DEVELOPMENT OF SUBSEA ROBOT NOMAD WITH A MICRO COMPUTER BASED INTELLIGENT CONTROL SYSTEM

ZHONGQUN LU

001311

DEVELOPMENT OF SUBSEA ROBOT NOMAD WITH A MICRO COMPUTER BASED INTELLIGENT CONTROL SYSTEM

By

ZHONGQUN LU, P.ENG.

B.Eng in Electrical Engineering
M.Eng in Electrical Engineering

A Thesis Submitted To The School Of Graduate Studies
In Partial Fulfillment Of The Requirements
For The Degree Of PhD In Electrical Engineering

Faculty Of Engineering And Applied Science
Memorial University Of Newfoundland

August, 1996

St.John's Newfoundland Canada

0-612-36207-8

ABSTRACT

The goal of this PhD project was to design and develop a small inexpensive subsea robot with a micro computer based intelligent control system. The robot developed is called NOMAD. It could be the key element in a Distributed Marine Observation System(DMOS). Most Engineering PhDs are research oriented; this one has a design focus.

NOMAD uses an air/water ballast tank instead of a battery/motor system to drive itself vertically. To facilitate mission requirements, great efforts were made to develop a high performance onboard micro computer based control system. To deal with the uncertainty and nonlinearity of the robot model, investigations were conducted to check the potential of strategies based on Neural Networks and Fuzzy Logic. A Real Time Kernel software and an onboard micro computer with a Z180 CPU were used to implement a Fuzzy Variable Structure Switching (FVSS) control scheme and a multiple-task, multiple-layered control structure.

The design and development process for NOMAD are detailed in this thesis. The results of digital simulation, theoretical analysis and typical data recorded from tests in a deep water tank on the robot are presented. Successful tests and good agreement between data and analysis indicate great potential for industrial application of the technologies developed in this project.

ACKNOWLEDGMENTS

The author is greatly indebted to my supervisors, Dr. M. J. Hinchey and Professor D. A. Friis, for their excellent guidance, support and understanding.

Special thanks are also due to the Technical Staff in the Electronics Laboratory and in the Center for Computer Aided Engineering (CAE), who made their services readily available throughout the hardware and software development phase of this project, and to the staff in the University's Technical Services, who gave great help during the construction and testing phase of the project.

The author gratefully acknowledges financial support from the author's supervisor, Dr. M. J. Hinchey, the Faculty of Engineering and Applied Science and the School of Graduate Studies of Memorial University of Newfoundland.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1 INTRODUCTION

With the significant increase of human activities on and in the sea, long term observation of the marine environment is becoming more and more important in order to control the level of water pollution, to keep the ecological balance, and to develop offshore industry (Busby and Vadus, 1990).

At offshore oil well sites, particulate drilling wastes are discharged during exploration and production drilling operations. In addition large volumes of water along with oil are released over the lifetime of a field. Sediment on the sea floor at different locations needs to be picked up as an indication of pollution. Researchers in the Hibernia Management and Development Company Ltd. (HMDC) indicate that the water body from the benthic boundary layer, where drilling wastes concentrate, should be sampled or monitored over a long period of time (http://www.hiberniabank.com/, 1995). The results of chemical analysis from these samples of sediments and waters could be used to predict the impact on the marine ecology.

Near fish farms, water temperature, tidal currents, water turbidity and levels of dissolved oxygen at different depths are quite closely related to ecological impact on marine species. Research indicates that high rates of organic matter into sediments result in low levels of dissolved oxygen there and this together with anaerobic metabolism by sulfate reducing bacteria lead to reduced numbers of species and biomass of benthic fauna (http://www.hiberniabank.com/, 1995).

To oceanographers, the pH, turbidity, velocity of sound, disturbance stress at different depths are all essential to their research.

Military powers pay great attention to submarine forces. Noise detectors are often mounted at strategic sites on the seabed locations to find and record the noise from propellers of passing submarines.

§ 1.1 Concept of the Distributed Marine Observation System (DMOS)

So far many sophisticated underwater vehicles have been developed, such as the Autonomous Benthic Explorer (ABE) developed by Woods Hole Oceanographic Institution in the USA (Yoerger, Bradley and Walden, 1991) and Odyssey developed by MIT (Leonard, J.J. 1995). Usually these are equipped with Long Base Line or GPS navigation systems, and sonar scanners, and use batteries as their main power source. They can be launched from a ship, or from a near-by harbor from which they can head to the working sites (Nomoto and Hattori, 1986). Most of the developed subsea vehicles have a torpedo shape. When they are required to perform a descent or ascent maneuver, they have to do this by going into a helical trajectory as shown in Figure 1.1.1. This consumes a lot of energy. Obviously, it is also not safe and cost effective to leave such a sophisticated subsea vehicle at the working sites for a long term observation mission, which may take weeks or months, even though they may have that capability.

Therefore, a cost effective and energy saving system for missions with vertical movements is needed. This thesis describes a subsea robot known as

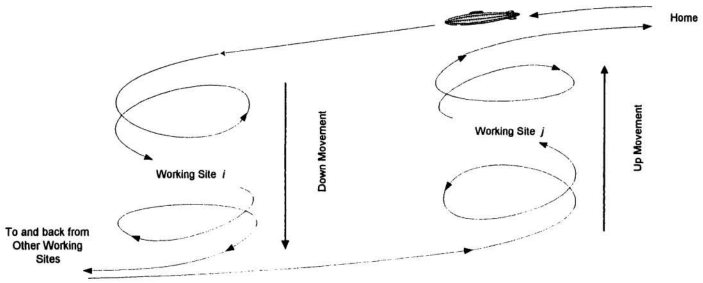


Figure 1.1.1 The Helical Trajectory of a Torpedo-like Subsea Robot Carrying Out Vertical Movement Missions

NOMAD with an onboard micro computer based intelligent controller which was developed for this purpose (Lu, Hinchey and Friis, 1994). It is suitable for long term observation and also for obtaining samples from the sea floor at different sites of interest, as shown in Figure 1.1.2.

For sediment and water sampling missions, a helicopter using its navigation system could carry NOMAD to the site of interest and drop it into the sea. When NOMAD dives into the water, its onboard micro computer would trigger its mission routine. When NOMAD arrives on the sea floor, a mechanical hand mounted on its bottom could grab a sample of the sediment. On the way back to the surface, NOMAD could carry out a water sampling mission. At a specified depth, a small chamber could be opened and then closed under the control of the onboard micro computer. In this case, the velocity of the robot would be set to rather slow, to allow the robot to get the sample of the water body at the precise specified depth. If it is not required to sample the water on the way back to the surface, NOMAD could return to the surface at the maximum speed.

