2000

A Predictive Fuzzy-Neural Autopilot for the Guidance of Small Motorised Marine Craft

Richter, Ralph

A Predictive Fuzzy-Neural Autopilot for the Guidance of Small Motorised Marine Craft

A Fuzzy-Neural Approach

---

by

Ralph Richter

A thesis submitted to the University of Plymouth

in partial fulfilment for the degree of

Doctor of Philosophy

November 2000

University of Plymouth
Department of Mechanical and Marine Engineering
Drake Circus
PLYMOUTH
Devon PL4 8AA

United Kingdom

— to my wife Jana and my sons Neil & Lewis. —

Ralph Richter

A Predictive Fuzzy-Neural Autopilot for the Guidance of Small Motorised Marine Craft

Abstract

This thesis investigates the design and evaluation of a control system, that is able to adapt quickly to changes in environment and steering characteristics. This type of controller is particularly suited for applications with wide-ranging working conditions such as those experienced by small motorised craft.

A small motorised craft is assumed to be highly agile and prone to disturbances, being thrown off-course very easily when travelling at high speed but rather heavy and sluggish at low speeds. Unlike large vessels, the steering characteristics of the craft will change tremendously with a change in forward speed. Any new design of autopilot needs to be to compensate for these changes in dynamic characteristics to maintain near optimal levels of performance.

This study identifies the problems that need to be overcome and the variables involved. A self-organising fuzzy logic controller is developed and tested in simulation. This type of controller learns on-line but has certain performance limitations.

The major original contribution of this research investigation is the development of an improved self-adaptive and predictive control concept, the *Predictive Self-organising Fuzzy Logic Controller* (PSoFLC). The novel feature of the control algorithm is that is uses a neural network as a predictive simulator of the boat's future response and this network is then incorporated into the control loop to improve the course changing, as well as course keeping capabilities of the autopilot investigated.

The autopilot is tested in simulation to validate the working principle of the concept and to demonstrate the self-tuning of the control parameters. Further work is required to establish the suitability of the proposed novel concept to other control.

Contents

List of Tables

List of Figures

Nomenclature

Abbreviations/ Acronyms

ANN ...	Artificial Neural Network
BP ...	Back-propagation
DEL ...	Delay in Reward
DOF ...	degree of Freedom
FLC ...	Fuzzy Logic Controller
MIMO ...	Multi-Input Multi-Output
NN ...	Neural Network
PD ...	Proportional plus Derivative
PI ...	Performance Index
PID ...	Proportional plus Integral plus Derivative
PSoFLC ...	Predictive Self–organising Fuzzy Logic Controller
SISO ...	Single-Input Single-Output
SOC ...	Self–organising Controller
SoFLC ...	Self–organising Fuzzy Logic Controller
TIA ...	Time in Advance

Fuzzy Logic Symbols

NB, NM, NS ...	Negative Big, Medium, Small
PB, PM, PS ...	Positive Big, Medium, Small
Z ...	Zero
μ ...	Membership Function

Variables

ψ ... heading	$\dot{\psi}$ ... yaw rate	$\ddot{\psi}$ ... yaw acceleration
δ ... rudder angle	$\dot{\delta}$ ... rudder rate	ε ... error
τ ... time constant	K ... gain	C ... criterion

Subscripts

$_a$ ... actual	$_d$ ... desired	$_m$ ... measured
$_c$ ... current		

Acknowledgements

I would like to thank Dr. L. Bradbury who greatly supported me during the last phase of writing this thesis when it came to proofreading and tieing up all the lose ends. I am very grateful to Mr. D. Winwood from Cetrek Ltd. who made his private boat available for some testing and helped me by interfacing the instruments and steering device on his boat.

I would like to thank my supervisors Dr. M. N. Polkinghorne and Mr. P. Nurse and my director of studies Prof. R. S. Burns for the time and the always friendly and helpful assistance during my stay at the University of Plymouth.

Special thanks to my wife for keeping up with me, catering for me and so allowing me to concentrate fully on my research. I also would like to thank Mr. Brian Lakey and Mr. Steve Roberts for all their time and support when it came to proofreading and all the tricky questions of the English language.

Finally, thanks to all the occupants of the research office SM105 for their patience with me and my noisy PCs. In particular, to Cathy-Ann Radix who always had an open ear for listening to the problems associated with the typesetting of this document.

Thank you.

Ralph Richter

Declaration

At no time during the registration for the degree Doctor of Philosophy has the author been registered for any other University award.

This study was financed with the aid of a studentship from the University of Plymouth and carried out in collaboration with Cetrek Ltd.

Relevant scientific seminars and conferences were regularly attended at which work was often presented; external institutions were visited for consultation purposes and several papers prepared for publication in journals and conference proceedings.

Signed: Ralph Richter

Date: November 2000

Chapter 1

Introduction and Structure of the Thesis

1.1 Aim and Main Objectives of the Research

Over centuries helmsmen have been steering ships. The task of manoeuvring the vessel safely through both rough and calm sea lies with the helmsman. Depending of the environment, this can sometimes demand a high level of skill, expertise and decision making. At other times it can be boring and tedious to concentrate for long periods of time. Sailors are still looking for suitable devices to assist them in their navigational task and vessel operation. There has been an increased use of electro and mechanical devices to automate this process with certain degree of success. Recently there have been large advances in technology both in this and other applications.

Of particular interest are the developments in the field of neural networks and fuzzy logic. This research investigates the possibilities of employing up-to-date techniques such as fuzzy logic and neural networks to perform course-keeping/ course-changing control aimed at a specifically small, highly responsive maritime craft. The novel combination of both fuzzy logic and neural networks in the fashion described in this thesis is unique.

The aim of this research is to develop an autopilot which is able to quickly and reliably adapt its control parameters to changes in environment and steering characteristics. The aim is to give the autopilot self-tuning capabilities, so that it can modify its control parameters automatically without the need of input from the human helmsman. This will have the effect of improving vessel control which could result in reduced fuel costs and travel time. Importantly, it will also improve passenger comfort/ cargo safety, and reduce the risk of human

error.

In order to achieve the aim of this research, the following objectives were identified. It is necessary to undertake a study of historic autopilot development, and to identify the limitations of the current technologies being used. For benchmarking purposes a conventional PD autopilot was therefore developed and tested in course keeping and course changing modes of operation. An alternative design using SoFLC was also developed to demonstrate its performance abilities. This controller forms the basis of the research undertaken and has been expanded to create a novel predictive form of self-organising fuzzy logic controller. All these types of controllers have been tested in simulated conditions. The results being analysed and relevant comparison regarding made. Conclusions have been drawn regarding the performance advantages obtained and recommendations are provided for further research.

1.2 Issues Related to Ship Control Systems

Considerable information about the transient behaviour of the craft is needed for the successful control of such a small and responsive vessel. Owing to the size and possible high forward speeds, such a craft is a highly responsive plant and very sensitive to all types of disturbances, such as change in speed, loading and weather, *etc*. Even fundamental characteristics can change. Consider the behaviour at different forward speeds; at the low speed end, the vessel can be seen as a displacement boat, whereas at higher speeds the vessel can go into planing mode and thus change the steering characteristics entirely. Different loading conditions will change the mass and therefore the inertia, draft, added mass *etc* and therefore the time transient behaviour of the vessel responding to a rudder change will alter too. It now depends upon the 'intelligence' of the controller to cope with such a wide range of working conditions [45]. Sophisticated hardware and software is needed to identify the working environment and to activate the correct control procedure in order to produce an optimal control performance [6, 31]. Optimal control performance is desirable because it means an improved course accuracy which results in savings in fuel and travelling time [28].

It is imperative to emphasise the need for, and the advantages and disadvantages of using autopilots for course keeping and course changing control. Under various sea conditions

control and steering of any size vessel can become both boring and tedious. This can lead to a decrease in safety due to lack of concentration. In these circumstances the helmsman's task is to maintain the vessel on a desired course to achieve some preset destination. However, the helmsman will also attempt to continually optimise the vessel's performance by minimising heading error and rudder usage. It has proven [45, 14, 124] to be very helpful to employ a device (electrical or mechanical) to do the course-keeping, thereby allowing the helmsman to concentrate on other crucial activities, *eg* navigation, route planning, *etc.* Furthermore, a human being needs to utilise helpful aids and devices to detect very slow translation and rotation of the vessel. To this aim, the human's main input is visual data from the compass. If the data from the compass is used as an input to an automated control device, it is possible for the resulting autopilot to achieve acceptable levels of performance when compared to that of the original helmsman. The three main components of a control loop can be seen as:

- the controller,
- the plant and
- the feedback device.

The signal flow between the components can be summarised as:

- error detection,
- decision making (controlling),
- application of the control action.

The error detector is a device which subtracts the actual heading from the desired heading. This error is then fed into the control algorithm, which outputs a signal to the control actuator, the rudder on the ship. So, the signal is a desired rudder angle which drives the vessel back on course.

The precise nature of the autopilot's performance is therefore highly dependent on the methodology used within the controller, but clearly there is significant potential for further improvement using modern techniques.

This research takes into account all factor related to ship control for small motorised vessels. It gives particular emphasis to course changing and course keeping. By introducing the use of artificial neural networks and fuzzy logic to perform the task of ship control, it is

possible to develop a new design of autopilot which should outperform the alternative approaches currently used. The novel combination of fuzzy logic control plus neural network system identification allows for enhanced control performance, and is an original contribution to knowledge. The system operation is validated using simulated testing. Performance is then compared to other systems.

The emphasis of this research is based upon the demonstration of the capability of the new system to adapt its control parameters rapidly in response to changes in the operating environment.

1.3 Layout of the Thesis

Chapter 1 (Introduction) provided an overview over the research aim and objectives. It also considers some aspects related to ship control systems to outline the background of this research.

Chapter 2 (Survey – The History of Piloting) provides a brief analysis to justify the need for this research, and the use of automated steering devices. It also provides an introduction to the field of marine autopilots, the history and development of autopilot design from the turn of the century via the P, PD, PID control law to the advanced and adaptive concepts of the present day. Modern control algorithms are introduced and briefly explained in order to put this research into the right context. This thesis concentrates only on specific techniques currently employed which are applicable to the research being undertaken. An overview of the historical development of autopilots is included, and an understanding of the limitations of modern techniques is provided.

Chapter 3 (Heading Control using PD, Fuzzy Logic and Self-organising Fuzzy Logic) provides an overview of PD control, fixed rulebase *fuzzy logic control* (FLC), and also *self-organising fuzzy logic control* (SoFLC). Particular emphasis is given to the basic building blocks of the SoFLC. These are rulebase design, performance index structure, and rulebase update algorithm. The SoFLC forms an important part within the novel control design proposed by this study.

Chapter 4 (The PSoFLC) describes the development of the novel *Predictive Self-organising Fuzzy Logic Controller* (PSoFLC). A method is demonstrated which reduces the adaptation time of a self-organising fuzzy logic controller (SoFLC) acting in a new, unknown environment. This chapter also contains an extension to 'classic' fuzzy logic as proposed by Zadeh [127, 128]. A novel defuzzification method is introduced (*section 4.4, A Novel Defuzzification Method Using a Normalisation Technique*) which provides a much smoother control surface when irregularly placed, and asymmetrically shaped, fuzzy sets are used in the output window. Applied to ship control, this can prevent erratic rudder movements which could cause extensive wear and tear on the rudder mechanism as well as waste fuel and reduce forward speed.

The functionality of the controller and the predictor modules are separately explained, and individual test results are shown. This method of using a neural network to determine a mathematical model does not require specialist knowledge about the plant and environment. Furthermore, the design engineer can concentrate on the input and output data, and the neural network will find a relationship between the two. It also enables the identification of the influence of individual parameters to the overall response. The unique combination of both modules, the predictor and the SoFLC, forms this novel controller and is the basis for the originality of this study.

Chapter 5 (Simulation Test Results) contains the simulated tests results of the predictive controller (PSoFLC) and the results are analysed with detailed graphs showing the various responses. The tests include step response tests without and with disturbances as well as course following tests. The test results of the PSoFLC are compared and analysed with namely a PD controller tuned for high speed and also with the SoFLC. Each section of the chapter contains a discussion of the test results which highlights the similarities and differences identified.

Chapter 6 (Discussion) identifies the key points derived from the test results and identifies benefits and limitations of the novel predictive self-organising fuzzy logic controller.

The new fuzzy logic defuzzification technique is discussed in a variety of test environments. The development of a testbed to host the controllers is introduced, and the predictor

design is discussed in detail. A comparison of the controllers is undertaken and relevant conclusions made.

Chapter 7 (Conclusions and Suggestions for Future Work) contains the general conclusions of this research and lists some possible ways to implement the new control concept into other applications. It also reflects on the covered areas of this research and its objectives.

Appendix A (Neural Networks: Theory) Theory and applications of *artificial systems* such as *neural networks* are explained in appendix A. Here, examples are shown, of how neural networks can be used for control and system identification. This appendix provides examples of industrial applications utilising this technique. The neural network part is subdivided into two sections, *Neural Networks for Control* and *Neural Networks for System Identification*.

Appendix B (Fuzzy Logic: Theory) provides an overview of *fuzzy logic*, the theory and their application in industry. The principles of the fuzzy set theory are explained as well as operators used to formulate 'fuzzy' rules. Various defuzzification methods are explained.

It is necessary to introduce the two techniques uniquely combined in this research. The interested reader can find the main principles of neural networks and fuzzy logic in the following two appendices A and B respectively.

Appendix C (The Simulation Set-Up) contains explanation on the simulation method used, based upon a 52 ft (16m) life boat simulator by Browning [20]. The development of an *Integrated Autopilot Testbed* used for data logging and testing can be found in appendix C.2. The data-logging interface is explained together with the communication protocol (NMEA 0183 [72]) used between the controller and boat. Some operational guidance is provided on how to operate the software, *eg* selecting various control regimes, and the the gauges displayed on the screen are explained.

Appendix D (NMEA Messages considered) lists the NMEA messages [72] used for communication. These messages are exchanged between the simulation and the testbed software using the serial ports.

Appendix E (Rulebases) consists of a time series of rulebases developed during the training of both self-organising controllers. A graphical representation (control map) of the rulebases accompanies each set of numbers.

Appendix G (Papers, Publications, Presentations) contains a copy of all related research publications based upon this study.

Chapter 2

Survey – The History of Piloting

In 1922, Minorsky [66] emphasised in one very early paper the advantages of using an automated steering aid:

> *For merchant ships an accurate and reliable automatic steering device becomes a real money saving proposition, largely justifying its use.*
>
> *On battleships, by its use the absence or reduction of yawing in action means a better efficiency in gunfire, increased maneuvering speed and also a greater cruising radius.*
>
> *Quotation: Minorsky [66], p. 280*

The control task of ship navigation can be subdivided into two major divisions. The course related autopilot attempts to optimise ship orientation rather than the ship's position. The main control task is therefore to maintain or change, the heading of the ship to minimise the error from the desired course. The track related autopilot optimises the position of the vessel and not its orientation.

To clarify the above: this work only considers course and directional tasks of an autopilot and not position oriented strategies summarised as navigation. Of course, there are combinations possible and nowadays those are most commonly available on the market. Nevertheless, the control of the vessel is broken down as mentioned - the control of the vessel's orientation as the base level with the navigation unit sitting on the next hierarchical level above (figure 2.1). control is a complex task and for this purpose of design must be considered in isolation.

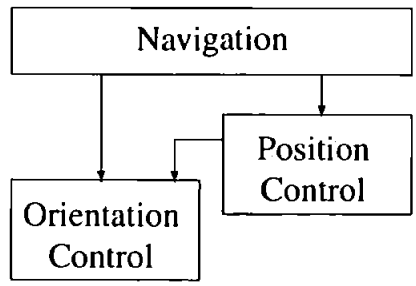


Figure 2.1. Navigation Layer

2.1 Early Developments until 1930

A very important and early paper was published by Minorsky [66] in 1922. This paper discusses the stability problems of automated steering and developed the basic theory of 'directional stability of automatically steered bodies'.

Furthermore, Minorsky subdivided the control problem into individual, smaller problems such as rudder position control, rudder angular velocity control and rudder angular acceleration control.

Similarly, Sperry [98] described the first installation of a gyrocompass aboard a ship in 1922. In this publication, he considered the problems that occur with automatic steering using a gyrocompass. In this very early work one can find all the elements that make up the control loop of an automated steering system for course keeping purposes. The steering device proposed by Minorsky was installed and tested on the battleship *New Mexico* [67]. By 1932, this application had been installed on more than 400 merchant ships all over the world [21]. Nevertheless, before it became such a vast success, some problems with the gyroscope principles needed to be solved.

In 1923, Schuler [95] described the behaviour of pendulums and gyroscopes when accelerated in a horizontal direction. The doubts raised by Martienssen [64] in 1906 based on calculating gyroscopic compasses errors under north-south acceleration were fundamental for further research in this field. He discovered very great errors of the gyroscopic compass device, and simply concluded that this device, or at least this design, is useless for accurate navigation. However, Schuler suggested a re-designed device which overcame the problems mentioned above.

Utilisation of the new design, and the subsequently derived equations finally led to the successful gyroscopic devices now commonly used. The difficulties of the early years have been overcome and gyroscopes can be found in most navigation devices which require a high degree of accuracy.

The autopilot used for the period 1930 to 1950 was a rather simple controller as proposed by Minorsky in 1922 ([66], p. 282). The heading error produces a signal which is then directly used to adjust the steering mechanism. The controller can be seen as a proportional controller. It is possible to adjust the control parameter (K_P... proportional gain) to suit different conditions *eg* ship loading. This simple device cannot cope with a wide range of conditions, *ie* in rough weather conditions when the proportional controller forces the steering mechanism to be heavily used and which therefore wears out very quickly. A weather adjustment is therefore necessary to prevent this excessive wear. In most cases a simple deadband is introduced to avoid high frequency and small magnitude movements. The rudder position is then only changed if the control output exceeds a small specified rudder angle. A different method to avoid rudder wear by including a delay feedback was proposed. The rudder cannot stop or change direction until this angle has been reached. Nomoto and Motoyama [74] described this method as 'negative backlash'.

2.2 Development of the PID Autopilot

During the period, overshadowed by two world wars, the autopilots used were mainly simple mechanical devices following a simple proportional rule.

$$\delta_d = K_P \cdot \psi_e \tag{2.1}$$

where: δ_d ... desired rudder angle,

K_P ... proportional gain constant,

ψ_e ... heading error.

These devices were not very satisfactory and could not prevent overshooting and therefore often caused transient oscillation.

Schiff and Gimprich [93] introduced the addition of a rate control term. In the 1950s, an improvement in stability could be achieved by the introduction and use of the mainly first derivative of the heading error ($\dot{\varepsilon}$) or the rate of turning (angular velocity $\dot{\psi}$). The first commercial autopilot utilising this technique was installed in 1951 on the S. S. United States [63].

The control rule of this autopilot may be defined as:

$$\delta_d = K_P \cdot \psi_e + K_D \cdot \dot{\psi}_e \tag{2.2}$$

where: K_D ... differential gain constant.

This was referred to as the PD (proportional plus derivative) controller. In 1953, Motora [71] suggested to applying a low-pass filter to the output signal to prevent rudder oscillation. According to Rydill [91], who analysed the effectiveness of the PD controller, this may generate a loss in stability and he therefore recommended the use of a quadratic delay technique (a second order filter) to overcome this problem. Applying this filter reduces the high frequency rudder movements with less damaging effect on stability.

Schiff and Gimprich [93] also proposed the addition of an integral term, but it was not considered further because it was thought to make the ship response sluggish. However, the integral term finally found consideration in the control equation; the resulting control law being summarised as follows (equation 2.3).

$$\delta_d = K_P \cdot \psi_e + K_D \cdot \dot{\psi}_e + K_I \cdot \int \psi_e \, dt \tag{2.3}$$

where: K_I ... integral gain constant.

The consideration of the integral term now allowed to maintain the ships course in the presence of steady state disturbances, such as tidal currents and cross winds. Bech in 1972 [13] emphasised on the needs to tune the autopilot with the demands of optimal propulsion economy in mind. The application of the PID control laws in ship autopilots when operating in rough seas was further analysed by Blanke [15].

The PID (proportional plus integral plus derivative) control rule was formulated. The addition of the integral term assisted in minimising the rudder movements as well as the

steering gear lags. Constant disturbances, causing an offset were now taken into account and the PID autopilot was fully capable of dealing with them.

However, the introduction of the integral term may slow down the rudder response and cause a sluggish ship response [69]. An acceleration term is therefore introduced to the PID control rule 2.3 to compensate the slowed down rudder response. The extended control equation can be written as:

$$\delta_d = K_P \cdot \psi_e + K_D \cdot \dot{\psi}_e + K_I \cdot \int \psi_e \, dt + K_A \cdot \ddot{\psi}_e \tag{2.4}$$

where: K_A ... acceleration gain constant.

Controllers based on the PID format could not prevent the generation of high frequency rudder movements [122] in certain operating conditions (*eg* periodic wave patterns). Those high frequency rudder movements can have a detrimental effect on the hull's yawing movement [6], and can cause extensive wear of the steering gear.

The introduction of a deadband in the rudder loop can lead to unstable behaviour (the wind-up of the integral causes this effect). The deadband is a threshold value which the demanded rudder change has to exceed in order to be executed. If the demanded rudder change is less then this deadband, then this control action is simply ignored.

2.3 Adaptive Autopilots

The PID controller can be tuned to work under certain specific conditions. If these conditions change – due to weather (*eg* waves, wind, tide or current) [30], speed or load, the controller will not operate near its setting point [6]. To maintain a high level of performance, a further tuning adjustment of the control parameters is then required to ensure satisfactory autopilot performance.

> *The dynamic behaviour of the ship and hence also the parameters of this model are dependent on the external circumstances and the applied thrust power. When the ship is steered with an autopilot it is necessary to adjust the parameters of the autopilot dependent on the change of the steering characteristics of the ship.*
>
> *Quotation: van Amerongen and Udink ten Cate [8]*

It was determined that the performance of even the most advanced PID controller could be improved by adjusting its parameters according to the operating environment of the control system (ship and autopilot). Van Amerongen [2] defined two disadvantages of the PID–type controllers:

- *It is difficult to adjust manually. Because the operator, the watch officer, has many other tasks and lacks the insight into control theory, his adjustment will seldom be optimal.*
- *The optimal adjustment varies and is not known by the user. changing circumstances require manual re-adjustment of a series of settings of the autopilot. This holds not only for variations in the parameters of the process but also when due to a varying traffic situation the required performance changes.*

Quotation: van Amerongen [2]

The PID parameter adjustment may be achieved either manually or automatically. The disturbances, and therefore the effects to the hull, may also be subdivided into two major categories:

1. disturbances that cause a 'small' deviation of the desired course and
2. disturbances which change the vessel's characteristics and consequently the steering characteristics.

Weather and tidal changes such as waves, wind and current are associated with category 1. Changing the mass of the vessel whilst loading/ unloading and the resulting draft, displacement and inertia, the quantity of water under the keel and alterations in the forward speed, all alter the handling characteristics of the vessel and are therefore associated with the second category. Small adjustments required to compensate for the disturbances defined by category 1 may be overcome by automatic adjustments. Changes to the autopilot parameter settings to counteract disturbances of category 2 are mainly undertaken by the operator [8]. These adjustments therefore demand a significant knowledge of both the handling characteristics of the ship and the environment/ disturbances. Research dating back until 1972 [44, 7, 83, 3, 62] and more recent work [31, 124, 96, 132], including the one described in this thesis, concentrates upon the possibility of automatically adjusting the control parameter for both types of disturbances. This will 'de–skill' the operating of the vessel and therefore achieve an improvement in safety and economics.

An heuristic approach to the adjustment of the PID gains was undertaken by Oldenburg [78]. Other researchers, such as Brink *et al.* [17] and Ohtsu *et al.* [77], have chosen an stochastic adaptation algorithm.

The human factor which is the major source of errors can be taken out of the loop. Employing an automatic device to do the steering can also improve the stability of the vessel in roll [4, 51, 5, 37, 16, 110]. Safety improvements are realised by allowing the operator to concentrate fully on navigation and collision avoidance.

2.4 Self-tuning Controller

The process of self-tuning is referred to as the on-line adjustment of controller parameters.

In the early 1970s, researchers concentrated on self–tuning or self–adjusting control to overcome the problems which occur when classical control algorithms are applied to areas with changing environment and/or uncertainties. Åström and Wittenmark [10] published a paper in 1972 which considered a SISO (single–input single–output) system with constant but unknown parameters. For this kind of system optimal control algorithms can be formulated and solved using non–linear stochastic control theory. However, obtaining the solution is very impractical because the computational demands needed in order to cover a wide range of working conditions. A different approach to solving this problem is by taking knowledge of the process into account and the fact the system has constant but unknown control parameters. One way of finding these parameters is by employing strategies which will converge to the optimal strategies. Those algorithms will be referred to as *self–tuning* or *self–adjusting* strategies [10].

Further research in this area was published by Clarke and Gawthrop [29] and Lim and Forsythe [59] who utilised a cost function which was minimised in order to change the controller's parameters. In 1990, Vahedipour *et al.* [114] developed a pseudo derivative feedback autopilot. Kallström and Åström [52], Mort and Linkens [70], Brink and Tiano [18] looked into self-tuning methods.

The H_∞ approach, a frequency based, robust control technique, was applied to marine autopilot design too but was found to be particulary appropriate for flight control systems

and gas-turbine designs [32] where the emphasis is on high performance, robust designs and reliable systems [36, 37].

It is clear, that the classical and tuned PID autopilot has limitations. It is always fascinating how human operators can cope with a very wide range of unknown and uncertain conditions. The latest research in this field attempts to adapt human abilities such as learning and experience to the design of a controller with an increased level of performance [54, 109, 113, 130, 121].

2.4.1 Model Reference Adaptive Control (MRAC)

The *Model Reference* approach is based upon the comparison of measured, actual data and data of an ideal mathematical model (*reference model*) which represents the desired response. An error function containing both information of the reference model and the vessel to control is derived. By adjusting the controller's parameters, this function (criterion) is then minimised in such a way, that the actual response follows closely the response of the model.

In 1973, van Amerongen and Udink ten Cate underlined the importance of adapting the parameters of the autopilot and compared two methods of model referencing. In the paper [8], both of the following approaches to tackle the 'fixed settings problem' are described.

Layne [57] takes the same principle in his Fuzzy-Model-Reference-Learning-Controller (FMRLC), but he does not adjust a conventional controller, but the "learning algorithm seeks to adjust the fuzzy controller so that the closed–loop system ... acts like a pre-specified reference model".

Lightbody [58] undertook further research on the idea of MRAC. Here, the controller is a back-propagation neural network and the error between the reference model and the plant is used to adapt (teach) the neural network controller.

The Mathematical Model A mathematical model of the ship is the counterpart of the actual ship. The control action is applied to both the model and the ship, then the control parameters are adapted following a criterion such as $J = \frac{1}{2}\varepsilon^2$ with the error defined as $\varepsilon = \psi_m - \psi_a$. The result of the sensitivity coefficient and error adjusts the feedback signal obtained from the rate gyro. The adaptation takes place exactly by adjusting this rate feedback

signal.

Nomoto's model [75] (equation 2.5) is often the basis for this technique in maritime control application.

$$\tau_1\tau_2\dddot{\psi}\cdot(\tau_1+\tau_2)\ddot{\psi}+\dot{\psi}=K\delta+K\tau_3\dot{\delta} \tag{2.5}$$

where: ψ ... heading δ ... rudder angle

τ_x ... time constants K ... gain

The constant K and time constants τ_1, τ_2 and τ_3 are related to the mass and speed as well as to the hydrodynamics of the vessel. The rudder variable is δ and the variable ψ belongs to the course. The transformation into Laplace domain assuming zero initial conditions gives:

$$\frac{\dot{\psi}(s)}{\delta(s)}=\frac{K(\tau_3 s+1)}{(\tau_1 s+1)(\tau_2 s+1)s} \tag{2.6}$$

It has been found by van Amerongen and Udink ten Cate [8] that this model is too simple to describe the complete ship's behaviour, so the rudder angle should not exceed 5°. However, this model is feasible since under normal steering, a ship often makes only small deviations from the straight line path [57]. For most applications however, a model suited for rudder bigger than 5° is needed. An extended transfer function (as proposed by Bech and Smitt [12]) can be used. If the thrust power remains constant ($\frac{K}{\tau_1\tau_2}$, $\frac{\tau_1+\tau_2}{\tau_1\tau_2}$, $\tau_3\approx$ constant), the transfer function can be re-written (substituting $(\frac{1}{K})\dot{\psi}=H(\dot{\psi})$):

$$\dddot{\psi}+\left(\frac{1}{\tau_1}+\frac{1}{\tau_2}\right)\ddot{\psi}+\frac{K}{\tau_1\tau_2}H(\dot{\psi})=\frac{K}{\tau_1\tau_2}(\tau_3\dot{\delta}+\delta) \tag{2.7}$$

If the rudder rate (rudder-angular velocity) is neglected and if $a_1=\frac{1}{\tau_1}+\frac{1}{\tau_2}$ and $K'=\frac{K}{\tau_1\tau_2}$ then equation 2.7 simplifies to:

$$\dddot{\psi}+a_1\ddot{\psi}+K'H(\dot{\psi})=K'\delta. \tag{2.8}$$

$H(\dot{\psi})$ represents a non-linear function of $\dot{\psi}$ and can be obtained from the relationship between δ and $\dot{\psi}$. When the the external conditions do not change ($\dddot{\psi}=\ddot{\psi}=\dot{\delta}=0$), then $H(\dot{\psi})$ can be found from the relationship between δ and $\dot{\psi}$. A spiral test gives an approximation of

$H(\psi)$:

$$H(\psi) = a\psi^3 + b\psi. \tag{2.9}$$

Figure 2.2 shows the block diagram of adaptive autopilot design using a reference model. The model represents the desired behaviour of the ship.

Sensitivity Models This *sensitivity model* technique is especially designed to prevent course instability of very large ships. The adaptation process with the sensitivity model is in fact based on a continuous hill climbing technique. The criterion used in that approach can be defined as:

$$C = \int_0^T \frac{1}{2}\varepsilon^2 dt \tag{2.10}$$

where: C ... criterion, ε ... error.

Using the steepest descent method, the gain K_D of the rate feedback signal is adjusted. This approach is not stable under all circumstances [38].

Liapunov Approach This approach follows the principle of direct adjustment of the controller's parameters. Assuming that the model's transfer function and that of the system are of the same order, a difference between the state variables of the system and the model is utilised to adjust the system's parameters in order to minimise this difference.

Existing differences between the state vectors of the the model and the system are minimised by altering the system parameter. The process is assumed to be linear and that non-stochastic disturbances occur. A low-pass filter also is required in rough seas.

The model represents a desired response and the system should follow this response as closely as possible. There are some difficulties when the Liapunov technique is applied to non-linear ships. A low–pass filter is required to filter out the measurement noise.

However, according to van Amerongen and Udink ten Cate [8], when the results of both techniques are compared, no significant difference can be found.

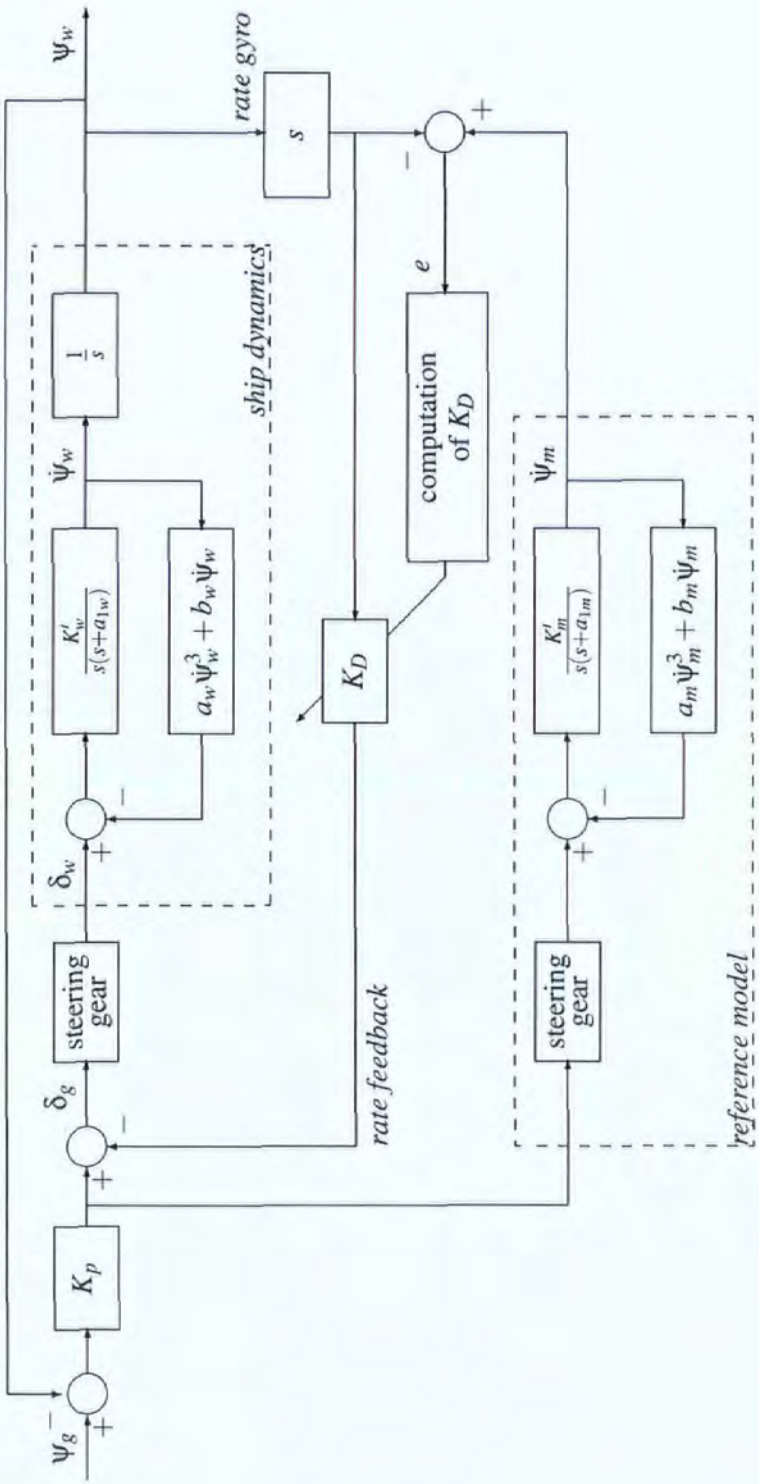


Figure 2.2. Model Reference Adaptive control (MRAC) - Block Diagram [8]

2.5 Modern Control Algorithms

It has been demonstrated that adaptive control is the control technique of the future. In recent research (M. Polkinghorne *et al* [82]) self-organising (self-tuning) [29, 125, 101, 115] methods have been used for the control of processes in uncertain, varying environments [60, 129, 61]. Existing adaptive controllers, *eg* self-organising fuzzy logic controllers (SoFLC) [81], learn by employing a heuristic approach. In order to learn, they initially start off with a poor performance. Because of their adaptive nature, the errors made during this low performance work are detected and the control parameters are adjusted in such a way as to avoid the same error in the future. The following sections describe the main differences between self-tuning controllers, SoFLC and the control idea PSoFLC of *this* research.

2.5.1 Self-tuning Autopilots

First developments of cost functions (criteria) for adaptive course-keeping autopilots were undertaken by Åström and Eykhoff [9] in 1971. The method used was based on a least squares parameter estimator and a minimum variance control technique.

$$J = \int_0^T (\psi_e^2 + \lambda_1 \dot{\psi} + \lambda_2 \delta^2) dt \tag{2.11}$$

Special attention should be given to the cost function. Assuming a vessel is left to yaw naturally without high frequency rudder corrections, the distance travelled during a 400 nautical miles journey does not increase more than a quarter of a mile when the deviation of the course remains $\pm 2°$ [73]. However, each rudder movement causes a drag and so a loss in forward speed and increased fuel consumption.

In 1975, Clarke and Gawthrop [29] developed a more generalised self-tuning controller. Publications in the late 70's and early 80's show the applicability of self-tuning controllers to the marine field [69, 43].

2.5.2 Optimal Control

It has been demonstrated by Burns [22] that it is possible to design an optimal multi-variable ship guidance system that controls position, heading and speed simultaneously, and such a

system can work within the constraints required in port approaches.

2.5.3 Neural Networks

The first notable paper utilising neural networks (for the principles on neural networks *see* section A) for the ship control application was published by Endo *et al* [31]. The training data to teach the neural network were generated by a PD controller. Further work in this field has been published by the author of this thesis [23, 24] and by many other researchers [11, 39, 46, 50, 97, 107, 113, 116, 121, 123, 120, 133, 132, 134]. A key paper was published by Hearn [41] where the use of a back-propagation neural network for on-line learning was detailed. In reality, the controller was not truly learning on-line, but was using a relatively fast computer in order that the learning could be achieved within the sampling time of the system. The training of the network was finished within approximately 0.5 seconds.

The back-propagation learning algorithm is based on the gradient (steepest descent) method. It minimises an error function. In the case of back-propagation, the error (E) of a neuron is defined as:vspace-5mm

$$E = \frac{1}{2}(d - y)^2 \tag{2.12}$$

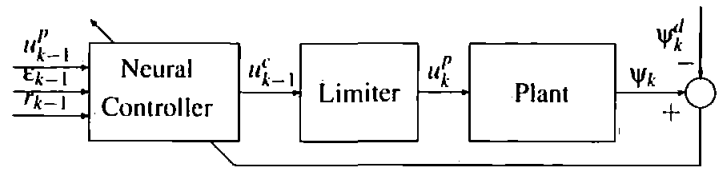
where: E ... error,

d ... desired output,

y ... actual output (*ie* sensor reading of the plant's response).

The desired output vector, in the case of a ship autopilot a single output, contains only the desired course. The system response is a function of the rudder angle and using the chain rule to derive a error measure for each individual neuron a control signal can be learned which minimises the difference between desired and actual course (figure 2.3). Further work in this field is being undertaken by Zhang *et al* [131, 132, 42]. More work on track keeping and related tasks such as rudder roll stabilisation and course keeping has been undertaken by several researchers [120, 105, 87].

There have been publications in the field of predictive control, such as Montague *et al* [68], Saint–Donat [92] and Tolle [111, 112].



where: ψ ... heading angle, k ... time step k
u ... surge, p ... plant
ε ... error, c ... current
r ... yaw rate, d ... desired.

Figure 2.3. Direct Neural Control Scheme [41]

2.5.4 Fuzzy Logic

A further method of simulating human behaviour is achieved by using linguistic variables and derived rules. The controller's task is to use a human-like way of thinking. The thoughts are placed into a knowledge base in the form of rules (rulebase), and the inputs are given in a fuzzified format. The use of so called fuzzy sets supports the human way of expressing every day actions and understandings. Fuzzy sets represent the mathematical equivalent of linguistic variables, *eg* tall, hot, cold, *etc*, used by the human language to express relationships and/or rules.

Nowadays, even more advanced techniques are used. Self-organising fuzzy logic control [26, 46, 60, 81, 82, 105, 106, 34, 33, 107, 109] or SoFLC combined with the model reference adaptive control (MRAC) technique [27, 53, 57, 99] is a recent development in this field to-date.

The principles of fuzzy logic are outlined in appendix B. It is necessary to understand the principles and functionality of both, neural nets (*see* appendix A) and fuzzy logic in order to understand the underlying aim of this research.

A neuro-fuzzy hybrid system called ANFIS (Adaptive-Network-based Fuzzy Inference System) was developed and introduced by Jang [48] in 1993. Since then researchers such as Sutton and Craven [104] have used this technique successfully for the guidance of autonomous vessels. This system uses a fuzzy system as an input layer. The successive layers (layer 2, 3 and 4) are artificial neurons. The final output of the ANFIS system is not a single number or vector as in supervised neural networks, moreover it activates a function with parameters. Each neuron in the output layer represents a different set of parameters for this

output function.

2.6 Summary

This chapter provided an historic overview of control systems in maritime applications. With the recent advances in technology other revenues of control can be pursuit to give an even better performance over a wider operating range. The direction of research and technology can be seen in non-linear and adaptive control, enabling the controller to change its control parameters, *ie* when changes in the operating environment occur. Some aspects of model reference adaptive control (MRAC) were explained. This technique is using an internal mathematical representation of the expected behaviour of the process. If the real response varies to that of the reference model, controller parameters are adjusted in such a way that the real behaviour is following the referenced response more closely.

Chapter 3

Heading Control using PD, Fuzzy Logic and Self-organising Fuzzy Logic

3.1 Controlling the Vessel with a Proportional + Derivative (PD) Controller

Since the mathematical description of the vessel used for the simulations already contained an integral term (Nomoto model), the controller does not need to contain one as well [69].

The controller was tuned at full speed using the following technique as explained in Cetrek's user manual for the 715/730 Autopilot Series [25]. This technique is easily understood by customers and leads to good control performance.

The differential gain was set to zero and the proportional gain increased gently until marginal stability was achieved. Then the differential gain was increased to 'drive back' the oscillation and to decrease the overshoot. Given the general form of a PD controller (*see* also Chapter 1)

$$\delta_d = K_p \cdot \psi_e + T_d \cdot \dot{\psi} \tag{3.1}$$

the following values were found:

$$K_p = 1.0,$$

$$T_d = 4.44\text{s}.$$

The tuning has been performed under calm conditions at full speed. The boat simulation

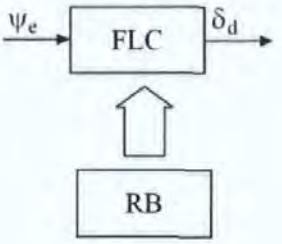


Figure 3.1. PSoFLC – FLC only

developed by Browning [20] is a key element for all further tests. This simulation was used during all development states of the various controllers considered in this study.

3.2 Fixed Rulebase Fuzzy Logic Controller

The testbed as described in appendix C.2 contains a fixed rulebase fuzzy logic controller, referred to as FLC[1]. This controller (figure 3.1) is the first step to the development of the full PSoFLC. In fact, the FLC is part of both of the more advanced systems the SoFLC and the Predictive SoFLC. The FLC is implemented using fuzzy singletons in the output window and a 7×7 rulebase. The output window can theoretically consists of 49 fuzzy singletons. Each combination of rules $E_k \rightarrow C_k$ has its own fuzzy output set ($E_k \rightarrow C_k \rightarrow O_k$). An initial rulebase using 9 named output singletons is laid out in table 3.1.

Figure 3.2 shows the fuzzy input windows as they are used in all fuzzy logic controllers of this research. The input windows stay fixed. The heading error ranges from $-30°$ to $+30°$. If the heading error exceeds the range it saturates. The shape and placement of the

[1] for an overview of the theory of fuzzy logic please refer to appendix B

Table 3.1. Fixed Rulebase FLC: Rulebase

	error rate						
	NB	NM	NS	Z	PS	PM	PB
NB	+30	+30	+30	+30	+15	+5	+2
NM	+30	+30	+15	+2	0	0	-2
NS	+15	+5	+5	+2	0	-2	-5
Z	+5	+5	+2	0	-2	-5	-5
PS	+5	+2	0	-2	-5	-5	-15
PM	+2	0	0	-2	-15	-30	-30
PB	-2	-5	-15	-30	-30	-30	-30

	error rate						
	NB	NM	NS	Z	PS	PM	PB
NB	PVB	PVB	PVB	PVB	PB	PM	PS
NM	PVB	PVB	PB	PS	ZZ	ZZ	NS
NS	PB	PM	PM	PS	ZZ	NS	NB
Z	PM	PM	PS	ZZ	NS	NM	NM
PS	PM	PS	ZZ	NS	NM	NM	NB
PM	PS	ZZ	ZZ	NS	NB	NVB	NVB
PB	NS	NM	NB	NVB	NVB	NVB	NVB

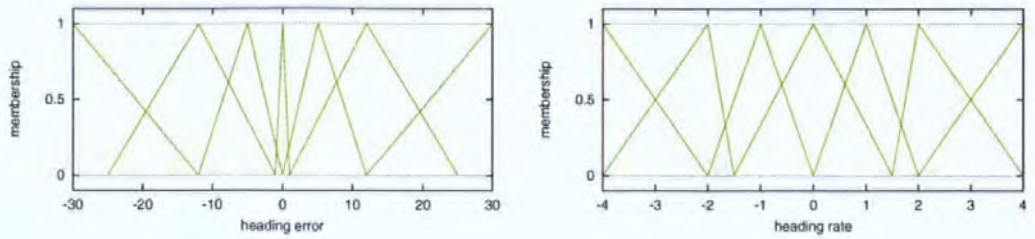


Figure 3.2. Fuzzy Input Windows

fuzzy sets are chosen in such a way that a finer control is achieved around the set point ($\psi_e \approx 0$) for course keeping and coarser control for course changing ($|\psi_e| \gg 0$). Similar, the heading rate ($\dot{\psi}$) ranges from -6 °/s to $+6$ °/s. When fed into the controller, the turning rate simply saturates on these values. The fuzzy sets in the heading rate input window are equally placed and regularly shaped.

In order to evaluate the fuzzy logic rulebase, a method was required to visualise the input/ output relationships of the fuzzy logic controller. To this aim a three–dimensional plot called the control surface was used. In these plots the two input variables, heading error and yaw rate, were plotted as x and y axes respectively. The third dimension z was the single output variable of the controller, *ie* the desired rudder angle. In the surface plot (figure 3.3), the knowledge of the controller was represented.

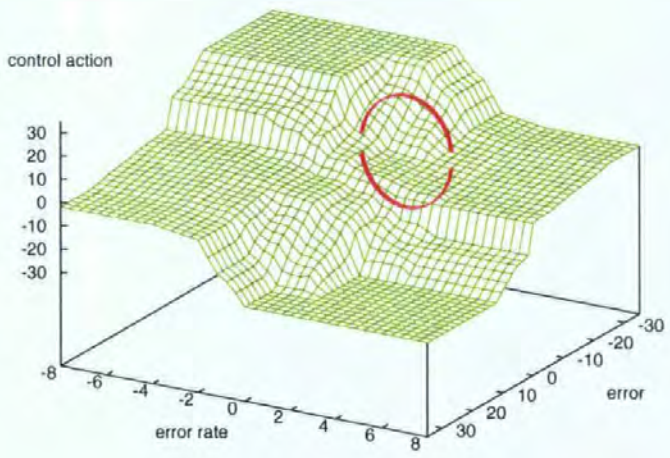


Figure 3.3. Control Surface using singletons

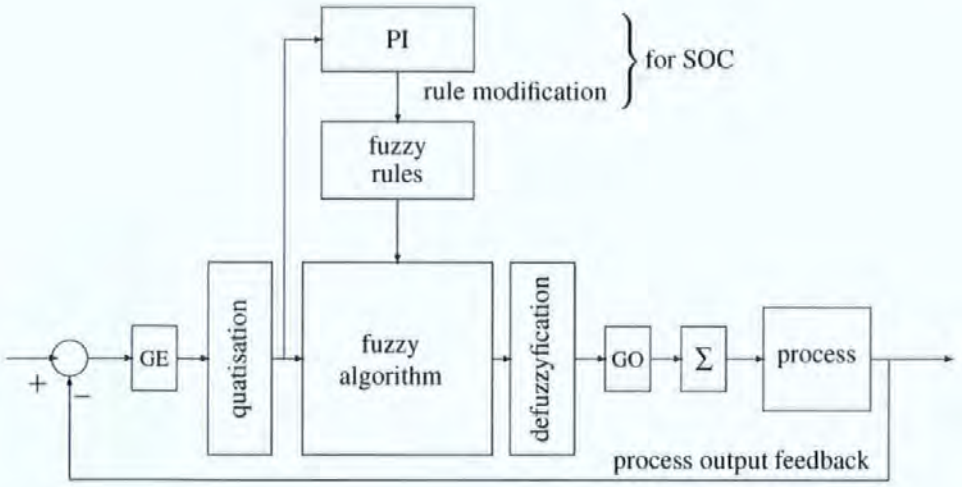


Figure 3.4. Block Diagram of a Self-organising Fuzzy Logic Controller (SoFLC) [102]

3.3 Self-organising Fuzzy Logic Control (SoFLC) – Overview

In very recent years, words like *self-tuning* or *self-organising* have been connected with modern techniques including fuzzy logic and neural networks [108, 106, 34, 33].

In connection with fuzzy logic, the word *self-organising* does not fully describe the method of adapting the fuzzy logic controller to a new environment. In fact, most algorithms only change the actual value of an output rule rather then the structure of the rulebase. The expression *self-tuning* is a more accurate description of the process.

With self-tuning algorithms, the two tasks of straightforward control and gradual learning are combined. In terms of fuzzy logic, the control is performed by the fuzzy logic methods described in appendix B (p. 142). Simultaneously, the operating environment is observed as well as the controller's response within that environment. Adaptation can now be achieved by utilising the obtained information to change the fuzzy rulebase in order to improve future outputs of the controller. Information on acceptable and unsuitable combinations of environment variables and associated control operation gives a measure of performance. This information may be stored in a similar form as the rulebase of the fuzzy logic controller, the *Performance Index*.

A block diagram of the SoFLC as used by Sugiyama [102] can be seen in figure 3.4. The controller works as follows: If the observation of the operating environment indicates that

the plant output provides a satisfactory level of performance then no alteration of the rulebase is performed. Conversely, as the performance level deteriorates, then the performance index indicates the magnitude of adjustment required to drive the plant output back to a satisfactory level.

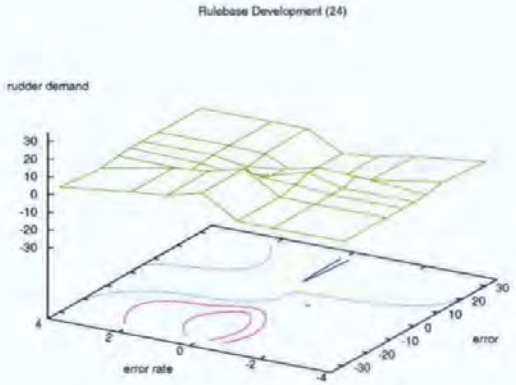
The disadvantage with this self-organising technique using a performance index is that the performance measured corresponds to a control action, hence rudder change, n time steps back in the past. It is very difficult to relate the current stage of the vessel back to the rudder action which has caused this state. The time delay is about one time constant of the vessel, approximately 63% of the steady state yaw rate ($\dot{\psi}$) have been reached. If a longer time period is considered, then it is more likely that another rudder change was applied. However, rudder changes made earlier still have effect on the vessel. Furthermore, and more important, the control action has been applied and caused the poor, present state.

This technique has been used in several applications and its performance and reliability has been tested and approved. The controller used is a standard fuzzy logic controller using fuzzy singletons in the output window. A fuzzy singleton is defined as a fuzzy set where only one element of the universe of discourse has a membership value greater than zero (*see* figure B.12 in appendix B). This simple set will be used to keep the self-tuning algorithm easy to understand as well as easy to implement.

3.4 The Self-organising Fuzzy Logic Controller (SoFLC)

This section discusses the self-organising method using historic data as previously introduced by [125]. It discusses and explains the fundamental principles of self-organising fuzzy logic controllers. The performance of this kind of controller is demonstrated in simulations and serves as a reference (benchmark) to evaluate the achieved performance of the *Predictive Self-Organising Fuzzy Logic Controller* (PSoFLC) of this research.

The input fuzzy windows used in both self-organising controllers have already been introduced (figure 3.2, page 25). Considering the fuzzy windows again, it can be noticed that the input window for the heading error is highly irregular, using asymmetrically shaped sets. This layout allows to emphasise regions of control, in this case, give finer control in the



the changed rulebase at 11:37:37.02 (24)

	NB	NM	NS	Z	PS	PM	PB
NB	0.00	0.00	0.16	12.41	6.97	5.41	0.55
NM	0.00	0.00	0.37	12.44	7.36	5.34	-2.23
NS	0.00	0.00	1.96	2.57	-0.61	-1.46	-0.56
Z	0.00	0.00	5.17	-0.30	-3.32	-1.04	0.00
PS	0.31	3.08	3.98	-0.59	-2.99	-0.02	0.00
PM	2.11	-4.19	-8.25	-9.97	-0.74	0.00	0.00
PB	0.20	-4.41	-8.26	-10.40	-0.01	0.00	0.00

Figure 3.5. Rulebase Visualisation

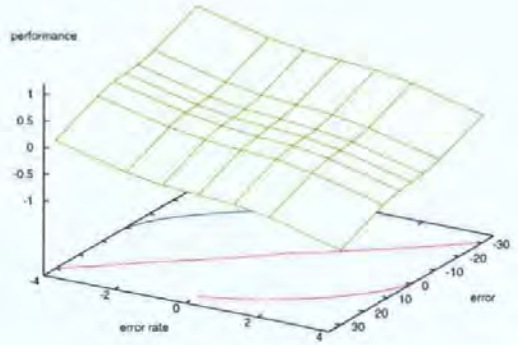
centre of the window, around zero heading error.

3.4.1 Structure of the Rulebase

The controller's intelligence is stored in its rulebase. A two-dimensional structure has been chosen to adapt a PD (proportional+derivative) control strategy. The error ψ_e (in rows) and the rate of error $\dot{\psi}_e$ (in columns) identify the structure. The table in figure 3.5 shows the final rulebase after completion of the 20° step response test manoeuvre which lasts about 12 min. For intermediate rulebases, please see appendix E. The inclusion of an integral term as additional input into the rulebase was not found to be necessary since this, the constant offset, can easily be achieved by shifting the whole rulebase to either side. The rulebase used does not need to be symmetrical around its centre point. A shift of the control plane (*see* control surface plot in figure 3.3) along one of the two input axis has the same effect. This shift results in a fixed value $\neq 0$ in the centre of the rulebase $\psi_e = 0$ and $\dot{\psi} = 0$. Effectively, the centre point of the demanded rudder movement is no longer 0° but a constant value $\neq 0°$. The self-adapting nature of the controllers considered make it possible to modify the rulebase to suit this requirement.

3.4.2 The Performance Index

The performance index (PI) states the performance of the controller. It shows how well or poorly the controller reacts to its desired state. The performance index is more than a performance measure. It reflects the desired response of the process. It therefore has build in



	rate of change of error						
error	NB	NM	NS	Z	PS	PM	PB
NB	1.00	0.52	0.42	0.40	0.38	0.29	-0.20
NM	0.71	0.23	0.14	0.12	0.09	0.00	-0.48
NS	0.63	0.15	0.06	0.03	0.01	-0.08	-0.57
Z	0.60	0.12	0.02	0.00	-0.02	-0.12	-0.60
PS	0.57	0.08	-0.01	-0.03	-0.06	-0.15	-0.63
PM	0.48	0.00	-0.09	-0.12	-0.14	-0.23	-0.71
PB	0.20	-0.29	-0.38	-0.40	-0.42	-0.52	-1.00

Figure 3.6. The Performance Index

information which can be seen as a reference model. The values of ≈ 0 indicate the desired response and incorporating this information into the performance index is process dependant. The performance index output is a measure which can be used directly to adjust the rulebase. The structure of the performance index used in this application copies the structure of the rulebase. This means that when a PD type control rule is used, then the performance index should also be of PD type, having the error and the rate as inputs. The measured parameter(s) of the process (plant behaviour) are used as inputs to the performance index. In the table (in figure 3.6) the rows are the heading error ψ_e and the columns are the rate of change of the heading error $\dot{\psi}_e$. In general, the performance index has the same, or at least a similar (using the same input variables), structure as the fuzzy rulebase used in the fuzzy inference (forward phase). The fuzzy implication as used in the fuzzy logic controller is here applicable too.

IF E **IS** E_k **THEN IF** C **IS** C_k **THEN** PI **IS** PI_k

$$E_k \rightarrow C_k \rightarrow PI_k \tag{3.2}$$

E_k represents a fuzzy subset from the universe of heading errors, C_k a subset from the change in error (rate) universe and PI_k represents to performance index output.

The performance index is a measure of the controllers quality. The smaller the absolute number, the better the performance, and the closer the process is to the desired state. The number and sign itself indicate the error that the utilised rules must be changed in order to increase performance, get closer to the desired state.

As an example: if the heading error is big ($\psi_e \gg 0$) and the vessel is turning quickly in the direction to reduce the error ($\dot{\psi} \ll 0$), then the vessel is at a desired state, and the controller acts in the desired way, that is, turning quickly to reduce the error. Note the small corrective value in the left bottom (right top) corner of the performance index. If, on the other hand, the error is big and the vessel turns away from the desired course (error and turning rate have the same sign), then the controller suggests a non-desirable action and the performance index indicates this with a number $\neq 0$.

The performance index itself is derived from PD (proportional+derivative) data. It represents a plane in a three dimensional space (*see* table in figure 3.6). Figure 3.6 shows the performance index in a three dimensional plot. The surface is smooth and monotone in each direction. The coloured curves in the x–y plane show the contours of the surface.

3.4.3 The Rulebase Update Algorithm

The performance index as discussed above gives an indication of the direction and magnitude by which the rulebase has to be changed.

As previously described by Procyk and Mamdani [83], the performance index only provides a measure of performance of the overall controller. Now, this output has to be converted into real, corrective, values to the process/ controller that should have been applied some control actions in the past causing the present poor performance. It is important and not trivial to link the current state to previous control action(s).

Before any analysis can be completed, the change on the control actuator has to take effect and a certain period of time has to pass by. This time delay or *delay in reward* (DEL) [102] is characterised by the time constants of the process. For this application a DEL of one time constant (DEL $= kT = 1 \times T_{\text{system}}$) has been chosen.

The rule changing algorithm consists of three main phases.

1. straightforward fuzzy logic control (fuzzification, fuzzy inference, defuzzification). The active rules and values are stored for later use in the tuning. A control output is created which is fed into the process (control actuator).

2. The process 'reacts' in an appropriate (in its characteristic) way and the actual output

value is measured by a device in the feedback loop of the control system.

3. This output value is forwarded to the performance index (PI) which will generate a measure of the controller performance. If poor performance is measured, adjustments of the rulebase are needed. Now, n time steps later, the rules and values responsible of this control output are changed according to the performance index and their influence toward the final output. So, if this combination of rules is activated in the future, the control output will be different, hopefully it will produce a smaller error than it did before.

There are a maximum of two rules overlapping each other in each of the input windows. This means, that in this application with two input windows, a maximum of four rules are active in the output window which need consideration for the defuzzification.

$$\begin{aligned} n_{\text{active}} &= n_{\text{ov}}^{\,n_{\text{input}}} \\ 4 &= 2^2 \end{aligned} \tag{3.3}$$

where:

n_{active} ... number of active rules,

n_{ov} ... number of overlapping fuzzy sets,

n_{input} ... number of input windows, dimensions.

Fuzzy singletons are used because of their easy implementation, The following equation represents the update:

$$R_{(t+)} = R_{(t-\text{DEL})} + \eta \cdot \mu(R_{(t-\text{DEL})}) \cdot PI_t, \tag{3.4}$$

where:

$R_{(t+)}$ ... new rule, the + indicates a future use,

$R_{(t-\text{DEL})}$ ... rule output at time $t - \text{DEL}$,

η ... gain,

$\mu(R_{(t-\text{DEL})})$ ... membership function (influence) of rule $R_{(t-\text{DEL})}$,

PI_t ... performance index output at time t which measures the process state at time t which is caused by a control output at time $t - \text{DEL}$.

This equation effectively moves the singleton along the universe of discourse. Since the performance index is scaled between ± 1 the gain to boost the output of the PI for rulebase adaptation is set to 3.5, $\frac{1}{10}^{th}$ of the maximum rudder movement. The physical rudder limit of the boat was $\pm 35°$. The values inside the rulebase saturate at $\pm 35°$. No overrules are used to guide the adaptation of the rulebase.

3.5 Summary

The disadvantages are clear since this technique is based on the assumption that, the controller output n time steps (DEL) previously is responsible for the present state of the process. If the process measurement indicates a poor state, these rules should be changed. This technique only allows to adjust control parameters which already performed to an unsatisfactory level and is therefore retrospective.

To avoid the application of a control action which does not improve the current situation, knowledge is required to assess the control action's effect on the current state. Using measurements, this knowledge is only available in the future. By relating the measured state to a control action back in time, rules can be identified which caused the current, measured state. A simulation (running faster than real time) could also give an indication of the control action's effect. This simulated state could give vital clues about the quality of the control action about to be applied.

Chapter 4

The Predictive Self-organising Fuzzy Logic Controller

This chapter will concentrate on the method used to make predictions of a future state of the ship (plant) which is then utilised to optimise the rulebase of the fuzzy logic controller. For this purpose, this section explains the principles of adaptive-modelling using neural networks, and their application in a predictive controller.

Self–organising controllers (SOC) use present data to evaluate control performance. Of course, the present performance is related to a control action in the past. This means that at the point in time that the control action is applied, the future effect of that action is unknown. The action to be applied can improve the situation, but it is also possible that it can make the situation worse. The aim for a controller should be to test and validate the effect of its action on the plant before the action is actually applied. This can be achieved by running a simulation faster than real-time which will obtain information about a future state (n time steps ahead) of the plant when a certain control action is applied. In a predictive controller, this future state is taken into account when the performance is validated. To run a simulation, a mathematical model of the plant and the environment is required. This mathematical model is called the *Predictor* in this application.

An innovative form of mathematical modelling is used to forecast the plant's behaviour and is explained later in section 4.2.

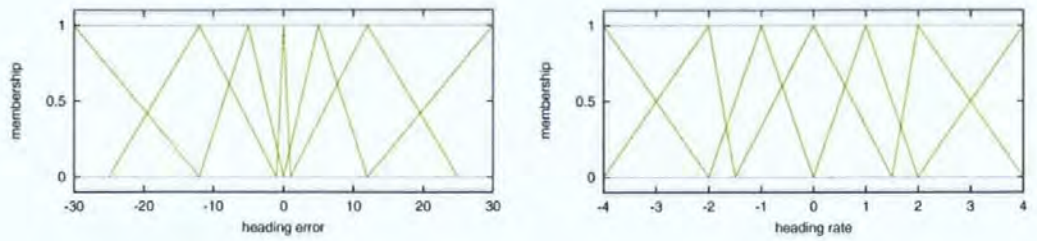
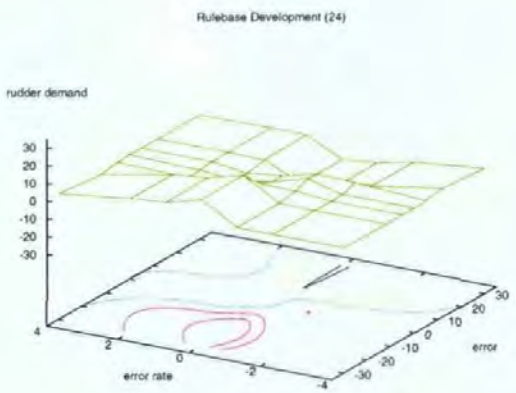


Figure 4.1. PSoFLC – Fuzzy Input Windows

4.1 The Embedded Self-organising Fuzzy Logic Controller (SoFLC)

4.1.1 Rulebase Structure

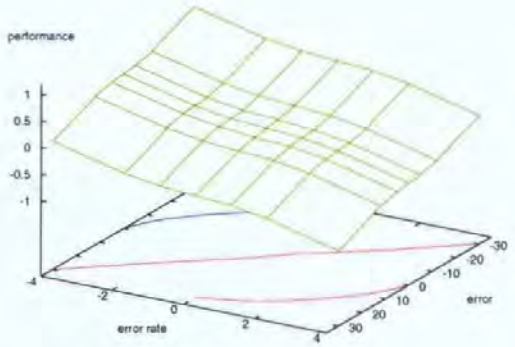
The *Predictive Self-organising Fuzzy Logic Controller* utilises the rulebase structure of the FLC as discussed in section 3.2. The controller uses two input variables the heading error (ψ_e) and turning rate ($\dot{\psi}$). This defines the structure of the rule base. A maximum of two fuzzy sets overlap in each of the two input windows. This results in a maximum of four active rules in the output window which combined and defuzzified form the final output value. The input windows are shown in figure 4.1. These are the same input windows as found in the FLC and SoFLC from the previous chapter (section 3.2, page 25). A sample rulebase can be seen in figure 4.2.



the changed rulebase at 11:37:37.02 (24)

	NB	NM	NS	Z	PS	PM	PB
NB	0.00	0.00	0.16	12.41	6.97	5.41	0.55
NM	0.00	0.00	0.37	12.44	7.36	5.34	-2.23
NS	0.00	0.00	1.96	2.57	-0.61	-1.46	-0.56
Z	0.00	0.00	5.17	-0.30	-3.32	-1.04	0.00
PS	0.31	3.08	3.98	-0.59	-2.99	-0.02	0.00
PM	2.11	-4.19	-8.25	-9.97	-0.74	0.00	0.00
PB	0.20	-4.41	-8.26	-10.40	-0.01	0.00	0.00

Figure 4.2. PSoFLC – Fuzzy Rulebase



	rate of change of error						
error	NB	NM	NS	Z	PS	PM	PB
NB	1.00	0.52	0.42	0.40	0.38	0.29	-0.20
NM	0.71	0.23	0.14	0.12	0.09	0.00	-0.48
NS	0.63	0.15	0.06	0.03	0.01	-0.08	-0.57
Z	0.60	0.12	0.02	0.00	-0.02	-0.12	-0.60
PS	0.57	0.08	-0.01	-0.03	-0.06	-0.15	-0.63
PM	0.48	0.00	-0.09	-0.12	-0.14	-0.23	-0.71
PB	0.20	-0.29	-0.38	-0.40	-0.42	-0.52	-1.00

Figure 4.3. The Performance Index

4.1.2 Performance Index

Like in the FLC, the performance index is of the same structure as the rulebase. The performance index as used in the SoFLC and extended PSoFLC is visualised in figure 4.3. The performance index is normalised between ± 1. A scaling factor can be applied to extend the range and amplify the output.

The performance index used within the PSoFLC is the same as used in the SoFLC described in the previous chapter (section 3.4.2, page 28).

4.1.3 Rulebase Adaptation

$$R_{(t+)} = R_t + \eta \cdot \mu(R_t) \cdot PI_{(t+\mathrm{TIA})}. \tag{4.1}$$

where:

$R_{(t+)}$ ... new rule, the $+$ indicates a future use,

R_t ... rule output at time t (now),

η ... learning gain,

$\mu(R_t)$ ... membership function (influence) of rule R_t,

$PI_{(t+\mathrm{TIA})}$ ... performance index output at time $(t+\mathrm{TIA})$ which measures the process state at time $(t+\mathrm{TIA})$ which is caused by a control output at time t (now).

The learning gain η is set to $\eta = 0.35$. This is $\frac{1}{10}^{\text{th}}$ of the gain applied within the SoFLC, 10 is the maximum number of predictor cycles before a control action is passed to the process.

The values inside the rulebase saturate at $\pm 35°$, limiting the rudder demand δ_d within the physical limits of the rudder. No overrules are used to guide the adaptation of the rulebase.

Following the analogy of Sugiyama [102], using predictor terminology, the DEL becomes the *Time–in–Advance* (TIA). For this application a TIA of two time constants (TIA = $kT = 2 \times T_{\text{system}}$) has been selected. Two time constants have been chosen to allow the rudder change to take considerable effect on the hull's movement. Together with the performance index, an improvement in the vessel's state is therefore expected within this time period. Choosing only one time constant provides the system with insufficient time to change the vessel's turn rate, and approach the steady state response. Selecting a TIA value of three time constants would allow the vessel to reach a steady state. However, operating in a real environment, rudder changes are applied more frequently than once every 3 time constants. Investigation has shown that selecting 2 time constants as TIA is therefore a compromise between the two scenarios. During the prediction, the rudder is not changed, so the effect of the rudder in during this time is exactly determined. No other rudder input influences the future state of the vessel, so a future state can be related to one control action more reliably and accurately.

Figure 4.4 shows the block diagram of the rulebase adaptation module only. The predicted data is fed into the performance index which returns a measure of performance. Utilising this output from the performance index, the rulebase is updated. If the performance index indicates a good performance ($|PI| \approx 0$) then the calculated rudder is applied to the ship, otherwise the process of calculating a rudder demand δ_d, predicting, performance measuring is repeated until the maximum number of cycles (10 in this application) is reached or the performance index gives no reason for further repetition. The block diagram shows the parts as found in the PSoFLC. The data utilised by the performance index is coming from the predictor and is the heading error ψ_e and turning rate $\dot{\psi}$.

4.2 The Predictor

To find a fully representative model of the plant is always a problem in control engineering. It has been shown [117, 85, 132] that neural networks are well able to learn the transient

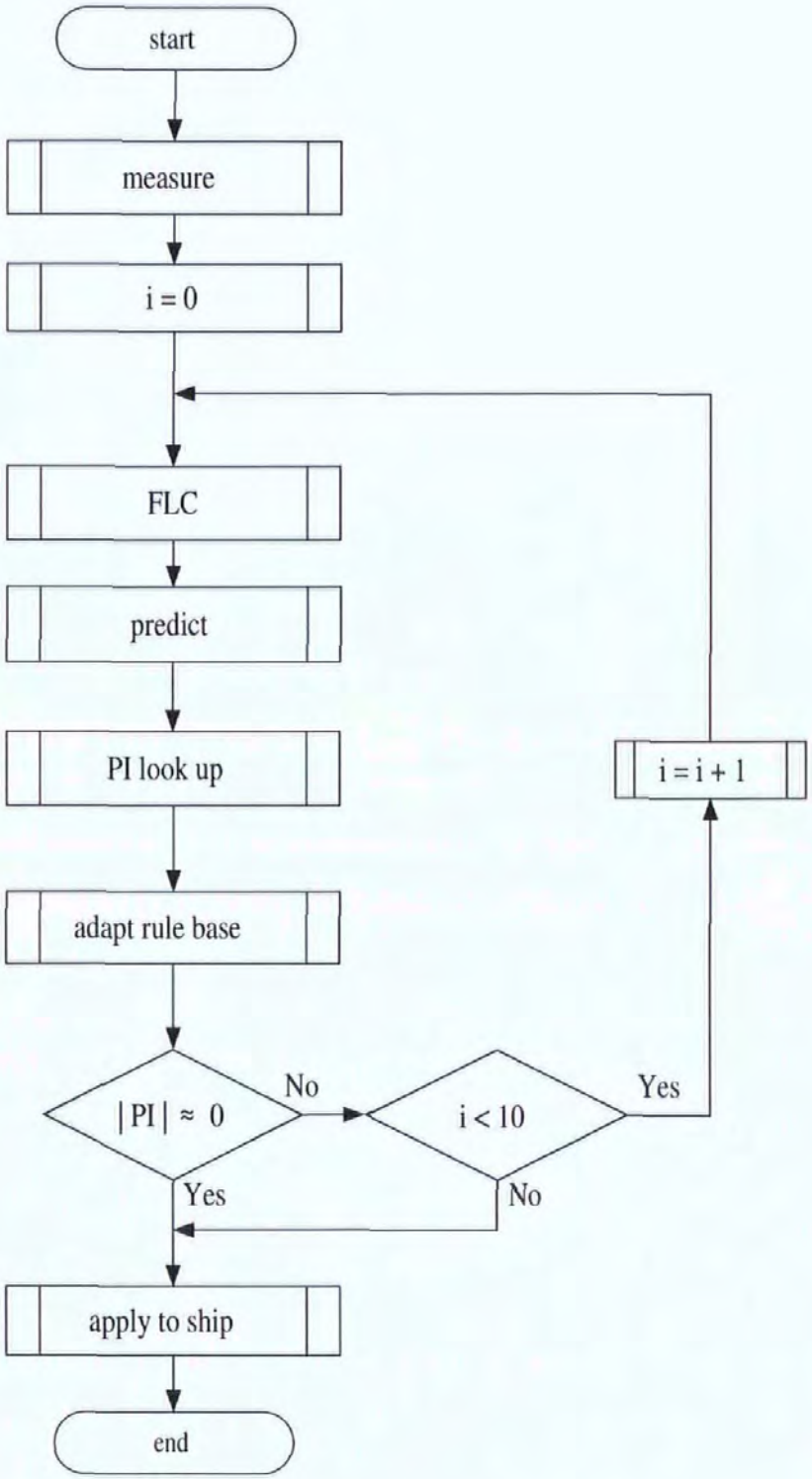


Figure 4.4. PSoFLC – rulebase adaptation

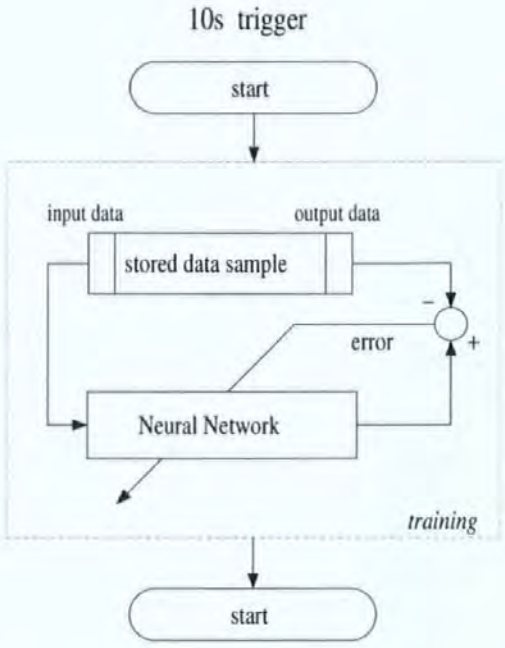


Figure 4.5. PSoFLC – Predictor Adaptation

behaviour of a process. So, it should be possible to obtain a very specific mathematical model by measuring the time transient behaviour of the vessel and teach, on-line, a neural network utilising this data. The neural network mathematical model is the adaptive module referred to as the *Predictor*. This Predictor, a block diagram of the training is displayed in figure 4.5, will then be employed in the PSoFLC. As long as the training data set contains most, if not all, the significant data, the neural network back propagation learning algorithm will relate the inputs to the outputs and deliver a mathematical model which, in control terms, represents the transfer function.

Ideally the predictor should be a 12 degree of freedom mathematical model, containing 12 state variables, three translation (x, y, z) and three rotational (ϕ, θ, ψ) plus their first derivatives.

The set of Euler's differential equations are used to discuss the selection of parameters for the predictor.

$$
\begin{aligned}
\text{surge}: \quad & m\Big\{\dot{u}+qw-rv-X_G(q^2+r^2)+Y_G(pq-\dot{r})+Z_G(pr+\dot{q})\Big\} &&= X \\
\text{sway}: \quad & m\Big\{\dot{v}+ru-pw-Y_G(r^2+p^2)+Z_G(qr-\dot{p})+X_G(qp+\dot{r})\Big\} &&= Y \\
\text{heave}: \quad & m\Big\{\dot{w}+pv-qu-Z_G(p^2+q^2)+X_G(rp-\dot{q})+Y_G(rq+\dot{p})\Big\} &&= Z
\end{aligned}
\tag{4.2}
$$

$$
\begin{aligned}
\text{roll}: \quad & I_{xx}\dot{p}+(I_{zz}-I_{yy})qr+m\Big\{Y_G(\dot{w}+pv-qu)-Z_G(\dot{v}+ru-pw)\Big\} &&= K \\
\text{pitch}: \quad & I_{yy}\dot{q}+(I_{xx}-I_{zz})rp+m\Big\{Z_G(\dot{u}+qw-rv)-X_G(\dot{w}+pv-qu)\Big\} &&= M \\
\text{yaw}: \quad & I_{zz}\dot{r}+(I_{yy}-I_{xx})pq+m\Big\{X_G(\dot{v}+ru-pw)-Y_G(\dot{u}+qw-rv)\Big\} &&= N
\end{aligned}
\tag{4.3}
$$

Where:

$$
\begin{aligned}
X &= X_H+X_P+X_R+X_O \\
Y &= Y_H+Y_P+Y_R+Y_O \\
Z &= Z_H+Z_P+Z_R+Z_O \\
K &= K_H+K_P+K_R+K_O \\
M &= M_H+M_P+M_R+M_O \\
N &= N_H+N_P+N_R+N_O
\end{aligned}
\tag{4.4}
$$

with:

$_H$ ... forces/ moments acting on the hull

$_P$ ... forces/ moments acting on the propeller

$_R$ ... forces/ moments acting on the rudder

$_O$ ... others forces/ moments (such as stabilisers, fins, other external forces such as wind, wave current *etc*)

By moving the the ships co-ordinate system into the centre of gravity of the ship (figure 4.6), the equations in (4.2) and (4.3) can be simplified to:

$$
\begin{aligned}
\text{surge}: \quad & m\Big\{\dot{u}+qw-rv\Big\} &&= X \\
\text{sway}: \quad & m\Big\{\dot{v}+ru-pw\Big\} &&= Y \\
\text{heave}: \quad & m\Big\{\dot{w}+pv-qu\Big\} &&= Z \\
\text{roll}: \quad & I_{xx}\dot{p}+\Big(I_{zz}-I_{yy}\Big)qr &&= K \\
\text{pitch}: \quad & I_{yy}\dot{q}+\Big(I_{xx}-I_{zz}\Big)rp &&= M \\
\text{yaw}: \quad & I_{zz}\dot{r}+\Big(I_{yy}-I_{xx}\Big)pq &&= N
\end{aligned}
\tag{4.5}
$$

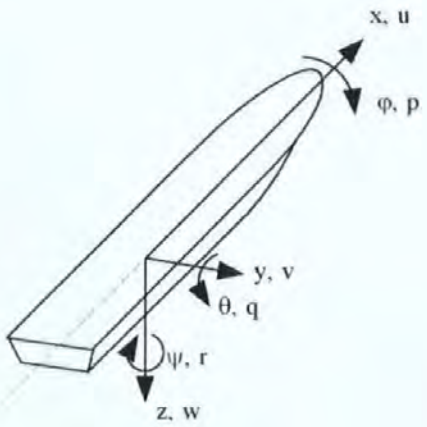


Figure 4.6. The Ship Co-ordinate System

$$
\begin{aligned}
X &= X_{\dot{u}}\dot{u} + X_u(u+u_c) + X_{uu}u^2 + X_{uuu}u^3 + X_{vv}v^2 + X_{rr}r^2 \\
&\quad + X_{\delta\delta}\delta_A^2 + X_{un}un_A + X_{nn}n_A^2 + X_{ua}u_a + X_{zz}z^2 + X_{\theta\theta}\theta^2 \\
Y &= Y_{\dot{v}}\dot{v} + Y_v(v+v_c) + Y_{\dot{r}}\dot{r} + Y_r r + Y_{nn}n_A^2 + Y_{vvv}v^3 + Y_{rvv}rv^2 \\
&\quad + Y_{nn\delta}n_A^2\delta_A + Y_{nn\delta\delta\delta}n^2\delta_A^3 + Y_{\delta vv}\delta_A v^2 + Y_{va}v_a \\
Z &= Z_{\dot{w}}\dot{w} + Z_w w + Z_z z + Z_\theta\theta + Z_q q \\
K &= K_{\dot{p}}\dot{p} + K_p p + K_\delta\delta_a \\
M &= M_{\dot{q}}\dot{q} + M_q q + M_\theta\theta + M_z z + M_w w \\
N &= N_{\dot{v}}\dot{v} + N_v(v+v_c) + N_{\dot{r}}\dot{r} + N_r r + N_{nn}n_A^2 + N_{vvv}v^3 + N_{rvv}rv^2 \\
&\quad + N_{nn\delta}n_A^2\delta_A + N_{nn\delta\delta\delta}n_A^2\delta_A^3 + N_{\delta vv}\delta_A v^2 + N_{va}v_a
\end{aligned}
\tag{4.6}
$$

According to Burns [21], the linear coefficients $N_\delta\delta_A$, $N_r r$, $N_v v$ and the nonlinear coefficient $N_{rvv}rv^2$ have major relevance.

Considering only the linear terms from the equations above (equation 4.5) the yaw rate relationship can be written:

$$\dot{r} \sim \frac{N_r}{I_{zz}}r + \frac{N_\delta}{I_{zz}}\delta \tag{4.7}$$

It was found that a step in time characterising the state of the small motorised craft as used here in the validating simulation is sufficiently described by the following state variables: the actual heading, yaw rate, forward speed, desired rudder angle, current rudder angle, roll and pitch angle as well as the time to the previous sample.

Actual Heading (ψ) Heading is the quantity to control and is the most important value to be included in the mathematical model. By including heading into the model, orientation dependent data (tide, current, wind) is considered. This way, more knowledge about the environment is obtained and embedded into the mathematical model.

Yaw Rate ($\dot{\psi}$) being the first derivative of the quantity to control, this is an important value to consider.

Forward Speed ($u = \dot{x}$) A change in forward speed changes the steering characteristics of the vessel. This can be seen by the influence of the forward speed v in the yaw equation (set of equations 4.6).

An increase in forward speed causes the vessel to lift out of the water. This then results in a reduction in the 'carried mass' and reduces the resistance. This non-linear term ($\sim v^2$) has a major influence to the systems state [21], and it is therefore included as a state variable. The position in x can now be easily obtained using the direction of travel (yaw, ψ) and the forward speed. However, for piloting purposes as considered in this research, the actual position is not a relevant information.

Actual Rudder Angle (δ_a) **/ Desired Rudder Angle** (δ_d) The rudder is one of the two actuators the boat has. The second one being the propeller directly linked to forward speed. The rudder is linked to the yaw rate and is the prime actuator responsible for a change in the vessels orientation. It is therefore considered as an input into the model. Both rudder angles, desired and actual, are included into the model to simulate the response of the steering gear.

Pitch Angle (θ) **/ Roll Angle** ϕ First tests revealed that the model was not very accurate when the roll and pitch angles were not included. This might be explained by the close coupling of pitch to heave which is not considered either, and rudder to roll. A linear relationship can be noticed between the roll moment and the rudder (equation 4.6). This explains the improvement of the model when the roll angle was included as a model input.

By including one of the two coupled variables a significant improvement of the model was achieved. Including those two values into the input vector, more information about the

environment can be obtained. The neural network should be able to estimate a magnitude of the sea state as well as anticipate low frequency wave patterns.

Time to the Previous Sample (Δt) The sampling time is not constant. To draw any conclusions from the change in any of the above values, the time has to be included to connect the values.

Nomoto's simple model of a ship

$$\frac{\psi}{\delta} = \frac{1}{s} \cdot \frac{K}{s+a} \tag{4.8}$$

indicates a second order system, a system of first order plus an integrator. A linear relationship can be described with two points. If a rate term (first derivative) is included, a third point is required to make a smooth transition from one section to another, this means that the function is differentiable in all points. Hence three points find consideration in the vector.

4.2.1 Predictor Requirements

The time for learning and predictions is limited. Ideally, a continuous controller without time delay will result in the best control. It is therefore important to use an as short as possible sampling time but at least half the time constant of the process under control. 40 records (input/ output data pairs) are stored as training data. These 40 records represent the last 20s of the vessel's transient behaviour (sampling time 0.5s). Since the training of the network happens approximately every 10 seconds, a sample window of 20 seconds allows each set to be exposed to the neural network together with one previous 10s window and one future. A graphical representation of the timeline is displayed in figure 4.7. This enables the algorithm to be exposed to already learnt data and new data to allow for a smooth transition. It also helps the algorithm to restore and interpolate the relevant data. Measurements and analysis have shown the dominant time constant of the ship in yaw to be approximately 1.25s travelling at full speed. This relates to approximately 1,050 rpm and 21 knots ($39\ km \cdot h^{-1}$) for this model. The time constant increases with reducing forward speed. Travelling at full speed is therefore the 'worst case scenario' with respect to time available for learning and predictions. Also, at full speed the vessel plus controller are most sensitive/ acceptive to disturbances and

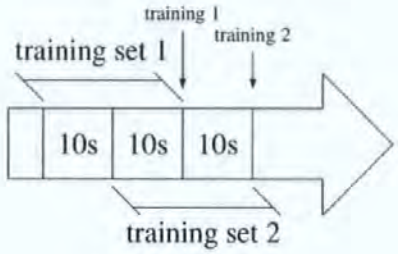


Figure 4.7. Timeline

noise. The chosen sampling time of 0.5s is just less than half the time constant of the vessel in yaw. The sampling time of 0.5s was chosen not to overload the processor with too much data but still 'over sample' twice the dominant frequency. Using a shorter time would overload the controller with messages which then will cause a long backlag of queued messages. A time delay would be the result. A longer time was not feasible, as the dynamics would not be sufficiently captured. The boat's time constant changes with a any change in speed. By increasing the speed, the vessel becomes more responsive, the time constant becomes shorter.

One training cycle of the network is completed within the time window of one time constant. The main work has been carried out on a PC with a Pentium 266 MHz processor and 64 MB RAM running OS/2. Running on a PC with more processing power, the learning time could be decreased even further which results in less idle time of the actual control loop and improved performance. A further development stage of the software could result in a parallel-process learning routine. The learning could be carried out in the background and would therefore not interfere with the control loop. The current set-up is semi-parallel; every 10s one complete teaching cycle is performed to update the model.

The structure of the neural network (*see* figure 4.8) as used in the predictor shows all the input variables on the left hand side. The four output values, heading acceleration, roll, pitch and rudder angles are located on the right hand side of the diagram. It can be seen, that the network consists of two hidden layers with twelve neurons in the first hidden layer and five neurons in the second. Other researchers, *ie* Balasuriya and Hoole [11], have chosen a similar network structure for neural network control of marine craft. Here a two hidden layer neural network is trained to control the heading of a 161m cargo vessel.

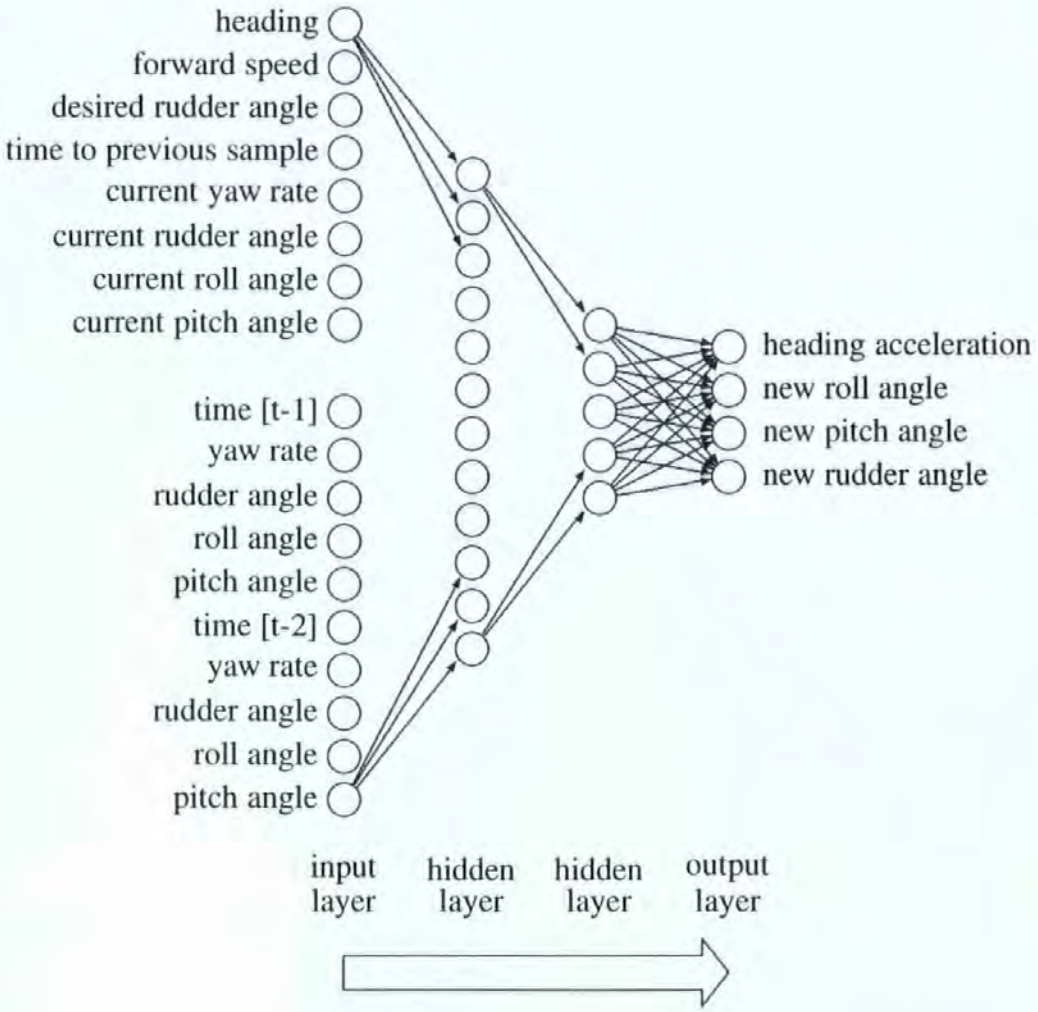


Figure 4.8. Structure of the Neural Network inside the Predictor Module

After extensive experimentation it proved best to have a network with two hidden layers. A single hidden layer network was unable to capture the the plant's behaviour accurately even after long training times.

