537}\right)$$

$$\tau_{3com} = -\mathbf{k}_1 \cdot \mathbf{x}_h$$

$$\mathbf{k}_1 \equiv place(0, \quad \mathbf{k}_2 \cdot \mathbf{x}_h, \quad \mathbf{k}_3 \cdot \mathbf{x}_h \cdot (\psi - \psi_d)) \wedge \mathbf{k}_2 \equiv place(0, \quad -67.3, \quad -11.7)$$

$$\mathbf{k}_3 \equiv place(0, \quad -69.7, \quad -43.5)$$

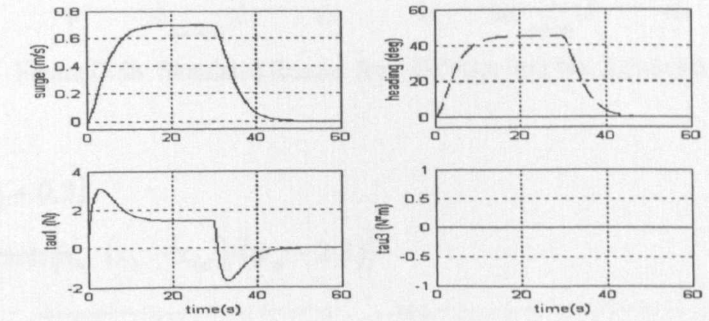


Figure D.56: Simulated Results from GP+GA Run No. 19 (no waves)

Run No. 20

$$\tau_{1com} = (93.3 + u_d) \cdot \tanh\left(h_p' \cdot (u - u_d)/(17.9 + u_d)\right)$$

$$\tau_{3com} = -\mathbf{k} \cdot \mathbf{x}_h$$

$$\mathbf{k} \equiv place\left(0, \quad -\exp(\psi), \quad -\exp(\psi) \cdot sign\left(\mathbf{h}_h' \cdot (\mathbf{x}_h - \mathbf{x}_{hd})\right)\right)$$

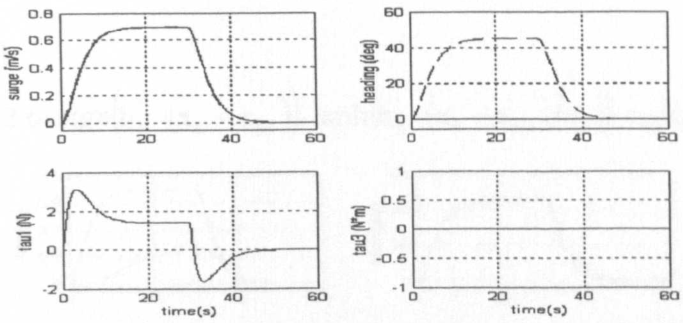


Figure D.57: Simulated Results from GP+GA Run No. 20 (no waves)

D.4. RESULTS OBTAINED IN THE GP+GA OPTIMISATION WITH WAVES

Run No. 2

$$\tau_{1com} = 2.34 \cdot u_d$$

$$\tau_{3com} = \sin\left(56.2 \cdot \tanh\left(z/\tanh\left(z/y\right)\right)\right) \approx 56.2 \cdot \tanh\left(z/\tanh\left(z/y\right)\right)$$

$$y = \sin\left(26.4 \cdot \tanh\left(z/\left(sign(z) \cdot \sin\left(\psi_d - \varepsilon_h\right)\right)\right)\right) \wedge z = \mathbf{h}_\mathrm{h}^{'} \cdot \left(\mathbf{x}_\mathrm{h} - \mathbf{x}_\mathrm{hd}\right)$$

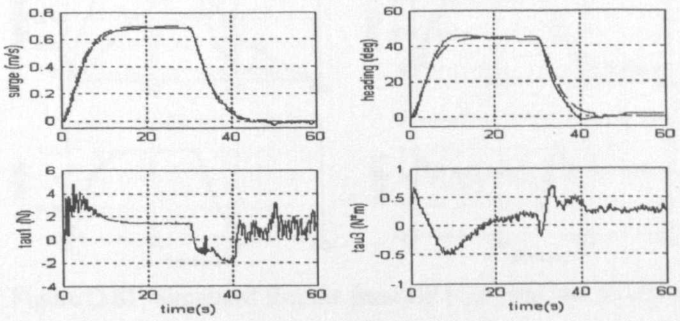


Figure D.58: Simulated Results from GP+GA Run No. 2 (waves)

Run No. 3

$$\tau_{1com} = \varepsilon_p / \left(u_d + 0.3\right)$$

$$\tau_{3com} = 65.6 \cdot \tanh\left(\mathbf{h}_\mathrm{h}^{'} \cdot \left(\mathbf{x}_\mathrm{h} - \mathbf{x}_\mathrm{hd}\right) / \left(\psi_d - 2.3\right)\right)$$

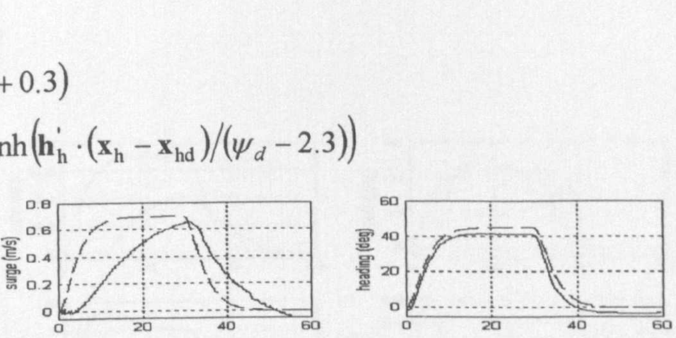


Figure D.59: Simulated Results from GP+GA Run No. 3 (waves)