For long term marine observation missions, the robot could be required to stay at a working site up to weeks and in some cases months. The robot would spend most of its time " sleeping" on the sea floor; then at pre-programmed times, it would move up and down vertically and record various physical data versus depth. After completing these measurements, the robot would go back to the sea floor to sleep. The recorded data would be stored in the memory area of the onboard computer. At preset intervals, the robot would surface and transfer data to a shore

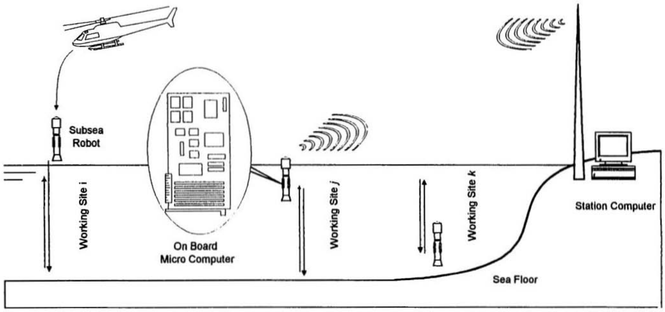


Figure 1.1.2 The Distributed Marine Observation System with the Subsea Robot NOMAD

station by radio signals. When the shore station has received these data, an acknowledgment signal would be sent to the robot which would allow the onboard computer to clear its memory area. Using radio communication, there would also be an opportunity here for the control personnel at the shore station to modify missions and pre-programmed routines stored in the onboard computer. When the robot has finished its mission, it would return to the water surface and emit a radio signal that a helicopter could home in on. Discharged color dye could also be used to help a helicopter pilot locate the robot and retrieve it.

During a mission, a number of NOMAD robots could be dropped from a helicopter at different spots of interest. They would form an observation and data collection network. Obviously, the robots used in this Distributed Marine Observation System (DMOS) would be the key components in this system. As noted, they would be able to stay at the work sites for a long time or scan the sea water vertically. They would also be able to use a mechanical hand to grasp sediment on the sea floor or chambers to sample water at different specified depths. The low cost and energy saving features of NOMAD make it quite different from traditional subsea robots.

A prototype of NOMAD with an onboard micro computer based intelligent control system has been designed, built and tested in this PhD project. Instead of using a battery, DC-motor, and propeller as the main propulsion system, NOMAD uses a simple water/air, ballast tank plus gravity and buoyancy to make itself move vertically up and down (Lu, Hinchey and Friis, 1994) (Hinchey and Muggeridge 1994). A comparison of NOMAD with traditional subsea robots is given in Table 1.1.1 (Allmendinger, 1990) .

Table 1.1.1 The Features of Subsea Robot NOMAD Compared with Traditional Ones

	Robot NOMAD	Traditional Under Water Vehicles
Main Power Supply	Compressed Air	Battery
Energy Consumption	Energy Consumption only at Start and End of Vertical Movement	Significant Energy Consumption throughout Vertical Movement
Maximum Speed of Vertical Movement	> 2m/s	Usually <1m/s
Ratio of Pay Load Mass to Vehicle Mass	Higher (1:3)	Lower (1:5 ~ 1:10)
Locating Method	Using the Navigation System on the Helicopter	Using the Navigation System on the Subsea Vehicle
Development Costs (Not Including Research Labor Costs)	Low (Single NOMAD < 10 K Can$)	High (Usually >100 K Can$)
Maintenance Expense During the Mission	10$/day	100-1000$/day
Suitability for the Task of Long Term Observation	Good	Not Worthy

§ 1.2 Overview of This PhD Project

NOMAD was designed by the author in the first half of 1994. Manufacture was started in June 1994. Assembly was performed by the author in February 1995. During this period, every effort was made to develop a good computer control system. Many investigations and studies were conducted for different control strategies, including: Proportional-Integral-Derivative (PID), Sliding Mode (Slotine, 1983 and 1984), Artificial Neural Network (Ishibuchi,1993), Fuzzy Logic (Self, 1990), and Nonlinear Variable Structure Control (DeYong and Polson, 1992). The author's studies focused on developing a controller with Neural Networks, Fuzzy Logic and with a Variable Structure. Most of author's studies were based on digital simulation, and some on digital-analog hybrid simulation. The tools used in the simulations were C^{++}, Togai Tilshell (Fuzzy Logic) and Matlab (Neural Networks). A suitable control strategy, Fuzzy Variable Structure Switching (FVSS), and its software framework, were developed and tested during these studies.

Since January 1995, the author has concentrated on electronic aspects such as: computer interfaces, communication hardware, sensors, conditioners and drivers to physically implement the control strategies. In the process of the development of NOMAD, a great number of technical difficulties had to be overcome such as: the water proofing of the onboard electronics, the automatic maintenance of constant working air pressure and the digital communication between the onboard micro computer and the station computer.

By May 1995 the subsea robot NOMAD with its micro computer based control system and FVSS control strategy was ready for testing in a water test.

Tests were carried out during June of 1995 to December of 1995, mainly for optimization of the control parameters. These tests on NOMAD were conducted in the Deep Tank of the Ocean Engineering Research Center of Memorial University of Newfoundland. The results of these tests indicated that the design and development of NOMAD was successful. The vast data recorded in the tests compared well with the results of theoretical studies and digital simulations. The tests showed that NOMAD was reliable and dependable . NOMAD was presented to the public on Canadian Engineering Day of 1996. It has also been demonstrated for the engineering students and now it is being enhanced for industrial applications.

§ 1.3 Thesis Outline

This thesis gives a detailed description of the design and development process for the subsea robot NOMAD. Every subsystem of NOMAD was designed and constructed in this project. However, the main emphasis was put on the control strategies and their computer implementation. The thesis is organized as follows.

Chapter One is the introductory chapter. A brief review of the previous work before this project is presented. Based on the mission requirements from applications in the offshore industry, a new concept of a " Distributed Marine Observation System" (DMOS) is put forward. The research interest is declared in this chapter. It is to physically develop a prototype of a single degree of freedom subsea robot with a micro computer based control system, that could be a critical part of the Distributed Marine Observation System. As noted above, this new subsea robot is called NOMAD.

Chapter Two gives the detailed technical requirements for the robot based on the nature of the mission, the procedures for its overall design and the flow chart for its development. The architecture of the subsea robot and the design of its main subsystems are detailed in the last section of this chapter.

Chapter Three studies the dynamic behaviors of the mechanical setup of the robot. The model of the robot is divided into two main portions: the model of the robot body and the model of the ballast tank. The latter is the main actuator of the robot. Both of the models are highly nonlinear. Linearization and simplification are applied to the original models. Block diagrams are established for the design of the computer control system which is detailed in the following chapters.

Chapter Four to Chapter Six study control strategies which are candidates for possible use in NOMAD. Chapter Four discusses two classical control strategies PID control and Sliding Mode Control. In Chapter Five, a new strategy - Neural Network Based Adaptive Control is put forward for model identification and depth control. This scheme consists of three main parts: the Neural Network for Model Identification (NNI), the Neural Network for Adjustment (NNA) and an Optimal Linear Reference (OLR) model. Results of digital simulation for this scheme are presented. Because there are some inherent defects in classical control strategies and the Neural Network Based Control System is computationally expensive, a more reliable and economical control scheme, Fuzzy Variable Structure Switching (FVSS) Control, is put forward and detailed in Chapter Six. This strategy causes the robot to undergo a self-excited oscillation known as a limit cycle. To enhance the system stability and diminish the amplitude of the limit cycle, hysteresis and phase lead compensation were introduced into this control strategy.

Digital simulation in the time domain and the describing function method of analysis in the frequency domain are applied to the FVSS controller with and without hysteresis and phase lead compensation. The study indicated that this control scheme is not only stable and robust but also energy saving. Because of this, it was then adapted to be physically installed on the robot.