The transfer function of all neurons is the *Sigmoid* function as described in appendix A. The size (number of hidden layers and number of neurons in each of the layers) of the neural network represents the degree of freedom (DOF) of the entire system. The predictor shall simulate the behaviour of the vessel and the working environment.

It is accepted that no direct relationship exists between the the degree of freedom of the system and the number of neurons and their connectivity within the network. Experiments have shown [56] that, the more complex a problem (*ie* higher the degree of freedom), the more neurons are needed to identify the problem.

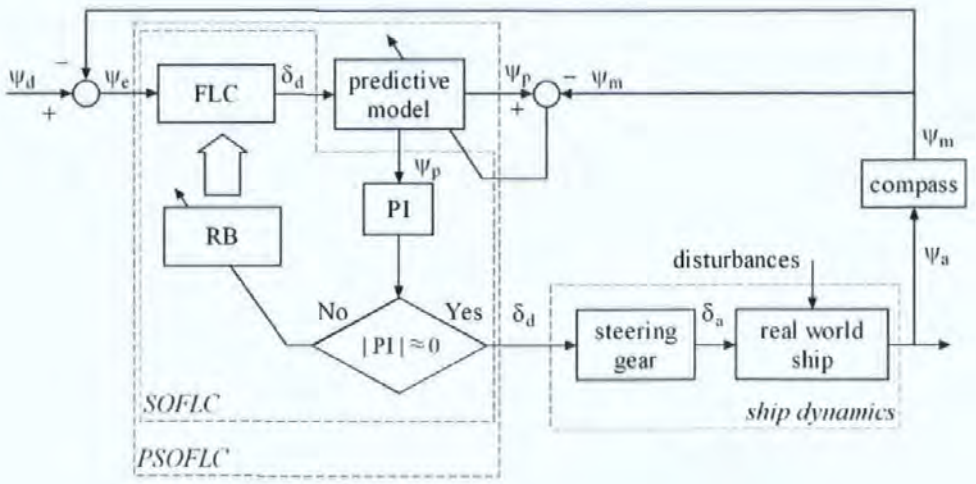


Figure 4.10. Predictive SoFLC

- the feedback sensors.

The controller itself is a combination of 3 modules (figures 3.1, 4.4 and 4.5). The individual modules can also be seen as layers. The most primitive layer, providing basic control, is the FLC (figure 3.1). The FLC utilises the rulebase (RB). The next layer is the 'rulebase adaptation module'. The 'rulebase adaptation module' (figure 4.4) is responsible for the adjustments of the rulebase employed by the FLC to generate a desired rudder angle. The Performance Index (PI) provides the necessary value to do exactly that.

- The PI can be fed with values from either (not combined) states, measured or predicted.
- The PI is not concerned with the source of the state data it evaluates *eg* in the SoFLC it is a current state that is evaluated, in the PSoFLC it is the predicted state.

The desired control command, hence the desired rudder angle, is fed into the predictor. The predictor as explained in detail in section 4.2 provides an estimated future state (future heading) of the vessel some time ahead. This new state is evaluated using the PI. All necessary alteration of the rules, which are responsible for the predicted state, are modified when a poor performance is observed. The Predictor always contains the current steering characteristics of the vessel. If those characteristics change, then the Predictor is adjusted to minimise the differences between the real world ship and the internal model (Predictor). The block diagram underlining this process can be seen in figure 4.5.

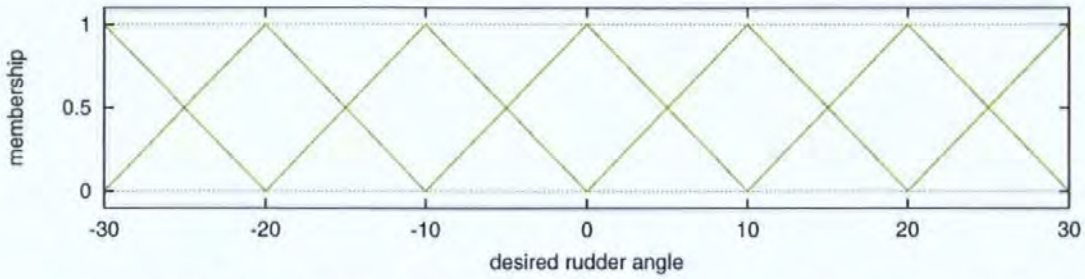


Figure 4.13. Fuzzy Output Window – equally placed sets (desired rudder)

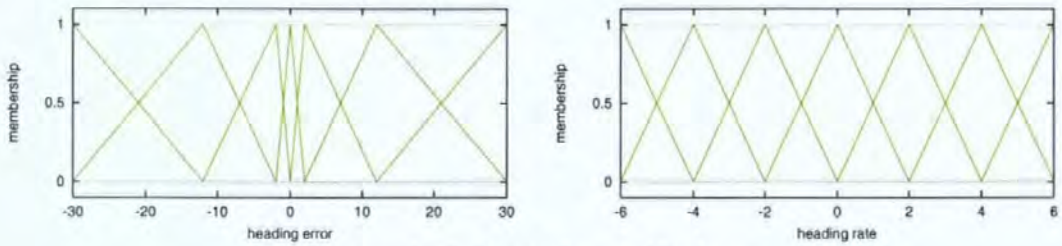


Figure 4.14. Fuzzy Input Windows

on exploring variants of defuzzification methods. The main problems of the roughly shaped control surface are caused by the size and shape of the fuzzy sets in the output window. In particular the area covered by the set is considered to have a detrimental influence. If the fuzzy sets are equal in size and equally placed over the universe of discourse, then the occurrence of a 10% error can probably be accepted. However, if the sets are different in size and/or shape, or the sets are not covering the universe with equal spacing, then the error increases and the control surface becomes undesirable.

Very broad sets (*see* figure 4.15) attract a very wide range in the universe of discourse, so if such a rule is active, it will dominate the final output because of its influence with its first moment of area ($A \cdot cg$). As a comparison, figure 4.13 shows a fuzzy output window with regularly shaped, equally placed fuzzy sets. Note that, the defuzzification method *centre of*

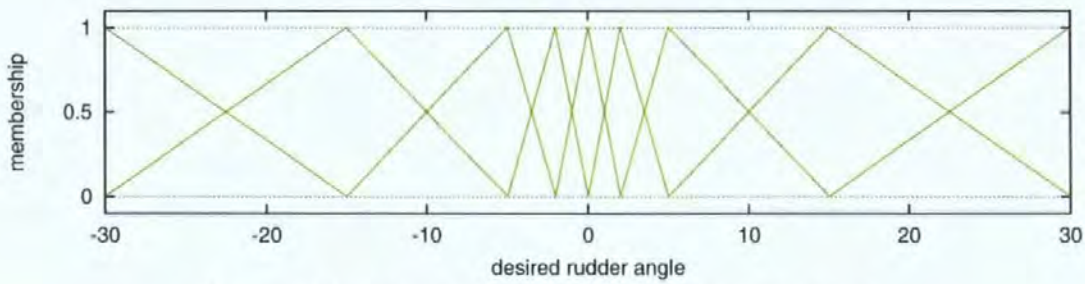


Figure 4.15. Fuzzy Output Window (desired rudder)

Table 4.1. Comparison of Defuzzification Methods

	rudder rate activity in $[°^2/s^2]$	rudder activity in $[°^2]$	heading ISE in $[°^2]$
standard method	17.333	4.614	7.164
new method	2.601	2.575	7.529
percentage drop	**-84.99%**	-44.19%	5.09%

difference between the two methods compared is the additional code to *normalise* the influence of the fuzzy set.

The standard (Centre of Area) and the new defuzzification methods are compared and numbers can be found in table 4.1. The percentages in table 4.1 are obtained using equation 4.11

$$\text{percentage drop } [\%] = -\frac{\text{standard method} - \text{new method}}{\text{standard method}} \times 100\% \tag{4.11}$$

The first four seconds of the test are omitted due to excessive noise when calculating the rate and the saturation of the desired rudder. The *rudder rate activity* is calculated as follows:

$$\text{rudder rate activity} = \frac{1}{T}\int_0^T \dot{\delta}^2 \mathrm{d}t \tag{4.12}$$

with

$$\dot{\delta} = \frac{\mathrm{d}\delta}{\mathrm{d}t} \approx \frac{\Delta\delta}{\Delta t} = \frac{\delta_{d(k)} - \delta_{d(k-1)}}{t_{(k)} - t_{(k-1)}} \tag{4.13}$$

as rudder rate approximation or discrete

$$\begin{aligned} \text{rudder rate activity} &= \frac{1}{T}\sum_k \dot{\delta}^2_{(k)} \cdot (t_{(k)} - t_{(k-1)}) \\ &= \frac{1}{T}\sum_k \frac{(\delta_{d(k)} - \delta_{d(k-1)})^2}{t_{(k)} - t_{(k-1)}} \end{aligned} \tag{4.14}$$

and the heading ISE is determined using

$$\text{heading ISE} = \frac{1}{T}\int_0^T {\psi_e}^2 \mathrm{d}t. \tag{4.15}$$

The heading error of the two methods is nearly identical, the new defuzzification method

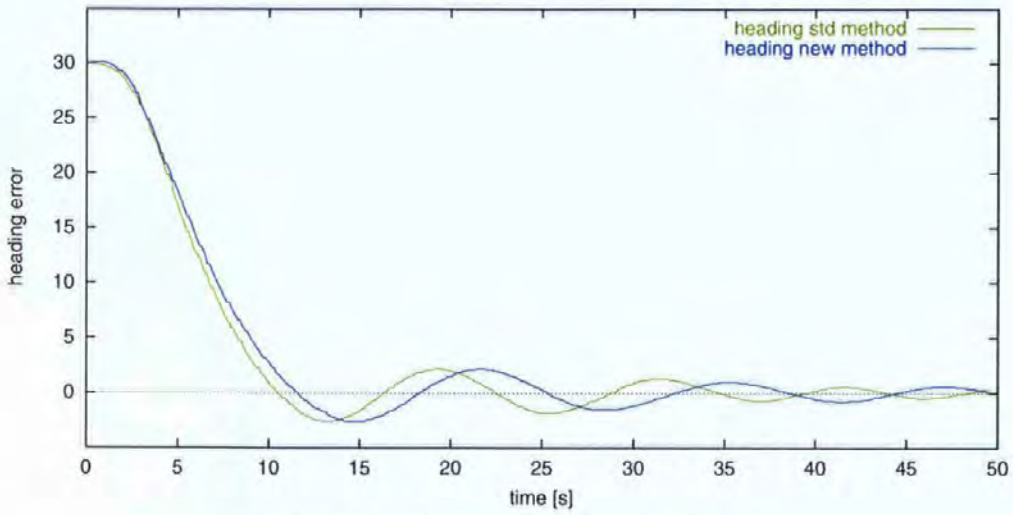


Figure 4.16. Heading Comparison

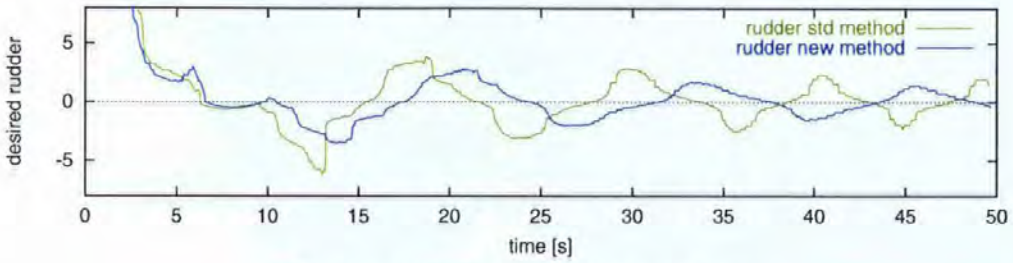


Figure 4.17. Rudder Comparison

shows only an non-significant deterioration (5%) here. The rudder activity itself was reduced to about one half. A comparison of the values for the rudder ISE (error between actual δ_a and desired rudder δ_d) shows no significant difference. A reduction of approximately 2% was found. The comparison of the heading errors, figure 4.16, shows a slight increase in ISE of the heading. This indicates that:

the defuzzification method does not have a big influence on the actual controller's performance with respect to heading.

With respect to the rudder rate activity, an almost 85% drop between the two methods compared could be achieved.

Considering the desired rudder graph (figure 4.17) it can be see that the standard method causes more rudder activity, hence bigger rudder deflections in certain conditions. At about 13s the desired rudder angle rises sharply. At about 24s a steep drop can be noticed using the standard defuzzification method. This can also be noticed in the rudder-rate plot 4.18. This new, improved defuzzification method, demonstrates a much smoother response, less rudder

Initial tests involving only heading, heading rate and desired rudder with respect to the input vector have proven the network to be too small. It was impossible for the network to build a working model, from this limited set of input/output data. The output vector in this case contained only one value, the heading acceleration. By using this model, the predictor led the control algorithm to unsatisfactory behaviour shown in an oscillating, undamped response.

More data was found to be relevant and therefore included in the input vector, namely roll and pitch information. For internal processing, those data were also included in the output vector and linked to the input vector when predictions are made more than one time constant ahead.

Different numbers of neurons in the hidden layers were also tested. Using an increased number of neurons in the second hidden layer did not improve the overall performance of the neural network model and the model therefore reverted to the size of the former structure.

The next section will explain the adaptive model, the *Predictor* in more detail.

4.2.2 Development of a Prediction Strategy Employing a Neural Network

A method has to be determined which is capable of teaching the network on-line. This can be achieved by learning measured data whilst a journey (simulated or real) takes place. In this way, the model can adapt itself to respond exactly as would the vessel when working in the same environment (mass loading, forward speed, disturbances, *etc*).

The neural network model will adapt itself if the ship characteristics change, as a result of this the model has an unique ability to represent the current state of the ship at all times. The training sets the network is exposed to during the training are important for the quality of the neural network model. To represent a particular state, all characteristic values have to be taken into account. On the input side of the predicting neural network (*see* figure 4.8) there are the forward speed, rudder angle and desired rudder angle, heading, heading rate, roll and pitch angles and the heading acceleration. These data are related to the new heading acceleration on the output side which when twice integrated gives the new heading angle.

During tests of the predictor it was observed that the predictions of the model improved

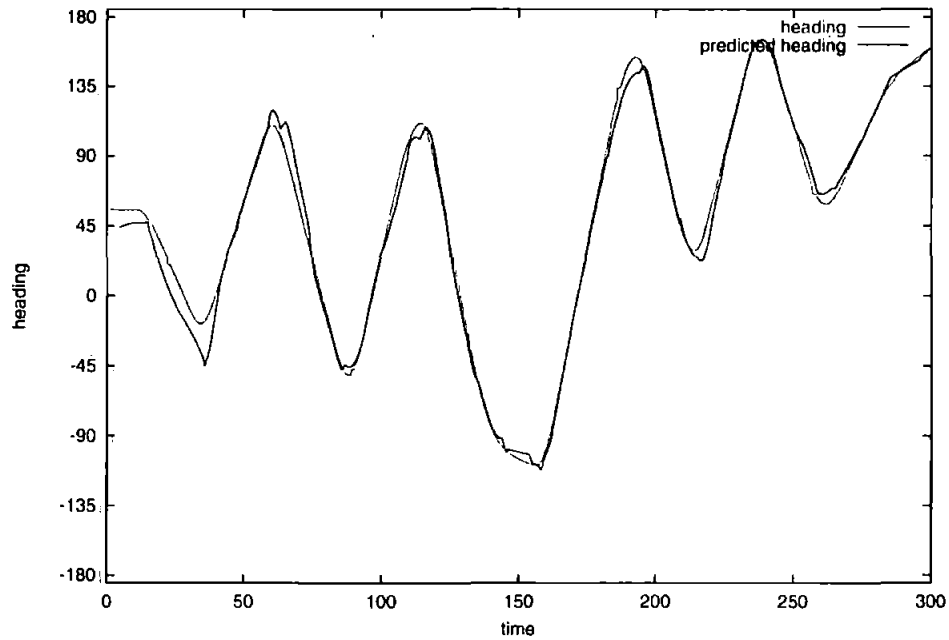


Figure 4.9. A Test Run of the Predictor Module (Heading)

when pitch and roll angles were considered as additional inputs. These values are not considered, however, in the process of adapting the controller but they are internally used by the predictor only. Analysing equations 4.6 the coupling between roll, pitch and yaw become apparent. By including those extra values in the state vector a better model was achieved.

The neural network will establish a relationship between the data on the input ports and the expected data on the output ports. The main outputs of the predictor is the new heading related values ($\ddot{\psi}_p$, $\dot{\psi}_p$, ψ_p). The change in these values depends largely on the applied rudder. The environmental forces such as waves, tide and wind will push the vessel off its desired course. The data representing the environment including the vessel are considered as inputs and the vessel's response (heading, roll, pitch) as outputs. When predicting a new heading, the current and past two states are required. This combined data is fed into the neural network which will return the new, expected, state of the vessel one time step ahead. If a prediction further in the future is required, then the prediction process is repeated with the most recently calculated figures fed back into the input vector as the most recent sample. For adapting the control parameters, only the predicted heading information is considered.

4.2.3 Evaluation of the Prediction Module

The predictor was tested in simulation. Figure 4.9 shows the behaviour of this module. A zig–zag manoeuvre was performed and the neural network was trained with measured/ simulated data. The predictor was then used to predict the ships heading, rudder, roll and pitch 2 time constants ahead (approximately 3s at full speed 1050 rpm, 21 knots). In the beginning there were rather large errors. With an increase in time and the number of learned conditions, this error was reduced. The Predictor does not need long term accuracy since it is only required to predict a short time period ahead (up to 3 time constants). Furthermore, a second order system reaches 95% of the final steady state output to a step input after 3 time constants. Predictions will be largely effected by inaccurate internal representation when travelling at full speed. So the circumstances presented are the 'worst case scenario'.

The predictor is unable to foresee changes in heading demand (operator driven desired course change) and this can be seen in the rather large differences between actual and predicted heading when the heading demand changes. After a small period of time, the actual response follows closely the predictions.

The neural network predicts the future actual heading
at sufficient accuracy to be suitable as a predictor.

Figure 4.9 also shows the close matching of the true and predicted headings when a constant rate of turn is reached. In essence, that is the expected predictor result.

4.3 The Integrated System

In this section a new kind of a fuzzy logic self-organising technique is introduced. The block diagram seen in figure 4.10 shows all the modules discussed earlier in this chapter. These modules, when combined, become the novel *Predictive Self-organising Fuzzy Logic Controller* (PSoFLC).

The control system consists of 3 main parts:vspace-5mm

- the plant exposed to disturbances ⇒ requirement for control,
- the controller and

The control output is only passed to the process (simulated or real world ship) if the predicted performance of the process following the control action shows a good, desired performance. If the resulting state is poor (the PI indicates that low performance state), the rulebase is adjusted to meet the requirements.

The left-hand side of figure 4.10 represents the controller with all its components. The working principle of the controller (providing the desired rudder angle δ_d and adapt the rulebase if necessary) is summarised as follows:

1. An error signal ψ_e, $\dot{\psi}$ is determined from the *desired course* ψ_d and *actual course* ψ_a.
2. This signal is passed to a fuzzy logic controller and a *desired rudder angle* δ_d is generated.
3. The desired rudder angle δ_d is fed into the predictor to give a future state of the vessel.
4. **IF** the rudder change results in an improved state (*ie* the *heading error* ψ_e is reduced), **THEN** the *desired rudder angle* δ_d is applied to the ship's real rudder.
5. **ELSE**, the rules involved in the calculation of this *poor performance* rudder angle δ_d are modified in order to improve future performance.

The characteristics of the vessel is learned on-line while the vessel is in operation. This insures an always up-to-date model (Predictor) representing the current control environment. A flow chart to visualise the above is given in figure 4.4.

To evaluate the controller's performance, a manoeuvre has been executed. The manoeuvre set up is a square 'figure of 8' in the following form. Commencing with a course of 90 degrees (East), followed by 90 degree turns to the North, West, South, East, South, West, North and finishing off in eastern direction. All the course plots can be seen in chapter 5.

In figures 5.2 and 5.3 (*see* chapter 5), the course of the vessel and the heading and rudder changes are analysed. The manoeuvre is performed three times to demonstrate the learning capabilities of the controller. The learning effect can clearly be seen in the third manoeuvre (third 'figure of 8') where the course follows more closely the desired shape than previous ones. The controller started off with an empty rulebase, which meant that, under all circumstances the desired rudder angle remains at zero.

The rudder will only start to move to one side when the measured state (orientation, position) indicates a low performance level of the controller and the adaptation process causes some alterations in the rulebase. This, of course, can take some time when the SoFLC is used, since the SoFLC 'waits' a period of time before it measures the state. Only when the performance has already deteriorated does the SoFLC adjust control parameters and subsequently tries to improve the state.

Both sampling time, and the time which must pass to allow the control action to take effect (delay in reward, DEL), play a key role in the time delay and low response time of this kind of controller. *NB*: The response time here refers to the time taken to change the knowledge base and not the controller's reaction time.

4.4 An Original Defuzzification Method Using a Normalisation Technique

During the first development of the FLC using a geometrical approach, some questions remained unanswered. For instance, the control surface was not smooth and it contained plateaus and cliffs (figure 4.11). This effect was particularly bad when irregularly shaped and non-uniformly distributed fuzzy sets in the input windows were used.

Work undertaken during this research programme has identified that:

Current defuzzification methods cause a rough and undesirable control behaviour.

This section highlights a new idea and its implementation to overcome these problems.

To improve the appearance of the control surface one has to smooth out the cliffs and plateaus.

Initial experiments by the author, with fuzzy logic, have demonstrated that: the control surface is highly non-linear when irregular fuzzy sets are used.Not only do irregularly shaped sets show such behaviour, regularly shaped triangular sets, non-equally placed in the universe of discourse, produce a very similar response. The reason for such 'abnormal' behaviour lays in the mathematics of the employed defuzzification method. A solution developed by the author to overcome the problem can be found later in section 4.4. To aid comparison, the

surface shown in figure 4.11 has plateaus and cliffs which are produced using the '*centre of area* defuzzification method'.

Each rudder movement slows down the vessel due to the increased drag effects. So, rudder changes have to be minimised in order to save time, fuel, money and downtrack journey distance. Nevertheless, the controller should react precisely in order to minimise heading error. The application of an averaging filter is possible but will result in a more sluggish behaviour.

Eleven years of experience in autopilot development has indicated to the engineers at Cetrek Ltd. [119] that a response resulting from a control surface as shown in figure 4.11 will cause extensive rudder wear as well as uncomfortable rides (due to the sporadic and harsh rudder movements). The author observed how small changes on the input side (small variation in heading and turn rate) induced very rough behaviour and response by the rudder. According to Cetrek Ltd., fuel is wasted and bearings wear out much more quickly if a controller shows such a response surface. A far smoother response (desired rate of rudder movement) is required to secure a comfortable ride as well as minimise wear and tear.

A better defuzzification method is required. This section will introduce a new defuzzification method for fuzzy logic controllers.

Given the performance problems (rough and sporadic changes of the output value *see also* [19], section 10.6) associated with the standard fuzzy logic control, it was found to be necessary to generate an enhanced version to overcome these performance problems.

Figure 4.12 shows the resulting control surface which is much smoother, without the steep cliffs and plateaus. Such a control behaviour is much more desirable than the one seen in figure 4.11 as previously discussed. Analysis of the defuzzification method has identified that the shape of the control surface is dependent upon a) the position and b) the shape of the used fuzzy sets as well as c) upon the defuzzification method used.

The non–desirable surface profile is inherent in the fuzzy logic defuzzification method employed. The maximum deviation (error) is less than 10% if regularly-shaped, equally-placed sets (as in figure 4.13) are used [88][1]. To improve the controller, more research has been undertaken by the author in the field of classical fuzzy logic, with particular emphasis

[1] *See* appendix F for translation of the relevant pages.

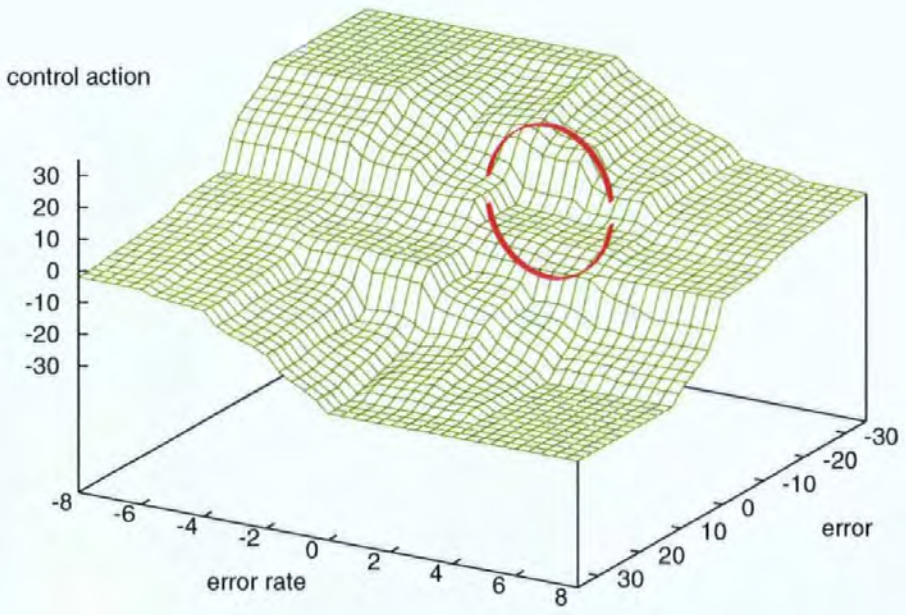


Figure 4.11. Control Surface using Conventional Fuzzy Logic and Irregularly Shaped Sets

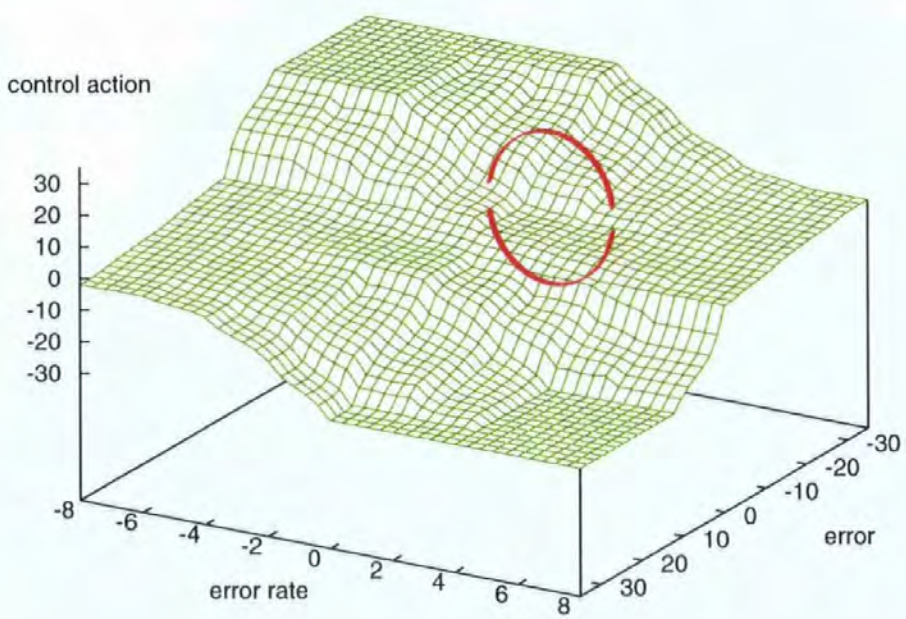


Figure 4.12. Improved Control Surface

area has been employed. The equation for the overall centre of area is:

$$\text{overall Centre of Area} = \frac{\sum_i A_i cg_i}{\sum_i A_i} \tag{4.9}$$

The problem of rough behaviour lies within equation 4.9; a wide fuzzy set covers a wide range within the universe of discourse and is therefore very 'fuzzy' and should be less important than smaller, more precise, sets. The first moment of area does exactly the opposite, the wider a set (larger the area) the more influence this set has on the final output, the more weight it brings in.

One possible way to overcome this problem is to use irregularly shaped fuzzy sets. When the degree of membership changes, the position of the centre of gravity changes, so a smoother transition between sets could be achieved. 'Leaning' the set over to one side, results in a more gradual change when the set 'is left'. However, this proves difficult to realise on both flanks.

The solution introduced is quite simple. To make the sets equally important, the weight of the sets is normalised. The weight indicates the influence of the set relative to the overall output value. The area is defined and cannot be changed, so a new factor is required. To equalise the weight of the set (all sets contribute the same amount towards the final output), each set receives a density (ρ). The weight is calculated as: $w = A \cdot \rho$. The density of a set is simply the reciprocal of the area under the full set. The *active weight* (dimensionless) becomes:

$$\text{active weight} = \frac{\text{active area}}{\text{area of full set}} \tag{4.10}$$

This simple algorithm (equation 4.10) normalises the the influence of the set to the overall output. To defuzzify, the active (and normalised) weight is used and the active area is not considered. This new technique clearly demonstrates (*see* figure 4.17) the improved behaviour due to smoothed, yet still accurate response. The smoothed control action results in less harsh rudder action. This will reduce rudder wear and improve the energy efficiency of the vessel when the fuzzy logic is applied to an autopilot.

To obtain some analytical data, a 90° step change is performed with exactly the same settings. The fuzzy input windows (figure 4.14) and rulebases are identical. The only key

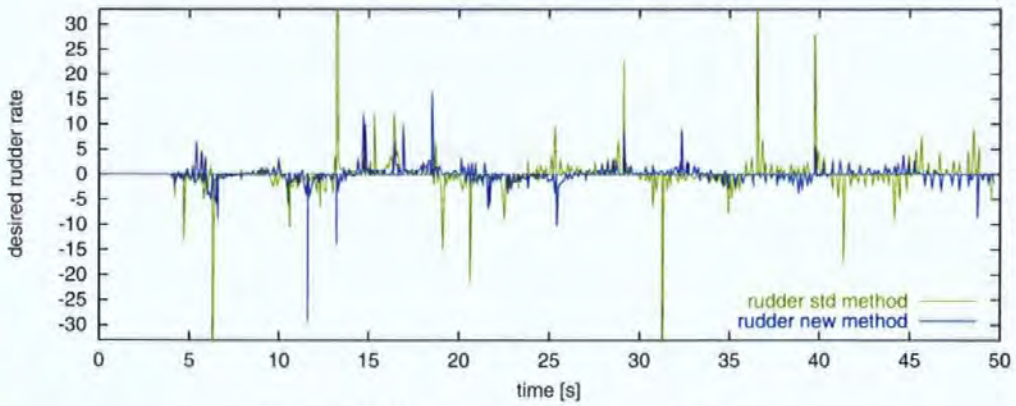


Figure 4.18. Rudder Rate Comparison

deflection and flatter slopes (smaller rate of change) then the standard method.

4.5 Summary

The development and test of the *Predictor* module was investigated in this chapter. The requirements from a control point of view were clarified as well as test results were presented and discussed. The Predictor is an essential part of the novel control strategy explained in this thesis. It allows the validation of control actions *before* they are applied to the real craft and helps to reduce the execution of unnecessary or performance-reducing actions.

Conclusions may be drawn from the predicted behaviour about the current performance of the controller and necessary adjustments can be executed *before* a performance deterioration take place.

Due to some unexpected behaviour and undesirable performance characteristics of the developed fuzzy logic controller, more effort was taken to overcome the very harsh response. In certain areas of the control surface very steep cliffs were found which delivered an undesirable response, putting unnecessary stress on the steering gear and producing more drag. These shortcomings spurred the development and testing of a new defuzzification method. This new method allows a wider application of rules in the output window; especially when they are unevenly spread over the universe of discourse and the fuzzy sets themselves are irregularly shaped. The new resulting control surface shows a much smoother relief compared to the standard defuzzification methods.

The fuzzy logic controller embedded in the predictive self-organising fuzzy logic con-

troller uses movable fuzzy singletons. This allowed an easy implementation of the adaptive algorithm in form of a computer program. It was shown that there is only a small variation between the new defuzzification method and defuzzification of fuzzy singletons.

This chapter considered the *Self-organising Fuzzy Logic Controller*. Covering the structure of the rulebase and the performance index, the self-organising technique is explained. The update algorithm utilises the result of the performance index to adjust parameters of the fuzzy logic rulebase to counter act performance deterioration.

Expanding on the theory of SoFLC, the Predictor is added to the controller to form the PSoFLC. The system is now able to evaluate (and apply or omit accordingly) a control action before it will be executed. This results in faster learning of desired rudder actions when compared to standard SoFLC.

Since all control actions are evaluated by the predictor and performance index before application it is believed to obtains a stable system. If the predictive module represents the plant under control and hence 'poor' performance is detected, a stable controller can be guaranteed.

Chapter 5

Simulation Test Results

To demonstrate the controller's performance, a manoeuvre has been set up to repeat the same conditions for the different autopilots used. The SoFLC is compared with the novel *Predictive Self-organising Fuzzy Logic Controller* (PSoFLC). This comparison demonstrates the ability of the PSoFLC to adapt quickly and therefore to give a better performance in a much shorter time.

5.1 Tests without Disturbances

All the tests in this section have been undertaken without disturbance effects being considered. Two different tests have been set up. Firstly a 'figure of 8' test which provides a visual reference of the performances of the controllers in question and is included to demonstrate in a qualitative way how the vessel responds with various controller options. The second test is a classic step response test in which the step is a change in course demand of $\pm 20°$.

5.1.1 Course Following Test – 'Figure of 8'

These simulations have been executed in ideal conditions – calm water, no disturbance. The autopilot is required to do a square 'figure of 8' (figure 5.1) at a set forward speed. The following table 5.1 contains the desired course information and the times for which the course should be kept.

The learning of the SoFLC is slow, therefore the vessel does not reach the desired course

Table 5.1. Figure of 8 – Course Definition

course	time
90°	40s
0°	40s
-90°	40s
180°	40s
90°	40s
180°	40s
-90°	40s
0°	40s
90°	40s

before the next course change happens (40s) which results in an steadily increasing rudder demand. When the manoeuvre is repeated for the first time (2nd 'figure of 8'), the rulebase contains rudder angles and therefore the rudder is deflected more than in the first 'figure of 8' but it is still increasing. Comparing this with the response of the PSoFLC, the initial rudder response following a 90° course change builds up in a similar manner but this controller adjusts the rulebase so quickly, that even on the very first course change a control action can be seen rather than the current maximum *learnt* value. The rudder is driven back due to the fact, that the controller comes close to the desired course. The peaks still show the increasing values for the rudder angles in the rulebase until they settle.

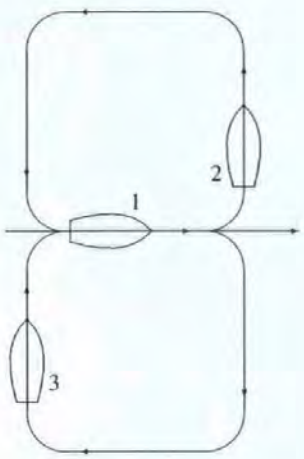


Figure 5.1. Figure of 8 – Course Plot

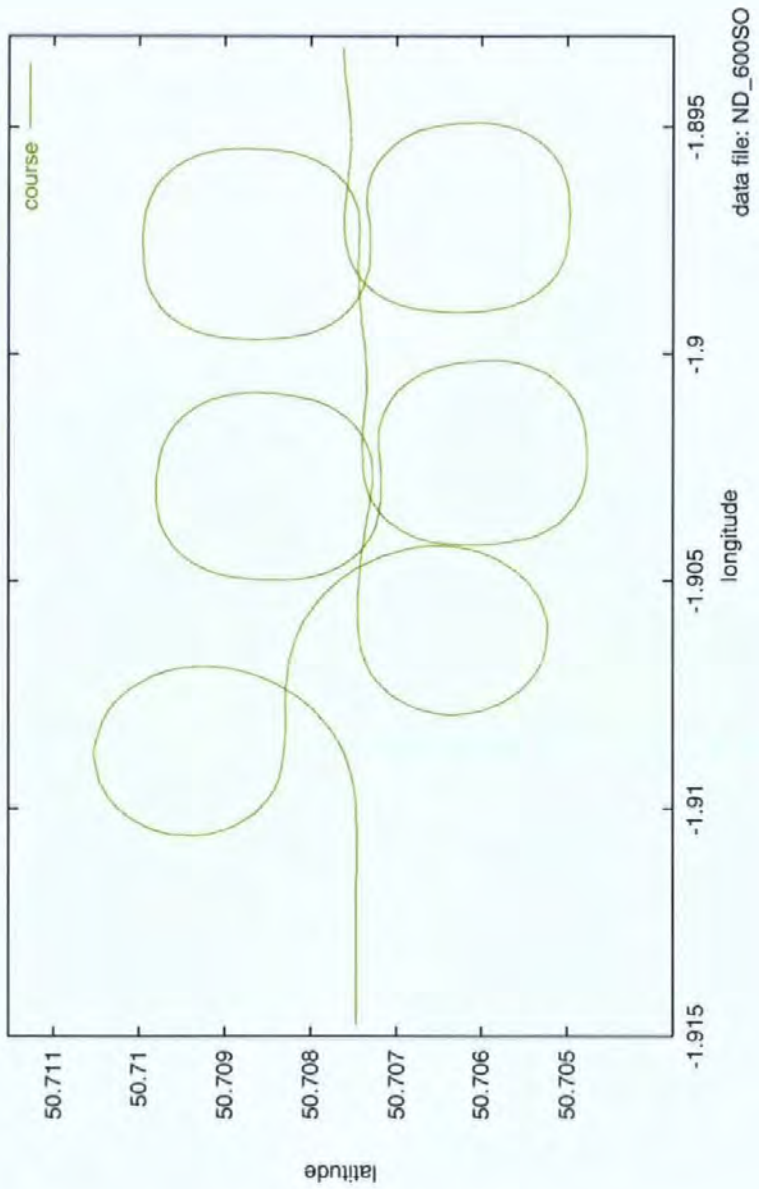


Figure 5.2. Course Plot – SoFLC, no disturbances

Figure 5.2 shows the same 'figure of 8' course (*see* table 5.1) repeated three times using the SoFLC. The first 360s of the 'figure of 8' test show one feature of the learning. The rudder deflection has to be learnt. This is an iterative process, starting with very small deflections and gradually increasing the rudder movement until the vessel responses as defined in the PI. It can be seen that the first 'figure of 8' is almost unrecognisable. But once the controller is taught, it repeats the manoeuvre staying closer to the desired course.

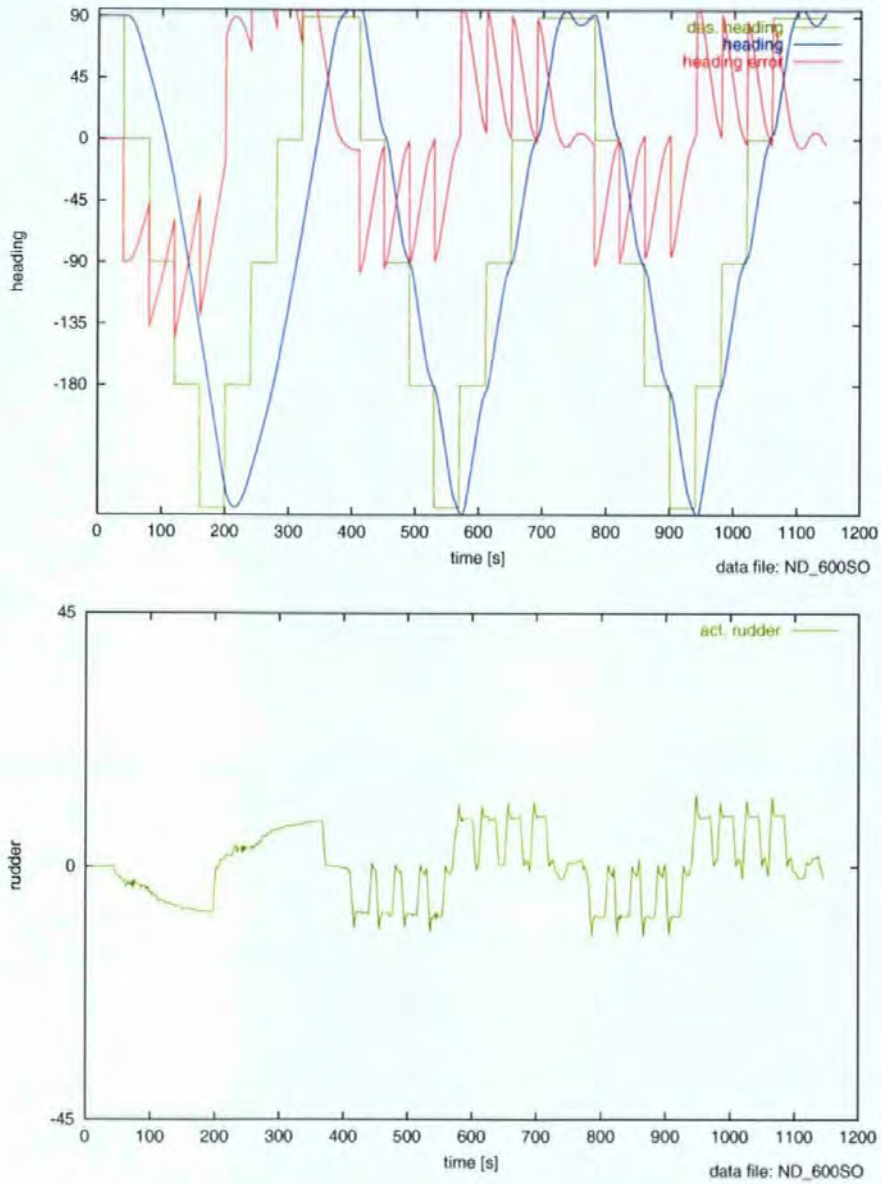


Figure 5.3. Rudder and Heading Plot – SoFLC, no disturbances

Figure 5.3 represents the heading and rudder responses whilst undertaking the 'figure of 8' manoeuvre. The bottom graph shows the steadily increasing rudder particularly well. The first 'figure of 8' (360s) can clearly be seen as the learning phase. When the manoeuvre is repeated (second and third 'figure of 8') the controller response is much improved, resulting in a more recognisable 'figure of 8'. From the rudder activity itself, it can be assumed that manoeuvre two and three look very similar due to the similarity in rudder activity. Looking at figure 5.2 and the top graph of figure 5.3, this can be confirmed. Inherently, the rudder response does not return to zero between early turns as the turn in incomplete when the next turn begins.

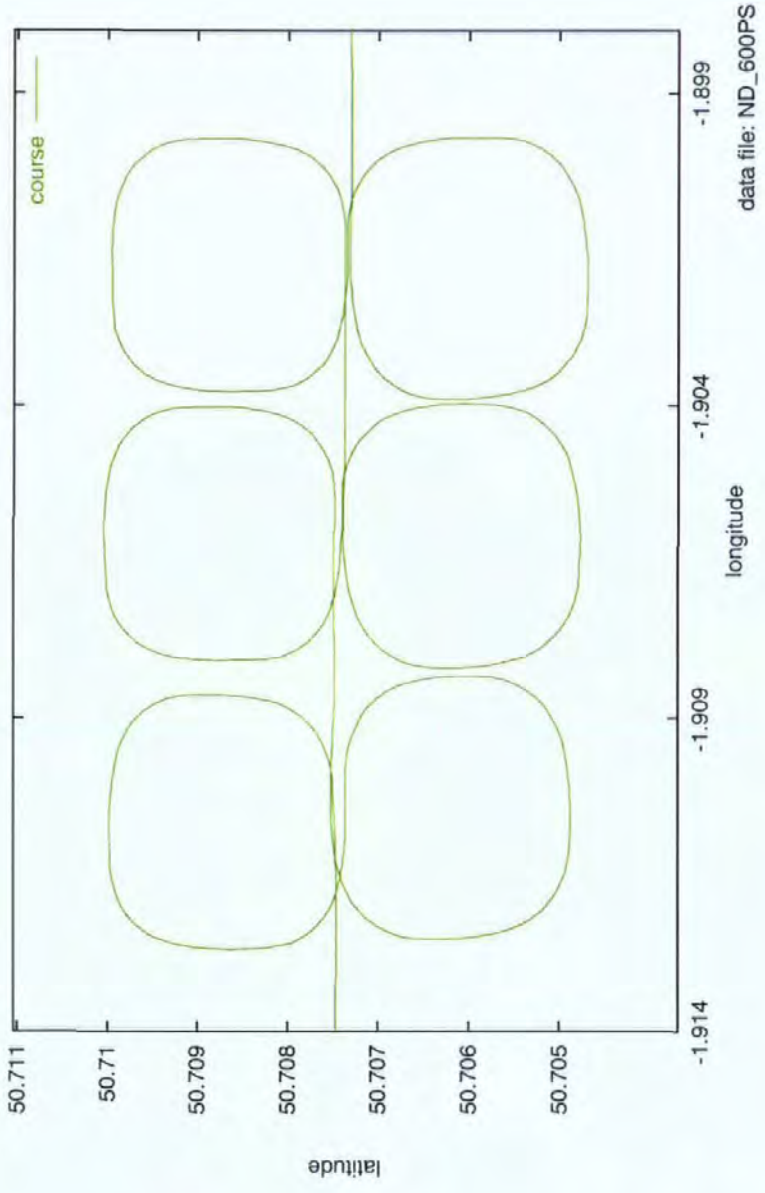


Figure 5.4. Course Plot – PSoFLC, no disturbances

The course plot of the same manoeuvre but utilising the Predictive-SoFLC can be seen in figure 5.4. Here even the first 'figure of 8' can clearly be identified as such. The rulebase adaptation is much quicker, giving a better control right from the start. A constant improvement can be noticed too. The last 'figure of 8' is more closed than the two before and the turns become more symmetrical. The overall visual performance appears to be much better then the SoFLC throughout the test.

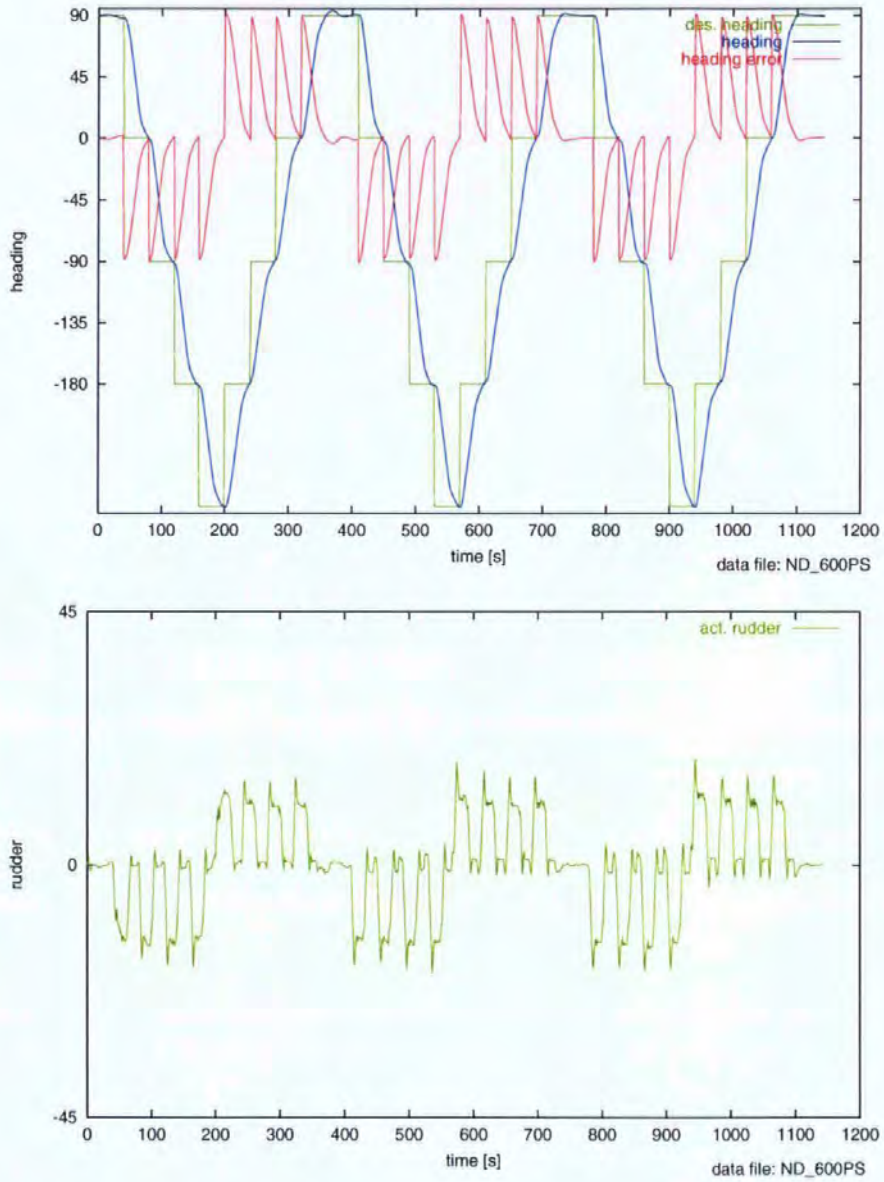


Figure 5.5. Rudder and Heading Plot – PSoFLC, no disturbances

Considering the top graph in figure 5.5 a horizontal line (zero error) can be noticed at the end of each but the first four-turn block. The rudder activity (bottom graph) reflects this too. The rudder activity becomes smaller and smaller when the three 'outrun' periods are considered. The learning phase can be noticed here as well. The first peak in the rudder plot has a gently increasing ramp in the rudder demand. This shows the build-up of the rulebase. Comparing this with the rudder activity of figure 5.3 it can clearly be seen, that the PSoFLC learns more quickly, building the rulebase required for the 90° turn almost within the time required for the first turn, hence the complete reduction of the error to 0°. At the second turn a jump in rudder demand is noticed as soon as the desired course changes.

5.1.2 Discussion of the 'Figure of 8' Manoeuvres

The graphs (course plots) in figures 5.2 and 5.4 show the course travelled for the different controllers.

The 'figure of 8' manoeuvres where undertaken to provide a visual reference of the capabilities of the various controllers. This emphasised the learning capability of the SoFLC and PSoFLC, and the differences between them. It was clearly evident that the PSoFLC adapted more rapidly than the SoFLC, which was demonstrated by the improved (more recognisable) 'figure of 8' in figure 5.4.

5.1.3 Step Response $\pm 20°$ Test

In figures 5.6 – 5.17 the step response of the vessel plus controller can be observed. Three controllers, namely the PD, SoFLC, PSoFLC, are compared. The manoeuvre has been repeated for three different forward speeds to demonstrate robustness. The plots start off with the heading and rudder data for 16.7 $km \cdot h^{-1}$ (9 knots, 450 rpm), in the order PD, SoFLC, and PSoFLC. The second series shows the same manoeuvre for a forward speed of 22.2 $km \cdot h^{-1}$ (12 knots, 600 rpm) while the third series shows the response at 38.9 $km \cdot h^{-1}$ (21 knots, 1050 rpm). The results are summarised in table 5.6 and table 5.7 later in section 5.1.5.

5.1.4 Graphs

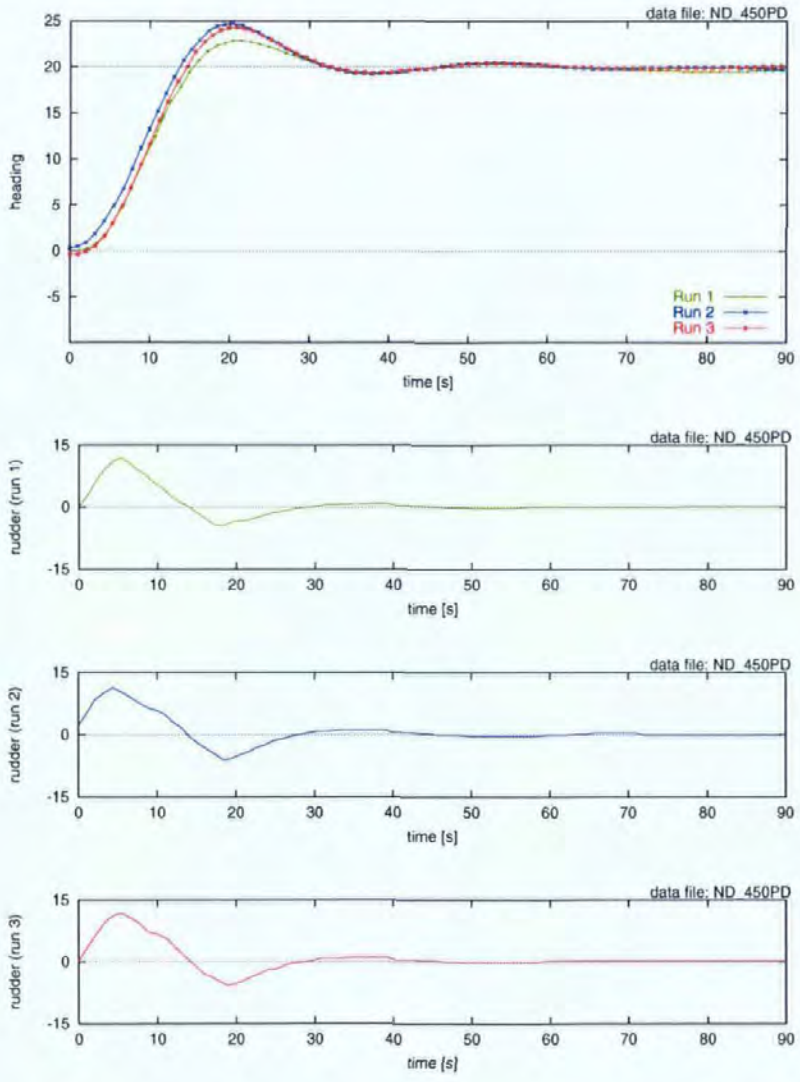


Figure 5.6. Step Response Test $\pm 20°$ at 450 rpm (PD)

Figure 5.6 shows the heading and rudder angles for three step changes of 90s duration performed by a PD controller. The test is performed with the engine running at 450 rpm propelling the vessel to 9 knots (16.7 km $\cdot$ h^{-1}). The test was a continuous process and the small differences in the responses were caused by the minor variations in the starting points for run 1,2 and 3. This is shown by the variations in the graphs. The sequential nature of the runs caused this variation. A course change at the end on a run results in a slightly different starting point for the following step change.

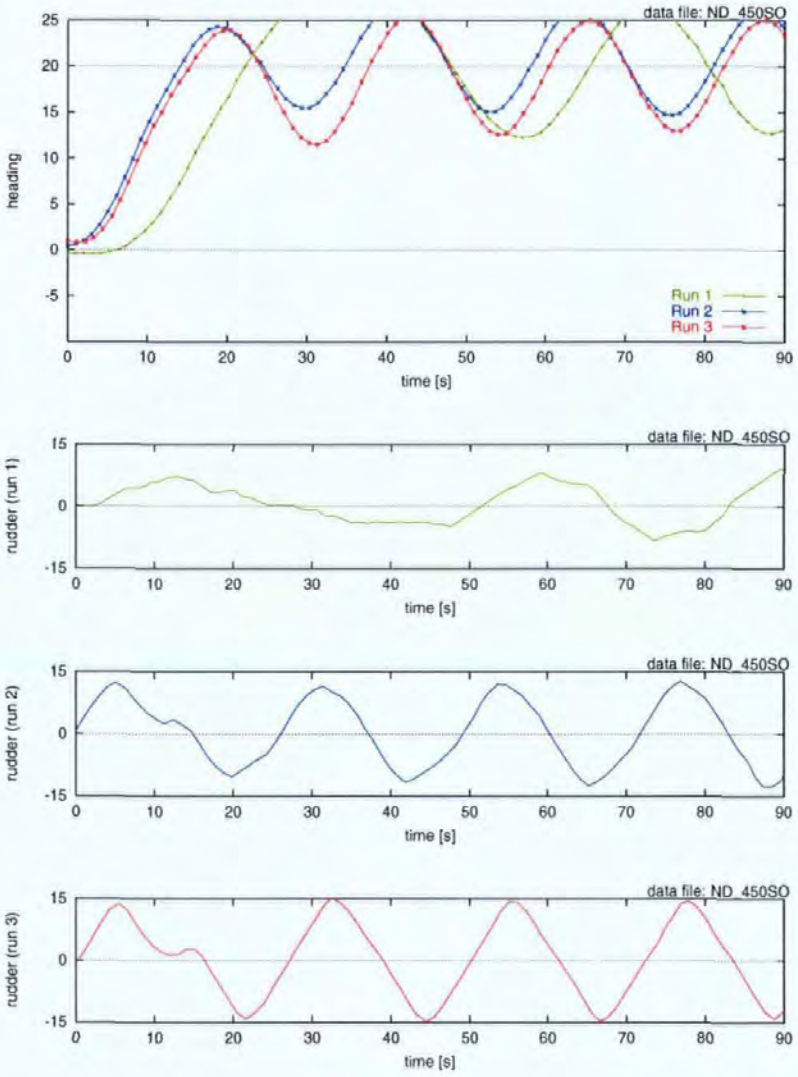


Figure 5.7. Step Response Test $\pm 20^{\circ}$ at 450 rpm (SoFLC)

The same $\pm 20^{\circ}$ step change was performed by the SoFLC. The simulated results are visualised in figure 5.7. The controller clearly has difficulties keeping the course under control. The first step change (green) shows a steadily increasing rudder demand. The second and third run use almost the same rudder demand to compensate. However, the response is not or only slightly damped.

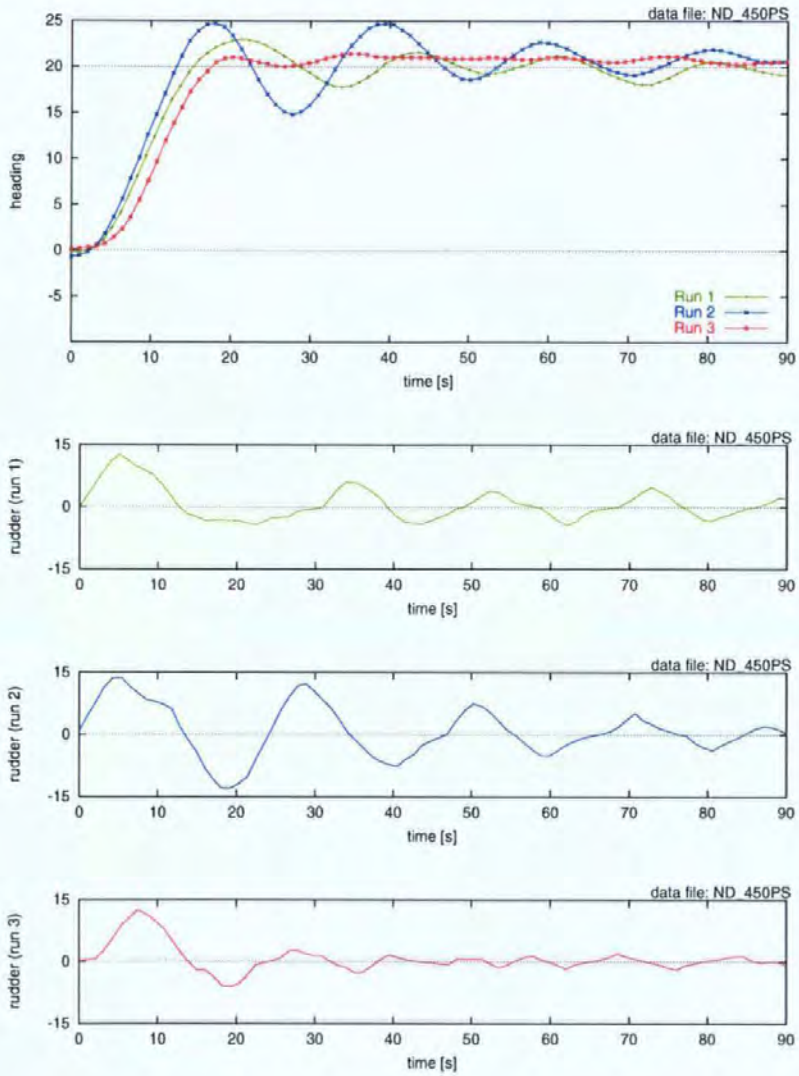


Figure 5.8. Step Response Test $\pm 20^{\circ}$ at 450 rpm (PSoFLC)

Figure 5.8, the same $\pm 20^{\circ}$ at 450 rpm is performed by the PSoFLC. There is a noticeable improvement over the three runs noticeable, indicated by the diminishing rudder demand and improved heading performance.

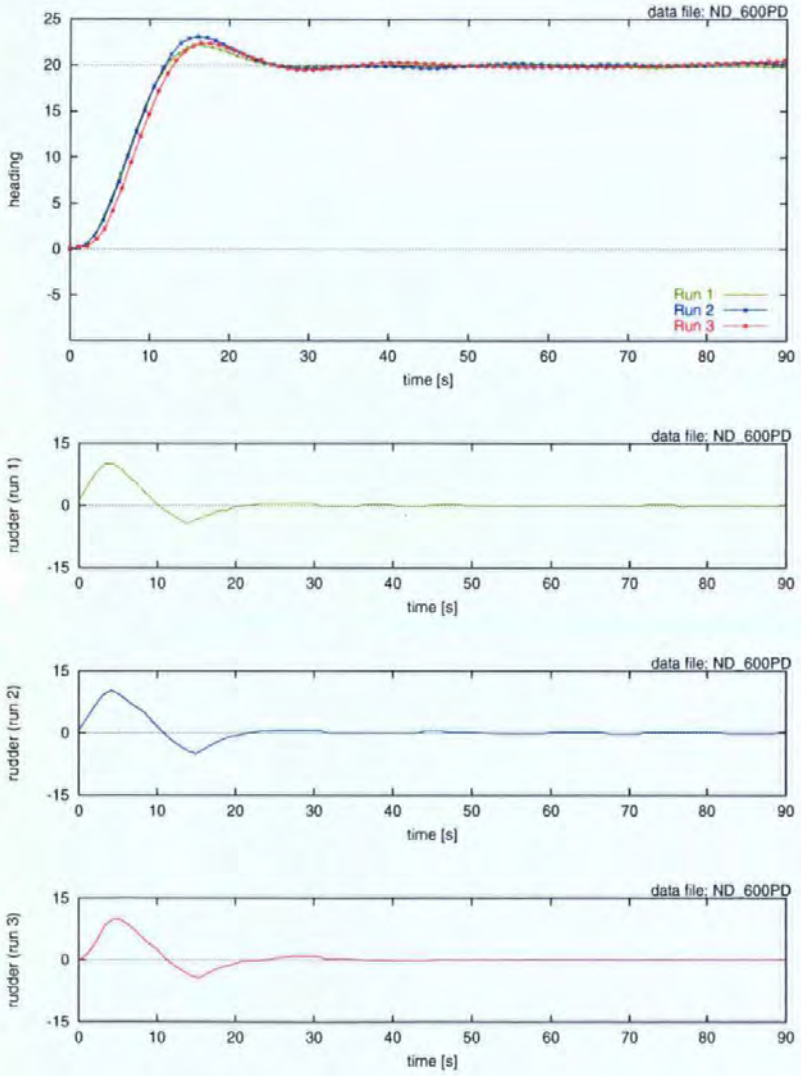


Figure 5.9. Step Response Test $\pm 20°$ at 600 rpm (PD)

In figure 5.9, the PD controller is utilised acting to a $20°$ step change at 600 rpm which results in a forward speed of 12 knots ($22.2\ \mathrm{km \cdot h^{-1}}$) through the water. The response shows a clear overshoot before the response settles. A slight second overshoot can be noticed.

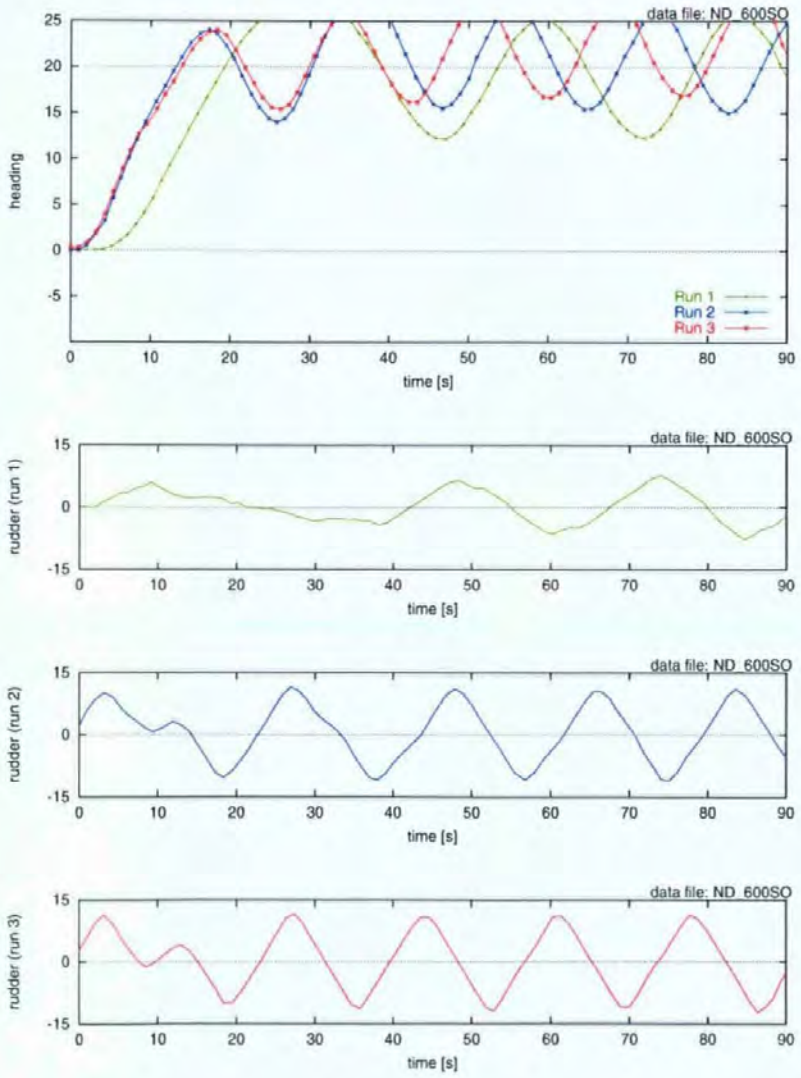


Figure 5.10. Step Response Test $\pm 20^{\circ}$ at 600 rpm (SoFLC)

Figure 5.10, the SoFLC at the speed of 12 knots shows a similar response to the response seen at the lower speed. During first run, the rudder demand slowly increases during the first run, and the response is very oscillatory during the second and third runs.

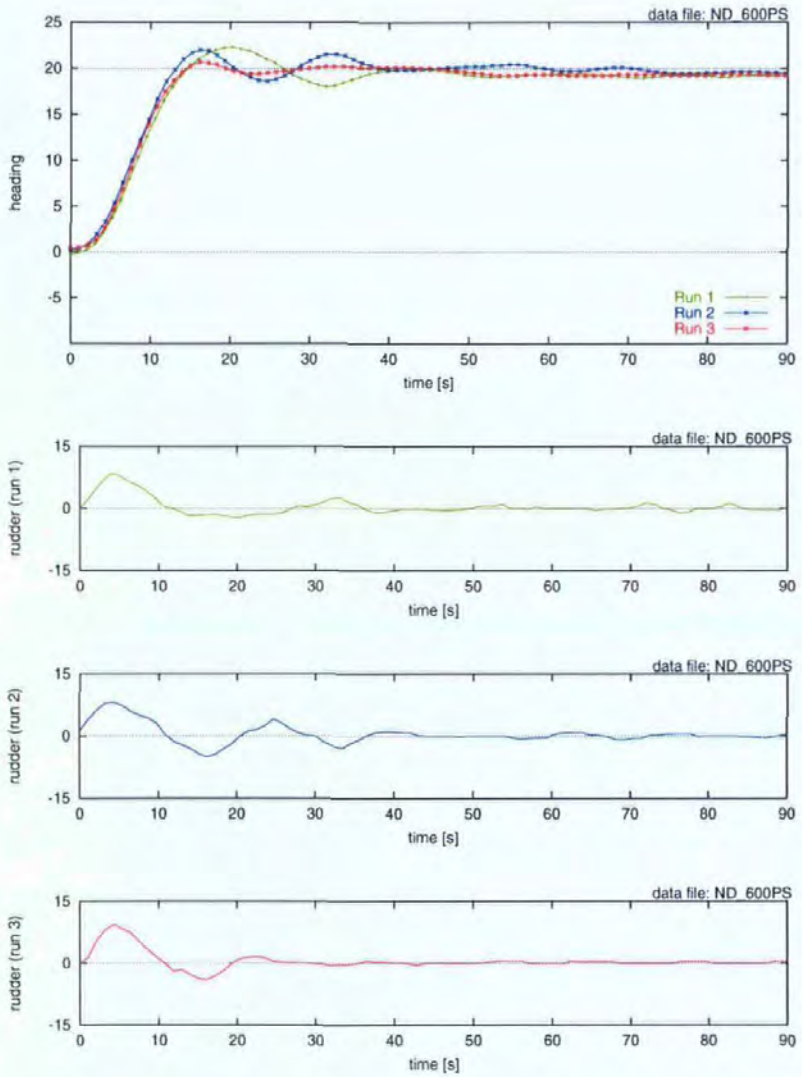


Figure 5.11. Step Response Test $\pm 20^{\circ}$ at *600 rpm* (PSoFLC)

The response of the vessel to a 20° step change and a forward speed of 12 knots is displayed in figure 5.11. The heading response shows an improvement over all three runs with very little rudder activity and oscillatory behaviour.

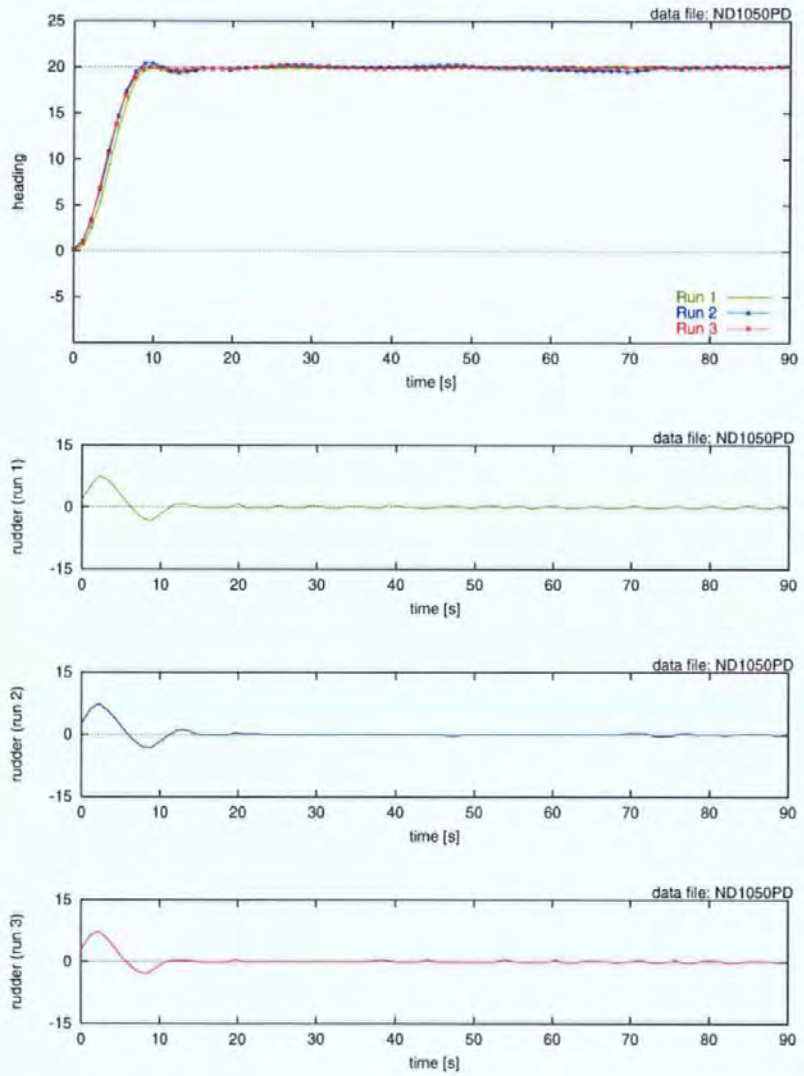


Figure 5.12. Step Response Test $\pm 20°$ at 1050 rpm (PD)

Figure 5.12 visualises the response of the vessel to a step change in heading demand of 20°. The vessel is now travelling at full speed of 21 knots (38.9 $\mathrm{km \cdot h^{-1}}$). This is the tuning speed of the PD controller, its response is displayed in these graphs. The graph shows two very small swings before settling. However, the setting value is not reached, so there is no overshoot on run 3. The minute differences in the runs are caused by the slight differences in the starting conditions. The steps were performed in sequence and a course error at the end of one run gives a different starting point for the following step change.

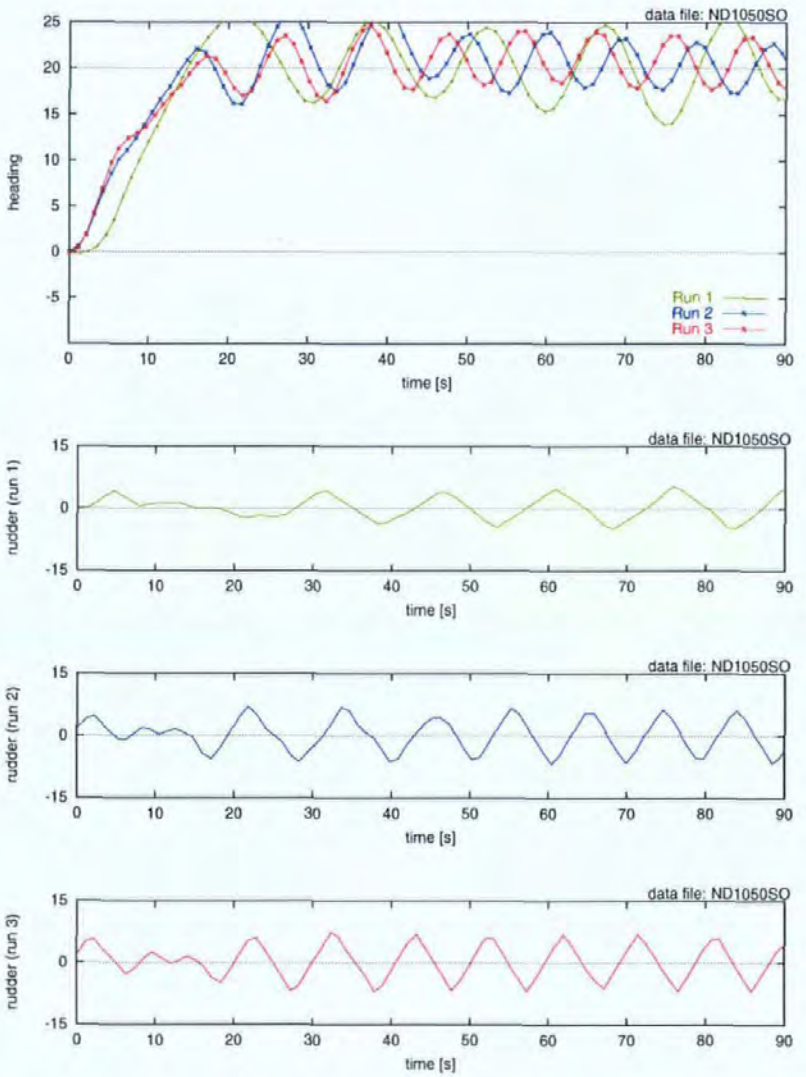


Figure 5.13. Step Response Test $\pm 20^{\circ}$ at 1050 rpm(SoFLC)

Figure 5.13, the SoFLC is controlling the vessel. As seen before, the rudder demand increases during the first run. The heading and corresponding rudder responses show a very oscillatory behaviour in the second and third runs. Neither the heading nor the rudder peaks increase over time, but do not diminish either.

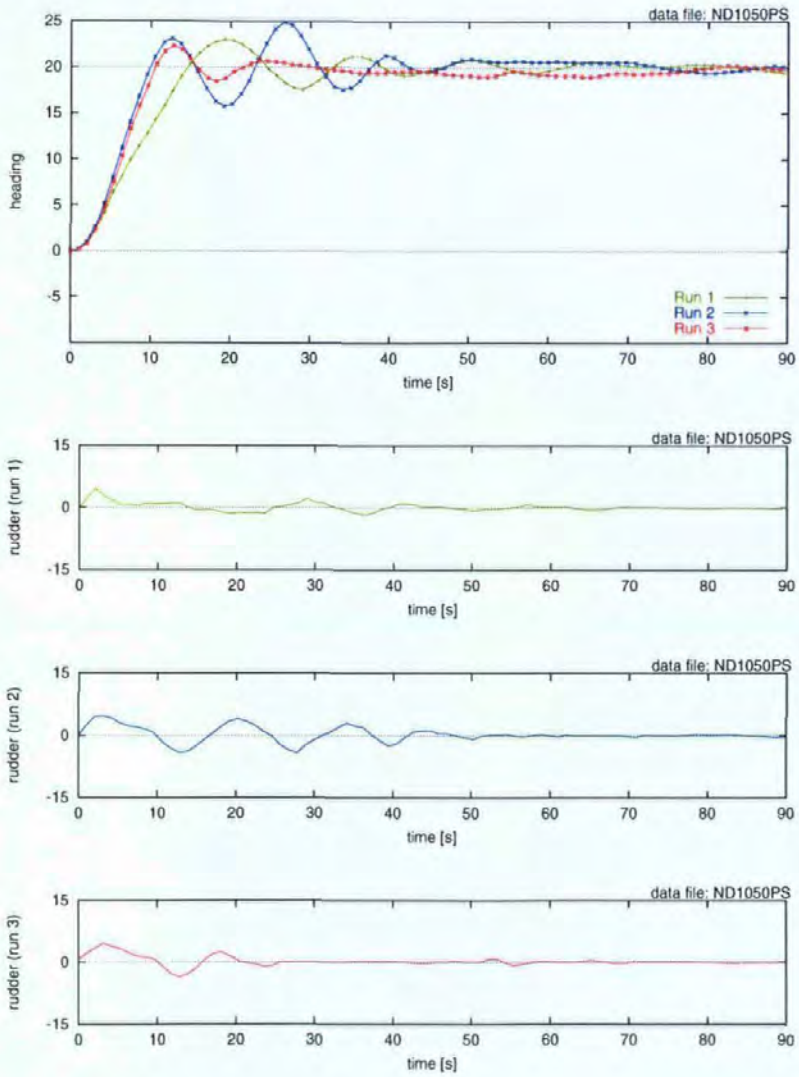


Figure 5.14. Step Response Test $\pm 20°$ at *1050 rpm* (PSoFLC)

Figure 5.14 shows the response of the vessel when under control of the Predictive Self-organising Fuzzy Logic Controller. The heading performance improves with each run. The third run shows two bigger overshoots and one smaller 'swing'.

5.1.5 Comparison of the Step Change Test Results

The step change tests were undertaken without disturbances present for a variety of forward speeds, *ie* 450 rpm, 600 rpm and 1050 rpm. Each controller was subjected to the same test conditions and course pattern. A comparison can therefore be undertaken to quantify the differences. Detailed analytical results of the comparisons are given below.

Step Response Test at 450 rpm – 9 knots (Figures 5.6, 5.7, 5.8 and 5.15)

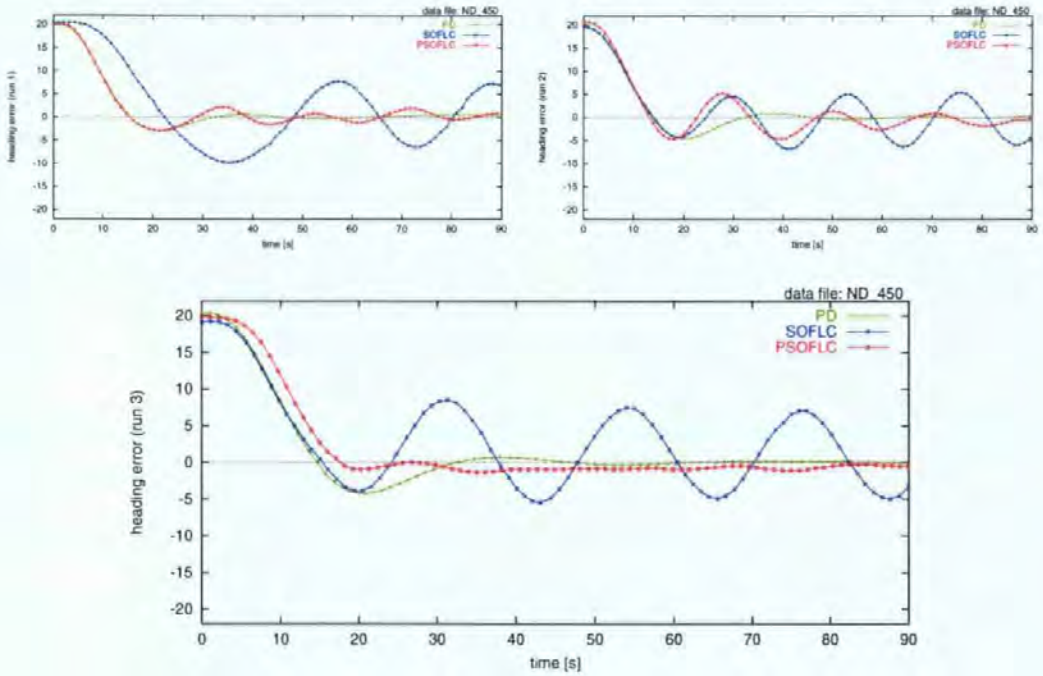


Figure 5.15. Step Response Test $\pm 20°$ at 450 rpm errors

At urban slow speed (450 rpm = 9 knots) the PSoFLC shows very little overshoot. However, the rise time is the longest of all three controllers (11.9s). The rudder activity of the PD and the PSoFLC are in the same range, 11.2 $^{\circ 2}$ and 11.0 $^{\circ 2}$ respectively. The rudder activity of the SoFLC is far higher, 81.2 $^{\circ 2}$. This increase in activity is expected when the heading errors are examined. The SoFLC ends up in a very oscillatory behaviour, driving the rudder between $\pm 15°$ and the course responds with a similar behaviour, oscillating between $\pm 5°$.

Considering the heading error plot, figure 5.15, the PSoFLC performance is let down by its course keeping accuracy. The course change of the third run (first 30s) shows that the PSoFLC achieves a much reduced overshoot when compared to both the PD controller and SoFLC. The PSoFLC indicates the best performance with respect to overshoot. At 9 knots

Table 5.2. Controller Comparison, 450 rpm

criteria	PD	SoFLC	PSoFLC	% drop PSoFLC v PD	% drop PSoFLC v SoFLC
ISE 0–30s [$^{\circ 2}$]	109.70	105.12	123.90	12.95%	17.87%
ISE [$^{\circ 2}$]	36.62	49.76	41.82	14.21%	-15.95%

criteria	PD	SoFLC	PSoFLC
rudder activity[$^{\circ 2}$]	11.20	81.19	10.99
rise time to 60% steady state	11.2s	10.8	11.9s
overshoot	20.5%	19.5%	4.5%

the PSoFLC only overshoots by 4.5% compared with 20.5% and 19.5% of the PD and SoFLC respectively.

PD The PD controller shows a smooth response, one big (20.5%) overshoot and little rudder activity.

SoFLC The rudder demand steadily increases from run to run. This is expected since the rulebase is set to zero at start up. Only when the learning progresses, a rudder demand $\delta_d \neq 0$ is generated by the controller. The controller however is not able to reduce the rules once they are built-up, resulting in the oscillatory behaviour as seen in all step response tests performed by the SoFLC. This could have been overcome using a lower learning gain. However, using a lower learning gain value would have prevented the rulebase from developing during the given time frame for comparison. this would have resulted in a very slow and lethargic controller.

PSoFLC It can be noticed, that the rudder demand is reduced even at the first run. This indicates that the controller after om.nly 40s has adapted the rules to a degree where control action can take place and no saturation occurs. By saturation, a necessary update of the rule in one direction away from zero is implied. The PSoFLC has the least rudder activity of only 10.99 $^{\circ 2}$.

Step Response Test at 600 rpm – 12 knots (Figures 5.9, 5.10, 5.11 and 5.16)

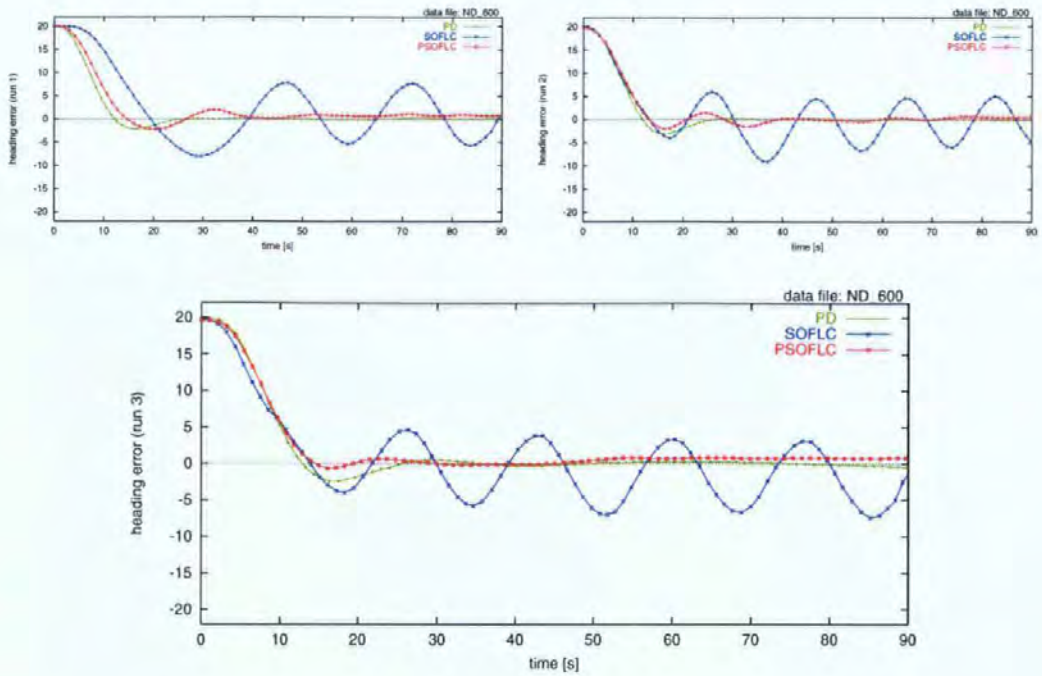


Figure 5.16. Step Response Test $\pm 20^{\circ}$ at 600 rpm errors

At medium speed (600 rpm = 12 knots), all controllers show similar behaviour to that observed at slow speed. Again, the PSoFLC has the least overshoot (third run). The only difference (from the 9 knots test), is that the rise times of all three controllers show much less divergence. However, the PSoFLC has the longest, 9.7s. The SoFLC demonstrates the same oscillatory behaviour, increased in frequency, reduced in amplitude. The rudder activity of the PSoFLC here (4.95 $^{\circ 2}$) is marginally less than that of the PD (5.76 $^{\circ 2}$). The same applies to the ISE, where the PSoFLC is 4% less than the PD. When considering only the first

Table 5.3. Controller Comparison, 600 rpm

criteria	PD	SoFLC	PSoFLC	% drop PSoFLC v PD	% drop PSoFLC v SoFLC
ISE 0–30s [$^{\circ 2}$]	92.01	86.32	87.70	-4.68%	1.60%
ISE [$^{\circ 2}$]	30.70	40.21	29.48	-3.99%	-26.69%

criteria	PD	SoFLC	PSoFLC
rudder activity[$^{\circ 2}$]	5.76	50.32	4.95
rise time to 60% steady state	8.8s	8.5s	9.7s
overshoot	12.0%	17.0%	3.0%

30s of the course change, then an improvement of 4.7% compared to the PD controller can be observed. This indicates, that the PD controller had performed better in the later phase ($t > 30s$).