Run No. 4

$$\tau_{1com} = \sin(u_d)$$

$$\tau_{3com} = \left(12.9 + 2.6 \cdot sign\left(\mathbf{h}_h^{'} \cdot (\mathbf{x}_h - \mathbf{x}_{hd})\right) \cdot \tanh\left(\mathbf{h}_h^{'} \cdot (\mathbf{x}_h - \mathbf{x}_{hd})/\varepsilon_h\right)\right) \cdot \varepsilon_h / \exp(\psi_d)$$

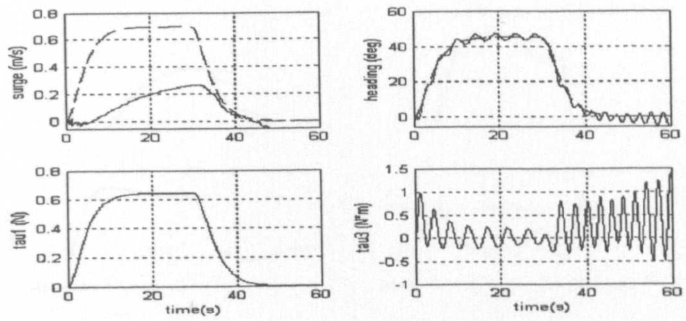


Figure D.60: Simulated Results from GP+GA Run No. 4 (waves)

Run No. 5

$$\tau_{1com} = 2.1534 \cdot u_d$$

$$\tau_{3com} = 2 \cdot \exp(\sin(70.2)) \cdot (2 \cdot \varepsilon_h + \sin(3.1)) \approx 7.1482 \cdot \varepsilon_h + 0.1487$$

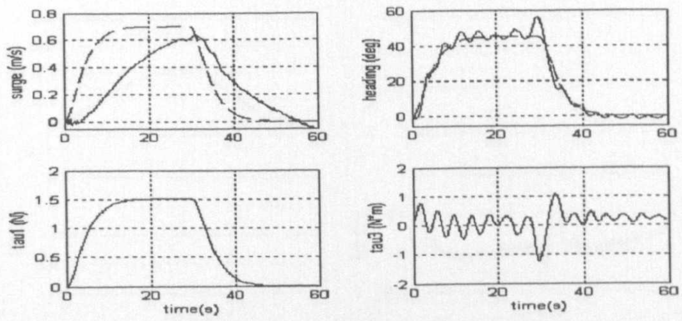


Figure D.61: Simulated Results from GP+GA Run No. 5 (waves)

Run No. 6

$$\tau_{1com} = \varepsilon_p + u_d$$

$$\tau_{3com} = \psi_d - \psi$$

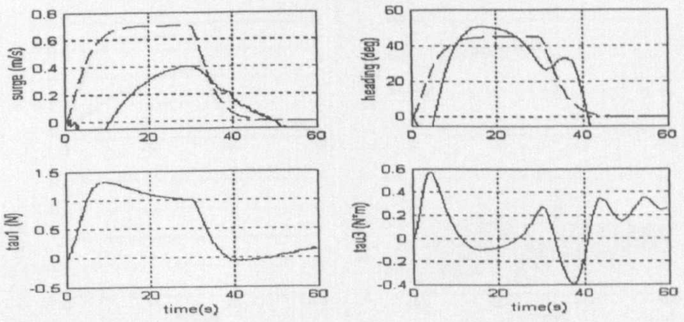


Figure D.62: Simulated Results from GP+GA Run No. 6 (waves)

Run No. 7

$$\tau_{1com} = 2 \cdot \sin(\varepsilon_p) + u$$

$$\tau_{3com} = \varepsilon_h$$

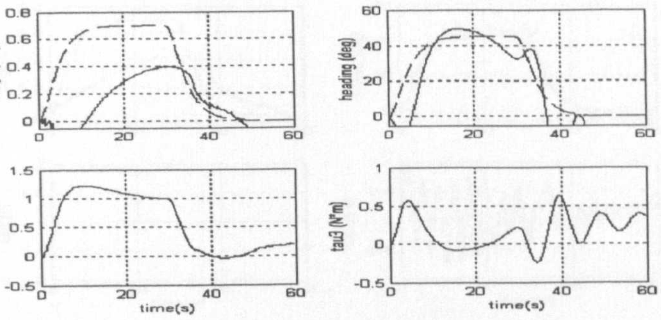


Figure D.63: Simulated Results from GP+GA Run No. 7 (waves)

Run No. 8

$$\tau_{1com} = 1$$

$$\tau_{3com} = (\psi_d - 997.92) \cdot \tanh\left(\mathbf{h}_\mathrm{h}^{'} \cdot (\mathbf{x}_\mathrm{h} - \mathbf{x}_\mathrm{hd})/74\right)$$

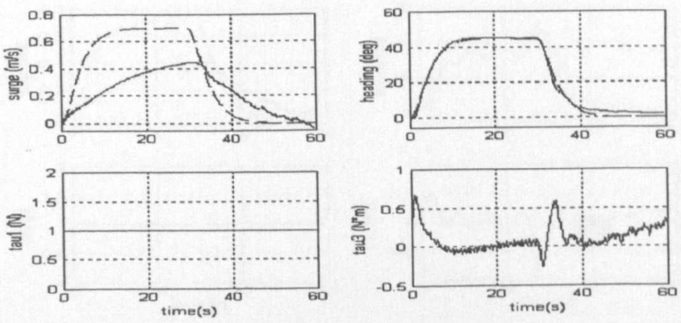


Figure D.64: Simulated Results from GP+GA Run No. 8 (waves)

Run No. 9

$$\tau_{1com} = 1$$

$$\tau_{3com} = \sin(21) \cdot \varepsilon_h / \sin(21.9) \approx 9.1918 \cdot \varepsilon_h$$