Chapter Seven deals with the software architecture. Because of the complexity of the control process, a multiple-task, multiple-layer software architecture is developed using Dynamic C and a micro computer with a Z180 CPU. The main aspects of the software are discussed: its Real Time Kernel for the implementation of the software structure; the Interruption for the time management of computing capacity; Space Management of the memory capacity. The function configuration chart of the software is presented at the end of this chapter.

Chapter Eight presents the process of development and tests in practical engineering terms. As an important step in the development, a detailed digital simulation at the mechanical component level and a digital-analog hybrid simulation at the system management level were carried out to confirm the suitability of the control system. In these two simulations, identical parameters and computer hardware and software as that mounted on the robot were used. In this chapter, the setup and results of some dry-land tests are presented. At the end of this chapter, the process of the physical tests in the deep tank are also presented. The dynamic performance of the robot NOMAD are recorded under different test conditions. The results from theoretical analysis and calculation in previous chapters are confirmed by the tests and compared with the data records from the physical tests. A number of recorded charts in different conditions are presented in this chapter.

Chapter 9 gives conclusions from this project and illustrates the experience gained. It also gives recommendations for future work.

CHAPTER 2 OVERALL DESIGN AND DEVELOPMENT PROCEDURES

This chapter gives the detailed technical requirements based on the nature of the mission. The procedure of general system design for the subsea robot NOMAD and the development schedule are presented. The architecture of the subsea robot and the design of its main subsystems are detailed in the last section.

§ 2.1 Technical Requirements for the Subsea Robot NOMAD

As mentioned in the first chapter, the subsea robot NOMAD with the capability of carrying out different inspection missions will play a critical role in the Distributed Marine Observation System. To perform such missions the author concluded that the robot should have the following features:

(1) It should be energy saving during missions with large vertical movements.

(2) It should accurately indicate the present depth as the independent variable of the recorded marine physical data.

(3) It should be able to hover at a specified depth with an error not exceeding 0.5m.

(4) It should have an onboard computer which controls the lower level dynamics and higher level behaviors of the subsea robot.

(5) It should have reliable communication facilities which allows the robot to contact the shore station computer when necessary.

(6) It should be easy to maintain, easy to recharge or change the main energy

source (the gas bottle and auxiliary batteries), and easy to attach/detach different payloads.

(7) It should be light in weight and have high ratio of payload to robot self weight.

(8) It should have a low cost of development and operation.

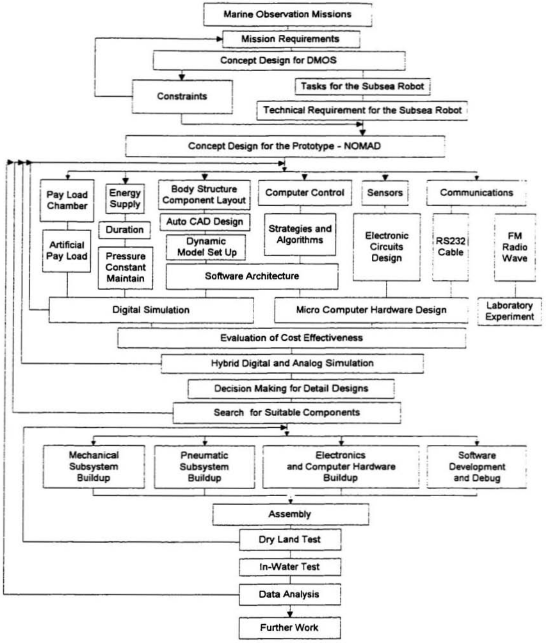
§ 2.2 The Flow Chart for the Development of the Subsea Robot NOMAD

In general, underwater vehicles are typically designed with a particular mission in mind. This leads to a set of requirements that define how various subsystems should be integrated. However, a system architecture for an underwater vehicle whose function is to act as a prototype for some new technologies presents unique problems to the system designer (Henrickson and Dzielski, 1995). Since there are no well-defined missions and detailed specifications, the subsea robot as a whole must be adaptable to systems that have yet to be designed. In a conventional design one has already narrowed the possible solution space into a small region. In the innovative design with new technologies, one has a wide region of possible solutions. The main point is that at the start we did not know whether there is a worthwhile working solution to bring the concept idea to a successful prototype.

A technology test bed is constantly being used in a system integration mode. In other words, experimental or developmental subsystems are constantly being integrated into the submersible. There is also a requirement to support the design and testing of the new system at various times during its development. The testing may occur in a pure simulation environment, where various concepts for the new system are evaluated, or in a hardware-in-the-loop simulation, where the new system is debugged while in operation.

Figure 2.2.1 is a flow chart for the development of NOMAD. The first stage in the process is the conceptual design stage. The second stage is the main design process which consists of theoretical design and simulation of the overall system together with detailed design of all the subsystems: energy supply, body layout and mechanical components, sensors and electronics, computer hardware and software, communications, and payload. Cost effectiveness analysis was carried out throughout the main design process. The search for suitable components was also involved in the process. The design requires the most effective and suitable technology instead of the most sophisticated. Every possible alternative was thoroughly studied.

The third stage is the construction stage. This mainly consists of subsystem assembly (electrical, mechanical and pneumatic subsystems) and control strategy implementation (computer hardware and software). Many technical modifications are made at this stage. The last stage is the test stage, consisting of the dry-land tests, in-water tests and data analysis. As seen in Figure 2.2.1, many interactions occurred between the different stages and phases of development, so that "on-line" designs and tests became essential in the development process.



Marine Observation Missions
Mission Requirements
Concept Design for DMOS
Constraints
Tasks for the Subsea Robot
Technical Requirement for the Subsea Robot
Concept Design for the Prototype - NOMAD
Pay Load Chamber
Energy Supply
Body Structure Component Layout
Computer Control
Sensors
Communications
Artificial Pay Load
Duration
Pressure Constant Maintain
Auto CAD Design
Dynamic Model Set Up
Strategies and Algorithms
Software Architecture
Electronic Circuits Design
RS232 Cable
FM Radio Wave
Digital Simulation
Micro Computer Hardware Design
Laboratory Experiment
Evaluation of Cost Effectiveness
Hybrid Digital and Analog Simulation
Decision Making for Detail Designs
Search for Suitable Components
Mechanical Subsystem Buildup
Pneumatic Subsystem Buildup
Electronics and Computer Hardware Buildup
Software Development and Debug
Assembly
Dry Land Test
In-Water Test
Data Analysis
Further Work

Figure 2.2.1 Procedures in the Development of Subsea Robot NOMAD

§2.3 General Design of the Subsea Robot NOMAD

Subsea robot NOMAD is the critical part of the Distributed Marine Observation System. To satisfy the requirements of its missions, it should be designed to have a modular structure in its physical layout and a layered structure in its control software. An overview of its main physical subsystems is given below.

1. Main driving mechanisms

Taking advantage of the gravity and buoyancy forces, the subsea robot will consume no energy during descent or ascent. The only energy consumption will occur at the point where the direction of movement is changed. That makes it possible for NOMAD to work on long duration missions.