PD The PD controller shows a smooth response, overshooting 12.0% before settling.

SoFLC The SoFLC drives the vessel in a very oscillatory manner. The amplitude is reduced and the frequency increased when compared with the SoFLC response at 9 knots. Comparing the overshoot of 17.0% it is considerable more then the overshoot of the PD and PSoFLC, 12.0% and 3.0% respectively.

PSoFLC Considering the first 30 seconds only, the course changing phase, the ISE of the PSoFLC is smaller than the ISE of the PD but marginally larger than the SoFLC. The PSoFLC overshoots only by 3.0%. The course keeping accuracy is not as good as the one produced by the PD controller. This is indicated by the decreased difference in ISE when the full 90s of the third run are considered. Over the full 90s of the third run, the ISE of the PSoFLC is 4% less (4.7% in the first 30s) then the ISE produced by the PD controller and 26.7% less then the one of the SoFLC. The PSoFLC has the least rudder activity of only $4.95\ ^{\circ 2}$.

The numbers show that the SoFLC shows the best performance during the first 30s of run 3. But since the SoFLC fails to settle within an acceptable band, this performance is not representative for the performance of the complete step response test.

Step Response Test at 1050 rpm – 21 knots (Figures 5.12, 5.13, 5.14 and 5.17)

At high speed (which is the tuning speed of the PD) the PD controller shows clearly the best response with respect to ISE. However, the rudder activity is higher than the PSoFLC, $1.89\ ^{\circ 2}$ and $1.48\ ^{\circ 2}$ respectively. The PD also has the shortest rise time of only 5.3s. The rise time of the PSoFLC is 6.7s and an overshoot is also noticeable. The SoFLC again produces a response of marginally stability as seen in the two previous tests.

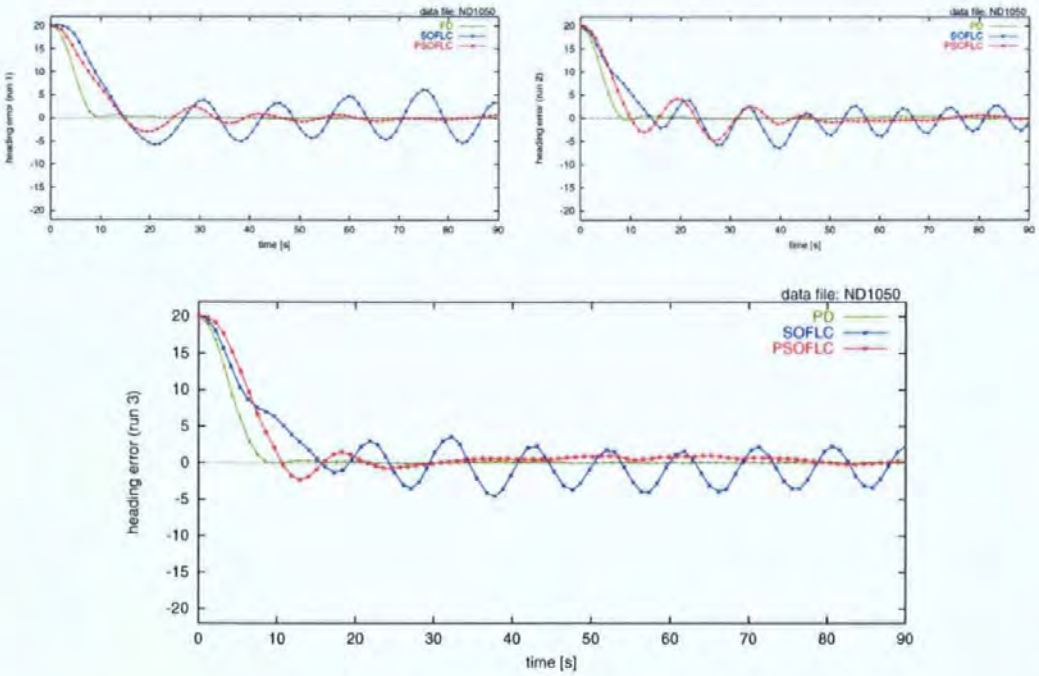


Figure 5.17. Step Response Test $\pm 20^{\circ}$ at 1050 rpm errors

PD At full speed (1050 rpm, 21 knots, the tuning speed of the PD controller), the PD controller shows the best ISE performance, with an ISE of only 16.579 $^{\circ 2}$. Virtually no overshoot is noticed.

SoFLC As seen at the slower forward speeds, the SoFLC has a very oscillatory behaviour, oscillating between $\pm 3^{\circ}$. The amplitude is reduced and the frequency increased when compared with the lower speeds. The rudder activity is about 8.4 times greater than that of the PD and 10.7 times greater than the PSoFLC.

Table 5.4. Controller Comparison, 1050 rpm

criteria	PD	SoFLC	PSoFLC	% drop PSoFLC v PD	% drop PSoFLC v SoFLC
ISE 0–30s [$^{\circ 2}$]	49.72	70.21	73.02	46.86%	4.01%
ISE [$^{\circ 2}$]	16.58	27.22	24.59	48.31%	-9.67%

criteria	PD	SoFLC	PSoFLC
rudder activity[$^{\circ 2}$]	1.89	15.90	1.48
rise time to 60% steady state	5.3s	7.5s	6.7s
overshoot	n/a	13.0%	9.5%

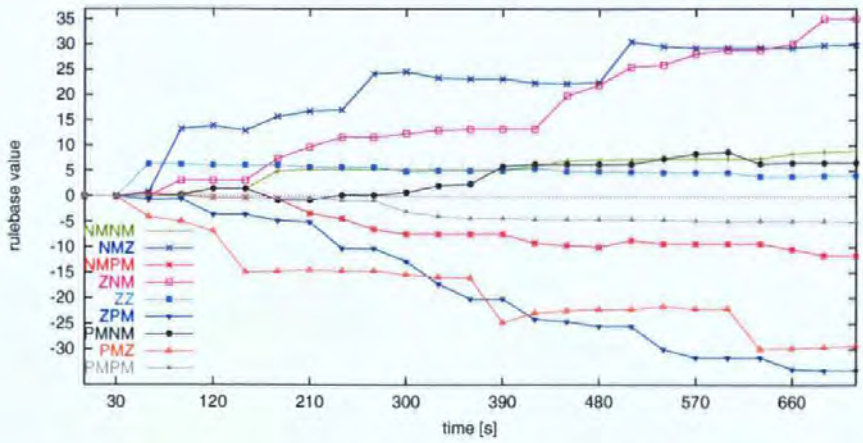


Figure 5.18. SoFLC Rulebase Development, 9 Selected Values

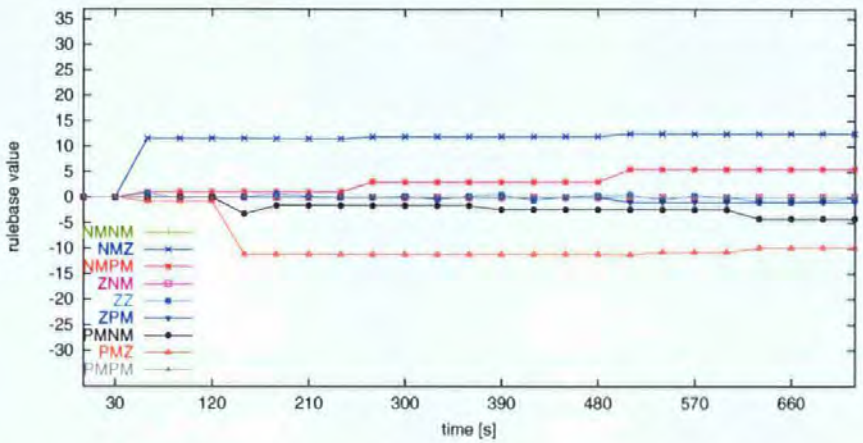


Figure 5.19. PSoFLC Rulebase Development, 9 Selected Values

PSoFLC The PSoFLC has a 48% increased ISE at this speed and a sightly longer (1.4s longer) rise time but it has the least rudder activity of only 1.48 $^{\circ 2}$. Figure 5.17 also shows two pronounced overshoots before settling. The rudder activity of the PSoFLC dropped by 21.7% when compared with that of the PD.

5.1.6 Analysis of the Rulebase Development

Appendix E contains two full sets of (24 individual) rulebases produced by the SoFLC and PSoFLC, logged in 30s intervals during a step response test at full speed. Figures 5.21 and 5.20 show a selection of four rulebases (start, number 8, 16 and final). The development of representative values in the rulebase is plotted in figures 5.18 and 5.19. Only 9 of the more important rules are monitored and their position in the rulebase is highlighted below in table 5.5. These rules are represented by the linguistic names NM/NM, NM/Z, NM/PM,

Table 5.5. Monitored Fields

	rate of change of error						
error	NB	NM	NS	Z	PS	PM	PB
NB	0.00	0.00	0.00	0.00	0.00	0.00	0.00
NM	0.00	**0.00**	0.00	**0.00**	0.00	**0.00**	0.00
NS	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Z	0.00	**0.00**	0.00	**0.00**	0.00	**0.00**	0.00
PS	0.00	0.00	0.00	0.00	0.00	0.00	0.00
PM	0.00	**0.00**	0.00	**0.00**	0.00	**0.00**	0.00
PB	0.00	0.00	0.00	0.00	0.00	0.00	0.00

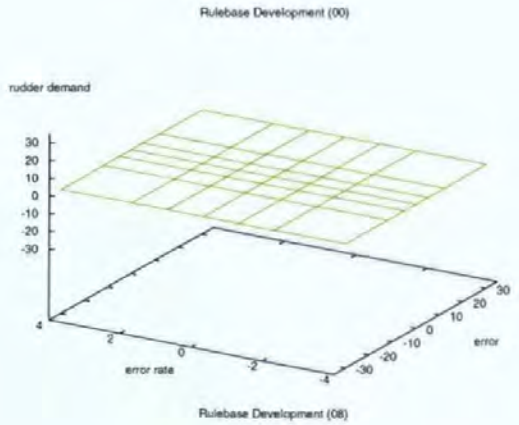
Z/NM, Z/Z, Z/PM, PM/NM, PM/Z, and PM/PM. Only nine rules are considered in this analysis as to increase the number generates confusion and restricts understanding. The rules chosen have been selected as they represent all areas of the rulebase activity and therefore indicate trends in the learning.

The individual plots in the SoFLC (figure 5.18) cross over and show a high activity. So that inner rules have a larger magnitude than outer rules. It would be expected that the largest changes in values (biggest gradient) should occur along the diagonal form NB/NB to PB/PB. This has not occurred as the learning is too immature and not all rules have been 'hit'. No settling can be noticed. Several of the values drift off to the maximum/ minimum value ± 35.

A zone of influence would have smoothed the rulebase, but would not have made a significant difference to the effect observed during the learning.

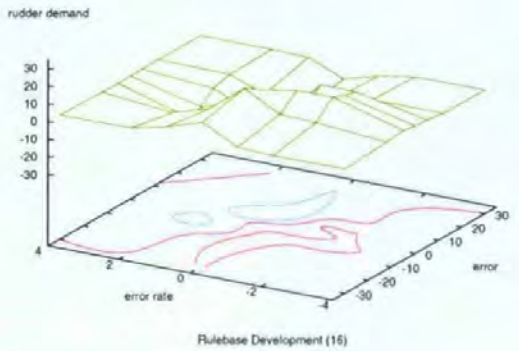
Since there is no governing rule guiding the learning, it can be argued that the rulebase eventually will settle with only the maximum/ minimum values and will emulate a bang–bang controller (between the $\pm$ limits). Another criterion is required which combines rudder activity (actuator) and heading error (the control objective) to form a new cost function, *ie* such as the one used by Åström and Eykhoff [9] (equation 2.11, page 19).

The rulebase development of the PSoFLC (figure 5.19) is smoother and appears to settle quickly. Learning is apparent only when the step changes were applied. The settling of the rules is very rapid. The learning is focused and effective unlike the distracted learning of the SoFLC. Unlearning effects were minimal. No double cross overs are observed, meaning that the outer rules have an expected greater magnitude than smaller, inner, rules. Performance of the PSoFLC is demonstrated as being effective and is a clear improvement over the SoFLC, showing expected learning trends.



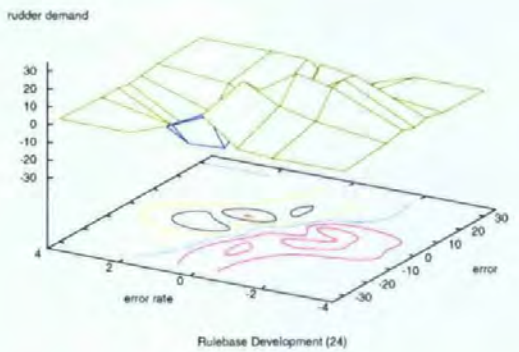
the preset rulebase

	NB	NM	NS	Z	PS	PM	PB
NB	0.00	0.00	0.00	0.00	0.00	0.00	0.00
NM	0.00	0.00	0.00	0.00	0.00	0.00	0.00
NS	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Z	0.00	0.00	0.00	0.00	0.00	0.00	0.00
PS	0.00	0.00	0.00	0.00	0.00	0.00	0.00
PM	0.00	0.00	0.00	0.00	0.00	0.00	0.00
PB	0.00	0.00	0.00	0.00	0.00	0.00	0.00



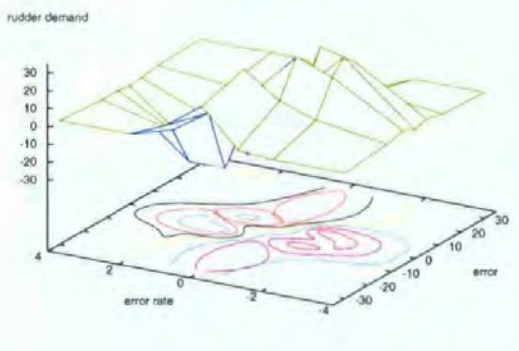
the changed rulebase at 17:29:14.36 (8)

	NB	NM	NS	Z	PS	PM	PB
NB	0.00	0.00	0.71	11.23	4.77	0.17	0.00
NM	1.85	5.19	6.11	16.99	1.56	-4.41	-0.55
NS	3.15	14.48	14.38	10.96	-8.14	-12.82	-0.72
Z	2.03	11.61	3.69	5.73	-3.98	-10.25	-0.37
PS	1.66	9.92	10.23	-10.81	-18.46	-3.46	-0.20
PM	0.15	0.20	-2.75	-14.69	-8.03	-0.83	0.00
PB	0.00	-0.42	-3.56	-8.98	-0.71	0.00	0.00



the changed rulebase at 17:33:14.38 (16)

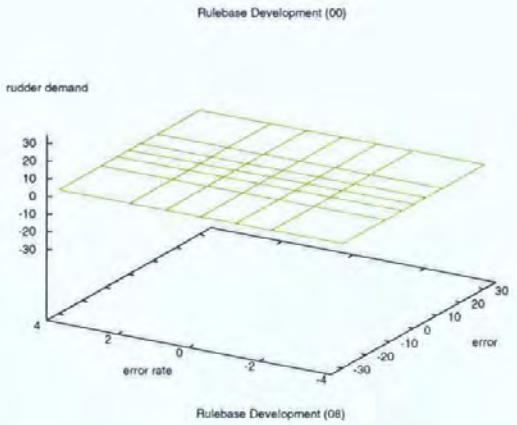
	NB	NM	NS	Z	PS	PM	PB
NB	0.00	0.00	1.38	16.68	6.64	-0.65	-0.72
NM	2.44	7.24	12.05	22.46	-1.27	-9.99	-2.01
NS	4.54	20.98	25.25	12.11	-24.55	-25.74	-1.87
Z	2.25	21.97	10.29	4.93	-10.32	-25.53	-3.37
PS	4.03	20.50	22.97	-16.61	-31.55	-15.88	-2.59
PM	1.94	6.35	-2.51	-22.22	-13.43	-4.52	-1.00
PB	0.72	0.58	-5.72	-14.49	-0.71	0.00	0.00



the changed rulebase at 17:37:36.73 (24)

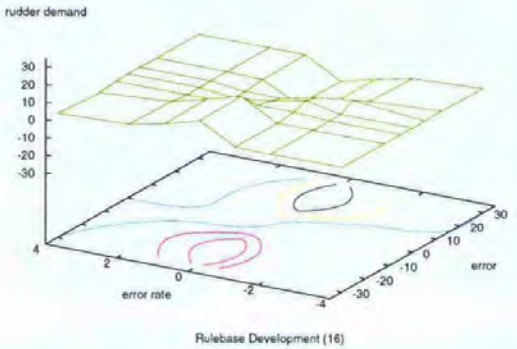
	NB	NM	NS	Z	PS	PM	PB
NB	0.00	0.00	1.38	22.34	8.18	-0.60	-0.62
NM	2.71	8.85	15.74	29.76	1.33	-11.74	-2.48
NS	5.17	27.50	35.00	14.53	-35.00	-35.00	-2.45
Z	2.25	35.00	16.58	3.94	-15.05	-34.19	-3.59
PS	4.23	28.42	33.60	-18.49	-35.00	-19.38	-2.81
PM	1.94	6.54	-6.14	-29.40	-16.54	-4.95	-1.00
PB	0.60	0.28	-8.06	-19.83	-0.71	0.00	0.00

Figure 5.20. SoFLC – Rulebases



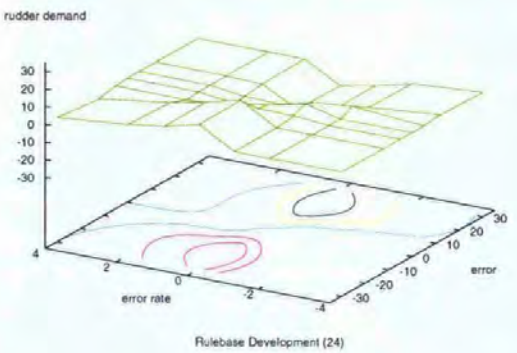
the preset rulebase

	NB	NM	NS	Z	PS	PM	PB
NB	0.00	0.00	0.00	0.00	0.00	0.00	0.00
NM	0.00	0.00	0.00	0.00	0.00	0.00	0.00
NS	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Z	0.00	0.00	0.00	0.00	0.00	0.00	0.00
PS	0.00	0.00	0.00	0.00	0.00	0.00	0.00
PM	0.00	0.00	0.00	0.00	0.00	0.00	0.00
PB	0.00	0.00	0.00	0.00	0.00	0.00	0.00



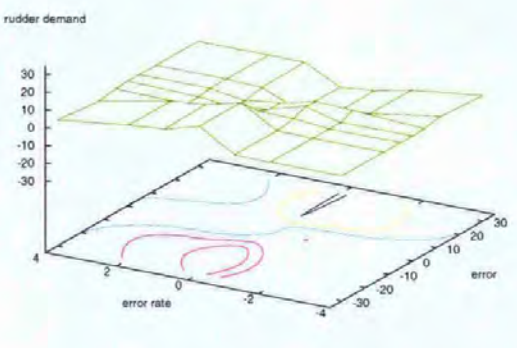
the changed rulebase at 11:29:20.30 (8)

	NB	NM	NS	Z	PS	PM	PB
NB	0.00	0.00	0.00	11.24	5.62	1.70	0.08
NM	0.00	0.00	0.14	11.47	6.07	1.03	-1.62
NS	0.00	0.00	1.22	1.13	-0.30	-0.50	-0.36
Z	0.00	0.00	4.02	-0.01	-2.06	-0.02	0.00
PS	0.12	4.17	1.85	-2.30	-2.18	0.00	0.00
PM	0.69	-1.63	-6.33	-11.20	-0.61	0.00	0.00
PB	-0.04	-2.22	-6.42	-11.39	0.00	0.00	0.00



the changed rulebase at 11:33:20.39 1(16)

	NB	NM	NS	Z	PS	PM	PB
NB	0.00	0.00	0.00	11.75	6.54	4.00	0.29
NM	0.00	0.00	0.17	11.88	6.96	2.95	-2.70
NS	0.00	0.00	1.74	1.99	-0.47	-0.86	-0.60
Z	0.00	0.00	4.82	0.35	-2.61	-0.06	0.00
PS	0.02	3.42	1.51	-2.88	-2.50	0.00	0.00
PM	1.16	-2.41	-7.91	-11.20	-0.64	0.00	0.00
PB	0.28	-2.52	-7.43	-11.32	0.00	0.00	0.00



the changed rulebase at 11:37:37.02 (24)

	NB	NM	NS	Z	PS	PM	PB
NB	0.00	0.00	0.16	12.41	6.97	5.41	0.55
NM	0.00	0.00	0.37	12.44	7.36	5.34	-2.23
NS	0.00	0.00	1.96	2.57	-0.61	-1.46	-0.56
Z	0.00	0.00	5.17	-0.30	-3.32	-1.04	0.00
PS	0.31	3.08	3.98	-0.59	-2.99	-0.02	0.00
PM	2.11	-4.19	-8.25	-9.97	-0.74	0.00	0.00
PB	0.20	-4.41	-8.26	-10.40	-0.01	0.00	0.00

Figure 5.21. PSoFLC – Rulebases

5.1.7 Discussion of Step Change Test Results

Statistical analysis of the plots are summarised in table 5.6. The PD controller cannot always be out-performed when operated in an environment without disturbances. Considering only the trained PSoFLC then an improvement over the PD can be noticed as well. Very noticeably, at low and medium speed, the PSoFLC overshoots clearly less than both of the competing controllers. This has been achieved with a slight increase in the rise time.

The first of the two tables (table 5.6) only compares SoFLC with PSoFLC in order to show the ability and speed of adaptation. After the initial training period, the PD controller is added for comparison (table 5.7). At this stage, both adaptive methods will have reached an almost steady state in their learning, assuming that no further change in environment takes place.

An improvement can be seen when the SoFLC is compared with the PSoFLC. At the speed of 12 knots (600 rpm) (figures 5.10 and 5.11) the ISE error dropped by over 26% and at full speed (1050 rpm) (figures 5.13 and 5.14) by still approximately 9.7% and approximately 16% at slow speed of 9 knots (figures 5.7 and 5.8). The PSoFLC has the least rudder activity at all three tested forward speeds.

This indicates, that the SoFLC has learnt the environment and achieved a desirable control performance over time. However, the PSoFLC in comparison demonstrated a much better, hence faster, adjustment of the control parameters and therefore the overall error was reduced considerably especially during the initial step change. This proves that the predictive algorithm adapts much faster than the standard self–organising method.

In all cases, both the SoFLC and the PSoFLC start with an empty rulebase, which can be seen in the initial phase of the application of the controller (*see* first runs). The rudder stays centred ($\delta \approx 0$) until some control action has been learnt and a rudder angle is applied in order to reduce a heading error. The expressions as found in table 5.6 can be seen as $ISE = \frac{1}{T}\int_0^T \psi_e{}^2 \mathrm{d}t$ and $rudder\ activity = \frac{1}{T}\int_0^T \delta^2 \mathrm{d}t$ and the percentages are obtained using the equation below.

$$\text{percentage drop } [\%] = -\frac{\text{ISE}_{\text{benchmark controller}} - \text{ISE}_{\text{PSoFLC}}}{\text{ISE}_{\text{benchmark controller}}} \times 100\% \qquad (5.1)$$

Table 5.6. Summary - Error Comparison of Step Response without Disturbances

controller rpm	SoFLC ISE [°²]	PSoFLC ISE [°²]	improvement of PSoFLC over SoFLC
450	44.400	30.315	-31.72%
600	39.620	26.190	-33.90%
1050	28.030	20.879	-25.51%

Table 5.7. Summary - Error Comparison, last step only

controller	PD				SoFLC			
rpm	ISE [°²]	rise time	over-shoot	rud. activity [°²]	ISE [°²]	rise time	over-shoot	rud. activity [°²]
450	36.619	11.2s	20.5%	11.20	49.759	10.8s	19.5%	81.19
600	30.704	8.8s	12.0%	5.76	40.211	8.5s	17.0%	50.32
1050	16.579	5.3s	n/a	1.89	27.222	7.5s	13.0%	15.90

controller	PSoFLC				ISE improvement of PSoFLC over	
rpm	ISE [°²]	rise time	over-shoot	rud. activity [°²]	PD	SoFLC
450	41.821	11.9s	4.5%	10.99	14.21%	-15.95%
600	29.478	9.7s	3.0%	4.95	**-3.99%**	-26.69%
1050	24.589	6.7s	9.5%	1.48	48.31%	-9.67%

Figures 5.15, 5.16, 5.17 summarise the heading errors for all three autopilots. The controllers PD, SoFLC and PSoFLC are displayed in different colour, green, blue and red respectively. The first plot of the three on the same page clearly highlights the better performance of the PSoFLC when compared directly to the pilot using historic data for parameter adjustment. Here the advantage of using an internal predictor is clearly observable, resulting in faster learning and error reduction. The SoFLC is still oscillating even after six course changes (the plots only show the even steps for better presentation) while the predictive controller shows hardly any overshoot at all. The SoFLC shows marginally stable response at all three speeds. Figures 5.15 - 5.17 show the heading errors of the individual runs of all three controllers in one graph respectively.

Table 5.7 shows a very noticeable result. After the initial training period, the PSoFLC outperforms the PD controller. At medium speed the PSoFLC ISE error dropped by approximately 4%. During the initial testing, this was not expected but is a welcome fact.

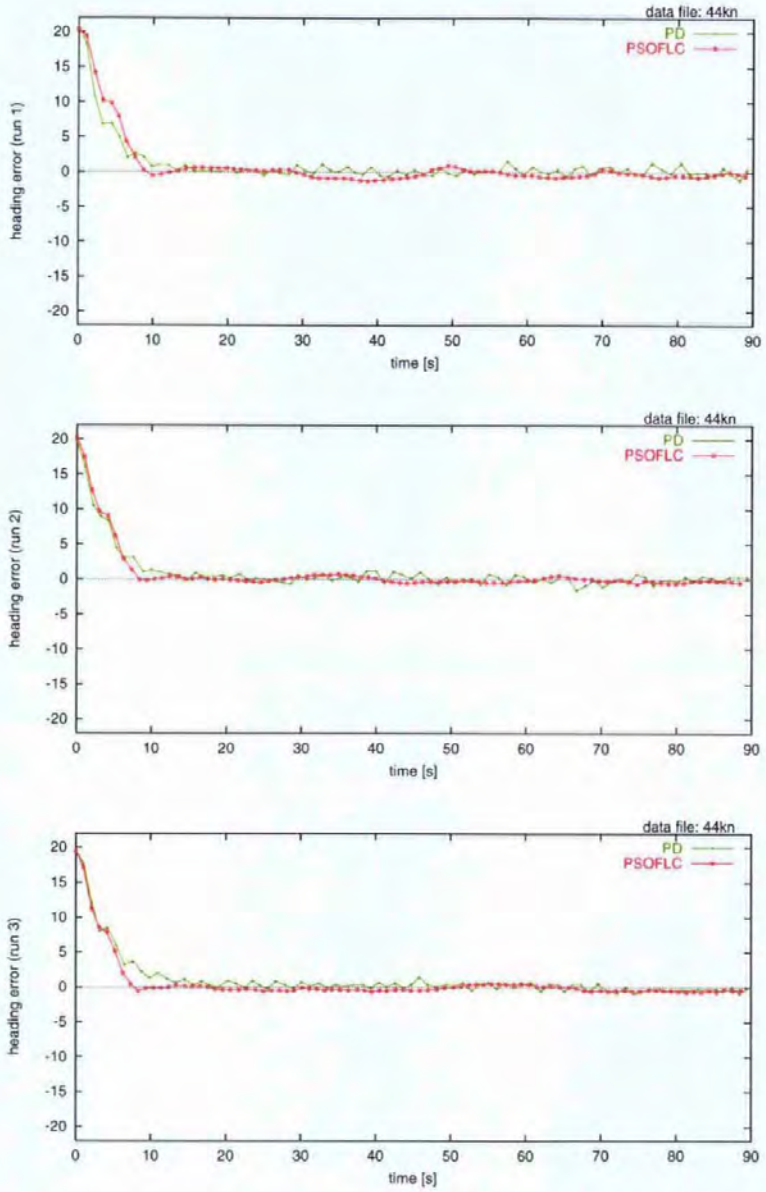


Figure 5.22. Step Response Test $\pm 20^{\circ}$ at 44 knots errors

Figure 5.22 once more shows the heading error. This time, the vessel is travelling at 44 knots. The PD controller doesn't settle as quickly as the PSoFLC does, and during course keeping, the PD response is more oscillatory.

Table 5.8. Controller Comparison, first 30s

rpm	ISE_{PD}	ISE_{SoFLC}	ISE_{PSoFLC}	% drop PD vs PSoFLC	% drop SoFLC vs PSoFLC
450	109.70	105.12	123.90	12.95%	17.87%
600	92.01	86.32	87.70	-4.68%	1.60%
1050	49.72	70.21	73.02	46.86%	4.01%

Table 5.9 Summary - Error Comparison, travelling at 44 knots (79.2 $km \cdot h^{-1}$) last step only

controller	ISE [°²]	rise time	over-shoot	5% sett-ling time	rudder ac-tivity [°²]
PD	12.755	5.4s	n/a	15.67s	1.669
PSoFLC	11.383	4.2s	2.9%	7.36s	0.489
improvement	-10.757%	-22.2%	n/a	-53.03%	-70.70%

When the same PD controller is installed on a different vessel, *ie* one travelling at 44 knots (81.5 $km \cdot h^{-1}$) the PD controller shows an oscillating behaviour, whereas the PSoFLC adapts and shows a smoother response. Figure 5.22 shows three consecutive step changes of the PD and PSoFLC in the same environment. The high speed tests are performed in calm environment as the previous step change tests. The controllers sampling time remained unchanged throughout the simulated tests. Particularly in figure 5.22 it can be noticed that the dominant frequency and the sampling frequency almost match.

From this finding, it can be concluded, that the sampling frequency and the frequency of storing data for visualisation should have been increased to allow for the decreased time constant of the vessel travelling at such high speed. Both tests, the PD and the PSoFLC, use the same frequency and environmental settings, so a qualitative comparison is still possible and valid. However, the results shown in figure 5.22 and table 5.9 are not invalidated by the lower sampling frequency used.

The control performance of the PSoFLC shows a considerable improvement over the PD performance when the environment is unknown to both controllers.

The PSoFLC has a settling time of 7.4s, which is less than one-half of the settling time of the PD's settling time (15.7s). The PD shows a very oscillating response, changing course at a much higher frequency than the PSoFLC.

5.2 Test with Disturbances

5.2.1 The Disturbances

The life boat simulation [20] includes some model of environmental disturbances. Waves are modelled as a simple sine wave. The maximum height and length can be varied by the operator.

For the following tests, the vessel is exposed to weather disturbances such as wind, tide and waves. The disturbances are summarised in table 5.10. The wind gusts are randomly applied within the specified limit. All angles specified (*see* table 5.10 and figure 5.23) are absolute angles. For both the SoFLC and the PSoFLC, the initial rulebase of the controller is empty. This is the worst case scenario for both controllers.

Table 5.10. Summary of Disturbances acting on the Vessel

		acting from
waves	1.0 m	+40°
tide	3.7 km · h^{-1}	+45°
wind	9.3 km · h^{-1}	+50°
gusts	1.8 km · h^{-1}	

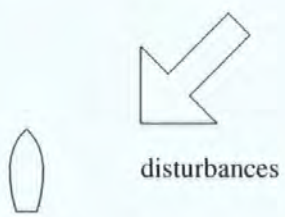


Figure 5.23. Straight Line Test with Disturbances

The disturbances (wave height) acting on the vessel are presented in figure 5.24. This image shows the wave height during a 300s period.

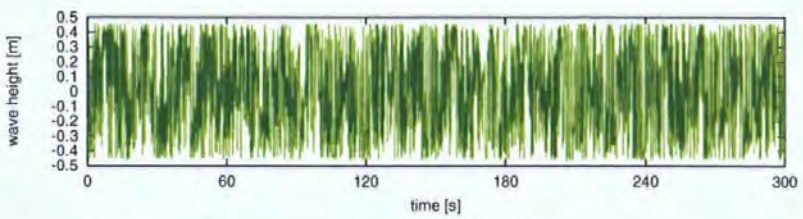


Figure 5.24. Acting Disturbances (Wave Height)

5.2.2 Straight Line Test

The following figures (figures 5.25–5.33) show the results of the tests carried out with disturbances present. The controller is asked to steer the vessel in a straight line orienting the vessel exactly North (0°). The tests are performed at three different forward speeds, 9, 12 and 21 knots. For analysis, a 120s window is selected starting at 380s. The first 380s are used to tune the controllers to work in the exposed environment. This test is performed to demonstrate the course keeping capabilities of the variuous controllers.

5.2.3 Graphs

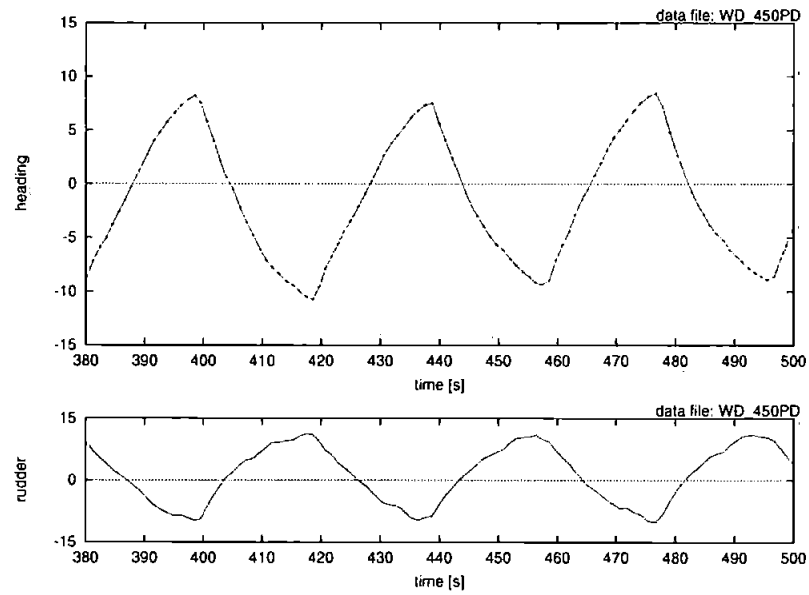


Figure 5.25. Straight Line at 450 rpm with Disturbances PD

Figure 5.25 shows the heading and corresponding rudder movement for the straight line $-10°$ and $+8°$.

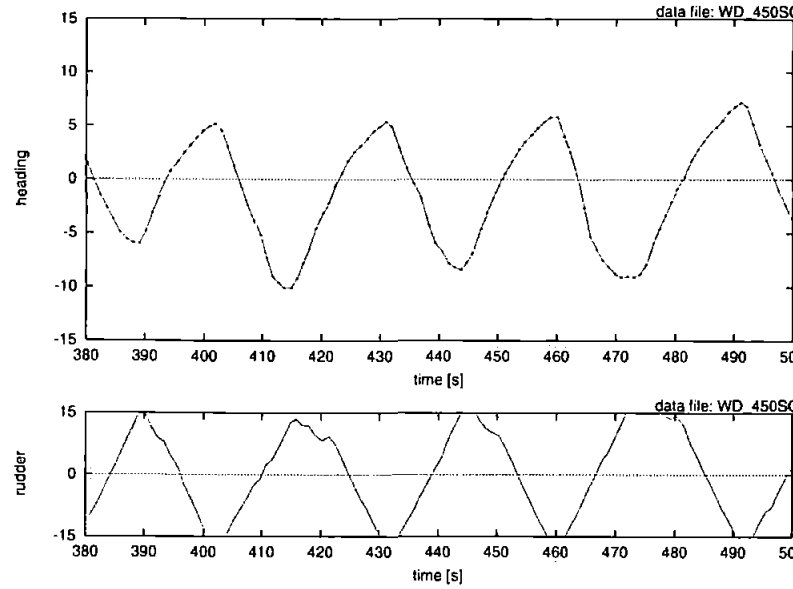


Figure 5.26. Straight Line at 450 rpm with Disturbances SoFLC

Figure 5.26 shows the response of the vessel when under control of the SoFLC at the same slow forward speed of 9 knots. The heading oscillates between approximately $-9°$ and $+6°$. The rudder oscillates between approximately $-17°$ and $+18°$.

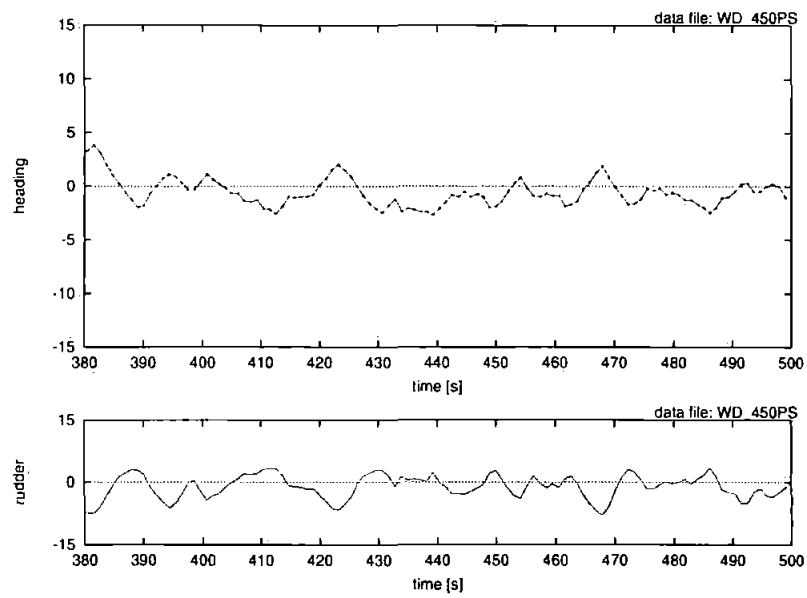


Figure 5.27. Straight Line at 450 rpm with Disturbances PSoFLC)

The response of the vessel when under control of the PSoFLC is displayed in figure 5.27. Both, the heading error and the rudder do not show a harmonic oscillation as seen before. The heading stays within a band of $-2°$and $+4°$. The rudder operates between $-8°$ and $+3°$.

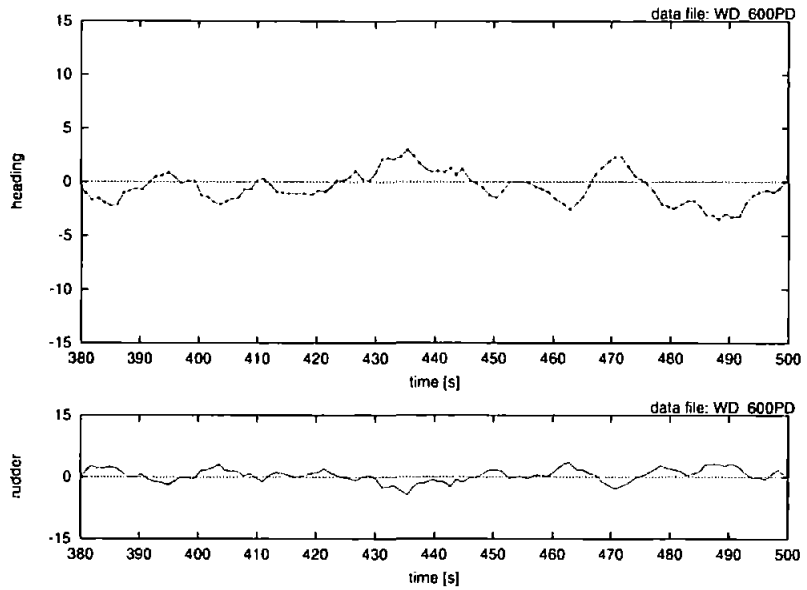


Figure 5.28. Straight Line at 600 rpm with Disturbances PD

Figure 5.28 visualises the heading and rudder response of the PD controller at a forward speed of 12 knots (22.2 $km \cdot h^{-1}$) . During the 120s displayed, the course error does not exceed $\pm 3°$. The rudder operates in approximately the same region, between $\pm 3°$.

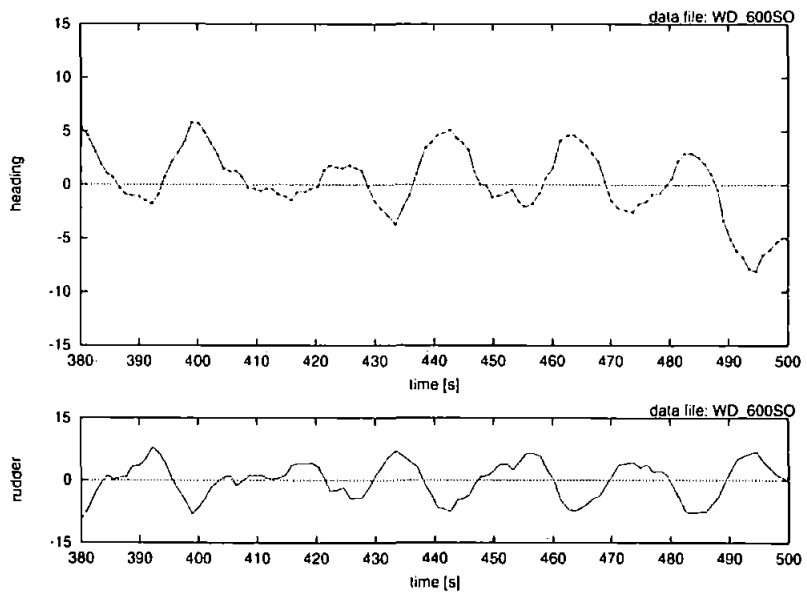


Figure 5.29. Straight Line at 600 rpm with Disturbances SoFLC

Figure 5.29 displays the response of the SoFLC. The heading error shows a tendency to oscillatory behaviour. The amplitude of the oscillation is approximately 10° but wandering off. The rudder stays within a $\pm 10°$ band.

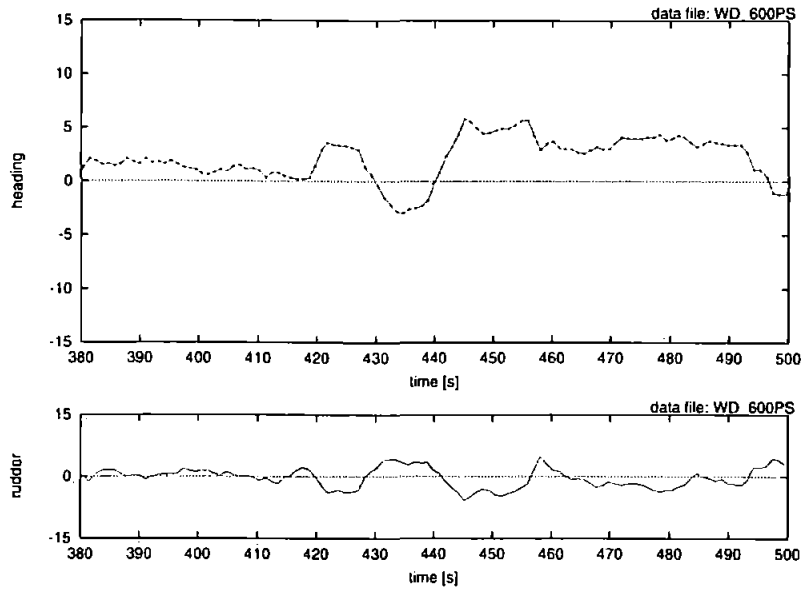


Figure 5.30. Straight Line at 600 rpm with Disturbances PSoFLC

Figure 5.30 shows the a 120s window from the straight line test performed at a 12 knots speed. The heading shows no sign from low frequency oscillatory behaviour. The vessel drifted off course for approximately 40s before reducing this error. The rudder activity is low.

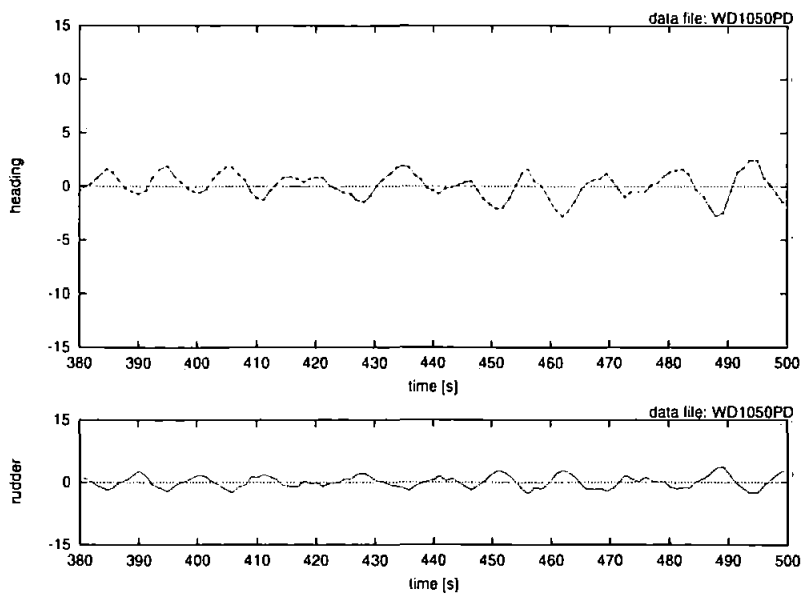


Figure 5.31. Straight Line at 1050 rpm with Disturbances PD

Figure 5.31 shows the vessel's response travelling at full speed of 21 knots (38.9 $km \cdot h^{-1}$). Some oscillatory behaviour can be noticed but the amplitude is small (approximately $\pm 2°$).

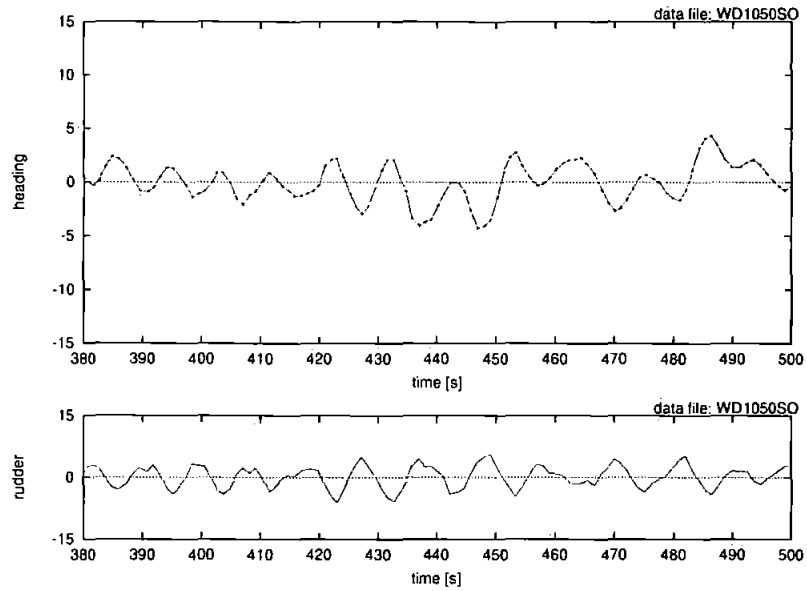


Figure 5.32. Straight Line at 1050 rpm with Disturbances SoFLC

A more oscillatory response can be noticed by the SoFLC as displayed in figure 5.32. The amplitude is increased to that seen by the PD controller. The rudder activity reflects this movement, oscillating between $\pm 5°$.

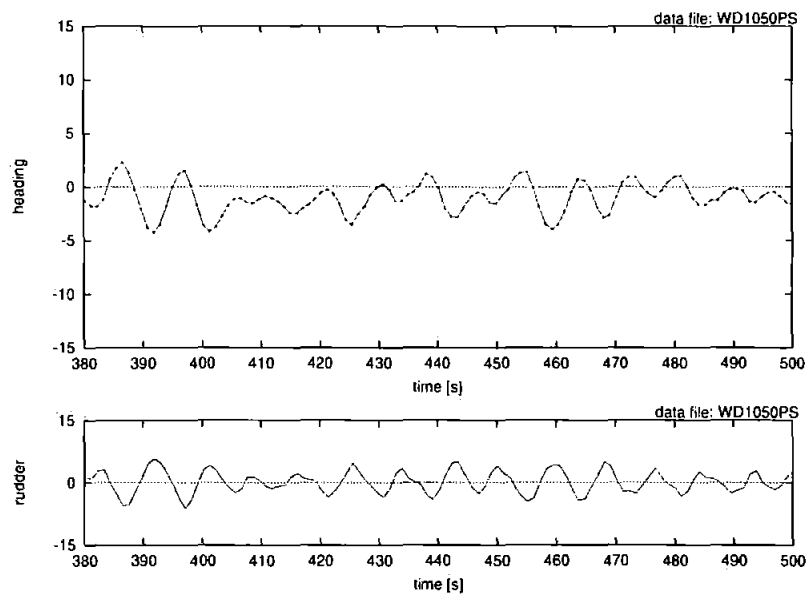


Figure 5.33. Straight Line at 1050 rpm with Disturbances PSoFLC

Figure 5.33 displays the response of the vessel when under control of the PSoFLC. The heading oscillates between $-4°$ and $+3°$.

5.2.4 Discussion

Course Keeping Test at 450 rpm – 9 knots

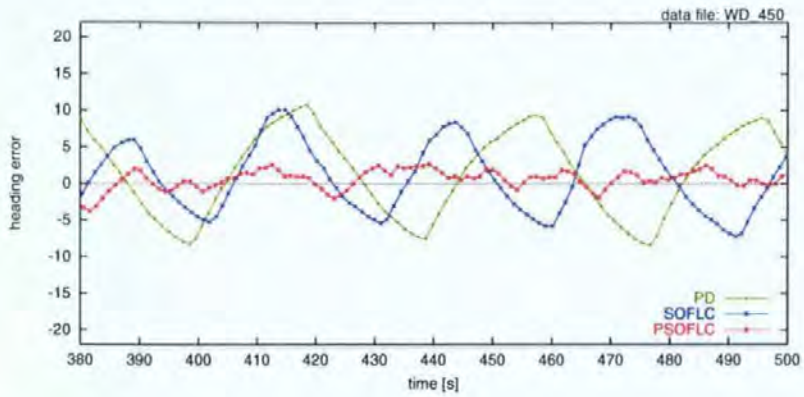


Figure 5.34. Straight Line at 450 rpm with Disturbances (errors)

As seen in the figures 5.25, 5.26, 5.27 and 5.34, at low speed all the controllers apply more rudder than at higher speeds, which indicates the better manoeuvrability of the vessel at higher forward speed.

The rudder activity of the SoFLC in figure 5.26 shows an increasing tendency which indicates a very oscillating behaviour. When the SoFLC is utilised at a urban slow speed as in figure 5.26 then the heading error cannot be reduced which leads to the conclusion that this type of controller is marginally stable.

The PSoFLC keeps the heading error between $\pm 4°$whereas the other two controllers exceed that boundary by a very noticeable amount.

Course Keeping Test at 600 rpm – 12 knots

At medium speed (600 rmp = 12 knots, figures 5.28, 5.29, 5.30 and 5.35), all controllers can handle the vessel keep the course between reasonable bounds. The PD controller demonstrates the best alternative here, with the smallest ISE and the least rudder activity. All three controllers keep the error within $\pm 5°$during the 120s time sample.

Course Keeping Test at 1050 rpm – 21 knots

At high speed (1050 rpm = 21 knots, figures 5.31, 5.32, 5.33 and 5.36) the vessel is at its most responsive state. The ISE become very similar and the rudder activity differs only

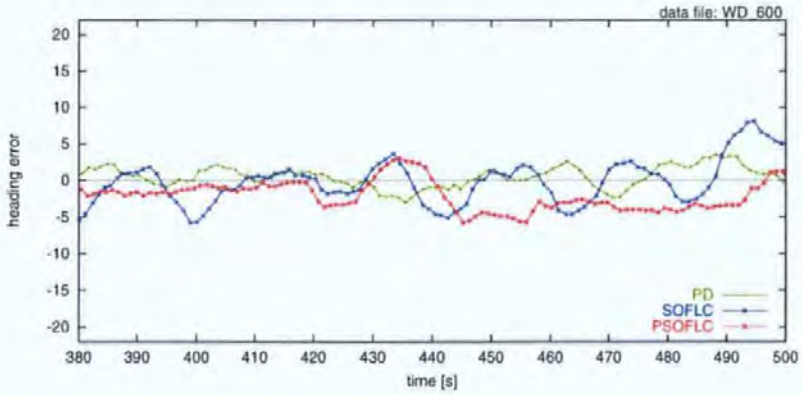


Figure 5.35. Straight Line at 600 rpm with Disturbances (errors)

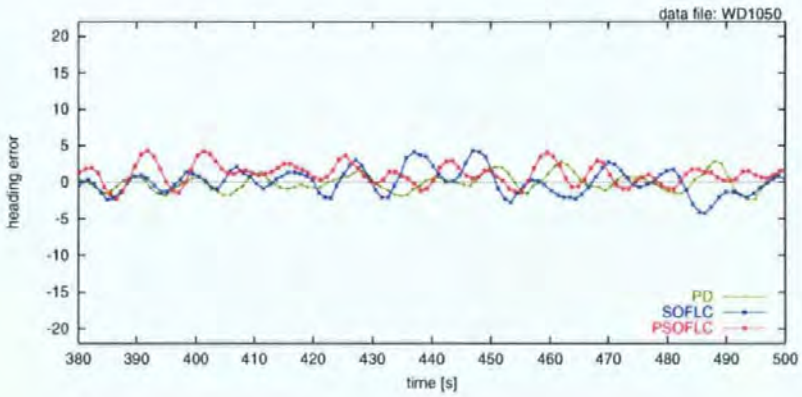


Figure 5.36. Straight Line at 1050 rpm with Disturbances (errors)

slightly in absolute terms. However, the PD shows the best result here as well. The PSoFLC comes in as second closely followed by the SoFLC.

General

The PSoFLC performs well with the changing environment and the disturbances. A summary of the errors of the different controllers can be found in table 5.11. The expressions found in this table are the same as before; $ISE = \frac{1}{T}\int_0^T \psi_e{}^2 \mathrm{d}t$ for the heading error and $\textit{rudder activity} = \frac{1}{T}\int_0^T \delta^2 \mathrm{d}t$.

The percentages in the tables 5.11 and 5.12 are obtained as follows:

$$\text{percentage drop } [\%] = -\frac{\text{ISE}_{\text{benchmark controller}} - \text{ISE}_{\text{PSoFLC}}}{\text{ISE}_{\text{benchmark controller}}} \times 100\%.$$

At higher speed, the rudder becomes far more effective. At lower speeds, the disturbances take over and therefore the rudder has to work harder.

Table 5.11. Summary - Error Comparison Course Keeping Response with Disturbances

controller rpm	PD ISE [°²]	SoFLC ISE [°²]	PSoFLC ISE [°²]	improvement of PSoFLC over SoFLC
450	34.173	35.607	4.665	-86.90%
600	1.728	14.148	11.962	-15.45%
1050	0.914	5.532	4.380	-20.82%

Table 5.12. Summary - Error Comparison, 120s Time Window

controller	PD		SoFLC		PSoFLC		improvement of PSoFLC over	
rpm	ISE [°²]	rud. activity [°²]	ISE [°²]	rud. activity [°²]	ISE [°²]	rud. activity [°²]	PD	SPFLC
450	36.194	48.34	25.398	116.28	1.985	9.04	**-94.94%**	-92.18%
600	2.278	2.63	9.298	18.86	8.539	5.76	274.85%	-8.16%
1050	1.324	2.09	3.146	7.01	3.091	7.01	133.46%	-1.75%

It can be seen in all plots, that the high rudder activity is a compilation of multiple frequencies. The dominant (low) frequency is mainly influenced by the vessel itself whereas the higher frequencies are caused by the disturbances such as waves.

It is interesting to observe, that the PSoFLC has not shown a *preferred* speed. It seems that at all speeds the heading errors produced by the PSoFLC are of the same magnitude whereas the other controllers work better at the higher speeds. This can be seen in the improved performance (reduced ISE) at higher speed.

Chapter 6

Discussion

6.1 General Points

The main objective of this research was to explore how a predictive control can be applied to a motorised marine craft. The concept of predictive control is not new but up until now, considerable knowledge and experience in the marine and control field was required to successfully implement such a predictive controller. The method discussed here shows an alternative controller design combining on-line predictive and adaptive methods. The aim is to show how to achieve and sustain a desirable control performance even if the operator has limited control knowledge.

In this work an approach is shown which achieves this aim.

PD and PID type controllers are still widely used in autopilots. They have proven to be stable and effective. It is current practice to factory pre-set the PID parameters in the hope that they will be suitable for a range of vessels. The result is that they are nowhere near their optimum setting for any particular craft. Commercial pressures demand more sophisticated guidance and better tuned autopilots. Past research in the field of PID control [10, 29] concentrated on the improvement of tuning the PID control algorithm by retaining the structure of the PID but changing the controller parameters as required. Those adjustments have increased the working range of the classical PID but fall short of delivering the performance required from modern autopilots.

Other control techniques [35] have been investigated and found their way into applications in marine autopilots. The increase in computing power of up-to-date chip technology

has expanded the field of autopilot design using purely digital hardware.

With the new approaches, non-linear methods can be pursued and the human intuition finds its way directly into the design of the autopilots by considering the human way of thinking and where possible the human way of learning, *eg* fuzzy logic.

The working environment is inherently non-linear. This is conflicting with classical control methods which required linearisation. With the advent of powerful computers, emphasis on this requirement diminishes and non-linear methods now appear and find application in current customer products. One of the biggest problems to resolve for a small boat autopilot is the extremely high susceptibility to small changes in a wide range of working conditions. For a non-adjustable, pre-set autopilot this creates a difficulty in sustaining the maximum efficiency of the autopilot, by limiting the range of optimal performance.

There is an increasing customer demand for high-tech performance without the need for much operator input, according to Cetrek Ltd. - manufacturer of marine autopilots. Self-tuning and self-organising controllers have found their way into many branches of modern industries. The more complex and sophisticated a controller, the more effort needs to be put into setup and installation. It is common practice to have an initialisation routine (manual or automatic) for a system where critical control parameters are to be adjusted. This routine requires a high level of knowledge and expertise in those fields to which the controller is applied. In practical applications where the operator has no control engineering knowledge then the system itself needs to have built-in self-identification and adaptation facilities. To the author's knowledge, there is no commercially available small craft autopilot that can effectively 'boot' itself from such a standing start (empty rulebase) and with such rapidity as that found in the PSoFLC developed as a result of this research investigation. Previous work on SoFLC for small craft [81] has show that (given the same working range) only a slow learning rate (adapting) could be achieved. Manual initialisation of the system requires the operator or installer to enter boat information. This shortcoming is inherent in the adaptation algorithm employed. This information might not be available to the installer of the control system. The boat, even the same type, can be differently equipped and therefore have a different mass. These and other variations cause different steering characteristics.

6.2 New Fuzzy Logic Defuzzification

Although the defuzzification methods commonly known work well, when used as a controller they can put unnecessary load and stress on the actuator if the control surface is not smooth. This effect applies to all FLCs utilising 'centre of area' defuzzification methods.

Chapter 4.4 shows a novel defuzzification method to overcome rough and stochastic behaviour as found in fuzzy logic control to-date. Applying a controller with rough and stochastic behaviour to a ship would result in unnecessary wear and tear of the rudder gear. By applying the new defuzzification method a smoother but still accurate response can be achieved, increasing the lifespan of the actuator equipment. The method demonstrated here gives engineers a simple solution to overcome these shortcomings without sacrificing any of the benefits gained by using fuzzy logic. Both fuzzy singletons and the 'normalisation technique', do not consider any area 'under' the set. Only their position in the universe of discourse is taken into account. The calculated output of controllers using fuzzy singletons therefore is almost equivalent to the output of controllers using the normalisation technique (*compare* figure B.13 [page 156] figure 4.12 [page 52]). The defuzzified output will be the same when regularly shaped sets are used and the singleton is placed in the centre of the set. The output value will vary slightly when the sets are irregularly shaped (*ie* lean to one side).

6.3 Test Bed

The development of a graphical user interface (appendix C.2) proved to be most valuable. All the testing of the various controllers has been done using this program. This program allows the simulations to be exactly repeated for each controller.The test bed logs data (time, longitude, latitude, heading, desired head, rudder, demanded rudder, speed, wind amplitude, wind velocity, wave height, controller, type) which is then available in a file after the program is exited. Off-line analysis is therefore possible. Rudder angle and heading gauges are displayed on the screen which give an instant reading of their respective positions. History plots of rudder angle and heading are visible at the bottom of the screen. For a screen shot and more detail on the test bed developed, be referred to appendix C.2.

6.4 Predictor Design

Models are the basis for nondestructive development and preliminary design. Mathematical modelling (the mathematical representation of a plant [ship]) has been used in control engineering for a long time.Using models in the operational controller itself, as reference models or other comparative means, has only become part of controller design since the development of fast computers. The inclusion of a mathematical model inside the controller allows the development of a predictive scheme. To fully utilise both techniques (internal modelling and self-organising fuzzy logic) an adaptive internal model is required.

Here, an on-line learning neural network has been used as the adaptive model. The emphasis is on *adaptive* since the controller to be developed has an adaptive nature and is able to change characteristics when the environment changes. It was demonstrated [96, 86, 132, 84] that neural networks are good at capturing time variant processes. The neural network also acts as a filter. When properly trained it will interpolate between points and smoothen out noisy data.

The back-propagation network as used in the Predictor has only two hidden layers. During operation, the network is constantly fed (0.5s sample time) with new sensor data. Once every 10s the network is trained to capture the latest changes in the environment. The output of the network suggests a future state from a given (current and past) state. The length of the time window fed to the network is, in this set-up, 20 seconds, 40 samples. The size of the training set depends not only on the time constant of the plant under control but also on the environment.

If the controller was to be applied to a different application, a different time window would be chosen in order to capture the process and environment dynamics. Using faster hardware, the sample size could be increased. Care has to be taken however, not to overfeed the neural network. Finding the optimum size of the neural network outstands the scope of this research. Experiments have to be undertaken to find a good compromise between recall quality and learning time when deciding on the samples size as well as on the size of the neural network itself. There should always be at least three time steps present in the input vector so the network is able to capture time transient behaviour.

6.5 Controller Comparison

This section discusses the results as visualised in chapter 5. The controllers to be considered are namely the PSoFLC, SoFLC and the PD.

The test vessel on which the autopilot has been tested is a 52 ft (15.8m) life boat simulation [20]. The PSoFLController developed in this investigation combines self-organising with predictive methods. This combination forms an unique controller, able to 'boot' from scratch and to deliver acceptable control in a very short time.

The results from chapter 5 clearly show the advantages of the PSoFLC compared to the SoFLC when exposed to the same environment. The PSoFLC adapts very quickly, much quicker then the SoFLC. The course plotted in figures 5.2 ('figure of 8' manoeuvre, SoFLC) shows a very distorted first 'figure of 8'. The same manoeuvre, the same starting point but executed by the PSoFLC (fig 5.4) shows a much more recognisable first 'figure of 8'. Giving both controllers some time to adapt, *ie* comparing the third 'figure of 8', then only a small improvement of the PSoFLC can be observed. So, over time, both methods will learn the environment and control the vessel. This not only applies to the 'figure of 8' test but also to the performed step response tests and the course keeping test. However, both adaptive methods did not demonstrate superior behaviour compared with a tuned PD controller.

Comparing the results from chapter 5 the following conclusions may be made. The PSoFLC demonstrates a smoother response at all speeds and the SoFLC enters into an oscillatory response at all speeds. The SoFLC is unable to 'unwind' (reduce) previously learnt controller gains, hence the oscillatory behaviour. This also indicates, that the update algorithm used in the SoFLC does not necessarily converge in all circumstances. Furthermore, it can be expected, that the produced controller may become unstable under certain circumstances.

Looking at the rulebases produced by the SoFLC (appendix E.2.2), the rulebases are not symmetrical with respect to the absolute value. The absolute values should be the same with respect to the centre position although the sign will be opposite.

At the centre of the rulebase, where there is no heading error and no rate of turn, a zero would be expected; indicating no necessary control output (rudder angle). The rulebases (*see* appendix E) generated by the iterative process in the SoFLC all show similar asymmetrical

settings. From that it can be concluded that the update algorithm used by the SoFLC caused this asymmetric rulebase/ response. Since the manoeuvre was symmetrical ($\pm 20°$) and the environment did not cause drift which could result in an offset, the rulebase was expected not to show a centre offset and to be almost symmetrical with small perturbations.

Investigating the rulebases generated by the PSoFLC, this asymmetry is not found (*see* appendix E.1.2). This is shown by the more symmetric response of the controller (figures 5.15–5.17, page 75), oscillating equally around the desired value.

The PD response shows the following trends: with respect to increased forward speed, a reduction in ISE, a reduction in percentage overshoot, a reduction in rise time and a reduction in rudder activity are seen (table 5.7).

The SoFLC does not show those trends. With respect to increased forward speed, a reduction in ISE, a reduction in rise time and a reduction in rudder activity can be noticed. The criterion percentage overshoot does not follow a trend.

The PSoFLC shows similar qualitative behaviour to the SoFLC. With respect to increased forward speed, a reduction in ISE, a reduction in rise time and a reduction in rudder activity are noticed. The criterion percentage overshoot does not follow a trend.

This is not a statistically validated observation due to the small sample size of only three samples. However, the inconsistency is important to notice as it would otherwise show a reduction in percentage overshoot (table 5.7).

The inconsistency in trends suggests non-linear activity across the whole range. This is in accord with the non-linear nature of the embedded fuzzy logic controller.

The whole working range of the FLC has to be seen as a non-linear range with small linear sections. Those linear sections are the sections when exactly the same two rules are hit.

Chapter 7

Conclusions and Suggestions for Future Work

7.1 Conclusions

The aim of this investigation was to develop a control scheme which is able to adapt itself quickly to changes in the working environment and in plant (process) characteristics. The research objectives, as laid out in section 1.1 to develop such a controller and demonstrate its capabilities have been met. This thesis demonstrates the ability of a novel type of controller (combining fuzzy logic as a controller and a neural network utilised as a model) to evaluate the controller's performance before it is applied. The evaluation is done by predicting future stages.

A literature survey identified the current state of technology. Some aspects relevant to this research undertaken to support this study were highlighted in chapter 2 of the thesis. From the literature, it was evident, that current self-organising techniques such as the SOC of Sugiyama [101] and Yamazaki [125] are well suited for course keeping applications which requires a slow learning. Other techniques such as *Model Reference Adaptive Control* (MRAC) is targeted at purely linear control, *eg* there is only one adaptive parameter, K_D.

The limitations of SoFLC are analysed in chapter 3. The technique of analysing a present state, and identifying the responsible control action (which occurred in the past), and isolating others which occurred since, is a difficult process and other control actions were issued in between, is a difficult process. It is possible that the assumed control action is not the primary

source which caused the state and therefore its influence is not always to be determined.

The Predictive Self-organising Fuzzy Logic Controller as explained in chapter 4 offers a novel approach to overcome the problems as they are experienced with current self-organising techniques. The addition of a *Predictor* module to the self-organising control structure is discussed and its implementation and utilisation in the overall control system is discussed.

Simulated test results are shown in chapter 5. The tests include a $\pm 20°$ step change undertaken without any disturbance present, and a straight line course keeping test with disturbances. The graphs are discussed and analysed. From the results it can be concluded, that the addition of the *Predictor* module significantly improved the learning of the self-organising fuzzy logic controller resulting in a more rapid, more focused learning.

An essential part of this work has been to prove that it is possible, from an empty rulebase of the fuzzy logic controller, to rapidly learn a useful set of steering parameters when the learning is done under controlled conditions. This means that the controller would not require the operator to input specialist knowledge. This removes the need for any installation and set-up expertise to initialise the autopilot system, which is a significant cost saving to the customer. The learnt rulebase and also the neural network model can then be used to continually improve the performance, probably at a slower learning rate to avoid external disturbances causing transient effects.

The PSoFLC demonstrates significant reduction in learning times which will allow the generation of larger databases within acceptable sea-trial duration. This is certainly not the case with slow learning techniques currently used.

This thesis illustrates the advantages of fuzzy logic applied to the steering of small motorised craft which are difficult to steer due to their design. The PD autopilots could not demonstrate a better performance (reduced ISE) over the vast range of working conditions of such craft. Nevertheless, the results shown (chapter 5) are for only the specially tuned PD controller for that vessel. The pilot tuning was a compromise which had to encompass the displacement mode at low speed and the planing mode at high speed. More attention was given to the high speed mode when tuning the PD controller.

Both, neural networks (appendix A) and fuzzy logic (appendix B), are used in the novel

controller design investigated, developed and discussed in this thesis. Fundamental research was not only carried out in the field of combining the two techniques into a novel control strategy but also in the fundamentals of the techniques used.

So far, the classic tuned PD controller has not always been outperformed by the unique algorithm introduced here. Considering the learning curve the controller goes through, it may be concluded that the PSoFLC will perform better in the long term than the classic control algorithm (PD), especially when the environment changes. The PSoFLC shows a considerable potential to improve over a greater variety of conditions.

7.2 Suggested Future Work

The control strategy resulting from the research is not only applicable to autopilot design but also applicable to other areas of process control.

An improvement of the benchmark SoFLC (originally a side-line study only), has been the incorporation of a third dimension to the rulebase, namely a speed variable. This three dimensional controller uses seven fuzzy sets in either input window named as heading error, turn rate and speed.

The flexibility of such general controller designs to expand and improve when provided with additional useful data has been demonstrated. The system effectively becomes an 'expert' system database which can instantly respond to a set of input data with a learnt response. The overhead of adding this third input has proven to be very small as all of the software routines are common for the two dimension system and it is therefore a very attractive way of improving the performance without utilising the predictive module. Almost all modern small vessels that are likely to be fitted with an autopilot will have a source of speed data, either as ground speed from a GPS, or as water speed from an electronic log. Critical to producing the enlarged, three dimensional, database is the ability to accurately and rapidly learn without significant installer expertise. Since this controller is based on the SoFLC algorithm it has similar shortcomings as discussed in chapter 5. An increase in the dimension of the rulebase could allow other sensor data to be considered, such as forward and downward looking sonar. This data could be fed into the controller for use in *eg* in confined, shallow waters, or applied

to the navigation layer where it is analysed and appropriate decision are drawn from it.

At the time of writing this thesis, a three dimensional-rulebase-fuzzy-logic autopilot is being tested by Cetrek Ltd. However no results are yet available. It is believed that this controller will adapt as quickly as a the standard SoFLC (two dimension) with the advantage of not having to re-learn when the speed of the vessel changes. So once a condition is learnt, it will not be forgotten as quickly as it happens with the standard SoFLC. However, the need for re-training can not be completely neglected since the craft can change it's steering characteristics due to other influences such as mass, *eg* when loaded/unloaded.

Current limitations in hardware design do not allow the implementation of the Predictive Self-organising Fuzzy Logic Controller in the production line of Cetrek Ltd. yet. More extensive testing has to follow and hardware has to improve to make this novel control strategy widely available.

It is also possible to replace the current back-propagation neural network used as the internal model with a more sophisticated adaptive model. During this research a neural network was found to be suitable for the predictive controller. More research in this field could lead to an improvement in predictive quality resulting in even more rapid adaptation of the controller to environmental changes (as demonstrated here). Another application could be in the field of reconfigurable control where the whole control strategy has to be modified due to some (unexpected) dramatic change in process characteristics, *ie* loss of one propeller on a twin propelled craft or loss a rudder.

References

[1] C. von Altrock. *Industrial Applications of Fuzzy Logic and Intelligent Systems.*, chapter 14 Fuzzy Logic Application in Europe, pages 227–310. In Yen et al. [126], 1995. ISBN 0-7803-1048-9.

[2] J. van Amerongen. A model reference adaptive autopilot for ships. practical results. In *Proc. of the 8th IFAC Triennial World Congress*, pages 1007–1014, Kyoto - Japan, 1981. IFAC Control Science and Technology.

[3] J. van Amerongen. *Adaptive Steering of Ships: A Model Reference Approach to Improved Manoeuvring and Economical Course-Keeping.* Ph.D. thesis, Delft University of Technology, 1982.

[4] J. van Amerongen and J. C. van Cappelle. Mathematical modelling for rudder roll stabilization. In *Proc. of the 6th Ship Control Symposium (6th SCSS)*, pages 1–10, Ottawa - Canada, 1981.

[5] J. van Amerongen, P. G. M. van der Klugt, and J. B. M. Pieffers. Rudder roll stabilization controller design and experimental results. In *Proc. of the 8th Ship Control Systems Symposium (8th SCSS)*, pages 120–142, The Hague - Holland, 1987.

[6] J. van Amerongen and H. R. van Nauta Lemke. Experiences with a digital model reference adaptive autopilot. In J. Vlietstra, editor, *Ship Operation Automation*, volume III, pages 141–152. IFIP, North–Holland Publishing Company, 1980.

[7] J. van Amerongen, H. R. van Nauta Lemke, and J. C. T. van der Veen. Optimum steering of ships with an adaptive autopilot. In *Proc. of 5th Ship Control Systems Symposium (5th SCSS)*, pages J2 4–1, Maryland - USA, 1978.

[8] J. van Amerongen and A. J. Udink ten Cate. Model reference adaptive autopilots for ships. *Automatica*, 11:441–449, 1975. original version in: Proc. IFAC/ IFIP Symposium on Ship Operation Automation, Oslo, Norway, July 1973.

[9] K. J. Åström and P. Eykhoff. System identification - a survey. *Automatica*, 7:123–162, 1971.

[10] K. J. Åström and B. Wittenmark. On self tuning regulators. *Automatica*, 9:185–199, 1973.

[11] B. A. A. P Balasuriya and P. R. P. Hoole. Feedforward neural network controller for ship steering. In T. I. Fossen, editor, *Workshop on Control Applications in Marine Systems (CAMS'95)*, pages 400–405, Trondheim - Norway, 10–12 May 1995. International Federation of Automatic Control (IFAC).

[12] M. Bech and L. Wagner-Smitt. Analogue simulation of ship manoeuvres. Report Hy-14, Hydro-og Aerodynamisk Laborium, Lyngby, Denmark, September 1969.

[13] M. I. Bech. Some aspects of the stability of automatic course control of ships. *Journal of Mechanical Engineering Science*, 14(7):123–131, 1972.

[14] G. K. Blackwell, J. Rangachari, and C. T. Stockel. Ships that pass in the night. Intelligent management of a fuzzy event system. In *SCS Multiconference*, pages 542–547, Copenhagen - Denmark, 1991.

[15] M. Blanke. *Ship Propulsion Losses related to Automatic Steering and Prime Mover Control.* Ph.D. thesis, Technical University of Denmark, 1981.

[16] M. Blanke and A. C. Christensen. Rudder-roll damping autopilot robustness to sway-yaw-roll couplings. In *Proc. of the 10th Ship Control Symposium (10th SCSS)*, pages 5, 251–265, Ottawa - Canada, 1993.

[17] A. W. Brink, G. E. Baas, A. Tiano, and E. Volta. Adaptive automatic course-keeping control of a supertanker and a containership – a simulation study. *International Shipbuilding Progress*, 26(301):189–214, 1979.

[18] A. W. Brink and A. Tiano. Self-tuning adaptive control of large ships in non-stationary conditions. *Proc. of International Shipbuilding Progress*, 28(323):162–178, 1981.

[19] M. Brown and C. Harris. *Neurofuzzy Adaptive Modelling and Control.* Series in Systems and Control Engineering. Prentice Hall, Prentice Hall International (UK) Limited, Campus 400, Mayland Avenue, Hemel Hempstead, Hertfortshire, HP2 7EZ, UK, 1994. ISBN 0-13-134453-6.

[20] A. W. Browning. *A Mathematical Model to Simulate Small Boat Behaviour.* Ph.D. thesis, School of Design, Engineering and Computing, Electronics Institute, Bournemouth - UK, 1990.

[21] R. S. Burns. *The Automatic Control of Large Ships in Confined Waters.* Ph.D. thesis, School of Manufacturing, Materials and Mechanical Engineering (SMMME), Plymouth Polytechnic, Drake Circus, Plymouth, Devon PL4 8AA, United Kingdom, September 1984.

[22] R. S. Burns. The design, development and implementation of an optimal guidance system for ships in confident waters. In *Proc. of the 9th Ship Control Systems Symposium (9th SCSS)*, volume 3, pages 3.384–3.3401, Bethesda - USA, 1990.

[23] R. S. Burns and R. Richter. The application of neural networks for the control of small and large vessels. In T. I. Fossen, editor, *Workshop on Control Applications in Marine Systems (CAMS'95)*, pages 393–399, Trondheim - Norway, 10–12 May 1995. International Federation of Automatic Control (IFAC).

[24] R. S. Burns, R. Richter, and M. N. Polkinghorne. A multivariable neural network ship mathematical model. In R. S. Burns, editor, *Proc. of the 1st International Conference on Marine Transport into the 21st Century (MarTrans'95)*, pages 747–756, Plymouth - UK, 30 August – 1 September 1995. University of Plymouth & Wessex Institute of Technology, Computational Mechanics Publications, ISBN 1-85312-330-7. Also

published by: Computational Mechanics Publications, Boston MA, 1995, ISBN 1-56252-254-x, pp 747-756.

[25] Cetrek Ltd., 1 Factory Road, Upton, Poole, Dorset, BH16 5SJ - UK. *Pilot 715 and Pilot 730*, user's manual edition, March 1998. Issue 04, Reference: 807300.

[26] S. Chiu, S. Chand, D. Moore, and A. Chaudhary. Fuzzy logic for control of roll and moment for a flexible wing aircraft. *IEEE Control Systems*, 11(4):42–48, June 1991.

[27] J.-L. Choi, Lee D.-I., and Hwang C.-S. A model reference self-organising fuzzy logic controller for auto depth control system of a submersible vehicle. In T. I. Fossen, editor, *Workshop on Control Applications in Marine Systems (CAMS'95)*, pages 417–423, Trondheim - Norway, 10–12 May 1995. International Federation of Automatic Control (IFAC).

[28] D. W. Clarke. Do autopilots save fuel? In *IMechE Conference on the Priorities of Reducing the Fuel Bill*, pages 2–25, 1982.

[29] D. W. Clarke, D. Phil, and P. J. Gawthrop. Self-tuning controller. *IEE Proceedings*, 122 part D(9):929–934, September 1975.

[30] H. Eda. Directional stability and control of ships in waves. *Journal of Ship Research*, pages 205–218, 16 September 1972.

[31] M. Endo, J. van Amerongen, and A. W. P. Bakkers. Applicability of neural networks to ship steering. In *IFAC Workshop on Control Applications in Marine Systems (CAMS'89)*, pages 221–232, Lyngby, 1989. International Federation of Automatic Control (IFAC).

[32] N. A. Fairbairn and M. J. Grimble. H_∞ marine autopilot design for course-keeping and course-changing. In *Proc. of the 9th Ship Control Systems Symposium (9th SCSS)*, volume 3, pages 311–335, Bethesda - USA, 1990.

[33] H. N. Farbrother and B. A. Stacey. Aspects of remotely operated vehicle control – a review. *Underwater Technology*, 19(1):24–36, Spring 1993.

[34] H. N. Farbrother, B. A. Stacey, and R. Sutton. Fuzzy self-organising control of a remotely operated submersible. In *Proc. IEE International Conference Control '91*, pages 499–504, Edinburgh - UK, 1991. IEE.

[35] T. I. Fossen. *Guidance and Control of Ocean Vehicles.* John Wilet & Sons, Chichester, New York, Brisbane, Toronto, January 1994. ISBN 0-471-94113-1.

[36] M. J. Grimble and M. A. Johnson. H_∞ robust control design – a tutorial review. *Computing and Control Engineering Journal*, pages 275–282, November 1991.

[37] M. J. Grimble, M. R. Katebi, and Y. Zhang. H_∞-based ship fin-rudder roll stabilization design. In *Proc. of the 10th Ship Control Symposium (10th SCSS)*, pages 5, 251–265, Ottawa - Canada, 1993.

[38] C. C. Hang and P. C. Parks. Comparitive studies of model reference adaptive control systems. *IEEE Transactions Automation Control*, AC-18:419–428, 1973.

[39] G. Hardier. Recurrent neural networks for ship modeling and control. In T. I. Fossen, editor, *Workshop on Control Applications in Marine Systems (CAMS'95)*, pages 39–62, Trondheim - Norway, 10–12 May 1995. International Federation of Automatic Control (IFAC).

[40] C. J. Harris, editor. *Advances in Intelligent Control.* Taylor and Francis, London, UK, 1994.

[41] G. E. Hearn, P. Sen, and Y. Zhang. Ship motion control by an on-line trained neural controller. In *3rd International Conference Manoeuvring and Control of Marine Craft*, pages 75–88, Southampton - UK, 7–9 September 1994.

[42] G. E. Hearn, Y. Zhang, and P. Sen. Comparison of siso and simo neural control strategies for ship track keeping. *IEE Proceedings - Control Theory and Applications*, 144(2):153–165, March 1997.

[43] S. M. Hodder and D. N. Shields. Application of self-tuning control to ships. In *Proc. Second Int. Conference on Systems Engineering*, pages 187–198, Coventry - UK, 1982.

[44] G. Honderd and J. E. W Winkelman. An adaptive autopilot for ships. In *Proc. of the 3rd Ship Control Systems Symposium (3rd SCSS)*, pages paper 111b–1, Bath - UK, 1972.

[45] K. Hora. On shipboard use of the intelligent ship concept for safe navigation. In *Sixth IMLA Conference o Maritime Education and Training*, pages 10–14, Bremen - Germany, 1990.

[46] Y. Inoue and J. Du. Self-tuning fuzzy control for dynamic positioning system of offshore structures. In T. I. Fossen, editor, *Workshop on Control Applications in Marine Systems (CAMS'95)*, pages 378–384, Trondheim - Norway, 10–12 May 1995. International Federation of Automatic Control (IFAC).

[47] A. Iwata, Y. Nagasaka, S. Kuroyagani, and N. Suzumura. Realtime ECG data compression using dual three layered neural networks for a digital holter monitor. In *Proc. of the 1991 International Conference on Artificial Neural Networks*, pages 1673–1676, Finland, 1991.

[48] J.-S. R. Jang. ANFIS: Adaptive-network-based fuzzy inference systems. *IEEE Transactions on Systems, Man and Cybernetics*, 23:665–685, May 1993.

[49] I. M. Jess. A linguistic self-organising autopilot for yaw control of a warship. MSc thesis, Royal Naval Engineering College Manadon, Plymouth, Devon PL5 3AQ - UK, June 1990.

[50] D. Jiang and X. Jia. Neural networks for course-keeping control. In T. I. Fossen, editor, *Workshop on Control Applications in Marine Systems (CAMS'95)*, pages 405–408, Trondheim - Norway, 10–12 May 1995. International Federation of Automatic Control (IFAC).

[51] C. G. Källström. Control of yaw and roll by a rudder/ fin stabilization system. In *Proc. of the 6th Ship Control Symposium (6th SCSS)*, pages F2 3–1, 3–17, Ottawa - Canada, 1981.

[52] C. G. Källström and K. J. Åström. Adaptive autopilots for steering of large tankers. Technical Report TFRT 3145, Lind Institute of Technology, 1977.

[53] P. J. King, K. J. Burnham, and D. J. G. James. A model reference self-organising fuzzy logic controller. *Control and Computers*, 23(1):6–10, 1995.

[54] P. J. King and E. H. Mamdani. The application of fuzzy control systems to industrial processes. *Automatica*, 13:235–242, 1977.

[55] R. Langari and J. Yen. *Industrial Applications of Fuzzy Logic and Intelligent Systems.*, chapter 1 Introduction to Fuzzy Logic Control, pages 3–39. In Yen et al. [126], 1995. ISBN 0-7803-1048-9.

[56] J. Lawrence. *Neuronale Netze - Computersimulation biologischer Intelligenz.* Systhema Verlag GmbH, Frankfurter Ring 224, D-8000 München 40, Germany, 1992. ISBN 3-89390-271-6.

[57] J. R. Layne and K. M. Passino. Fuzzy model reference learning control for cargo ship steering. *IEEE Control Systems*, 13(6):23–34, December 1993.

[58] G. Lightbody and G. W. Irwin. Direct neural model reference adaptive control. *IEE Proc. Control Theory Applications*, 142(1):31–43, January 1995.

[59] C. C. Lim and W. Forsythe. Autopilot for ship control. *IEE Proceedings*, 130:281–294, 1983.

[60] D. A. Linkens and S. B. Hasnain. Self-organising fuzzy logic controller and application to muscle relaxant anaesthesia. *IEE Proceedings (D)*, 138(3):274–284, May 1991.

[61] D. A. Linkens, M. Mahfouf, and M. Abbod. Self-adaptive and self-organising control applied to nonlinear multivariable anaesthesia: a comparative model-based study. *IEE Proceedings (D)*, 139(4):381–394, July 1992.

[62] A. R. J. M. Lloyd. *Sea Keeping – Ship Behaviour in Rough Weather*, pages 87–123. Ellis Horwood, London - UK, 1988.

[63] R. M. Luke and F. West. An integrated steering system. *Journal of the SNAME (Society of Naval Architects and Marine Engineers)*, June 1960. New England Section.

[64] O. Martienssen. Die Verwendbarkeit des Rotationskompasses als Ersatz des magnetischen Kompasses. *Journal of Physics (German original: Physikalische Zeitschrift)*, 7:535–543, 1906.

[65] W. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. In *Bulletin of Mathematical Biophysics*, volume 5, pages 115–133, 1943.

[66] N. Minorsky. Directional stability of automatically steered bodies. *Journal of the American Society of Naval Engineers*, 34(2):280–309, 1922.

[67] N. Minorsky. Automatic steering tests. *Journal of the American Society of Naval Engineers*, 42:285 ff., 1930.

[68] G. A. Montague, M. J. Willis, M. T. Tham, and A. J. Morris. Artificial neural network based multivariable predictive control. In *Proc. of IEE 2nd International Conference on Artificial Neural Networks*, pages 119–123, Bournemouth - UK, 1991. IEE.

[69] N. Mort. *Autopilot Design for Surface Ship Steering Using Self-Tuning Controller Algorithms.* Ph.D. thesis, University of Sheffield, Sheffield - UK, 1983.

[70] N. Mort and D. A. Likens. Self-tuning controllers for surface ships course and track-keeping. In *Proc. of the Symopsium on Ship Steering Automatic Control*, pages 175–188, Genova - Italy, June 1980.

[71] S. Motora. On the automatic steering and yawing of ships in rough seas. *Journal of SNA of Japan*, pages 61–68, 1953.

[72] National Marine Electronics Association. *Standard for Interfacing Electronic Devices*, NMEA 0183, version 2.00 edition, January 1992.

[73] K. Nomoto. Directional stability of automatically steered ships with particular reference to their bad performance in rough seas. In *Proc. of 1st Symposium on Ship Manoeuvrability*. DTMB, May 1960. DTMB report No. 1461.

[74] K. Nomoto and T. Motoyama. Loss of propulsive power caused by yawing with particular reference to automatic steering. *Japan Shipbuilding and Marine Engineering*, pages 71–80, 1960.

[75] K. Nomoto, T. Taguchi, K. Honda, and S. Hirano. On the steering qualities of ships. *Proc. of the International Shipbuilding Progress*, 4(35):354–370, July 1957.

[76] M. Odom and R. Sharda. A neural network for bankruptcy prediction. *Proc. of IEEE*, 2:163–168, 1990.

[77] K. Ohtsu, M. Horigome, and G. Kitagawa. A new ship's autopilot design through a stochastic model. *Automatica*, 15:255–268, 1979.