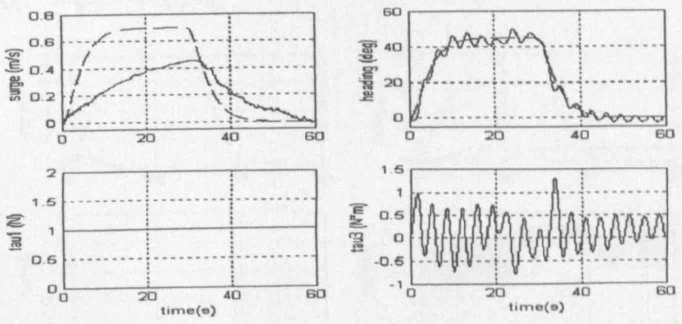


Figure D.65: Simulated Results from GP+GA Run No. 9 (waves)

Run No. 10

$\tau_{1com} = 1$

$\tau_{3com} = 8.7123 \cdot \varepsilon_h$

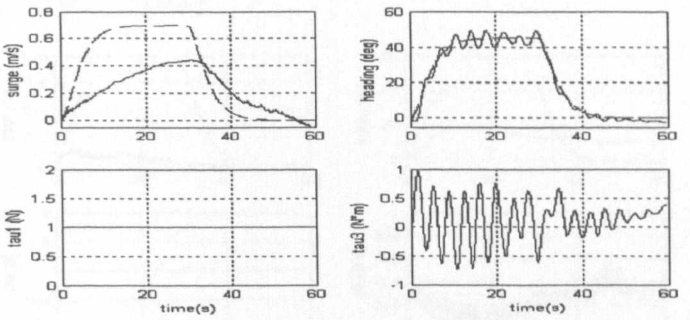


Figure D.66: Simulated Results from GP+GA Run No. 10 (waves)

Run No. 11

$\tau_{1com} = 1$

$\tau_{3com} = \varepsilon_h - \sin(\psi) - (\varepsilon_h - \sin(24.9)) \cdot sign\left(\mathbf{h}_h^{'} \cdot (\mathbf{x}_h - \mathbf{x}_{hd})\right)$

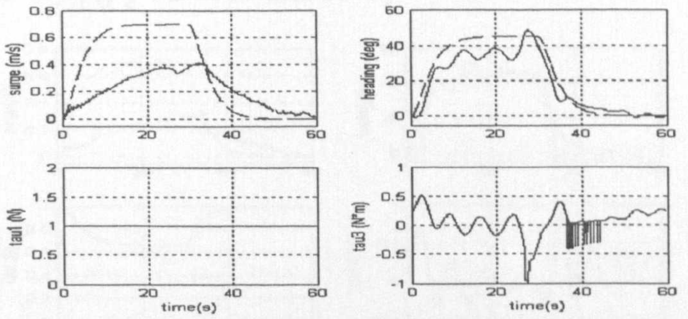


Figure D.67: Simulated Results from GP+GA Run No. 11 (waves)

Run No. 12

$\tau_{1com} = 1$

$\tau_{3com} = \sin(\sin(\sin(\sin(\sin(\sin(\sin(\sin(-\mathbf{k} \cdot \mathbf{x}_h))))))))$

$\mathbf{k} \equiv place(0, \quad -\psi, \quad -20.4)$

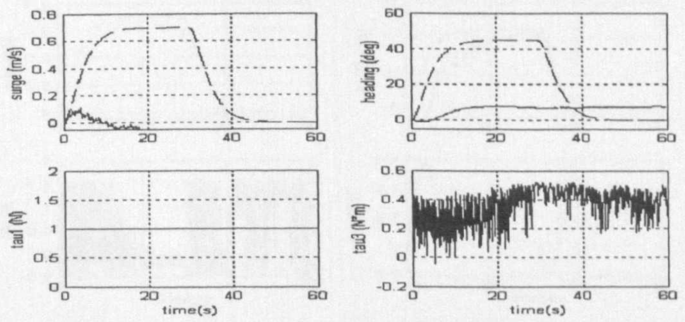


Figure D.68: Simulated Results from GP+GA Run No. 12 (waves)

Run No. 13

$$\tau_{1com} = 1$$

$$\tau_{3com} = \exp(-\mathbf{k} \cdot \mathbf{x}_h + \psi_d)$$

$$\mathbf{k} \equiv place(0, \quad -\psi, \quad -16.2)$$

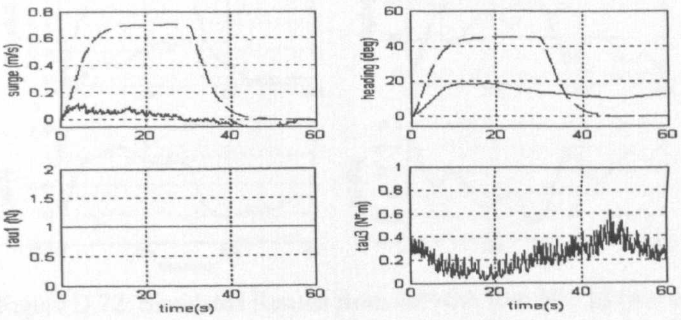


Figure D.69: Simulated Results from GP+GA Run No. 13 (waves)

Run No. 14

$$\tau_{1com} = \sin(7.7 - u_d)$$

$$\tau_{3com} = \varepsilon_h \cdot (67.2 - \varepsilon_h)/(\psi_d + 29.3) \approx 2.2935 \cdot \varepsilon_h$$

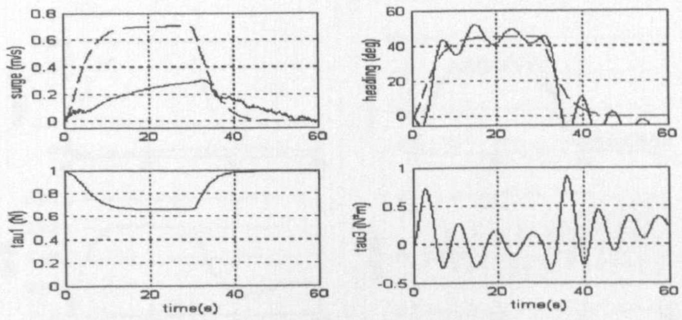


Figure D.70: Simulated Results from GP+GA Run No. 14 (waves)