A ballast tank is installed on the upper part of the robot. The water level in the ballast tank is controllable. As shown in Figure 2.3.1, the tank has holes top and bottom. With the top hole closed, letting gas bottle air flow into the ballast tank forces water out the bottom and causes the robot to surface; letting air out the top and water in through the bottom causes the robot to sink. This is also the method used to adjust the subsea robot to a neutral buoyancy when the robot is commanded to hover at a specified depth. Pneumatic plugs within the ballast tank are used to open or close its openings and thus control the flow of water into and out of the tank. The plugs are controlled by solenoid valves. A solenoid valve is also used to control the flow of compressed air from the gas bottle into the ballast tank. The valves are controlled by an onboard micro computer which is powered by DC batteries. When a

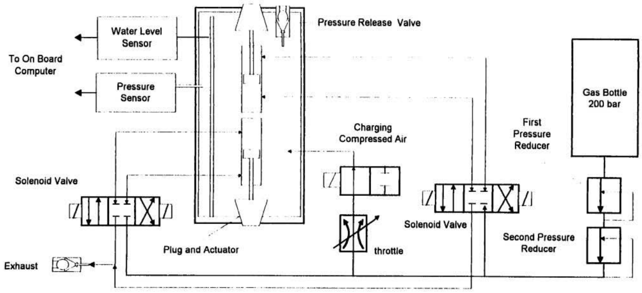


Figure 2.3.1 Layout of the Pneumatic Subsystem

" 0" is sent out by the micro computer, the valves are in the OFF position. The openings top and bottom are closed and there is no air flow from the gas bottle. In the OFF position the valves do not draw power from the batteries either. Use of pneumatic plugs allows rapid filling and purging of the ballast tank and thus fast response. A water level sensor is installed in the tank forming a closed control loop to adjust the weigh/buoyancy force and obtain the neutral buoyancy point. High pressure within the ballast tank could be used to create strong water or air jets to accelerate the subsea robot (Muggeridge and Hinchey, 1992) and it could also be used to blow away mud accumulated on the subsea robot if it sat on the sea floor for a long time. However, this would drain the compressed air bottle fast and would only be used in rare cases.

2 Main Energy Supply System

A standard gas bottle with maximum pressure 200 bar and 11 liter capacity is adapted as a main energy source. Gas bottles in the same series with higher maximum pressure and larger capacity could be mounted on the subsea robot as well. NOMAD is designed to have a modular structure. It only takes a couple of minutes to change a gas bottle.

A first stage pressure reducer (SCUBA) reduces the output air pressure from the bottle to 13 bar. A second stage adjustable reducer (FESTO LRI/8FT) reduces the pressure from 13 bar to around 3 bar. Since the reference pressure used by the pressure reducer is the pressure in the surrounding waters, the stabilized working pressure is with respect to the pressure at the current depth. To operate the solenoid valves, the working pressure of the pneumatic control system must be kept constant around 3 bar for any gas consumption rate or robot depth. A flow control valve (FESTO GRO-M5) is used to adjust the flow rate when the ballast tank is being charged. The ballast tank has a relief valve (FESTO H1/2 B) to vent pressure to the surroundings if the relative pressure across its wall becomes too high.

3. Sensors

NOMAD has four different types of electronic sensors on board. These detect respectively: the depth of the subsea robot, the pressure in the ballast tank, the water level in the ballast tank and the wave action of the surroundings.

(1) Depth sensor

A pressure transducer (OMEGA PX181) is used to detect the depth. This electronic sensor transfers the pressure difference between its input and a reference pressure to a voltage range of 0 ~ 5 V. Then the voltage signal is sent to a 12 bit A/D converter. Since PX181 is a relative pressure sensor, it is important to keep a stable reference pressure on the subsea robot. The sensor is fixed in a watertight box, in which the air pressure is kept at 1 bar. The input to the sensor is opened to the surrounding water.

(2) Pressure Sensor in the Ballast Tank

A pressure transducer is used to measure the air pressure in the ballast tank. The output of the transducer is fed to the onboard micro computer through an 8 bit

A/D converter. The onboard micro computer adjusts the pulse width of the electrical current supplying the solenoid valve which controls the air flow charging to the ballast tank so that the pressure inside the tank can be charged to a preset value which is proportional to the intensity of the water/air jets. As mentioned before, the water/air jets help to accelerate the initial speed of the subsea robot.

(3) Water Level Transducer in the Ballast Tank

A continuous water level transducer was going to be installed in the ballast tank as shown in Figure 2.3.2. For this sensor, the capacitance between the conductor inside the probe and the water varies with the water level: the higher the water level, the larger area of the two polar plates, the bigger the capacitance, the wider the pulse output from the Mono-Pulse Generator, and the higher the voltage output from the Lower Pass filter (Allocca and Stuart, 1984).

This type of water level transducer is able to obtain continuous water level measurement. However, it is expensive (1000 US$) and very sensitive to electromagnetic disturbance from surroundings. In the prototype of NOMAD, five micro switches with small floaters are mounted along the axis of the ballast tank. They are connected to the digital I/O ports of the onboard micro computer. The advantage of this alternative is that it is reliable, economic and simple. Each switch can indicate two water levels by the status of "ON" and "OFF". Hence 10 water levels could be precisely detected. The water levels between the switches would be estimated by the software.

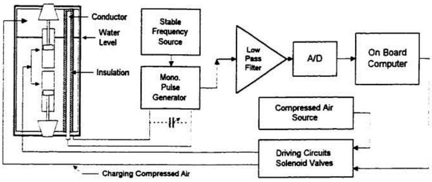


Figure 2.3.2 The Continuous Water Level Transducer and the Water Level Control Loop

(4) Wave Action Sensor

This sensor is designed to detect the wave action of the surroundings. It transfers the robot rocking to a series of pulses. As shown in Figure 2.3.3, the sensor consists of two micro switches, a mass and a chamber. If the subsea robot is subjected to a big wave disturbance, the mass will move between the left and right ends. The outputs from the switches are fed to the serial I/O port of the onboard computer. The time duration t_1, t_2 and the frequency are used to scale the " wave action", where t_1 is the time lag between the back edge of phase B and the front edge of phase A and t_2 is the width of the pulse.

The wave action sensor delivers information about surroundings to the higher level controller of the robot. This allows the onboard micro computer to take the state of surroundings into account when it makes decisions. It could also record the marine dynamic data at different depths.

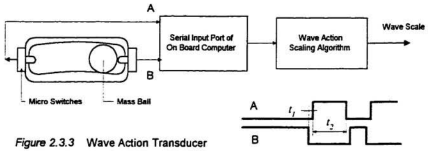

A
Serial Input Port of On Board Computer
Wave Action Scaling Algorithm
Wave Scale
B
Micro Switches
Mass Ball
A
t_1
t_2
B

Figure 2.3.3 Wave Action Transducer

4. The Onboard Micro Computer

To facilitate missions of long term observation an onboard computer is necessary. It should have sufficient input ports and output ports to receive the information from different sensors and drive the different actuators, as well as sufficient memory capacity including EPROM, battery backed RAM, and EEPROM to store the system program, mission description and recorded data. A local expandable bus is also required to supply the power , facilitate the means for data exchange and access to instruments in the payload chamber.

A micro computer named " Rugged Giant", from the Company of Z-World Engineering, was selected to be used as the onboard computer for NOMAD. It has seven A/D inputs, two D/A outputs, ten digital inputs, two serial communication ports, and a PLC Bus Expansion Port (Zworld Engineering, 1992). The layout of the micro computer with Z180 CPU used in the onboard control system of NOMAD is shown in Figure 2.3.4.