[78] J. Oldenburg. Experiment with a new adaptive autopilot intended for controlled turns as well as for straight course steering. In *Proc. of the 4th Ship Control Systems Symposium (4th SCSS)*, volume 1, pages 152–160, Hague, 1975.

[79] J.-J. Østergaard. Fuzzy control of cement kilns – a retrospective summary. In *First European Congress on Fuzzy and Intelligent Technologies in Aachen (EUFIT93)*, pages 552–558, Aachen, Germany, 1993.

[80] Y.-H. Pao. *Adaptive Pattern Recognition and Neural Networks.* Addison-Wesley Publishing Company, Inc., Reading, Menlo Park, New York - USA, 1989.

[81] M. N. Polkinghorne. *Self-organising Fuzzy Logic Autopilot for small Vessels.* Ph.D. thesis, School of Manufacturing, Materials and Mechanical Engineering, University of Plymouth, Drake Circus, Plymouth, Devon PL4 8AA, November 1994.

[82] M. N. Polkinghorne, R. S. Burns, and G. N. Roberts. Initial full scale sea trial results for a self-organising fuzzy logic autopilot. In *Applications of Artificial Intelligence for Technological and Business Processes*, volume I of *Technology Transfer Series*, pages 18–27. University of Plymouth, Plymouth - UK, 1996. ISBN 0-905227-53-0.

[83] T. J. Procyk and E. H. Mamdani. A linguistic self-organising controller. *Automatica*, 15(1):15–30, 1979.

[84] R. Richter and R. S. Burns. An artificial neural network autopilot for small vessels. In R. Poley and R. Zobel, editors, *Proc. of the First Conference of the UKSS'93*, pages 168–172, Keswick Hotel, Keswick, Cumbria - UK, 13–15 September 1993. United Kingdom Simulation Society. ISBN 0-9516509-1-2.

[85] R. Richter, R. S. Burns, and M. N. Polkinghorne. Application of an artificial neural network to model complex plant behaviour. In *IFAC '95 - Workshop on Motion Control, Munich - Germany*. International Federation of Automatic Control (IFAC), 9–11 October 1995.

[86] R. Richter, R. S. Burns, M. N. Polkinghorne, and P. Nurse. Modelling and control of surface ships by employing a fuzzy-neural solution. In *11th International Conference on Systems Engineering (ICSE '96)*, pages 7–12, Howard R. Huges College of Engineering, University of Nevada - Las Vegas, Box 454005, Las Vegas, NV 89154-4005, USA, 9–11 July 1996. University Nevada - Las Vegas.

[87] G. N. Roberts. Ship roll damping using rudder and stabilising fins. In A. Tiano, editor, *Proc. on Control Applications in Marine Systems (CAMS'92)*, pages 129–138, Genova - Italy, 8–10 April 1992.

[88] R. Rojas. *Theorie der neuronalen Netze - Eine systematische Einführung*. Springer-Verlag, Berlin, Heidelberg, New York, 1993. ISBN 3-540-56253-9.

[89] F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–408, 1958. reprinted in Anderson and Rosenfeld 1988.

[90] D. Rumelhart and J. McClelland. *Parallel Distributed Processing*. MIT Press, Cambridge, Massachusetts, 1986.

[91] L. J. Rydill. A linear theory for the steered motion of ships in waves. *Transactions of RINA (Royal Institute of Naval Architects)*, 101(1):81–112, 1959.

[92] J. Saint-Donat, N. Bhat, and T. J. McAvoy. *Neural Net Based Model Predictive Control.*, part 1, chapter 8, pages 52–58. In Harris [40], 1994.

[93] L. Schiff and M. Gimprich. Automatic control of ships by proportional control. In *Transactions of the Society of Naval Architects and Marine Engineers*, volume 57, pages 94–124, 1949.

[94] E. Schöneburg. Stock price prediction using neural networks: A project report. In *Neurocomputing*, volume 2, pages 17–27. Elsevier Science Publishing, 1990.

[95] M. Schuler. The disturbance of pendulum and gyroscopic apparatus by the acceleration of the vehicle. *Navigation, Journal of The Institute of Navigation*, 14(1):26–32, Spring 1967. original version in 'Journal of Physics', Vol. XXIV, July 1923.

[96] P. Sen, Y. Zhang, and G. E. Hearn. Adaptive process control in ship manoeuvring by neural networks. In *Proc. of ACEDC'94*, pages 28–36, Plymouth - UK, 1994. University of Plymouth. ISBN 0-905227-33-6.

[97] R. Simensen. Simulation of an artificial neural network for ship steering control. In *Proc. of the 2nd Conference of the United Kingdom Simulation Society (UKSS'95)*, pages 65–72, Scotland, 1995. UKSS.

[98] E. Sperry. Automatic steering. In *Proc. of the 30th General Meeting of the Society of Naval Architects and Marine Engineers*, volume 57, pages 53–61, New York, 1922.

[99] R. I. Stephens, K. J. Burnham, and P. J. Reeve. A practical approach to the design of fuzzy controllers with application to dynamic ship positioning. In T. I. Fossen, editor, *Workshop on Control Applications in Marine Systems (CAMS'95)*, pages 370–377, Trondheim - Norway, 10–12 May 1995. International Federation of Automatic Control (IFAC).

[100] C. Stevens. Die Nervenzelle. In *Gehirn und Nervensystem*, pages 2–13, 1988.

[101] K. Sugiyama. *Analysis and Synthesis of the Rule-Based Self-Organising Controller.* Ph.D. thesis, Dept. of Electrical and Electronic Engineering, Queen Mary College, University of London, Mile End Road, London E1 4NS, United Kingdom, March 1986.

[102] K. Sugiyama. Rule-based self-organising controller. In M. M. Gapta and T. Yamakawa, editors, *Fuzzy Computing*, pages 341–353. Elsevier Science Publishers BV, North Holland, 1988.

[103] R. Sutton. *Fuzzy Set Models of the Helmsman Steering a Ship in Course-keeping and Course-changing Modes.* Ph.D. thesis, Dept. of Mechanical and Manufacturing Systems Engineering, University of Wales, Institute of Science and Technology, Cardiff - UK, April 1987.

[104] R. Sutton, P. J. Craven, and R. S. Burns. Intelligent steering control of an autonomous underwater vehicle. *Journal of Navigation*, 2000. unpublished, excepted for publication.

[105] R. Sutton and I. M. Jess. A design study of a self-organising fuzzy autopilot for ship control. *Journal of Systems and Control Engineering*, 205:35–47, 1991.

[106] R. Sutton and I. M. Jess. Real–time application of a self-organising autopilot to warship yaw control. In *Proc. IEE International Conference Control '91*, pages 827–832, Edinburgh - UK, 1991. IEE.

[107] R. Sutton and A. Pringle. A fuzzy hybride autopilot for a very large crude carrier. In R. S. Burns, editor, *Proc. of the 1st International Conference on Marine Transport into the 21st Century (MarTrans'95)*, pages 731–737, Plymouth - UK, 30 August – 1 September 1995. University of Plymouth & Wessex Institute of Technology, Computational Mechanics Publications, ISBN 1-85312-330-7. Also published by: Computational Mechanics Publications, Boston MA, 1995, ISBN 1-56252-254-x, pp. 747-756.

[108] R. Sutton, G. N. Roberts, and P. J. S Fowler. The scope and limitations of a self-organising fuzzy controller for warship roll stabilisation. In *Proc. of the International Conference on Modelling and Control of Marine Craft*, pages 88–105, Exeter - UK, April 1990.

[109] R. Sutton and S. D. H. Taylor. Neuro-fuzzy techniques applied to a ship autopilot design. *Journal of Navigation*, 49(3):410–430, September 1996.

[110] A. Tiano, N. Mort, D. A. Derradji, M. Cuneo, A. Ranzi, and W. W. Zhou. Rudder roll stabilisation by neural network control. In G. N. Roberts and M. M. A. Porzanjani, editors, *Proc. of 3rd International Conference Manoeuvring and Control of Marine Craft (MCMC'94)*, pages 33–44, Southampton - UK, 7-9 September 1994.

[111] H. Tolle and E. Ersü. Neurocontrol: Learning control systems inspired by neuronal architectures and human problem solving. Lecture Notes in Control and Information Science, 1992. 172, Springer-Verlag, Berlin, Germany.

[112] H. Tolle, P. C. Parks, E. Ersü, M. Hormel, and J. Militzer. *Learning Control With Interpolating Memories.*, part 1, chapter 5, pages 113–139. In Harris [40], 1994.

[113] M. Tomera and L. Morawski. Neural-network-based fuzzy logic marine autopilot. In *Proc. of the 3rd International Symposium on Methods and Models in Automation and Robotics*, pages 1207–1212, Miedzyzdroje, Poland, September 1996.

[114] A. Vahedipour, O. J. Flower, and M. M. A. Pourzanjani. Pseudo derivative feedback design study of an autopilot with reference to non-linearities and time delay effects. In *Proc. of the International Conference on Modelling and Control of Marine Craft*, pages 88–105, Exeter - UK, April 1990.

[115] P. Vega, C. Prada, and V. Aleixandre. Self-tuning predictive PID controller. *IEE Proceedings (D)*, 138(3):303–311, May 1991.

[116] M. I. Waldock, G. N. Roberts, and R. Sutton. Artifical neural network control of a bottom profiling unmanned unterwater vehicle. In T. I. Fossen, editor, *Workshop on Control Applications in Marine Systems (CAMS'95)*, pages 409–416, Trondheim - Norway, 10–12 May 1995. International Federation of Automatic Control (IFAC).

[117] A. S. Weigend, B. A. Huberman, and D. E. Rumelhart. Predicting the future: A connectionist approach. *International Journal of Neural Systems*, 1(3):193–209, 1990.

[118] N. Wiener. *Cybernetics*. MIT Press, Cambridge, Massachusetts, 1948.

[119] D. Winwood. Cetrek ltd. private conversation, 7 October 2000. 1 Factory Road, Upton, Poole, Dorset, BH16 5SJ - UK.

[120] N. A. J. Witt. Ship guidance using neural networks. In *Applications of Artificial Intelligence for Technological and Business Processes.*, volume 1 of *Technology Transfer Series*, pages 41–49. University of Plymouth, Plymouth - UK, 1996. ISBN 0-905227-53-0.

[121] N. A. J. Witt and K. M. Miller. A neural-network autopilot for ship control. In *Proc. of the International Conference on Maritime Communications and Control*, London, 1993. paper 4, 13 pages.

[122] N. A. J. Witt, R Sutton, and K. M. Miller. Recent technological advances in the control and guidance of ships. *Journal of Navigation*, 47(2):236–258, May 1994.

[123] N. A. J. Witt, R Sutton, and K. M. Miller. A track keeping neural network controller for ship guidance. In T. I. Fossen, editor, *Workshop on Control Applications in Marine Systems (CAMS'95)*, pages 385–392, Trondheim - Norway, 10–12 May 1995. International Federation of Automatic Control (IFAC).

[124] H. Yamato, T. Koyama, and T Nakagawa. Automatic berthing using the expert system. In A. Tiano, editor, *Proc. on Control Applications in Marine Systems (CAMS'92)*, pages 173–183, Genova - Italy, 8–10 April 1992.

[125] T. Yamazaki. *An Improved Algorithm for a Self-Organising Controller, and its Experimental Analysis.* Ph.D. thesis, Dept. of Electrical and Electronic Engineering, Queen Mary College, University of London, Mile End Road, London E1 4NS, United Kingdom, September 1982.

[126] J. Yen, R. Langari, and L. A. Zadeh, editors. *Industrial Applications of Fuzzy Logic and Intelligent Systems.* IEEE Press, PO Box 1331, 445 Hoes Lane, Piscataway, NJ 08855-1331, USA, 1995. ISBN 0-7803-1048-9.

[127] L. A. Zadeh. Fuzzy sets. *Information and Control*, 8:338–353, 1965.

[128] L. A. Zadeh. Outline of a new approach to analysis of complex systems and decision processes. *IEEE Transactions on Systems, Man and Cybernetics*, SMC-3:28–44, 1973.

[129] B. S. Zhang and J. M. Edmunds. Self-organising fuzzy logic controller. *IEE Proceedings (D)*, 139(5):460–464, September 1992.

[130] Y. Zhang, G. E. Hearn, and P. Sen. An on-line trained adaptive neural controller. *IEEE Control Systems Magazine*, 15(5):67–75, October 1995.

[131] Y. Zhang, G. E. Hearn, and P. Sen. A neural network approach to ship track-keeping control. *IEEE Journal of Oceanic Engineering*, 21(4):513–527, October 1996.

[132] Y. Zhang, G. E. Hearn, and P. Sen. Neural network approach to a class of ship control problems. In P. A. Wilson, editor, *Proc. of the 11th Ship Control Systems Symposium (11th SCSS)*, volume 1, pages 115–150, Computational Mechanics Publications, Ashurst Lodge, Ashurst, Southampton, SO40 7AA - UK, 14–18 April 1997. University of Southampton, Computational Mechanics Publications.

[133] Y. Zhang, P. Sen, and G. E. Hearn. A multivariable neural controller for automatic ship berthing. *IEEE Control Systems Magazine*, 17(4):31–45, August 1997.

[134] R. W. Zibikowski. *Recurrent neural Networks: Some Control Aspects*. Ph.D thesis, Faculty of Engineering, Glasgow - United Kingdom, November 1994.

Index

Appendices

Appendix A

Neural Networks: Theory

This chapter gives an introduction into the theory behind *neural networks*. Their ability to learn has fascinated biologists and computer scientists for the last half century. The first sections will explain the biological basics and how they are modelled in the computer. Section A.4 concentrates on the popular back-propagation learning algorithm as used in the autopilot's prediction module.

A.1 History and Introduction

Neural networks were first studied by neuro-biologists in an attempt to emulate some of the processes of pattern recognition that occur in the human brain - *see* McCulloch and Pitts [65], Wiener [118], and Rosenblatt [89].

However, it soon became clear, that neural networks had many other technological applications and this is now considered in the literature on their application to a wide range of problems including the general field of control engineering.

Before considering the neural network models, it is perhaps useful to briefly describe the biological neural network which spawned their artificial counterparts.

Serious investigations started in 1942, by the leading neuro-biologist McCulloch and the statistician Pitt [65]. The paper *A Logical Calculus of Ideas Imminent in Nervous Activity* [65] tangents fields such as digital computing, *electronic brains* and macroscopic intelligence.

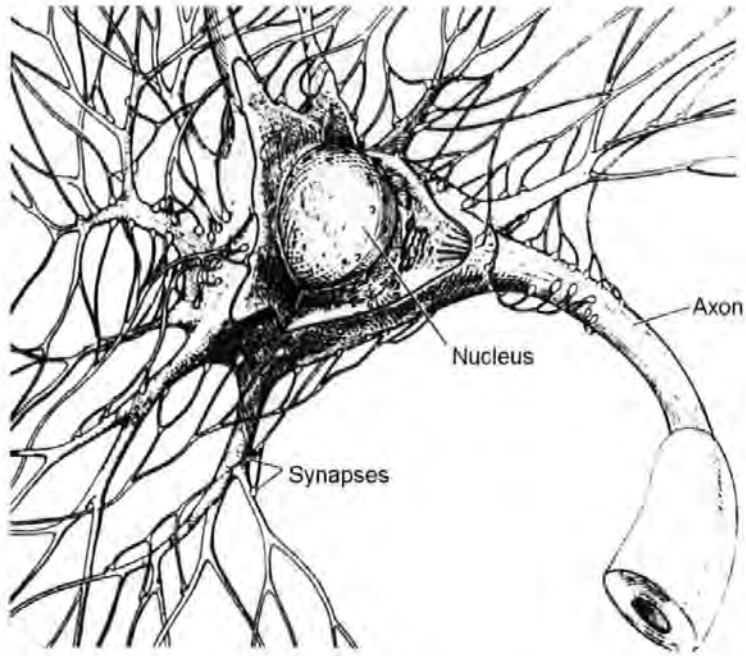


Figure A.1. Structure of a Biological Neuron [100]

A.2 The Neuron

A.2.1 The Biological Structure

Artificial neural networks, such as computer programs, try to adapt biological neural networks as information processing units. To understand artificial neural networks, the biological principles have to be understood.

A *neuron* as seen in figure A.1 is the basic element of the brain. The inter-connection of a very large number (tens of billions) of these processing units form the neural network. The transmission of the signals between the cells is chemical in nature. Each neuron receives signals from other neurons. The signals are electrical impulses. Depending on the excitement of a neuron, the frequency of it's output signal changes. This is called the firing rate of the neuron. The link between the neurons is called *axon*. An axon can be attached to more than one neuron. Furthermore, each neuron possesses several incoming ports for the connection of an axon.

This port is called *dendrite*. The link between axon and dendrite is a chemical fluid which can be understood as a variable resistor which changes the strength of the incoming signals. The region, where all three items work together is called *synapse* (*see* figure A.2).

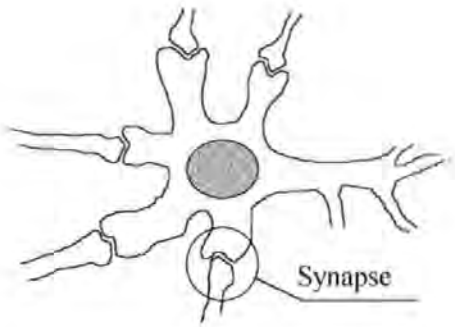


Figure A.2. Layout of a Synapse

During learning, the efficiency of the synapse is changed. The more this path is used, the less resistant the fluid becomes. If the port is not used at all, the connection can break up completely and die.

If the sum of the incoming signals over a defined time is bigger than the threshold of that neuron the neuron will fire. The output of a biological cell is an electrical impulse. The neuron firing rate increases with increasing excitation.

A.2.2 The McCulloch-Pitts Cell

In 1942, McCulloch and Pitts suggested a model for a network which was built using very simple neurons. This neuron is called a McCulloch-Pitts cell and the network resulting from the inter-connection of these cells is called a McCulloch-Pitts network. The McCulloch-Pitts cell is one of the simplest in structure and functionality, because it uses exclusively binary signals. The output of each cell is either 1 or 0 and the incoming signals can only be 1 or 0. Furthermore, these networks have supporting and hampering connections. If there is at least one hampering signal going into a cell, the output of that cell will be zero. If no hampering signal occurs, and the number of supporting incoming signals is greater than a predefined threshold, then a McCulloch-Pitts cell outputs a 1, otherwise the output will also be zero.

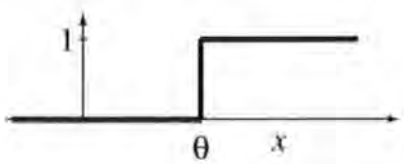


Figure A.3. Step Function

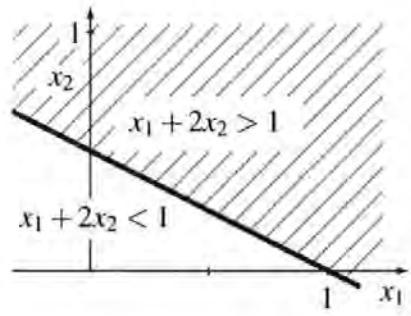


Figure A.4. Geometric Interpretation of the Input Space divided by a Perceptron

Basically, if the number n of incoming supporting connections is less than the threshold θ, then this cell will never fire.

$$output = \begin{cases} 0/1 & \begin{cases} 1 & \sum x_i > \theta \\ 0 & \sum x_i \leq \theta \end{cases} \quad \text{no hampering signal} \\ 0 & \text{at least one hampering signal} \end{cases} \tag{A.1}$$

A.2.3 The Perceptron

Because of the many disadvantages of early models of neurons, Rosenblatt, an American psychologist, developed a different model – the *perceptron*. The main criteria of the perceptron is that it only takes weighted inputs.

A simple perceptron is a McCulloch-Pitts cell which uses weighted inputs to calculate the total input of that cell. Consider n inputs, $w_1, \ldots, w_n$ weights and a threshold θ, the neuron will fire (output = 1) only if $\sum w_i x_i \geq \theta$ otherwise the output will be 0. Thus it is possible to emphasise some signals, such that there are more and less important signals going into the cell.

$$output = \begin{cases} 1 & \sum w_i x_i \geq \theta \\ 0 & \text{otherwise} \end{cases} \tag{A.2}$$

The geometrical interpretation (figure A.4) of the perceptron's function can be formulated as follows: the perceptron divides the input space into two regions. Region one only contains data where the neuron fires (output = 1) and the other region contains data which result in

zero output. This linear approach does not represent the actual behaviour of a biological neuron. It was found, that the neuron's firing rate is limited. To correct this other transfer functions are employed.

A.2.4 Neuron with Steady Transfer Function

From the above sections, the mathematical algorithm can be extracted: the incoming signal is defined as x, the outgoing signal as y and the synapse carries the symbol w. Before the neuron is activated, all incoming signals are summed up to form the total input I. This value is the parameter of the transfer function utilised by the neuron to calculated the output value y which represents the firing rate of the biological neuron. The structure of an artificial neuron is shown in figure A.5.

Possible transfer functions and all the mathematics of a single neuron have been summarised in figure A.6. When implementing a neural network on a computer, is is possible to use different transfers function in the neurons. It is common practice to use alternative functions in the layers. Often the sigmoid function is used in the hidden layer(s) and a linear expression for the neurons in the output layer. More on layers and the organising of neurons follows in the sections below.

A.3 The Inter-connection

To simulate the behaviour of the human brain a network of neurons is needed, a so called neural network (net). The neurons are usually organised into groups called layers. A neural

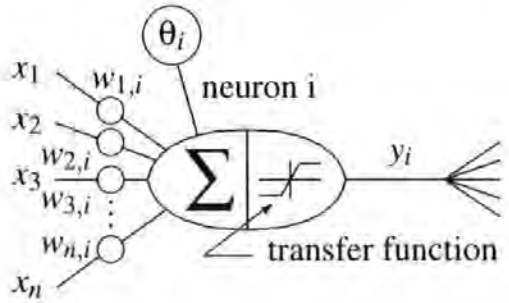


Figure A.5. Main Structure of an Artificial Neuron

$$I_i^l = \sum_{j=1} x_{j,i}^{l-1} \cdot w_{j,i}^l + \theta_i^l \tag{A.3}$$

θ_i^l ... the threshold, which moves the transfer function (graph) in the horizontal direction
x_j^{l-1} ... output of neuron j in the previous layer
$w_{j,i}^l$ ... weight between neuron i in layer l and the neuron j in layer l-1
I_i^l ... total input of neuron i in layer l

where (transfer function) could be:

linear: $f(I_i^l) = I_i^l$ (A.4)

Sigmoid function: $f(I_i^l) = \frac{1}{1+e^{I_i^l}}$ (A.5)

hyperbolic tangent: $f(I_i^l) = \tanh I_i^l$ (A.6)

hard limiter / threshold function: (perceptron) $f(I_i^l) = \begin{cases} -1 & I_i^l \leq 0 \\ +1 & I_i^l > 0 \end{cases}$ (A.7)

Figure A.6. Mathematics Preliminaries of a Neuron

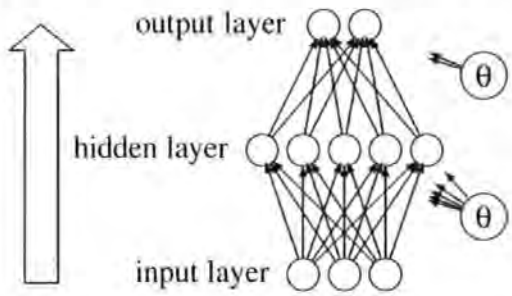


Figure A.7. Single-Layer Network

network consists of at least an input and an output layer and eventually hidden layer(s). The terms *single* and *multi* refer to the number of hidden layers in the network A *single-layer* network consists of three layers, one input and one output layer and a single hidden layer. Simple tasks can be solved by a single layer network but for difficult problems *multi-layer* networks are needed. The main structure of a single-layer network is shown below (figure A.7).

The inter-connection between the neurons in different layers can be seen in figure A.7.

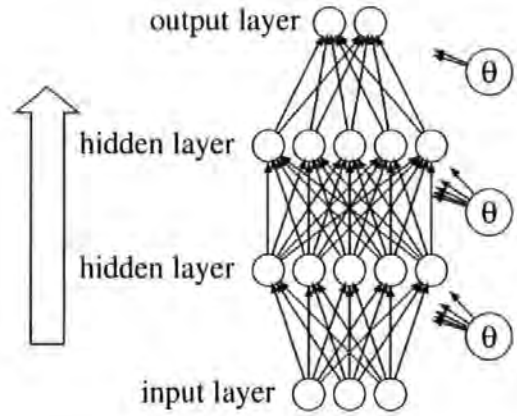


Figure A.8. Multi-Layer Network

It is not necessary to have a network where the connections are only between neurons of different layers, but it is easier to understand and to design a network in this way. The majority of neural networks are organised in this way. Some tasks do not require hidden layers. The number of hidden layers and the number of neurons in each hidden layer are free to be defined and will determine the performance of the network in terms of speed and quality. For most tasks a single-layer network is sufficient.

The behaviour of a multi-layer network (*see* figure A.8) in general is not very different to a single-layer network. The user has to find the optimum in size and structure to be satisfied with the results and the speed. A small network is faster but if the task is too difficult, important information may be lost. Conversely, if the network is too large, the output can be noisy and the computing speed, especially during the learning, is slow.

The order of the system should be reflected by the degree of freedom of the network. Degree of freedom refers here to the number of variable parameters in the system. In terms of neural networks, the degree of freedom is the number of all variable weights on the path from one neuron to another.

Sometimes it is possible to improve the learning quality of the network by re-arranging the neurons into more hidden layers but less neurons in each. As yet, there seems to be no fixed or rigourous guidelines for the construction of a network. Experience of the user and the character of the problem will determine the successful application.

A.4 Learning with Back-propagation

The back-propagation is probably the best known and analysed learning algorithm for neural networks.

A.4.1 General Facts

The two main tasks of a brain – learning and recall – are the most interesting. Learning itself is the process of calibration of the synaptic efficiency, or in the words of artificial networks, the weights. Using this principle some models of neurons and their connections have been investigated, *ie* single-layer networks, multi-layer networks and self-organising networks. The networks can be classified into three groups, depending upon the learning principle, *eg* supervised learning, learning with critic, and one group unsupervised learning (self-organising network). The latter is utilised to obtain relationships between the input and the output vector by the creation of an iterative process without a teacher (as in supervised learning) and also without evaluative values (as learning with critic).

Clustering is the process of arranging data into groups. The network tries to find characteristics in data, and then sorting the data into one of the clusters. In other words, it is known that given data contains items out of n groups. The network has to find the borders between the data to place them in one of the groups.

The results of the student (the network) can be only as good as the training data of the teacher/ supervisor. For supervised learning a vector of input data and one vector of the desired outputs which is associated to the input vector is needed. Clearly, one problem, besides the program for learning, is to have good sets of training data. A set of training data is considered to be a pair of vectors containing input/ desired output data each.

A.4.2 The Derivation of the Learning Algorithm

In this section, a method of supervised learning is discussed and together with the way this may be used in order to develop a 'learning controller' for the steering of a small craft.

Rumelhart's and McClelland's [90] contributions to neural networks are fundamental to further investigation. One way to use the supervised learning is by using the back-

propagation algorithm. The neural network used in this algorithm is a multi-layer perceptron network, and the transfer function used must be steady at each point, *eg* the sigmoid. The back-propagation rule needs the error between computed output by the network (straightforward or phase 1) and the desired output given by the teacher. To adjust the weights on the path between one neuron and the next neuron, the error is back propagated, starting with the output layer via the hidden back to the input layer. This process is the second stage, or the learning phase. The process – computing forward and error propagation backwards – is repeated with different pairs of training data, until a maximum number of epochs is reached or the maximal error approaches a preset value, *ie* $\varepsilon = 0.05$. One *epoch* is the process of passing the data from the input layer to the output layer to obtain the 'actual' output vector, calculating and back-propagating the error between actual and desired values, and the adaptation of the weights inside the neural network. The interesting feature of back-propagation is that no knowledge about the process is required, but a good teacher will have to provide sufficient data to cover the entire operation envelope of the system considered. The disadvantage of this method is that the student does not have any self-learning capabilities and therefore cannot respond better than the teacher. In addition, data from a teacher is required, which is not always available.

So far only the forward or recall phase of the network has been discussed. However to store information the network must be taught. In order to understand the simplest type of learning algorithm, the back-propagation algorithm is used as an example. This technique is based on using the steepest-descent method (gradient method) to minimise the error. This can be seen in equation A.8.

$$E_i^l = \frac{1}{2}(d_i^l - y_i^l)^2 \tag{A.8}$$

d ... desired output l ... layer
E ... error i ... index of neuron in layer
y ... actual (computed) output of the neuron

Considering equation A.8, it can be seen that the actual output is a function of the weights as well as of other parameters. The task is to define a Δw which reduces the error of equation A.8.

Equation A.9 shows the correct mathematical expression with η as a proportional factor

representing the *learning rate*.

$$\Delta w = -\eta \frac{\partial E}{\partial w} \tag{A.9}$$

With y as the computed output resulting from the equations A.5 in figure A.6 and the *total input* into that neuron equation A.9 becomes:

$$\frac{\partial E}{\partial w} = \frac{\partial E}{\partial I} \frac{\partial I}{\partial w} \tag{A.10}$$

$$\frac{\partial I}{\partial w} = \frac{\partial}{\partial w} \sum wy \tag{A.11}$$

$$\frac{\partial}{\partial w} \sum wy = y \tag{A.12}$$

Defining

$$\delta = \frac{\partial E}{\partial I} \tag{A.13}$$

equation A.9 can be rewritten as:

$$\Delta w = -\eta \delta y \tag{A.14}$$

This gives us a corrective value by which the weight has to be adjusted to reduce the error defined in equation A.8.

An extensive mathematical explanation and the complete derivation of the back-propagation learning algorithm can be found in the literature, *eg* [80, 40].

A.4.3 Summary of Theory

The equations for a back-propagation neural network are summarised and are shown in figure A.9.

$x_i^l = Sigmoid(I_i^l)$	straightforward (phase 1)(A1.1)
$I_i^l = \sum_j w_{j,i}^l \cdot x_j^{l-1} + \theta_i^l$	(A1.2)
$\tilde{w}_{j,i}^l = w_{j,i}^l + \Delta w_{j,i}^l$	(A2.1)
$\Delta w_{j,i}^l = \eta \delta_j^l x_j^{l-1}$	back-propagation (phase 2) (A2.2)
$\delta_j^l = x_i^l(1-x_i^l) \cdot (d_i - x_i^l)$	for output layer neurons (A2.3)
$\delta_j^l = x_i^l(1-x_i^l) \cdot \sum_k \delta_k^{l+1} w_{j,k}^{l+1}$	for inner neurons (A2.4)

η ... learning rate
δ_i^l ... error of neuron i in layer l
I_i^l ... total input of neuron i in layer l
$\Delta w_{j,i}^l$ ... weight increment for $w_{j,i}^l$
x_i^l ... output of neuron i in layer l
$w_{j,i}^l$ ... weight on path from neuron j in layer $l-1$ to neuron i in layer l
Sigmoid() ... transfer function (*see* figure A.6)

Figure A.9. The Learning Equations

A.5 Applications of Neural Networks

The multi-layer back-propagation networks are probably the most common structures used to tackle problems found in industrial applications. They can be used in robotics for pattern recognition, such as speech and image recognition, as well as for encrypting applications where the main task is to associate consequences with specific facts. An important factor is that the data fed to the network has to contain *all the relevant* information to the problem and also that the network should be large enough - contain a sufficient number of neurons and associated weights. Finding *proper* data is not trivial and the emphasise is on relevant data. It is well known, that neural networks can interpolate between points and find relationships in the data. The degree of freedom of the system should match the degree of freedom of the network, otherwise the network will not be able to identify and rebuild existing relationships between the facts and the consequences. The developer has to find a compromise between demanded accuracy and learning effort. The larger the neural network is the longer it takes to teach it. A further problem has to be considered which results in a unusable neural network *ie over fitting*. This will occur when too much data is presented to the network but the data does not contain enough significant information and the degree of freedom of the network

is too great. The network will then learn the individual 'fingerprint' of the training sets and not interpolate between these. The relationship between input and output vectors is lost. It also can happen, that noise within the data becomes very important and is interpreted as data rather than filtered out.

Neural networks have become an important tool in many areas of industry and business. They are present in applications such as medicine [47], finance [76], marketing, insurance *eg* in risk assessment, quality control and engineering.

A.5.1 Application of a Neural Networks for Data Encryption and Compression

Neural networks can also be used for data compression [47]. The compression code is found by inserting a bottleneck into the network structure. Consider the example of 8 data lines which can carry signals where only one has a high input and the others are low. The network has simply to reconstruct the input layer on its output layer and the bottleneck, the hidden layer, contains the data in a compressed format.

The structure of the example network is 8 inputs, 3 neurons in the only hidden layer and 8 output neurons. Considering only binary data for the output of the 3 hidden neurons, eight states can be thus encoded by the network. The network structure is shown in figure A.10.

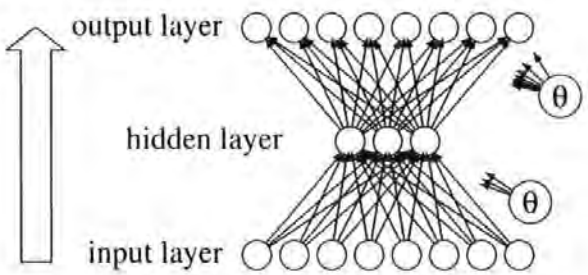


Figure A.10. Network structure to find a binary code

A.6 Using Neural Networks for System Identification

There are many applications in industry which use the capturing capabilities of neural networks. Neural networks can be found in image processing and analysis as well as in machine monitoring. The network is well able to identify situations (patterns of input signals) and draw conclusions which can otherwise only be made by experts. The expert acts as a teacher in the first place and the neural network then copies this knowledge with its own understanding. For system identification mostly back-propagation networks are used but, Hopfield networks and self-organising Kohonen maps are found as well.

Nowadays, neural networks are employed in business finance *eg* to predict share states. Schöneburg [94], did some investigation in this field by applying such networks. However, much more research is needed in this area as he found there were considerable limitations. Although, the exact relationship does not need to be known, it is important to know the parameters which influence a decision. The neural network will pick up the relationship in most circumstances providing important values are not suppressed.

Neural networks are important techniques when used for image analysis. They have been successfully implemented into software for optical character recognition (OCR). As the name indicates, the scanned image, which is in binary format, is checked for the appearance of text. Those areas with text are analysed and the characters extracted. The image, hopefully, containing only one character is passed to a pre-trained neural network, containing all the possible characters of a font. The network will then come up with a possible match. Depending on the strength of the signal (indicating the likelihood of a positive identification) the character is then passed on or questioned. It is possible, that all characters are passed on to form a word which is then checked against a list of possibilities, and which can incorporate a spell check program.

Many more applications using neural networks are being developed in the field of speech recognition. They use a similar process to those used inference being the input signal, where it represents a frequency spectrum related to time. Here sound is digitised, divided into small windows and the numbers stored in a vector before being passed to a neural network for analysis.

This is a vast field and one which will involve future intensive studies in complex systems.

A.7 Using Neural Networks for Control

Numerous applications have been applied in the field of control systems. Research by Zhang *et al* [132] considers the application of neural networks for ship position and orientation control where heading errors as well as positional errors are minimised during travel and berthing. Feeding measured information into the network and assuming certain ship characteristics, the error can be back propagated through the plant and the network, which acts as the autopilot and optimises the control performance.

The ability of neural networks to assimilate, mix, compress and recall data has been initially investigated by Richter and Burns [84]. This paper reflects the application of combining three specially tuned PID autopilots for a marine craft into a single module. The PID autopilots are fine tuned to work in only one sea state. All the data is then fed into the neural network to combine the 'knowledge' of all PID controllers and this new controller therefore inherits the high performance of each PID autopilot without the need of adjusting parameters when a change of the sea state occurs.

A.8 Neural Networks Summary

This section covered some aspects of neural networks and their application in science and industry. The neuron models, including the perceptron, from the early and mid 1940's [65] have been fundamental to existing networks. These simple processing units as described biologically and mathematically in this chapter form powerful tools when interconnected and help to solve complex tasks of the modern world.

Considering the back-propagation learning algorithm, the neurons are organised in layers and only neurons between neighbouring layers are connected. Simple tasks can be solved by single-layer networks whereas more complex problems require a greater degree of freedom which results in an increase in the number of neurons and their organisation in layers. Multi-layer networks, networks with more then one hidden layer, are used for more difficult and complex tasks.

The back-propagation learning algorithm has been discussed and the mathematical derivation was also shown. The final equations of the learning are summarised to a generic algo-

rithm which can easily be implemented into program code.

Existing applications demonstrate the wide acceptance of neural networks amongst researchers and engineers in development laboratories, academic institutions, and most importantly, in industry and high street products.

Appendix B

Fuzzy Logic: Theory

B.1 Introduction and History

Fuzzy logic is a more general case of the classical Boolean logic. The classical logic is a subset of the fuzzy logic. Basically the classical Boolean logic is fuzzy logic using step functions to describe the degree of membership of a value to a set. In the mid 1960's, L. Zadeh [127] developed the modern fuzzy logic. His intention was to model problems which contained a degree of fuzziness within the data or even within the rules used to make a decision. The values accepted are not only 0 and 1 as known from the classical logic, but also all values in between. Therefore a few definitions have to be made as well as new operators introduced and explained.

It is important to mention the differences between probability theory and the theory of fuzzy logic. As Zadeh [128] formulates it, the membership function $\mu(.)$ defines the possibility of an value x being an element of a fuzzy set X_k ($x \in X_k$). Considering this, the main difference between probability theory and fuzzy logic is that fuzzy logic (possibility) deals with imprecise data of events, whereas probability theory deals with the randomness of occurring (or not occurring) events [103]. The uncertainty of an event happening or not is understood as randomness. The imprecision of fuzzy sets, however, considers the degree membership of an element to a set (a fuzzy set) with imprecise, non crisp, boundaries. Another view may be given by "IT **IS** WARM"; fuzzy statements are not imprecise about the event in question (IT **IS**) but refer to the quantity in a vague manner (WARM).

Implied to fuzzy logic is the fuzzy set theory which will be explained in outline later

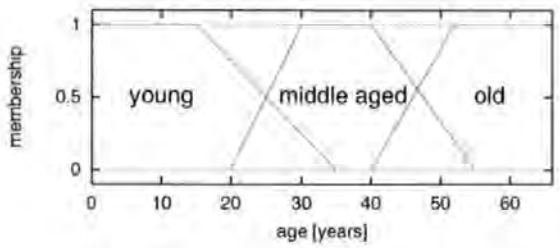


Figure B.1. Fuzzy Sets of Different Age Groups of People

in this chapter. Expressions like fuzzy set, fuzzy operator, fuzzy rules, fuzzification and defuzzification will be discussed here. Examples demonstrate the use of this kind of logic and the utilisation of those principles for control tasks.

The big advantage of using fuzzy logic for control tasks is that it can easily cope with linguistic variables. The human brain is not very good in 'visualising' numbers. It is much easier to visualise terms like tall, short, hot, cold *etc*. But the problem with linguistic variables is the precision they are used with, *eg* tall is not equal 185 cm it is more like about 180 cm. So these terms have to be transferred into 'crisp' numbers before being used in calculations. Fuzzy logic allows using the spoken language (linguistic variables) to define rules and algorithms for control purposes.

B.2 The Main Principle

B.2.1 The Fuzzy Set

The main difference to the Boolean logic can be seen in an additional function called membership function. This is a measure how much a value belongs to a set. In the classical logic, such a function does not exist. Either a value belongs to a set or not. But what will happen if the border becomes unsharp, fuzzy, not just yes or no? The best way to illustrate this is by using an example.

Example: Age of a Person

Fuzzy sets can easily be obtained from a survey. A typical question asked could be: what is understood by a 'young' person? Figure B.1 shows a fuzzy window containing three fuzzy

sets for the variable age. The sets are named young, middle aged and old. The universe of discourse in this case is the age of a person. In most cases it is not very useful to define a point in life to say a person is middle aged. To formulate rules for a computer or any other automated machine, these linguistic terms have to be transformed into numbers so that calculations can be done with them. To do this, fuzzy set theory comes into play. Back to the example, the following facts can be used:

- People between 0 and 18 are definitely young,
- from 20 onwards they gradually become middle aged and
- after 60 they are called old.
- Between 18 and 35 they become less young but more middle aged.
- Having completed the 30th year of life, they are definite middle aged.
- From 40 onwards someone belongs more and more to the old age group.[1]

Ie at the age of 26 someone is 40% to young and 30% middle aged.

In this example trapezoidal fuzzy sets have been used. Other shapes of sets, such as triangular, Gaussian, s-shaped, *etc* sets, are widely used. The advantage of using triangular sets is the simplicity of the function(s) employed to describe a triangle. Generally, the degree of membership is defined as a function $\mu(x)$, $0 \leq \mu(x) \leq 1$. The membership function is often obtained subjectively by one or more human experts. Averaging and other statistical methods can help to find the most appropriate functions. The process of decision making using fuzzy logic is divided into three major steps.

1. The first step is the transformation of any measured value into fuzzy terminology. This process is called fuzzification.

2. The second step is the processing of the obtained fuzzy value(s) by applying the fuzzy rules (inference).

3. The final step is the process of converting a linguistic term into a sharp, crisp number is called defuzzification. Different defuzzification methods are discussed later in this appendix.

[1] The author does not want to offend any reader by putting them in to an age group. Just treat it with a smile.

B.3 Fuzzy Operators

The two operators discussed here, are the the set union and set intersection. The two sets, A and B, are displayed as a dashed line and the result of the operation is a solid line.

The set union $(A \cup B)$ of the two sets A and B is shown in figure B.2. The graphical

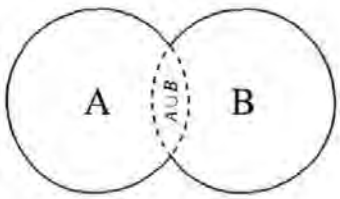


Figure B.2. Set Union of Two Sets $(A \cup B)$

interpretation of the set intersection $(A \cap B)$ is displayed in figure B.3. The two logical

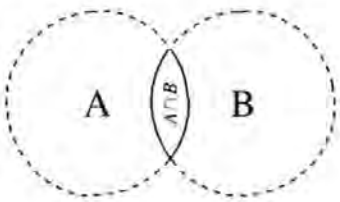


Figure B.3. Set Intersection of Two Sets $(A \cap B)$

operators can be modelled in different ways in fuzzy logic terms. Below, the three most commonly used fuzzy implementations of the set union and set intersection are described. The tree operators set union, set intersection and complement $(\cup, \cap, \sim)$ create new sets from existing sets. In boolean logic, those operators have their equivalent in OR, AND and NOT $(\vee, \wedge, \neg)$. So, the set union operator can be modelled with the OR operator of the boolean logic. Assuming two sets A and B, so $\mu_A, \mu_B : X \rightarrow 0,1$ is valid. The membership function $\mu_{A \cup B}$ of the set union of $A \cup B$ is:

$$\mu_{A \cup B}(x) = \mu_A(x) \vee \mu_B(x) \quad \forall x \in X \tag{B.1}$$

The set intersection can be modelled in a similar manner:

$$\mu_{A \cap B}(x) = \mu_A(x) \wedge \mu_B(x) \quad \forall x \in X \tag{B.2}$$

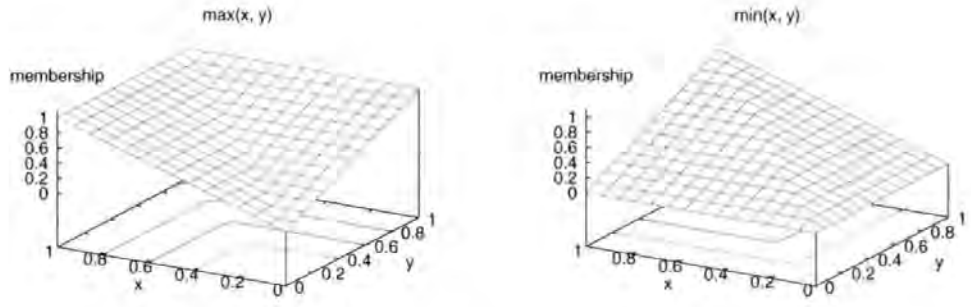


Figure B.4. Max/Min Fuzzy Operator

The complement of a boolean set ($\sim A$) is:

$$\mu_{\sim A}(x) \quad = \quad \neg \mu_A(x) \quad \forall x \in X \tag{B.3}$$

A whole range of functions exist which could be used to implement the fuzzy operators. However, the functions have to fulfil certain conditions.

B.3.1 The Min/Max Implementation

By looking at the two figures above, the fuzzy union can be modelled as the maximum (equation B.4) of two sets. The fuzzy intersection (AND, $\tilde{\wedge}$) is the opposite of the fuzzy union (OR, $\tilde{\vee}$), and therefore the fuzzy intersection can be modelled with the minimum (equation B.5) function. The fuzzy NOT ($\tilde{\neg}$) can be seen as the complement $x \rightarrow 1 - x$. The equations B.1, B.2, B.3 can be re-written:

$$\mu_{A \cup B}(x) = \mu_A(x) \; \tilde{\vee} \; \mu_B(x) \quad \forall x \in X, \tag{B.4}$$

$$\mu_{A \cap B}(x) = \mu_A(x) \; \tilde{\wedge} \; \mu_B(x) \quad \forall x \in X, \tag{B.5}$$

$$\mu_{\sim A}(x) = \tilde{\neg} \, \mu_A(x) \quad \forall x \in X. \tag{B.6}$$

Figure B.4 shows the graphical representation of the min and max operators. Other possible operators are the bounded max/min operators and the Yager-union/intersection.

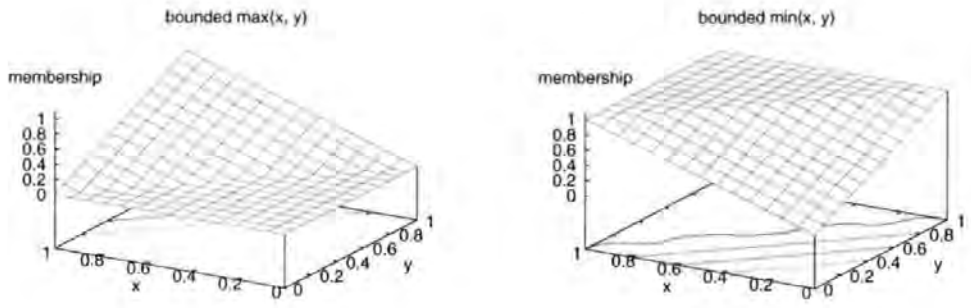


Figure B.5. Bounded Max/Min Fuzzy Operator

The Bounded Sum/Difference Implementation

In Figure B.5 one can find the bounded min and bounded max operators.

$$\begin{aligned} Z(x,y) &= max(0, x+y-1) \\ Z(x,y) &= min(1, x+y) \end{aligned} \tag{B.7}$$

The Yager-Union/Intersection Implementation

This method describes a whole family of functions. By modifying only one parameter (p) the shape of the function can be changed. If $p \gg 1$ the function approximates the min/max function from above. The graphical representation can be found in figure B.6.

$$\begin{aligned} Z_p(x,y) &= max(0, x^p + y^p)^{\frac{1}{p}} - 1) \text{ with } p \geq 1 \\ Z_p(x,y) &= min(1, x^p + y^p)^{\frac{1}{p}}) \end{aligned} \tag{B.8}$$

Figure B.6 shows the graphical representation of the Yager-Union and Yager-intersection operators.

Using the fuzzy operators, rules can be formulated. In the next example, the room temperature is controlled. A simple set of rules to employ can look like:

- **IF** TEMPERATURE **IS** WARM **AND** ROOM TEMPERATURE DROPS QUICKLY **THEN** OPEN VALVE A LOT
- **IF** TEMPERATURE **IS** WARM **AND** ROOM TEMPERATURE RISES **THEN** CLOSE VALVE A BIT

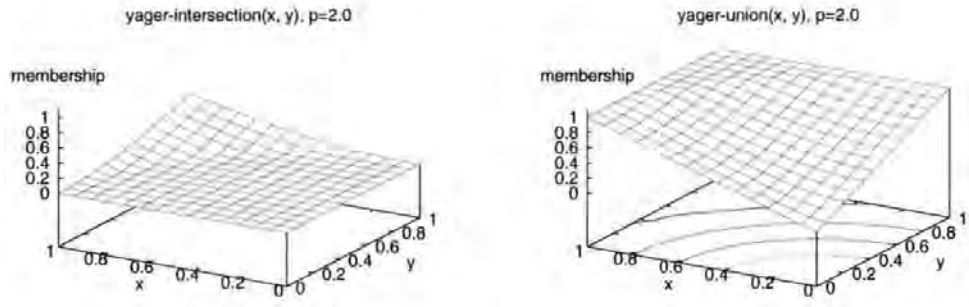


Figure B.6. Yager-Union/IntersectionFuzzy Operator

- **IF** TEMPERATURE **IS** WARM **AND** ROOM TEMPERATURE RISES QUICKLY **THEN** CLOSE VALVE A LOT.

The heater example is a simple control task. The variable (plant) to control is the room temperature, the control actuator is the valve of the heating element. There is one sensor in the feedback path which measures the temperature. From this measurement the rate of change of the temperature can be calculated. The input variables are the actual measured temperature and the temperature change rate. Both measured values are fuzzified using the fuzzy sets from figure B.7.

To complete the example, more fuzzy operations are needed. The actions are formulated as linguistic terms, too. But no device can work on such a basis. So, each fuzzy action has to be converted into a precise action to adjust the actuator, in this case, the position of the valve which controls the flow rate of the heating element.

B.4 Fuzzy Rules

With the the set theory in place, rules can be formed in the *fuzzy* manner. One can say:

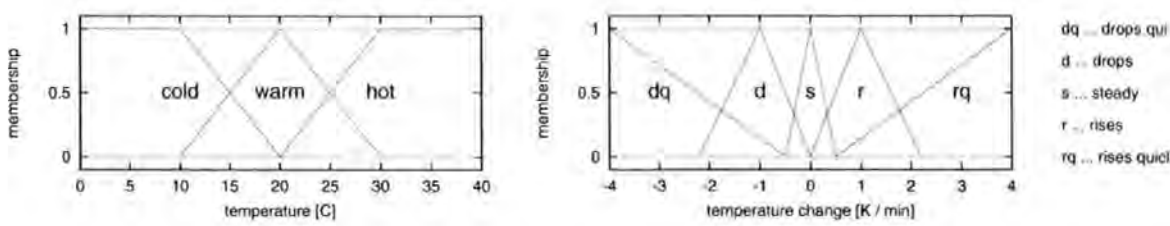


Figure B.7. Input Windows

IF ERROR **IS** POSITIVE SMALL **THEN IF** ERROR RATE **IS** ZERO

THEN DESIRED RUDDER **IS** NEGATIVE SMALL

The rule connects the error and the rate of the error in a similar way as the PD control rule (equation 2.2) does. The measured value is *fuzzified*, *eg* its membership value to the individual fuzzy subsets of the universe of discourse is obtained. The universes are assumed to be finite and discrete, in such a way that each universe is a set of elements

$$E = \{e\}, C = \{c\}, O = \{o\}$$

E ... universe of the error $E = \{e\}$,

C ... universe of the error rate $C = \{c\}$,

O ... universe of the output $O = \{o\}$,

With this in mind, rules can be formulated. In more symbolic notation rule k looks like:

IF E **IS** E_k **THEN IF** C **IS** C_k **THEN** O **IS** O_k

E_k ... fuzzy subset of E, $E_k \subset E$

C_k ... fuzzy subset of C, $C_k \subset C$

O_k ... fuzzy subset of O, $O_k \subset O$.

The fuzzy subsets, E_k, C_k and O_k, defined as ordered pairs where $\mu(.)$ represents the membership value giving the degree to which the element (measured value) is a member of the subset. Considering rule k, the ordered pairs are:

$$\begin{aligned} E_k &= \left\{\left(e, \mu_{E_k}(e)\right)\right\}, \\ C_k &= \left\{\left(r, \mu_{C_k}(r)\right)\right\}, \\ O_k &= \left\{\left(o, \mu_{O_k}(o)\right)\right\}. \end{aligned} \tag{B.9}$$

A control rule is an implication

$$E_k \rightarrow C_k \rightarrow O_k \tag{B.10}$$

which produces a relation matrix $\mathfrak{R}$ in hyperspace. Considering the three universes, a rule R_k of the $\mathfrak{R}$ is given by the outer product

$$R_k = E_k \times C_k \times O_k. \tag{B.11}$$

Now, a whole range of rules can be defined in such a way. A controller will use several such implications and the resulting (combined) relationship matrix $\mathfrak{R}$ is obtained using the union of the individual implications, *ie*

$$\mathfrak{R} = R_1 \tilde{\vee} R_2 \tilde{\vee} \ldots \tilde{\vee} R_k \tilde{\vee} \ldots \tilde{\vee} R_n = \tilde{\bigvee_n} R_n. \tag{B.12}$$

$\mathfrak{R}$ is a matrix of membership values $\mu_R(e,c,o)$. All control rules are stored within this matrix $\mathfrak{R}$ which represents the the fuzzy algorithm in its entirety. Using the implication of equation B.10 the controller function can be described, *eg* the inference from the error and change of error into the control action. The values e_i, c_i and o_i are individual elements of the universes of discourse E, C and O respectively. Theses elements can be represented in time. Considering the sampling time T, the elements can be formulated as $e(iT)$, $c(iT)$ and $o(iT)$, where i is the sample number $1 < i < \infty$, $i \in \mathbb{N}$. However, the values for e and c have to be obtained from the actual process. The first step is to scale the measured values, *eg* multiplying them with an appropriate scaling factor or gain, such as GE and GC, and then quantising the scaled result to the closest element in the universe of discourse. Considering the i^{th} sample, the process output $y(iT)$ and S as the process set-point, the values for $e(iT)$ and $c(iT)$ are calculated from

$$\begin{aligned} e(iT) &= Q\Big[\Big\{S - y(iT)\Big\} \times GE\Big] \\ c(iT) &= Q\Big[\Big\{y(iT) - y(iT - T)\Big\} \times GC\Big] \end{aligned} \tag{B.13}$$

However, the quantisation procedure of this application is reduced to *capping* the inputs, *ie* if the measured value is outside the universe, the value is set to the universe boundary. This applies to both, the error and the rate. The controller output is a fuzzy subset $O(iT)$ obtainable by utilising the fuzzy implication from equation B.10 which gives individual membership values as:

$$\mu_{O_{(iT)}}(o) = \mu_R\Big(e(iT), c(iT), o\Big). \tag{B.14}$$

This fuzzy subset has now to be *defuzzified* in order to produce on crisp output value. Some defuzzification methods are explained in more detail in appendix B.5 below. Before the defuzzified value is used, it is scaled to calculate the actual control action. This third scaling

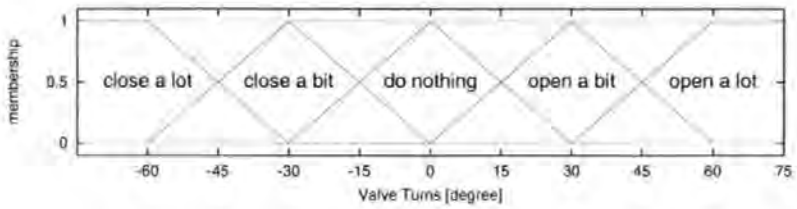


Figure B.8. Output Window - Valve Positions

factor is the output gain *GO*.

B.5 The Different Defuzzification Methods

The fuzzy sets displayed in figure B.8 can be seen as a general output window. Considering the example, negative means 'to close' whereby positive means 'to open', 'big' is associated with 'a lot' and 'medium' to a normal action.

If only the last four rules are taken into account, the rules can be written into a table as visualised in table B.1. Only the rules for the right temperature, warm being the goal, are considered.

Table B.1. Fixed Rulebase

	change of temperature $\Delta\vartheta$; the temperature ...				
temperature ϑ	**drops quickly**	**drops**	**doesn't change**	**rises**	**rises quickly**
cold					
warm	open a lot	open a bit	do nothing	close a bit	close a lot
hot					

Obviously, more rules are required to control this situation. In table B.2, a complete set of rules for two input variables can be found. The two input parameters are the temperature and the change of the temperature in the room. The temperature is divided into 3 categories, whereas the temperature change is split into five sets.

Finally, to control the device or plant, the general (linguistic) output of the rulebase has to be transformed into a crisp number. This process is called defuzzification. In most cases, not only one rule is activated. Using two input windows with a maximum of two sets overlaps, the maximum number of activated rules is four. Therefore a maximum of four rules have to

Table B.2. Full Fixed Rulebase

temperature ϑ	drops quickly	drops	doesn't change	rises	rises quickly
cold	open a lot	open a lot	open a bit	do nothing	close a bit
warm	open a lot	open a bit	do nothing	close a bit	close a lot
hot	open a bit	do nothing	close a bit	close a lot	close a lot

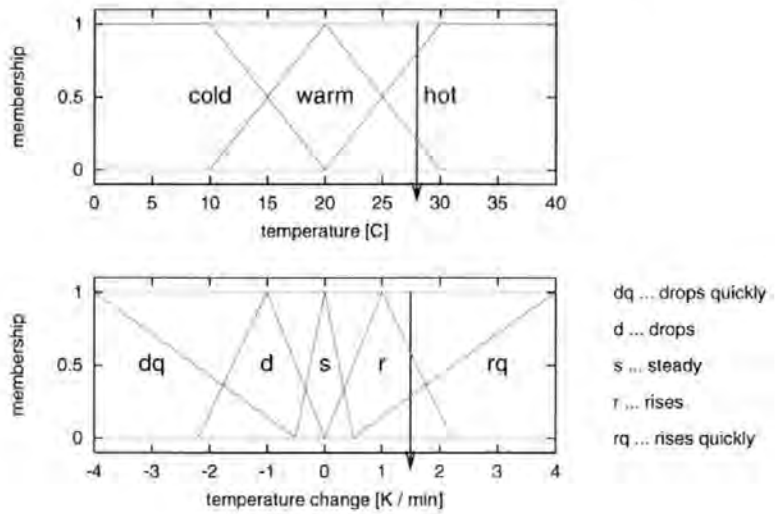


Figure B.9. Input Windows

be taken into consideration to calculate the final output.

Assuming a temperature of ϑ of 28 °C and a temperature rise $\Delta\vartheta$ of 1.5 $K \cdot min^{-1}$ (*see* figure B.9) then the following applies:

A temperature of $\vartheta = 28\ ^{\circ}C$ gives

$$\mu_{warm}(\vartheta) = 0.2667$$

$$\mu_{hot}(\vartheta) = 0.7333$$

A temperature change of $\Delta\vartheta = 1.5\ K \cdot min^{-1}$

$$\mu_{rises}(\Delta\vartheta) = 0.6667$$

$$\mu_{rises\ quickly}(\Delta\vartheta) = 0.2857$$

This results in four active rules. In this case only two different ones, one rule is hit three times with different values as seen in the following tables B.3–B.5. The numbers represent the membership of the sets active.

B.5.1 Centre of Area

Probably the most common defuzzification method is the centre of area. Two implementation of this defuzzification method are known. One which considers exactly the area covered by

Table B.3. Applied Fixed Rulebase 1

	change of temperature $\Delta\vartheta$; the temperature ...				
temperature ϑ	**drops quickly**	**drops**	**doesn't change**	**rises**	**rises quickly**
cold					
warm				close a bit	close a lot
hot				close a lot	close a lot

Table B.4. Applied Fixed Rulebase 2

	change of temperature $\Delta\vartheta$; the temperature ...				
temperature ϑ	**drops quickly**	**drops**	**doesn't change**	**rises**	**rises quickly**
cold					
warm				$\min(\mu_{warm},\mu_{rises})$	$\min(\mu_{warm},\mu_{rises\ quickly})$
hot				$\min(\mu_{hot},\mu_{rises})$	$\min(\mu_{hot},\mu_{rises\ quickly})$

Table B.5. Applied Fixed Rulebase 3

	change of temperature $\Delta\vartheta$; the temperature ...				
temperature ϑ	**drops quickly**	**drops**	**doesn't change**	**rises**	**rises quickly**
cold					
warm				0.2667	0.2667
hot				0.6667	0.2857

the sets, overlaps are considered only once. In contrast the overlaps can be ignored, therefore some small areas are considered twice. The final output is only very little influenced by the difference in the techniques.

Centre of Area with 'full' Fuzzy Sets

This method does not consider the overlapping of the sets. Basically the sets are taken as they are, and the centre is obtained utilising equation B.15. The graphical interpretation is shown in figure B.10. The numerator is the first moment of the area of a set.

$$output = overall\ cg = \frac{\sum_i A_i \cdot cg_i}{\sum_i A_i} \tag{B.15}$$

cg ... centre of gravity

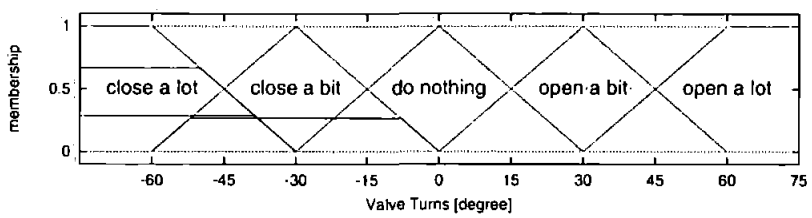


Figure B.10. Active Rules

Centre of Area with 'cropped' Fuzzy Sets

Here, the overlapping area is only taken into account once. Therefore cross-over points of the sets have to be calculated and if a rule is hit twice or more times, only the maximum is used (*see* figure B.11). In the example, only two sets (close a bit and close a lot) produce the crisp output.

B.5.2 The Mean Of Maxima Method

This method is not widely used and has a tendency to give unsatisfactory results. It only uses the position of the maxima of each set and it ignores the area under the set. The average of

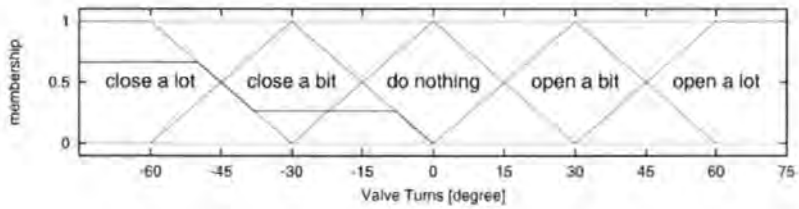


Figure B.11. Active Rules

those positions will give the final, crisp output.

$$output = \frac{1}{n} \cdot \sum_{i=1}^{n} u_i \tag{B.16}$$

$u_i \ldots$ position of the maximum

B.5.3 The Fuzzy Singleton Method

If the fuzzy sets in the output window become very slim (*see* blue lines in figure B.12), do not overlap or even touch each other, then they become fuzzy singletons. The defuzzification using fuzzy singletons reduces to the calculation of the centre of gravity of lines. The membership gives directly the 'area'. No further converting is necessary. A control surface resulting from fuzzy singletons in the output window can be seen in figure B.13.

$$output = overall\ cg = \frac{\sum_i A_i \cdot cg_i}{\sum_i A_i} \tag{B.17}$$

$A_i \ldots \mu(output)$,

$cg_i \ldots$ the position in the universe of discourse

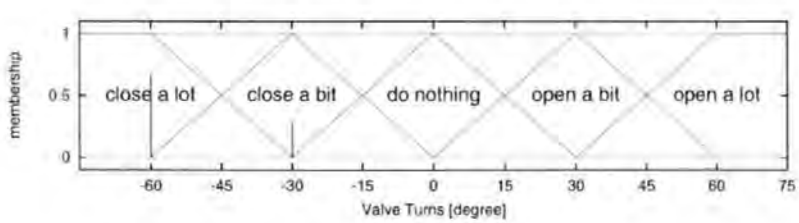


Figure B.12. Fuzzy Singletons

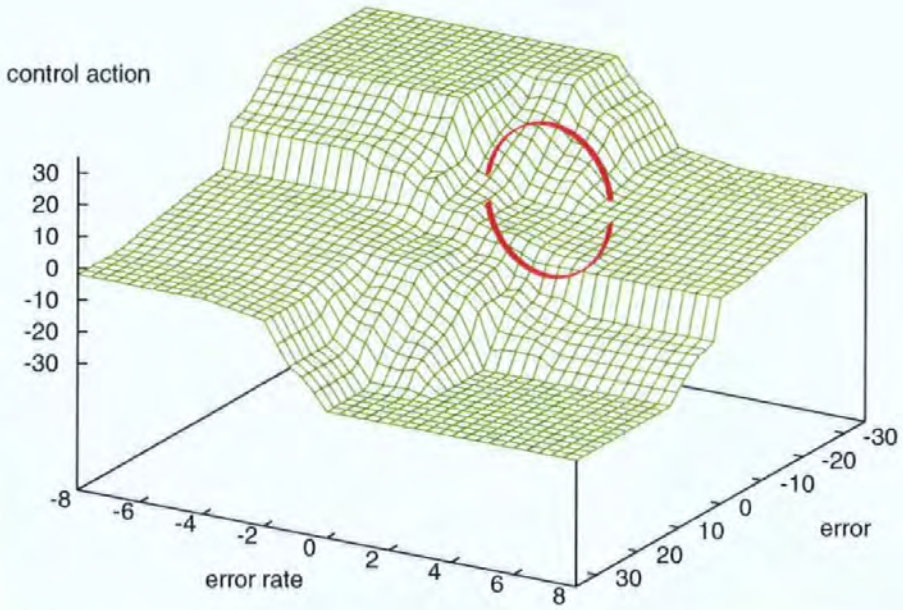


Figure B.13. Control Surface using Fuzzy Singletons

B.6 Applications of Fuzzy Logic

This section highlights briefly some aspects of fuzzy logic important to control purposes. The main principles of fuzzy logic as proposed by Zadeh [127] in 1965 were explained earlier in this chapter.

The normal way of expressing rules and knowledge is by using linguistic variables, but it is difficult to implement thoughts on a digital machine. The idea of using so-called fuzzy sets and validating a fact against them opens up an interesting field. Therefore, crisp and precise values are fuzzified and then treated as fuzzy data. The combination of the values, *eg* if the fact belongs partially to more than one set, is passed through a rulebase and a range of output data is created. Each data represents a membership of a fuzzy set in the output window. After defuzzification, a crisp value is produced and provides the controller's crisp output. The big advantage of using fuzzy logic is (a) the easy understanding of the rules and relationship since elements of the human language are used to express them and (b) the ability to merge together possibly disparate information in order to generate a deterministic output.

The applications of fuzzy logic reach from high street products to process control. One can find fuzzy logic in washing machines (AEG, Goldstar), photographic equipment (Canon, Minolta, Ricoh, Sanyo), temperature control (refrigerator by Whirlpool), vacuum cleaners (Philips, Siemens) *etc* [55]. The automotive industry adapted fuzzy logic in power train and transmission control (GM-Saturn, Honda, Mazda) and Nissan utilised it in engine control. One of the first application using fuzzy logic was the automation of the cement kiln operation [1, 79].

The main idea is still the same, which is to *use human understandable expressions*, to deal with imprecise data and utilise the information *ie* with a computer. More applications will occur in the industrial field. This thesis discusses one potential area in control engineering but fuzzy logic found utilisation in other areas such as design, finance, engineering and medicine [60] and many others. Many industrial applications around the world are listed and briefly explained in the book [126] edited by Yen *et al.*

B.7 Fuzzy Logic Summary

It has been demonstrated that fuzzy logic is a good tool for control tasks [49, 26, 81, 99] including ship heading control. It is relatively easy to use and easy to implement. The major advantage is the capability of using linguistic variables to describe human thoughts. These thoughts are summarised in a fuzzy rulebase which can be single or multi dimensional. The rulebase can combine facts from different sources, *eg* universe of discourses.

This chapter therefore highlighted some aspects of fuzzy logic. It discussed the main principles as proposed by Zadeh [127, 128] including the definition of fuzzy sets and fuzzy operators.

For application to control tasks, control rules are defined by the engineer and stored within the controller in a rulebase. In order to produce one single control output, the data produced by the fuzzy logic have to be defuzzified. The different defuzzification methods are explained and discussed.

This chapter has provided an introduction to self-organising fuzzy logic control which has been utilised to form the *predictive self-organising fuzzy logic controller* as explained in

chapter 4. The working principles of the performance index are briefly discussed as well as the method of the rule changing utilising the outcome of the performance index.

Appendix C

The Simulation Set-Up

C.1 The Ship Mathematical Model

Today, the majority of control tasks are undertaken using digital controllers. They are developed off-line, without being attached to the practical hardware (plant). Mathematical models substitute for the plants being controlled in the design process. To use a model instead of the real plant has certain benefits. For example, the plant is not involved in the design process of the new controller at any time. The key factor being to save resources, *ie* hardware, time, money, and sacrificed test equipment which can be very expensive. Using computer models, the real plant remains untouched. Furthermore, the real plant can still be used while a new controller is being designed and there is no loss in production due to maintenance or down time. Finally, the plant might not be available for the length of time that the design and testing of a controller requires. Often a simulation using models can run faster than real time with obvious advantages. More testing can be done in the same or even in less time. Also, a computer model can be copied and run on several machines at the same time simulating different conditions. The physical hardware remains untouched and the off-line time can so be reduced to a minimum.

These are only a few factors which highlight the advantages of using models. The testing of any new device can be first achieved during simulation employing physical and/ or computer models.

In a previous research project in collaboration with Cetrek Ltd. of Poole (UK) a six degree of freedom lifeboat model was developed by Browning [20] at Bournemouth Poly-

Table C.1. Lifeboat Parameters

number of propellers	2	
number of rudders	2	
length	$51'8\frac{1}{2}"$	15.8m
beam	17'0"	5.18m
draft	$3'7\frac{5}{8}"$	1.11m
displacement		32,415t

technic. The model represents, with its original set-up, a lifeboat with the parameters as shown in table C.1.

This lifeboat simulation fulfils the requirements of this research to investigate small, highly responsive, motorised vessels. This simulation is the basis for all further investigations. The boat simulation runs on a 386SX PC based machine in approximately real time. An interface to an external controller (autopilot) has already been included. It is therefore possible to attach another PC or digital device which is using a serial port (RS232) for communication purposes.

The simulation too has its limitations. It was found that the waves, as simulated by Browning, are Gaussian Random noise within operator-specified limits. The 'sensor readings' are sent via the serial communications port. This implies discrete handling of the measured data, sampling frequency is dependant on the power of the processor and communication frequency is fixed and limited by the UART initialisation and used (NMEA) protocol.

C.2 The Integrated Autopilot Testbed

The testbed is the part of the setup which contains the controller and data–logger. Data is exchanged using the RS232 serial ports on both PCs using a protocol based on messages. The messages exchanged follow the widely used National-Marine-Electronics-Association 0183 (NMEA) format [72]. This standard provides the foundation for the communications between maritime devices, since it includes messages to control heading and rudder.

C.2.1 Graphical User Interface and NMEA Messages

The first task to be solved was the creation of an independent interface for communication with the lifeboat simulation. This program acts as a data-logging (monitoring) program. Various heading controllers for the ship or simulation are embedded. Using the Function-keys, a choice of *heading* controllers can be made. The range is:

- with a simple P controller and also includes
- PD (*see* section 3.1),
- PID,
- fixed rulebase fuzzy logic,
- SoFLC and the
- PSoFLC (function keys F1–F6 respectively).

The main windows displays the course travelled. Each square of the grid represents a distance of 100m. The two windows underneath show the history of the heading error and the rudder position. On the right-hand side, the visual appearance of Cetrek devices has been copied to show the heading error and the current rudder position in an analogue manner. The absolute heading can be seen in a digital form in the compass device (top). The current time and the elapsed time of the run is located in the right top corner of the screen with an underlying world grid displaying the current position on the globe. All those devices are updated constantly at a sampling time depending on the speed of the computer that the program runs.

The programming language used is C++. The code is written in an object oriented manner to allow easy re-use of program components. A screen shot of the program can be seen in figure C.1. One module is handling all the NMEA communication with the lifeboat simulation.

The messages between the model (or real world ship) are exchanged, using the NMEA standard. The messages considered are summarised in appendix D.

A special message containing information about the vessel's current state is transmitted by the lifeboat simulation using the customised MOD (model) message (*see* appendix D for

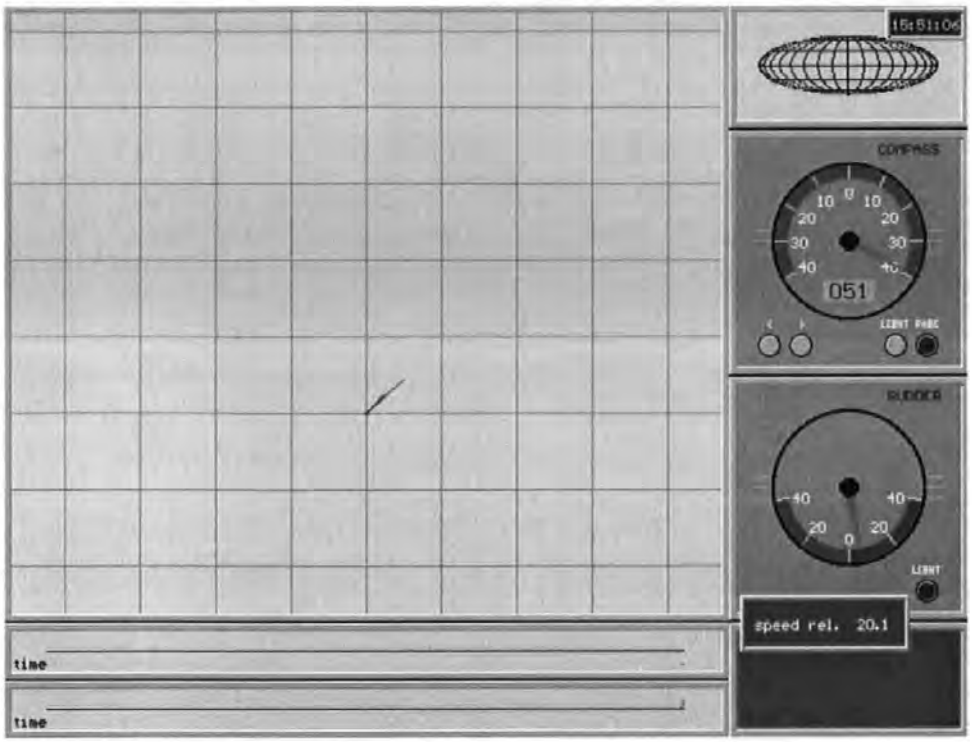


Figure C.1. Screen Shot of the Program

syntax and message content). It contains data about heading, rudder angle, the yaw rate and roll and pitch angle.

The autopilot is fully embedded into this environment. It uses the data from the NMEA messages sent by the lifeboat as input variables. The autopilot itself uses only the heading information and its derivative (ψ and $\dot{\psi}$). However, when running the PSoFLC far more information is considered. To build the internal mathematical model of the vessel, information used is as follows:

- rudder,
- roll,
- pitch,
- speed,
- heading.

To 'communicate' with the ship, information about the direction and the speed to drive the rudder is transmitted directly into the distribution box of the vessel (during simulation).

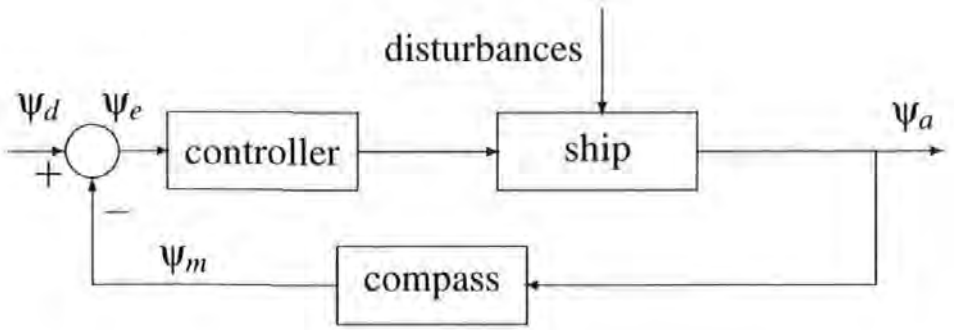


Figure C.2. General Layout of the Control Task

The information transmitted represents a flow rate of the oil in the pump driving the rudder shaft. The simulation has a variable speed oil pump.

During tests using a real boat, fitted with Cetrek devices, the autopilot software was executed on a laptop. The software sends a RSA (*see* appendix D for syntax and message content) message, only containing a desired rudder angle. Then the rudder loop inside the '619 Distribution Box' controls the rudder to the desired position. When the distribution box receives this message, it overrides the built-in autopilot and uses the received messages as 'dodge' command (as it does in power–steer mode). The built-in autopilot has an input device for entering a desired course and giving power-steer commands. Two buttons on the device increase/decrease the value to be changed in predefined steps. In power-steer mode, a press of the button will have the effect of moving the rudder to the desired angle.

The actual test bed not only includes a control loop for the course keeping but also a rudder loop to control the desired position of the rudder. Here, a fixed rulebase fuzzy logic controller has been used to achieve good performance with minimal development effort. The rudder loop is not used when operated on the real ship during sea trials but during simulations.

The block layout of the controller developed (*see* figure C.2) shows a standard control loop with negative feedback to provide a heading error vector including ψ and $\dot{\psi}$. Figure C.3 shows a more detailed layout with both the outer and a inner loop (heading control loop and rudder control loop).

The actual rudder position is transmitted by the boat's rudder feedback device in NMEA format.

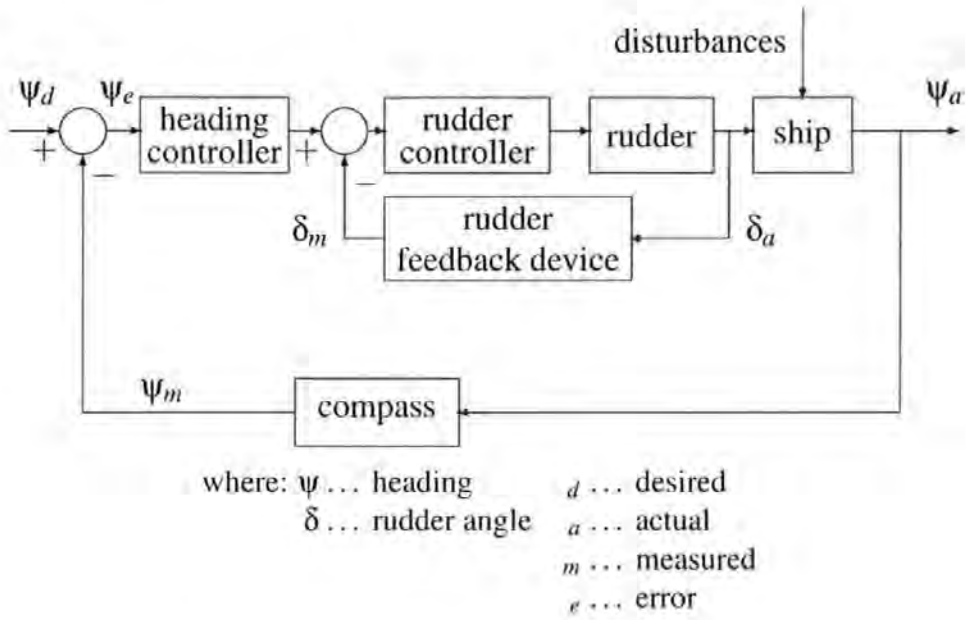


Figure C.3. Detailed Layout of the Control Task

C.2.2 Data–Logging

Part of the test bed is the data-logger. All events (heading information, rudder changes, positions *etc*) are held in an array in memory. This array is saved to file when the program is exited. The sample rate can be reset in the source code or it can be switched to an event–driven logger. If the event–driven logger is selected, points are only stored when an event (events can be OR'ed) occurs. Giving each record a time–stamp proved to be very valuable since the sampling frequency is not constant when logging events. This time-stamp is then used as an x-axis in history plots.

The program can be started with a manoeuvre filename as a parameter from the command line. The manoeuvre file contains information on the planned course to execute. There are two numbers in each record – 1) desired heading, and 2) the time required for this heading to be maintained. This feature has been used to set up simulation runs and to guarantee the same working conditions for each controller used during testing and for comparison.

Data is written to a file which makes it easy to repeat and analyse the simulation off-line. The number of saved points depends on the memory available in the machine. In order to save time, all points are held in memory (dynamic list) until the program is stopped. Points held in memory appear in green, points which cannot be stored in memory (due to lack of free resources and therefore not in the file) are shown as yellow crosses. The points are

simply connected by a line to show the covered course in the track window.

C.3 Summary

This chapter highlighted some aspects of mathematical modelling of ships and the characteristics of the sample vessel.

The program developed for testing and comparison of autopilot designs has been discussed in detail. This software provides the testbed for all the work carried out including development and testing in simulation. The graphical user interface allows easy visual analysis of the process. It gives instant readings of vital information such as rudder angle, travelled course and heading error.

Appendix D

NMEA Messages considered

Geographic Position – Latitude/ Longitude

`$xxGLL, llll.ll,n,yyyyy.yy,m,hhmmss.ss,A*hh<CR><LF>`

with: llll.ll ... the vessel's latitude,

n ... N/S for north/ south respectively,

yyyy.yy ... the vessel's longitude,

m ... E/W for east/ west respectively,

hhmmss.ss ... UTC of position

A*hh ... A = data valid

This message gives the longitude as well as the latitude of the vessel's position on the globe. This data is used to plot the true way covered by the vessel.

Water Speed and Heading

`$xxVHW,h.h,T,m.m,M,s.s,N,x.x,K*hh<CR><LF>`

with: h.h ... the vessel's heading,

T ... true,

m.m ... the vessel's heading,

M ... magnetic,

s.s ... the vessel's speed,

N ... knots,

s.s ... the vessel's speed,

K ... $km \cdot h^{-1}$

Out of this message, the relative speed of the vessel can be extracted as well as the current heading.

Wind Speed and Direction

`$xxMWV,xxx.x,T,yyy.y,K,A*hh<CR><LF>`

with: xxx.x ... the wind angle,

T ... True or Relative (R) wind angle,

yyy.y ... the wind speed

K ... K ($km \cdot h^{-1}$), M ($m \cdot h^{-1}$)

A*hh ... A = data valid

Customised Message (MODel)

`$xxMOD,xxx.x,M,yy.y,L,zz.z,ppp.p,rrr.r<CR><LF>`

with: xxx.x ... the vessel's heading,

M ... Magnetic or True heading,

yy.y ... the rudder angle in °

L ... L ... left, R ... right

zz.z ... the yaw rate

ppp.p ... the pitch angle

rrr.r ... the roll angle

Appendix E

Rulebases

The first 30s are executed by the PD to align the vessel to 0°. There are 25s after each 20°step change to serve exactly the same purpose, to realign the vessel in case the autopilot tested was unable to keep the desired course. This has been included to have always the same starting positions when initiating a new step. During the realigning, the PD controller is functioning. The sequence is displayed in table E.1.

Table E.1. Step Response Manoeuvre

		heading	time	operating controller
	0	0°	10s	PD
	30	0°	20s	PD
run 1	120	20°	90s	PSoFLC/SoFLC
	145	20°	25s	PD
	235	0°	90s	PSoFLC/SoFLC
	260	0°	25s	PD
run 2	350	20°	90s	PSoFLC/SoFLC
	375	20°	25s	PD
	465	0°	90s	PSoFLC/SoFLC
	490	0°	25s	PD
run 3	580	20°	90s	PSoFLC/SoFLC
	605	20°	25s	PD
	695	0°	90s	PSoFLC/SoFLC
	720	0°	25s	PD

Figure E.1 shows the time setup for the step change. The purple dots indicate a sampling point of the rulebase. The contents and its visualisation can be found in the following section.

The heading and rudder charts only display the data as collected when the PSoFLC or SoFLC were operating, *eg* no rulebase update was performed during the 'alignment' when

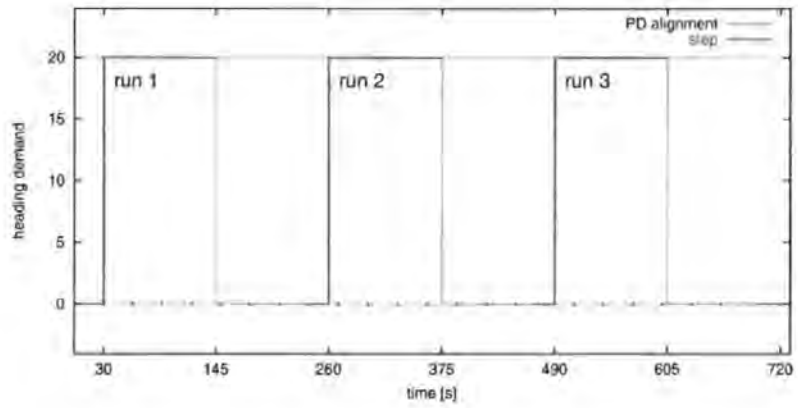


Figure E.1. The Step Response Test

the PD controller was operating.

During this 12 min period, not all rules have been activated and so some areas of the rulebase remain unchanged or little changed whereas other areas show a big activity. *Rules remain unchanged*, does not mean the controller was performing well in the considered regions, it only means there was no rule update required in those areas. It is largely due to the environment that the *inner* rules of the controller were sufficient to cover the existing environment. The controller can only learn the response of a environment is it exposed to. The controller only learns, what is needed according to the current and past vessel states and environmental conditions.

From the interaction of these rules, the non-linearities of the controller's rulebase become apparent.

E.1 PSoFLC

E.1.1 Measured Values

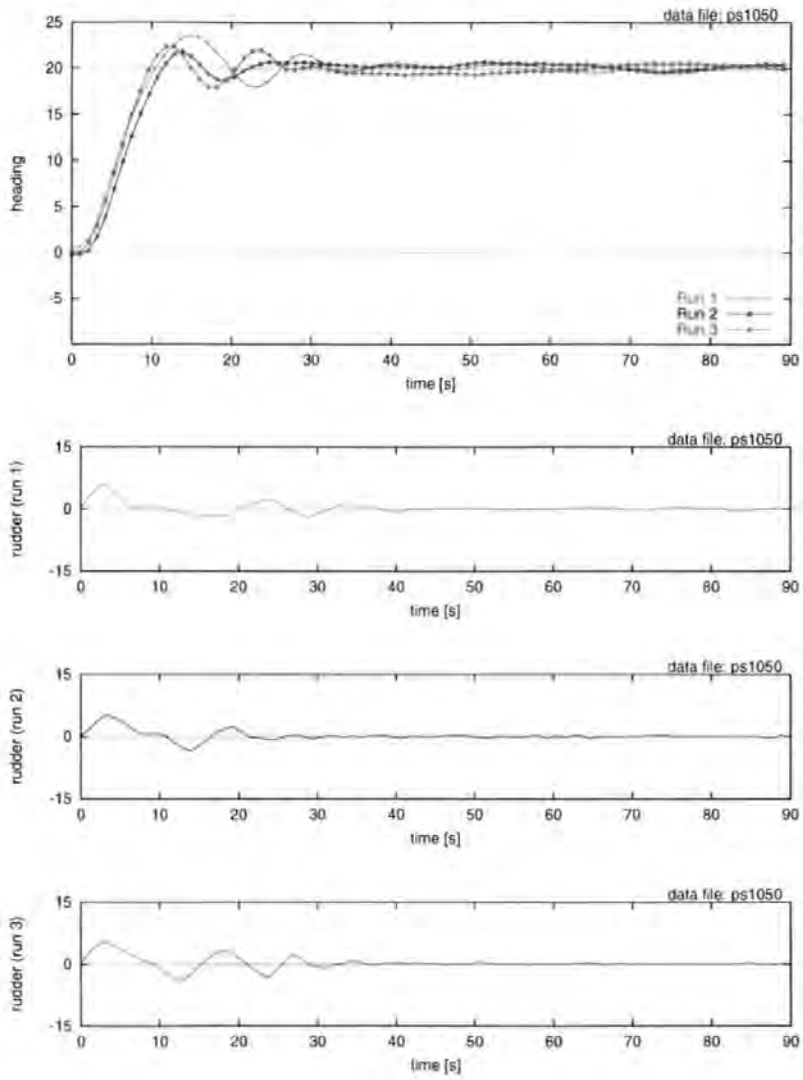
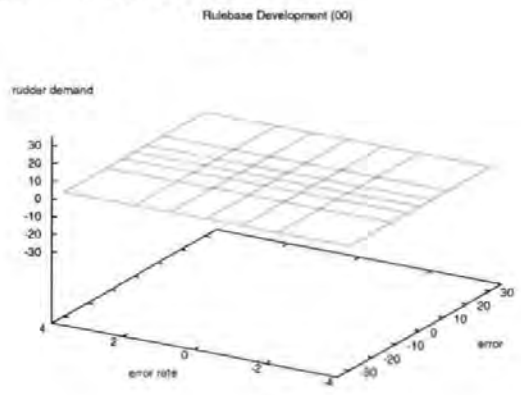


Figure E.2. Step Response 1050 rpm, 21 knots - Rulebase logging

The figure E.2 above shows the heading during the $\pm 20°$step change where the rulebases following where logged.