Run No. 15

$$\tau_{1com} = \sin(1.8 + u)$$

$$\tau_{3com} = \sin(\varepsilon_h)$$

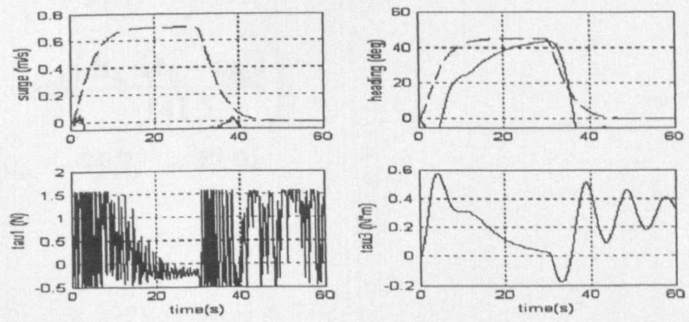


Figure D.71: Simulated Results from GP+GA Run No. 15 (waves)

Run No. 16

$$\tau_{1com} = 2 \cdot \varepsilon_p$$

$$\tau_{3com} = \sin(\sin(\psi + 59.2))$$

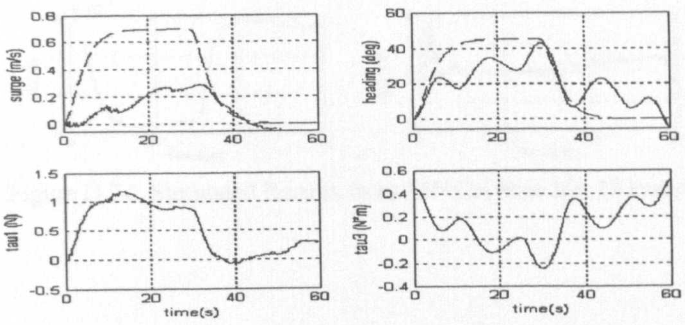


Figure D.72: Simulated Results from GP+GA Run No. 16 (waves)

Run No. 17

$$\tau_{1com} = 0.0431 \cdot \varepsilon_p \cdot sign\left(h_p' \cdot (u - u_d)\right) \cdot \tanh\left(\frac{h_p' \cdot (u - u_d)}{\varepsilon_p / 40.8}\right)$$

$$\tau_{3com} = 9.9 \cdot \varepsilon_h$$

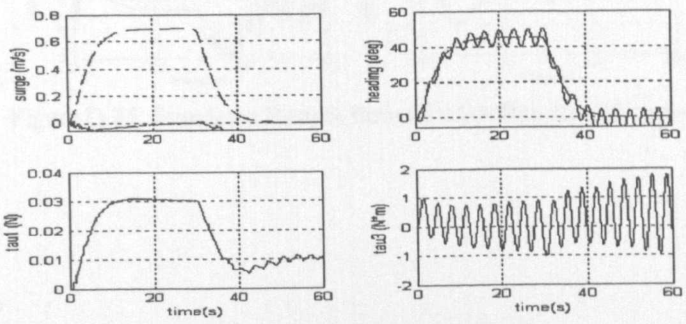


Figure D.73: Simulated Results from GP+GA Run No. 17 (waves)

Run No. 18

$$\tau_{1com} = \varepsilon_p \cdot \tanh\left(\frac{h_p' \cdot (u - u_d)}{97.1}\right)$$

$$\tau_{3com} = -\mathbf{k} \cdot \mathbf{x}_h \cdot \tanh\left(\frac{\mathbf{h}_h' \cdot (\mathbf{x}_h - \mathbf{x}_{hd})}{141.5}\right)$$

$$\mathbf{k} \equiv place(0, \quad -71.7, \quad -72.9)$$

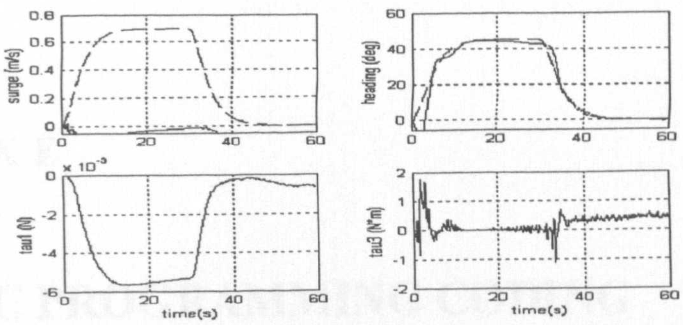


Figure D.74: Simulated Results from GP+GA Run No. 18 (waves)

Run No. 19

$\tau_{1com} = 0$

$\tau_{3com} = 0$

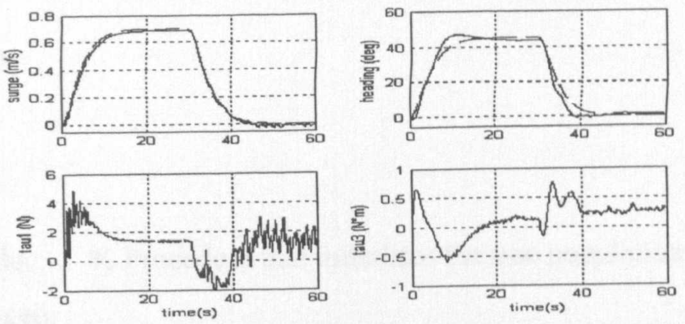


Figure D.75: Simulated Results from GP+GA Run No. 19 (waves)