When the system is under development, a serial RS232 port is used to

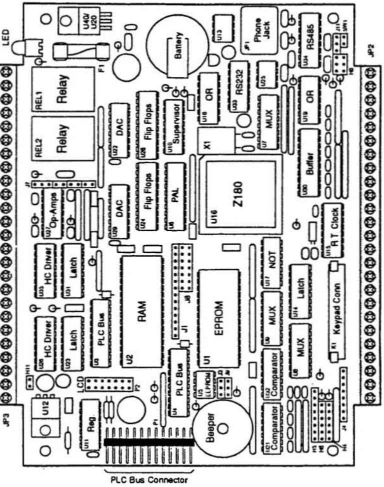


Figure 2.3.4 Layout of the Micro Computer with Z180 CPU
Used in the Onboard Control System of NOMAD

connect the onboard micro computer to the station computer. The system program written in Dynamic C is debugged and compiled to machine code and then downloaded to the EPROM of the onboard computer, while the mission description and mission related parameters are downloaded to the EEPROM of the onboard computer. The data recorded by different sensors are stored in battery backed RAM on the onboard computer. They are to be uploaded to the station computer later. When the subsea robot works in autonomous mode or in remote automatic mode it will be commanded based on the agenda stored in the EEPROM and the decisions made by the higher level control of the onboard computer.

A short wave radio emitter/receiver will be connected to the onboard computer and an antenna will be mounted on the top of the robot. When the robot emerges at the water surface, it will be able to communicate with the station computer to send the recorded data or receive instructions for agenda and mission modification.

A set of DC batteries is mounted on the subsea robot, with the capacity of 1200 mAh, rechargeable. The biggest users of the DC electrical power are the solenoid valves (24 V, 20mA). However, they would not drain any energy when the robot is "asleep" except for a tiny current supply for a time counter working all the time. When a specified time in the agenda arrives, an interruption request is sent to the CPU, the subsea robot then wakes up and enters into mission mode. The battery is expected to last for a long period, which depends on the mission intensity. However, if the instruments in the payload chamber consume significant power, an extra battery would be needed.

5. Payload

For most missions, the payload will be attached at the bottom of the subsea robot. To sample the sediment on the sea floor, a simple mechanical hand could be mounted on the robot frame. The hand could be driven by a small pneumatic cylinder using the power source in the air bottle and controlled by the onboard computer through solenoid valves. Electronic instruments could be sealed in the payload chamber. An open structure could also be applied to the payload if the instruments are watertight. When the mass density of the payload is much greater than that of water, the buoyancy chamber on the top of the robot should be changed. Buoyancy chambers with different sizes have been made for compensating the higher payload mass density. Presently the maximum payload for NOMAD is 16 kg. Because of using gravity and buoyancy forces to drive the robot up and down, the robot would not consume more energy when it carries a heavier payload. However, the acceleration will be smaller and the response time will be longer.

§ 2.4 Architecture of the Subsea Robot NOMAD

The general configuration of the subsea robot NOMAD is shown in Figure 2.4.1 and 2.4.2. The robot is designed to be approximately neutrally buoyant when the ballast tank contains 6 kg of sea water (Effective height of the ballast tank, $H_{eff} = 400\ mm$; Diameter of the ballast tank, $D_{tank} = 200\ mm$: $\frac{1}{2}\left(\frac{H_{eff}\pi D_{tank}^{2}}{4}\right) \approx 6\ kg$). A full ballast tank is denoted as +6 kg (i.e. the Specific Gravity of the whole robot is greater than 1), and an empty ballast tank is denoted as -6kg (i.e. the Specific Gravity of the whole robot is less than 1). These are the maximum forces which drive the robot up and down. The buoyancy chamber on the top is used to balance the payload. The pure buoyancy force created by the buoyancy chamber could be adjusted by putting some water into the chamber. The neutral status is searched by the onboard computer at the stage when the robot is thrown into the water.

Four aluminum boxes which are completely watertight are attached to the frame of the robot. These contain the computer, electronics and sensors, solenoid valves, and communications. The connectors for cables and pneumatic pipes that run through these boxes are also water tight. On the end of every exhaust pipe open to the water, one-way valves are mounted. Hence the three subsystems with different working mediums - electricity, air, and water are totally insulated. That is, the solenoid valves, electronics, and pneumatic parts are kept dry all the time.

The payload and other heavier components are well below the buoyancy chamber and ballast tank. Due to the roughly neutral weight of the air bottle, the large separation of the center of buoyancy and the center of gravity produces a stable platform which can maintain vertical posture even in a rough sea.

Following are the technical details of NOMAD developed in this PhD project:

- Weight: 35 kg
- Height: 1.3 m
- Maximum diameter: 0.3 m

- Working air pressure: 3 ~ 13 bar
- Working voltage: 24V
- Maximum working current: 100 mA ;
    - " sleep" current: <1 mA
- Battery capacity: 1200 mAh
- Standard Volume (under 1 bar pressure) of gas needed for changing direction of motion once:
    - from upward to downward: 0.02 standard liters
    - from downward to upward: 0.35 standard liters
- Air bottle capacity: 200 bar 11 liter
- Maximum exploration depth: 100 m (Needs Enhancement)
- Maximum driving force: 60N
- Maximum speed: 2 m/s
- Depth location error : <0.4 m

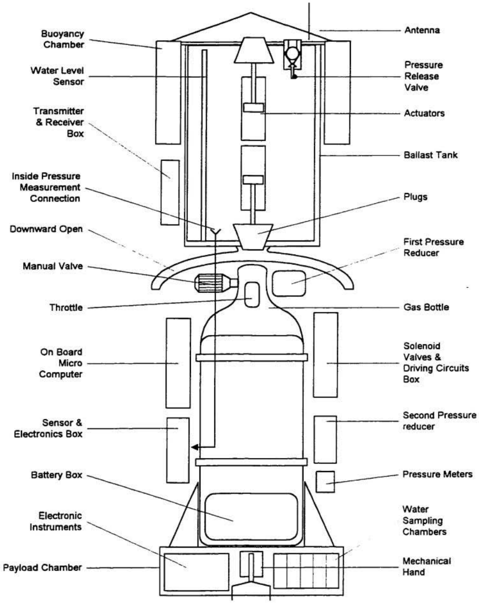


Figure 2.4.1 Architecture of the Subsea Robot NOMAD

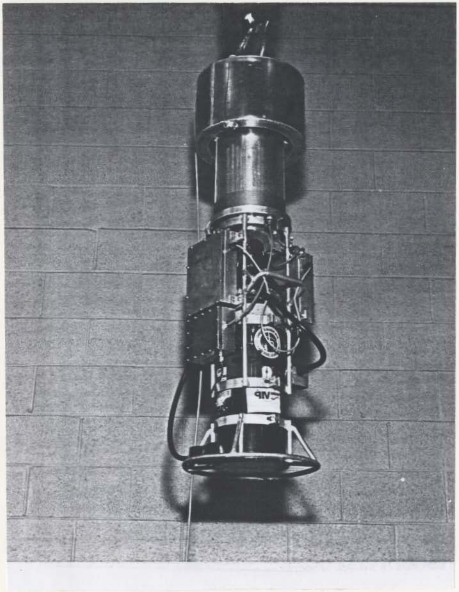


Figure 2.4.2 The Subsea Robot NOMAD During the Tests in the Deep Tank

CHAPTER 3 THE DYNAMIC MODEL OF THE SUBSEA ROBOT NOMAD

This chapter studies the dynamic behavior of the mechanical setup of the robot NOMAD. The model of the robot is divided into two main portions: the model of the robot body and the model of the ballast tank. The latter is the main actuator of the robot. Both of the models are highly nonlinear. Linearization and simplifications are applied to the original models. Block diagrams are established for the design of the computer control system which will be described in the following chapters.