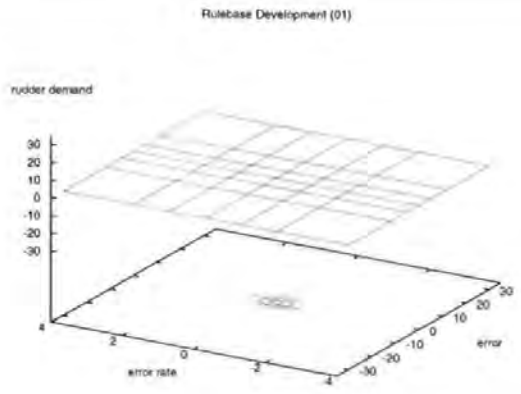
The following tables and graphs represent the rulebase, not the control surface, in approx. 30s intervals.

E.1.2 The Rulebases



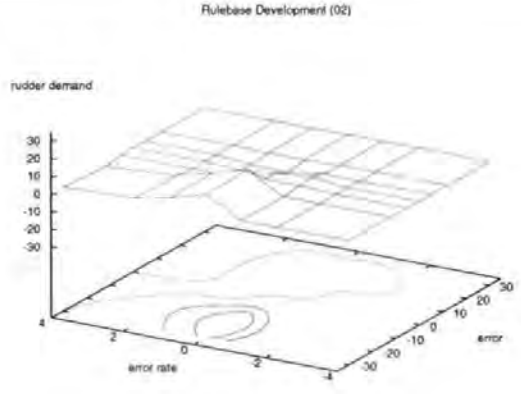
the preset rulebase

	NB	NM	NS	Z	PS	PM	PB
NB	0.00	0.00	0.00	0.00	0.00	0.00	0.00
NM	0.00	0.00	0.00	0.00	0.00	0.00	0.00
NS	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Z	0.00	0.00	0.00	0.00	0.00	0.00	0.00
PS	0.00	0.00	0.00	0.00	0.00	0.00	0.00
PM	0.00	0.00	0.00	0.00	0.00	0.00	0.00
PB	0.00	0.00	0.00	0.00	0.00	0.00	0.00



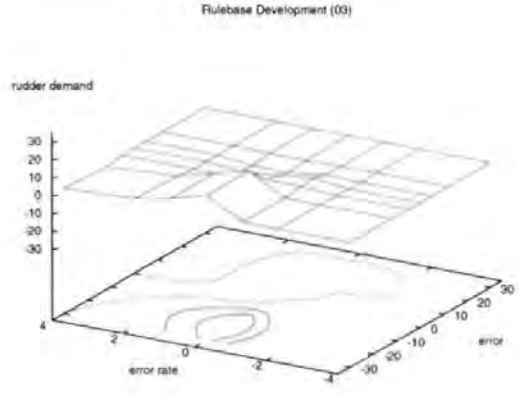
the changed rulebase at 11:25:50.27 (1)

	NB	NM	NS	Z	PS	PM	PB
NB	0.00	0.00	0.00	0.00	0.00	0.00	0.00
NM	0.00	0.00	0.00	0.00	0.00	0.00	0.00
NS	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Z	0.00	0.00	0.00	0.19	0.00	0.00	0.00
PS	0.00	0.00	0.00	0.03	0.00	0.00	0.00
PM	0.00	0.00	0.00	0.00	0.00	0.00	0.00
PB	0.00	0.00	0.00	0.00	0.00	0.00	0.00



the changed rulebase at 11:26:20.28 (2)

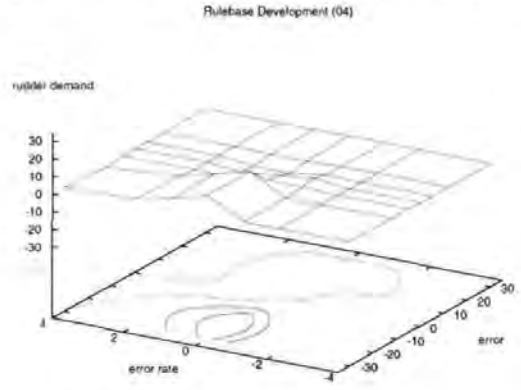
	NB	NM	NS	Z	PS	PM	PB
NB	0.00	0.00	0.00	11.24	5.62	1.70	0.08
NM	0.00	0.00	0.14	11.51	6.08	1.03	-1.62
NS	0.00	0.00	1.07	1.33	-0.22	-0.50	-0.36
Z	0.00	0.00	2.24	0.77	-1.28	-0.02	0.00
PS	0.00	0.00	0.28	-2.43	-2.05	0.00	0.00
PM	0.00	0.00	-0.14	-0.83	-0.61	0.00	0.00
PB	0.00	0.00	0.00	0.00	0.00	0.00	0.00



the changed rulebase at 11:26:50.29 (3)

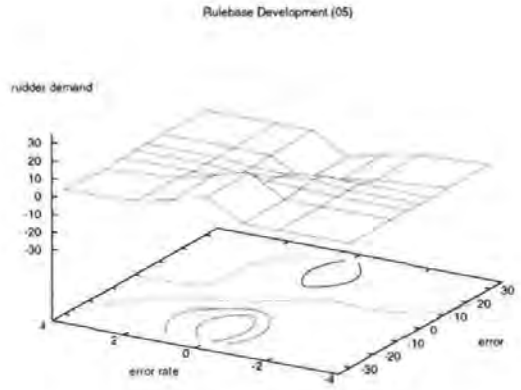
	NB	NM	NS	Z	PS	PM	PB
NB	0.00	0.00	0.00	11.24	5.62	1.70	0.08
NM	0.00	0.00	0.14	11.51	6.08	1.03	-1.62
NS	0.00	0.00	1.07	1.30	-0.24	-0.50	-0.36
Z	0.00	0.00	2.25	-0.19	-1.49	-0.02	0.00
PS	0.00	0.00	0.49	-2.37	-2.11	0.00	0.00
PM	0.00	0.00	-0.13	-0.82	-0.61	0.00	0.00
PB	0.00	0.00	0.00	0.00	0.00	0.00	0.00

The adaptation of the rules is very rapid. During the first 30s after the step change, the majority of the adaptation is complete. Rulebases (1), (2), (3) and (4) clearly demonstrate that. There is only very little difference between rulebases (2), (3) and (4). Rulebase (2) is sampled about 30s into the first step change. Most of the adaptation occurred before this sample. This can be concluded from the differences in rulebase (1) and (2).



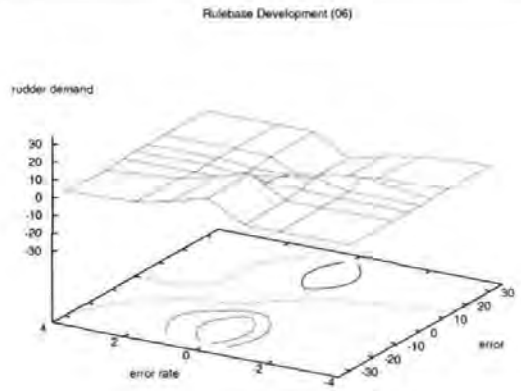
the changed rulebase at 11:27:20.29 (4)

	NB	NM	NS	Z	PS	PM	PB
NB	0.00	0.00	0.00	11.24	5.62	1.70	0.08
NM	0.00	0.00	0.14	11.51	6.08	1.03	-1.62
NS	0.00	0.00	1.13	1.38	-0.26	-0.50	-0.36
Z	0.00	0.00	2.39	0.04	-1.57	-0.02	0.00
PS	0.00	0.00	0.48	-2.39	-2.12	0.00	0.00
PM	0.00	0.00	-0.13	-0.82	-0.61	0.00	0.00
PB	0.00	0.00	0.00	0.00	0.00	0.00	0.00



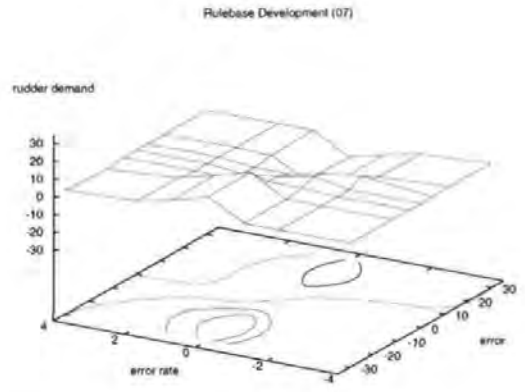
the changed rulebase at 11:27:50.29 (5)

	NB	NM	NS	Z	PS	PM	PB
NB	0.00	0.00	0.00	11.24	5.62	1.70	0.08
NM	0.00	0.00	0.14	11.51	6.08	1.03	-1.62
NS	0.00	0.00	1.13	1.38	-0.26	-0.50	-0.36
Z	0.00	0.00	2.39	0.04	-1.57	-0.02	0.00
PS	0.00	0.00	0.48	-2.39	-2.12	0.00	0.00
PM	0.22	-3.24	-6.78	-11.21	-0.61	0.00	0.00
PB	-0.07	-2.25	-6.42	-11.39	0.00	0.00	0.00



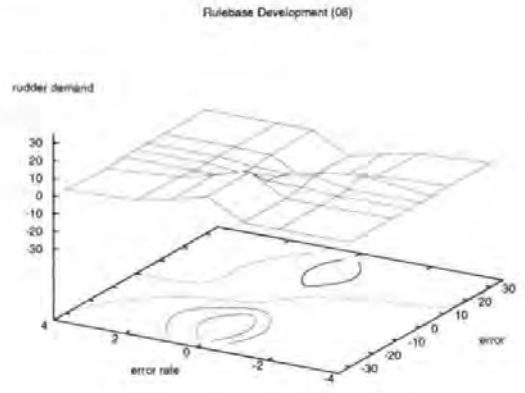
the changed rulebase at 11:28:20.30 (6)

	NB	NM	NS	Z	PS	PM	PB
NB	0.00	0.00	0.00	11.24	5.62	1.70	0.08
NM	0.00	0.00	0.14	11.47	6.07	1.03	-1.62
NS	0.00	0.00	1.17	1.05	-0.30	-0.50	-0.36
Z	0.00	0.00	4.06	0.62	-1.97	-0.02	0.00
PS	0.12	4.17	1.97	-2.02	-2.16	0.00	0.00
PM	0.69	-1.63	-6.33	-11.20	-0.61	0.00	0.00
PB	-0.04	-2.22	-6.42	-11.39	0.00	0.00	0.00



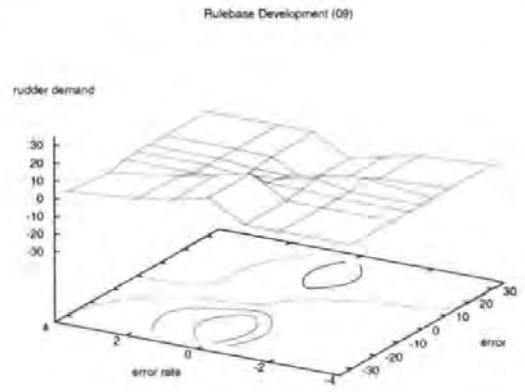
the changed rulebase at 11:28:50.30 (7)

	NB	NM	NS	Z	PS	PM	PB
NB	0.00	0.00	0.00	11.24	5.62	1.70	0.08
NM	0.00	0.00	0.14	11.47	6.07	1.03	-1.62
NS	0.00	0.00	1.22	1.13	-0.30	-0.50	-0.36
Z	0.00	0.00	4.05	0.17	-2.06	-0.02	0.00
PS	0.12	4.17	1.87	-2.27	-2.19	0.00	0.00
PM	0.69	-1.63	-6.33	-11.20	-0.61	0.00	0.00
PB	-0.04	-2.22	-6.42	-11.39	0.00	0.00	0.00



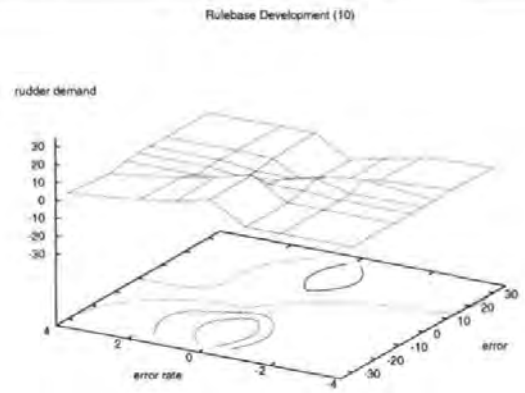
the changed rulebase at 11:29:20.30 (8)

	NB	NM	NS	Z	PS	PM	PB
NB	0.00	0.00	0.00	11.24	5.62	1.70	0.08
NM	0.00	0.00	0.14	11.47	6.07	1.03	-1.62
NS	0.00	0.00	1.22	1.13	-0.30	-0.50	-0.36
Z	0.00	0.00	4.02	-0.01	-2.06	-0.02	0.00
PS	0.12	4.17	1.85	-2.30	-2.18	0.00	0.00
PM	0.69	-1.63	-6.33	-11.20	-0.61	0.00	0.00
PB	-0.04	-2.22	-6.42	-11.39	0.00	0.00	0.00



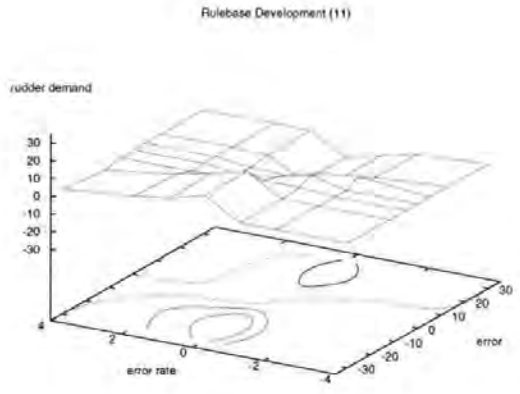
the changed rulebase at 11:29:50.33 (9)

	NB	NM	NS	Z	PS	PM	PB
NB	0.00	0.00	0.00	11.75	6.54	4.00	0.29
NM	0.00	0.00	0.14	11.85	7.00	3.00	-2.70
NS	0.00	0.00	1.22	1.13	-0.34	-0.61	-0.60
Z	0.00	0.00	4.02	-0.01	-2.06	-0.02	0.00
PS	0.12	4.17	1.85	-2.30	-2.18	0.00	0.00
PM	0.69	-1.63	-6.33	-11.20	-0.61	0.00	0.00
PB	-0.04	-2.22	-6.42	-11.39	0.00	0.00	0.00



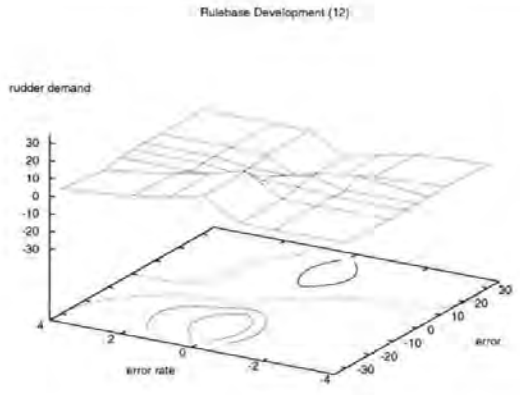
the changed rulebase at 11:30:20.35 (10)

	NB	NM	NS	Z	PS	PM	PB
NB	0.00	0.00	0.00	11.75	6.54	4.00	0.29
NM	0.00	0.00	0.15	11.86	6.96	2.95	-2.70
NS	0.00	0.00	1.46	1.38	-0.58	-0.86	-0.60
Z	0.00	0.00	4.55	0.21	-2.53	-0.06	0.00
PS	0.12	4.17	1.81	-2.70	-2.47	0.00	0.00
PM	0.69	-1.63	-6.32	-11.23	-0.64	0.00	0.00
PB	-0.04	-2.22	-6.42	-11.39	0.00	0.00	0.00



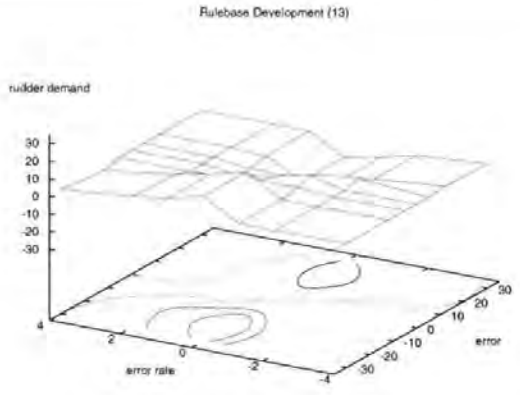
the changed rulebase at 11:30:50.36 (11)

	NB	NM	NS	Z	PS	PM	PB
NB	0.00	0.00	0.00	11.75	6.54	4.00	0.29
NM	0.00	0.00	0.15	11.86	6.96	2.95	-2.70
NS	0.00	0.00	1.46	1.38	-0.58	-0.86	-0.60
Z	0.00	0.00	4.44	-0.42	-2.54	-0.06	0.00
PS	0.12	4.17	1.73	-2.86	-2.48	0.00	0.00
PM	0.69	-1.63	-6.32	-11.23	-0.64	0.00	0.00
PB	-0.04	-2.22	-6.42	-11.39	0.00	0.00	0.00



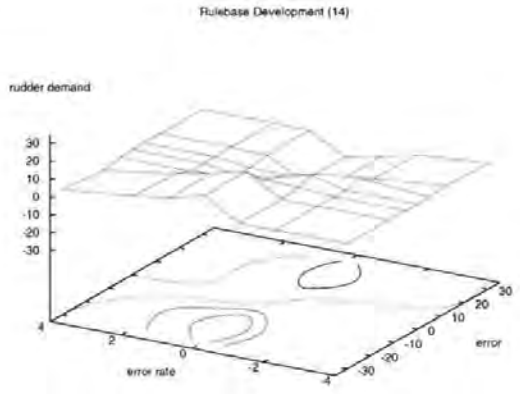
the changed rulebase at 11:31:20.37 (12)

	NB	NM	NS	Z	PS	PM	PB
NB	0.00	0.00	0.00	11.75	6.54	4.00	0.29
NM	0.00	0.00	0.15	11.86	6.96	2.95	-2.70
NS	0.00	0.00	1.49	1.42	-0.58	-0.86	-0.60
Z	0.00	0.00	4.48	0.24	-2.52	-0.06	0.00
PS	0.12	4.17	1.73	-2.85	-2.47	0.00	0.00
PM	0.69	-1.63	-6.32	-11.23	-0.64	0.00	0.00
PB	-0.04	-2.22	-6.42	-11.39	0.00	0.00	0.00



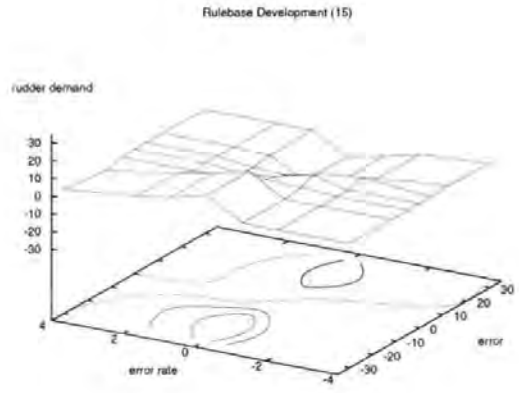
the changed rulebase at 11:31:50.37 (13)

	NB	NM	NS	Z	PS	PM	PB
NB	0.00	0.00	0.00	11.75	6.54	4.00	0.29
NM	0.00	0.00	0.16	11.87	6.96	2.95	-2.70
NS	0.00	0.00	1.76	1.71	-0.58	-0.86	-0.60
Z	0.00	0.00	4.95	0.50	-2.52	-0.06	0.00
PS	0.02	3.42	1.55	-2.81	-2.47	0.00	0.00
PM	1.16	-2.41	-7.91	-11.20	-0.64	0.00	0.00
PB	0.28	-2.52	-7.43	-11.32	0.00	0.00	0.00



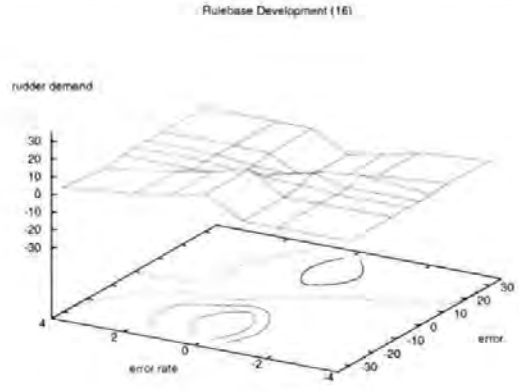
the changed rulebase at 11:32:20.37 (14)

	NB	NM	NS	Z	PS	PM	PB
NB	0.00	0.00	0.00	11.75	6.54	4.00	0.29
NM	0.00	0.00	0.17	11.88	6.96	2.95	-2.70
NS	0.00	0.00	1.75	1.77	-0.60	-0.86	-0.60
Z	0.00	0.00	4.83	-0.67	-2.74	-0.06	0.00
PS	0.02	3.42	1.51	-2.89	-2.50	0.00	0.00
PM	1.16	-2.41	-7.91	-11.20	-0.64	0.00	0.00
PB	0.28	-2.52	-7.43	-11.32	0.00	0.00	0.00



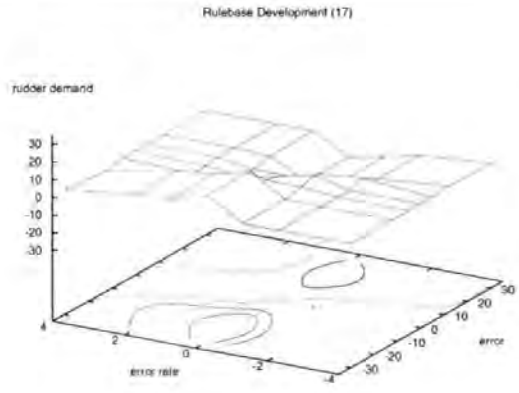
the changed rulebase at 11:32:50.38 (15)

	NB	NM	NS	Z	PS	PM	PB
NB	0.00	0.00	0.00	11.75	6.54	4.00	0.29
NM	0.00	0.00	0.17	11.88	6.96	2.95	-2.70
NS	0.00	0.00	1.74	1.99	-0.48	-0.86	-0.60
Z	0.00	0.00	4.82	0.07	-2.62	-0.06	0.00
PS	0.02	3.42	1.51	-2.89	-2.50	0.00	0.00
PM	1.16	-2.41	-7.91	-11.20	-0.64	0.00	0.00
PB	0.28	-2.52	-7.43	-11.32	0.00	0.00	0.00



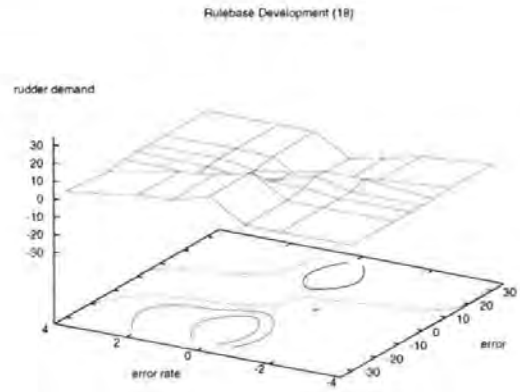
the changed rulebase at 11:33:20.39 1(16)

	NB	NM	NS	Z	PS	PM	PB
NB	0.00	0.00	0.00	11.75	6.54	4.00	0.29
NM	0.00	0.00	0.17	11.88	6.96	2.95	-2.70
NS	0.00	0.00	1.74	1.99	-0.47	-0.86	-0.60
Z	0.00	0.00	4.82	0.35	-2.61	-0.06	0.00
PS	0.02	3.42	1.51	-2.88	-2.50	0.00	0.00
PM	1.16	-2.41	-7.91	-11.20	-0.64	0.00	0.00
PB	0.28	-2.52	-7.43	-11.32	0.00	0.00	0.00



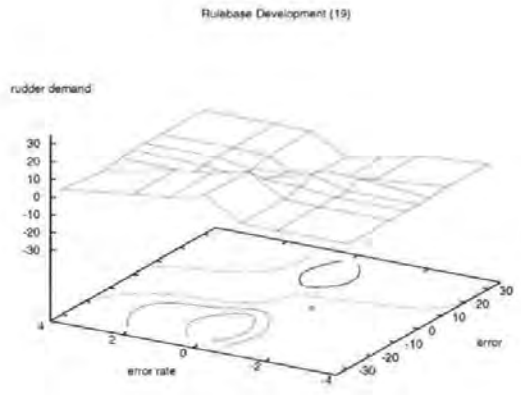
the changed rulebase at 11:33:50.43 (17)

	NB	NM	NS	Z	PS	PM	PB
NB	0.00	0.00	0.16	12.41	6.97	5.41	0.55
NM	0.00	0.00	0.37	12.44	7.36	5.34	-2.23
NS	0.00	0.00	2.06	2.18	-0.77	-1.46	-0.56
Z	0.00	0.00	5.23	0.47	-3.19	-1.04	0.00
PS	0.02	3.42	1.74	-3.17	-3.00	-0.02	0.00
PM	1.16	-2.41	-7.89	-11.29	-0.75	0.00	0.00
PB	0.28	-2.52	-7.43	-11.32	0.00	0.00	0.00



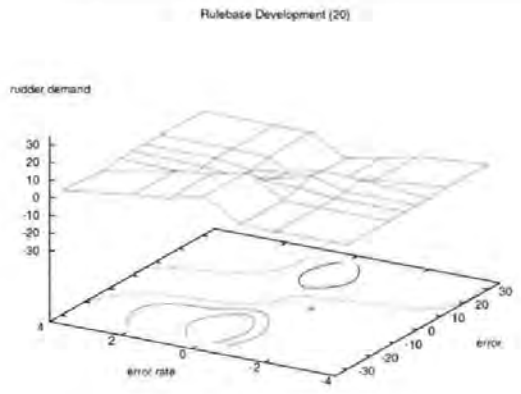
the changed rulebase at 11:34:20.43 (18)

	NB	NM	NS	Z	PS	PM	PB
NB	0.00	0.00	0.16	12.41	6.97	5.41	0.55
NM	0.00	0.00	0.37	12.44	7.36	5.34	-2.23
NS	0.00	0.00	2.06	2.34	-0.76	-1.46	-0.56
Z	0.00	0.00	5.36	-0.54	-3.47	-1.04	0.00
PS	0.02	3.42	3.89	-0.58	-2.99	-0.02	0.00
PM	1.16	-2.41	-7.45	-10.78	-0.73	0.00	0.00
PB	0.28	-2.52	-7.43	-11.32	0.00	0.00	0.00



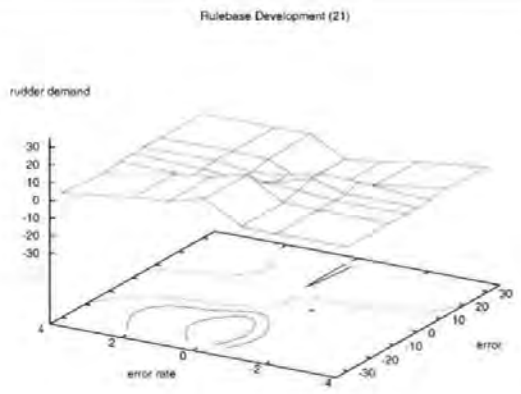
the changed rulebase at 11:34:50.43 (19)

	NB	NM	NS	Z	PS	PM	PB
NB	0.00	0.00	0.16	12.41	6.97	5.41	0.55
NM	0.00	0.00	0.37	12.44	7.36	5.34	-2.23
NS	0.00	0.00	2.05	2.48	-0.70	-1.46	-0.56
Z	0.00	0.00	5.36	0.31	-3.40	-1.04	0.00
PS	0.02	3.42	3.89	-0.60	-2.99	-0.02	0.00
PM	1.16	-2.41	-7.45	-10.78	-0.73	0.00	0.00
PB	0.28	-2.52	-7.43	-11.32	0.00	0.00	0.00



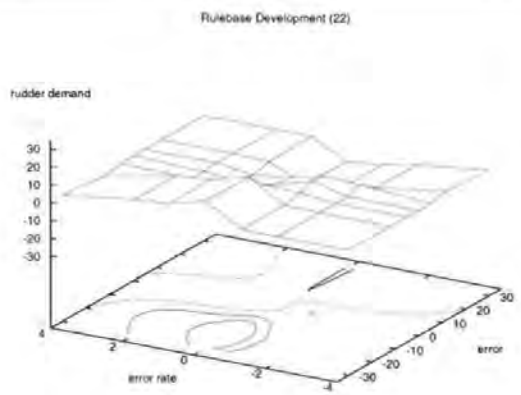
the changed rulebase at 11:35:20.51 (20)

	NB	NM	NS	Z	PS	PM	PB
NB	0.00	0.00	0.16	12.41	6.97	5.41	0.55
NM	0.00	0.00	0.37	12.44	7.36	5.34	-2.23
NS	0.00	0.00	2.05	2.48	-0.70	-1.46	-0.56
Z	0.00	0.00	5.35	0.03	-3.41	-1.04	0.00
PS	0.02	3.42	3.88	-0.62	-2.99	-0.02	0.00
PM	1.16	-2.41	-7.45	-10.78	-0.73	0.00	0.00
PB	0.28	-2.52	-7.43	-11.32	0.00	0.00	0.00



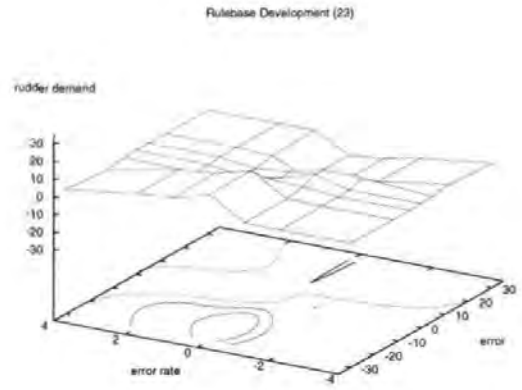
the changed rulebase at 11:35:50.51 (21)

	NB	NM	NS	Z	PS	PM	PB
NB	0.00	0.00	0.16	12.41	6.97	5.41	0.55
NM	0.00	0.00	0.37	12.44	7.36	5.34	-2.23
NS	0.00	0.00	2.02	2.43	-0.70	-1.46	-0.56
Z	0.00	0.00	5.24	-0.89	-3.42	-1.04	0.00
PS	0.31	3.08	3.98	-0.59	-2.99	-0.02	0.00
PM	2.11	-4.19	-8.25	-9.97	-0.74	0.00	0.00
PB	0.20	-4.41	-8.26	-10.40	-0.01	0.00	0.00



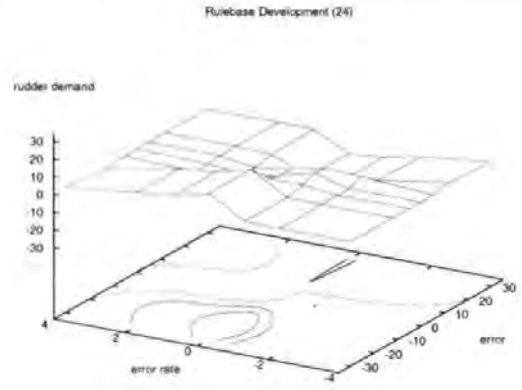
the changed rulebase at 11:36:20.53 (22)

	NB	NM	NS	Z	PS	PM	PB
NB	0.00	0.00	0.16	12.41	6.97	5.41	0.55
NM	0.00	0.00	0.37	12.44	7.36	5.34	-2.23
NS	0.00	0.00	1.99	2.41	-0.69	-1.46	-0.56
Z	0.00	0.00	5.20	-0.94	-3.40	-1.04	0.00
PS	0.31	3.08	3.98	-0.59	-2.99	-0.02	0.00
PM	2.11	-4.19	-8.25	-9.97	-0.74	0.00	0.00
PB	0.20	-4.41	-8.26	-10.40	-0.01	0.00	0.00



the changed rulebase at 11:36:50.54 (23)

	NB	NM	NS	Z	PS	PM	PB
NB	0.00	0.00	0.16	12.41	6.97	5.41	0.55
NM	0.00	0.00	0.37	12.44	7.36	5.34	-2.23
NS	0.00	0.00	1.96	2.49	-0.65	-1.46	-0.56
Z	0.00	0.00	5.18	-0.68	-3.36	-1.04	0.00
PS	0.31	3.08	3.98	-0.59	-2.99	-0.02	0.00
PM	2.11	-4.19	-8.25	-9.97	-0.74	0.00	0.00
PB	0.20	-4.41	-8.26	-10.40	-0.01	0.00	0.00

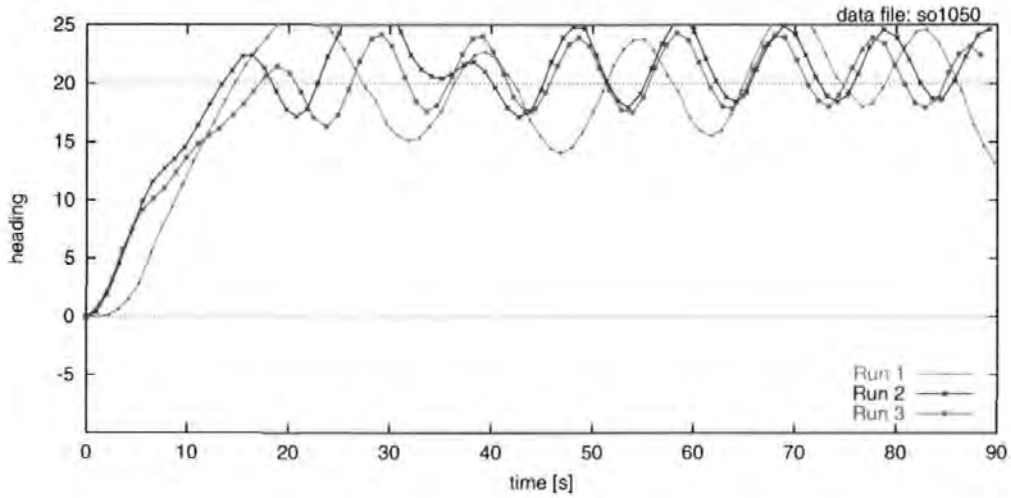


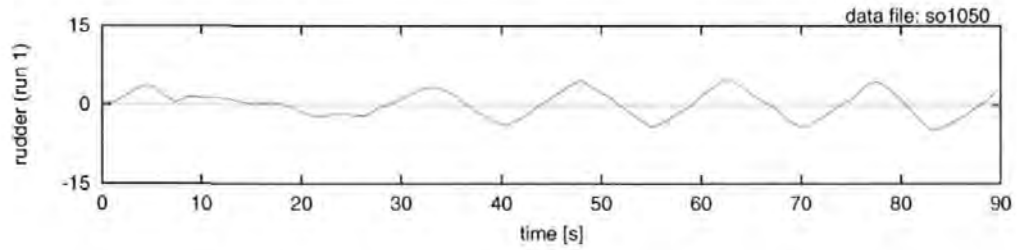
the changed rulebase at 11:37:37.02 (24)

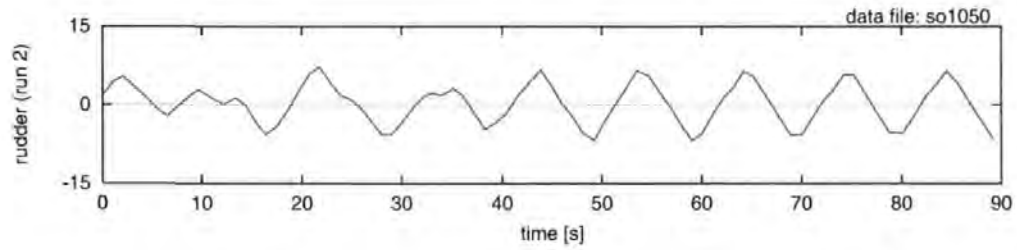
	NB	NM	NS	Z	PS	PM	PB
NB	0.00	0.00	0.16	12.41	6.97	5.41	0.55
NM	0.00	0.00	0.37	12.44	7.36	5.34	-2.23
NS	0.00	0.00	1.96	2.57	-0.61	-1.46	-0.56
Z	0.00	0.00	5.17	-0.30	-3.32	-1.04	0.00
PS	0.31	3.08	3.98	-0.59	-2.99	-0.02	0.00
PM	2.11	-4.19	-8.25	-9.97	-0.74	0.00	0.00
PB	0.20	-4.41	-8.26	-10.40	-0.01	0.00	0.00

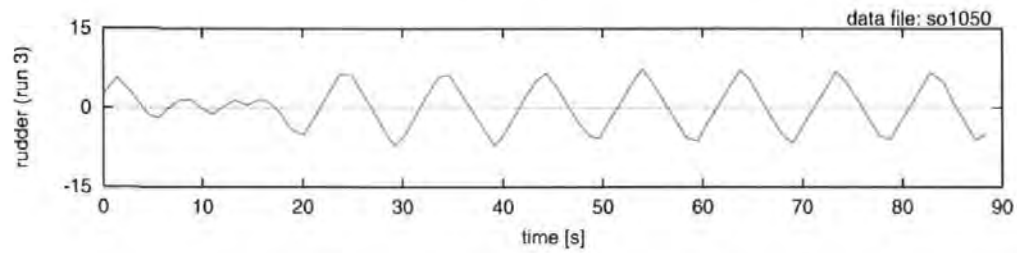
E.2 SoFLC

E.2.1 Measured Values

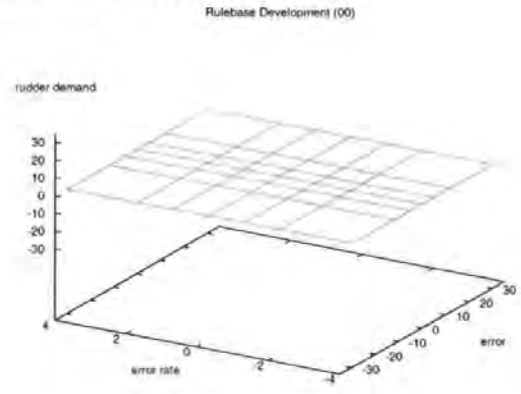






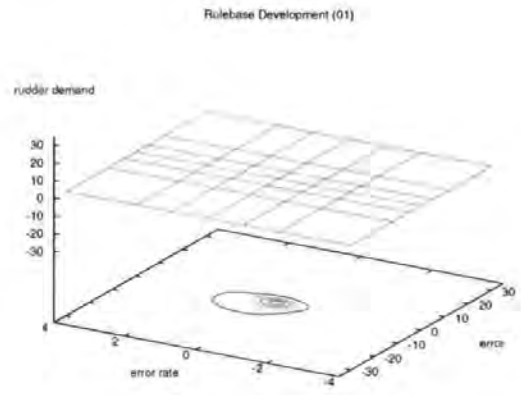


E.2.2 The Rulebases



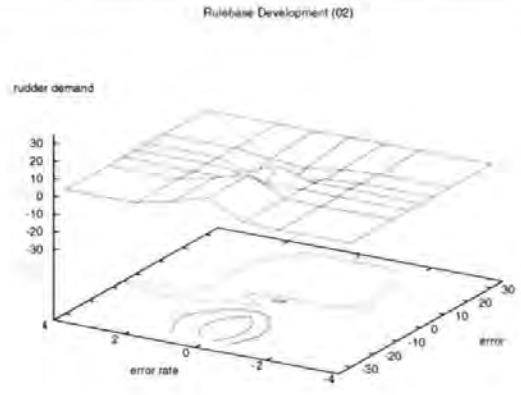
the preset rulebase

	NB	NM	NS	Z	PS	PM	PB
NB	0.00	0.00	0.00	0.00	0.00	0.00	0.00
NM	0.00	0.00	0.00	0.00	0.00	0.00	0.00
NS	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Z	0.00	0.00	0.00	0.00	0.00	0.00	0.00
PS	0.00	0.00	0.00	0.00	0.00	0.00	0.00
PM	0.00	0.00	0.00	0.00	0.00	0.00	0.00
PB	0.00	0.00	0.00	0.00	0.00	0.00	0.00



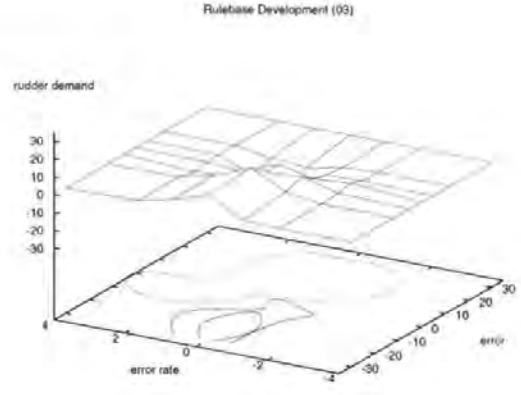
the changed rulebase at 17:25:44.36 (1)

	NB	NM	NS	Z	PS	PM	PB
NB	0.00	0.00	0.00	0.00	0.00	0.00	0.00
NM	0.00	0.00	0.00	0.00	0.00	0.00	0.00
NS	0.00	0.00	0.00	0.01	0.01	0.00	0.00
Z	0.00	0.00	0.00	0.03	0.01	0.00	0.00
PS	0.00	0.00	0.00	0.00	0.00	0.00	0.00
PM	0.00	0.00	0.00	0.00	0.00	0.00	0.00
PB	0.00	0.00	0.00	0.00	0.00	0.00	0.00



the changed rulebase at 17:26:14.36 (2)

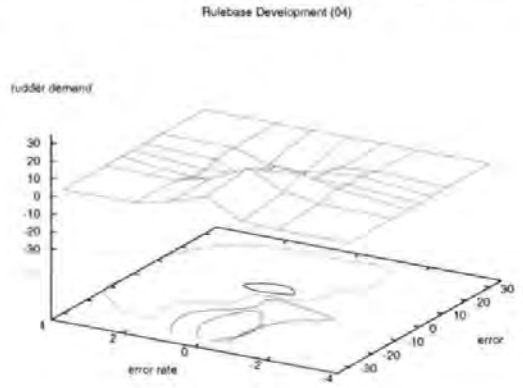
	NB	NM	NS	Z	PS	PM	PB
NB	0.00	0.00	0.71	11.23	4.77	0.17	0.00
NM	0.00	0.00	0.81	10.13	4.80	0.50	0.00
NS	0.00	0.00	0.61	0.15	-0.56	-0.70	0.00
Z	0.00	0.74	0.56	6.33	-0.73	0.00	0.00
PS	0.00	0.19	-1.25	-5.69	-5.77	0.00	0.00
PM	0.00	0.00	-1.01	-4.11	-2.41	0.00	0.00
PB	0.00	0.00	0.00	0.00	0.00	0.00	0.00



the changed rulebase at 17:26:44.36 (3)

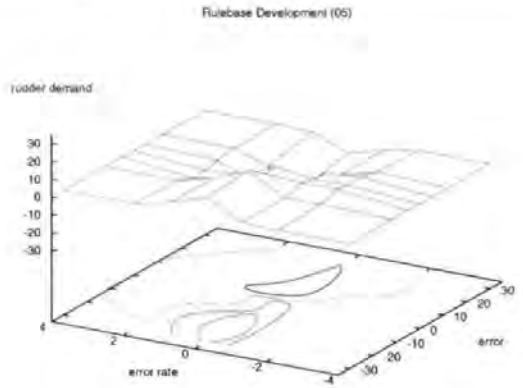
	NB	NM	NS	Z	PS	PM	PB
NB	0.00	0.00	0.71	11.23	4.77	0.17	0.00
NM	0.00	0.43	3.44	13.22	4.95	0.14	0.00
NS	0.00	1.03	6.75	5.07	-0.59	-2.24	0.00
Z	0.00	3.10	1.63	6.22	-1.71	-0.58	0.00
PS	0.00	0.68	0.04	-8.58	-8.37	0.00	0.00
PM	0.00	0.16	-0.91	-4.97	-3.13	0.00	0.00
PB	0.00	0.00	0.00	0.00	0.00	0.00	0.00

The adaptation of the rules is not as rapid as can be observed when the PSoFLC is operating. This is seen by the differences between the rulebases. It appears that the rulebase does not settle, furthermore, it is constantly updated.



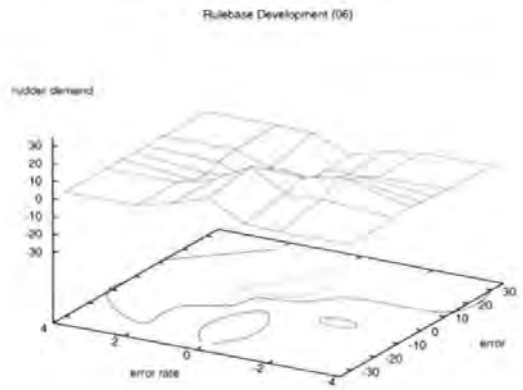
the changed rulebase at 17:27:14.36 (4)

	NB	NM	NS	Z	PS	PM	PB
NB	0.00	0.00	0.71	11.23	4.77	0.17	0.00
NM	0.34	1.41	4.36	13.85	4.39	-0.33	0.00
NS	0.42	4.48	9.41	6.66	-1.97	-4.22	0.00
Z	0.00	3.10	2.53	6.08	-2.17	-3.62	-0.20
PS	0.40	4.18	1.95	-11.51	-12.85	-1.96	-0.20
PM	0.15	1.44	-0.07	-6.90	-5.65	-0.40	0.00
PB	0.00	0.00	0.00	0.00	0.00	0.00	0.00



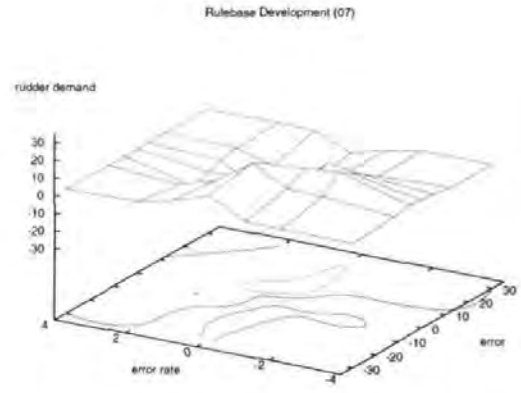
the changed rulebase at 17:27:44.36 (5)

	NB	NM	NS	Z	PS	PM	PB
NB	0.00	0.00	0.71	11.23	4.77	0.17	0.00
NM	0.34	1.41	1.07	12.91	4.39	-0.33	0.00
NS	0.42	4.48	4.45	5.72	-1.97	-4.22	0.00
Z	0.00	3.10	2.53	6.08	-2.17	-3.62	-0.20
PS	0.40	4.18	1.95	-11.51	-12.85	-1.96	-0.20
PM	0.15	1.44	-3.89	-14.96	-6.36	-0.40	0.00
PB	0.00	0.00	-3.45	-8.98	-0.71	0.00	0.00



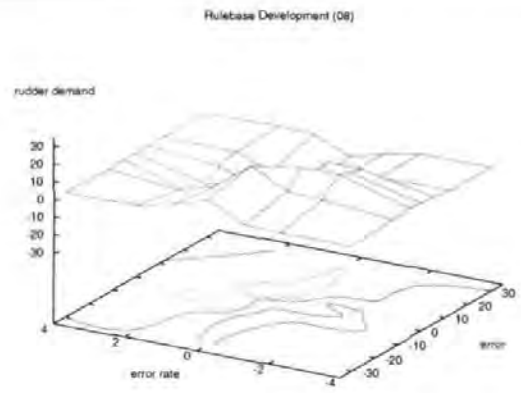
the changed rulebase at 17:28:14.36 (6)

	NB	NM	NS	Z	PS	PM	PB
NB	0.00	0.00	0.71	11.23	4.77	0.17	0.00
NM	1.85	4.90	4.01	15.63	4.67	-0.56	0.00
NS	3.15	12.47	8.06	8.85	-2.49	-5.23	0.00
Z	1.93	7.34	3.31	6.08	-2.57	-4.82	-0.20
PS	1.56	4.92	5.00	-11.64	-14.15	-2.08	-0.20
PM	0.15	-0.80	-4.47	-14.74	-6.77	-0.40	0.00
PB	0.00	-0.42	-3.56	-8.98	-0.71	0.00	0.00



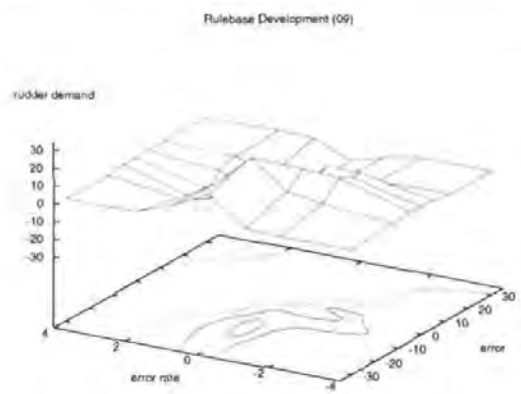
the changed rulebase at 17:28:44.36 (7)

	NB	NM	NS	Z	PS	PM	PB
NB	0.00	0.00	0.71	11.23	4.77	0.17	0.00
NM	1.85	5.14	5.47	16.69	2.62	-3.46	-0.55
NS	3.15	13.84	11.83	10.56	-5.16	-10.36	-0.55
Z	1.93	9.66	3.69	5.73	-3.98	-5.15	-0.20
PS	1.56	5.23	7.57	-10.51	-15.60	-2.31	-0.20
PM	0.15	-0.80	-4.06	-14.56	-6.98	-0.40	0.00
PB	0.00	-0.42	-3.56	-8.98	-0.71	0.00	0.00



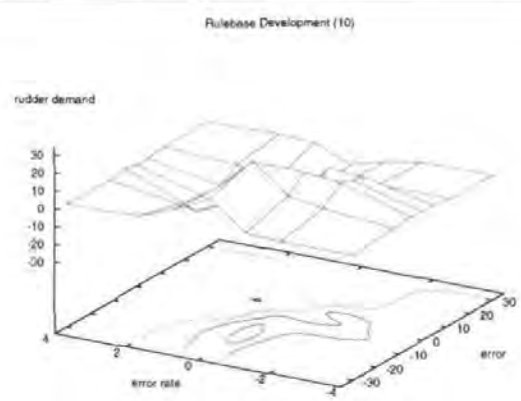
the changed rulebase at 17:29:14.36 (8)

	NB	NM	NS	Z	PS	PM	PB
NB	0.00	0.00	0.71	11.23	4.77	0.17	0.00
NM	1.85	5.19	6.11	16.99	1.56	-4.41	-0.55
NS	3.15	14.48	14.38	10.96	-8.14	-12.82	-0.72
Z	2.03	11.61	3.69	5.73	-3.98	-10.25	-0.37
PS	1.66	9.92	10.23	-10.81	-18.46	-3.46	-0.20
PM	0.15	0.20	-2.75	-14.69	-8.03	-0.83	0.00
PB	0.00	-0.42	-3.56	-8.98	-0.71	0.00	0.00



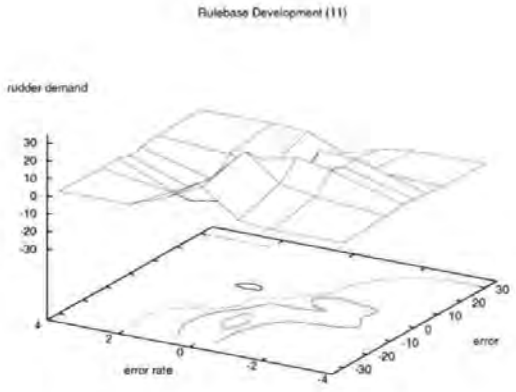
the changed rulebase at 17:29:44.36 (9)

	NB	NM	NS	Z	PS	PM	PB
NB	0.00	0.00	1.38	16.68	6.64	-0.65	-0.72
NM	1.85	5.19	8.20	24.11	5.15	-6.49	-1.51
NS	3.15	14.48	15.80	16.34	-6.50	-12.61	-0.56
Z	2.03	11.61	3.69	5.73	-3.98	-10.25	-0.37
PS	1.66	9.92	10.23	-10.81	-18.46	-3.46	-0.20
PM	0.15	0.20	-2.75	-14.69	-8.03	-0.83	0.00
PB	0.00	-0.42	-3.56	-8.98	-0.71	0.00	0.00



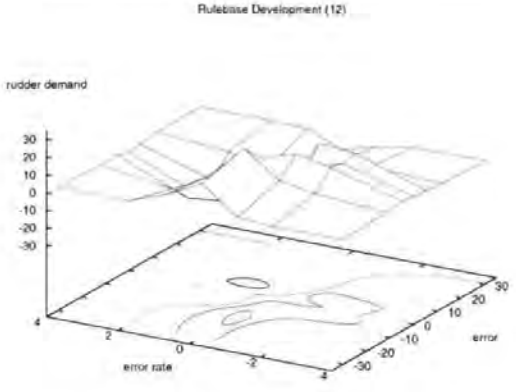
the changed rulebase at 17:30:14.37 (10)

	NB	NM	NS	Z	PS	PM	PB
NB	0.00	0.00	1.38	16.68	6.64	-0.65	-0.72
NM	1.85	5.19	8.32	24.56	5.20	-7.42	-1.51
NS	3.15	14.48	16.33	16.65	-7.62	-16.34	-0.56
Z	2.03	12.33	4.13	4.82	-5.25	-12.77	-2.47
PS	1.69	10.49	11.45	-12.57	-21.25	-9.79	-2.27
PM	0.18	0.75	-2.02	-15.37	-9.80	-3.05	-0.89
PB	0.00	-0.42	-3.56	-8.98	-0.71	0.00	0.00



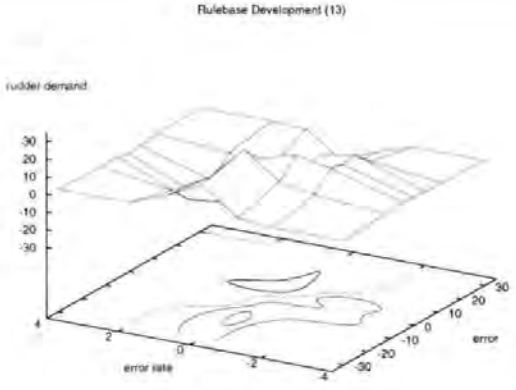
the changed rulebase at 17:30:44.37 (11)

	NB	NM	NS	Z	PS	PM	PB
NB	0.00	0.00	1.38	16.68	6.64	-0.65	-0.72
NM	1.85	5.19	8.66	23.32	3.69	-7.42	-1.51
NS	3.15	14.61	18.19	13.37	-14.16	-17.43	-1.24
Z	2.03	12.97	6.59	5.04	-5.25	-17.20	-3.37
PS	1.83	13.87	14.69	-13.37	-24.07	-13.08	-2.59
PM	0.31	2.06	0.14	-15.87	-11.10	-3.94	-1.00
PB	0.00	-0.42	-3.56	-8.98	-0.71	0.00	0.00



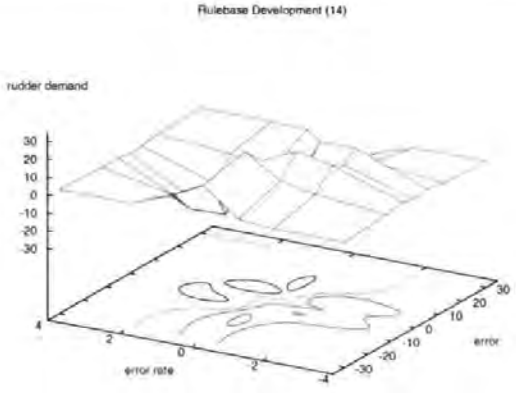
the changed rulebase at 17:31:14.38 (12)

	NB	NM	NS	Z	PS	PM	PB
NB	0.00	0.00	1.38	16.68	6.64	-0.65	-0.72
NM	1.85	5.19	8.68	23.15	3.50	-7.42	-1.51
NS	3.15	14.61	18.56	12.60	-15.74	-17.87	-1.24
Z	2.03	13.24	7.12	5.04	-7.47	-20.14	-3.37
PS	1.83	15.35	17.09	-13.50	-27.77	-14.74	-2.59
PM	0.31	2.43	1.07	-16.03	-12.55	-4.32	-1.00
PB	0.00	-0.42	-3.56	-8.98	-0.71	0.00	0.00



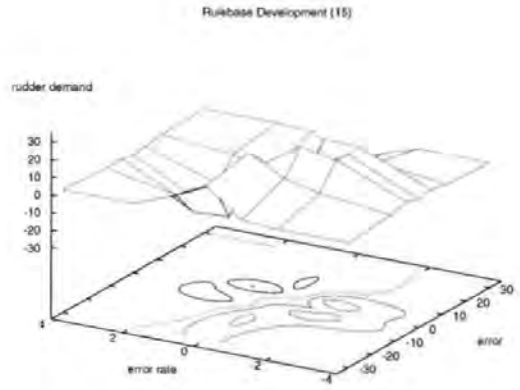
the changed rulebase at 17:31:44.38 (13)

	NB	NM	NS	Z	PS	PM	PB
NB	0.00	0.00	1.38	16.68	6.64	-0.65	-0.72
NM	1.85	5.19	8.68	23.15	3.50	-7.42	-1.51
NS	3.15	14.61	18.56	12.60	-15.74	-17.87	-1.24
Z	2.03	13.24	7.12	5.04	-7.47	-20.14	-3.37
PS	1.83	14.77	10.93	-21.00	-27.77	-14.74	-2.59
PM	1.53	5.94	-5.63	-24.85	-12.55	-4.32	-1.00
PB	0.72	0.58	-5.72	-14.49	-0.71	0.00	0.00



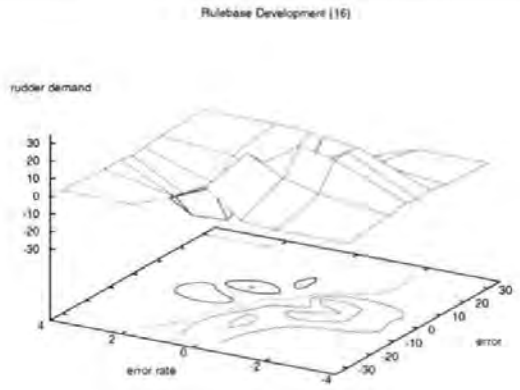
the changed rulebase at 17:32:14.38 (14)

	NB	NM	NS	Z	PS	PM	PB
NB	0.00	0.00	1.38	16.68	6.64	-0.65	-0.72
NM	2.16	5.86	9.85	22.29	0.51	-9.23	-1.87
NS	3.88	16.77	20.96	11.96	-21.96	-22.83	-1.73
Z	2.03	13.24	7.64	5.44	-8.45	-24.16	-3.37
PS	4.03	19.06	18.85	-18.69	-30.33	-15.88	-2.59
PM	1.94	6.35	-3.14	-22.91	-13.30	-4.52	-1.00
PB	0.72	0.58	-5.72	-14.49	-0.71	0.00	0.00



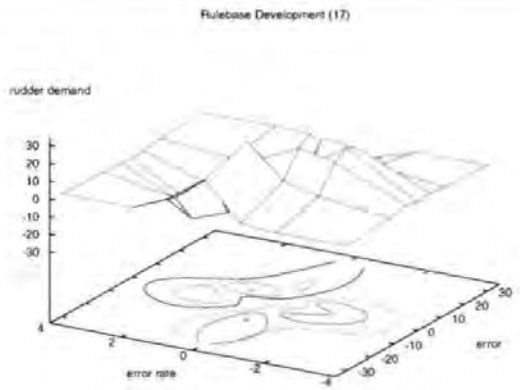
the changed rulebase at 17:32:44.38 (15)

	NB	NM	NS	Z	PS	PM	PB
NB	0.00	0.00	1.38	16.68	6.64	-0.65	-0.72
NM	2.44	7.05	11.54	22.23	-0.69	-9.64	-2.01
NS	4.54	20.16	23.91	11.58	-23.45	-24.42	-1.87
Z	2.25	19.82	9.58	4.93	-9.31	-24.61	-3.37
PS	4.03	20.28	21.24	-17.70	-30.98	-15.88	-2.59
PM	1.94	6.35	-2.87	-22.48	-13.35	-4.52	-1.00
PB	0.72	0.58	-5.72	-14.49	-0.71	0.00	0.00



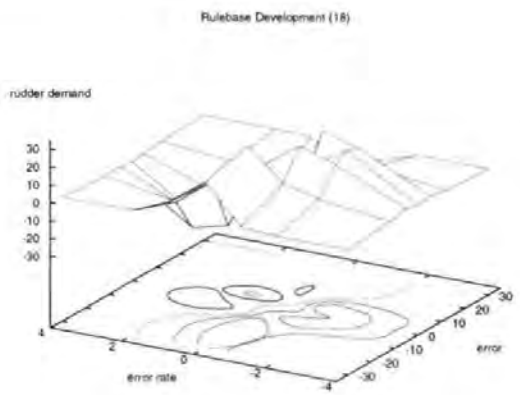
the changed rulebase at 17:33:14.38 (16)

	NB	NM	NS	Z	PS	PM	PB
NB	0.00	0.00	1.38	16.68	6.64	-0.65	-0.72
NM	2.44	7.24	12.05	22.46	-1.27	-9.99	-2.01
NS	4.54	20.98	25.25	12.11	-24.55	-25.74	-1.87
Z	2.25	21.97	10.29	4.93	-10.32	-25.53	-3.37
PS	4.03	20.50	22.97	-16.61	-31.55	-15.88	-2.59
PM	1.94	6.35	-2.51	-22.22	-13.43	-4.52	-1.00
PB	0.72	0.58	-5.72	-14.49	-0.71	0.00	0.00



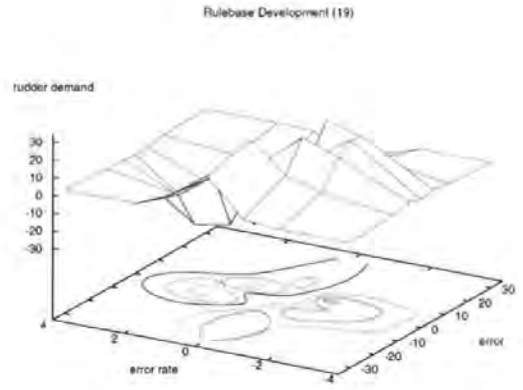
the changed rulebase at 17:33:44.38 (17)

	NB	NM	NS	Z	PS	PM	PB
NB	0.00	0.00	1.38	22.34	8.18	-0.60	-0.62
NM	2.44	7.24	12.05	30.55	6.85	-8.64	-1.91
NS	4.54	20.98	25.25	15.71	-18.90	-25.74	-1.87
Z	2.25	25.54	11.72	4.76	-10.74	-25.53	-3.37
PS	4.03	21.21	23.68	-16.61	-31.55	-15.88	-2.59
PM	1.94	6.35	-2.51	-22.22	-13.43	-4.52	-1.00
PB	0.72	0.58	-5.72	-14.49	-0.71	0.00	0.00



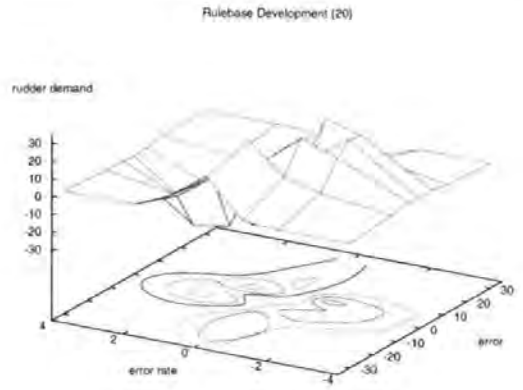
the changed rulebase at 17:34:14.38 (18)

	NB	NM	NS	Z	PS	PM	PB
NB	0.00	0.00	1.38	22.34	8.18	-0.60	-0.62
NM	2.44	7.36	12.70	29.58	5.37	-9.36	-2.35
NS	4.54	21.62	27.89	14.00	-24.08	-29.35	-2.31
Z	2.25	25.88	13.51	4.67	-10.84	-30.08	-3.59
PS	4.23	23.20	29.13	-16.86	-34.15	-16.91	-2.81
PM	2.06	7.39	-0.41	-21.68	-14.50	-4.64	-1.00
PB	0.72	0.58	-5.72	-14.49	-0.71	0.00	0.00



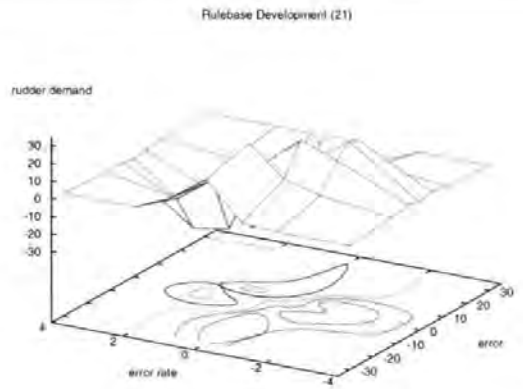
the changed rulebase at 17:34:44.38 (19)

	NB	NM	NS	Z	PS	PM	PB
NB	0.00	0.00	1.38	22.34	8.18	-0.60	-0.62
NM	2.44	7.36	12.94	29.21	4.75	-9.40	-2.35
NS	4.54	21.81	29.27	12.76	-27.43	-31.49	-2.31
Z	2.25	28.09	14.15	4.67	-11.19	-31.70	-3.59
PS	4.23	26.60	32.05	-17.15	-35.00	-18.67	-2.81
PM	2.06	8.35	1.11	-22.15	-15.76	-4.88	-1.00
PB	0.72	0.58	-5.72	-14.49	-0.71	0.00	0.00



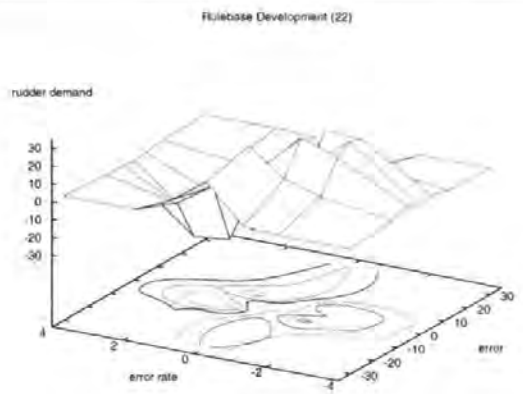
the changed rulebase at 17:35:14.38 (20)

	NB	NM	NS	Z	PS	PM	PB
NB	0.00	0.00	1.38	22.34	8.18	-0.60	-0.62
NM	2.44	7.36	12.97	29.24	4.74	-9.40	-2.35
NS	4.54	21.81	29.56	13.08	-27.95	-31.49	-2.31
Z	2.25	28.86	14.30	4.67	-11.19	-31.70	-3.59
PS	4.23	27.66	33.36	-17.00	-35.00	-19.02	-2.81
PM	2.06	8.70	1.66	-22.09	-15.98	-4.93	-1.00
PB	0.72	0.58	-5.72	-14.49	-0.71	0.00	0.00



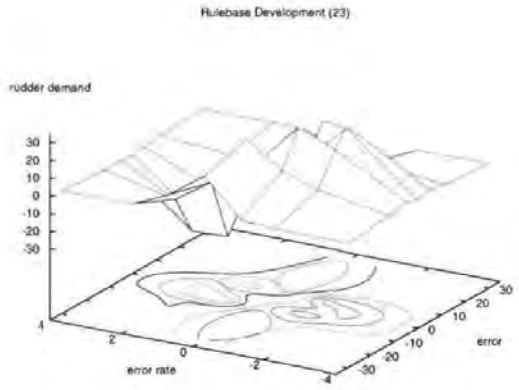
the changed rulebase at 17:35:44.38 (21)

	NB	NM	NS	Z	PS	PM	PB
NB	0.00	0.00	1.38	22.34	8.18	-0.60	-0.62
NM	2.44	7.36	12.97	29.24	4.74	-9.40	-2.35
NS	4.54	21.81	29.56	13.05	-27.98	-31.49	-2.31
Z	2.25	28.86	14.24	3.85	-11.92	-31.70	-3.59
PS	4.23	25.37	25.17	-21.55	-35.00	-19.02	-2.81
PM	1.94	6.37	-7.42	-30.06	-16.08	-4.93	-1.00
PB	0.60	0.28	-8.06	-19.83	-0.71	0.00	0.00



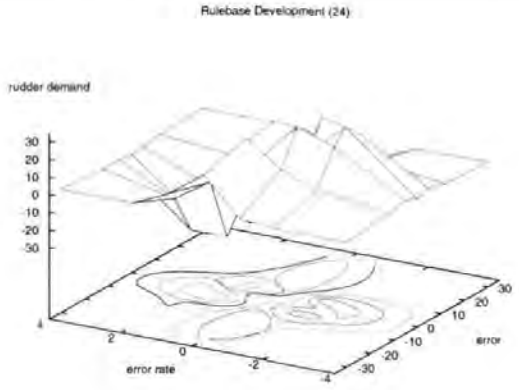
the changed rulebase at 17:36:14.38 (22)

	NB	NM	NS	Z	PS	PM	PB
NB	0.00	0.00	1.38	22.34	8.18	-0.60	-0.62
NM	2.60	8.28	14.44	29.25	2.78	-10.46	-2.46
NS	5.06	24.83	32.30	13.43	-32.30	-34.59	-2.43
Z	2.25	29.98	14.43	3.85	-13.11	-33.93	-3.59
PS	4.23	27.92	29.10	-20.38	-35.00	-19.05	-2.81
PM	1.94	6.54	-6.91	-29.89	-16.25	-4.93	-1.00
PB	0.60	0.28	-8.06	-19.83	-0.71	0.00	0.00



the changed rulebase at 17:36:44.38 (23)

	NB	NM	NS	Z	PS	PM	PB
NB	0.00	0.00	1.38	22.34	8.18	-0.60	-0.62
NM	2.71	8.72	15.44	29.74	1.55	-11.68	-2.48
NS	5.17	26.69	35.00	14.46	-34.37	-35.00	-2.45
Z	2.25	35.00	16.27	3.94	-14.28	-34.19	-3.59
PS	4.23	27.92	31.36	-19.27	-35.00	-19.05	-2.81
PM	1.94	6.54	-6.60	-29.64	-16.32	-4.93	-1.00
PB	0.60	0.28	-8.06	-19.83	-0.71	0.00	0.00



the changed rulebase at 17:37:36.73 (24)

	NB	NM	NS	Z	PS	PM	PB
NB	0.00	0.00	1.38	22.34	8.18	-0.60	-0.62
NM	2.71	8.85	15.74	29.76	1.33	-11.74	-2.48
NS	5.17	27.50	35.00	14.53	-35.00	-35.00	-2.45
Z	2.25	35.00	16.58	3.94	-15.05	-34.19	-3.59
PS	4.23	28.42	33.60	-18.49	-35.00	-19.38	-2.81
PM	1.94	6.54	-6.14	-29.40	-16.54	-4.95	-1.00
PB	0.60	0.28	-8.06	-19.83	-0.71	0.00	0.00

Appendix F

R. Rojas: *Theorie der neuronalen Netze – Eine systematische Einführung [88]*

(Translation)

Chapter 10 Fuzzy logic and neural networks

10.2.2 Fuzzy values and inverse operation (page 214 ff.)

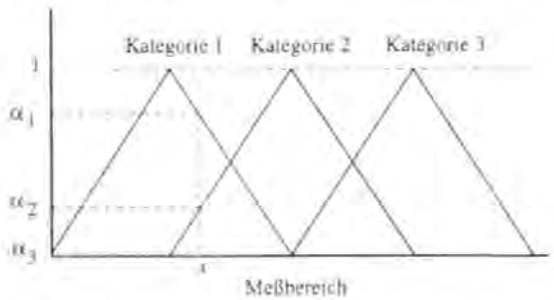


Abb. 10.10 Kategorisierung mit dreieckigen Funktionen

The transformation of the value x in a fuzzy category is achieved by reading off the membership values α_1, α_2, α_3 for the value x from the graphs representing the membership functions. Abb. 10.10 shows an example where α_1 and α_2 are positive whilst α_3 is zero.

It is important to be able to reconstruct the original value x from the membership values α_1, α_2 and α_3 by using the inverse operation. The *centre of gravity* [COG] method efforts the means of achieving this. Abb. 10.12 shows the dissection of the value x into the membership values α_1, α_2 and α_3. From α_1, α_2, α_3 one can re-construct x. For that purpose, the area under the triangle, up to the respective height α_1, α_2 and α_3, is calculated. It is fact that, when this method is used, the *horizontal component of the centre of gravity* of the total area

is a valid approximation of x.

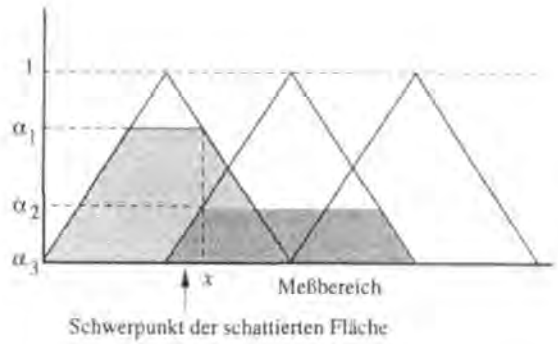


Abb. 10.12 Schwerpunktmethode

It is possible to calculate an approximation for x utilising this centre of gravity method for all values of x, for which at least two of the three numbers α_1, α_2 and α_3 are different from zero. Abb. 10.13 shows the deviation from x and its approximation for a categorisation, in which the basis of the triangle is 2 units in length, the height is as usual 1. The positioning of the sets corresponds to Abb. 10.12 and x lays between 1 and 2. The figure shows, that the maximal deviation from the true value can not be bigger than 10%.

Abb. 10.13

The quality of the results of the centre of gravity method is dependant on the appropriate positioning of the category triangles. When a weighted COG, which means differently weighted sets, is considered, different triangle combinations to the ones shown here can be used.

Schwerpunkt ... centre of gravity	Meßbereich ... universe of discourse
Kategorie ... category, *here* set	dreieckig ... triangular
Abweichung ... deviation	schattierte Fläche ... shaded area

Appendix G

Papers, Publications, Presentations

---

[1] R. S. Burns and R. Richter. Automatic guidance of ships using neural networks. Neural Network Seminar for Rolls Royce and Associates Ltd., September 1994.

[2] R. S. Burns and R. Richter. An introduction of neural networks for the control of small and large vessels. Neural Network Seminar for Rolls Royce and Associates Ltd., October 1994.

[3] R. S. Burns and R. Richter. The application of neural networks for the control of small and large vessels. In *Workshop on Control Application in Marine Systems (CAMS '95), Trondheim, Norway*, pages 393–399, 10–12 May 1995.

[4] R. S. Burns and R. Richter. The application of neural networks for the modelling of process dynamics. In Dr Martyn Polkinghorne, editor, *Applications of Artificial Intelligence for Technological and Business Processes*, volume 2 of *Technology Transfer Series*, pages 10–17. University of Plymouth, University of Plymouth, Drake Circus, Plymouth, Devon PL4 8AA, UK, 1996. ISBN 0-905227-57-3.

[5] R. S. Burns and R. Richter. A neural network approach to the control of surface ships. *Journal Control Engineering Practice*, 4(3):411–416, 1996. Special section on Control Applications in Marine Systems.

[6] R. S. Burns, R. Richter, and M. N. Polkinghorne. A multivariable neural network ship mathematical model. In *First International Conference on Marine Transport into the 21st Century (MarTrans '95)*, pages 747–756, Southampton - UK, 30 August – 1 September 1995. Computational Mechanics Publications, ISBN 1-85312-330-7. Also published by: Computational Mechanics Publications, Boston MA, 1995, ISBN 1-56252-254-x, pp 747-756.

[7] R. Richter and R. S. Burns. An artificial neural network autopilot for small vessels. In R. Poley and R. Zobel, editors, *Proc. of the First Conference of the UKSS'93*, pages 168–172, Keswick Hotel, Keswick, Cumbria, UK, 13–15 September 1993. United Kingdom Simulation Society. ISBN 0 9516509 1 2.

[8] R. Richter, R. S. Burns, and M. N. Polkinghorne. Application of an artificial neural network to model complex plant behaviour. In *IFAC '95 - Workshop on Motion Control, Munich, Germany*, 9–11 October 1995.

[9] R. Richter, R. S. Burns, and M. N. Polkinghorne. A review of the historical development of ship motion control and simulation. In *Symposium: Applications of Advanced Marine Control*, pages 1–8, Plymouth, UK, 30 November 1995. ISBN 0-905227-46-8, also published in *Applications of Artificial Intelligence for Technological and Business Processes, Technology Transfer Series* vol (1), 1996, ISBN 0-905227-53-0.

[10] R. Richter, R. S. Burns, M. N. Polkinghorne, and P. Nurse. Modelling and control of surface ships by employing a fuzzy-neural solution. In *11th International Conference on Systems Engineering (ICSE '96)*, pages 7–12, Howard R. Huges College of Engineering, University of Nevada - Las Vegas, Box 454005, Las Vegas, NV 89154-4005, USA, 9–11 July 1996.

[11] R. Richter, R. S. Burns, M. N. Polkinghorne, and P. Nurse. A predictive ship control using a fuzzy-neural autopilot. In P. A. Wilson, editor, *Proc. of 11th Ship Control Systems Symposium*, pages 161–172, Southampton, UK, April 1997.

[12] P. Robinson, Peter Nurse, Steve Roberts, Ralph Richter, Guido Bugman, and Roland Burns. Single site myoelectric control of a complex robotic hand. In J. O. Gray, editor,

Workshop on Advanced Robotics and Intelligent Machines, page paper number 8, University of Salford, Dept. of Electronic and Electrical Engineering, Salford, M5 4WT, UK, 25–26 March 1997. University of Salford. ISSN 1363-2698.

R. S. Burns and R. Richter.

An introduction of neural networks for the control of small and large vessels.

Neural Network Seminar for Rolls Royce and Associates Ltd., October 1994.

An Introduction to and the Application of Neural Networks for the Control of Small and Large Vessels

Dr Roland BURNS
and
Mr Ralph RICHTER

School of Manufacturing, Materials
and Mechanical Engineering

University of Plymouth

United Kingdom

1. Introduction to Neural Networks

The brain is the most complex structure we know. Its powerful capabilities, like thinking, remembering, problem-solving and learning, are very fascinating to model. We use artificial neural nets to simulate the behaviour of the human brain such as learning and recall of patterns. First applications were developed for pattern recognition in the early 1940's. The first principles were published by Frank Rosenblatt in 1957. He developed an element called *perceptron*, as shown in figure 1, which attracted attention in the world of neural computing. His perceptron is a device to recognise abstract and geometric patterns.

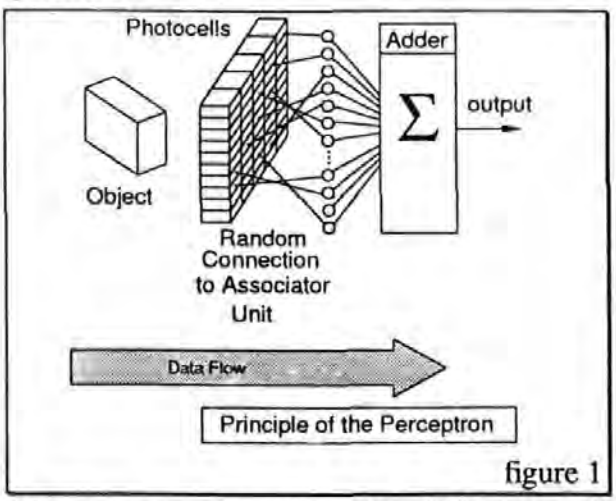


figure 1

The perceptron consists of a 400 photocell grid and was mainly developed for optical pattern recognition. The electrical output of the photocells were collected by the associator unit passing the random connections. The new multi layer system, developed in the 1960's, could learn and recall complex tasks. A non linear transfer function was used.

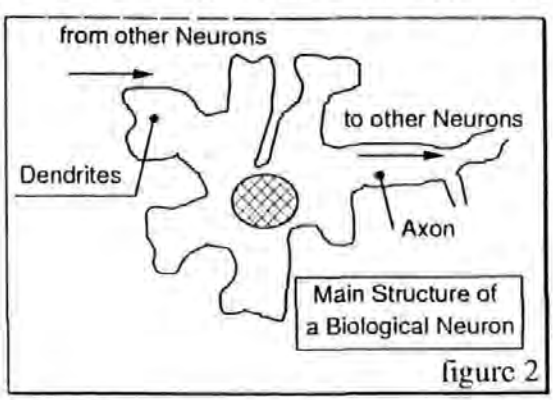


figure 2

To understand actions and algorithms in neural computing it is necessary to look at biological neural nets and their architecture. A *neuron* is the basic element of the brain. A diagram of a neuron is detailed in figure 2.

The structure of the brain is an interconnection of a very large (tens of billions) number of neurons. The transmission of signals in the brain is chemical in nature. Each neuron receives an input signal from other neighbouring neurons. The connection path between two neurons is called an *axon* and the incoming ports *dendrites*.

The connections between axons and dendrites are called *synapses* (see figure 3). In order to understand the biological model, the axon is an electrical cable and the dendrites is a socket. To carry information a link is needed. The synapse, the link or plug, changes the effectiveness of the incoming spike.

During a learning phase the efficiency of the synapse is modified. The sum of the incoming signals, the *total input*, is used by the receiving neuron to generate an output. This output of one neuron is the input for many other neurons except those neurons in the

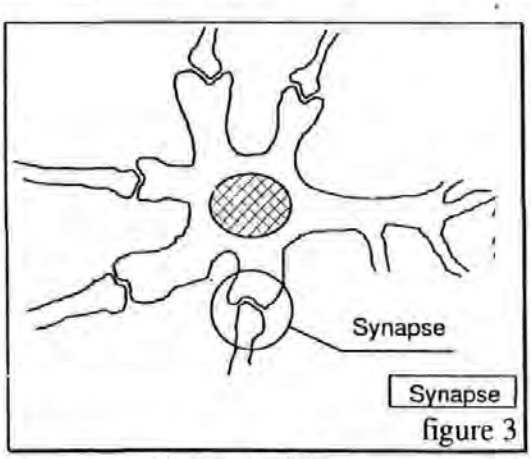


figure 3

output layer. The artificial neuron is a simple model of the biological neuron which has the form as displayed in figure 3.

The denotation of the signals depends on your point of view. Assuming the present neuron, all incoming signals are called x and the output is called y, this y, or output, is then an incoming signal for the next neuron and is then called x (figure 4).

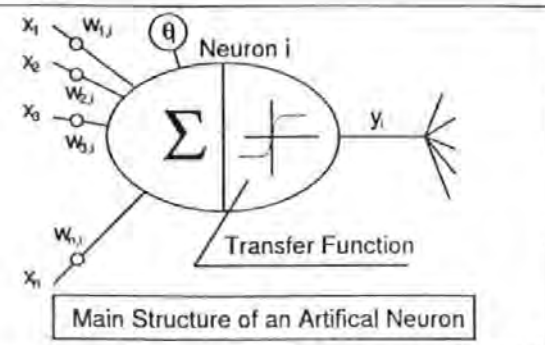


figure 4

As you can see, the synapse is modelled as a modifiable weight which is associated which each axon (connection to a neuron). The neurons output formed by the transfer function is a single number that represents the rate of firing - the activity of the neuron. To compute the output, the neuron multiplies each incoming signal by the associated weight and adds together all these weighted inputs to form the total input and uses this to create the output by using the transfer function. The reaction of the artificial network depends on both the transfer used function and the weights.

The output of the neuron in the mathematical sense is defined as:

$$I_i^k = \sum_{j=1} x_j^{k-1} \cdot w_{j,i}^k + \theta_i^k \qquad (1)$$

θ^k_i ..the threshold, which moves the transfer function (graph) in the horizontal direction.

x^{k-1}_j output of neuron j in the previous layer

$w^k_{j,i}$ weight between neuron i in layer k and the neuron j in layer k-1

I^k_i ...total input of neuron i in layer k

$y_i^k = f(I_i^k)$ where $f(I_i^k)$ could be:

linear $f(I_i^k) = I_i^k$

Sigmoid function $f(I_i^k) = \dfrac{1}{1+e^{-I_i^k}}$

hyperbolic tangent $f(I_i^k) = \tanh(I_i^k)$

$$f(I_i^k) = \begin{cases} -1 & I_i^k \le 0 \\ +1 & I_i^k > 0 \end{cases}$$ hard limiter or threshold function

2. Types of Networks

2.1 Architecture of Single Layer Nets

In the past, many forms of neural nets and their algorithms were investigated. Serious investigations started in 1943, by the head neurobiologist Warren McColloch and statistician Walter Pitt. The paper "A Logical Calculus of Ideas Imminent in Nervous Activity" brings together fields such as digital computing, "electronic brains" and macroscopic intelligence. The first conference on artificial intelligence was organised in 1956 by famous names such as Marvin Minsky, John McCarthy, Claude Shannon and Nathanial Rochester.

To simulate the behaviour of the human brain we need a *network* of neurons, a socalled neural network (net). The neurons are usually organised into groups called layers. A neural net consists of at least an input and an output layer and eventually hidden layer(s). In order to understand the following facts, with 'single' we mean the number of hidden layers. Actually, a single layer net consists of three layers, one input and one output layer and a *single* hidden layer. The words *one* and *single* are synonyms for each other. Simple tasks can be solved by a one layer network but for difficult problems we need *multi layer nets*. The main structure of a single layer net is shown below.

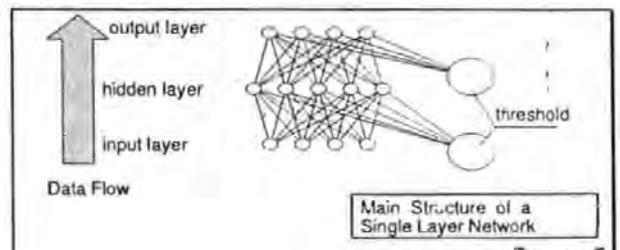


figure 5

The interconnection between the neurons in different layers can be seen in figure 5. It is not necessary to have a net where the connections are only between neurons of different layers, but it is easier to understand and to design a net in this way. The majority of modern neural nets are organised in this way. Some tasks do not require hidden layers. The number of hidden layers and the number

of neurons in each hidden layer is free to define and will determine the performance of the net in speed and quality. For the majority of tasks a single layer net is sufficient.

2.2 Multi Layer Nets

The behaviour of a multi layer net (see figure 6) in general is not very different to a single layer net. The user has to find a optimum in size to be satisfied with the

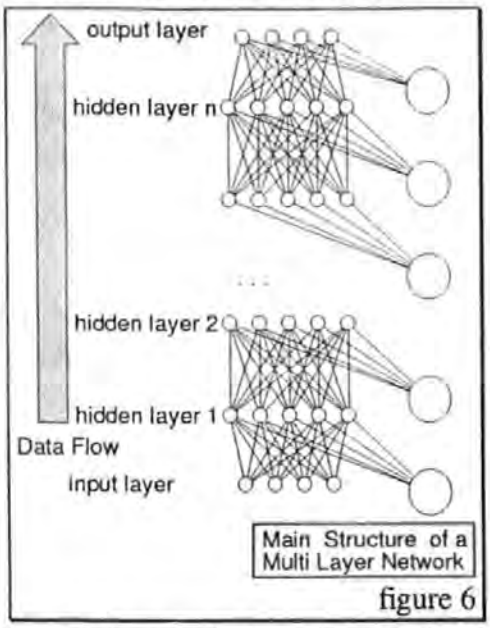


figure 6

results and the speed. A small net is faster but if the task is too difficult, important information may be lost. Conversely, if the net is too large, the output can be noisy and the computing speed, especially during the learning, is slow.