Run No. 20

$\tau_{1com} = \exp(1)$

$\tau_{3com} = -\mathbf{k} \cdot \mathbf{x}_h$

$\mathbf{k} \equiv place(0, \quad -4.3, \quad -\psi)$

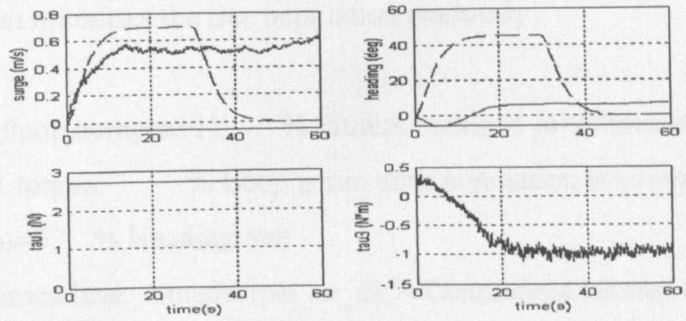


Figure D.76: Simulated Results from GP+GA Run No. 20 (waves)

APPENDIX E

GENETIC PROGRAMMING CODING

```
% gpmain.m
% Main GP code

clear
cont=0;
inicializar_popdec;    % Procedure that initialises the tree population randomly
for cont=1:numgen,
        select_popdec;          % Procedure that creates the new population
end
```

```
% inicializar_popdec.m
% Procedure that initialises the tree population randomly


fitness=zeros(totpop,numgen+1);      % Fitness matrix is initialised to 0
for numarbol=1:totpop,          % Loop is run until population is complete
        psi_or_u=0;   % Heading tree
        rama1=inicializar_ramadec(psi_or_u);% Generates a subtree with 3-7 nodes
        if rand(1)<5/11,          % Root is a function with 1 argument
                arbol=[rama1];
                nodopral=createmainroot1;   % Creates the main root
        else      % Root is a function with 2 arguments
```

```
                rama2=inicializar_ramadec(psi_or_u); % Generates a 2nd subtree
                arbol=[rama1;rama2];
                arbol=adjustindex(arbol);      % Adjusts the indexes in the tree matrix
                nodopral=createmainroot2;   % Creates the main root
        end
        arbol=[arbol;nodopral];          % Adds the main root to the tree matrix
        poppsi(numarbol)={arbol};   % Stores new heading tree matrix
        psi_or_u=1;   % Propulsion tree
        rama1=inicializar_ramadec(psi_or_u);% Generates a subtree with 3-7 nodes
        if rand(1)<6/11,          % Root is a function with 1 argument
                arbol=[rama1];
                nodopral=createmainroot1;    % Creates the main root
        else      % Root is a function with 2 arguments
                rama2=inicializar_ramadec(psi_or_u); % Generates a 2nd subtree
                arbol=[rama1;rama2];
                arbol=adjustindex(arbol);      % Adjusts the indexes in the tree matrix
                nodopral=createmainroot2;   % Creates the main root
        end
        arbol=[arbol;nodopral];          % Adds the main root to the tree matrix
        popu(numarbol)={arbol};        % Stores new propulsion tree matrix
        fitness(numarbol,1)=calcular_fitness(poppsi{numarbol},popu{numarbol});
        % Evaluates the performance of both trees through a zig-zag manoeuvre
end
```

```
% select_popdec.m
% Procedure that creates the new population

for i=1:totpop/2,
        % Two trees are chosen through tournament selection
```

```
pos1=select_arbol(totpop,fitness(:,cont));
pos2=select_arbol(totpop,fitness(:,cont));
if rand(1)<probcross,
        % Trees are crossed with a certain probability
        [newpsi1,cambiopsi1,newpsi2,cambiopsi2]=cross_arbol(poppsi{pos1
        }, poppsi{pos2});
        [newu1,cambiou1,newu2,cambiou2]=cross_arbol(popu{pos1},
        popu{pos2});
        cambio1=cambiopsi1 | cambiou1;
        % Determines if the heading tree has been altered
        cambio2=cambiopsi2 | cambiou2;
        % Determines if the heading tree has been altered
        if rand(1)<probmut,   % 1st tree is mutated with a certain probability
                newpsi1=mut_arbolpsi(newpsi1);
                newu1=mut_arbolu(newu1);
                cambio1=1;
        end
        if rand(1)<probmut,   % 2nd tree is mutated with certain probability
                newpsi2=mut_arbolpsi(newpsi1);
                newu2=mut_arbolu(newu1);
                cambio2=1;
        end
        % If changes have taken place trees are revaluated
        if cambio1==1,          newfit1=calcular_fitness(newpsi1,newu1);
        else    newfit1=fitness(selarbolpos1,cont);  end
        if cambio2==1,          newfit2=calcular_fitness (newpsi2,newu2);
        else    newfit2=fitness(selarbolpos2,cont);  end
 else   % If crossover does not take place
        if rand(1)<probmut    % 1st trees are mutated with certain probability
                newpsi1=mut_arbolpsi(poppsi{pos1},);
                newu1=mut_arbolu(popu{pos1});
                newfit1=calcular_fitness(newpsi1,newu1);   % Reevaluation
            else
                newpsi1=poppsi{pos1};newu1=popu{pos1};
                newfit1=fitness(pos1,cont);
            end
            if rand(1)<probmut    % 2nd trees are mutated with certain probability
                newpsi2=mut_arbolpsi(poppsi{pos2},);
                newu2=mut_arbolu(popu{pos2});
                newfit2=calcular_fitness(newpsi2,newu2);   % Reevaluation
            else
                newpsi2=poppsi{pos2}; newu2=popu{pos2};
                newfit2=fitness(pos2,cont);
            end
        end
        % New trees are stored in new population cell array
        newpoppsi{2*i-1}=newarbolpsi1;    newpopu{2*i-1}=newarbolu1;
        newpoppsi{2*i}=newarbolpsi2;      newpopu{2*i}=newarbolu2;
        newfitness(2*i-1)=newfit1;   newfitness(2*i)=newfit2;
end
% Updates storage matrices
poppsi=newpoppsi;   popu=newpopu;       fitness(:,cont+1)=newfitness';
```

```
% select_arbol.m
% Function that selects a tree through tournament selection

function arbolchosen=select_arbol(totpop,fitness)
        tourgroupsize=8;
        % Eight trees are chosen at random and their positions are stored in the
        % vector choice
        choice=randperm(totpop);    choice=choice(1:tourgroupsize);
        % The cost values of the chosen trees are stored in fitnesschoice
        for k = 1:tourgroupsize,
                fitnesschoice(k)=fitness(choice(k));
        end
        [menor, posmenor]=min(fitnesschoice);        % Selects the minimum cost
        arbolchosen=choice(posmenor);         % Returns the position of the best tree
```

```
% cross_arbol.m
% Function that crosses two trees

function [children1,cruce1,children2,cruce2]=cross_arbol(parent1,parent2)
        maxnodo1=size(parent1,1);
        maxnodo2=size(parent2,1);
        % Random choice of two crossover nodes
        crossnodo1=randperm(maxnodo1);  crossnodo1=crossnodo1(1);
        crossnodo2=randperm(maxnodo2);  crossnodo2=crossnodo2(1);
        nodo1=find_terminal(parent1,crossnodo1);  % Returns the node number of
        nodo2=find_terminal(parent2,crossnodo2);  % the subtree last node
        ramacruce1=parent1(nodo1:crossnodo1,:);  % Subtrees to cross
        ramacruce2=parent2(nodo2:crossnodo2,:);
        nodosrama1=crossnodo1-nodo1;        nodosrama2=crossnodo2-nodo2;