§ 3.1 Theoretical Calculation for the Model of the Robot Body

It is necessary to estimate the hydrodynamic characteristics for the robot before developing the control considerations (Yoerger, Cooke and Slotine, 1990). When a robot moves close to or at a water surface, it generates waves. Because the energy in these waves is supplied by the robot, the generation process can be modeled as a drag force. However, as subsea robots usually do not operate close to the water surface, such drag is insignificant. Wind generated waves die out as one moves down from the water surface. At only one half a wave length down into deep water, the motion is only about 5% of that near the water surface. The wave length dependence means that only long wavelength waves or swells are felt well down into the water. This is where robots tend to operate. Swells have periods around 10 seconds. Because the time it takes for a water particle to move around a robot is typically much smaller than this, the flows around the robot are approximately quasi-steady. This implies that wake drag forces are also approximately quasi-steady. This in turn implies that Reynolds Number is the dominant parameter governing the load. Reynolds number can be estimated as:

$$R_e = \rho U_{rel} D / \mu \approx 1000 \times (0.05 \sim 2) \times 0.3 / 0.001 = 1.5 \times 10^4 \sim 6 \times 10^5$$

where,

μ = 0.001kg/ms is the viscosity of water;

ρ = 1000 kg/m^3 is the density of water (testing done in fresh water);

U_{rel} = (0.05 m/s ~ 2 m/s) is the speed of the robot relative to the swell;

D = 0.3m is the characteristic diameter of the robot.

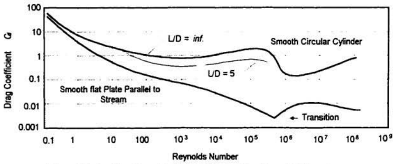


Figure 3.1.1 The Drag Coefficient VersusThe Reynolds Number

As shown in the Figure 3.1.1, the drag coefficient is nearly a constant within $R_e \in (10^3, 6 \times 10^5)$.

According to Morison's Equation the basic dynamic model for a small subsea robot is

$$Md^2Z / dt^2 = -F_i - F_d \qquad \textit{(Equation 3.1.1)}$$

where, M is mass of the robot, Z is vertical distance moved by the robot, F_i is the inertia force on the robot, F_d is the drag force on the robot , and

$$F_i = C_m \rho\, V_{disp}(d^2Z / dt^2 - dU / dt)$$

where, C_m is inertia force coefficient, determined by the geometric shape of the submersible which is estimated to be 0.5 (White, 1986); V_{disp} is the volume of displacement of the robot, U is the swell wave particle speed, and

$$F_d = 0.5 C_d A_{front} \rho\, (dZ / dt - U)\, |dZ / dt - U|$$

where, C_d is drag coefficient determined by Reynolds Number and the geometric shape of the robot; A_{front} is the front area of the robot (Sarpkaya and Isaacson, 1984).

Define:

$$dx/dt = dZ/dt - U.$$

Include the driving force F_p into the dynamic equation. It then becomes:

$$M(d^2x / dt^2 + dU / dt) = -a_1\, dx / dt |dx / dt| - a_2\, d^2x / dt^2 + F_p$$

Because of the quasi-steady assumption, the term dU/dt may be ignored. Assuming $x = x_1$, $dx/dt = x_2$, the state space equation is

$$\begin{aligned} &dx_1 / dt = x_2 \\ &(M + a_2) dx_2 / dt = -F_d + F_p = -a_1 x_2 |x_2| + F_p \end{aligned} \qquad \textit{(Equation 3.1.2)}$$

where,
$$\begin{aligned} a_1 &= 0.5 C_d A_{front} \rho = 0.5 \times (1 \sim 10) \times \pi \times (0.15)^2 \times 1000 \approx 35 \sim 350 \\ a_2 &= C_m \rho\, V_{disp} = 0.5 \times 1000 \times 0.035 = 17.5 \end{aligned}$$

§ 3.2 Experimental Estimation for the Body Model

Even though the robustness of the controller makes the system capable of tolerating a rather big model error, a more precise model is always helpful, especially in the simulation. In the digital simulation, the real world plant will be replaced by its mathematical model when evaluating different control algorithms. In order to have a more precise hydrodynamic model, experiments for identifying the coefficients in the equation

$$(M + a_2)\dot{x}_2 = -a_1 x_2 |x_2| + F_p \qquad \textit{(Equation 3.2.1)}$$

were conducted.

Equation 3.2.1 always holds with different initial conditions of $x_1(0)$ and $x_2(0)$ or with different driving force F_p. At $t=iT$ and jT we have,

$$\begin{aligned} (M+a_2)\dot{x}_{21}(i)+a_1 x_{21}(i)|x_{21}(i)| &= F_{p1} \\ (M+a_2)\dot{x}_{22}(j)+a_1 x_{22}(j)|x_{22}(j)| &= F_{p2} \end{aligned} \qquad \textit{(Equation 3.2.2)}$$

where,

F_{p1}, F_{p2} are different driving forces in the experiment;

$\dot{x}_{21}(i), x_{21}(i)$ are the acceleration and velocity at the time $t = iT$, and under the driving force F_{p1};

$\dot{x}_{22}(j), x_{22}(j)$ are the acceleration and velocity at the time $t=jT$ and under the driving force F_{p2}.

Properly choosing the driving forces F_{p1} and F_{p2} and letting:

$$\begin{vmatrix} \dot{x}_{21}(i) & x_{21}(i)|x_{21}(i)| \\ \dot{x}_{22}(j) & x_{22}(j)|x_{22}(j)| \end{vmatrix} \neq 0 \qquad \textit{(Equation 3.2.3)}$$

gives

$$\begin{bmatrix} M+a_2 \\ \\ a_1 \end{bmatrix} = \begin{bmatrix} \begin{vmatrix} F_{p1} & x_{21}(i)|x_{21}(i)| \\ F_{p2} & x_{22}(j)|x_{22}(j)| \end{vmatrix} \\ \\ \begin{vmatrix} \dot{x}_{21}(i) & F_{p1} \\ \dot{x}_{22}(j) & F_{p2} \end{vmatrix} \end{bmatrix} \frac{1}{\begin{vmatrix} \dot{x}_{21}(i) & x_{21}(i)|x_{21}(i)| \\ \dot{x}_{22}(j) & x_{22}(j)|x_{22}(j)| \end{vmatrix}}. \qquad \textit{(Equation 3.2.4)}$$

Usually a_1 and a_2 vary with the velocity of the robot. However, when the Reynolds Number is within $(10^3, 6\times10^5)$, it will not introduce significant error if they are treated as constants. The results of tests conducted in the deep tank of Ocean Engineering Research Center, Memorial University confirmed that a_1 and a_2 have no significant variation at different velocities of the robot. The detail of the experiment to identify the parameters of a_1 and a_2 is discussed in Chapter 8. The final result of the experiment shows: $a_1 \approx 37.5$, $a_2 \approx 15$.