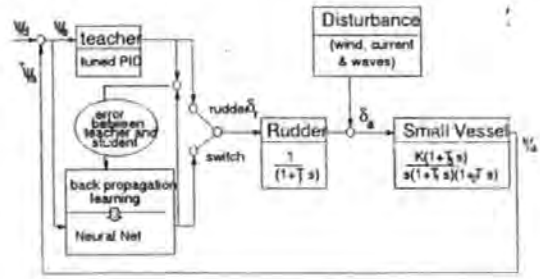
3. Learning of a Neural Net

3.1 General Facts

The two main tasks of a brain - learning and recall - are the most interesting for us. Learning itself is the process of the calibration of the synaptic efficiency, or in the words of artificial nets, the weights. Using this principle some models of neurons and their connections have been investigated, i.e. single layer nets, multi layer nets and self organising nets. The nets can be classified into three groups, depending upon the learning principle, e.g. *supervised learning* (as discussed in this paper), *learning with critic* and the last group *unsupervised learning (self organising nets)*. The latter is utilised to obtain relationships between the input and the output vector by the creation of an iterative process without a teacher (as in supervised learning) and also without evaluative values (as learning with critic). If we inspect supervised learning, we must consider, that the results of the student (our net) can be only as good as the training data of the teacher/ supervisor. For supervised learning we need a vector of input data and one vector of the desired outputs which is associated to the input vector. The reader can easily see, that one problem, besides the program for learning, is to have good sets of training data. We interpret a set of training data as a pair of input/ desired output vectors.

3.2 Back Propagation

Rumelhart's contributions to neural nets (1986) are fundamentals for further investigation. In this paper, the method of supervised learning will be discussed and how to use this in order to develop a learning controller for the steering of small craft.
One way to utilise the supervised learning is by using the *back propagation* algorithm.
The control model is displayed in figure 7.
The neural net used in this algorithm is a multi layer net and the transfer function is the *Sigmoid*. The back propagation rule needs the *error* between computed output by the net



ψ_a ψ_d heading	actual/ desired
$\delta_a\delta_r$ rudder	actual/ requested
ψ_e	error heading
T_1, T_2, T_3, T_r	time constants

figure 7

(straight forward or phase 1) and the desired output given by the teacher. To adjust the weights on the path between one neuron and the next neuron, the error is back propagated, starting with the output layer back to the first hidden layer (layer netxt to the input layer). This process is the second or the learning phase.

The steps - computing forward and error propagation backwards - is repeated with different pairs of training data until a maximum number of epochs is reached or the global error approaches an acceptable value less than i.e. $\varepsilon = 0.05$.

The interesting feature of back propagation is, that we do not need any knowledge about the process, this is what we will use the controller for, but we need a good teacher. However, this is on the other hand a disadvantage because our student does not have any self organising capabilities and so this can not be a better response than the teacher. The final algorithm (equations) is displayed below.

straight forward (phase 1)

$$x_i^l = Sigmoid\left(I_i^l\right) \quad (2)$$

$$I_i^l = \sum_j w_{i,j}^l x_j^{l-1} + \theta_i^l \quad (3)$$

$$\tilde{w}_{j,i}^l = w_{j,i}^l + \Delta w_{j,i}^l \quad (4)$$

back propagation (phase 2)

$$\Delta w_{j,i}^l = \eta \delta_j^l x_j^{l-1} \quad (5)$$

$$\delta_j^l = x_i^l\left(1 - x_i^l\right) \cdot \left(d_i - x_i^l\right) \quad \text{output layer (6)}$$

$$\delta_j^l = x_i^l\left(1 - x_i^l\right) \cdot \sum_k \delta_k^{l+1} w_{j,k}^{l+1} \quad \text{inner neur. (7)}$$

η .....learning coefficient
x^l_i.....output of neuron i in layer l
δ^l_i.....error of neuron i in layer l
I^l_i.....total input of neuron i in layer l
w^l_{ji} .....weight on path from neuron j in layer l-1 to neuron i in layer l
Δw^l_{ji}.....weight increment for w^l_{ji}
Sigmoid() transfer function

Note, the learning rules for the thresholds θ are the same as the rules for the weights. The threshold is a weight with the associated input of 1.0 .

4. Application of Neural Nets for Ship Steering

4.1 Introduction

Many collisions and groundings of marine vessels occur in the approaches to a port where the traffic density is intense. This suggests that there is a need for automatic guidance systems to deal with the problems of surface ships manoeuvring in confined waters, possibly under shore control as part of the port's Vessel Traffic Services.

Modern sea going vessels have a range of navigation aids including global positioning system (GPS) receivers, Doppler sonar, gyrocompass as well as hypobolic aids such as Loran C and Decca. Current trends include the use of standard interfaces to network communication systems using computer tools such as electronic charts to form integrated navigation systems. It is also possible to employ the navigational data to provide best estimates of state vectors (Kalman filter) and optimal guidance strategies. Such techniques require powerful computing facilities, particularly if the dynamic characteristics of the vessel are changing, as may be the case in a manoeuvring situation or changes in forward speed.

Chapter (4.3) of this paper investigates the possibility of training a Neural Network to behave in the same manner as an optimal ship guidance system, the objective being to provide a system that can adapt its parameters so that it provides optimal performance over a range of conditions, without incurring a large computational penalty.

A series of simulation studies have been undertaken to compare the performance of a trained neural network with that of the original optimal guidance system over a range of forward speeds. It is demonstrated that a single network has comparable performance to a set of optimal guidance control laws, each computed for different forward speeds.

Since the increase in the number of computers, more and more modern tech-

niques have been used such as neural computing (neural nets), fuzzy logic, etc.
Conventional ship autopilots are based on proportional, integral and derivative (PID) control algorithms and are used to control the ship's heading in an open seaway. These controllers are developed to work under specific conditions and so they are not working at their optimal point and need to be reset to take into account the vessel's handling characteristics and environmental conditions. It is not current practice to use an automatic system in the approaches to a port and in many cases control of the vessel is handed over to a pilot at this stage. However, it is in the pilotage phase of the voyage, where the traffic density is intense, that the risks of collision and grounding are highest. In addition, it has been highlighted /1/ that over 80 percent of all marine accidents are due to human error.
The main idea of a modern controller is to merge all the beneficial features of several controllers to create an intelligent controller, i.e. with a behaviour like a human helmsman.

4.2 An Artifical Neural Network Autopilot for Small Vessels

4.2.1 *Creation of the Training Data*

In the previous chapters, a teacher for the neural net was mentioned. The idea that this study is based on is, that one PID controller is tuned for one particular sea state and this *tuned PID controller* is used as one teacher for the neural net. If a training file, which contains input and output data of the PID controller, is created, the neural net will learn to respond like its teacher. But if the training data file consists of data pairs of more than one teacher, i. e. data of several tuned PID controllers in several sea states, the neural net will learn the behaviour of the tuned PID controllers at its optimal point or close to it. We know that the ship parameters such as weight, inertia, draught and speed, have key effects in the behaviour of the ship. So, if we want, we could tune PID controllers for more specific situations and create more relevant data.
The PID controllers, used as teachers, are tuned firstly for a very small heading error and not for a smooth rudder movement. Tests have proved that the neural net will work as a damper too.
The training file for the neural net contains PID information of different tuned controllers in the associated sea states from port and starboard directions and the current output of the controllers. Further difficulties can arise when the inputs of the net have very big differences in the values.
Suppose the heading error and the rate of change of the heading error are in the order of 10^{-1} and less than 10^{1} and the integral of the heading error is bigger than 10^{2}, emphasis will be placed on the input neuron for the integral and the small changes of the other two neurons are not taken into consideration. In this case, the net will only learn the rudder offset to remove the average disturbance without alternating.

4.2.2 *Training the Network*

The net consisting of 10 neurons in each of the 2 chosen hidden layers. The architecture

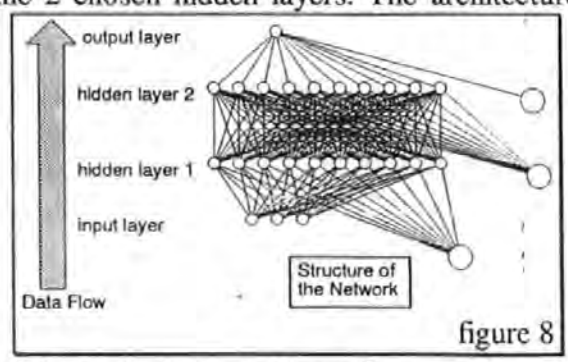


figure 8

of the net is shown in figure 8.
During the learning, the training sets are randomly selected until the given number, in this case 60,000 is reached. It is possible to formulate the stop condition in association with the actual error between computed output of the net and the desired output given by the teacher.

4.2.3 Results

In the following graphs you can see the learning process and the comparison of a trained Neural Network to a PID controller.

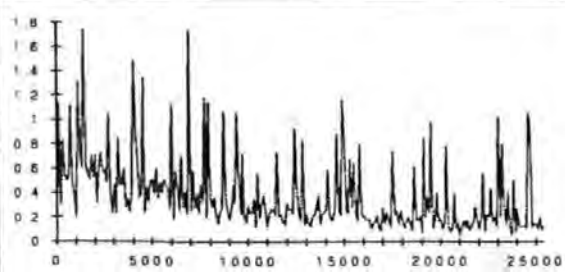


figure 9: learning

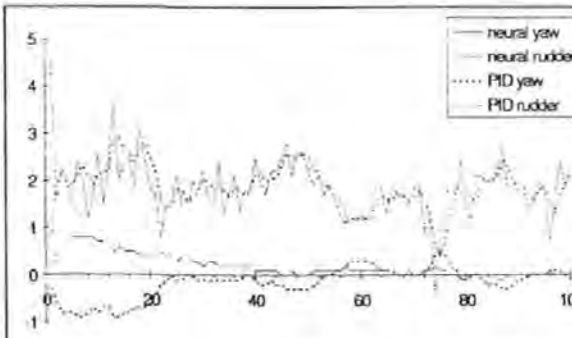


figure 10: comparison of Neural Net to PID in sea state 3

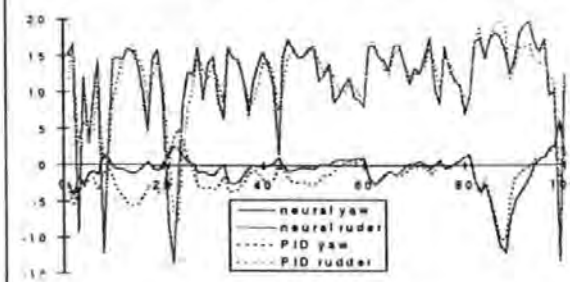


figure 11: comparison of Neural Net to PID in sea state 5

Sea State	3	4	5
RMS Yaw Error Neural Net	0.103	0.197	1.863
RMS Yaw Error PID	0.194	0.579	2.387

table 1: results

As you can see, the Neural Net is able to react in the manner of its teacher(s). It is possible to train a Neural Net with the data of more than one teacher and the network will pick up the behaviour of all the supervisors.

4.3 Optimal Ship Guidance using a Neural Network Approach

4.3.1 Background

This suggests that there is a need for automatic guidance systems for marine vehicles in confined waterways, such as many of the world's major ports, even to the extent of allowing transfer of control from ship to shore using the port's Vessel Traffic Services (VTS). As electronic navigation aids become more sophisticated and the use of satellite Global Positioning System (GPS), particularly used in differential mode, becomes more widespread, the concept of fully automatic pilotage in port approaches becomes a tangible reality.

It has been demonstrated by Burns **/4/** that it is possible to design an optimal multivariable ship guidance system that controls position, heading and speed simultaneously, and that such a system can work within the constraints required in port approaches.

By the use of multivariable system theory, it is possible to construct a mathematical model of a surface ship that can respond to control inputs (rudder and main engines) and also disturbance inputs (wind, waves and current). Such a mathematical description normally requires a set of non-linear differential equations.

Based on a multivariable model with a control vector **u**, a disturbance vector **w** and a state vector **x** an optimal control policy may be formulated that minimises a performance index, or cost function. A problem with this approach is that if the dynamic characteristics of the vessel change (due to variations in forward speed for example) then the guidance system is sub-optimal, and its parameters need to be re-computed. This places a large computational burden on the ships navigational computer, which must perform its calculations during the sample period.

4.3.2 Ship Mathematical Model

Ship motions in surge, sway, heave, roll, pitch and yaw can be described by a Eulerian set of non-linear differential equations of the form:

$$\begin{aligned} m(\dot{u} + qw - rv) &= X \\ m(\dot{v} + ru - pw) &= Y \\ m(\dot{w} + pv - qu) &= Z \\ I_x \dot{p}(I_z - I_y) &= L \\ I_y \dot{p}(I_x - I_z) &= M \\ I_z \dot{p}(I_y - I_x) &= N \end{aligned} \quad (8)$$

The terms X, Y, Z, L, M and N represent all the external forces and moments acting on the hull and include both linear and non-linear components. These equations can be arranged as a set of state equations in terms of the state vector **x**, control vector **u** and disturbance vector **w**, where:

$$x^T = (\delta_A \;\; n_A \;\; x \;\; u \;\; y \;\; v \;\; z \;\; w \;\; \phi \;\; p \;\; \theta \;\; q \;\; \psi \;\; r) \quad (9)$$

$$u^T = (\delta_D \;\; n_D) \quad (10)$$

$$w^T = (u_c \;\; v_c \;\; u_a \;\; v_a \;\; \zeta_x \;\; \zeta_y) \quad (11)$$

The vessel used in the simulation had the following parameters:

Length	= 161 m
Draught	= 9 m
Beam	= 23 m
Displacement	= 17000 tonnes
Number of propellers	= 1
Number of rudders	= 1
Maximum rudder angle	= ±35 degrees

The dynamic characteristics of the vessel may be described in terms of its open-loop eigenvalues. When u = 7.717 m/s (15 knots), these are:

s = -0.5, -0.039, -0.0755,
0, 0, -0.5, 0 -0.00913 (12)

When the vessel is travelling at 2.572 m/s (5 knots), they become:

s = -0.5, -0.013, -0.0252,
0, 0, -0.5,0 -0.00265 (13)

These results are shown in Figure 12 and demonstrates that the vessel becomes less manoeuvrable at low speeds, thus requiring a control policy that takes this into account.

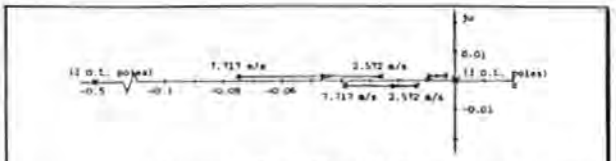


figure 12: Effect of Forward Speed on Open-Loop Eigenvalues

4.3.3 Optimal Guidance Policy

Given the state equations:

$$\dot{x}(t) = F(t)x(t) + G_c(t)u + G_D(t)w(t) \quad (14)$$

and the quadratic criterion to be minimised:

$$J = \int_{t_0}^{t_1} \left\{ (x - r)^T Q(x - r) + u^T R u \right\} dt \quad (15)$$

where **r** is the desired state vector. It can be shown Burns /5/ that the optimal control is:

$$u_{opt} = -\left(Sx + R^{-1} G^T m\right) \quad (16)$$

where **S** is the optimal feedback gain matrix calculated from solution of the Riccati equations and **m** is the command vector, computed from a knowledge of the system model and the desired state vector.

Figure 13 illustrates the departure of the eigenvalues from their assigned positions as the forward speed is reduced from 7.717 m/s to 2.572 m/s. To maintain a fixed closed-loop eigenvalue array, it is necessary to re-compute the feedback matrix **S** at each forward speed.

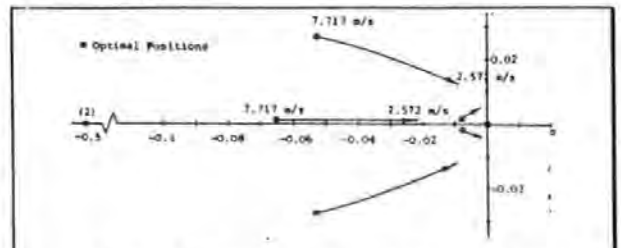


figure 13: Eigenvalue Departure During a Speed Reduction.

Figure 14 shows a simulated approach into the Port of Plymouth using the assigned closed-loop eigenvalues. The problem of way-point overshoot is overcome by using way-point advance and dual-mode control as shown in Figure 15. Under a dual-mode policy, when the advanced way-point is reached, the controller switches from track to course weighting, thus suppressing ψ_t and emphasising ψ_c.

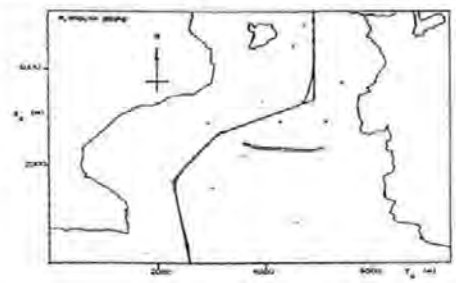


Figure 14: Simulated Approach into the Port of Plymouth.

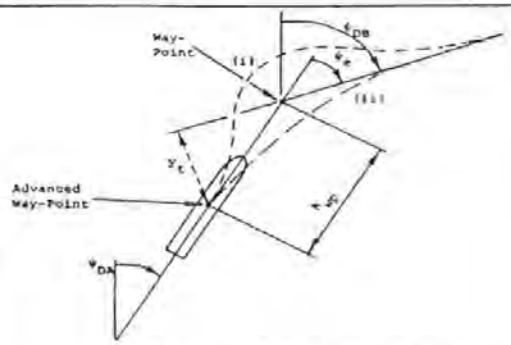


Figure 15: Way-Point Advance.

4.3.4 *Neural Network Guidance System*

4.3.4.1 Supervised Learning using Back Propagation

The output of each artificial neuron is calculated by multiplying each incoming signal by an associated weight, and adding together all the weighted inputs to form the total input and uses a Sigmoid function to create the output.

A neural network consists of at least an input layer, and output layer and a single intermediate, or hidden layer. All neurons are interconnected. A single hidden layer can be used for simple applications, but for more complex situations, multi-layer networks are employed.

The problem is to find the optimum size that gives the best balance between accuracy and speed (Richter and Burns /6/).

In this application the network is trained using supervised learning. A subset of the state vector (only those terms that affect the demanded rudder) are input to both the optimal guidance system and the neural net, which both compute the demanded rudder δ_D Differences between the two values are used to train the network using back propagation learning as shown in Figure 16.

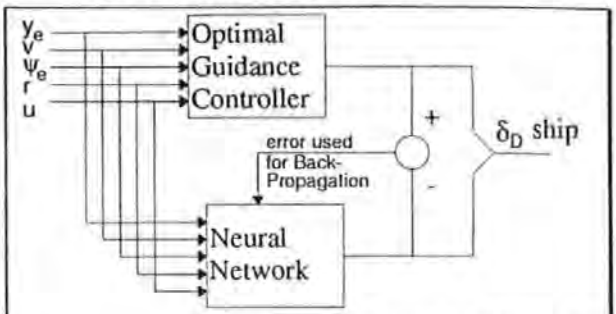


figure 16: Supervised Learning using Back Propagation

4.3.4.2 Training the Network

Training data from the optimal guidance system over a range of speeds from 2.572 to 7.717 m/s was collected in the simulated approach to the Port of Plymouth. During learning, the training sets are randomly selected and run over a given interval, usually until 100,000 - 200,000 samples have been taken. Parameters such as momentum, learning coefficient and number of neurons in hidden layer are then varied until a global minimum error is achieved. For this application, this occurred for the following values:

input neurons	= 5
output neurons	= 1
number of hidden layers	= 2
neurons in hidden layer	= 10
learning coefficient	= 0.6
momentum	= 0.4
number of samples	= 200,000
learning time	= 2230 seconds

4.3.5 *Results*

The weight coefficient matrices were used in an neuro-optimal hybrid controller to test the system, the neural network controlling the rudder, the optimal guidance system controlling the main engines.

4.3.5.1 Low Speed Approach

In the low speed approach, the vessel commenced its run at 2.572 m/s. Figure 17

shows the cross-track error ψ_e for both the optimal and neural controllers. Figures 18 and 19 give the corresponding results for the heading error ψ_e and rudder δ_D. It can be seen that there is good correlation for heading and rudder, but there is an offset with the cross-track error. This is possibly due to the way in which the training data was scaled.

The dual-mode operation can clearly be seen. At t = 580 seconds the first way-point is approached and the controller drives the rudder hard-over as it enters course-changing mode. When the heading error is 20 degrees at t = 700 seconds, track-keeping mode is resumed.

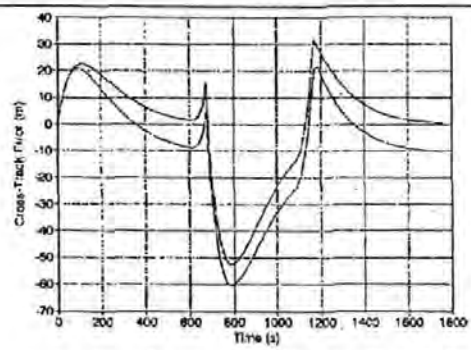


figure 17: Cross-Track Error - Low Speed Approach

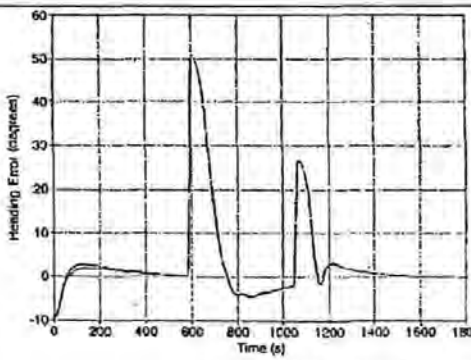


figure 18: Heading Error - Low Speed Approach

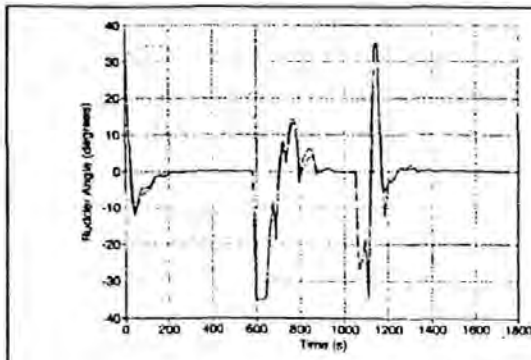


figure 19: Rudder Angle - Low Speed Approach

4.3.5.2 High Speed Approach

Here the speed of approach was 7.717 m/s. Figures, and show the cross track error, heading error and demanded rudder respectively. As with the low speed approach, there is good correlation between the two controllers for rudder and heading, with an offset on the cross-track data. At high speeds, it can be seen that the rudder excursions are far smaller, indicating that both controllers have adapted to the change in vessel dynamics.

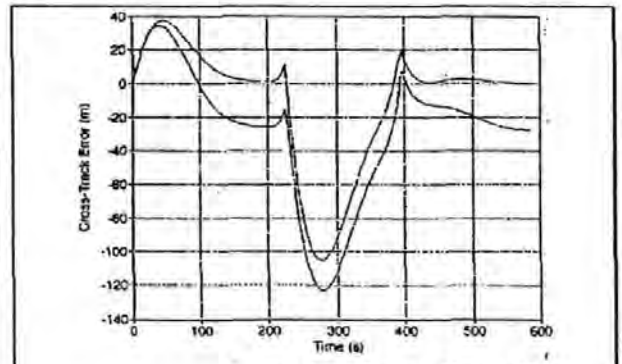


figure 20: Cross-Track Error - High Speed Approach

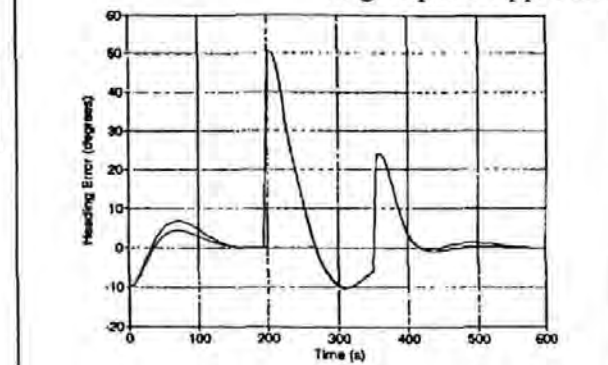


figure 21: Heading Error - High Speed Approach

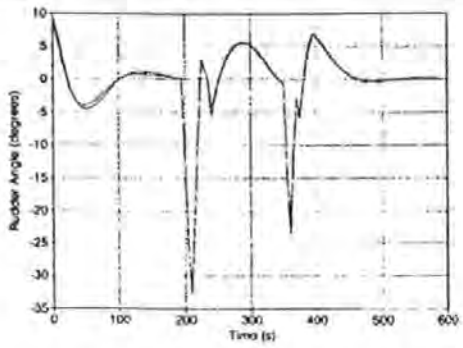


figure 22: Rudder Angle - High Speed Approach

4.3.5.3 Controller Performance

In order to compare the performance of the two controllers, a set of generalised performance indices may be defined:

$$J_y = \int_{t_0}^{t_1} (y_D - y_A)^2 dt$$

$$J_\psi = \int_{t_0}^{t_1} (\psi_D - \psi_A)^2 dt \qquad (17)$$

$$J_\delta = \int_{t_0}^{t_1} \delta_A{}^2 dt$$

Table 2 gives the comparative results.

speed	Optimal	Controller		Neural	Controller	
(m/s)	J_y	J_ψ	J_δ	J_y	J_ψ	J_δ
2.572	0.764E6	111.4	67.2	0.968E6	111.7	63.6
7.717	0.852E6	43.2	61.1	0.138E7	43.9	63.4

table 2: Comparative Results

From Table 2 it can be seen that in terms of heading and rudder, the two controllers perform in a similar manner, but the optimal controller provides better track-keeping performance.

4.3.6 Conclusions

The results of this initial study demonstrate that a neural network may be trained from data provided by an optimal guidance system. The trained network performs in a slightly sub-optimal manner - but has the advantage that it does not have to re-compute controller parameters for different forward speeds. At this stage it is not known how the network would cope with another way-point configuration.

The properties of multi-layer neural networks are not yet fully understood. It would appear however, that a ship guidance system is a potential application of the technique. There is extensive scope for further research in this field, particularly in the design of un-supervised learning networks that adapt in an on-line manner.

4.4 References

/1/ Panel on Human Error in Merchant Marine Safety.: "Human Error in Merchant Marine Safety". National Academy of Science, Washington DC 1976.

/2/ M. Endo, J. van Amerongen, A. W. P. Bakkers: "Application of Neural Networks to Ship Steering"

/3/ Yoh-Han Pao: Case Western Reserve University "Adaptive Pattern Recognition and Neural Networks" Addison- Wesley Publishing Company, Inc.

/4/ R S Burns: "The Design, Development and Implemaentation of an Optimal Guidance System for Ships in Confident Waters". Proceedings of the 9th Ship Control Systems Symposium, Vol 3, Bethesda 1990, USA.

/5/ R S Burns: "Application of the Riccati Equation in Control and Guidance of Marine Vehicles", Proceedings of Workshop on the Riccati Equation in Control, Systems and Signals. International Federation of Automatic Control, Como, Italy 26-28th June 1989.

/6/ R Richter, R S Burns: "An Artificial Neural Network Autopilot for Small Vessels", UKSS 93, 13-15th September 1993, Keswick, UK, The Society for Computer Simulation, 1993.

5. Appendix A

The example to explain the mathematics for the back propagation algorithm is the well known XOR function. For that reason we design a net with 2 input and 1 output neuron. In the only hidden layer we place two neurons. The structure of the net is displayed in figure 23.

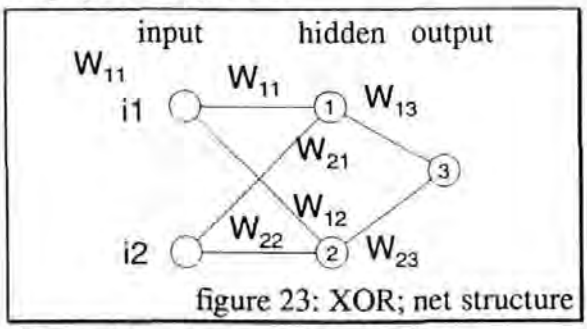


figure 23: XOR; net structure

Our training data are shown below.

x_1	x_2	$x_1 \otimes x_2$
0	0	0.1
0	1	0.9
1	0	0.9
1	1	0.1

(table 3)

Remember the transfer function (sigmoid), we have to scale our desired outputs between 0.1 and 0.9. To rescale the output of the net, we can use the following equation:

$new = \frac{output - 0.1}{0.8}$ in general

$new = \frac{output - 0.1}{0.8} \cdot |\text{maximum of the teacher}|$, if values between $\pm\alpha$ are requested $new = \frac{output - 0.5}{0.4} \cdot |\alpha|$. Now, we have a net and training data, only the learning algorithm is missing. With the known knowledge, it is possible to calculate the output of the net. The weights, we have to use, are randomly initialised. The error between the computed output and the desired output given by the teacher gives us the sign and the speed we have to change the weights to reduce this error.

5.1.1 Mathematical Background of Back Propagation

The equations for phase 1 (neuron 1) are:

$$I_1 = i_1 \cdot w_{1,1} + i_2 \cdot w_{2,1} + 1 \cdot \theta_1$$

(total input) (18)

$$x_1 = Sigmoid(I_1) = \frac{1}{1 + e^{-I_1}}$$

(output) (19)

similarly for neuron 2 and for neuron 3 see below,

$$I_3 = x_1 \cdot w_{1,3} + x_2 \cdot w_{2,3} + 1 \cdot \theta_3$$

(total input) (20)

$$x_3 = Sigmoid(I_3) = \frac{1}{1 + e^{-I_3}}$$

(output) (21)

x_3 is the final output of the net and we have the error

$$E_3 = \frac{1}{2}(d - x_3)^2. \quad (22)$$

The value ½ is inserted with a mathematical sense. The reason is explained below. The main aim of learning is to reduce the error as fast as possible. Using the gradient of the error for one special weight, we have the method to minimise the difference between calculated output of one neuron and the desired output by modifying this weight. We define an increment $\Delta w_{1,3}$ proportional to $-\partial E / \partial w_{1,3}$.

$$\Delta w_{1,3} = -\eta \frac{\partial E_3}{\partial w_{1,3}} \quad \eta \text{ .. learning rate} \quad (23)$$

The error E_3 (equation 22) is formed in terms of the output x_3 and this is a function of I_3 (total input equation 20). Using the partial derivative $-\partial E / \partial w_{1,3}$, we can say

$$\frac{\partial E_3}{\partial w_{1,3}} = \frac{\partial E_3}{\partial I_3} \cdot \frac{\partial I_3}{\partial w_{1,3}}. \quad (24)$$

Using the equation 20 to get I_3 we can write

$$\frac{\partial I_3}{\partial w_{1,3}} = \frac{\partial}{\partial w_{1,3}}\left(\sum_{i=1}^{2} x_i \cdot w_{i,3} + \theta_3\right)$$

$$= \frac{\partial}{\partial w_{1,3}}\left(x_1 \cdot w_{1,3} + x_2 \cdot w_{2,3} + \theta_3\right) \qquad (25)$$

$$\frac{\partial I_3}{\partial w_{1,3}} = x_1$$

If we now define

$$\delta_3 = -\frac{\partial E_3}{\partial I_3} \qquad (26)$$

so can we formulate $\Delta w_{1,3}$ in the following form

$$\Delta w_{1,3} = \eta \delta_3 x_1 \ . \qquad (27)$$

To get $\delta_3 = -\partial E_3 / \partial I_3$ we use the chain rule to declare the partial derivative in one term (rate of change of error) and a second term (rate of change of the output x_3) where we consider the inputs to the same neuron. That is written

$$\delta_3 = -\frac{\partial E_3}{\partial I_3} = -\frac{\partial E_3}{\partial x_3}\frac{\partial x_3}{\partial I_3} \ . \qquad (28)$$

The separated factors can be calculated as follows:

$$\frac{\partial E_3}{\partial x_3} = -(d - x_3) \qquad (29)$$

and

$$\frac{\partial x_3}{\partial I_3} = f\,(I_3) \ . \qquad (30)$$

From which we simplify

$$\delta_3 = (d - x_3) f\,(I_3) \qquad (31)$$

$$\Delta w_{1,3} = \eta (d - x_3) f\,(I_3) x_1 = \eta d_3 x_1 . \qquad (32)$$

By using Sigmoid as the transfer function, we remember equation,

$$x_3 = Sigmoid(I_3) = \frac{1}{1 + e^{-I_3}}$$

$$\frac{\partial x_3}{\partial I_3} = \frac{\partial}{\partial I_3}\left(\frac{1}{1 + e^{-I_3}}\right)$$

$$\frac{\partial x_3}{\partial I_3} = +\frac{1}{\left(1 + e^{-I_3}\right)^2} e^{-I_3}$$

that we can transform into

$$\frac{\partial x_3}{\partial I_3} = x_3 \cdot (1.0 - x_3) \qquad (33)$$

and so finally

$$\Delta w_{1,3} = \eta \cdot (d - x_3) \cdot x_3 (1.0 - x_3) \cdot x_1 \ . \qquad (34)$$

Analogue to the output neuron 3, we can use the general equations to change the weights in the other layers. But there are some different circumstances if the considered neuron is nót in the output layer. We can still write the general equations

$$\Delta w^l_{j,i} = -\eta \frac{\partial E}{\partial w^l_{j,i}}$$

$$= -\eta \frac{\partial E}{\partial I^l_i}\frac{\partial I^l_i}{\partial w^l_{j,i}}$$

$$= -\eta \frac{\partial E}{\partial I^l_i} x^{l-1}_j$$

$$= -\eta \left(\frac{\partial E}{\partial x^l_i}\frac{\partial x^l_i}{\partial I^l_i}\right) x^{l-1}_j$$

$$= -\eta \left(\frac{\partial E}{\partial x^l_i}\right) f\,(I_i) x^{l-1}_j$$

$$\Delta w^l_{j,i} = \eta d^l_i x^{l-1}_j \ . \qquad (35)$$

But we can not evaluate the factor $-\partial E / \partial x^l_i$ directly. Suppose, the present layer is the layer before the output layer. We know the error of the neurons output in the "next" (in this case: output ~) layer and so we can formulate

$$-\frac{\partial E}{\partial x^l_j} = -\sum_j \frac{\partial E}{\partial I^{l+1}_j}\frac{\partial I^{l+1}_j}{\partial x^l_i}$$

$$= \sum_j \left(\frac{\partial E}{\partial I^{l+1}_j}\right)\frac{\partial}{\partial x^l_i}\sum_k x^l_k w^{l+1}_{i,k}$$

$$= \sum_j \left(\frac{\partial E}{\partial I^{l+1}_j}\right) w^{l+1}_{i,k}$$

$$-\frac{\partial E}{\partial x^l_j} = -\sum_j \delta^{l+1}_j w^{l+1}_{i,j} \ , \qquad (36)$$

which means, that δ for an internal node can be evaluated in terms of δ in the next layer. Finally we can say

$$\Delta w^l_{j,i} = \eta \delta^l_i x^{l-1}_j \quad \text{where}$$

$$\delta^l_j = x^l_i \left(1 - x^l_i\right) \cdot \left(d_i - x^l_i\right)$$

for l = output layer (37)

$$\delta_j^l = x_i^l\left(1-x_i^l\right)\cdot\sum_k \delta_k^{l+1} w_{j,k}^{l+1}$$

for l = internal layer (38)

Note, the learning rules for the thresholds θ are the same as the rules for the weights. The threshold is a weight with the associated input 1.0. The conclusion is displayed below.

straight forward (phase 1)

$$x_i^l = Sigmoid\left(I_i^l\right) \quad (39)$$

$$I_i^l = \sum_j w_{i,j}^l x_j^{l-1} + \theta_i^l \quad (40)$$

$$\widetilde{w}_{j,i}^l = w_{j,i}^l + \Delta w_{j,i}^l \quad (41)$$

back propagation (phase 2)

$$\Delta w_{j,i}^l = \eta \delta_j^l x_j^{l-1} \quad (42)$$

$$\delta_j^l = x_i^l\left(1-x_i^l\right)\cdot\left(d_i - x_i^l\right) \text{ output layer} \quad (43)$$

$$\delta_j^l = x_i^l\left(1-x_i^l\right)\cdot\sum_k \delta_k^{l+1} w_{j,k}^{l+1} \text{ inner neur.} \quad (44)$$

η ..... learning coefficient

x^l_i ..... output of neuron i in layer l

δ^l_i ..... error of neuron i in layer l

I^l_i ..... total input of neuron i in layer l

w^l_{ji} ..... weight on path from neuron j in layer l-1 to neuron i in layer l

Δw^l_{ji} ..... weight increment for w^l_{ji}

Sigmoid() .. transfer function

Using the standard algorithm, explained in this chapter, the learning may be very frequent and/ or the approach to zero is not rapid enough. To reduce this effect, we insert a momentum, often described as inertia. The momentum is an additive term to Δw^l_{ji} which considers the last change of the weight. The term is working like a damper or low pass filter. The new equation to calculate the change of the weight Δw^l_{ji} is

$$\Delta_p w_{j,i}^l = \eta \delta_j^l x_j^{l-1} + \alpha \Delta_{p-1} w_{j,i}^l \quad . \quad (45)$$

p means the present training set, and with $\Delta_{p-1} w_{j,i}^l$, we consider the direction and rate of the last modification of the weight made for the training set before. With this change we get a smoother and more rapid learning. With this term we can chose a bigger learning rate which means faster learning.

R. S. Burns and R. Richter.

The application of neural networks for the control of small and large vessels.

In *Workshop on Control Application in Marine Systems (CAMS '95)*, pages 393–399, Trondheim - Norway, 10–12 May 1995.

The Application of Neural Networks for the Control of Small and Large Vessels

Dr Roland BURNS
and
Mr Ralph RICHTER

School of Manufacturing, Materials and Mechanical Engineering

University of Plymouth

United Kingdom

1. Introduction

Many collisions and groundings of marine vessels occur in the approaches to a port where the traffic density is intense. This suggests that there is a need for automatic guidance systems to deal with the problems of surface ships manoeuvring in confined waters, possibly under shore control as part of the port's Vessel Traffic Services.

Modern sea going vessels have a range of navigation aids including global positioning system (GPS) receivers, Doppler sonar, gyrocompass as well as hypobolic aids such as Loran C and Decca. Current trends include the use of standard interfaces to network communication systems using computer tools such as electronic charts to form integrated navigation systems. It is also possible to employ the navigational data to provide best estimates of state vectors (Kalman filter) and optimal guidance strategies. Such techniques require powerful computing facilities, particularly if the dynamic characteristics of the vessel are changing, as may be the case in a manoeuvring situation or changes in forward speed.

Chapter (1.2) of this paper investigates the possibility of training a Neural Network to behave in the same manner as an optimal ship guidance system, the objective being to provide a system that can adapt its parameters so that it provides optimal performance over a range of conditions, without incurring a large computational penalty.

A series of simulation studies have been undertaken to compare the performance of a trained neural network with that of the original optimal guidance system over a range of forward speeds. It is demonstrated that a single network has comparable performance to a set of optimal guidance control laws, each computed for different forward speeds.

Since the increase in the number of computers, more and more modern techniques have been used such as neural computing (neural nets), fuzzy logic, etc.

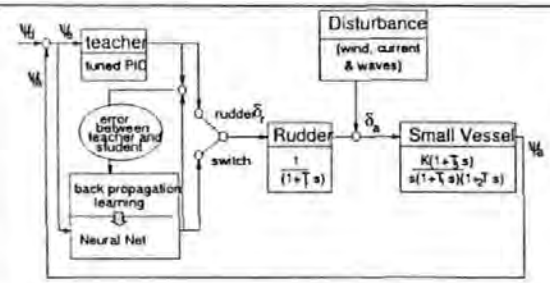
Conventional ship autopilots are based on proportional, integral and derivative (PID) control algorithms and are used to control the ship's heading in an open seaway. These controllers are developed to work under specific conditions and so they are not working at their optimal point and need to be reset to take into account the vessel's handling characteristics and environmental conditions. It is not current practice to use an automatic system in the approaches to a port and in many cases control of the vessel is handed over to a pilot at this stage. However, it is in the pilotage phase of the voyage, where the traffic density is intense, that the risks of collision and grounding are highest. In addition, it has been highlighted /1/ that over 80 percent of all marine accidents are due to human error.

The main idea of a modern controller is to merge all the beneficial features of several controllers to create an intelligent controller, i.e. with a behaviour like a human helmsman.

1.1. An Artifical Neural Network Autopilot for Small Vessels

1.1.1. Creation of the Training Data

The idea that this study is based on is, that one PID controller is tuned for one particular sea state and this *tuned PID controller* is used as one teacher for the neural net (see fig 1).



ψ_a ψ_d .....actual/ desired heading
$\delta_a \delta_r$.....actual/ requested rudder
ψ_e .....error heading
T_1, T_2, T_3, T_r ....time constants

figure 1

If a training file, which contains input and output data of the PID controller, is created, the neural net will learn to respond like its teacher. But if the training data file consists of data pairs of more than one teacher, i. e. data of several tuned PID controllers in several sea states, the neural net will learn the behaviour of the tuned PID controllers at its optimal point or close to it. We know that the ship parameters such as weight, inertia, draught and speed, have key effects in the behaviour of the ship. So, if we want, we could tune PID controllers for more specific situations and create more relevant data.

The PID controllers, used as teachers, are tuned firstly for a very small heading error and not for a smooth rudder movement. Tests have proved that the neural net will work as a damper too.

The training file for the neural net contains PID information of different tuned controllers in the associated sea states from port and starboard directions and the current output of the controllers. Further difficulties can arise when the inputs of the net have very big differences in the values.

Suppose the heading error and the rate of change of the heading error are in the order of 10^{-1} and less than 10^{1} and the integral of the heading error is bigger than 10^{2}, emphasis will be placed on the input neuron for the integral and the small changes of the other two neurons are not taken into consideration. In this case, the net will only learn the rudder offset to remove the average disturbance without alternating.

1.1.2. *Training the Network*

The net consisting of 10 neurons in each of the 2 chosen hidden layers. The architecture of the net is shown in figure 2.

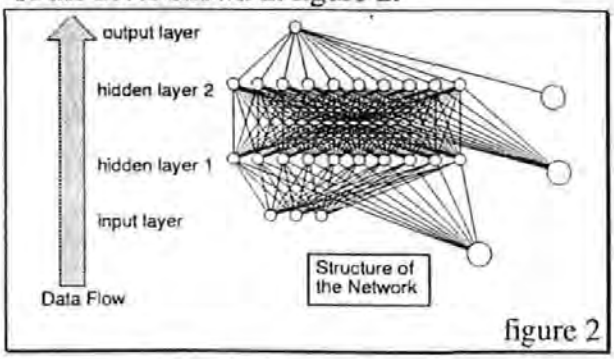


figure 2

During the learning, the training sets are randomly selected until the given number, in this case 60,000 is reached. It is possible to formulate the stop condition in association with the actual error between computed output of the net and the desired output given by the teacher.

1.1.3. *Results*

In the following graphs you can see the learning process and the comparison of a trained Neural Network to a PID controller.

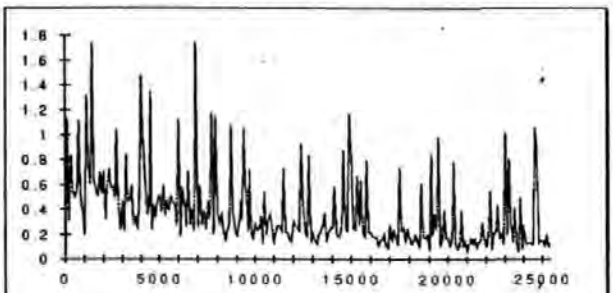


figure 3: learning

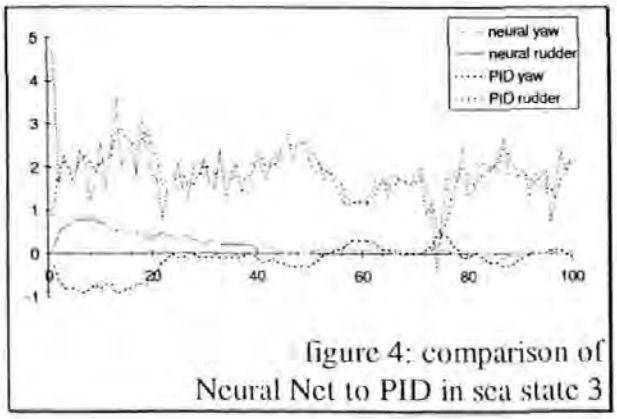


figure 4: comparison of Neural Net to PID in sea state 3

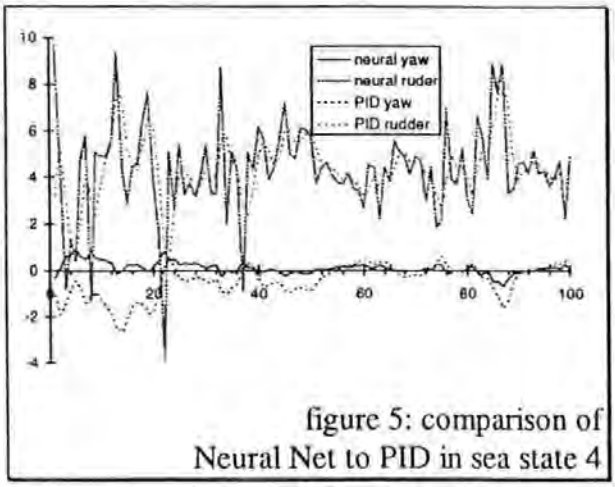


figure 5: comparison of Neural Net to PID in sea state 4

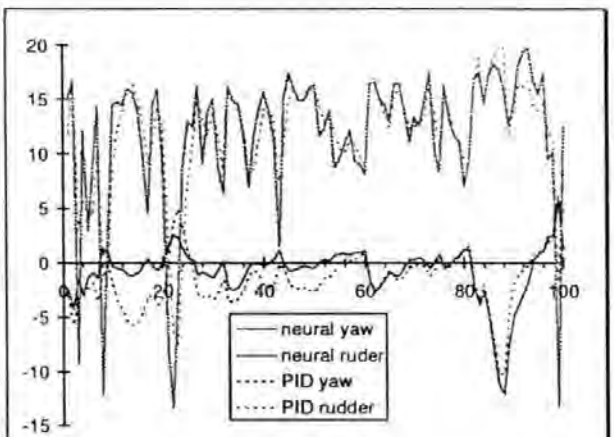


figure 6: comparison of Neural Net to PID in sea state 5

Sea State	3	4	5
RMS Yaw Error Neural Net	0.103	0.197	1.863
RMS Yaw Error PID	0.194	0.579	2.387

table 1: results

As you can see, the Neural Net is able to react in the manner of its teacher(s). It is possible to train a Neural Net with the data of more than one teacher and the network will pick up the behaviour of all the supervisors.

1.2. Optimal Ship Guidance using a Neural Network Approach

1.2.1. Background

This suggests that there is a need for automatic guidance systems for marine vehicles in confined waterways, such as many of the world's major ports, even to the extent of allowing transfer of control from ship to shore using the port's Vessel Traffic Services (VTS). As electronic navigation aids become more sophisticated and the use of satellite Global Positioning System (GPS), particularly used in differential mode, becomes more widespread, the concept of fully automatic pilotage in port approaches becomes a tangible reality.

It has been demonstrated by Burns /4/ that it is possible to design an optimal multivariable ship guidance system that controls position, heading and speed simultaneously, and that such a system can work within the constraints required in port approaches.

By the use of multivariable system theory, it is possible to construct a mathematical model of a surface ship that can respond to control inputs (rudder and main engines) and also disturbance inputs (wind, waves and current). Such a mathematical description normally requires a set of non-linear differential equations.

Based on a multivariable model with a control vector **u**, a disturbance vector **w** and a state vector **x** an optimal control policy may be formulated that minimises a performance index, or cost function. A problem with this approach is that if the dynamic characteristics of the vessel change (due to variations in forward speed for example) then the guidance system is sub-optimal, and its parameters need to be re-computed. This places a large computational burden on the ships navigational computer, which must perform its calculations during the sample period.

1.2.2. Ship Mathematical Model

Ship motions in surge, sway, heave, roll, pitch and yaw can be described by a Eulerian set of non-linear differential equations of the form:

$$\begin{aligned} m(\dot{u} + qw - rv) &= X \\ m(\dot{v} + ru - pw) &= Y \\ m(\dot{w} + pv - qu) &= Z \\ I_x \dot{p}(I_z - I_y) &= L \\ I_y \dot{p}(I_x - I_z) &= M \\ I_z \dot{p}(I_y - I_x) &= N \end{aligned} \qquad (1)$$

The terms X, Y, Z, L, M and N represent all the external forces and moments acting on the hull and include both linear and non-linear components. These equations can be arranged as a set of state equations in terms of the state vector **x**, control vector **u** and disturbance vector **w**, where:

$$x^T = \{\delta_A \;\; n_A \;\; x \;\; u \;\; y \;\; v \;\; z \;\; w \;\; \phi \;\; p \;\; \theta \;\; q \;\; \psi \;\; r\} \quad (2)$$

$$u^T = (\delta_D \;\; n_D) \quad (3)$$

$$w^T = (u_c \;\; v_c \;\; u_a \;\; v_a \;\; \zeta_x \;\; \zeta_y) \quad (4)$$

The vessel used in the simulation had the following parameters:

Length	= 161 m
Draught	= 9 m
Beam	= 23 m
Displacement	= 17000 tonnes
Number of propellers	= 1
Number of rudders	= 1
Maximum rudder angle	= ±35 degrees

The dynamic characteristics of the vessel may be described in terms of its open-loop eigenvalues. When u = 7.717 m/s (15 knots), these are:

s = -0.5, -0.039, -0.0755,
0, 0, -0.5, 0 -0.00913 (5)

When the vessel is travelling at 2.572 m/s (5 knots), they become:

s = -0.5, -0.013, -0.0252,
0, 0, -0.5,0 -0.00265 (6)

These results are shown in Figure 7 and demonstrates that the vessel becomes less manoeuvrable at low speeds, thus requiring a control policy that takes this into account.

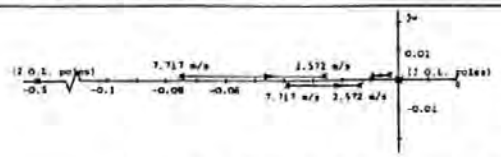


figure 7: Effect of Forward Speed on Open-Loop Eigenvalues

1.2.3. *Optimal Guidance Policy*

Given the state equations:

$$\dot{x}(t) = F(t)x(t) + G_c(t)u + G_D(t)w(t) \quad (7)$$

and the quadratic criterion to be minimised:

$$J = \int_{t_0}^{t_1} \left\{ (x - r)^T Q(x - r) + u^T Ru \right\} dt \quad (8)$$

where **r** is the desired state vector. It can be shown Burns /5/ that the optimal control is:

$$u_{opt} = -\left(Sx + R^{-1}G^T m\right) \quad (9)$$

where **S** is the optimal feedback gain matrix calculated from solution of the Riccati equations and **m** is the command vector, computed from a knowledge of the system model and the desired state vector.

Figure 8 illustrates the departure of the eigenvalues from their assigned positions as the forward speed is reduced from 7.717 m/s to 2.572 m/s. To maintain a fixed closed-loop eigenvalue array, it is necessary to re-compute the feedback matrix **S** at each forward speed.

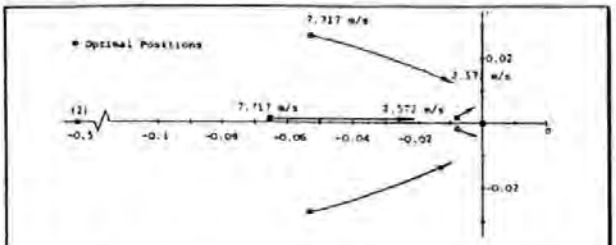


figure 8: Eigenvalue Departure During a Speed Reduction.

Figure 9 shows a simulated approach into the Port of Plymouth using the assigned closed-loop eigenvalues. The problem of way-point overshoot is overcome by using way-point advance and dual-mode control. Under a dual-mode policy, when the advanced way-point is reached, the controller switches from track to course weighting, thus suppressing y_t and emphasising ψ_c.

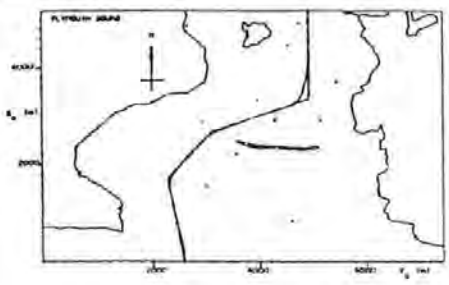


Figure 9: Simulated Approach into the Port of Plymouth.

1.2.4. Neural Network Guidance System

1.2.4.1. Supervised Learning using Back Propagation

The output of each artificial neuron is calculated by multiplying each incoming signal by an associated weight, and adding together all the weighted inputs to form the total input and uses a Sigmoid function to create the output.

A neural network consists of at least an input layer, and output layer and a single intermediate, or hidden layer. All neurons are interconnected. A single hidden layer can be used for simple applications, but for more complex situations, multi-layer networks are employed.

The problem is to find the optimum size that gives the best balance between accuracy and speed (Richter and Burns /6/).

In this application the network is trained using supervised learning. A subset of the state vector (only those terms that affect the demanded rudder) are input to both the optimal guidance system and the neural net, which both compute the demanded rudder δ_D Differences between the two values are used to train the network using back propagation learning as shown in Figure 1.

1.2.4.2. Training the Network

Training data from the optimal guidance system over a range of speeds from 2.572 to 7.717 m/s was collected in the simulated approach to the Port of Plymouth. During learning, the training sets are randomly selected and run over a given interval, usually until 100,000 - 200,000 samples have been taken. Parameters such as momentum, learning coefficient and number of neurons in hidden layer are then varied until a global minimum error is achieved. For this application, this occurred for the following values:

input neurons	= 5
output neurons	= 1
number of hidden layers	= 2
neurons in hidden layer	= 10
learning coefficient	= 0.6
momentum	= 0.4
number of samples	= 200,000
learning time	= 2230 seconds

1.2.5. Results

The weight coefficient matrices were used in an neuro-optimal hybrid controller to test the system, the neural network controlling the rudder, the optimal guidance system controlling the main engines.

1.2.5.1. Low Speed Approach

In the low speed approach, the vessel commenced its run at 2.572m/s. Figure 10 shows the cross-track error y_e for both the optimal and neural controllers. Figures 11 and 12 give the corresponding results for the heading error ψ_e and rudder δ_D. It can be seen that there is good correlation for heading and rudder, but there is an offset with the cross-track error. This is possibly due to the way in which the training data was scaled.

The dual-mode operation can clearly be seen. At t = 580 seconds the first way-point is approached and the controller drives the rudder hard-over as it enters course-changing mode. When the heading error is 20 degrees at t = 700 seconds, track-keeping mode is resumed.

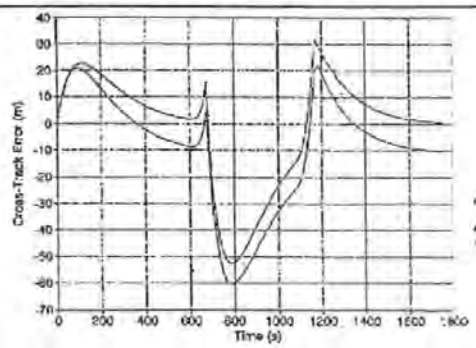


figure 10: Cross-Track Error - Low Speed Approach

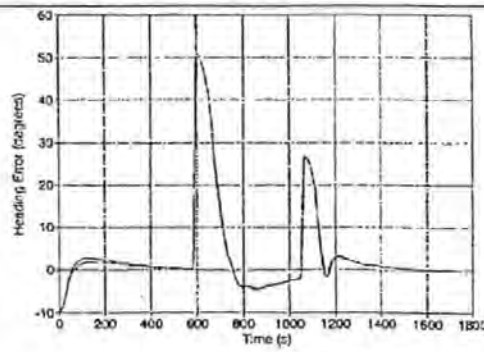


figure 11: Heading Error - Low Speed Approach

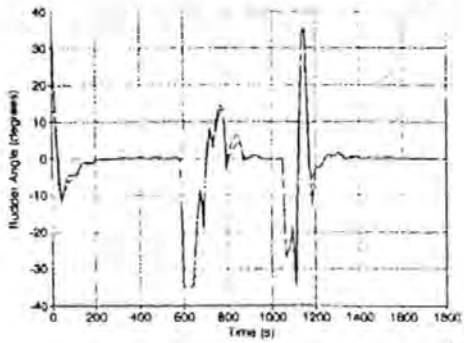


figure 12: Rudder Angle - Low Speed Approach

1.2.5.2. High Speed Approach

Here the speed of approach was 7.717 m/s. Figures 13, 14 and 15 show the cross track error, heading error and demanded rudder respectively. As with the low speed approach, there is good correlation between the two controllers for rudder and heading, with an offset on the cross-track data. At high speeds, it can be seen that the rudder excursions are far smaller, indicating that both controllers have adapted to the change in vessel dynamics.

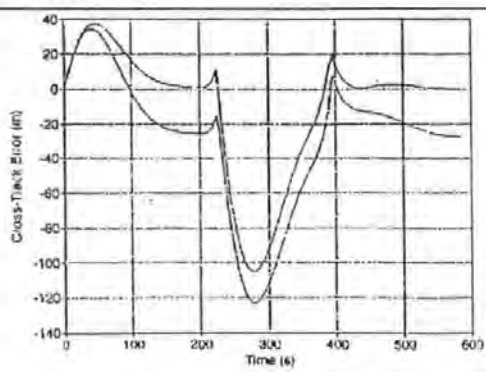


figure 13: Cross-Track Error - High Speed Approach

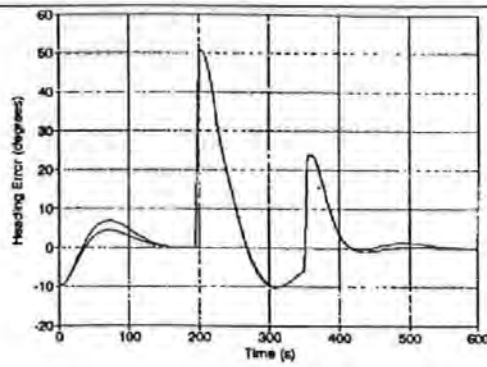


figure 14: Heading Error - High Speed Approach

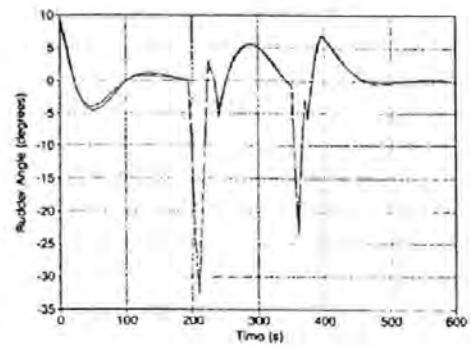


figure 15: Rudder Angle - High Speed Approach

1.2.5.3. Controller Performance

In order to compare the performance of the two controllers, a set of generalised performance indices may be defined:

$$J_y = \int_{t_0}^{t_1} (y_D - y_A)^2 \, dt$$

$$J_\psi = \int_{t_0}^{t_1} (\psi_D - \psi_A)^2 \, dt \qquad (10)$$

$$J_\delta = \int_{t_0}^{t_1} \delta_A{}^2 \, dt$$

Table 2 gives the comparative results.

speed	Optimal	Controller		Neural	Controller	
(m/s)	J_y	J_ψ	J_δ	J_y	J_ψ	J_δ
2.572	0.764E6	111.4	67.2	0.968E6	111.7	63.6
7.717	0.852E6	43.2	61.1	0.138E7	43.9	63.4

table 2: Comparative Results

From Table 2 it can be seen that in terms of heading and rudder, the two controllers perform in a similar manner, but the optimal controller provides better track-keeping performance.

1.2.6. Conclusions

The results of this initial study demonstrate that a neural network may be trained from data provided by an optimal guidance system. The trained network performs in a slightly sub-optimal manner - but has the advantage that it does not have to re-compute controller parameters for different forward speeds. At this stage it is not known how the network would cope with another way-point configuration.

The properties of multi-layer neural networks are not yet fully understood. It would appear however, that a ship guidance system is a potential application of the technique. There is extensive scope for further research in this field, particularly in the design of unsupervised learning networks that adapt in an on-line manner.

1.3. References

/1/ Panel on Human Error in Merchant Marine Safety.:
"Human Error in Merchant Marine Safety". National Academy of Science, Washington DC 1976.

/2/ M. Endo, J. van Amerongen, A. W. P. Bakkers: "Application of Neural Networks to Ship Steering"

/3/ Yoh-Han Pao: Case Western Reserve University "Adaptive Pattern Recognition and Neural Networks" Addison- Wesley Publishing Company, Inc.

/4/ R S Burns: "The Design, Development and Implemaentation of an Optimal Guidance System for Ships in Confident Waters". Proceedings of the 9th Ship Control Systems Symposium, Vol 3, Bethesda 1990, USA.

/5/ R S Burns: "Application of the Riccati Equation in Control and Guidance of Marine Vehicles", Proceedings of Workshop on the Riccati Equation in Control, Systems and Signals. International Federation of Automatic Control, Como, Italy 26-28th June 1989.

/6/ R Richter, R S Burns: "An Artificial Neural Network Autopilot for Small Vessels", UKSS 93, 13-15th September 1993, Keswick, UK, The Society for Compoter Simulation, 1993.

R. S. Burns and R. Richter.

The application of neural networks for the modelling of process dynamics.

In Dr Martyn Polkinghorne, editor, *Applications of Artificial Intelligence for Technological and Business Processes*, volume 2 of *Technology Transfer Series*, pages 10–17. University of Plymouth, University of Plymouth, Drake Circus, Plymouth, Devon PL4 8AA - UK, 1996. ISBN 0-905227-57-3.

The Application of Neural Networks for the Modelling of Process Dynamics

Professor Roland Burns and Mr Ralph Richter
School of Manufacturing, Materials and Mechanical Engineering

1 Introduction

The classical approach to modelling the dynamic behaviour of rigid bodies is to express their behaviour as a set of simultaneous linear and non-linear differential equations, and to obtain a solution for various input stimuli. An alternative approach is that of system identification whereby a given input such as a step, sinusoid or pseudo-random binary sequence (PRBS) is applied to the real system and from a set of input/output measurements using such techniques as linear least squares and maximum likelihood analysis a mathematical model may be obtained. This paper investigates the generation of a state variable representation of a ship in three degrees of freedom by the application of an Artificial Neural Network (ANN).

ANNs have been shown to demonstrate the capability to model highly complex plants. By the application of training data derived from the real environment, these networks can lean to emulate a wide range of differing conditions. Once trained, the neural network substitutes the plant and performs instead. The technique can be used not only to simulate the process dynamics of real systems, but also to act as a reference system model in adaptive and predictive control situations.

When considering motion control, the neural network philosophy is of particular interest. Using the non-linear time-invariant dynamic characteristics of a maritime vessel, a neural network is developed to model and control the motion of this process.

Using a carefully selected range of manoeuvres undertaken at various forward speeds, a comparison can be made between the conventional ship model and the neural network model developed.

2 Artificial Neural Networks

Artificial neural networks represent a powerful tool for simulating an understanding of complex relationships between patterns. Pattern can be understood not only as image, but also as number (vector, matrix) of data. The relationship between such vectors is often either not fully known or very difficult to describe using mathematical terms.

The 'genius' of the human brain to understand and to explain situations which are considered fascinating to biologists and engineers. First publications on neural computing was published in the early 1940's by *Frank Rosenblatt, Warren McCulloch, Norbert Wiener, Walter Pitts*. The importance of studies in the field of neuro-medicine is reflected by the number of Nobel prizes awarded to those researching neurology. Between 1901 and 1991, approximately 10%

of the prizes in medicine and physiology were awarded to researchers, whose work contributed directly to the advancement of neurological medicine.

It is the intention of this study to underline the ability of artificial neural networks to handle complex situations in addition to the biological neural network. A neural network has been designed to find (learn) and recall the behaviour of a large motorised marine vessel. It was determined that the initial task was to break down the problem into smaller sub units.

3 Mathematical Background of ANNs

To understand the actions and algorithms concerned with neural computing it is necessary to consider biological neural nets and their architecture.

A *neuron* is the basic element of the brain. A diagram of a neuron is detailed in Figure 1.

The structure of the brain is an interconnection of a very large (tens of billions) number of neurons. The transmission of signals in the brain is chemical in nature. Each neuron receives an input signal from other neighbouring neurons. The connection path between two neurons is called an *axon* and the incoming ports *dendrites*.

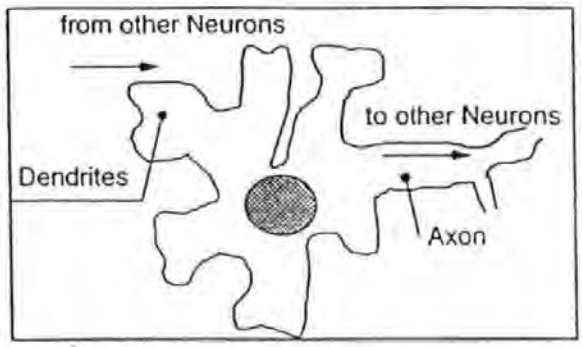


Figure 1 Structure of a biological neuron

The connections between axons and dendrites are called *synapse* (Figure 2). In order to understand the biological model, the axon is an electrical cable and the dendrites is a socket. To carry information a link is required. The synapse, the link or plug, changes the effectiveness of the incoming spike.

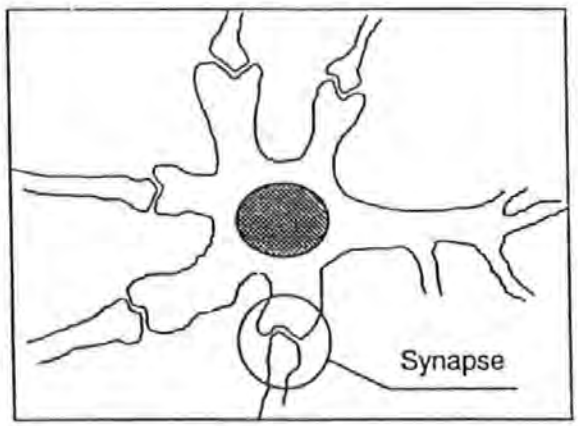


Figure 2 Synapse

During the learning phase the efficiency of the synapse is modified. The sum of the incoming signals, the *total input*, is used by the receiving neuron to generate an output. This output of one neuron is the input for many other neurons except those neurons in the output layer.

The artificial neuron is a simple model of the biological neuron which has the form as displayed in Figure 3.

The label of the signals depends on your view point. Assuming the present neuron, all incoming signals are called x and the output is called y, this y, or output, is then an incoming signal for the next neuron and is then called x. As demonstrated, the synapse is modelled as a modifiable weight which is associated which each axon (connection to a neuron). The neurons output formed by the transfer function is a single number that represents the rate of firing - the activity of the neuron. To compute the output, the neuron multiplies each incoming signal by the associated weight and adds

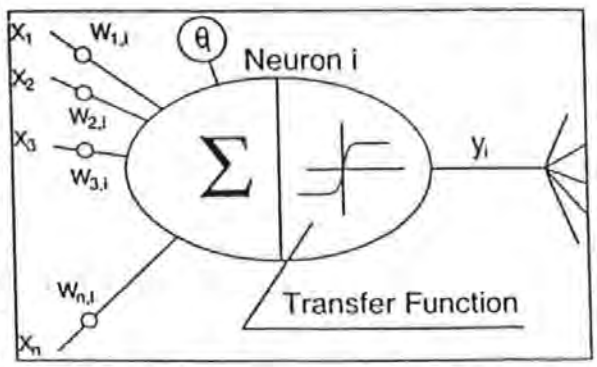


Figure 3 Main structure of an artificial neuron

together all these weighted inputs to form the total input and uses this to create the output by using the transfer function. The reaction of the artificial network depends on both the transfer function used and the weights.

The output of the neuron in the mathematical sense is defined as:

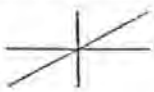
$$I_i^k = \sum_{j=1} x_j^{k-1} \cdot w_{j,i}^k + \theta_i^k \qquad \text{(Equation 1)}$$

$$x_i^l = f\left(I_i^l\right) \qquad \text{(Equation 2)}$$

θ^k_i	the threshold, which moves the transfer function (graph) in the horizontal direction.
x^{k-1}_j	output of neuron j in the previous layer
$w^k_{j,i}$	weight between neuron i in layer k and the neuron j in layer k-1
I^k_i	total input of neuron i in layer k

$y_i^k = f\left(I_i^k\right)$ where $f\left(I_i^k\right)$ (transfer function) could be:

linear $f\left(I_i^k\right) = I_i^k$



$f\left(I_i^k\right) = \tanh\left(I_i^k\right)$

Sigmoid function

$$f\left(I_i^k\right) = \frac{1}{1+e^{-I_i^k}}$$

hyperbolic tangent

$$f\left(I_i^k\right) = \begin{cases} -1 & I_i^k \le 0 \\ +1 & I_i^k > 0 \end{cases}$$

hard limiter or threshold function

4 Network Architecture

In the past, many forms of neural nets and their algorithms were investigated. Serious investigations started in 1943, by the head neuro-biologist *Warren McColloch* and statistician *Walter Pitt*. The paper [3] tangents fields like digital computing, 'electronic brains' and macroscopic intelligence. The first conference on artificial intelligence was organised in 1956 by famous names such as *Marvin Minsky*, *John McCarthy*, *Claude Shannon* and *Nathanial Rochester*.

To simulate the behaviour of the human brain we need a *network* of neurons, a so called neural network (Net). The neurons are usually organised into groups called layers. A neural net consists of at least an input and an output layer and eventually hidden layer(s). In order to understand the following facts, with 'single' we mean the number of hidden layers. In practise, a single layer net consists of three layers, these being one input and one output layer with a *single* hidden layer. The words *one* and *single* are synonyms for each other. Simple tasks can be solved by a one layer network but for difficult problems a *multi layer network* (Figure 4) is required.

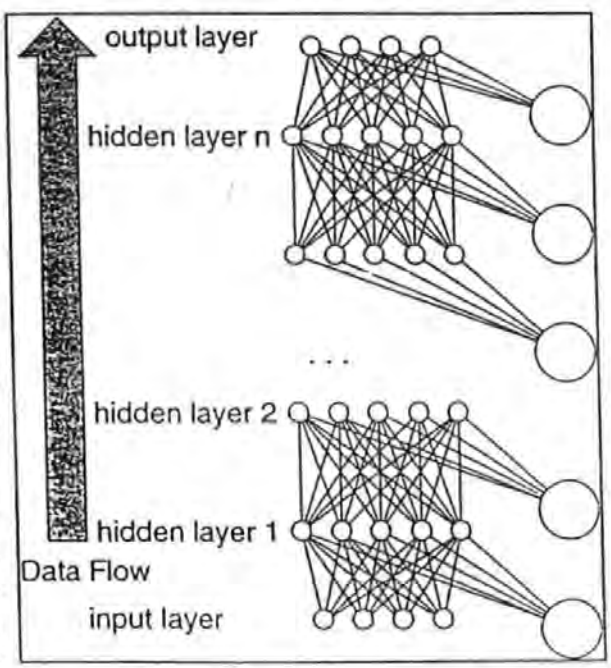


Figure 4 Multi layer network

The behaviour of a multi layer net in general is very similar to a single layer net. The user has to find the optimum network size to be satisfied with the derived results and the speed computation. A small net may be faster but if the task is too difficult then important

information may be lost, conversely, if the net is too large, then the output may become noisy and the subsequent computing speed, especially during the learning, is slow.

Rumelhart's contributions to neural nets ([4]) are fundamentals for further investigation.

The method of supervised learning utilised the *back propagation* algorithm (Figure 5).

The neural net used in this algorithm is a multi layer net and will be the *Sigmoid* transfer function. The back propagation rule requires the *error* between computed output by the net (straight forward or phase 1) and the desired output given by the 'teacher'. To adjust the weights on the path between one neuron and the next neuron, the error is back propagated, starting with the output layer back to the first layer after the input layer. This process is the second, or learning, phase. The process - computing forward and error propagation backwards - is repeated with different pairs of training data until a maximum number of data is reached or the maximal error approaches an error, i.e. $\varepsilon = 0.05$.

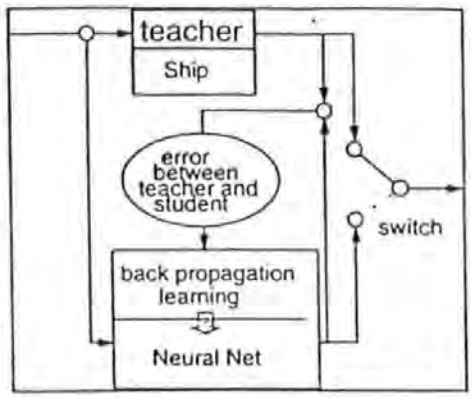


Figure 5 Layout

The interesting feature of back propagation is that we do not need any prior knowledge about the process. However, conversely this may prove to be a significant disadvantage because our student does not have any self organising capabilities and so can not be produce a response that is an improvement on that of the teacher.

$$I_i^l = \sum_j w_{i,j}^l x_j^{l-1} + \theta_i^l \qquad \textit{straight forward} \text{ (phase 1)}$$

$$x_i^l = Sigmoid\left(I_i^l\right)$$

$$\tilde{w}_{j,i}^l = w_{j,i}^l + \Delta w_{j,i}^l \qquad \text{(Eq. 3)}$$

$$\Delta w_{j,i}^l = \eta \delta_j^l x_j^{l-1} \qquad \textit{back propagation} \text{ (phase 2) (Eq. 4)}$$

$$\delta_j^l = x_i^l\left(1 - x_i^l\right) \cdot \left(d_i - x_i^l\right) \text{ output layer} \qquad \text{(Eq. 5)}$$

$$\delta_j^l = x_i^l\left(1 - x_i^l\right) \cdot \sum_k \delta_k^{l+1} w_{j,k}^{l+1} \text{ inner neurons} \qquad \text{(Eq. 6)}$$

η	learning coefficient
x_i^l	output of neuron i in layer l
δ_i^l	error of neuron i in layer l
I_i^l	total input of neuron i in layer l
w_{ij}^l	weight on path from neuron j in layer l-1 to neuron i in layer l
Δw_{ij}^l	weight increment for w_{ij}^l
Sigmoid()	transfer function

The learning rules for the thresholds θ are the same as the rules for the weights. The threshold is a weight with the associated input of 1.0 .

5 Ship Mathematical Model

Ship motions in surge, sway and yaw can be described [5] by an Eulerian set of non-linear differential equations of the form :

Surge Equation: (Equation 7)

$$m\dot{u} + mqw - mrv = X_u\dot{u} + X_u\left(u + u_c\right) + X_{uu}u^2 + X_{uuu}u^3 + X_{vv}v^2 + X_{rr}r^2 + X_{\delta\delta}\delta_A^2 + X_{un}un_A + X_{nn}n_A^2 + X_{ua}u_u + X_{zz}z^2 + X_{\theta\theta}\theta^2$$

Sway Equation: (Equation 8)

$$m\dot{v} + mur - mpw = Y_v\dot{v} + Y_v\left(v + v_c\right) + Y_r\dot{r} + Y_r r + Y_{nn}n_A^2 + Y_{vvv}v^3 + Y_{vvr}rv^2 + Y_{nn\delta}n_A^2\delta_A + Y_{nn\delta\delta\delta}n^2\delta_A^3 + Y_{\delta vv}\delta_A v^2 + Y_{va}v_a$$

Yaw Equation: (Equation 9)

$$I_z\dot{r}+(I_y-I_x)pr = N_v\dot{v}+N_v(v+v_c)+N_r\dot{r}+N_{nn}n_A^2+N_{vvv}v^3+N_rr+N_{rrv}rv^2$$
$$+N_{nn\delta}n_A^2\delta_A+N_{nn\delta\delta\delta}n_A^2\delta_A^3+N_{\delta vv}\delta_A v^2+N_{vu}v_a$$

Equations (7) to (9) can be arranged in the state matrix vector form: (Equation 10)

$$\dot{\mathbf{x}}(t)=\mathbf{F}(t)\mathbf{x}(t)+\mathbf{G}_c(t)\mathbf{u}(t)+\mathbf{G}_D(t)\mathbf{w}(t)$$

The corresponding discrete solution is: (Equation 11)

$$\mathbf{x}((k+1)T)=\mathbf{A}(T,kT)\mathbf{x}(kT)+\mathbf{B}(T,kT)\mathbf{u}(kT)+\mathbf{C}(T,kT)\mathbf{w}(kT)$$

where:

$$\mathbf{x}^T=(\delta_A\ n_A\ x\ u\ y\ v\ z\ w\ \phi\ p\ \theta\ q\ \psi\ r) \quad \text{(Equation 12)}$$

$$\mathbf{u}^T=(\delta_D\ n_D) \quad \text{(Equation 13)}$$

$$\mathbf{w}^T=(u_c\ v_c\ u_a\ v_a\ \zeta_x\ \zeta_y) \quad \text{(Equation 14)}$$

For this study, it was necessary to concentrate on three degrees of freedom. These being surge, sway and yaw.

6 Ship Model Application

The vessels parameters used in this simulation are given below (Table 1), and are based on the Morse and Price data for the *Mariner Hull* [6].

Table 1: Vessel Parameter

Length	=	161m
Draught	=	9m
Beam	=	23m
Displacement	=	17,000t
Number of propellers	=	1
Number of rudders	=	1
Maximum rudder angle	=	35°

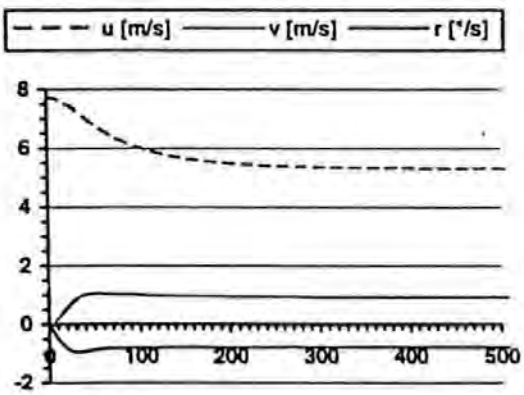


Figure 6 Settling Times

A neural network is required to model the behaviour of large ships. The precise relationships between many of the features and characteristics of these ships are not fully understood. To determine them, it is possible to employ a neural network. Rudder angle, and engine speed cause speed changes in the surge, sway and yaw directions (u, v, $\dot{\psi}$). Since we are not only interested in the steady state response of the vessel, but also in the transient behaviour, it is essential to consider the time elapsed since the last rudder change as an other input. Figure 6 indicates the various time periods required for the response to settle down.

Table 2 settling times

δ [°]	u	t [s]	v	t [s]	yaw rate	t [s]
0	7.588	275	0.278	410	-0.186	405
10	6.639	270	0.734	220	-0.534	180
-10	6.818	270	-0.669	230	0.476	230
20	5.915	295	0.870	240	-0.681	185
-20	6.043	290	-0.836	235	0.641	240
30	5.308	290	0.930	240	-0.782	155
-30	5.408	285	-0.907	240	0.747	165

Utilising an acceptable error of ±1%, we can determine from the data (Table 2) the following values. Therefore it is possible to state that if the time considered is bigger than the time to reach steady state, then the response has reached steady state, otherwise the response remains in the transient period and the operation of the artificial neural network is required.

7 Structure of the ANN

It is a pre-requisite that the variables to be investigated are considered before commencing design of the network's structure (see Table 3).

To learn the transient behaviour, it is necessary to determine the time elapsed since the last rudder change as an additional further input. Thus, the interface to the outside world is defined.

Table 3 Structure of the Network

Inputs		Outputs		
rudder angle	engine speed	forward speed	lateral speed	turning rate
δ		u	v	$\dot{\psi}$

A 3-6-6-6-3 network was identified to be suitable for this application. The quality of results obtained from a two hidden layer network proved unsatisfactory. Obviously, the transients, with their associated overshoots, are difficult to understand, and were therefore filtered out. Using more than two hidden layers the error is reduced and overshoots were replicated giving a suitable level of network performance.

8 Network Training

The learning method utilised for this study was the back propagation algorithm. This algorithm is based on the minimisation technique called steepest descent or gradient method [1]. The transfer function employed was the popular *Sigmoid* function. The output were in the limits between 0.0 and 1.0 ($0.0 < y < 1.0$), where those values are reached at infinity. Therefore, the desired outputs had to be scaled within these limits. During the learning process the trend of the error development was observed and it could be seen that the network stuck in local minima. By increasing the number of hidden layer, the error surface contains less troughs and a more constant learning was achieved. Furthermore the learning rate was adjusted from an initial large learning rate with gradual decrements until the finished level of learning was achieved (steady error).

9 Training Results

Results of the learning are given in Figure 7 to Figure 9. Figure 7 displays the forward speed and demonstrates how the response of the network closely follows that of the surge rate training data. An improved level of performance is identified by the response for sway rate data, and also for that of yaw rate with increments in rudder angle of 0°, -10°, +10°, -20, +20°, -30° and +30° is displayed.

Figure 7 SurgeVelocity Response

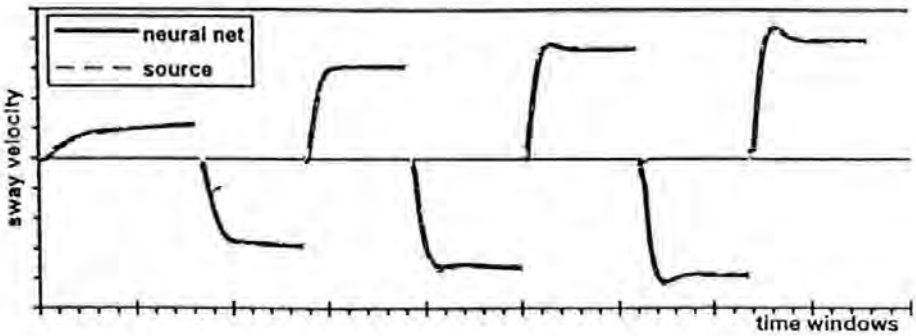


Figure 8 Sway Velocity Response

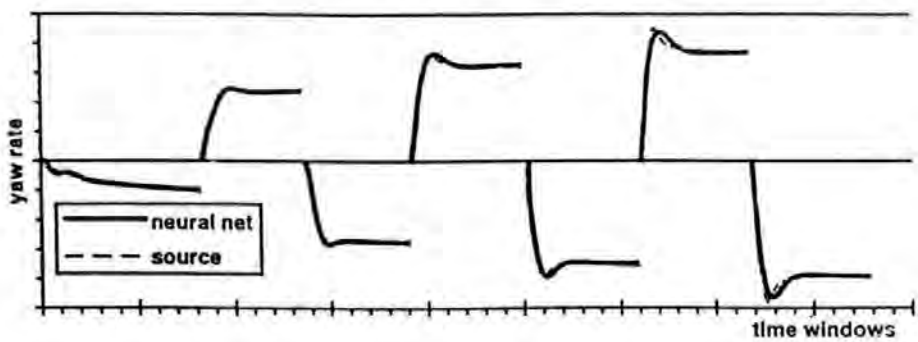


Figure 9 Yaw Rate Response

The actual outputs match very closely the desired outputs given in the training sets which clearly demonstrates the learning success of the network design utilised. Further work by the authors will concentrate on the implementation of this design of network during simulated sea trial conditions. Results will then be compared to those obtained from a traditional ship model to validate both the learning achieved, and the subsequent performance capability obtainable during simulation studies.

10 Conclusions

It has been demonstrated by this study that it is possible to simulate complex plant behaviour utilising neural networks. The advantage of employing a simulation using this technique is that it becomes possible to overcome the problems associated with formulating the relationship between the features to be investigated. This can be achieved by the neural network, thus allowing the designer to concentrate on alternative aspects of the design. The authors consider that the computational speed of the network far exceeds the required time for conventional differential equations because a significant amount of the training is undertaken off-line. During execution, the neural solution therefore allows for extension to far more complex mathematical models without incurring a notable slowing of the process time.

11 References

[1] **Yoh-Han Pao:**
Adaptive Pattern Recognition and Neural Networks, Addison Wesley Publishing Company Inc.

[2] **Ralph Richter, Dr Roland Burns:**
An Artificial Neural Network Autopilot for Small Vessels, proc. of UKSS '93, Keswick, UK.

[3] **Warren McCulloch, Walter Pitts:**
A Logical Calculus of the Ideas Immanent in Nervous Activity, Bulletin of Mathematical Biophysics, vol. 5, pp. 115-133.

[4] **D. Rumelhart, J. McClelland:**
Parallel Distributing Processing, MIT Press, Cambridge, Massachusetts 1986.

[5] **Burns, R. S.:**
Application of the Riccati Equation in Control, Systems and Signals, International Federation of Automatic Control, Como, Italy, 26-28th June 1989.

[6] **Morse, R. V.; Price, D.:**
Manoeuvring Characteristics of th Mariner Type Ship (USS Compas Island) in Calm Seas, Sperry Polaris Management, Sperry Gyroscope Company, New York, 1961.

R. S. Burns and R. Richter.

A neural network approach to the control of surface ships.

Journal Control Engineering Practice, 4(3):411–416, 1996. Special section on Control Applications in Marine Systems.

A NEURAL NETWORK APPROACH TO THE CONTROL OF SURFACE SHIPS

Dr Roland Burns and
Mr Ralph Richter

School of Manufacturing, Materials and Mechanical Engineering
University of Plymouth, Plymouth, Devon, UK

Abstract: Conventional ship autopilots are based on proportional, integral and derivative (PID) algorithms, and are generally set to work under specific conditions. Changes in either the vessel's handling characteristics or environmental conditions means that the system is not working at its optimal point. This paper explores the possibility of developing two neural network autopilots based on training data derived from:

a) a small vessel operating in a range of sea states, using differently tuned PID controllers for each sea state.

b) an optimal guidance system for a large ship sailing in calm water at varying forward speeds.

It is demonstrated that with the small vessel, a single neural network can cope with a range of sea states without the need for re-tuning. In the case of the large vessel, the trained network performed in a slightly sub-optimal manner - but had the advantage that it was not necessary to re-compute controller parameters at different forward speeds.

Keywords: neural control, navigation systems, optimal control, mathematical models, marine systems

1. INTRODUCTION

Many collisions and groundings of marine vessels occur in the approaches to a port where the traffic density is intense. This suggests that there is a need for automatic guidance systems to deal with the problems of surface ships manoeuvring in confined waters, possibly under shore control, as part of the port's vessel traffic services.

Conventional ship autopilots are based on proportional, integral and derivative (PID) control algorithms, and are used to control the ship's heading in an open seaway. These controllers are developed to work under specific conditions, and so they are not working at their optimal point and need to be reset to take into account the vessel's handling characteristics and environmental conditions. It is not current practice to use an automatic system in the approaches to a port, and in many cases control of the vessel is handed over to a pilot at this stage. However, it is in the pilotage phase of the voyage, where the traffic density is intense, that the risks of collision and grounding are highest. In addition, it has been highlighted in the 'Panel on Human Error in Merchant Marine Safety' (1976) that over 80 percent of all marine accidents are due to human error.

Modern sea-going vessels have a range of navigation aids, including global positioning system (GPS) receivers, Doppler sonar, and gyrocompasses, as well as hypobolic aids such as Loran C and Decca. Current trends include the use of standard interfaces to network communication systems using computer tools such as electronic charts to form integrated navigation systems. It is also possible to employ the navigational data to provide best estimates of state

vectors (Kalman filter) and optimal guidance strategies. Such techniques require powerful computing facilities, particularly if the dynamic characteristics of the vessel are changing, as may be the case in a manoeuvring situation or changes in forward speed.

The feasibility of neural networks to steer ships has been studied by Endo *et al.* (1989).