        fincross1=nodo1+nodosrama2;         fincross2=nodo2+nodosrama1;
        % If the resulting tree violates the size limit is replaced by its parent,
        % otherwise crossover takes place
        if (maxnodo2-nodosrama2+nodosrama1>30),
                if (maxnodo1-nodosrama1+nodosrama2>30)
                        children1=parent1;    children2=parent2;
                        cruce1=0;        cruce2=0;
                else
                children1(1:nodo1-1,:)=parent1(1:nodo1-1,:);
                children1(nodo1:fincross1,:)=ramacruce2;
                children1(fincross1+1:maxnodo1-nodosrama1+nodosrama2,:)
                =parent1(crossnodo1+1:end,:);
                % Updates the tree matrix indexes
                children1=actualizar_arbol (children1,nodo1,fincross1);
                children2=parent2;    cruce1=1;      cruce2=0;
        end
else
if (maxnodo1-nodosrama1+nodosrama2>30)
        children1=parent1;
        children2(1:nodo2-1,:)=parent2(1:nodo2-1,:);
        children2(nodo2:fincross2,:)=ramacruce1;
        children2(fincross2+1:maxnodo2-nodosrama2+nodosrama1,:)
        =parent2(crossnodo2+1:end,:);
        % Updates the tree matrix indexes
        children2=actualizar_arbol(children2,nodo2,fincross2);
        cruce1=0;      cruce2=1;
else
        children1(1:nodo1-1,:)=parent1(1:nodo1-1,:);
        children1(nodo1:fincross1,:)=ramacruce2;
        children1(fincross1+1:maxnodo1-nodosrama1+nodosrama2,:)
        =parent1(crossnodo1+1:end,:);
        children2(1:nodo2-1,:)=parent2(1:nodo2-1,:);
        children2(nodo2:fincross2,:)=ramacruce1;
        children2(fincross2+1:maxnodo2-nodosrama2+nodosrama1,:)
        =parent2(crossnodo2+1:end,:);
        % Updates the tree matrix indexes
        children1=actualizar_arbol(children1,nodo1,fincross1);
        children2=actualizar_arbol(children2,nodo2,fincross2);
        cruce1=1;
```

```
                cruce2=1;
        end
end
```

```
% mut_arbolpsi.m
% Function that mutates the heading and propulsion tree
% The function mut_arbolu.m is equivalent. The only difference lies in the function
% sets

function [newarbol]=mut_arbolpsi(arbol)
        maxnodo=size(arbolpsi,1);
        if rand(1)<0.5,            % Point mutation
                posibles_mut=0;
                for nodo=1:maxnodo,            % Counts number of nodes in tree
                        switch arbol (nodo,2)
                                case 0          posibles_mut=posibles_mut+3;
                                case 1          posibles_mut=posibles_mut+2;
                                case 2          posibles_mut=posibles_mut+2;
                                case 3          posibles_mut=posibles_mut+1;
                                case 4          posibles_mut=posibles_mut+2;
                                case 5          posibles_mut=posibles_mut+1;
                        end
                end
                % Randomly selects a mutation point
                mut_pos=randperm(posibles_mut);   mut_pos=mut_pos(1);
                % Finds that position in the tree
                nodo=0;         pos=0;
                while pos<mut_pos,
                        nodo=nodo+1;
        switch arbol(nodo,2)
                case 0          pos=pos+3;
                case 1          pos=pos+2;
                case 2          pos=pos+2;
                case 3          pos=pos+1;
                case 4          pos=pos+2;
                case 5          pos=pos+1;
        end
end
dif=pos-mut_pos;

% If dif=2 the chosen node is a terminal and it is substituted by a
% randomly chosen terminal from the terminal set
if dif==2,      arbol(nodo,3)= terminalset;    end
if dif==1,
        if (arbol(nodo,2)==0 | arbol(nodo,2)==1)
                % Chosen node is a 2-argument function and it is
                % substituted by a 2-argument function from the
                % function set
                arbol(nodo,4)= funset_2argpsi;
        elseif arbol(nodo,2)==2,
                % Chosen node is a terminal and it is substituted by a
                % terminal from the terminal set
                arbol(nodo,3)= terminalset;
                elseif arbol(nodo,2)==4
                % Chosen node is a 1-argument function and it is
                % substituted by a 1-argument function from the
                % function set
                arbol(nodo,4)= funset_1argpsi
        end
end
```

```
        if dif==0,
                if (arbol(nodo,2)==2 | arbol(nodo,2)==3)
                        % Chosen node is a 2-argument function and it is
                        % substituted by a 2-argument function from the
                        % function set
                        arbol(nodo,4)= funset_2argpsi;
                elseif (arbol(nodo,2)==1|arbol(nodo,2)==0|arbol(nodo,2)==4)
                        % Chosen node is a terminal and it is substituted by a
                        % terminal from the terminal set
                        arbol(nodo,5)= terminalset;
                elseif arbol(nodo,2)==5
                        % Chosen node is a 1-argument function and it is
                        % substituted by a 1-argument function from the
                        % function set
                        arbol(nodo,4)= funset_1argpsi;
                end
        end
        newarbol=arbol;
else    % Subtree mutation
        % Randomly selects a mutation point
        mut_pos=randperm(posibles_mut);   mut_pos=mut_pos(1);
        % Returns the node number of the subtree last node
        nodo_term=find_terminal(arbol,mut_pos);
        size_rama=mut_pos-nodo_term;
        psi_or_u=0;    % Heading tree
        % Generates a heading subtree with 3-7 nodes
        rama_mut=inicializar_ramadec(psi_or_u)
        size_rama_mut=size(rama_mut,1)-1;
        finmut=nodo_term+size_rama_mut;
        % If the mutated tree violates the size limit mutation does not take
        % place
```

```
                if maxnodo-size_rama+size_rama_mut>30,
                        newarbol=arbol;
                else
                        % The randomly generated subtree substitutes the old one
                        newarbol(1:nodo_term-1,:)=arbol(1:nodo_term-1,:);
                        newarbol(nodo_term:finmut,:)=rama_mut;
                        newarbol(finmut+1:maxnodo-size_rama+size_rama_mut,:)
                        =arbol(mut_pos+1:end,:);
                        % Updates matrix indexes
                        newarbol=actualizar_arbol(newarbol,nodo_term,finmut);
                end
        end
end
```