Hence the second order dynamic model for NOMAD is

$$\begin{cases} \dot{x}_1 = x_2 \\ 50\dot{x}_2 = -37.5\,|x_2|x_2 + F_p \end{cases} \qquad \textit{(Equation 3.2.5)}$$

Using this model to simulate the displacement response to the driving force F_{p1} and F_{p2} , the results are quite close to the data recorded from the deep tank test.

This is a nonlinear model which gives us a rough idea of the dynamic character of the robot body. It is used as a part of the plant model in the lower control level design. The block diagram of the body model is shown in Figure 3.2.1.

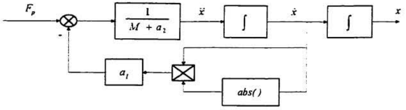


Figure 3.2.1 The Block Diagram of the Body Model

§ 3.3 Simplification of the Ballast Tank Model

As shown in Figure 3.3.1, the water level *h* in the ballast tank is fed back to the onboard micro computer and compared with a desired value h_d. The error e_h is used to drive the valves open and closed, so that *h* approaches h_d.

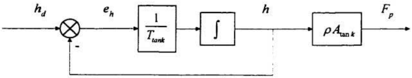


Figure 3.3.1 The Ballast Tank Model Represented by a First Order Device

If servo valves were used to control the flow into and out of the ballast tank, the model of the tank could be quite precisely represented by a first order device which mainly consists of an integral path and a feedback path as shown in Figure 3.3.1, where T_{tank} is the time constant and A_{tank} is the cross sectional area of the ballast tank. Assuming that the reference input is a step with height h_d, the response of the first order device to this input is

$$h(t) = h_d(1 - e^{-\frac{t}{T_{tank}}})$$

and its derivative and slope are

$$\frac{dh(t)}{dt} = \frac{h_d}{T_{tank}} e^{-\frac{t}{T_{tank}}} , \qquad k_{slope} = \left.\frac{dh(t)}{dt}\right|_{t=0} = \frac{h_d}{T_{tank}},$$

where k_{slope} is the slope of the response curve at the starting point. It corresponds to the maximum flow rate. This time response is shown in Figure 3.3.2. The above calculation is based on using servo valves to control the flow into or out of the ballast tank. Servo valves are quite sensitive to the impurities in sea water. The servo parts in these valves are easily clogged. They are also expensive, especially those that have high flow rates and corrosion-proof characteristics.

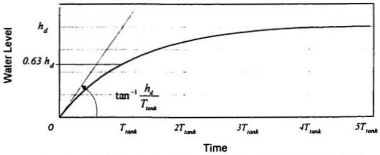


Figure 3.3.2 The Water Level Response of An Ideal First Order Device

To enhance the reliability and reduce the cost an opening-closing mechanism driven by solenoid valves is adopted. When the water in the tank is to be purged, the plug at the bottom of the tank is opened. The water outward flow rate is equal to the flow rate of the compressed air charged into the tank. This is adjustable by a throttle (see Figure 2.3.1). When the water is drawn into the tank, the plugs at both the top and the bottom of the tank are opened.

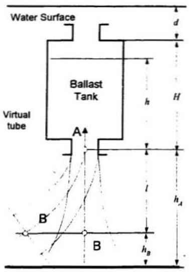


Figure 3.3.3 Calculation For the Draw-in Flow Rate

The inflow rate depends on the water level in the tank. To explain this, pick two points in Figure 3.3.3: Point *A* is at the middle of the nozzle, point *B* is an arbitrary point outside the tank. Ignoring the effect of vortices, a virtual stream tube could be imagined to connect points A and B. The law of conservation of energy requires that the water particle at point A contain the same amount of energy as at point B. Here the Bernoulli Equation for one-dimensional incompressible flow is valid (White, 1986). Then we have:

$$P_A + \rho g h_A + \frac{1}{2}\rho V_A^2 = P_B + \rho g h_B + \frac{1}{2}\rho V_B^2$$

where,

P_A, P_B are the pressures at points A and B;

h_A, h_B are the elevations of points A and B relative to a common reference;

V_A, V_B are the velocities of water particles at points A and B;

Substitution of the variables in Figure 3.3.3 into Bernoulli's Equation gives

$$(\rho dg + \rho hg) + \rho g h_A + \frac{1}{2}\rho V_A^2 = (\rho dg + \rho Hg + \rho lg) + \rho g h_B + \frac{1}{2}\rho V_B^2$$

$$\because\ h_B + l = h_A, \qquad \therefore\ \frac{1}{2}\rho V_A^2 - \frac{1}{2}\rho V_B^2 = \rho g(H-h),$$

where H is the height of the ballast tank, 0.47m.

If point B is far enough away from the nozzle, the velocity of the water particle at B is nearly zero, so that we have $V_A = \sqrt{2g(H-h)}$.

The block diagram of the ballast tank model using solenoid valves is shown in Figure 3.3.4, where

v_N is the water outflow velocity. It is adjustable by the air inflow rate which is set by the throttle.

v_p is the water inflow velocity. It equals $V_A = \sqrt{2g(H-h)}$;

A_{nozzle} is the cross sectional area of the nozzle, $A_{nozzle} = \pi \times 0.02^2 = 1.25 \times 10^{-3} m^2$;

Q is the water flow rate into or out of the ballast tank.

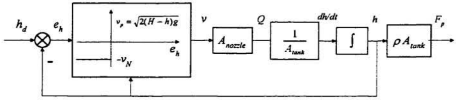


Figure 3.3.4 **The Block Digram of the Ballast Tank Using Solenoid Valves**

To make the water flow rate into and out of the tank symmetrical at the neutral buoyancy point, the air inflow rate is adjusted to make the water outflow velocity v_N roughly equal to the water inflow velocity v_p when the water level is at the half height of the ballast tank, as shown in Figure 3.3.5,

$$v_N = v_p = \sqrt{2(H-h)g}\Big|_{h=\frac{1}{2}H} = \sqrt{Hg}\,.$$

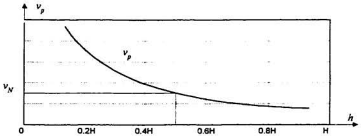


Figure 3.3.5 Making Outflow Velocity v_N Equal to Inflow Velocity v_p at the Half Height of the Ballast Tank

Substitution of the height H of the ballast tank, H ≈ 0.47m, gives

$$v_N = \sqrt{Hg} = \sqrt{0.47 \times 9.8} = 2.14 m/s$$

The time needed to drain the amount of water in the tank which allows the robot to shift from fully downward moving state to the neutral state is

$$t_{out} = \frac{F_{pmax}}{\rho \cdot v_N \cdot A_{nozzle}} = \frac{6kg}{1000kg/m^3 \times 2.14(m/s) \times 1.256 \times 10^{-3}(m^2)} = 2.23\,sec$$

(Equation 3.3.1)

where, F_{pmax} is the maximum driving force in either direction, $F_{pmax} = 6kg$.