Section 2 of this paper considers the scenario of a small vessel, operating under different environmental conditions, depicted here by a range of sea states. To maintain optimality, a PID autopilot is tuned to handle each sea state. The complete data set from this exercise is then used to train a neural network, the intention being that the single network can then perform over the complete range of sea states without the need for re-tuning.

Section 3 of this paper investigates the possibility of training a neural network to behave in the same manner as an optimal ship-guidance system, the objective being to provide a system that can adapt its parameters so that it provides optimal performance over a range of conditions, without incurring a large computational penalty.

A series of simulation studies have been undertaken to compare the performance of a trained neural network with that of the original optimal guidance system over a range of forward speeds. It is demonstrated that a single network has comparable performance to a set of optimal guidance control laws, each computed for different forward speeds.

With the advent of more powerful computers, an increasing number of modern techniques has been used such as neural computing (neural nets) and fuzzy logic for on-line control.

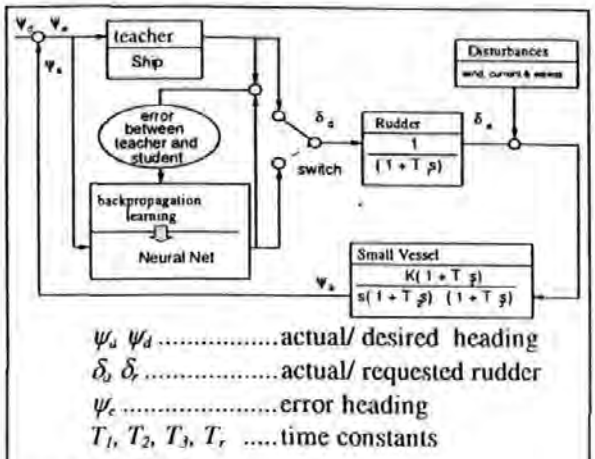
The main purpose of a modern autopilot is to merge all the beneficial features of several controllers to create an intelligent controller, i.e. with a behaviour like a human helmsman.

2. AN ARTIFICAL NEURAL NETWORK AUTOPILOT FOR SMALL VESSELS

2.1. *Creation of the Training Data*

The idea on which this study is based is that one PID controller is tuned for one particular sea state and this *tuned PID controller* is then used as one teacher for the neural network (see Fig. 1).

If a training file, which contains input and output data of the PID controller, is created, the neural net will learn to respond like its teacher. But if the training data file consists of data pairs from more than one teacher, i.e. data from several tuned PID controllers in several sea states, the neural net will learn the behaviour of the tuned PID controllers at the optimal point, or close to it. It is known that the



ψ_a, ψ_d .....actual/ desired heading
δ_a, δ_r .....actual/ requested rudder
ψ_e .....error heading
T_1, T_2, T_3, T_r .....time constants

Fig. 1 Structure of the system

ship parameters such as weight, inertia, draught and speed, have key effects on the behaviour of the ship. So, if necessary, the PID controller can be tuned for more specific situations and create more relevant data.

The PID controllers, used as teachers, are tuned firstly for a very small heading error and not for a smooth rudder movement. Tests have proved that the neural net will work as a damper too.

The training file for the neural net contains PID information for differently tuned controllers in the associated sea states from port and starboard directions, and the current outputs of the controllers. Further difficulties can arise when the inputs of the net have very big differences in the values.

Suppose the heading error and the rate of change of the heading error are in the order of 10^{-1}, and less than 10^{1}, and the integral of the heading error is bigger than 10^{2}, emphasis will be placed on the input neuron for the integral and the small changes of the other two neurons are not taken into consideration. In this case, the net will only learn the rudder offset to remove the average disturbance without alternating.

2.2. *Training the Network*

The net consists of 10 neurons, in each of the 2 chosen hidden layers. The architecture of the net is shown in Fig. 2.

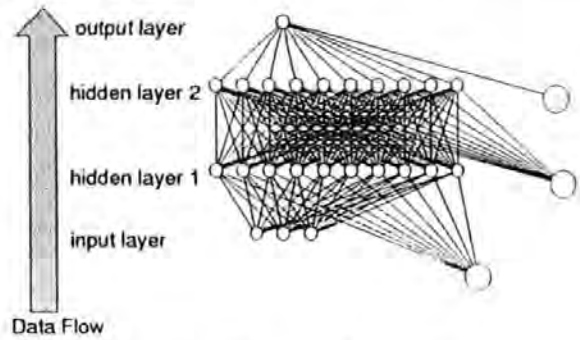


Fig. 2 Structure of the neural network

During the learning stage, the training sets are randomly selected until the given number, in this case 60,000, is reached. It is possible to formulate the stop condition in association with the actual error between the computed output of the net and the desired output given by the teacher.

2.3. Results

The following graphs depict the learning process and the comparison of a trained neural network to a PID controller.

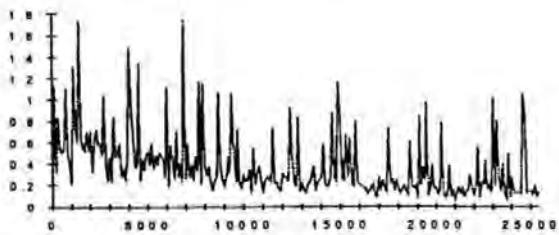


Fig. 3 Learning

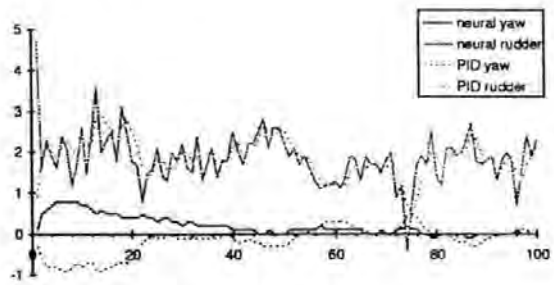


Fig. 4 Comparison of neural network to PID in sea state 3

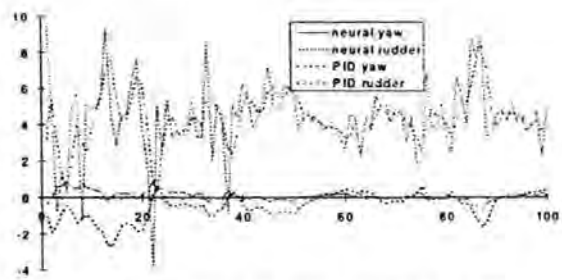


Fig. 5 Comparison of neural network to PID in sea state 4

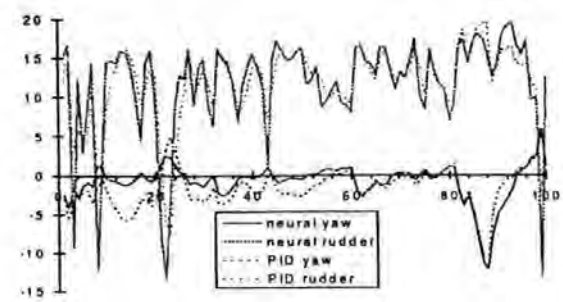


Fig. 6 Comparison of neural net to PID in sea state 5

Table 1 Results

Sea State	3	4	5
RMS Yaw Error Neural Net	0.103	0.197	1.863
RMS Yaw Error PID	0.194	0.579	2.387

It is clear that the neural network is able to react in the manner of its teacher(s). It is possible to train a neural network with the data from more than one teacher, and the network will pick up the behaviour of all the supervisors.

3. OPTIMAL SHIP GUIDANCE USING A NEURAL-NETWORK APPROACH

3.1. Background

This suggests that there is a need for automatic guidance systems for marine vehicles in confined waterways, such as many of the world's major ports, even to the extent of allowing a transfer of control from ship to shore using the port's vessel traffic services (VTS). As electronic navigation aids become more sophisticated, and the use of satellite Global Positioning System (GPS), particularly used in differential mode, becomes more widespread, the concept of fully automatic pilotage in port approaches becomes a tangible reality.

It has been demonstrated (Burns, 1990) that it is possible to design an optimal multivariable ship guidance system that controls position, heading and speed simultaneously, and that such a system can work within the constraints required in port approaches.

By the use of multivariable system theory, it is possible to construct a mathematical model of a surface ship that can respond to control inputs (rudder and main engines) and also to disturbance inputs (wind, waves and current). Such a mathematical description normally requires a set of non-linear differential equations.

Based on a multivariable model with a control vector **u**, a disturbance vector **w** and a state vector **x**, an optimal control policy may be formulated that minimises a performance index, or cost function. A problem with this approach is that if the dynamic characteristics of the vessel change (due to variations in forward speed, for example) then the guidance system is sub-optimal, and its parameters need to be re-computed. This places a large computational burden on the ship's navigational computer, which must perform its calculations during the sample period.

3.2. Ship Mathematical Model

Ship motions in surge, sway, heave, roll, pitch and yaw can be described by a Eulerian set of non-linear differential equations of the form:

$$\begin{aligned} m(\dot{u} + qw - rv) &= X \\ m(\dot{v} + ru - pw) &= Y \\ m(\dot{w} + pv - qu) &= Z \\ I_x\dot{p}(I_z - I_y) &= L \\ I_y\dot{p}(I_x - I_z) &= M \\ I_z\dot{p}(I_y - I_x) &= N. \end{aligned} \quad (1)$$

The terms X, Y, Z, L, M and N represent all the external forces and moments acting on the hull, and include both linear and non-linear components. These equations can be arranged as a set of state equations in terms of the state vector **x**, control vector **u** and disturbance vector **w**, where:

$$x^T = \{\delta_A \;\; n_A \;\; x \;\; u \;\; y \;\; v \;\; z \;\; w \;\; \phi \;\; p \;\; \theta \;\; q \;\; \psi \;\; r\} \quad (2)$$

$$u^T = (\delta_D \;\; n_D) \quad (3)$$

$$w^T = (u_c \;\; v_c \;\; u_a \;\; v_a \;\; \zeta_x \;\; \zeta_y). \quad (4)$$

The vessel used in the simulation had the following parameters:

Length	= 161 m
Draught	= 9 m
Beam	= 23 m
Displacement	= 17000 tonnes
Number of propellers	= 1
Number of rudders	= 1
Maximum rudder angle	= ±35 degrees.

The dynamic characteristics of the vessel may be described in terms of its open-loop eigenvalues. When u = 7.717 m/s (15 knots), these are:

s = -0.5, -0.039 , -0.0755,
0, 0, -0.5, 0 -0.00913. (5)

When the vessel is travelling at 2.572 m/s (5 knots), they become:

s = -0.5, -0.013, -0.0252,
0, 0, -0.5,0 -0.00265. (6)

These results are shown in Fig. 7, and demonstrate that the vessel becomes less manoeuvrable at low speeds, thus requiring a control policy that takes this into account.

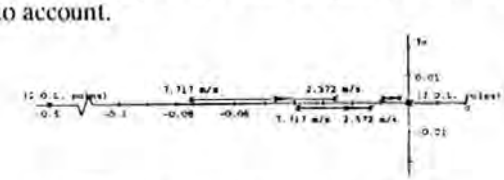


Fig. 7 Effect of forward speed on open-loop eigenvalues

3.3. Optimal Guidance Policy

Given the state equations:

$$\dot{x}(t) = F(t)x(t) + G_c(t)u + G_D(t)w(t) \quad (7)$$

and the quadratic criterion to be minimised:

$$J = \int_{t_0}^{t_1}\left\{(x-r)^T Q(x-r) + u^T Ru\right\}dt, \quad (8)$$

where **r** is the desired state vector, it can be shown (Burns, 1989) that the optimal control is:

$$u_{opt} = -\left(Sx + R^{-1}G^T m\right) \quad (9)$$

where **S** is the optimal feedback gain matrix calculated from solution of the Riccati equations and **m** is the command vector, computed from a knowledge of the system model and the desired state vector.

Figure 8 illustrates the departure of the eigenvalues from their assigned positions as the forward speed is reduced from 7.717 m/s to 2.572 m/s. To maintain a fixed closed-loop eigenvalue array, it is necessary to re-compute the feedback matrix **S** at each forward speed.

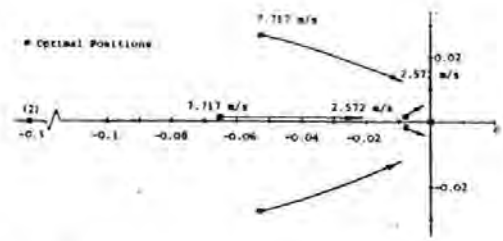


Fig. 8 Eigenvalue departure during a speed reduction

Figure 9 shows a simulated approach into the Port of Plymouth using the assigned closed-loop eigenvalues. The problem of way-point overshoot is overcome by using way-point advance and dual-mode control. Under a dual-mode policy, when the advanced way-point is reached, the controller switches from track to course weighting, thus suppressing y_t and emphasising ψ_c.

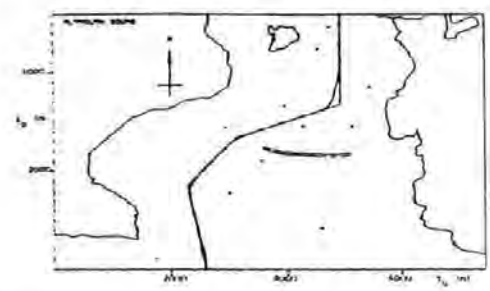


Fig. 9 Simulated approach into the port of Plymouth

3.4. Neural-network Guidance System

Supervised Learning using Backpropagation The output of each artificial neuron is calculated by multiplying each incoming signal by an associated weight, adding together all the weighted inputs to form the total input, and using a Sigmoid function to create the output.

A neural network consists of at least an input layer, an output layer and a single intermediate, or hidden layer. All neurons are interconnected. A single hidden layer can be used for simple applications, but for more complex situations, multi-layer networks are employed.

The problem is to find the optimum size that gives the best balance between accuracy and speed (Richter and Burns, 1993).

In this application the network is trained using supervised learning. A subset of the state vector (only those terms that affect the rudder required) are input to both the optimal guidance system and the neural net, which both compute the required rudder, δ_D. Differences between the two values are used to train the network using backpropagation learning as shown in Fig. 1.

Training the Network Training data from the optimal guidance system over a range of speeds from 2.572 to 7.717 m/s was collected in the simulated approach to the Port of Plymouth. During learning, the training sets were randomly selected and run over a given interval, usually until 100,000 - 200,000 samples had been taken. Parameters such as momentum, learning coefficient and number of neurons in hidden layer were then varied until a global minimum error was achieved. For this application, this occurred for the following values:

input neurons	= 5
output neurons	= 1
number of hidden layers	= 2
neurons in hidden layer	= 10
learning coefficient	= 0.6
momentum	= 0.4
number of samples	= 200,000
learning time	= 2230 seconds.

3.5. Results

The weight coefficient matrices were used in an neuro-optimal hybrid controller to test the system, the neural network controlling the rudder, and the optimal guidance system controlling the main engines.

Low-speed Approach In the low-speed approach, the vessel commenced its run at 2.572 m/s. shows the cross-track error y_e for both the optimal and neural controllers. Figures 11 and 12 give the corresponding results for the heading error ψ_e and rudder δ_D. It can be seen that there is good correlation for heading and rudder, but there is an offset with the cross-track error. This is possibly due to the way in which the training data was scaled.

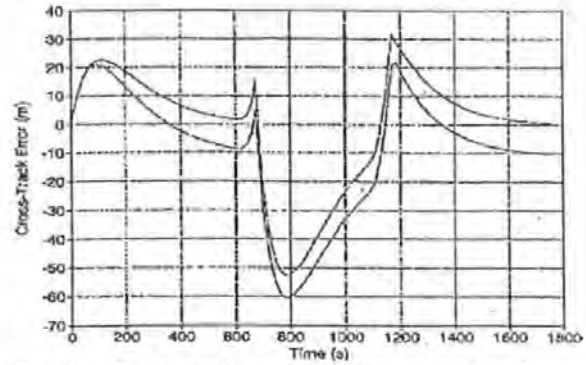


Fig. 10 Cross-track error - low-speed approach

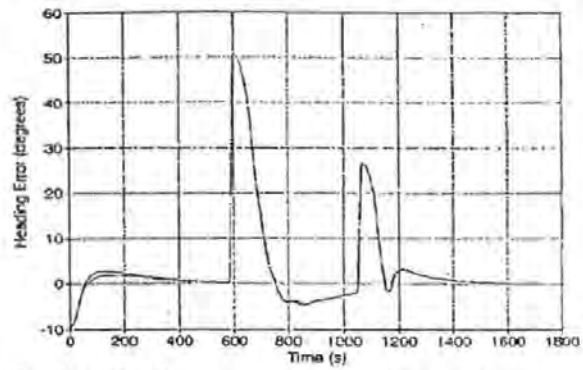


Fig. 11 Heading error - low-speed approach

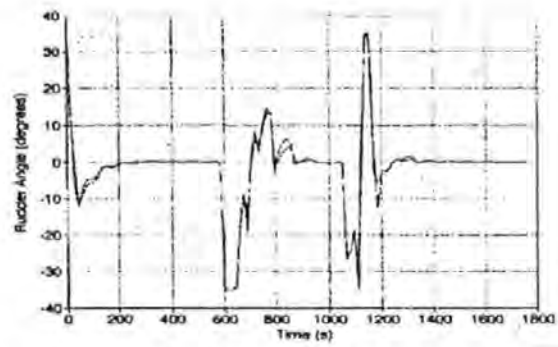


Fig. 12 Rudder angle - low-speed approach

The dual-mode operation can clearly be seen. At t = 580 seconds the first way-point is approached, and the controller drives the rudder hard-over as it enters the course-changing mode. When the heading error is 20° at t = 700 seconds, the track-keeping mode is resumed.

High-speed Approach Here the speed of approach was 7.717 m/s. Figures 13 to 15 show the cross track error, heading error and required rudder respectively. As with the low-speed approach, there is a good correlation between the two controllers for rudder and heading, with an offset on the cross-track data. At high speeds, it can be seen that the rudder excursions are far smaller, indicating that both controllers have adapted to the change in vessel dynamics.

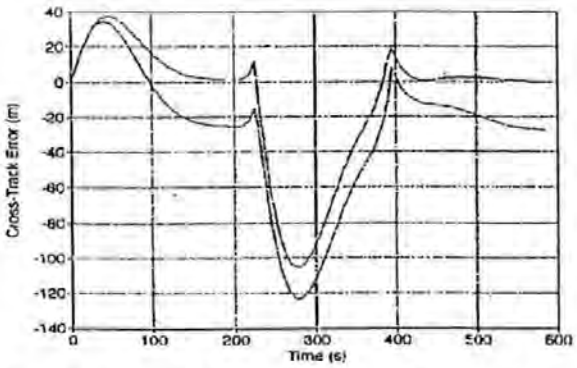


Fig. 13 Cross-track error - high-speed approach

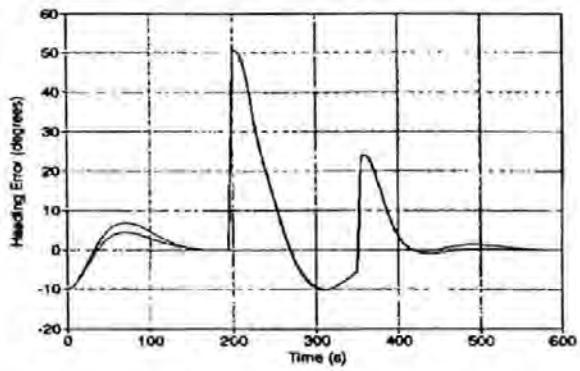


Fig. 14 Heading error - high-speed approach

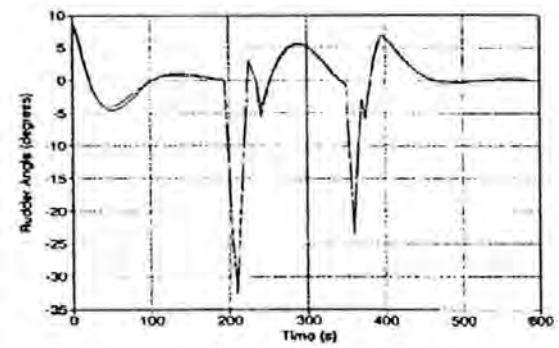


Fig. 15 Rudder angle - high-speed approach

Controller Performance In order to compare the performance of the two controllers, a set of generalised performance indices may be defined:

$$J_y = \int_{t_0}^{t_1} (y_D - y_A)^2 \, dt$$

$$J_\psi = \int_{t_0}^{t_1} (\psi_D - \psi_A)^2 \, dt \qquad (10)$$

$$J_\delta = \int_{t_0}^{t_1} \delta_A{}^2 \, dt$$

Table 2 gives the comparative results.

Table 2 Comparative Results

speed	Optimal Controller			Neural Controller		
(m/s)	J_y	J_ψ	J_δ	J_y	J_ψ	J_δ
2.572	0.764E6	111.4	67.2	0.968E6	111.7	63.6
7.717	0.852E6	43.2	61.1	0.138E7	43.9	63.4

From Table 2 it can be seen that in terms of heading and rudder, the two controllers perform in a similar manner, but the optimal controller provides better track-keeping performance.

4. CONCLUSIONS

The results of this initial study demonstrate that a neural network may be trained from data provided by an optimal guidance system. The trained network performs in a slightly sub-optimal manner - but has the advantage that it does not have to re-compute controller parameters for different forward speeds. At this stage it is not known how the network would cope with another way-point configuration.

The properties of multi-layer neural networks are not yet fully understood. It would appear however, that a ship's guidance system is a potential application of the technique. There is extensive scope for further research in this field, particularly in the design of unsupervised learning networks that adapt in an on-line manner.

5. REFERENCES

Burns, R. S. (1989). "Application of the Riccati Equation in Control and Guidance of Marine Vehicles", Proceedings of Workshop on the Riccati Equation in Control, Systems and Signals. International Federation of Automatic Control, Como, Italy 26-28th June 1989.

Burns, R. S. (1990). "The Design, Development and Implementation of an Optimal Guidance System for Ships in Confined Waters". Proceedings of the 9th Ship Control Systems Symposium, Vol 3, Bethesda, USA.

Endo, M., van Amerongen, J. and Bakkers, A. W. P. (1989). "Application of Neural Networks to Ship Steering", IFAC Workshop on Control Application.

Panel on Human Error in Merchant Marine Safety (1976). "Human Error in Merchant Marine Safety". National Academy of Science, Washington DC.

Richter, R. and Burns, R. S. (1993). "An Artificial Neural Network Autopilot for Small Vessels", UKSS 93, 13-15th September 1993, Keswick, UK, The Society for Computer Simulation.

R. S. Burns, R. Richter, and M. N. Polkinghorne.

A multivariable neural network ship mathematical model.

In *First International Conference on Marine Transport into the 21st Century (MarTrans '95)*, pages 747–756, Southampton - UK, 30 August – 1 September 1995. Computational Mechanics Publications, ISBN 1-85312-330-7.

Also published by: Computational Mechanics Publications, Boston MA, 1995, ISBN 1-56252-254-x, pp 747–756.

A Multivariable Neural Network Ship Mathematical Model

R S Burns, R Richter, M N Polkinghorne
School of Manufacturing, Materials and Mechanical Engineering,
Plymouth Teaching Company Centre, University of Plymouth, Devon, United Kingdom

Abstract

Conventional techniques to model plants require the utilisation of differential equations. The computation of such equations becomes slow in situations when the plants are highly complex. By taking training data from the real plant, it is possible to design and train a neural network which is capable of achieving a successful plant model using an off-line backpropagation technique. For a marine application, analysis of the results of this study is included which demonstrates how this technique may be applied, and the nature of the performance obtainable.

1 Introduction

The classical approach to modelling the dynamic behaviour of rigid bodies is to express their behaviour as a set of simultaneous linear and non-linear differential equations, and to obtain a solution for various input simuli. An alternative approach is that of system identification whereby a given input such as a sinusoid or pseudo-random binary sequence (PRBS) is applied to the real system and from a set of input/ output measurements a mathematical model may be obtained. This paper investigates the generation of a state variable representation of a ship in three degrees of freedom by the application of an Artificial Neural Network (ANN).

ANNs have been shown to demonstrate the capability to model highly complex plants. By the application of training data derived from the real environment, these networks can learn to emulate a wide range of differing conditions. Once trained, the neural network substitutes the plant and performs instead.

When considering motion control, the neural network philosophy is of particular interest. Using the non-linear time-invariant dynamic characteristics of a maritime vessel, a neural network is developed to model and control the motion of this process.

Using a carefully selected range of manoeuvres undertaken at various forward speeds, a comparison can be made between the conventional ship model and the neural network model developed.

2 Artificial Neural Networks

Artificial neural networks represent a powerful tool for simulation an understanding of complex relationships between patterns. Pattern can be understood not only as image, but also as number (vector, matrix) of data. The relationship between such vectors is often either not fully known or very difficult to describe using mathematical terms.

The 'genius' of the human brain to understand and to explain situations which are considered fascinating to biologists and engineers. First publications on neural computing was published in the early 1940's by *Frank Rosenblatt, Warren McCulloch, Norbert Wiener, Walter Pitts*. The importance of studies in the field of neuro-medicine is reflected by the number of Nobel prizes awarded to those researching neurology. Between 1901 and 1991, approximately 10% of the

prizes in medicine and physiology were awarded to researchers, whose work contributed directly to the advancement of neurological medicine.

It is the intention of this study to underline the ability of artificial neural networks to handle complex situations in addition to the biological neural network. A neural network has been designed to find (learn) and recall the behaviour of a large motorised marine vessel. It was determined that the initial task was to break down the problem into smaller sub units.

3 Mathematical Background of ANNs

To understand the actions and algorithms concerned with neural computing it is necessary to consider biological neural nets and their architecture.

A *neuron* is the basic element of the brain. A diagram of a neuron is detailed in Figure 1.

The structure of the brain is an interconnection of a very large (tens of billions) number of neurons. The transmission of signals in the brain is chemical in nature. Each neuron receives an input signal from other neighbouring neurons. The connection path between two neurons is called an *axon* and the incoming ports *dendrites*.

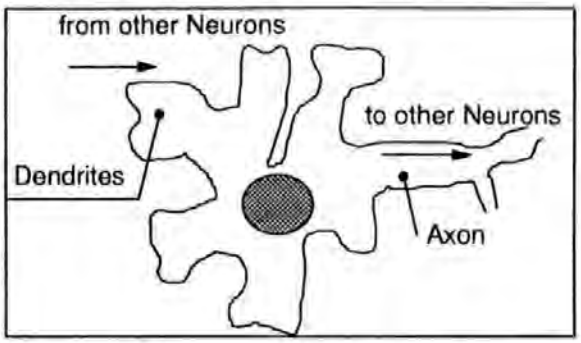


Figure 1 Structure of a biological neuron

The connections between axons and dendrites are called *synapse* (Figure 2). In order to understand the biological model, the axon is an electrical cable and the dendrites is a socket. To carry information a link is required. The synapse, the link or plug, changes the effectiveness of the incoming spike.

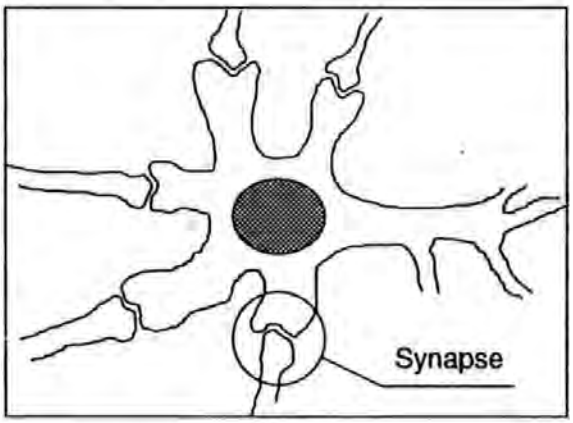


Figure 2 Synapse

During the learning phase the efficiency of the synapse is modified. The sum of the incoming signals, the *total input*, is used by the receiving neuron to generate an output. This output of one neuron is the input for many other neurons except those neurons in the output layer.

The artificial neuron is a simple model of the biological neuron which has the form as displayed in Figure 3.

The label of the signals depends on your view point. Assuming the present neuron, all incoming signals are called x and the output is called y, this y, or output, is then an incoming signal for the next neuron and is then called x. As demonstrated, the synapse is modelled as a modifiable weight which is associated which each axon (connection to a neuron). The neurons output formed by the transfer function is a single number that represents the rate of firing - the activity of the neuron. To compute the output, the neuron multiplies each incoming signal by the associated weight and adds

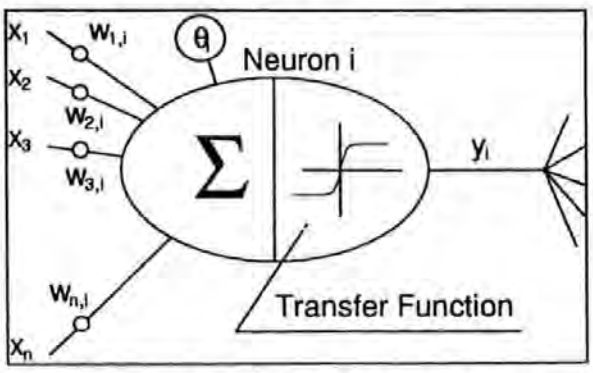


Figure 3 Main structure of an artificial neuron

together all these weighted inputs to form the total input and uses this to create the output by using the transfer function. The reaction of the artificial network depends on both the transfer function used and the weights.

The output of the neuron in the mathematical sense is defined as:

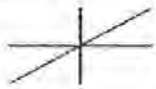
$$I_i^k = \sum_{j=1} x_j^{k-1} \cdot w_{j,i}^k + \theta_i^k \qquad \text{(Equation 1)}$$

$$x_i^l = f(I_i^l) \qquad \text{(Equation 2)}$$

θ^k_i	the threshold, which moves the transfer function (graph) in the horizontal direction.
x^{k-1}_j	output of neuron j in the previous layer
$w^k_{j,i}$	weight between neuron i in layer k and the neuron j in layer k-1
I^k_i	total input of neuron i in layer k

$y_i^k = f(I_i^k)$ where $f(I_i^k)$ (transfer function) could be:

linear $f(I_i^k) = I_i^k$



$f(I_i^k) = \tanh(I_i^k)$

Sigmoid function

$$f(I_i^k) = \frac{1}{1+e^{-I_i^k}}$$

hyperbolic tangent

$$f(I_i^k) = \begin{cases} -1 & I_i^k \le 0 \\ +1 & I_i^k > 0 \end{cases}$$

hard limiter or threshold function

4 Network Architecture

In the past, many forms of neural nets and their algorithms were investigated. Serious investigations started in 1943, by the head neuro-biologist *Warren McColloch* and statistician *Walter Pitt*. The paper [3] tangents fields like digital computing, 'electronic brains' and macroscopic intelligence. The first conference on artificial intelligence was organised in 1956 by famous names such as *Marvin Minsky*, *John McCarthy*, *Claude Shannon* and *Nathanial Rochester*.

To simulate the behaviour of the human brain we need a *network* of neurons, a so called neural network (Net). The neurons are usually organised into groups called layers. A neural net consists of at least an input and an output layer and eventually hidden layer(s). In order to understand the following facts, with 'single' we mean the number of hidden layers. In practise, a single layer net consists of three layers, these being one input and one output layer with a *single* hidden layer. The words *one* and *single* are synonyms for each other. Simple tasks can be solved by a one layer network but for difficult problems a *multi layer network* (Figure 4) is required.

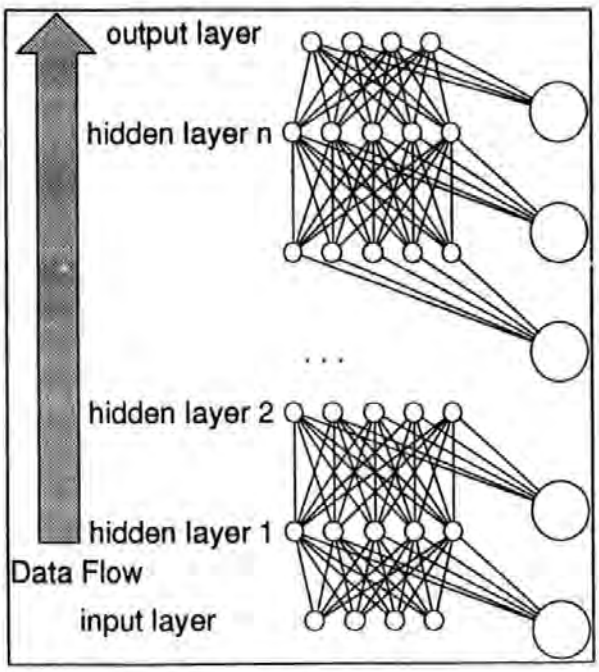


Figure 4 Multi layer network

The behaviour of a multi layer net in general is very similar to a single layer net. The user has to find the optimum network size to be satisfied with the derived results and the speed computation. A small net may be faster but if the task is too difficult then important

information may be lost, conversely, if the net is too large, then the output may become noisy and the subsequent computing speed, especially during the learning, is slow.

Rumelhart's contributions to neural nets ([4]) are fundamentals for further investigation.

The method of supervised learning utilised the *back propagation* algorithm (Figure 5).

The neural net used in this algorithm is a multi layer net and the neurons will use the *Sigmoid* transfer function. The back propagation rule requires the *error* between computed output by the net (straight forward or phase 1) and the desired output given by the 'teacher'. To adjust the weights on the path between one neuron and the next neuron, the error is back propagated, starting with the output layer back to the first layer after the input layer. This process is the second, or learning, phase. The process - computing forward and error propagation backwards - is repeated with different pairs of training data until a maximum number of data is reached or the maximal error approaches an error, i.e. $\varepsilon = 0.05$.

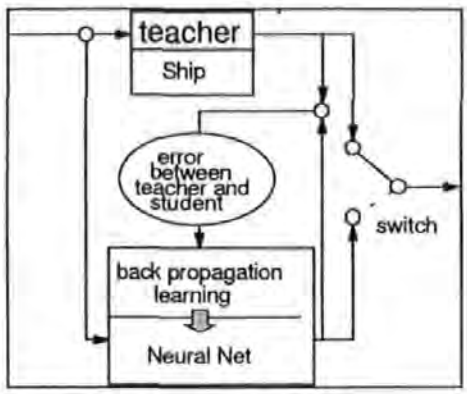


Figure 5 Layout

The interesting feature of back propagation is that we do not need any prior knowledge about the process. However, conversely this may prove to be a significant disadvantage because our student does not have any self organising capabilities and so can not be produce a response that is an improvement on that of the teacher.

$$I_i^l = \sum_j w_{i,i}^l x_j^{l-1} + \theta_i^l \qquad \textit{straight forward} \text{ (phase 1)}$$

$$x_i^l = Sigmoid\left(I_i^l\right)$$

$$\widetilde{w}_{j,i}^l = w_{j,i}^l + \Delta w_{j,i}^l \qquad \text{(Eq. 3)}$$

$$\Delta w_{j,i}^l = \eta \delta_j^l x_j^{l-1} \qquad \textit{back propagation} \text{ (phase 2) (Eq. 4)}$$

$$\delta_j^l = x_i^l\left(1 - x_i^l\right) \cdot \left(d_i - x_i^l\right) \text{ output layer} \qquad \text{(Eq. 5)}$$

$$\delta_j^l = x_i^l\left(1 - x_i^l\right) \cdot \sum_k \delta_k^{l+1} w_{j,k}^{l+1} \text{ inner neurons (Eq. 6)}$$

η	learning coefficient
x_i^l	output of neuron i in layer l
δ_i^l	error of neuron i in layer l
I_i^l	total input of neuron i in layer l
w_{ji}^l	weight on path from neuron j in layer l-1 to neuron i in layer l
Δw_{ji}^l	weight increment for w_{ji}^l
Sigmoid()	transfer function

The learning rules for the thresholds θ are the same as the rules for the weights. The threshold is a weight with the associated input of 1.0 .

5 Ship Mathematical Model

Ship motions in surge, sway and yaw can be described [5] by an Eulerian set of non-linear differential equations of the form :

Surge Equation: (Equation 7)

$$m\dot{u} + mqw - mrv = X_u\dot{u} + X_v\left(u + u_c\right) + X_{uu}u^2 + X_{uuu}u^3 + X_{vv}v^2 + X_{rr}r^2 + X_{\delta\delta}\delta_A^2 + X_{un}un_A + X_{nn}n_A^2 + X_{ua}u_a + X_{zz}z^2 + X_{\theta\theta}\theta^2$$

Sway Equation: (Equation 8)

$$m\dot{v} + mur - mpw = Y_v\dot{v} + Y_v\left(v + v_c\right) + Y_r\dot{r} + Y_r r + Y_{nn}n_A^2 + Y_{vvv}v^3 + Y_{rvv}rv^2 + Y_{nn\delta}n_A^2\delta_A + Y_{nn\delta\delta}n^2\delta_A^3 + Y_{\delta vv}\delta_A v^2 + Y_{va}v_a$$

Yaw Equation: (Equation 9)

$$I_z\dot{r}+(I_y-I_x)pr = N_v\dot{v}+N_v(v+v_c)+N_r\dot{r}+N_{nn}n_A^2+N_{vvv}v^3+N_r r+N_{rvv}rv^2 +N_{nn\delta}n_A^2\delta_A+N_{nn\delta\delta\delta}n_A^2\delta_A^3+N_{\delta vv}\delta_A v^2+N_{va}v_a$$

Equations (7) to (9) can be arranged in the state matrix vector form: (Equation 10)

$$\dot{\mathbf{x}}(t)=\mathbf{F}(t)\mathbf{x}(t)+\mathbf{G}_c(t)\mathbf{u}(t)+\mathbf{G}_D(t)\mathbf{w}(t)$$

The corresponding discrete solution is: (Equation 11)

$$\mathbf{x}((k+1)T)=\mathbf{A}(T,kT)\mathbf{x}(kT)+\mathbf{B}(T,kT)\mathbf{u}(kT)+\mathbf{C}(T,kT)\mathbf{w}(kT)$$

where:

$$\mathbf{x}^T=(\delta_A\ n_A\ x\ u\ y\ v\ z\ w\ \phi\ p\ \theta\ q\ \psi\ r) \quad \text{(Equation 12)}$$

$$\mathbf{u}^T=(\delta_D\ n_D) \quad \text{(Equation 13)}$$

$$\mathbf{w}^T=(u_c\ v_c\ u_a\ v_a\ \zeta_x\ \zeta_y) \quad \text{(Equation 14)}$$

For this study, it was necessary to concentrate on three degrees of freedom. These being surge, sway and yaw.

6 Ship Model Application

The vessels parameters used in this simulation are given below (Table 1), and are based on the Morse and Price data for the *Mariner Hull* [6].

Table 1: Vessel Parameter

Length	=	161m
Draught	=	9m
Beam	=	23m
Displacement	=	17,000t
Number of propellers	=	1
Number of rudders	=	1
Maximum rudder angle	=	35°

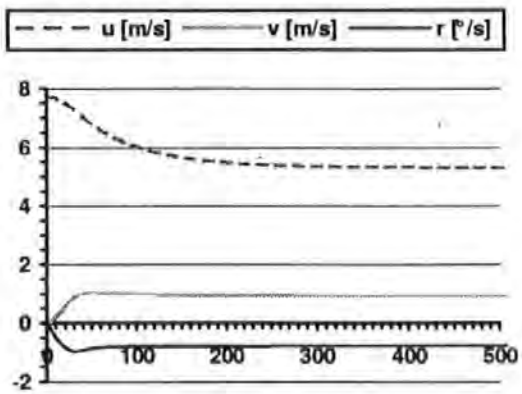


Figure 6 Settling Times

A neural network is required to model the behaviour of large ships. The precise relationships between many of the features and characteristics of these ships are not fully understood. To determine them, it is possible to employ a neural network. Rudder angle, and engine speed cause speed changes in the surge, sway and yaw directions (u, v, $\dot{\psi}$). Since we are not only interested in the steady state response of the vessel, but also in the transient behaviour, it is essential to consider the time elapsed since the last rudder change as an other input. Figure 6 indicates the various time periods required for the response to settle down.

Table 2 settling times

δ [°]	u	t [s]	v	t [s]	yaw rate	t [s]
0	7.588	275	0.278	410	-0.186	405
10	6.639	270	0.734	220	-0.534	180
-10	6.818	270	-0.669	230	0.476	230
20	5.915	295	0.870	240	-0.681	185
-20	6.043	290	-0.836	235	0.641	240
30	5.308	290	0.930	240	-0.782	155
-30	5.408	285	-0.907	240	0.747	165

Utilising an acceptable error of ±1%, we can determine from the data (Table 2) the following values. Therefore it is possible to state that if the time considered is bigger than the time to reach steady state, then the response has reached steady state, otherwise the response remains in the transient period and the operation of the artificial neural network is required.

7 Structure of the ANN

It is a pre-requisite that the variables to be investigated are considered before commencing design of the network's structure (see Table 3).

To learn the transient behaviour, it is necessary to determine the time elapsed since the last rudder change as an additional further input. Thus, the interface to the outside world is defined.

Table 3 Structure of the Network

Inputs		Outputs		
rudder angle	engine speed	forward speed	lateral speed	turning rate
δ		u	v	$\dot{\psi}$

A 3-6-6-6-3 network was identified to be suitable for this application. The quality of results obtained from a two hidden layer network proved unsatisfactory. Obviously, the transients, with their associated overshoots, are difficult to understand, and were therefore filtered out. Using more than two hidden layers the error is reduced and overshoots were replicated giving a suitable level of network performance.

8 Network Training

The learning method utilised for this study was the back propagation algorithm. This algorithm is based on the minimisation technique called steepest descent or gradient method [1]. The transfer function employed was the popular *Sigmoid* function. The output were in the limits between 0.0 and 1.0 ($0.0 < y < 1.0$), where those values are reached at infinity. Therefore, the desired outputs had to be scaled within these limits. During the learning process the trend of the error development was observed and it could be seen that the network stuck in local minima. By increasing the number of hidden layer, the error surface contains less troughs and a more constant learning was achieved. Furthermore the learning rate was adjusted from an initial large learning rate with gradual decrements until the finished level of learning was achieved (steady error).

9 Training Results

Results of the learning are given in Figure 7 to Figure 9. Figure 7 displays the forward speed and demonstrates how the response of the network closely follows that of the surge rate training data. An improved level of performance is identified by the response for sway rate data, and also for that of yaw rate with increments in rudder angle of 0°, -10°, +10°, -20, +20°, -30° and +30° is displayed.

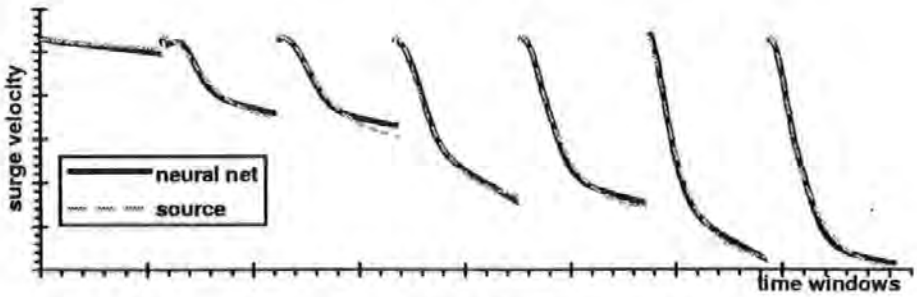


Figure 7 SurgeVelocity Response

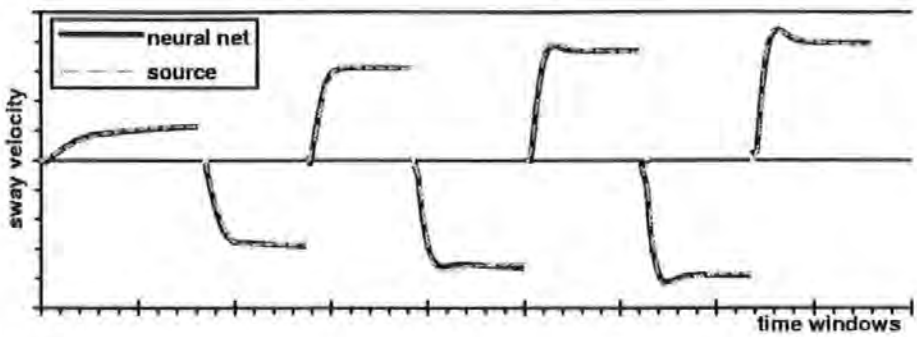


Figure 8 Sway Velocity Response

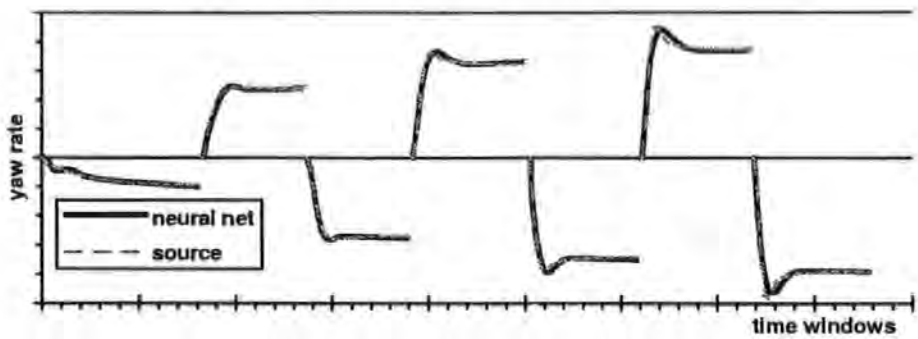


Figure 9 Yaw Rate Response

The actual outputs match very closely the desired outputs given in the training sets which clearly demonstrates the learning success of the network design utilised. Further work by the authors will concentrate on the implementation of this design of network during simulated sea trial conditions. Results will then be compared to those obtained from a traditional ship model to validate both the learning achieved, and the subsequent performance capability obtainable during simulation studies.

10 Conclusions

It has been demonstrated by this study that it is possible to simulate complex plant behaviour utilising neural networks. The advantage of employing a simulation using this technique is that it becomes possible to overcome the problems associated with formulating the relationship between the features to be investigated. This can be achieved by the neural network, thus allowing the designer to concentrate on alternative aspects of the design. The authors consider that the computational speed of the network far exceeds the required time for conventional differential equations because a significant amount of the training is undertaken off-line. During execution, the neural solution therefore allows for extension to far more complex mathematical models without incurring a notable slowing of the process time.

11 References

[1] **Yoh-Han Pao**:
Adaptive Pattern Recognition and Neural Networks, Addison Wesley Publishing Company Inc.

[2] **Ralph Richter, Dr Roland Burns**:
An Artificial Neural Network Autopilot for Small Vessels, proc. of UKSS '93, Keswick, UK.

[3] **Warren McCulloch, Walter Pitts**:
A Logical Calculus of the Ideas Immanent in Nervous Activity, Bulletin of Mathematical Biophysics, vol. 5, pp. 115-133.

[4] **D. Rumelhart, J. McClelland**:
Parallel Distributing Processing, MIT Press, Cambridge, Massachusetts 1986.

[5] **Burns, R. S.**:
Application of the Riccati Equation in Control, Systems and Signals, International Federation of Automatic Control, Como, Italy, 26-28th June 1989.

[6] **Morse, R. V.; Price, D.**:
Manoeuvring Characteristics of th Mariner Type Ship (USS Compas Island) in Calm Seas, Sperry Polaris Management, Sperry Gyroscope Company, New York, 1961.

R. Richter and R. S. Burns.

An artificial neural network autopilot for small vessels.

In R. Poley and R. Zobel, editors, *Proc. of the First Conference of the United Kingdom Simulation Society 1993*, pages 168–172, Keswick Hotel, Keswick, Cumbria, UK, 13–15 September 1993. ISBN 0-9516509-1-2.

AN ARTIFICAL NEURAL NETWORK AUTOPILOT FOR SMALL VESSELS

Ralph Richter

Institut für Produktionstechnik,
Lehrstuhl Produktionsautomatisierung/ Steuerungstechnik
Technical University of DRESDEN,

Dresden, Germany

Roland S Burns

School of Manufacturing, Materials & Mechanical Engineering,
University of Plymouth, Devon, U.K.

Abstract

Conventional autopilots for small crafts are based on proportional plus integral plus derivative (PID) control algorithms. The settings of the proportional, integral and derivative control parameters depend upon the vessel's handling characteristics and also the environmental conditions, e.g. the sea state.

Ideally the autopilot should be tuned in calm water conditions and then retuned for different sea states. In practice the controller parameters are pre-set at the factory and are rarely changed, and so most autopilots do not operate at their optimal settings.

This paper investigates the use of artificial neural networks (nets) as an alternative control strategy. A neural network supervised by PID controllers has been developed. The input vector to the network consists of heading error, rate of change of heading error and the integral of heading error. The output of the network is the requested rudder angle. The learning data for the neural network was generated using tuned PID information for a range of sea states.

The network employs the back propagation approach. The input to each artificial neuron consists of the sum of all inputs (multiplied by the weight) which is then used for the generation of a single output by using the Sigmoid as a transfer function.

The investigation has demonstrated that if the size of the network is too small, then important learning characteristics will be lost. On the other hand if the network is too large then the output may be noisy and the computational speeds, especially during the learning phase, will be slow.

The paper will display the initial results of the study. The work to-date has shown that a tuned PID controller for a given sea state will out-perform a general purpose neural controller designed to cope with a range of sea states. However, results indicate that when the PID controller operates in sea states that it has not been specifically tuned for, the neural controller provides superior performance.

1. Fundamental Principles

The brain is the most complex structure we know. Its powerful capabilities, like thinking, remembering, problem-solving and learning, are very fascinating to model. We use artificial neural nets to simulate the behaviour of the human brain such as learning and recall of patterns. First applications were developed for pattern recognition in the early 1940's. The first principles were published by Frank Rosenblatt in 1957. He developed an element called *perceptron*, as shown in figure 1.1, which attracted attention in the world of neural computing. His perceptron is a device to recognise abstract and geometric patterns.

The perceptron consists of a 400 photocell grid and was mainly developed for optical pattern recognition. The electrical output of the photocells were collected by the associator unit passing the random connections. The new multi layer system, developed in the 1960's, could learn and recall complex tasks. A non linear transfer function was used.

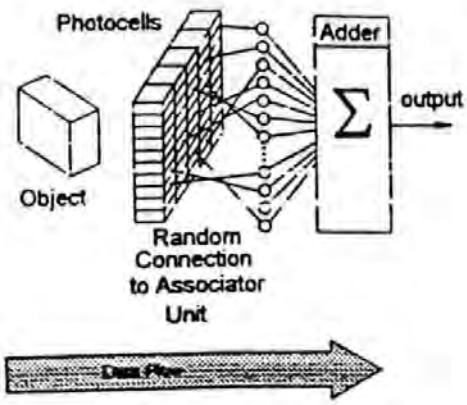


Fig. 1.1 Principle of the Perceptron

To understand actions and algorithms in neural computing it is necessary to look at biological neural nets and their architecture. A *neuron* is the basic element of the brain. A diagram of a neuron is detailed in figure 1.2.

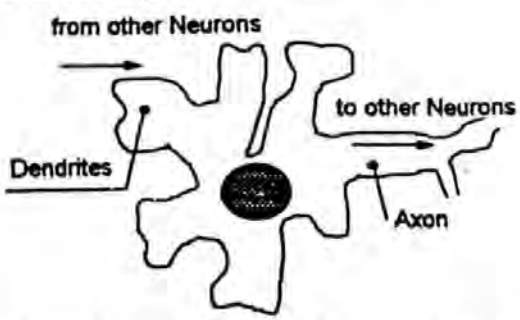


Fig. 1.2 Main Structure of a Biological Neuron

The structure of the brain is an interconnection of a very large (tens of billions) number of neurons. The transmission of signals in the brain is chemical in nature. Each neuron receives an input signal from other neighbouring neurons. The connection path between two neurons is called an *axon* and the incoming ports *dendrites*. The connections between axons and dendrites are called *synapses*. In order to understand the biological model, the axon is an electrical cable and the dendrites is a socket. To carry information a link is needed. The synapse, the link or plug, changes the effectiveness of the incoming spike.

During a learning phase the efficiency of the synapse is modified. The sum of the incoming signals, the *total input*, is used by the receiving neuron to generate an output. This output of one neuron is the input for many other neurons except those neurons in the output layer. The artificial neuron is a simple model of the biological neuron which has the form as displayed in figure 1.3.

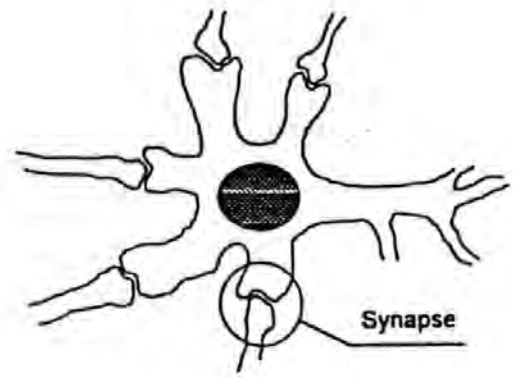


Fig. 1.3 Synapse

The denotation of the signals depends on your point of view. Assuming the present neuron, all incoming signals are called x and the output is called y, this y, or output, is then an incoming signal for the next neuron and is then called x (figure 1.4).

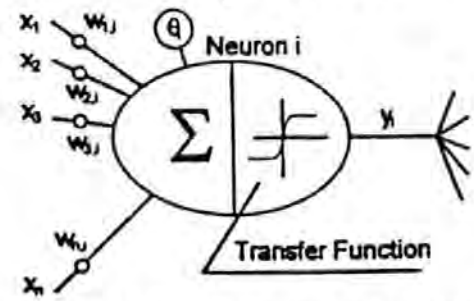


Fig. 1.4 Main Structure of an Artificial Neuron

As you can see, the synapse is modelled as a modifiable weight which is associated which each axon (connection to a neuron). The neurons output formed by the transfer function is a single number that represents the rate of firing - the activity of the neuron. To compute the output, the neuron multiplies each incoming signal by the associated weight and adds together all these weighted inputs to form the total input and uses this to create the output by using the transfer function. The reaction of the artificial network depends on both the transfer function used and the weights.

The output of the neuron in the mathematical sense is defined as:

$$I_i^k = \sum_{j=1} x_j^{k-1} \cdot w_{j,i}^k + \theta_i^k$$

θ^k_i the threshold, which moves the transfer function (graph) in the horizontal direction.

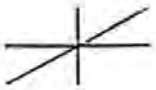
x^{k-1}_j output of neuron j in the previous layer

$w^k_{j,i}$ weight between neuron i in layer k and the neuron j in layer k-1

I^k_i total input of neuron i in layer k

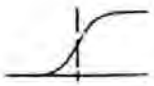
$y_i^k = f(I_i^k)$ where $f(I_i^k)$ (transfer function) could be:

linear $f(I_i^k) = I_i^k$



Sigmoid function

$$f(I_i^k) = \frac{1}{1+e^{-I_i^k}}$$



hyperbolic tangent

$$f(I_i^k) = \tanh(I_i^k)$$

$$f(I_i^k) = \begin{cases} -1 & I_i^k \le 0 \\ +1 & I_i^k > 0 \end{cases}$$ hard limiter

or threshold function

2. Types of Networks

2.1. Architecture of Single Layer Nets

In the past, many forms of neural nets and their algorithms were investigated. Serious investigations started in 1943, by the head neurobiologist Warren McColloch and statistician Walter Pitt. The paper "A Logical Calculus of Ideas Imminent in Nervous Activity" tangents fields like digital computing, "electronic brains" and macroscopic intelligence. The first conference on artificial intelligence was organised in 1956 by famous names such as Marvin Minsky, John McCarthy, Claude Shannon and Nathanial Rochester.

To simulate the behaviour of the human brain we need a *network* of neurons, a so called neural network (Net). The neurons are usually organised into groups called layers. A neural net consists of at least an input and an output layer and eventually hidden layer(s). In order to understand the following facts, with 'single' we mean the number of hidden layers. Actually, a single layer net consists of three layers, one input and one output layer and a *single* hidden layer. The words *one* and *single* are synonyms for each other. Simple tasks can be solved by a one layer network but for difficult problems we need *multi layer nets*. The main structure of a single layer net is shown below.

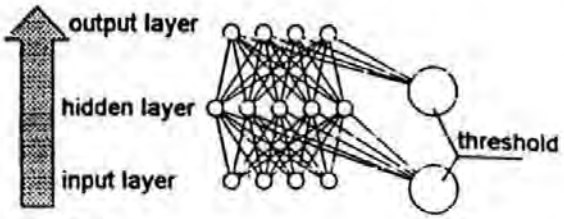


Fig. 2.1 Main Structure of a Single Layer Network

The interconnection between the neurons in different layers can be seen in figure 2.1. It is not necessary to have a net where the connections are only between neurons of different layers, but it is easier to understand and to design a net in this way. The majority of modern neural nets are organised in this way. Some tasks do not require hidden layers. The number of hidden layers and the number of neurons in each hidden layer is free to be defined and will determine the performance of the net in speed and quality. For the majority of tasks a single layer net is sufficient.

2.2. Multi Layer Nets

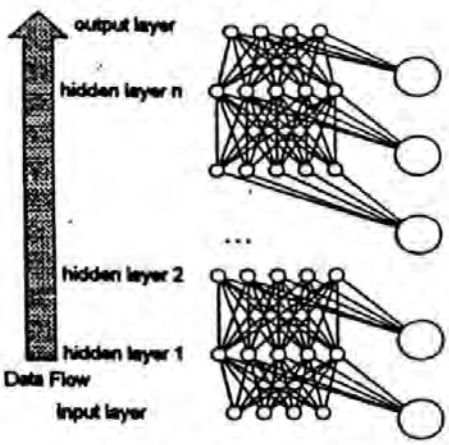


Fig. 2.2 Main Structure of a Multi Layer Netwo

The behaviour of a multi layer net in general is no very different to a single layer net. The user has t find the optimum in size to be satisfied with th results and the speed. A small net is faster but if th task is too difficult important information may b lost, conversely, if the net is too large, the output ca be noisy and the computing speed, especially durin the learning, is slow.

3. Learning of a Neural Net

3.1. General Facts

The two main tasks of a brain - learning and recall - are the most interesting for us. Learning itself is the process of calibration of the synaptic efficiency, or in the words of artificial nets, the weights. Using this principle some models of neurons and their connections have been investigated. i.e. single layer nets, multi-layer nets and self-organising nets. The nets can be classified into three groups, depending upon the learning principle, e.g. *supervised learning* (as discussed in this paper), *learning with critic* and one group *unsupervised learning* (*self-organising nets*). The latter is utilised to obtain relationships between the input and the output vector by the creation of an iterative process without a teacher (as in supervised learning) and also without evaluative values (as learning with critic). If we inspect supervised learning, we must consider, that the results of the student (our net) can be only as good as the training data of the teacher/ supervisor. For supervised learning we need a vector of input data and one vector of the desired outputs which is associated to the input vector. The reader can easily see, that one problem, besides the program for learning, is to have good sets of training data. We interpret a set of training data as a pair of input/ desired output vectors.

3.2. Back Propagation

Rumelhart's contributions to neural nets (1986) are fundamentals for further investigation. In this paper, the method of supervised learning will be discussed and how to use this in order to develop a learning controller for steering of small crafts.

One way to utilise the supervised learning is by using the *back propagation* algorithm. The control model is displayed in figure 3.1.

The neural net used in this algorithm is a multi-layer net and the transfer function is the *Sigmoid*. The back propagation rule needs the *error* between computed output by the net (straight forward or phase 1) and the desired output given by the teacher. To adjust the weights on the path between one neuron and the next neuron, the error is back propagated, starting with the output layer back to the first layer after the input layer. This process is the second or the learning phase.

The process - computing forward and error propagation backwards - is repeated with different pairs of training data until the end of the data is reached or the maximal error approaches its limiting value, i.e. $\varepsilon = 0.05$.

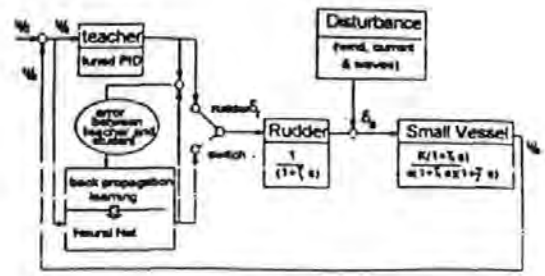


Fig. 3.1 Control Model

Ψ_a Ψ_d ..... actual/ desired heading
δ_a δ_r .....actual/ requested rudder
Ψ_e .....error heading
T_1, T_2, T_3, T_r .....time constants

The interesting feature of back propagation is, that we do not need any knowledge about the process, this is what we will use the controller for, but we need a good teacher. However, this is on the other hand a disadvantage because our student does not have any self-organising capabilities and so this can not be a better response than the teacher. The conclusion is displayed below.

$$x_i^l = Sigmoid\left(I_i^l\right) \quad \textit{straight forward} \text{ (ph. 1)(A1.1)}$$

$$I_i^l = \sum_j w_{i,j}^l x_j^{l-1} + \theta_i^l \qquad \text{(A1.2)}$$

$$\tilde{w}_{j,i}^l = w_{j,i}^l + \Delta w_{j,i}^l \qquad \text{(A2.1)}$$

$$\Delta w_{j,i}^l = \eta \delta_j^l x_j^{l-1} \qquad \textit{back propagation} \text{ (phase 2)} \quad \text{(A2.2)}$$

$$\delta_j^l = x_i^l\left(1 - x_i^l\right)\cdot\left(d_i - x_i^l\right) \text{ output layer} \qquad \text{(A2.3)}$$

$$\delta_j^l = x_i^l\left(1 - x_i^l\right)\cdot\sum_k \delta_k^{l+1} w_{j,k}^{l+1} \text{ inner neurons} \qquad \text{(A2.4)}$$

η learning coefficient
x^l_i output of neuron i in layer l
δ^l_i error of neuron i in layer l
I^l_i total input of neuron i in layer l
w^l_{ji} weight on path from neuron j in layer l-1 to neuron i in layer l
Δw^l_{ji} weight increment for w^l_{ji}
Sigmoid() transfer function

Note, the learning rules for the thresholds θ are the same as the rules for the weights. The threshold is a weight with the associated input of 1.0 .

4. Application of Neural Nets for Ship Steering

4.1. Review

Since the increase in the number of computers, more and more modern techniques have been used such as neural computing (neural nets), fuzzy logic, etc.

Conventional autopilots, based on PD or PID controllers are often used but they are developed to work under specific conditions and so they are not working at their optimal point. Alternatively the mariner was engaged by adjusting the working parameters of the PID autopilot. The main idea of a modern controller is to merge all the beneficial features of several controllers to create an intelligent controller, i.e. with a behaviour like a human helmsman.

4.2. Creation of the Training Data

In the previous chapters, a teacher for the neural net was mentioned. The idea that this study is based on is, that one PID controller is tuned for one particular sea state and this *tuned PID controller* is used as one teacher for the neural net. If a training file, which contains input and output data of the PID controller, is created, the neural net will learn to respond like its teacher. But if the training data file consists of data pairs of more than one teacher, i. e. data of several tuned PID controllers in several sea states used in their associated sea states, the neural net will learn the behaviour of the tuned PID controllers at its optimal point or close to it. We know that the ship parameters such as weight, inertia, draught and speed, have key effects in the behaviour of the ship. So, if we want, we could tune PID controllers for more specific situations and create more relevant data.

The PID controllers, used as teachers, are tuned firstly for a very small heading error and not for a smooth rudder movement. Tests have proved that the neural net will work as a damper too.

4.3. Testing the Autopilot

The net consists of 10 neurons in each of the 2 chosen hidden layers. Figure 4.1 shows a typical yaw response of the trained network, compared with a standard PID autopilot in sea state 4. Table 4.1 gives the RMS yaw error for both the PID and neural autopilots for sea states 3, 4 and 5.

Sea State	3	4	5
RMS Yaw Error PID Autopilot	0.193	0.579	2.387
RMS Yaw Error Neural Autopilot	0.103	0.197	1.863

Table 4.1
Comparison between PID and Neural Autopilots

Fig. 4.1 PID - Neural Response Sea State 4

5. Conclusions

The applicational benefits of the neural network could be expanded upon by:

- applications to different speeds,
- to different mass loadings,
- to different vessels.

Obviously the problem would then arise that the required data is hard to obtain since as additional alterations are introduced to the control problem, extra inputs will be required by the net to register these changes.

Current research into a general computer model for small vessels could prove an essential source of data for such a supervised learning network.

Alternatively modifications to the control action could be achieved by an unsupervised learning network operating in an on-line manner.

Given these points, the scope and potential for further research and development is huge. However it has successfully been proven that a neural network has the ability to actively control a small vessel in a superior fashion to a PID controller, and this may be regarded as a "milestone" in the application of neural techniques in this field.

References

Endo M, van Amerongen J, and Bakkers A W P "Application of Neural Networks to Ship Steering".

Neural Works Professional II
Neural Ware, Inc. 1991 (Computer package)

Yoh-Han Pao, Case Western Reserve University "Adaptive Pattern Recognition and Neural Networks"
Addison-Wesley Publishing Company, Inc.

Application of an Artifical Neural Network to Model Complex Plant Behaviour

presented by *Ralph RICHTER*

at the

IFAC'95 - Workshop on Motion Control

e-mail: ralphr@soc.plym.ac.uk
www: http://frog.cis.plym.ac.uk

R. Richter, R. S. Burns, and M. N. Polkinghorne.

Application of an artificial neural network to model complex plant behaviour.

In *IFAC '95 - Workshop on Motion Control, Munich - Germany*, 9–11 October 1995.

Application of an Artifical Neural Network to Model Complex Plant Behaviour

R Richter[1], R S Burns[1,2]
M N Polkinghorne[2]
[1)] School of Manufacturing, Materials and Mechanical Engineering
[2)] Plymouth Teaching Company Centre,
University of Plymouth, Devon, UNITED KINGDOM

Abstract

Since the expansion in the number of powerful computers and workstations available, simulations of complex structures (plants) are increasingly part of the design process.

By use of these complex simulations, off-line study of plants and/ or controller designs may be achieved when otherwise no realistic study could be undertaken. Differential equations are the main parts of those simulations. To increase the precision of the simulation results, more time consuming calculations are necessary but not always available.

In particular, neural networks demonstrate the capability to model highly complex plants. By the application of training data derived from real environment, these networks can learn to emulate a wide range of differing conditions. Once trained, the neural network substitutes the plant's model and performs instead.

When considering motions control, the neural network philosophy is of particular interest. Using the non-linear time-invariant dynamic characteristics of a maritime vessel, a neural network is developed to model and control the motion of this process. A comparative study is undertaken to validate the network operation.

1 Introduction

The classical approach to modelling the dynamic behaviour of rigid bodies is to express their behaviour as a set of simultaneous linear and non-linear differential equations, and to obtain a solution for various input stimuli. An alternative approach is that of system identification whereby a given input such as a sinusoid or pseudo-random binary sequence (PRBS) is applied to the real system and from a set of input/ output measurements a mathematical model may be obtained. This paper investigates the generation of a state variable representation of a ship in three degrees of freedom by the application of an Artificial Neural Network (ANN).

ANNs have been shown to demonstrate the capability to model highly complex plants. By the application of training data derived from the real environment, these networks can learn to emulate a wide range of differing conditions. Once trained, the neural network substitutes the plant and performs instead.

When considering motion control, the neural network philosophy is of particular interest. Using the non-linear time-invariant dynamic characteristics of a maritime vessel, a neural network is developed to model and control the motion of this process.

Using a carefully selected range of manoeuvres undertaken at various forward speeds, a comparison can be made between the conventional ship model and the neural network model developed.

2 Artificial Neural Networks

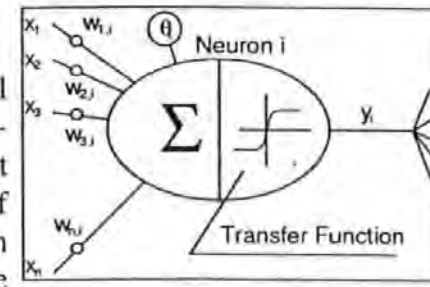


Figure 3 Main structure of an artificial neuron

Artificial neural networks represent a powerful tool for simulation and understanding of complex relationships between patterns. Pattern can be understood not only as image, but also as number (vector, matrix) of data. The relationship between such vectors is often either not fully known or very difficult to describe using mathematical terms.

The 'genius' of the human brain is to understand and to explain situations which are considered fascinating to biologists and engineers. First publications on neural computing was published in the early 1940's by *Frank Rosenblatt, Warren McCulloch, Norbert Wiener, Walter Pitts*. The importance of studies in the field of neuro-medicine is reflected by the number of Nobel prizes awarded to those researching neurology. Between 1901 and 1991, approximately 10% of the prizes in medicine and physiology were awarded to researchers, whose work contributed directly to the advancement of neurological medicine.

It is the intention of this study to underline the ability of artificial neural networks to handle complex situations in addition to the biological neural network. A neural network has been designed to find (learn) and recall the behaviour of a large motorised marine vessel. It was determined that the initial task was to break down the problem into smaller sub units.

3 Mathematical Background of ANNs

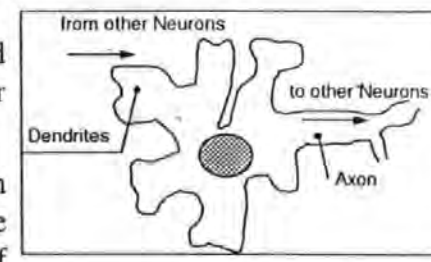


Figure 1 Structure of a biological neuron

To understand the actions and algorithms concerned with neural computing it is necessary to consider biological neural nets and their architecture.

A *neuron* is the basic element of the brain. A diagram of a neuron is detailed in Figure 1. The structure of the brain is an interconnection of a very large (tens of billions) number of neurons. The transmission of signals in the brain is chemical in nature. Each neuron receives an input signal from other neighbouring neurons. The connection path between two neurons is called an *axon* and the incoming ports *dendrites*.

The connections between axons and dendrites are called *synapse* (Figure 2). In order to understand the biological model, the axon is an electrical cable and the dendrites is a socket. To carry information a link is required. The synapse, the link or plug, changes the effectiveness of the incoming spike.

During the learning phase the efficiency of the synapse is modified. The sum of the incoming signals, the *total input*, is used by the receiving neuron to generate an output. This output of one neuron is the input for many other neurons except those neurons in the output layer.

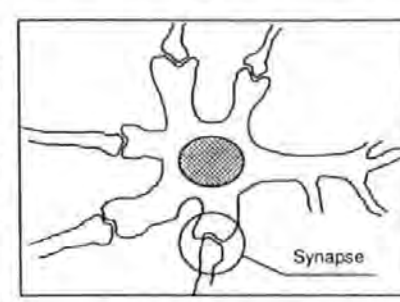


Figure 2 Synapse

The artificial neuron is a simple model of the biological neuron which has the form as displayed in Figure 3.

The label of the signals depends on your view point. Assuming the present neuron, all incoming signals are called x and the output is called y, this y, or output, is then an incoming signal for the next neuron and is then called x. As demonstrated, the synapse is modelled as a modifiable weight which is associated which each axon (connection to a neuron). The neurons output formed by the transfer function is a single number that represents the rate of firing - the activity of the neuron. To compute the output, the neuron multiplies each incoming signal by the associated weight and adds together all these weighted inputs to form the total input and uses this to create the output by using the transfer function. The reaction of the artificial network depends on both the transfer function used and the weights.

The output of the neuron in the mathematical sense is defined as:

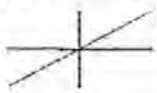
$$I_i^k = \sum_{j=1} x_j^{k-1} \cdot w_{j,i}^k + \theta_i^k \qquad \text{(Equation 1)}$$

$$x_i^l = f\left(I_i^l\right) \qquad \text{(Equation 2)}$$

θ^k_i	the threshold, which moves the transfer function (graph) in the horizontal direction.
x^{k-1}_j	output of neuron j in the previous layer
$w^k_{j,i}$	weight between neuron i in layer k and the neuron j in layer k-1
I^k_i	total input of neuron i in layer k

$y_i^k = f\left(I_i^k\right)$ where $f\left(I_i^k\right)$ (transfer function) could be:

linear $f\left(I_i^k\right) = I_i^k$



Sigmoid function

$$f\left(I_i^k\right) = \frac{1}{1+e^{-I_i^k}}$$

hyperbolic tangent

$$f\left(I_i^k\right) = \tanh\left(I_i^k\right)$$

$$f\left(I_i^k\right) = \begin{cases} -1 & I_i^k \le 0 \\ +1 & I_i^k > 0 \end{cases}$$

hard limiter or threshold function

4 Network Architecture

In the past, many forms of neural nets and their algorithms were investigated. Serious investigations started in 1943, by the head neuro-biologist *Warren McColloch* and statistician *Walter Pitt*. The paper [4] tangents fields like digital computing, 'electronic brains' and macroscopic intelligence. The first conference on artificial intelligence was organised in 1956 by famous names such as *Marvin Minsky*, *John McCarthy*, *Claude Shannon* and *Nathanial Rochester*.

To simulate the behaviour of the human brain we need a *network* of neurons, a so called neural network (Net). The neurons are usually organised into groups called layers. A neural net consists of at least an input and an output layer and eventually hidden layer(s). In order to understand the following facts, with 'single' we mean the number of hidden layers. In practice, a single layer net consists of three layers, these being one input and one output layer with a *single* hidden layer. The words *one* and *single* are synonyms for each other. Simple tasks can be solved by a one layer network but for difficult problems a *multi layer network* (Figure 4) is required.

The behaviour of a multi layer net in general is very similar to a single layer net. The user has to find the optimum network size to be satisfied with the derived results and the speed computation. A small net may be faster but if the task is too difficult then important information may be lost, conversely, if the net is too large, then the output may become noisy and the subsequent computing speed, especially during the learning, is slow.

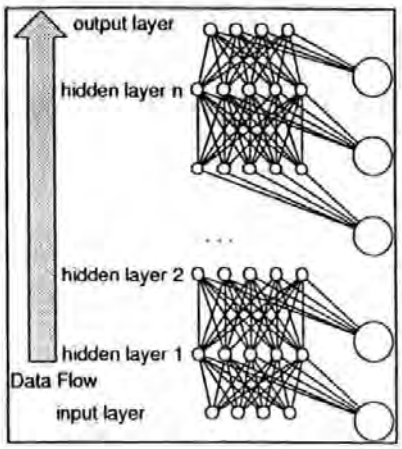


Figure 4 Multi layer network

Rumelhart's contributions to neural nets ([5]) are fundamentals for further investigation.

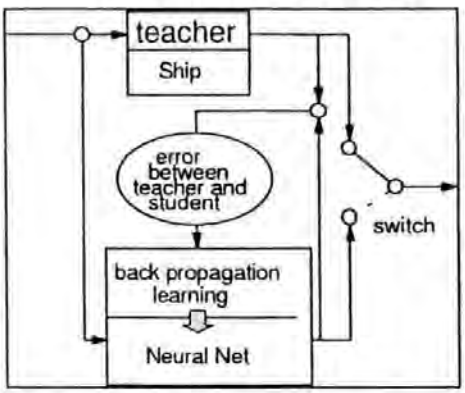


Figure 5 Layout

The method of supervised learning utilised the *back propagation* algorithm (Figure 5).

The neural net used in this algorithm is a multi layer net and will be the *Sigmoid* transfer function. The back propagation rule requires the *error* between computed output by the net (straight forward or phase 1) and the desired output given by the 'teacher'. To adjust the weights on the path between one neuron and the next neuron, the error is back propagated, starting with the output layer back to the first layer after the input layer. This process is the second, or learning, phase. The process - computing forward and error propagation backwards - is repeated with different pairs of training data until a maximum number of data is reached or the maximal error approaches an error, i.e. $\varepsilon = 0.05$.

The interesting feature of back propagation is that we do not need any prior knowledge about the process. However, conversely this may prove to be a significant disadvantage because our student does not have any self organising capabilities and so cannot produce a response that is an improvement on that of the teacher.

$$I_i^l = \sum_j w_{i,j}^l x_j^{l-1} + \theta_i^l \qquad \textit{straight forward}$$

(phase 1)

$$x_i^l = Sigmoid\left(I_i^l\right)$$

$$\tilde{w}_{j,i}^l = w_{j,i}^l + \Delta w_{j,i}^l \qquad \text{(Eq. 3)}$$

$$\Delta w_{j,i}^l = \eta \delta_j^l x_j^{l-1} \qquad \textit{back propagation}$$

(phase 2) (Eq. 4)

$$\delta_j^l = x_i^l\left(1 - x_i^l\right)\cdot\left(d_i - x_i^l\right)$$

for output layer (Eq. 5)

$$\delta_j^l = x_i^l\left(1 - x_i^l\right)\cdot\sum_k \delta_k^{l+1} w_{j,k}^{l+1}$$

for inner neurons (Eq. 6)

η	learning coefficient
x^l_i	output of neuron i in layer l
δ^l_i	error of neuron i in layer l
I^l_i	total input of neuron i in layer l
w^l_{ij}	weight on path from neuron j in layer l-1 to neuron i in layer l
Δw^l_{ij}	weight increment for w^l_{ij}
Sigmoid()	transfer function

The learning rules for the thresholds θ are the same as the rules for the weights. The threshold is a weight with the associated input of 1.0 .

5 Ship Mathematical Model

Ship motions in surge, sway and yaw can be described [6] by an Eulerian set of non-linear differential equations of the form :

Surge Equation: (Equation 7)

$$m\dot{u} + mqw - mrv = X_{\dot{u}}\dot{u} + X_u(u + u_c) + X_{uu}u^2 + X_{uuu}u^3 + X_{vv}v^2 + X_{rr}r^2 + X_{\delta\delta}\delta_A^2 + X_{un}un_A + X_{nn}n_A^2 + X_{ua}u_a + X_{zz}z^2 + X_{\theta\theta}\theta^2$$

Sway Equation: (Equation 8)

$$m\dot{v} + mur - mpw = Y_{\dot{v}}\dot{v} + Y_v(v + v_c) + Y_{\dot{r}}\dot{r} + Y_r r + Y_{nn}n_A^2 + Y_{vvv}v^3 + Y_{rvv}rv^2 + Y_{nn\delta}n_A^2\delta_A + Y_{nn\delta\delta\delta}n^2\delta_A^3 + Y_{\delta vv}\delta_A v^2 + Y_{va}v_a$$

Yaw Equation: (Equation 9)

$$I_z\dot{r} + (I_y - I_x)pr = N_{\dot{v}}\dot{v} + N_v(v + v_c) + N_{\dot{r}}\dot{r} + N_{nn}n_A^2 + N_{vvv}v^3 + N_r r + N_{rvv}rv^2 + N_{nn\delta}n_A^2\delta_A + N_{nn\delta\delta\delta}n_A^2\delta_A^3 + N_{\delta vv}\delta_A v^2 + N_{va}v_a$$

Equations (7) to (9) can be arranged in the state matrix vector form: (Equation 10)

$$\dot{\mathbf{x}}(t) = \mathbf{F}(t)\mathbf{x}(t) + \mathbf{G}_c(t)\mathbf{u}(t) + \mathbf{G}_D(t)\mathbf{w}(t)$$

The corresponding discrete solution is: (Equation 11)

$$\mathbf{x}((k+1)T) = \mathbf{A}(T,kT)\mathbf{x}(kT) + \mathbf{B}(T,kT)\mathbf{u}(kT) + \mathbf{C}(T,kT)\mathbf{w}(kT)$$

where:

$$\mathbf{x}^T = (\delta_A\ n_A\ x\ u\ y\ v\ z\ w\ \phi\ p\ \theta\ q\ \psi\ r) \quad \text{(Equation 12)}$$

$$\mathbf{u}^T = (\delta_D\ n_D) \quad \text{(Equation 13)}$$

$$\mathbf{w}^T = (u_c\ v_c\ u_a\ v_a\ \zeta_x\ \zeta_y) \quad \text{(Equation 14)}$$

For this study, it was necessary to concentrate on three degrees of freedom. These being surge, sway and yaw.

6 Ship Model Application

The vessels parameters used in this simulation are given below (Table 1), and are based on the Morse and Price data for the *Mariner Hull* [7].

Table 1: Vessel Parameter

Length	=	161m
Draught	=	9m
Beam	=	23m
Displacement	=	17,000t
Number of propellers	=	1
Number of rudders	=	1
Maximum rudder angle	=	35°

A neural network is required to model the behaviour of large ships. The precise relationships between many of the features and characteristics of these ships are not fully understood. To determine them, it is possible to employ a neural network. Rudder angle, and engine speed cause speed changes in the surge, sway and yaw directions (u, v, r). Since we are not only interested in the steady state response of

the vessel, but also in the transient behaviour, it is essential to consider the time elapsed since the last rudder change as another input. Figure 6 indicates the various time periods required for the response to settle down.

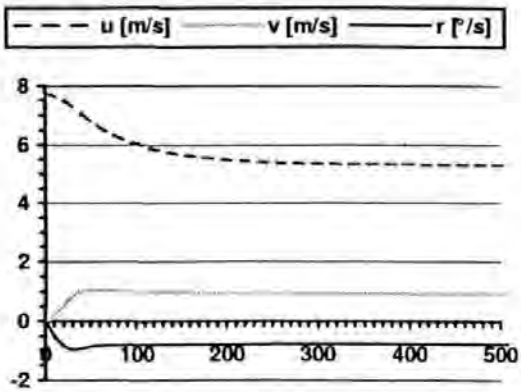


Figure 6 Settling Times

Utilising an acceptable error of ±1%, we can determine from the data (Table 2) the following values. Therefore it is possible to state that if the time considered is bigger than the time to reach steady state, then the response has reached steady state, otherwise the response remains in the transient period and the operation of the artificial neural network is required.

7 Structure of the ANN

It is a pre-requisite that the variables to be investigated are considered before commencing design of the network's structure (see Table 3).

Table 3 Structure of the Network

Inputs		Outputs		
rudder angle	engine speed	forward speed	lateral speed	turning rate
δ	n	u	v	r

To learn the transient behaviour, it is necessary to determine the time elapsed since the last rudder change as an additional further input. Thus, the interface to the outside world is defined.

A 3-6-6-6-3 network was identified to be suitable for this application. The quality of results obtained from a two hidden layer network proved unsatisfactory. Obviously, the transients, with their associated overshoots, are difficult to understand, and were therefore filtered out. Using more than two hidden layers the error is reduced and overshoots were replicated giving a suitable level of network performance.

8 Network Training

The learning method utilised for this study was the back propagation algorithm. This algorithm is based on the minimisation technique called steepest descent or gradient method [2]. The transfer function employed was the popular *Sigmoid* function. The

Table 2 settling times

δ [°]	u	t [s]	v	t [s]	yaw rate	t [s]
0	7.588	275	0.278	410	-0.186	405
10	6.639	270	0.734	220	-0.534	180
-10	6.818	270	-0.669	230	0.476	230
20	5.915	295	0.870	240	-0.681	185
-20	6.043	290	-0.836	235	0.641	240
30	5.308	290	0.930	240	-0.782	155
-30	5.408	285	-0.907	240	0.747	165

output were in the limits between 0.0 and 1.0 ($0.0 < y < 1.0$), where those values are reached at infinity. Therefore, the desired outputs had to be scaled within these limits. During the learning process the trend of the error development was observed and it could be seen that the network stuck in local minima. By increasing the number of hidden layers, the error surface contains less troughs and a more constant learning was achieved. Furthermore the learning rate was adjusted from an initial large learning rate with gradual decrements until the finished level of learning was achieved (steady error).

9 Training Results

Results of the learning are given in Figure 7 to Figure 9. Figure 7 displays the forward speed and demonstrates how the response of the network closely follows that of the surge rate training data. An improved level of performance is identified by the response for sway rate data, and also for that of yaw rate with increments in rudder angle of 0°, -10°, +10°, -20°, +20°, -30° and +30° is displayed.

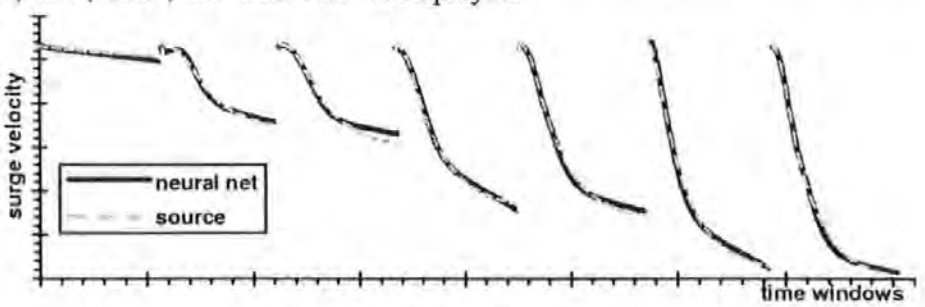


Figure 7 SurgeVelocity Response

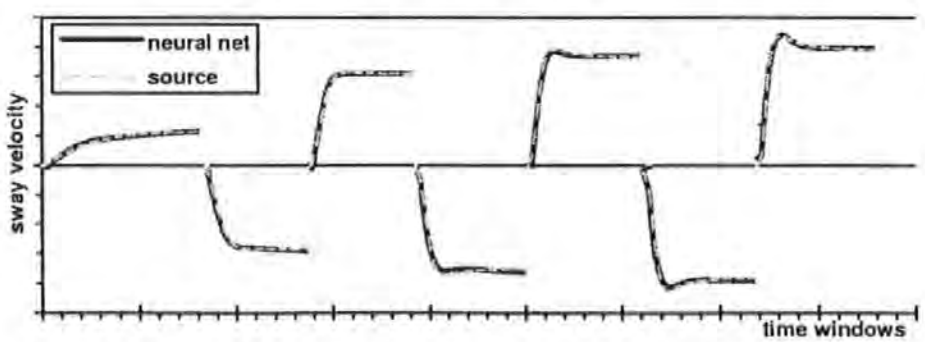


Figure 8 Sway Velocity Response

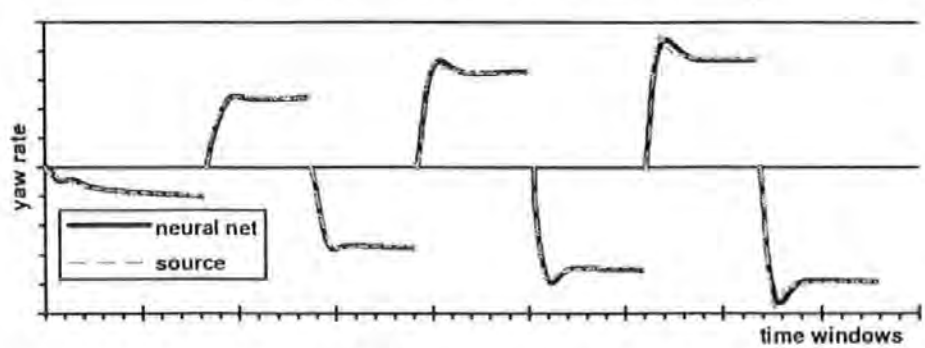


Figure 9 Yaw Rate Response

The actual outputs match very closely the desired outputs given in the training sets which clearly demonstrates the learning success of the network design utilised. Further work by

the authors will concentrate on the implementation of this design of network during simulated sea trial conditions. Results will then be compared to those obtained from a traditional ship model to validate both the learning achieved, and the subsequent performance capability obtainable during simulation studies.

10 Conclusions

It has been demonstrated by this study that it is possible to simulate complex plant behaviour utilising neural networks. The advantage of employing a simulation using this technique is that it becomes possible to overcome the problems associated with formulating the relationship between the features to be investigated. This can be achieved by the neural network, thus allowing the designer to concentrate on alternative aspects of the design. The authors consider that the computational speed of the network far exceeds the required time for conventional differential equations because a significant amount of the training is undertaken off-line. During execution, the neural solution therefore allows for extension to far more complex mathematical models without incurring a notable slowing of the process time.

11 References

[1] **Ralph Richter**:
Anwendung Neuronaler Netze in der Produktionstechnik, MSc Thesis (Diplomarbeit) p. 35, Lehrstuhl für Produktionstechnik, Technische Universität Dresden, May 1994, Germany.

[2] **Yoh-Han Pao**:
Adaptive Pattern Recognition and Neural Networks, Addison Wesley Publishing Company Inc.