APPENDIX F

COMPLEMENTARY SIMULATION RESULTS

F.1. SIMULATED RESULTS OF THE HAND TUNED CONTROLLERS

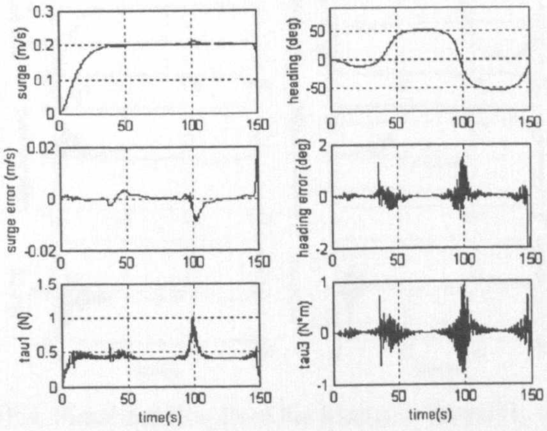


Figure F.1: Simulated Results of the Manually Tuned PID Controller When Tracking the Zig-Zag Manoeuvre Used in the Real Tests

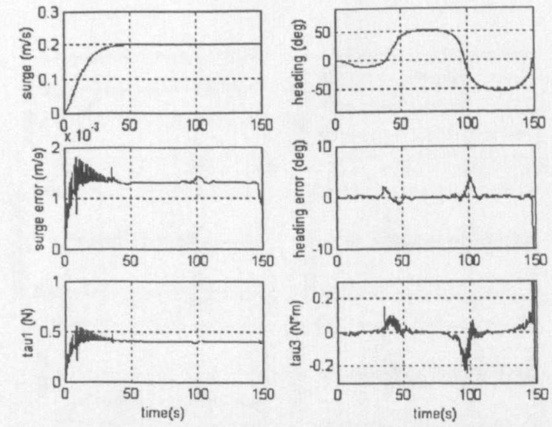


Figure F.2: Simulated Results of the Manually Tuned Pole Placement Controller When Tracking the Zig-Zag Manoeuvre Used in the Real Tests

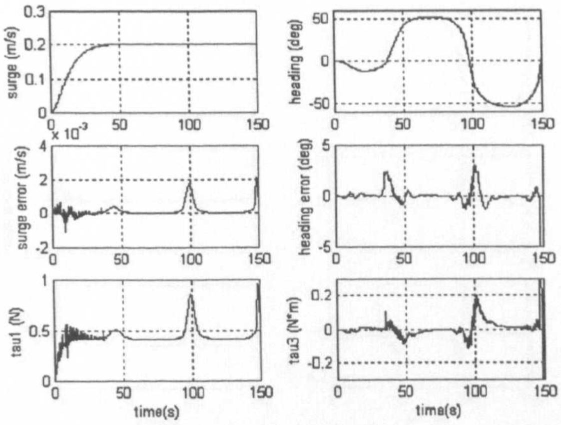


Figure F.3: Simulated Results of the Manually Tuned Sliding Mode Controller When Tracking the Zig-Zag Manoeuvre Used in the Real Tests

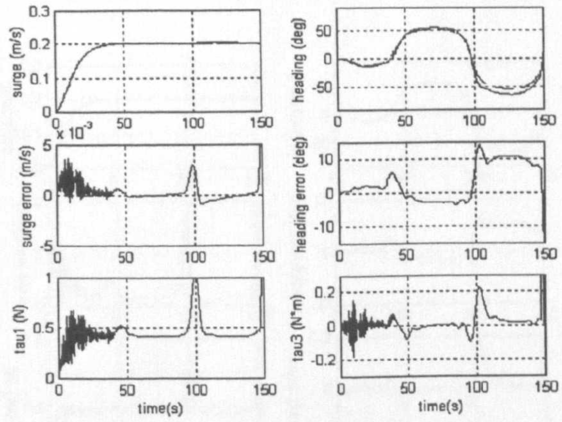


Figure F.4: Simulated Results of the Manually Tuned H_∞ Controller When Tracking the Zig-Zag Manoeuvre Used in the Real Tests

F.2. SIMULATED RESULTS OF THE OPTIMISED CONTROLLERS

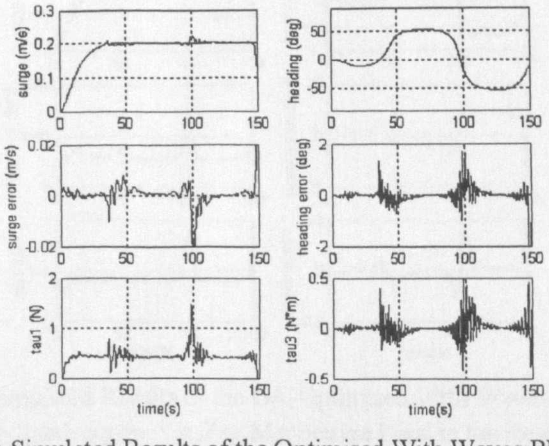


Figure F.5: Simulated Results of the Optimised With Waves PID Controller When Tracking the Zig-Zag Manoeuvre Used in the Real Tests

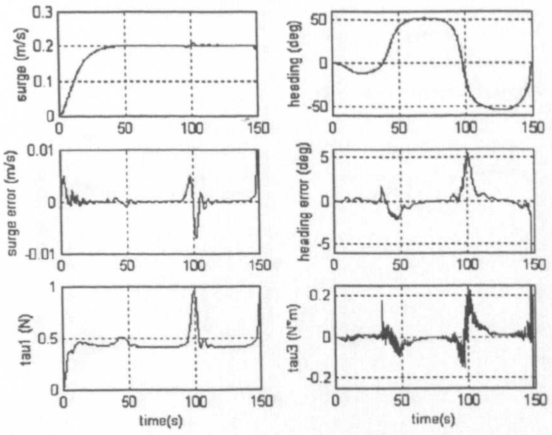


Figure F.6: Simulated Results of the Optimised With Waves SM+PI Controller When Tracking the Zig-Zag Manoeuvre Used in the Real Tests

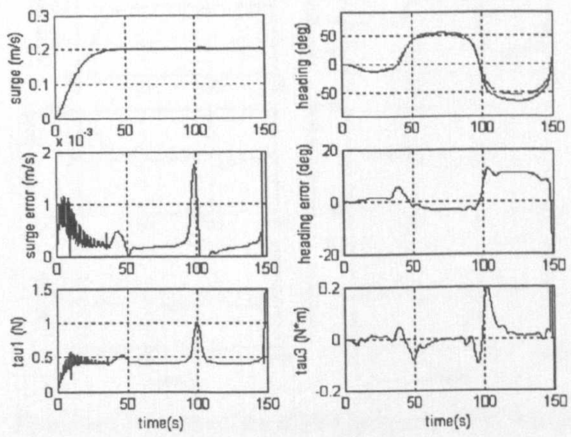


Figure F.7: Simulated Results of the GA-Optimised Without Waves H_∞ Controller When Tracking the Zig-Zag Manoeuvre Used in the Real Tests

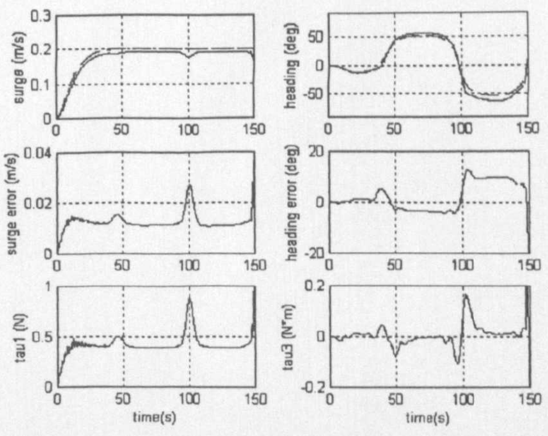


Figure F.8: Simulated Results of the GA-Optimised With Waves H_∞ Controller When Tracking the Zig-Zag Manoeuvre Used in the Real Tests

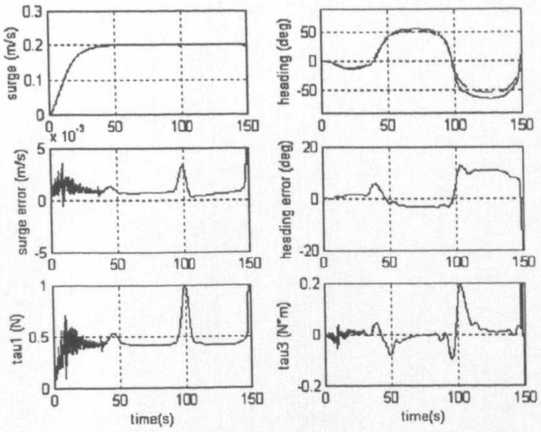


Figure F.9: Simulated Results of the sGA-Optimised Without Waves H_∞ Controller When Tracking the Zig-Zag Manoeuvre Used in the Real Tests

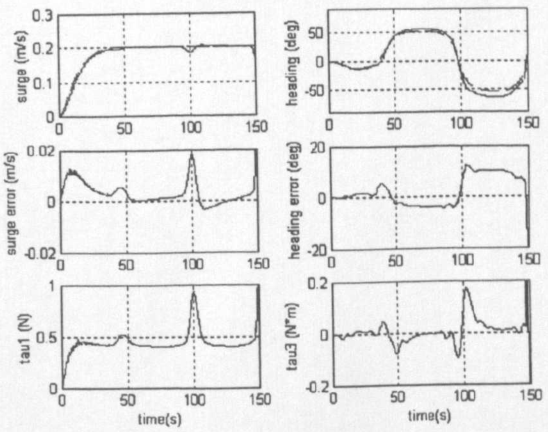


Figure F.10: Simulated Results of the sGA-Optimised With Waves H_∞ Controller When Tracking the Zig-Zag Manoeuvre Used in the Real Tests