[3] **Ralph Richter, Dr Roland Burns**:
An Artificial Neural Network Autopilot for Small Vessels, proc. of UKSS '93, Keswick, UK.

[4] **Warren McCulloch, Walter Pitts**:
A Logical Calculus of the Ideas Immanent in Nervous Activity, Bulletin of Mathematical Biophysics, vol. 5, pp. 115-133.

[5] **D. Rumelhart, J. McClelland**:
Parallel Distributing Processing, MIT Press, Cambridge, Massachusetts 1986.

[6] **Burns, R. S.**:
Application of the Riccati Equation in Control, Systems and Signals, International Federation of Automatic Control, Como, Italy, 26-28th June 1989.

[7] **Morse, R. V.; Price, D.**:
Manoeuvring Characteristics of the Mariner Type Ship (USS Compas Island) in Calm Seas, Sperry Polaris Management, Sperry Gyroscope Company, New York, 1961.

R. Richter, R. S. Burns, and M. N. Polkinghorne.

A review of the historical development of ship motion control and simulation.

In *Symposium: Applications of Advanced Marine Control*, pages 1–8, Plymouth - UK, 30 November 1995. ISBN 0-905227-46-8,

also published in *Applications of Artificial Intelligence for Technological and Business Processes, Technology Transfer Series* vol (1):1–8, 1996, ISBN 0-905227-53-0.

A Review Of The Historical Development Of Ship Motion Control And Simulation

Ralph Richter[1], Martyn N Polkinghorne[2], Roland S Burns[1,2]
University of Plymouth,
Devon,
United Kingdom

Abstract

This paper reflects a brief history on the development of ship simulations and ship autopilots and as such only to summarise the range of the developments have been made in this field.

1 Introduction

For merchant ships an accurate and reliable automatic steering device becomes a real money saving proposition, largely justifying its use.

On battleships, by its use the absence or reduction of yawing in action means a better efficiency in gunfire, increased maneuvering speed and also a greater cruising radius.

Quotation 1: Minorsky in [10], p. 280

An efficient helmsman keeps the ship accurately on her course by exerting a properly timed 'meeting' and 'easing' action on the rudder, i.e., by taking into consideration the elements characterizing the motion from the dynamical standpoint, namely, the instantaneous angular velocity of yawing as well as its time variations.

It has often been stated that the human intuition of the helmsman cannot be replaced by any mechanical contrivance whatever its nature may be.

Quotation 2: Minorsky in [10], p. 282

... if we connect the rudder with the direction indicating apparatus whatever its nature may be (magnetic, gyroscopic or radio compass) by any appropriate means, for example by means of an ordinary follow-up system.

Quotation 3: Minorsky in [10], p. 282

The control task of ship navigation can be subdivided into two major divisions. The course related autopilots attempt to optimise ship orientation *rather than the ship's position.* The main control task is therefore to maintain or change, the heading of the ship to minimise

[1] School of Manufacturing, Materials and Mechanical Engineering
[2] Plymouth Teaching Company Centre

the error from the desired value. Conversely, the track related autopilots optimise the position of the vessel and not its orientation.

2 Early Developments until 1930

An very important and early paper [10] was published by *MINORSKY* in 1922. This paper discusses the stability problems of automated steering and developed the basic theory of 'directional stability of automatically steered bodies'.

Furthermore *MINORSKY* subdivided the control problem into individual, smaller problems such as rudder position control, rudder angular velocity control, rudder angular acceleration control

Similary, *SPERRY* described the first installation of a gyrocompass aboard a ship in 1922. In this publication [16] he considered the problems that occur with automatic steering using a gyrocompass. In this very early application we can find all the elements that make up the control loop of an automated steering system for course keeping purposes. By 1932, this application had been installed on more than 400 merchant ships all over the world.

In 1923, *SCHULER* [15] desribed the behaviour of pendulums and gyroscopes when accelerated in a horizontally direction. The doubts rised by *MARTIENSSEN* [9] in 1906 based on calculating gyroscopic compasses errors under north-south acceleration were fundamental for further research in this field.

> *By working out some examples, Martienssen came up with very great errors of the compass and concluded therefrom that the gyroscopic compass is useless as an accurate direction indicator for navigation.*

Quotation 4: Schuler in [9], p. 26

However, *SCHULER* continued the quotation in the following way:

> *I asked myself the question: would this sort of acceleration error be capable of elimination by an appropriate construction?*
>
> *The aswer is, yes. And the solution is almost trivial.*

Quotation 5: Schuler in [9], p. 26

Utilisation of these, and the subsequently derived equations and thoughts finally led to the successful gyroscopic devices now common place. The difficulties of the first years have been covercome and gyroscopes can be found in most navigation devices which require a degree of accuracy.

The autopilot used for the period 1930 to 1950 was a rather simple controller. The heading error produced a signal which was then used to adjust the steering mechanism. The controller can be seen as a proportional controller. It was a possible to adjust the the control parameter (K_p) to suit different conditions eg. ship loading. Obviously this simple device could not cope with a wide range of conditions, i.e. in rough weather conditions when the proportional controller forced the steering mechanism to be heavily used and therefore worn out very quickly. A weather adjustment was therefore necessary to prevent this exess wear. In most cases a simple dead-band was introduced to aviod high frequently and small magnitude movements. The rudder was then only changed if the control output exeeds a small specified rudder angle. A different method to avoid rudder wear was given by including a delay feedback. This delay caused the rudder to move until a prespecified rudder angle has been reached. The rudder could not stop or change direction until this angle has been exeeded. *NOMOTO* [12] described this method as 'negative backlash'.

3 Post World War II

During this period, overshaded by the two world wars, the autopilots used were mainly simple mechanical devices following a simple proportional rule.

$$\delta = K_p \cdot \varepsilon_\psi$$

Where: ε is heading error **Equation 1**

Those pilots were not very satisfactory and could not prevent overshooting and therefore often caused transient oscillation.

In the 1950s, an improvement in stability could be achived by the introduction and use of the mainly first derivative of the heading error ($\dot{\varepsilon}_\psi$) or the rate of turning (angular velocity $\dot{\psi}$). The first commercial autopilot utilising this technique was installed in 1951 on the *S. S. UNITED STATES*. The control rule of this autopilot may can be defined as:

$$\delta = K_p \cdot \varepsilon_\psi + K_D \cdot \dot{\varepsilon}_\psi$$

Equation 2

At about the same time, a further term was also added to the contol equation this being the integrale of the heading error, the resulting control law being (Equation 3).

$$\delta = K_p \cdot \varepsilon_\psi + K_D \cdot \dot{\varepsilon}_\psi + K_I \cdot \int \varepsilon_\psi dt$$

Equation 3

Thus, the PID control rule was formulated. Furthermore the addition of the integral term assisted to neutralise the rudder movements as well as steering gear lags. Constant disturbances, causing an offset were now considered and the PID autopilot was fully capable of dealing with them.

Nevertheless, controllers based on the PID format could not prevent the high frequency rudder movements. The introduction of a dead-band in the rudder loop could lead into unstable behaviour. *MOTORA* [11] suggested in 1953 to apply a low-pass filter to the output signal to prevent rudder oscillating. According to *RYDILL* [14] this may generate a loss in stability and hence he recommended the use a quadratic delay technique.

4 Adaptive Autopilots

Soon it was determined, that even the most advanced PID performance could be improved by adjusting its parameters according to the environment that the control system (ship and autopilot) was operating in. This can be achieved by two methods; manually or automatically. The disturbances, and therefore the effects to the hull, can also be subdivided onto two major categories:

a) disturbances that cause a 'small' deviation of the desired course and

b) disturbances which change the vessel's characteristics and consequently the steering characteristics.

Weather and tidal changes like waves, wind and current can be associated with the first group. Changing the mass of the vessel whilst loading/ unloading and the resutling draft and displacement, the quantity of water under the keel and alterationss in the forward speed change the handling characteristics of the vessel and are therefore assosiated with the second category. Small adjustments required to compensate for the disturbances of group a) can be overcome by automatic adjustments. Disturbances of group b) require major corrections and are mainly undertaken by the operator. Those adjustments demand a significant knowledge on the handling characteristics of the ship and the environment/ disturbances.

4.1 Model Reference

This approach is based on the comparison of measured, actual data and data of an ideal mathematical model (reference model). An error function is derived using those data. This function (criterion) is then minimised.

In 1974, *VAN AMERONGEN* underlined in [17] the importance of adapting parameters of the autopilot and compared two methods of model referencing. In this paper he describes both of the following approaches to tackle the 'fixed settings problem'.

4.1.1 SENSITIVITY MODELS

> *The dynamic behaviour of the ship and hence also the parameters of this model are dependent on the external circumstances and the applied thrust-power. When the ship is steered with an autopilot it is necessary to adjust the parameters of the autopilot dependent on the change of the steering characteristics of the ship.*

Quotation 6: van Amerogen in [17], p. 441

This technique of the 'sensitivity model' is especially designed to prevent course instability of very large ships. The criterion used in that approch can be defined as:

$$C = \int_0^T \tfrac{1}{2}\varepsilon^2 dt$$

Equation 4

Using the steepest descent method, the gain K_d of the rate feedback signal is adjusted. Unfortunately this approach is not stable under all circumstances.

4.1.2 LIAPUNOV APPROACH

This approch follows the principle of direct adjustment of the controller's parameters. Assuming the same order of the model's transfer function and the system's one, a difference between the state variables of the system and the model is utilised to adjust the system's parameters in order to minimise this difference. Furthermore the process is assumed to be linear and that no stochastic disturbances occur. A low-pass filter also is required in rough seas. The difference between the system's and the model's responses is minimised by the differences between the state variables.

VAN AMERONGEN concluded that there is no significant difference between both approaches, the sensitivity model and the Liapunov approach.

4.2 Self-tuning Autopilots

First develpoments of "cost function for adaptive course-keeping autopilots" were undertaken by *ASTROM* and *EYKHOFF* [2] in 1973. The method used was based on a least squares parameter estimator and a minimum variance control technique.

Special attention should be given to the cost function. Assuming the vessel is left to yaw naturally (without high frequent rudder corrections), the traveled distance during a 400 miles journey will not increase more than a quarter of a mile when the deviation of the course remains ± 2° [13]. In contrast, each rudder movement causes a drag and so a loss in forward speed.

In 1975, *CLARKE* and *GAWTHROP* [5] developed a more generalised 'self-tuning controller'.

It has beeen demonstrated by *BURNS* [4] that it is possible to design an optimal multi-variable ship guidance system that controls position, heading and speed simultaneously, and such a system can work within the constraints required in port approaches.

5 Latest Developments/ Intelligent Control

It is very obvious, the classical and tuned PID autopilot has limitations. It is always fascinating how human operators can cope with a very wide range of unknown and uncertain conditions. Latest research in this field attempts to adapt human abilities like learning and experience to design a controller with an increased level of performance.

5.1 Neural Networks

The first noteable paper utilising this technique for the ship contrlo application was published by *ENDO* [6]. The training data to teach the neural network generated by a PD controller. Further work in this field is been published by the author himself and many other researchers. An very interesing paper was published by *HEARN* [8] where he explained the use of a backpropagation neural network for on-line learning. To be perfectly correct, the controller is not truely learning on-line, but using a relativly fast computer, the learning can be done within the sampling time of the system. The training of the network could be finished within approximately 0.5 seconds.

The back propagation learning (BP) algorithm is based on the gradient (steepest descent) method. It minimises an error function. In the case of BP, the error is defined as:

$$E = \frac{1}{2}(d - y)^2$$

d .. vector of the desired outputs

y .. vector of the actual outputs (actual plant response)

Equation 5

Obviously, the desired output vector, in the case of a ship autopilot a single output, contains only the desired course. The plant response is a function of the rudder angle and using the chain rule, and some further assumptions, a control signal can be learned which minimises the difference (error) between desired and actual course.

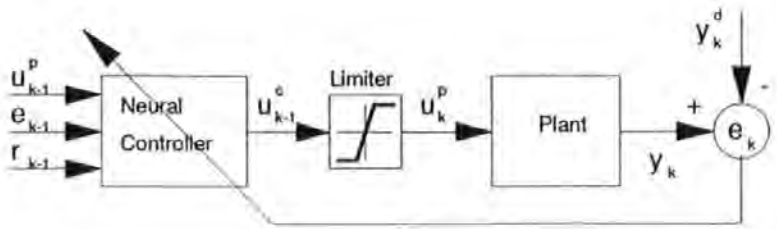


Fig. 1: Direct Neural Control Scheme [8]

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial u_k^p}\frac{\partial u_k^p}{\partial u_k^c}\frac{\partial u_k^c}{\partial w_{ij}}$$

w_{ij} ... weight to adjust to change the output

Equation 6

More work on track keeping and related tasks like rudder roll stabilisation, course keeping etc. is being undertaken by the University of Plymouth including the authors of this paper, N. *WITT* and D. R. *SUTTON*.

5.2 Fuzzy Logic

A further method of simulating human behavior is achieved by using linguistic variables and derived rules. The controller's task is to use a human-like way of thinking. The knowledge is put into a rulebase and the inputs are given in a fuzzy form. The use of so called fuzzy sets supports the human way of expressing every day actions and understandings - tall is tall and not 3.2 m - warm is about 22 °C.

Nowadays, even more advanced techniques are used. Self-organising fuzzy logic control is probably the latest development in that field.

5.3 Genetic Algorithms

To adapt the biological evolution is the background of genetic algorithms (GA). It is a search technique to optimise a cost function, in this case called fitness function. Using biological rules, eg. only the fittest survives and mutation are translated into a computer understandable format to find the absolute minimum/ maximum of the given fitness function.

6 References

[1] ASTROM, K. J.; EYKHOFF, P.: *System Identification - A Survey.* Automatica, Vol. 7, 1971.

[2] ASTROM, K. J.; WITTENMARK, B.: *On Self Tuning Regulators.* Automatica, Vol. 9, 1973.

[3] BURNS, Roland S.: *The Automatic Control of Large Ships in Confined Waters.* PhD Thesis, Plymouth Polytechnic, September 1984, United Kingdom.

[4] BURNS, Roland S.: *The Design, Development and Implementation of an Optimal Guidance System for Ships in Confident Waters.* proc. of the 9th Ship Control Systems Symposium, vol. 3, Bethesda, USA, 1990.

[5] CLARKE, D. W.; GAWTHROP, P. J.: *Self Tuning Controller.* proc. IEEE 122, part D, No 9, Sept. 1975.

[6] ENDO, M.; VAN AMERONGEN, J.; BAKKERS, A. W. P.: *Application of Neural Networks to Ship Steering.*

[7] FOSSEN, Thor I.: *Guidance and Control of Ocean Vehicles.* John Wilet & Sons, Chichester, New York, Brisbane, Toronto, January 1994, ISBN 0-471-94113-1.

[8] HEARN, G. E.: *Ship Motion Control by an On-line Trained Neural Controller.* proc. of the 3rd International Conference Manoeuvring and Control of Marine Craft MCMC'94, Southampton, UK, 7-9 September 1994.

[9] MARTIENSSEN, O.: Journal of Physics, Vol. VII, p. 525 ff., 1923, Kiel, Germany.

[10] MINORSKY, N.: *Directional Stability Of Automatically Steered Bodies.* Proc. of the 13th general meeting of SNAME (Society of Naval Architects and Marine Engineers), 1922, New York, U.S.A., pp. 280-309.

[11] MOTORA, S.: *On the Automatic Steering and Yawing of Ships in Rough Seas.* Journal of SNA of Japan, Japan, 1953.

[12] NOMOTO, K.: *Directional Stability of Automatically Steered Ships with Particular Reference to Their Bad Performance in Rough Seas.* Proc. of 1st Symposium on Ship Maneuvrability, DTMB, May 1960, DTMB report No. 1461, October 1960.

[13] NOMOTO, K.; MOTOYAMA, T.: *Loss of propulsive Power Caused by Yawing with Particular Reference to Automatic Steering.* Japan Shipbuilding and Marine Engineering, 120, Japan, 1960.

[14] RYDILL, L. J.: *A Linear Theory for the Steered Motion of Ships in Waves.* Trans. RINA, 1958.

[15] SCHULER, Maximilian: *The Disturbance of Pendulum and Gyroscopic Apparatus by the Acceleration of the Vehicle.* Navigation, Journal of The Institute of Navigation, Vol. 14, No. 1, Spring 1967.
original version in 'Journal of Physics', Vol. XXIV, July 1923, Kiel, Germany.

[16] SPERRY, Elmer: *Automatic Steering.* Journal of ASNE (American Society of Naval Engineers), 1922.

[17] VAN AMERONGEN, J.: *Model Reference Adaptive Autopilots for Ships.* Automatica, Vol. 11, pp. 441-449;
original version in: Proc. IFAC/ IFIP Symposium on Ship Operation Automation, July 1973, Oslo, Norway.

R. Richter, R. S. Burns, M. N. Polkinghorne, and P. Nurse.

Modelling and control of surface ships by employing a fuzzy-neural solution.

In *11th International Conference on Systems Engineering (ICSE '96)*, pages 7–12, Howard R. Huges College of Engineering, University of Nevada - Las Vegas, Box 454005, Las Vegas, NV 89154-4005, USA, 9–11 July 1996.

Modelling and Control of Surface Ships by Employing a Fuzzy-Neural Solution

Ralph Richter[1], Roland S. Burns[1,2], Martyn N. Polkinghorne[2], Peter Nurse[1]
[1] School of Manufacturing, Materials & Mechanical Engineering (SMMME)
[2] Plymouth Teaching Company Centre
University of Plymouth, Drake Circus, Plymouth, Devon PL4 8AA, United Kingdom
e-mail: ralphr@soc.plym.ac.uk

Abstract

The classical approach to modelling the dynamic behaviour of rigid bodies is to express their behaviour as a set of simultaneous linear and non-linear differential equations, and to obtain a solution for various input stimuli. An alternative approach is that of system identification whereby a given input such as a sinusoid or pseudo-random binary sequence (PRBS) is applied to the real system and from a set of input/output measurements a mathematical model may be obtained. This paper investigates a novel alternative to the state variable representation of a ship in three degrees of freedom, by the application of an Artificial Neural Network (ANN).
A surface ship is modelled by a set of non-linear differential equations in three degrees of freedom. Using measured hydrodynamic coefficients, a discrete, time varying, state variable mathematical model is constructed, and validated against full-scale sea trials.
Based on multivariable system theory it is possible to formulate an optimal control policy that minimises a performance index. However, if the dynamic characteristics of the vessel change (due to variations in forward speed, for example) then the guidance system is suboptimal and its parameters need to be re-computed.
The possibility of using a model (such as a neural network) of a vessel to predict the performance of the ship according to disturbances and rudder changes to optimise a rulebase of a fuzzy logic controller is described, with the objective of providing a system which adapts its parameters so that it provides optimal performance is provided over a range of conditions.

1. Introduction

Since the expansion in the number of powerful computers and workstations available, simulations of complex structures (plants) are increasingly part of the design process. By the use of these techniques, off-line study of the plants and/or controller designs may be achieved when otherwise no realistic study could be undertaken. Differential equations are used to describe the dynamic behaviour of the system being studied. To increase the accuracy of the design analysis, more time consuming calculations are necessary because of reduced stepsize, but this is not always available.

In particular, neural networks demonstrate the capability to model highly complex plants. By the application of training data derived from the real environment, these networks can learn to emulate a wide range of differing conditions. Once trained, the neural network substitutes the plant's model and operates instead. Artificial neural networks (ANNs) have previously demonstrated their capability to model highly complex plants [4]. By deriving data from the real world environment, e.g. by measuring critical values, these networks can learn to emulate a wide

range of different conditions. Once trained, the neural network can be used as a model in simulations and other applications. Considering a maritime vessel, the non-linear time-invariant dynamic characteristics are particularly difficult to model. Using a carefully selected range of manoeuvres undertaken at various forward speeds, a comparison can be made between a conventional ship model and the neural network developed.

2. Ship Mathematical Model

The classical approach of modelling the dynamic behaviour of rigid bodies is to express their behaviour in a set of simultaneous linear and non-linear differential equations, and to obtain a solution for various input stimuli.

Ship motions in surge, sway and yaw can be described [1] by an Eulerian set of non-linear differential equations of the form :

Surge Equation:

$$\begin{aligned} m\dot{u} + mqw - mrv = \quad & X_{\dot{u}}\dot{u} + X_u(u+u_c) + X_{uu}u^2 + X_{uuu}u^3 + X_{vv}v^2 + X_{rr}r^2 \\ & + X_{\delta\delta}\delta_A^2 + X_{un}un_A + X_{nn}n_A^2 + X_{ua}u_a + X_{zz}z^2 + X_{\theta\theta}\theta^2 \end{aligned} \tag{1}$$

Sway Equation:

$$\begin{aligned} m\dot{v} + mur - mpw = \quad & Y_{\dot{v}}\dot{v} + Y_v(v+v_c) + Y_{\dot{r}}\dot{r} + Y_r r + Y_{nn}n_A^2 + Y_{vvv}v^3 + Y_{rvv}rv^2 \\ & + Y_{nn\delta}n_A^2\delta_A + Y_{nn\delta\delta\delta}n^2\delta_A^3 + Y_{\delta vv}\delta_A v^2 + Y_{va}v_a \end{aligned} \tag{2}$$

Yaw Equation:

$$\begin{aligned} I_z\dot{r} + (I_y - I_x)pr = \quad & N_{\dot{v}}\dot{v} + N_v(v+v_c) + N_{\dot{r}}\dot{r} + N_{nn}n_A^2 + N_{vvv}v^3 + N_r r + N_{rvv}rv^2 \\ & + N_{nn\delta}n_A^2\delta_A + N_{nn\delta\delta\delta}n_A^2\delta_A^3 + N_{\delta vv}\delta_A v^2 + N_{va}v_a \end{aligned} \tag{3}$$

Equations (1) to (3) can be arranged in the state matrix vector form:

$$\dot{\mathbf{x}}(t) = \mathbf{F}(t)\mathbf{x}(t) + \mathbf{G}_c(t)\mathbf{u}(t) + \mathbf{G}_D(t)\mathbf{w}(t) \tag{4}$$

The corresponding discrete solution is:

$$\mathbf{x}((k+1)T) = \mathbf{A}(T,kT)\mathbf{x}(kT) + \mathbf{B}(T,kT)\mathbf{u}(kT) + \mathbf{C}(T,kT)\mathbf{w}(kT) \tag{5}$$

where:

$$\mathbf{x}^T = (\delta_A \ \mathrm{n}_A \ \mathrm{x} \ \mathrm{u} \ \mathrm{y} \ \mathrm{v} \ \mathrm{z} \ \mathrm{w} \ \phi \ \mathrm{p} \ \theta \ \mathrm{q} \ \psi \ \mathrm{r}) \tag{6}$$

$$\mathbf{u}^T = (\delta_D \ \mathrm{n}_D) \tag{7}$$

$$\mathbf{w}^T = (u_c \ \mathrm{v}_c \ u_a \ \mathrm{v}_a \ \zeta_x \ \zeta_y) \tag{8}$$

For this study, it was necessary to concentrate on three degrees of freedom. These being surge, sway and yaw.

The vessel's parameters used in this simulation are given below (Table 1), and are based on the Morse and Price data for the *Mariner Hull* [2].

A neural network is required to model the behaviour of large ships. The precise relationships between many of the features and characteristics of these ships are not fully understood. To determine them, it is possible to employ a neural network. Rudder angle, and engine speed cause speed changes in the surge, sway and yaw directions (u, v, r). Since we are not only interested in the steady state response of the vessel, but also in the transient behaviour, it is essential to consider the time elapsed since the last rudder change as another input.

Table 1: Vessel Parameter		
Length	=	161m
Draught	=	9m
Beam	=	23m
Displacement	=	17,000t
Number of propellers	=	1
Number of rudders	=	1
Maximum rudder angle	=	35°

Figure 1 indicates the various time periods required for the response to settle down.

Utilising an acceptable error of ±1%, we can determine from the data (Table 2) the following values. Therefore it is possible to state that if the time considered is bigger than the time to reach steady-state, then the response has reached steady-state, otherwise the response remains in the transient period and the operation of the artificial neural network is required.

Table 2 settling times

d [°]	u	t [s]	v	t [s]	yaw rate	t [s]
0	7.588	275	0.278	410	-0.186	405
10	6.639	270	0.734	220	-0.534	180
-10	6.818	270	-0.669	230	0.476	230
20	5.915	295	0.870	240	-0.681	185
-20	6.043	290	-0.836	235	0.641	240
30	5.308	290	0.930	240	-0.782	155
-30	5.408	285	-0.907	240	0.747	165

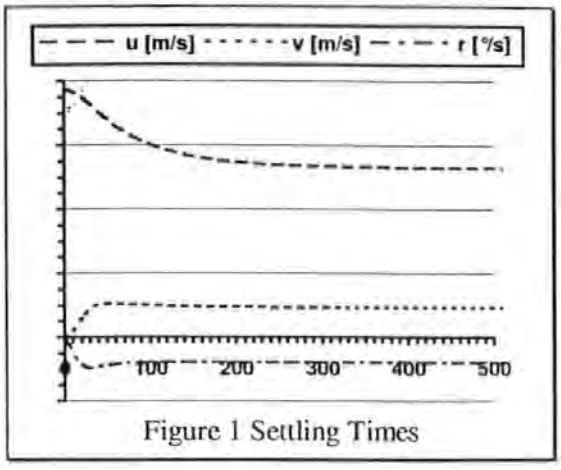


Figure 1 Settling Times

3. Ship Model using a Neural Network Approach

3.1 Structure of the ANN

It is a pre-requisite that the variables to be investigated are considered before commencing design of the network's structure (see Table 3).

Table 3 Structure of the Network

Inputs		Outputs		
rudder angle	engine speed	forward speed	lateral speed	turning rate
δ	n	u	v	r

To learn the transient behaviour, it is necessary to determine the time elapsed since the last rudder change as an additional further input. Thus, the interface to the outside world is defined.

A 3-6-6-6-3 network was identified as being most suitable for this application following a programme of heuristic experimentation. The quality of results obtained from a two hidden layer network proved unsatisfactory. Obviously, the transients, with their associated overshoots, are difficult to understand, and were therefore filtered out. Using more than two hidden layers the error was reduced and overshoots were replicated giving a suitable level of network performance.

4. Network Training

The learning method utilised for this study was the back propagation algorithm. This algorithm is based on the minimisation technique called steepest descent or gradient method. The transfer function employed was the popular *Sigmoid* function. The output was in the limits between 0 and 1 ($0 < y < 1$), where these values are reached at infinity. Therefore, the desired outputs had to be scaled within these limits. During the learning process the trend of the error development was observed and it could be seen that the network stuck in local minima. By increasing the number of hidden layers, the error surface contains less troughs and a more constant learning was achieved. Furthermore the learning rate was adjusted from an initial large learning rate with gradual decrements until the finished level of learning was achieved (steady error).

5. Training Results

Results of the learning are given in Figures 2 to 4. Figure 2 displays the forward speed and demonstrates how the response of the network closely follows that of the surge velocity training data. An improved level of performance is identified by the response for sway velocity data, and also for that of yaw rate with increments in rudder angle of 0°, -10°, +10°, -20°, +20°, -30° and +30° is displayed.

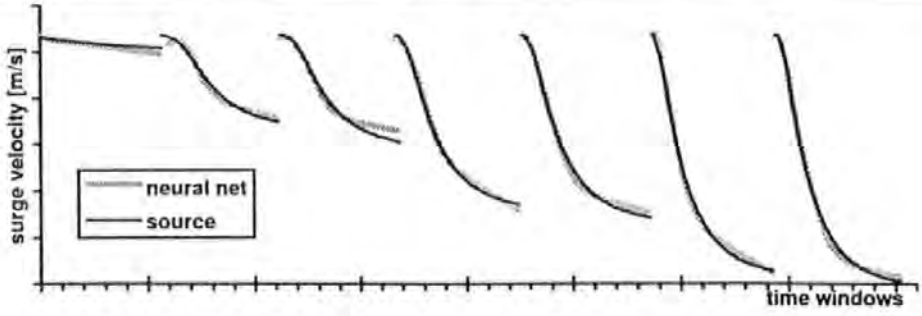


Figure 2 Surge Velocity Response

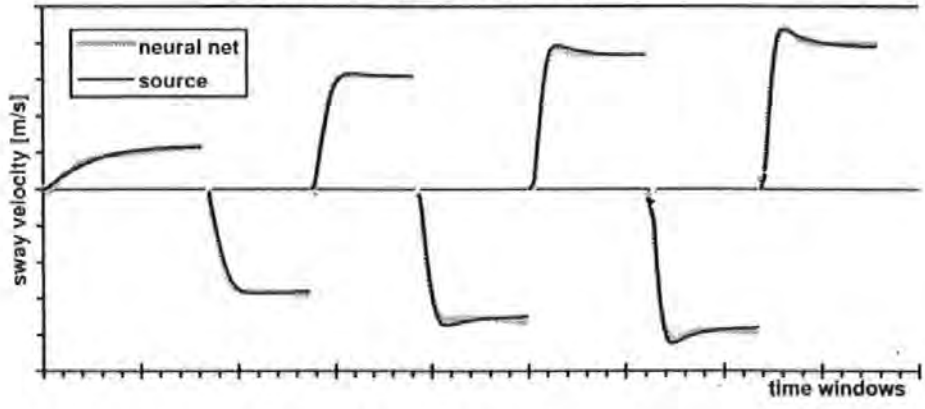


Figure 3 Sway Velocity Response

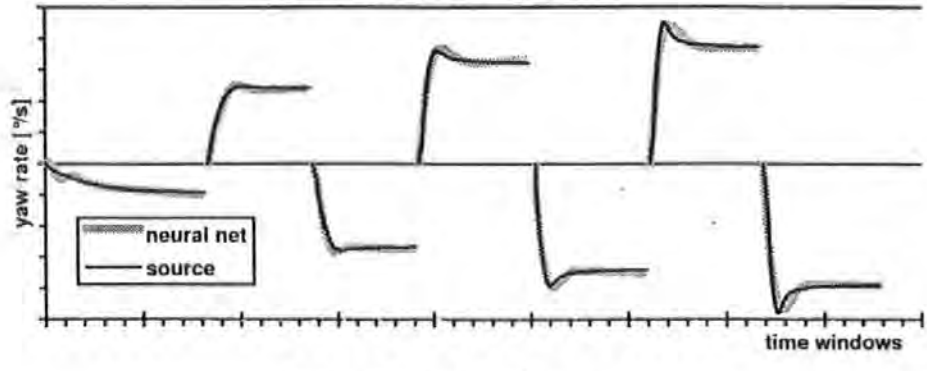


Figure 4 Yaw Rate Response

The actual outputs match very closely the desired outputs given in the training sets which clearly demonstrates the learning success of the network design utilised. Further work by the authors will concentrate on the implementation of this design of network during simulated sea trial conditions. Results will then be compared to those obtained from a traditional ship model to validate both the learning achieved, and the subsequent performance capability obtainable during simulation studies.

6. A Novel Predictive Control Principle

This kind of mathematical model can be used in a novel controller design. It has been found and demonstrated that adaptive control is the control technique of the future. Recent research into *self-organising* (*self-tuning*) methods, M. Polkinghorne *et al.*, has been applied for control purposes when processes operate in uncertain, varying environments. Such existing adaptive controllers, e.g. self-organising fuzzy logic controllers (SOFLC), learn by employing a heuristic approach. In order to learn they have to work with a poor performance. Due to adequate techniques, the errors made during this low performance work are detected and the control parameters are adjusted in such a way to avoid the same error in the future. To fully appreciate the novel approach of this research, it is a prerequisite that the conventional SOFLC technique be described. The SOFLC technique has been employed in several maritime applications ([3] and [5]), its performance and reliability having previously validated.

6.1 Performance Index Operation

The *Performance Index* (PI) determines the performance of the controller and it indicates the performance level of the controller when reacting. In general, the PI has a similar structure to the fuzzy rulebase used in the forward phase of the fuzzy inference. The output parameters of the process are used as inputs to the PI. Before any analysis can be done, the change on the control actuator has to take effect; a certain amount of time has therefore to elapse. This time delay is characterised by the time constants of the process and may be referred to as 'delay in reward'.

The PI outputs are a measure which can be used directly to adjust the rulebase. The rulechanging algorithm consists of three main phases.

- standard fuzzy logic control (defuzzification, fuzzy inference, defuzzification). The active rules and values are stored for later use in the tuning. A control output is created which is fed into the process (control actuator).
- The process 'reacts' in an appropriate (in its characteristic) manner with the actual output value being measured by a device in the feedback loop of the control system.
- This output value is then forwarded to the Performance Index (PI) which will generate a measure of the controller performance. If a zone of poor performance has been hit, rulebase adjustment is needed. Now, n time steps later, the rules and values hit to form this control output are adjusted according to the performance index criteria, therefore when this combination of rules is activated in the future, the control output will be modified, to produce a reduced error.

However, this technique is based on the assumption that the controller output n time steps before is responsible for the present state of the process. If the process is in a poor state, these rules must be changed which is a retrospective methodology only allowing adjustment of control parameters which already performed badly.

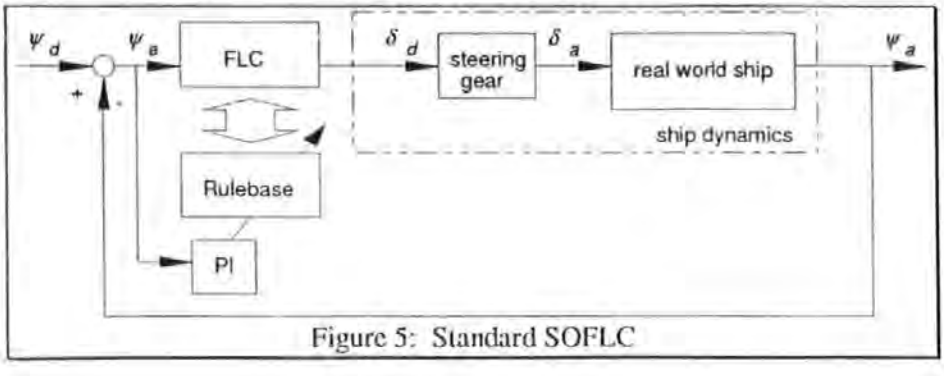


Figure 5: Standard SOFLC

By employing a predictive technique the disadvantage resulting from that time delay may be overcome, thereby producing an enhanced level of operational performance.

6.2 A New Model Predictive Technique

The block diagram (Figure 6) shows the principle of the new predictive self-organising fuzzy logic controller (PSOFLC). The control output is only passed to the process if the predicted performance of the process following the application of this control action indicates a satisfactory performance level. If the output is poor, then the rulebase is adjusted to meet the requirements.

To determine a high quality model is always a problem in this type of control application. It has been demonstrated [4] that neural networks are well able to learn the transient behaviour of a process. Therefore it must be possible to obtain a very specific mathematical model by measuring the present behaviour of the vessel and teaching a neural network on-line utilising this data. This adaptive model can than be used as the predictive model in the described PSOFLC. A method must therefore be identified to teach the network on-line by learning measured data whilst a journey takes place. In this manner, the model can adapt itself to behave exactly like the vessel when working in the same environment (mass loading, forward speed, *etc.*). The model will change itself if the ship characteristics change, so that the model will always represent the present state of the ship.

7. Conclusions

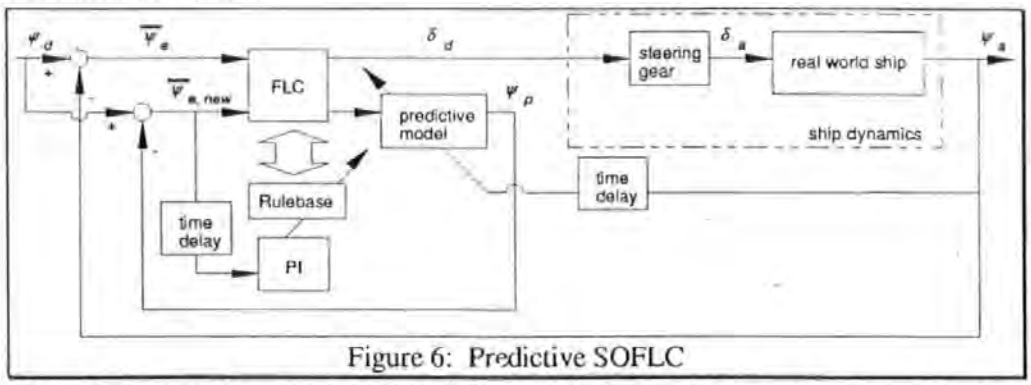


Figure 6: Predictive SOFLC

It has been demonstrated by this study that it is possible to simulate complex plant behaviour in an innovative manner by utilising a neural network. The advantage of employing a simulation using this technique is that it becomes possible to overcome the problems associated with formulating the relationship between the features to be investigated. This can be achieved by the neural network, thus allowing the designer to concentrate on alternative aspects of the design. During execution, the neural solution therefore allows for extension to far more complex mathematical models without incurring a notable slowing of the process time.

This new control technique indicates a new research area for predictive control which will allow controllers to avoid areas with unsatisfactory control performance and therefore will contribute significantly to improved efficiency and safety.

8. References

[1] BURNS, Dr. Roland: *Application of the Riccati Equation in Control, Systems and Signals*, International Federation of Automatic Control, Como, Italy, 26-28th June 1989.

[2] MORSE, R. V. and PRICE, D.: *Manoeuvring Characteristics of the Mariner Type Ship (USS Compas Island) in Calm Seas*, Sperry Polaris Management, Sperry Gyroscope Company, New York, 1961.

[3] POLKINGHORNE, M. N.: *A self-organising fuzzy logic autopilot for small vessels.* PhD Thesis, University of Plymouth, United Kingdom, 1994.

[4] RICHTER, R.; BURNS, R. S. and POLKINGHORNE, M. N.: *Application of an artificial neural network to model complex plant behaviour.* IFAC '95 - Workshop on Motion Control, Munich - Germany, 9-11 October 1995.

[5] SUTTON, R. and JESS, I. M.: *A Design Study of a Self-Organising Fuzzy Autopilot for Ship Control.* Proc. Institution Mech. Engineers, 205, Pt I, pp 35-47, 1991.

R. Richter, R. S. Burns, M. N. Polkinghorne, and P. Nurse.

A predictive ship control using a fuzzy-neural autopilot.

In P. A. Wilson, editor, *Proc. of 11th Ship Control Systems Symposium*, pages 161–172, Southampton, United Kingdom, April 1997.

A Predictive Ship Control using a Fuzzy-Neural Autopilot

R. Richter R.S. Burns M.N. Polkinghorne P. Nurse
University of Plymouth, Drake Circus,
Plymouth, Devon PL4 8AA, United Kingdom

Abstract

Intelligent methods of control which have been employed in an attempt to maintain optimal marine autopilot performance have all been retrospective in nature, thereby allowing performance levels to deteriorate before remedial action can be subsequently applied. Of significantly more interest would be a system which is capable of anticipating such performance deterioration prior to its occurrence so that corrective action may be applied in expectation of events by combining aspects of both modelling and control.

Advancing from the classical approach to modelling the dynamic behaviour of rigid bodies by expressing behaviour as a set of simultaneous differential equations using calculated hydrodynamic coefficients, or by the application of a series of pseudo-random binary sequence (PRBS) to the real system, a novel alternative to the state variable representation of a ship in three degrees of freedom is demonstrated employing an artificial neural network approach. Using this enhanced model, it is therefore possible for the neural network model to predict the performance of the ship, and for this information to then be channelled to an intelligent control device, with any necessary rudder changes to optimise a rulebase of a fuzzy logic controller then being calculated in an anticipatory mode of operation.

1 Introduction

The modern control techniques of H_∞ [4], Optimality [2], Self-Tuning [6], Model Reference [10], Neural Networks [3] and Fuzzy Logic [11] have all been applied to the field of ship control over recent years in an attempt to improve autopilot performance over the entire operating envelope. Whilst these techniques have successfully demonstrated that adaptive control methodologies

are the future of marine based autopilots, few actually offer a learning capability in the true sense, and of these all are retrospective learning, i.e. poor autopilot performance must be encountered before corrective action is applied to remedy the situation. Even very recent studies ([7]- [8]) employing self-organising fuzzy logic control has required that autopilots learn in a heuristic manner. Therefore it is only when errors due to poor performance are detected that control parameters are adjusted to prevent any recurrence should similar conditions be encountered in the future. The truly ideal form of autopilot control would be one with the capability of prediction so that deterioration in performance could be anticipated in advanced, with suitable remedial actions being undertaken prior to this occurrence thereby preventing any noticeable deviation from the optimal performance level at any times. Such a system would require a knowledge of alterations occurring within the dynamic characteristics of the vessel and the implications generated by these changes. To model boats in the conventional manner would be impractical on any large scale due to the considerable effort involved in both time and resources. However, if a novel manner of ship modelling could be derived involving a neural network [9] solution based upon relatively little information which was readily available, then this could be the key to developing such a predictive system. The system itself must have the capablility of on-line learning to be able to fully support the requirements of a predictive system, e.g. by the application of self-organising fuzzy logic. However, it is only by combining both neural and fuzzy aspects together into a composite autopilot system, that a fully predictive novel innovative control system may be obtained. This paper concentrates on the development of such a neural network ship model, consideration of the on-line tuning ability of the fuzzy autopilot when subjected to full scale sea trials, together with discussion of the implications when joined into the necessary composite system.

2 Development of a Neural Network Ship Model

2.1 Modelling techniques

The classical approach of modelling the dynamic behaviour of rigid bodies is to express their behaviour in a set of simultaneous linear and non-linear differential equations, and to obtain a solution for various input stimuli. Ship motions in surge, sway and yaw can be described [1] by an Eulerian set of non-linear differential equations of the form:

Length =	161m
Draught =	9m
Beam =	23m
Displacement =	17,000t
Number of propellers =	1
Number of rudders =	1
Maximum rudder angle =	35°

Table 1: Vessel Parameters

Surge Equation:

$$\begin{aligned} m\dot{u} + mqr - mrv = \; & X_u\dot{u} + X_u(u+u_c) + X_{uu}u^2 + X_{uuu}u^3 + X_{vv}v^2 \\ & + X_{rr}r^2 + X_{\delta\delta}\delta_A^2 + X_{un}un_A \\ & + X_{nn}n_A^2 + X_{ua}u_a + X_{zz}z^2 + X_{\theta\theta}\theta^2 \end{aligned} \tag{1}$$

Sway Equation:

$$\begin{aligned} m\dot{v} + mur - mpv = \; & Y_v\dot{v} + Y_v(v+v_c) + Y_r\dot{r} + Y_r r + Y_{nn}n_A^2 \\ & + Y_{vvv}v^3 + Y_{rvv}rv^2 + Y_{nn\delta}n_A^2\delta_A \\ & + Y_{nn\delta\delta\delta}n_A^2\delta_A^3 + Y_{\delta vv}\delta_A v^2 + Y_{va}v_a \end{aligned} \tag{2}$$

Yaw Equation:

$$\begin{aligned} I_z\dot{r} + (I_y - I_x)pr = \; & N_v\dot{v} + N_v(v+v_c) + N_r\dot{r} + N_r r + N_{nn}n_A^2 \\ & + N_{vvv}v^3 + N_{rvv}rv^2 + N_{nn\delta}n_A^2\delta_A \\ & + N_{nn\delta\delta\delta}n_A^2\delta_A^3 + N_{\delta vv}\delta_A v^2 + N_{va}v_a \end{aligned} \tag{3}$$

where:
$\mathbf{x}^T = (\delta_A\ n_A\ x\ u\ y\ v\ z\ w\ \phi\ p\ \theta\ q\ \psi\ r)$
$\mathbf{u}^T = (\delta_D\ n_D)$
$\mathbf{w}^T = (u_c\ v_c\ u_a\ v_a\ \zeta_x\ \zeta_y)$

Equations 1 to 3 can be arranged in the state matrix vector form:

$$\dot{\mathbf{x}}(t) = \mathbf{F}(t)\mathbf{x}(t) + \mathbf{G}_c(t)\mathbf{u}(t) + \mathbf{G}_D(t)\mathbf{w}(t) \tag{4}$$

The corresponding discrete solution is:

$$\dot{\mathbf{x}}((k+1)T) = \mathbf{A}(T,kT)\mathbf{x}(kT) + \mathbf{B}(T,kT)\mathbf{u}(kT) + \mathbf{C}(T,kT)\mathbf{w}(kT) \tag{5}$$

Whilst consideration of all 6 degrees of freedom within the neural model should be possible, this study concentrated only on the three, these being

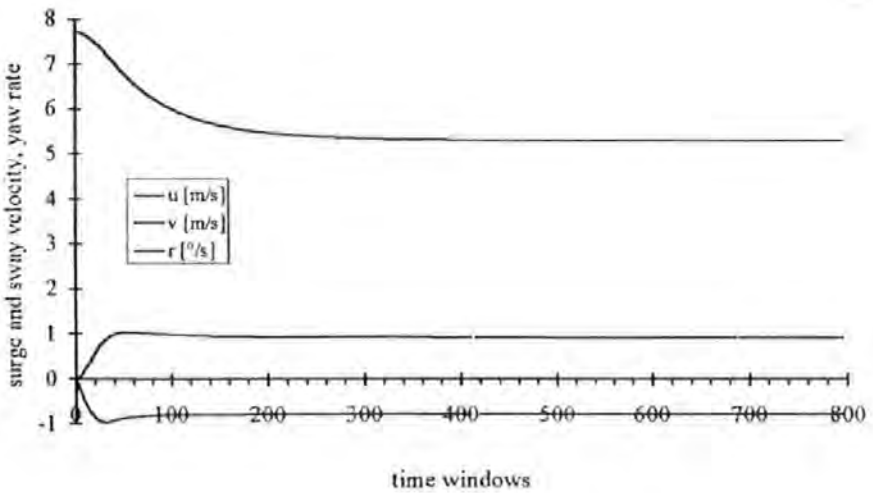


Figure 1: Settling Times

surge, sway and yaw. The vessel's parameters used in this simulation are given below (see table 1), and were based on previous work by Morse and Price which was directed towards the established Mariner Hull [5]. It has previously been demonstrated that artificial neural networks (ANNs) have the capability to model highly complex plants [9] by generating a series of connections between neurons within the network itself, each having a weighting value which may be scaled during training to replicate the relationships between the input/output data presented during this phase of development. By deriving this data from the real world environment, i.e. by measuring critical values, these networks can learn to emulate a wide range of different working conditions. When considering the application of a maritime vessel, the non-linear time-invariant dynamic characteristics are particularly difficult to model and realistic data may only be obtained by carrying out a carefully selected range of labourious manoeuvres undertaken at various forward speeds. In addition, the precise relationships between many of the features and characteristics of these ships is often not fully understood.

An alternative approach is that of system identification whereby a pseudo-random binary sequence (PRBS) is applied to the real system and the input/output measurements used to develop a model. However, by consideration of only crucial inputs/output relationships, then training data for a neural network may also produced instead. Such a model could fully described the ship dynamics with limited data quantities, thereby significantly simplifying the model generation process. In contrast to the extensive sea

Inputs			Outputs		
time elapsed	rudder angle	engine speed	forward speed	lateral speed	turning rate
t	δ	n	u	v	r

Table 2: Network Inputs and Outputs

trials necessary with the conventional approaches to model generation, the neural network can interpolate relationships given only limited information, assuming that the data provided describes the key aspects for the dynamic characteristics. For this application, the important parameters to be considered during measurement were identified as being rudder angle, and engine speed (input parameter) which cause velocity changes in the surge, sway and yaw directions (u, v, r) (output parameter). Whilst steady-state response is of limited value to the model generation process, the transient behaviour of the vessel is essential and consideration must be given to the time elapsed since the last rudder change as this will singificantly effect the learning requirements. Figure 1 indicates the various time periods required for the transient vessel response to settle down.

Utilising an acceptable error of $\pm 1\%$, it is possible to determine from the data the required values of settling time (table 3). From this information it may be deduced that if the time considered is greater than the time to reach steady-state, then the response has already reached steady-state, otherwise the response remains in the transient period and dynamic operation of the artificial neural network is required.

2.2 Structure of the Neural Network

It is a pre-requisite that the variables to be investigated are considered before commencing the initial design stages of the network's structure. Table 2 contains the required input and output variables considered to be the minimum requirement for this application.

To learn the transient behaviour, it is necessary to determine the time elapsed since the last rudder change as an additional further input to those previous discussed. The quality of results obtained from a two hidden layer network proved unsatisfactory with transients features being filtered out during the learning phase. Following further heuristic experimentation, the required network structure was identified as being a 3-6-6-6-3 which corresponds to input and output layers of 3 neurons in each, separated by 3 hidden layers of 6 neurons. Performance levels were significantly increased

$d[^\circ]$	u	t[s]	v	t[s]	yaw rate	t[s]
0	7.588	275	0.278	410	-0.186	405
10	6.639	270	0.734	220	-0.534	180
-10	6.818	270	-0.669	230	0.476	230
20	5.915	295	0.870	240	-0.681	185
-20	6.043	290	-0.836	235	0.641	240
30	5.308	290	0.930	240	-0.782	155
-30	5.408	285	-0.907	240	0.747	165

Table 3: Settling Times

with this network structure allowing correct learning of even the more subtle aspects of the training data provided, and avoiding local minima difficulties. In addition, is was discovered that with the final form of network structure, the error surface contains less troughs and a more constant learning was achieved.

3 Network Training

The learning method utilised for this study was the back propagation algorithm which is based upon the steepest gradient of descent minimisation technique and a Sigmoid transfer function with output limits between 0 and 1, i.e. $0 < y < 1$ where these values can only be reached with magnitudes of infinity. Therefore, the desired outputs required scaling to bring them within the numerical limits. Adjustment of the learning rate was preformed from a large initial value, by gradual decrements until the required termination level of learning was achieved which equated to the steady error characteristic.

Considering the training results in figure 2 for the vessel's forward speed, it is easily visible that a close comparison is present between the network response and the actual training reponse of the system. Clearly, there is close correlation between the two sets of responses with little discrepancy being apparent. Whilst the network closely follows that of the surge velocity training data, an improved level of performance is apparent with the responses for both sway velocity and also yaw rate when following defined increments of rudder angle set as 0°, -10°, $+10^\circ$, -20°, $+20^\circ$, -30° and $+30^\circ$. The evidence of successful nework learning is given in table 4.

During learning, approximately 4 million epochs were trained, the learning rate and the momentum being gradually changed. Starting off with a learning rate η of 0.3, and momentum a of 0.8, both have been reduced

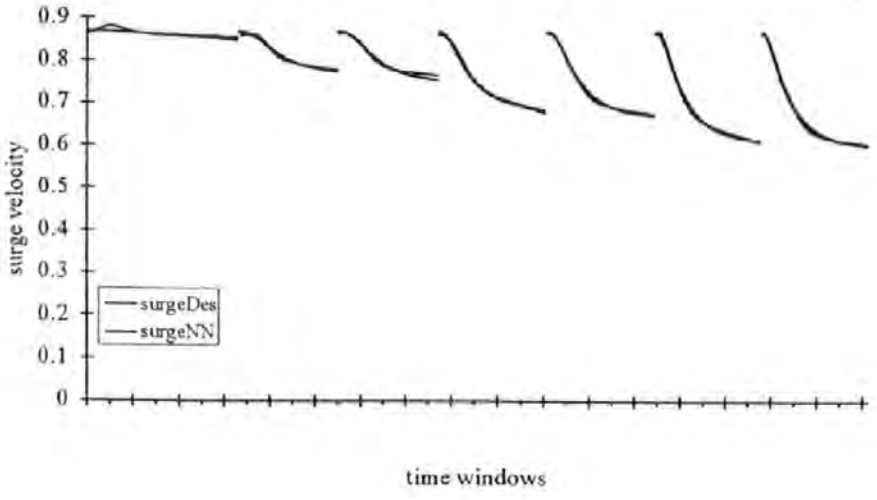


Figure 2: Surge Velocity Response

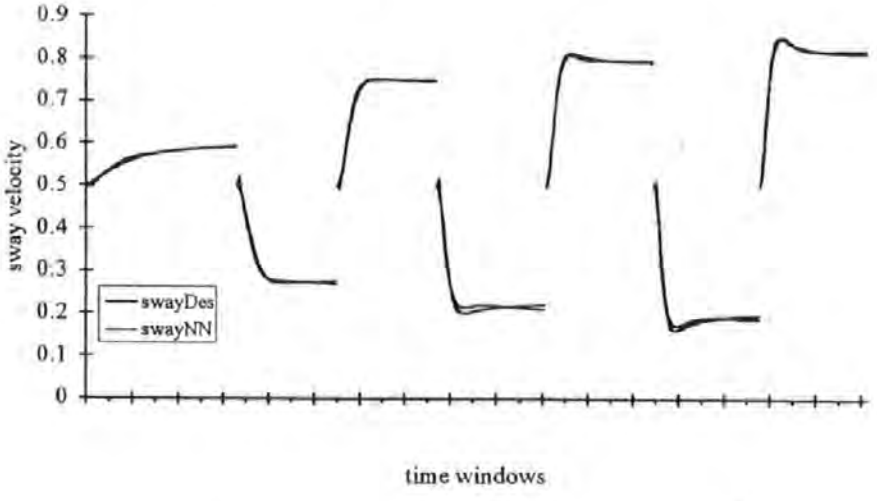


Figure 3: Sway Velocity Response

Parameter	RMS
Surge Velocity	0.097620
Sway Velocity	0.042320
Yaw Rate	0.168156

Table 4: Correlation of Network and Training Data for Ship Model

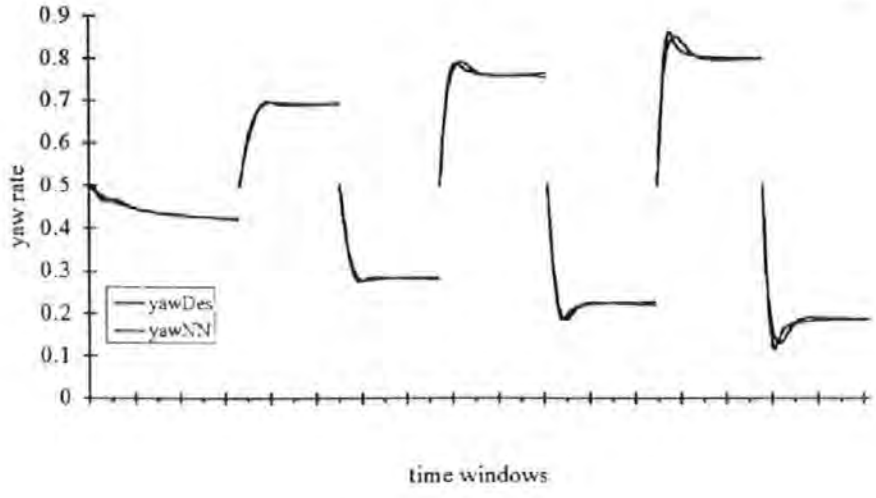


Figure 4: Yaw Rate Response

in three steps to final values of 0.05 and 0.4 respectively. The training file contained data for the 7 different rudder settings, these 0° , $\pm 10°$, $\pm 10°$ and $\pm 10°$. For each rudder angle δ, between 55 and 60 vectors were stored representing the time transient information.

4 Novel Intelligent Control Principles

4.1 Structure of the Fuzzy Controller

The real world inputs of heading error, and rate of change of heading error, must be converted to fuzzy values for use within the Fuzzy Logic Controller (FLC). This process may be achieved using the fuzzy input windows described (figure 5).

The form of FLC used for this study employs two rulebases, one for the gain of counter rudder (derivative term), and the other for rudder ratio (proportional term). Having established the function of the two rulebases, it is important to realise that vessel performance will only be satisfactory if the contents of each rulebase is correct. An initial information may be generated from consultations with experienced helmsmen. In practice, the rulebases therefore contain the helmsmen knowledge. This experience is non-linear in nature, and when utilised for the purposes of ship control, creates a significant increase in operating performance to be obtained. The output from the fuzzy controller is produced by a process of defuzzifying identified

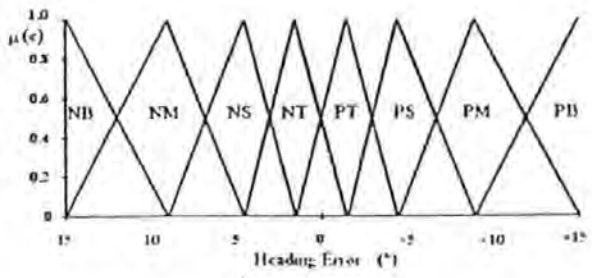


Figure 5: Input Window for Heading Error

fuzzy sets in a single output window of similar format to that used for input fuzzification. A deterministic output is produced by utilising a centre of area method.

Advantages of using the fuzzy technique include the ability to merge together several experiences, thereby producing a composite result which was not previously know, but capable of improved performance when compared to the known experience. For example, it is possible to ask the helmsman to determine the amount of rudder required for a small heading error, and for a large one. In reality, if the heading error was of medium value, then a conventional expert system would have to provide the closest known response (either that for a small or a large heading error). The fuzzy system can take both pieces of knowledge and combine them so that the result is somewhere in the middle. This form of reasoning, being inspired by natural processes, is much more human-like in approach.

During sea trials this novel non-linear design of fuzzy logic controller (FLC) improved performance by approximately 50% which constitutes a significant saving in energy usage over a long voyage.

4.2 Self-Organising Operation

In order to ensure that the rulebases are capable of correct operation, two performance indices are employed. Observations of the vessel performance are passed to the performance index in terms of the fuzzified heading error and fuzzified rate of change of heading error. Based on these observations, the performance index can enforce any required modifications to each rulebase so that it continually changes to match variations in the operating environment. The ability of the Self-Organising Controller (SOC) to achieve the correct modifications to the rulebases is fundamental to the successful operation and is therefore dependent upon the content of the performance index utilised.

This algorithm utilised combines the two tasks of control and learning.

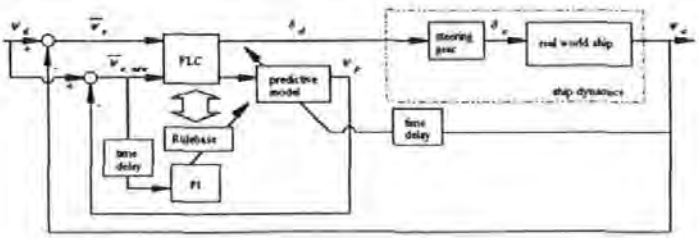


Figure 6: Predictive SOFLC

Learning must be achieved by observing the operating environment and the controller's effect within that environment. By utilising this information, changes in the fuzzy rulebase were determined in order that future activations of those rules will generate an improved level of performance. Having predetermined which observations are acceptable, and which are not, this information may be stored in a matrix format called a performance index (PI). If the observations of the operating environment indicate that the process is maintaining a satisfactory level of performance then no rule alterations will be required. Conversely, as the performance level deteriorates, then the magnitude of the rule changes increases.

4.3 Performance Index Development

The magnitude of each element in the respective PIs was determined based upon experience, observations and an understanding of the nature of the learning required and as such may be considered to be application dependant. Poor performances are penalised by large magnitude whilst desirable performance levels generate no modification.

5 Towards Predictive Ship Control

Having established the viability of both the modelling techniques using neural network, and of the intelligent self-organising fuzzy logic controller, they may then be combined together to form a predictive composite system as previously defined.

Obviously the learning of the controller has be amended to take into account the anticipatory aspects of the new methodology being employed. The block diagram (figure 6) demonstrates the principles of this new Predictive Self-Organising Fuzzy Logic Controller (PSOFLC). The control output is only passed to the process if the predicted performance of the process fol-

lowing the application of this control action indicates a satisfactory level or the time to check and alter the controller's performance has elapsed. If the output is poor, then the rulebase is adjusted to meet the requirements.

Similarly, the neural model may be employed on-line and than be used as the predictive model, for anticipating dynamic changes in the 'real' system, within the described PSOFLC. A method is therefore required to establish the criteria for learning data generation during the voyage itself. By this means, the model can adapt itself to behave exactly as the vessel will when subjected to the on-going disturbance effects and dynamic variations, e.g. mass loading, forward speed, etc.

6 Conclusions

It has been demonstrated by this study that it is possible to simulate complex plant behaviour in an innovative manner by utilising a neural network. The advantage of employing a simulation using this technique is that it becomes possible to overcome the problems associated with formulating the relationship between the features to be investigated. This can be achieved by the neural network, thus allowing the designer to concentrate on alternative aspects of the design. During execution, the neural solution therefore allows for extension to far more complex mathematical models without incurring a notable deterioration of the process time.

The principles of intelligent learning using a self-organising form of fuzzy logic controller have been validated by preliminary full scale sea trials. Aspects of this SOC remain relevant for the predictive controller, however certain areas of the learning process may require re-engineering to maximise the potential of utilising the new learning methodology.

This novel predictive controller is therefore a composite form of two established areas of innovation. Such a new control technique will allow controllers to avoid unsatisfactory control performance and therefore will contribute significantly to improved efficiency and safety whilst at sea.

References

[1] R.S. Burns. Application of the riccati equation in control, systems and signals. In *Proc. of the Int. Federation of Automatic Control*, Como, Italy, 26–28th June 1989.

[2] R.S. Burns. The design, development and implementation of an optimal guidance system for ships in confined waters. In *Ninth Ship Control Systems Symposium*, volume 3, pages 3.386–3.404, Bethesda, USA, 1990.

[3] M. Endo, J. van Amerongen, and A.W.P. Bakkers. Applicability of neural networks to ship steering. In *IFAC Workshop Control Applications in Marine Systems*, pages 221–232, Lyngby, 1989.

[4] N.R. Fairbrain and M. J. Grimble. Course-keeping and course-changing. In *Ninth Ship Control Systems Symposium*, volume 3, pages 3.311–3.335, Bethesda, USA, 1990.

[5] R.V. Morse and D. Price. Manoeuvring characteristics of the mariner type ship (uss compas island) in calm seas. Technical report, Sperry Polaris Management, Sperry Gyroscope Company, New York, 1961.

[6] N. Mort. *Autopilot Design for Surface Ship Steering Using Self-Tuning Controller Algorithms*. Phd thesis, University of Sheffield, UK, 1983.

[7] M.N. Polkinghorne, R.S. Burns, and G.N. Roberts. *Applications of Artificial Intelligence to Technological and Business Processes.*, volume 2 of *Green Series*, chapter A Novel Design of Self-Organising Fuzzy Logic Autopilot, pages 89–97. University of Plymouth, Drake Circus, Plymouth, devon PL4 8AA, UK, 1995.

[8] M.N. Polkinghorne, R.S. Burns, and G.N. Roberts. Operational performance of an initial design of self-organising fuzzy logic autopilot. In *Proc. IEE Conference Control 96*, Exeter, UK, 1996.

[9] R. Richter, R.S. Burns, and M.N. Polkinghorne. Application of an artificial neural network to model complex plant behaviour. In *IFAC'95 - Workshop on Motion Control*, Munich – Germany, 9–11 October 1995.

[10] J. van Amerongen and Udinke ten Cate A.J. Model reference adaptive autopilots for ships. *Automatica*, 11:441–449, 1975.

[11] J. van Amerongen, H.R. van Nauta Lemka, and J.C.T. van der Veen. An autopilot for ships designed with fuzzy sets. In *Proc. IFAC Conference on Digital Computer Applications to Process Control*, pages 479–487, The Hague, 1977.

P. Robinson, P. Nurse, S. Roberts, R. Richter, G. Bugman, and R. S. Burns.

Single site myoelectric control of a complex robotic hand.

In J. O. Gray, editor, *Workshop on Advanced Robotics and Intelligent Machines*, paper number 8, University of Salford, Dept. of Electronic and Electrical Engineering, Salford, M5 4WT, UK, 25–26 March 1997, ISSN 1363-2698.

Single Site Myoelectric Control of a Complex Robot Hand

by

Paul Robinson[1], Peter Nurse, Steve Roberts, Ralph Richter, Guido Bugmann[2] & Roland Burns

Myoelectric control methods have been used in commercial prosthetic hands for about twenty years. Lower arm muscle action results in the generation of electric potentials which may be detected at the skin surface. Commercial prosthetic hands use these potentials to activate a binary control action: hand open/hand closed. There is no control over the force exerted by the hand. This is set at some 'average' value thought to be most appropriate for a wide range of circumstances. Consequently available commercial hand are of only limited practical use. This paper describes an improved myolectric control system capable of controlling a multiple degree of freedom (DOF) robotic hand. Spectral analysis of a single site myolectric signal is combined with a neural network to provide up to seven control signals. Tests on a range of volunteers have validated the robustness of the system. As a man-machine interface (MMI) the method is shown to have many potential applications including a novel means of robot programming and as an intuitive interface to VR environments.

Keywords: Myoelectric; robot; prosthetic; control

1. Introduction

It is common for lower arm amputees to retain the muscle structure of the lower arm, Fig 1. Commercial prosthetic hands detect the electrical activity generated by the action of these muscles and use the signal to control the operation of the hand (Radix et al, 1996). The control system consists of a pair of electrodes attached to the skin surface above both the flexor and extensor muscles just below the elbow. Stump muscle action results in the generation of electric potentials, the magnitude of which is detected by surface electrodes and used to control the opening and closing of the prosthetic hand. When a detected signal exceeds a specified threshold value the hand will open. Closing the hand requires the signal from the second electrode to exceed the specified 'close threshold'. Variations on this simple two-site-two-state' control method have been developed but are not commonly used (Roberts et al. 1995). Existing NHS prosthetic hands are therefore severely limited in their ability to emulate the behaviour of a real hand. Only a single DOF pincher movement is available. It is widely accepted that any improvement in the design of prosthetic hands is dependent upon advances in myoelectric control methods.

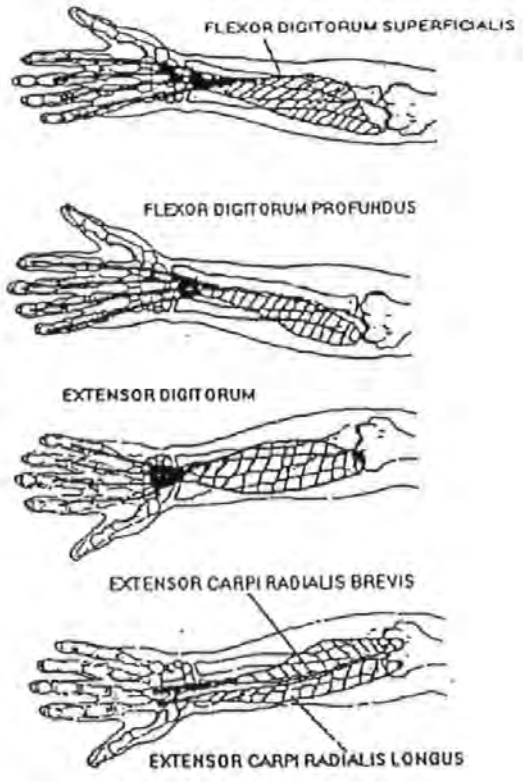


Figure1. Lower arm muscles contributing to ring finger motion.

Fundamental to the success of myolectric control is the ability to reliably detect specified muscle actions , i.e. hand movements in an able bodied person and the equivalent muscle action in an amputee, in the presence of noise. Each hand movement is the result of a

[1]School of Electronic, Communication & Electrical Engineering. School of Mechanical, Materials & Manufacturing Engineering, [2]School of Computing, University of Plymouth, Plymouth, PL48AA, UK. Email. probinson@plymouth.ac.uk

complex combination of muscle actions. To complicate matters further each individual produces slightly different muscles actions for the same hand movement. The electrical signal received at the upper arm surface electrode is a combination of time varying signals generated by several muscles. In order to reach the electrode these signals have propagated through different thickness' of tissue proportional to the distance between the electrode and the relevant muscle. Thus the received signal is a combination of time varying potentials, generated by a number of muscle actions, which have been attenuated and filtered according to their distance from the pick-up electrode. In addition the variables governing this process, e.g. thickness and quantity of arm bone and tissue, are different for each individual. It follows that any control system based upon the MEG (myoelectrogram) must be capable of discriminating between a wide variety of signals generated by similar muscle actions.

2. Experimental Procedure

A single pair of electrodes, situated about 1cm apart are attached below the elbow to the upper forearm of a range of able bodied volunteers. In addition a reference electrode is securely attached to the upper-arm using proprietary tape. Motion artefact, a serious source of signal degradation, is reduced to a minimum by ensuring a good contact surface between skin and electrode. Consideration of a range of electrodes resulted in the choice of Liberty Mutual MYO115 EMG research electrodes. Pick-up from the two electrodes is fed to a variable gain differential amplifier. The MYO115 includes on-board filtering giving a claimed 3dB response of 90-500Hz and a gain which may be customer specified between approximately 500 to 6000. Differential myoelectric signals of interest typically have magnitudes from a few microvolts up to a few millivolts. Without the filter these signals would tend to be swamped by induced 50 Hz noise created by nearby mains electrical equipment.

The amplified, differential myoelectric signal is fed to a PC using a PCI718 Advancetech PC interface card. The signals, sampled at a rate of 1 kHz, are captured from the arm surface in one second bursts, i.e. 1000 samples. A normalisation process ensures an RMS value of 1 volt. The resulting signal is applied to six filters operating within the range 0 Hz to 300 Hz, viz. <50Hz, 50-99 Hz, 100-149 Hz, 150-199 Hz, 200-249 Hz, and >250 Hz. Initial experiments used the LABTEC notebook and DaDisp signal processing package to examine the raw data. Subsequently a dedicated software package was developed which produced the filtering operations described above and calculated the RMS value of each filter output. This RMS value provides a measure of the power spectrum within each frequency band. Filters one and two, i.e. <99 Hz, are clearly operating below the low frequency -3dB level of the MYO115 electrode. Notwithstanding this unusual 'double-filtering' procedure results obtained from the <100 Hz frequency band were found to be crucial in identifying specific hand movements. The myoelectric signals associated with many specific hand movements have a significant low frequency, i.e. <100 Hz, content. The double filtering effect may therefore be regarded as a crude form of spectrum averaging.

The Nassi Schneidermann chart of Figure 2 illustrates the complete process. A neural net is fed the RMS values of the six filter outputs and trained to recognise specific patterns. Outputs from the neural network are then associated with specified hand movements. These positions are actioned in software by a complex, virtual hand and the result exhibited on a PC screen.

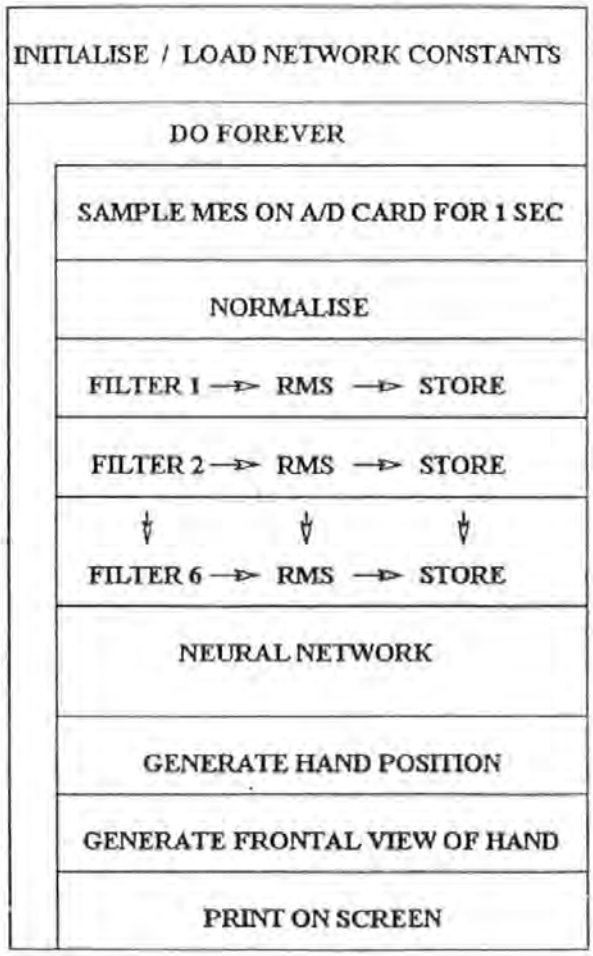


Figure 2 The Nassi Schneidermann Chart

A large number of experiments were performed in order to discover those hand/finger/wrist movements which could be most readily, and reliably, identified. Initial methods used the natural 'neural network' of the human brain. Two approaches were followed. In the first instance a spectrum analyser displayed the captured

myoelectric signals in real time. Observations seemed to show some correlation between specific hand/finger/wrist movements and the resulting spectra. These results were confirmed when the differential myoelectric signals produced by the hand movements were amplified and played through a loud speaker. After a short learning period it was discovered that specific hand/wrist/finger movements could be clearly identified by their audio signal. It was this discovery which convinced the authors that a simple neural network should be capable of being trained to identify individual hand movements.

3. The Neural Network

Neural networks, with their ability to learn and recognise relationships between patterns of inputs, are ideally suited to recognising complex myolectric signals. In this case hand/finger/wrist movements are known to result from muscle actions, but the pattern between a complex combination of input muscle signals (detected at a single site) and output movements remains unclear. By feeding the prepared data, from the six filters, to a neural net, a transfer function and hence the causal relationship, can be learned. The neural net performs as an identifier of different input signals

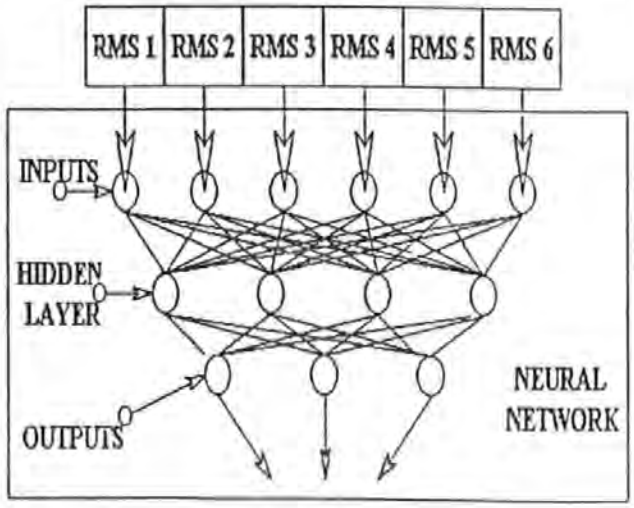


Figure 3 - The Neural Network

Empirical methods resulted in the choice of a back-propagation learning algorithm. This widely used algorithm learns from errors and is a form of the gradient minimisation problem. A simple neural network, Figure 3, consisting of six inputs, a hidden layer of four neurons and three outputs, is used to identify the hand action.

The neural network provides three binary output signals providing a potential of seven control signals: the relaxed condition, i.e. 000, is not considered to be a control signal. Training was done using a modified backpropagation algorithm with added momentum. This momentum helps to prevent the neural net becoming trapped in local minima of the error surface (Lau, 1992)

The network was trained with four training sets, representing four different moves. Each move was repeated four times to allow for variations. Initial experiments used the three neurons of the output layer to action three, single movements of the virtual hand, viz. a logical one at output one drives the hand to a specified position and ditto for outputs two and three. Clearly, however, the control system is capable of executing seven movements, i.e. the binary output range of the neural network.

For simplicity, and to prove the technology, only one action at a time is identified. The output of each neuron in the output layer lies between 0 and 1 due to the implemented transfer function. The Sigmoid function, i.e. $y=1/[1+\exp(-x)]$ was used for all neurons to compute the firing threshold of the neuron.

4. Results

Myoelectric signal spectra obtained from movement of the ring finger, i.e. the third finger, is shown in Figure 4. Scans S1 to S4 illustrate the results for the same individual repeating the ring finger movement four times. Care is taken to try to ensure that each action is identical to the previous movement.

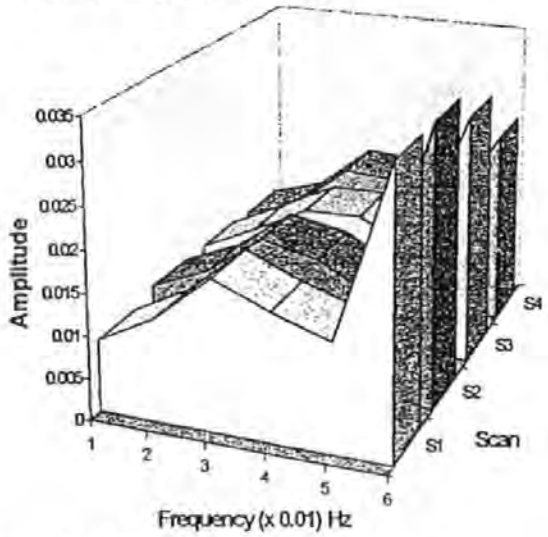


Figure 4 - Ring finger movements

In practice it is extremely difficult to ensure range of movement, speed of action and force applied remain constant. Differences between each set of experimental

data are clearly visible. However the general 'shape' of the spectra resulting from this single finger movement remain fairly constant.

Figure 5 shows typical spectra from four specific positions of the hand/finger/wrist..

S1. Relaxed hand, i.e. fingers and thumb held out straight.

S2. The ring finger bent so that the tip touches the palm.

S3. Hand relaxed, wrist bent inwards towards the palm.

S4. Little finger bent inwards to touch the palm.

The results of Figure 5 were obtained from a single individual. Different people produce unique 'spectral identities' but the overall relative patterns, for similar movements/positions, remain fairly consistent.

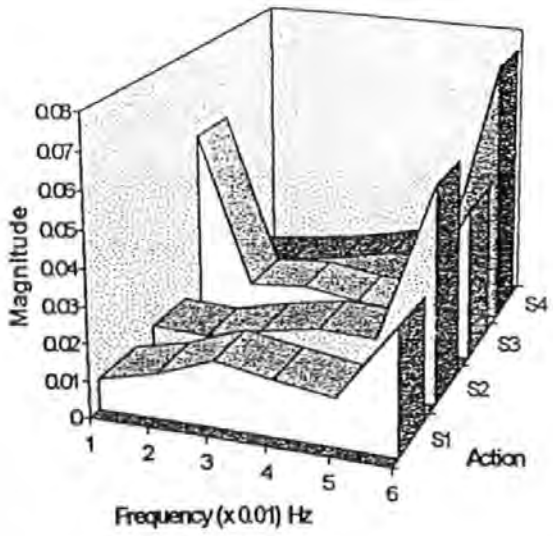
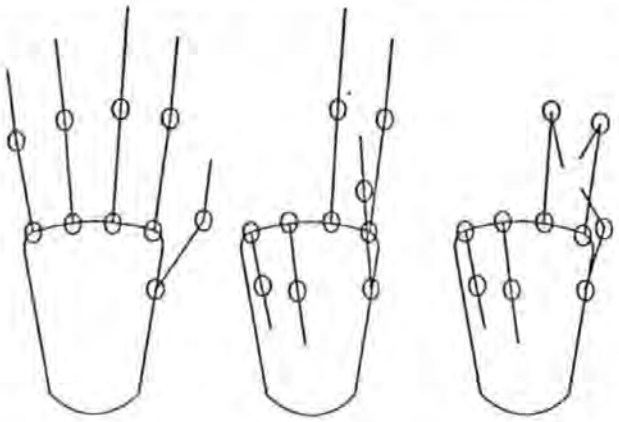


Figure 5 - Spectra from four hand positions

The above movements of the hand and/or wrist were identified by the neural network, and used to control the action of a virtual robot hand, Figure 6. In this case the three simple positions mentioned earlier are illustrated, i.e. the relaxed position and the two finger parallel and chuck grasps. These position were identified as being useful to a user of prosthetic hands. The positions adopted by the virtual hand do not necessarily replicate the position of the volunteers' control hand. The intention is not that the robot hand will exactly replicate the movements of the biological hand. The aim is for the user to be able to reliably control a complex prosthetic hand using simple muscle movements of the upper forearm.

From a practical viewpoint it is important that the exertion necessary for the muscle control does not become so excessive as to render the user exhausted after a short period of use. Simplicity of action is therefore of crucial practical importance. In this regard the amputee has one advantage over the able bodied person. In order to exert force the able bodied user must grasp an object or press the fingers against each other or



a. Hand relaxed b. Parallel grasp c. Chuck grasp

Figure 6. Useful hand positions

into the palm of the hand. Because of the way the forearm muscles are terminated in many amputees the ability of the muscles to emulate the action of varying force is retained. Crucially this ability is maintained without the need of an external opposing force. The magnitude of the myoelectric signal is proportional to the force exerted. The potential exists for this effect to d add a further dimension to the control system. It is intended in the near future that the magnitude the signal will be used to provide a greater combination of output signals and hence a more sophisticated control action.

5. Further Developments

The above results demonstrate that the frequency components of the MEG (myoelectrogram) signal from a single probe carry enough information to discriminate several hand positions. An advantage of the spectral approach is the robustness of the system against electrode efficacy fluctuations. Work towards a more refined analysis is pursued. For instance, at present, binary commands are extracted from the MEG, e.g. 'open hand', 'chuck grasp' etc. For a more natural operation of a prosthetic hand, it is necessary to also extract force information from the signal. It is well known that the magnitude of a MEG signal is proportional to the force applied. The authors believe that force information may also be encoded in the frequency spectrum of the signal. Success along these lines would lead to a force-sensitive system robust against fluctuations in probe efficacy.

At present, the signal is sampled during one second, its amplitude is normalised, its energy is computed in six

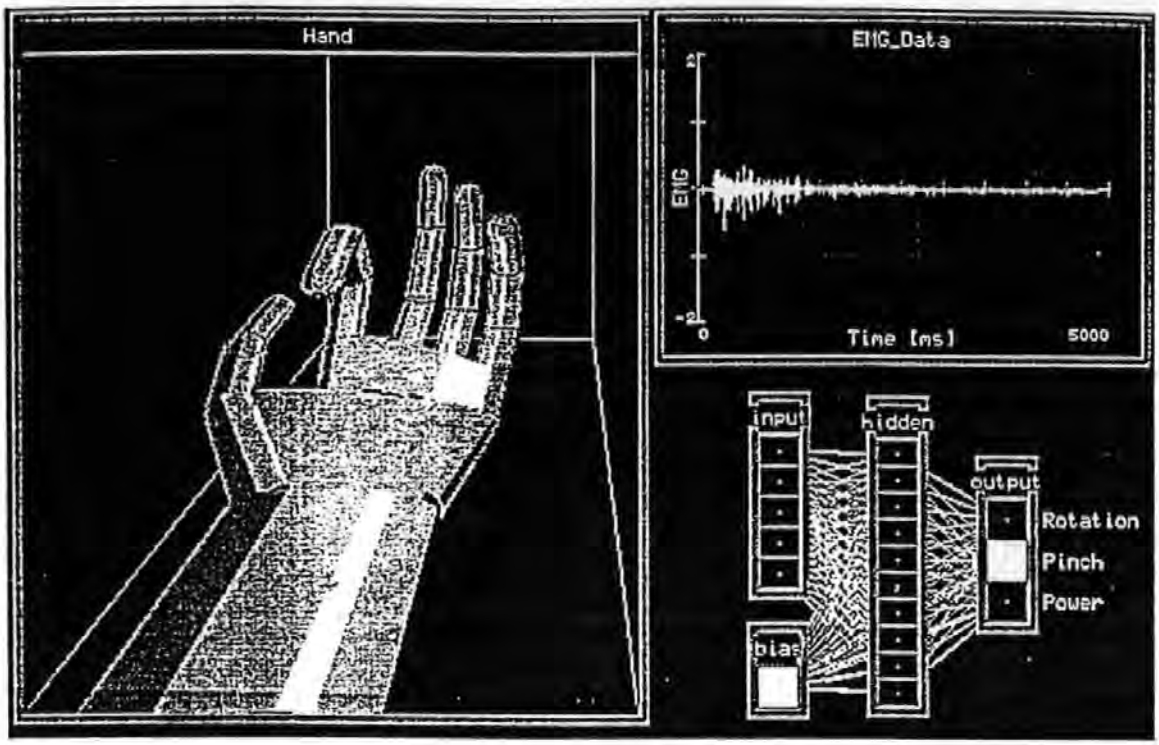


Figure 7 - Advanced virtual hand, associated EMG signal and neural network

frequency bands, then the results fed to the neural network which evaluates the hand position encoded by the signal. This procedure causes a relatively long reaction time. Shortening the reaction time to less than 100ms is desirable for a practical system. In order to achieve this objective, future developments are aimed at normalising the amplitude using hardware filters and a new low power analogue neural network chip developed at the University of Plymouth [Coue and Wilson, 1996a, 1996b].

Finally, the conception of a training procedure is to be investigated. To train the neural network it is necessary to know which intended movement corresponds to the recorded signals. One solution may involve a more complex virtual hand which adopts a sequence of positions, Figure 7, that the patient would be asked to copy with his (missing) hand. There is also the possibility that prosthetic users could chose which positions of the hand were of most interest. An office worker, for example, would be likely to chose a different set of actions to a manual worker. The neural network would then be trained to reproduce the movements of the virtual hand, using the signals recorded on the subject. Once the patient is satisfied that the neural network has learned to produce the correct movements, the neural net can be transferred onto the real prosthetic hand.

Figure 7, shows the system under development for training a neural network to reproduce the movements intended by the patient. The window to the left displays a moving virtual hand. The graph on the top right shows an example of a raw MEG signal recorded during the following sequence of positions: 1. Rest 2. Thumb / index finger in a pinched grip, 3. Thumb / index finger parted forcefully and finally 4. Rest. In the future is hoped that this system will operate in real time in response to the user lower arm muscle movements.

6. Conclusions

Myoelectric signals, obtained form a single site on the lower arm, are capable of controlling a complex robot hand. The robustness of the method has been successfully demonstrated using a number of able bodied volunteers. It was discovered that a trained neural network will often operate satisfactorily for a range of different individuals. The present system has been implemented using only four separate control actions. However seven separate actions will be implemented in the near future. It is also feasible to arrange for a 'library' of different hand movements to be stored and recalled by the user as and when required. Each of these libraries would contain a combination of up to seven specialised hand positions and/or force values.

Time delays experienced with the prototype system are unacceptable for practical applications. Implementation of dedicated analogue filters and neural networks, developed at the University of Plymouth, are expected to reduce these delays to approximately 100mS. This work is included in the next phase of the project.

Practical implementation of these control methods for disabled users is dependent upon the development of an improved NHS prosthetic hand. The ideal hand would be physically attractive, inexpensive, lightweight, include multiple finger joints and force control. A joint project with a major prosthetic manufacturer to construct such a hand is presently under evaluation. However it is unlikely that such a device will be available in the near future.

Other possible application areas, presently under investigation, include a novel method of industrial robot programming, robot teleoperation (NASA is evaluating myoelectric control methods for use with the shuttle robot arm) and as a biologically intuitive interface for VR environments. It is possible that in the near future myoelectric control will become an important MMI (man machine interface) technology enabling natural body actions to be used to control complex hardware and software systems.

Acknowledgements

The authors wish to thank the University of Plymouth (DevR), the BRA (British Robot Association) and the Tempus award scheme for supporting this work.

References

1. Radix, C., Roberts, S.M., Robinson, P., Nurse, P., Grosch, P., and Burns, R.S. 1996 'Tele-prosthetic systems for paraplegics', Proceedings 4th International Workshop on Advanced Robotics & Intelligent Machines, Salford.

2. Roberts, S.M., Nurse,P., Burns, R.S., and Robinson,P., 1995. 'Myoelectric prosthetic upper-limbs, past and present: a case for further development' Proceedings of Medimec 95, University of Bristol, Sept. 6-9.

3. Farry, K.A., Walker, I.D. and Baraniuk, R.G., 1996 'Myoelectric Teleoperation of a Complex Robot Hand' IEEE transactions on Robotics & Automation, Vol.12, No.5, pp. 775-788.

4. Lau, C., 1992. 'Neural Networks – Theoretical Foundations and Analysis', IEEE Press, ISBN 0-87942-280-7

5. Coue D. and Wilson G. (1996a) "CMOS Subthreshold-Mode I/V Converter for Analogue Neural Network Applications" Electronics Letters, 32, pp. 990-991

6. Coue D. and Wilson G. (1996b) "A 4-Quadrant Subthreshold Mode Multiplier for Analog Neural Network Applications", IEEE Transactions in Neural Networks, 7, pp. 1212-1219.