The time needed to draw in water which allows the robot to shift from the neutral state to a fully downward moving state can be estimated by

$$t_{in} = \int_{0.5H}^{0.9H} \frac{A_{tank}}{v_p A_{nozzle}} dh \approx \int_{0.5H}^{0.9H} \frac{A_{tank}}{A_{nozzle}\sqrt{2(H-h)g}} dh, \qquad \textit{(Equation 3.3.2)}$$

where, A_{tank} is the effective cross sectional area of the ballast tank, $A_{tank} = 3.14 \times 0.1^2 = 0.0314 m^2$; h is the current height of the water level. The subsea robot is approximately at the neutral state when the water level in the ballast tank is at about 1/2 H, and at the fully downward state when the water level is at 9/10 H; and in the fully upward moving state when the water level is at 1/10 H. Calculation of Equation 3.3.2 gives $t_{in} \approx 2.86\,sec.$

If servo valves are used for water level control in the ballast tank, the control loop could be described by an ideal first order model. It needs about 3 T_{tank} to 5 T_{tank} to respond to a step input.

Comparing these two schemes, the one using servo valves and the other using solenoid valves, an "equivalent time constant" for the latter could be approximated by

$$T_{tank} = \frac{1}{cons} \frac{t_{in} + t_{out}}{2} = 0.7 sec. \qquad \textit{(Equation 3.3.3)}$$

where *cons* is a constant between 3 to 5.

The time responses to different step inputs of the two schemes are shown in Figure 3.3.6. It can be seen in this figure that there is some asymmetry when the flow direction changes when using solenoid valves. However the equation

$$v_N = v_P\Big|_{h=\frac{1}{2}H} = \sqrt{2(H-h)g}\Big|_{h=\frac{1}{2}H} = \sqrt{Hg}$$

guarantees the same flow rate in either direction around the water level $h=0.5H$, which corresponds to the neutral state of the robot. Thus the asymmetric nonlinearity can be linearized smoothly at the equilibrium point. The effect of asymmetry on the dynamic behavior of the robot will be further studied by digital simulation at the mechanical component level in Chapter 8.

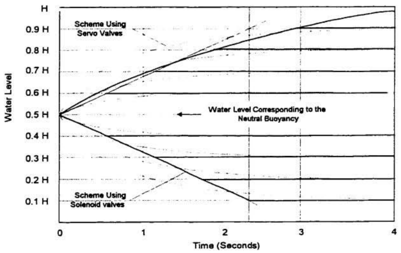


Figure 3.3.6 Comparison between Servo Valve and Solenoid Valve Schemes

Based on the above studies, the model of the water level control in the ballast tank can be roughly treated as a first order device with the equivalent time constant $T_{tank} = 0.7sec$, as shown in Figure 3.3.1.

§ 3.4 Linearization of the Body Model

The model of the robot NOMAD mainly consists of two parts: the ballast tank and the body of the robot. The ballast tank has been treated as a first order device with the time constant $T_{tank} = 0.7sec$ in both the scheme using servo valves and the one using solenoid valves. In the model of the robot body, there is a nonlinear feedback from $\dot{x}$ to $\ddot{x}$ as shown in Figure 3.2.1. The nonlinear characteristic of the feedback is shown in Figure 3.4.1. By Taylor Expansion, the nonlinear feedback could be expressed by a linear term in the vicinity of specified points $\dot{x} = \dot{x}_o$:

$$F_d = a_1\dot{x}|\dot{x}| \approx F_d(\dot{x}_0) + F_d'(\dot{x}_0)(\dot{x} - \dot{x}_0)... \qquad \textit{(Equation 3.4.1)}$$

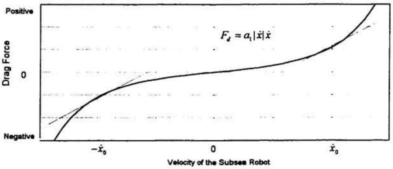


Figure 3.4.1 Linearization of the Nonlinear Drag Force Feedback

Taking the regular working speed $\dot{x}_o = 0.1m/s$ as the expansion point, Equation 3.4.1 could be expressed as

$$F_d = \begin{cases} +a_1|\dot{x}_o|\dot{x}_o + 2a_1\dot{x}_o(\dot{x} - \dot{x}_o) & \dot{x} > 0 \\ -a_1|\dot{x}_o|\dot{x}_o + 2a_1\dot{x}_o(\dot{x} + \dot{x}_o) & \dot{x} < 0 \end{cases} . \qquad \textit{(Equation 3.4.2)}$$

Then we have

$$F_d = \pm 37.5 \times 0.01 + 2 \times 37.5 \times 0.1(\dot{x} \mp 0.1) = 7.5\dot{x} \mp 0.375 .$$

Substitution of F_d into Equation 3.1.2 gives

$$(M + a_2)\ddot{x} = -F_d + F_p = -7.5\dot{x} \mp 0.375 + F_p \qquad \textit{(Equation 3.4.3)}$$

Taking Laplace Transform on both sides and substituting $(M + a_2) = 50kg$ into Equation 3.4.3 gives:

$$50s^2 X + 7.5sX = F_p(s) \pm 0.375\frac{1}{s}$$

$$X(s) = \frac{1}{(50s + 7.5)s}\left(\pm\frac{0.375}{s} + F_p(s)\right) = \frac{0.13}{(6.675 + 1)s}\left(F_p(s) \pm \frac{0.375}{s}\right) \qquad \textit{(Equation 3.4.4)}$$

Since $|F_p|$ is usually much greater than 0.375, the term $\pm\frac{0.375}{s}$ could be ignored. The transfer function of the robot body is

$$G_{body}(s) = \frac{0.13}{s(6.67s + 1)} . \qquad \textit{(Equation 3.4.5)}$$

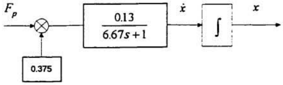


Figure 3.4.2 The Approximate Linear Model of the Robot Body

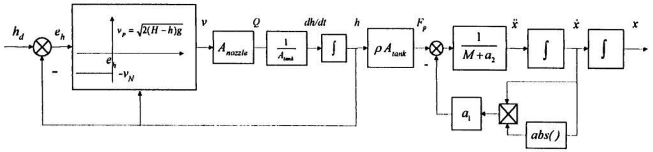


Figure 3.4.3 The Block Digram of The Whole Plant

So the body of the robot could be approximated as an inertial element with the time constant

$$T_{body}=6.67sec \qquad \textit{(Equation 3.4.6)}$$

followed by an integral device. The approximate linearized body model of the robot is shown in Figure 3.4.2.

Since the output of the ballast tank model as shown in Figure 3.3.4 is the input to the robot body model (Figure 3.2.1), the whole model of the plant has a cascade structure as shown in Figure 3.4.3.

Based on the discussion in Section §3.3 and current section, the block diagram of the whole plant could be approximated by a third order system as shown in Figure 3.4.4.

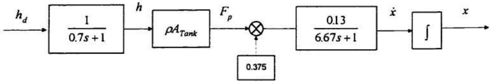


Figure 3.4.4 The Simplified Model of the Whole Plant

CHAPTER 4 CLASSICAL CONTROL STRATEGIES FOR ROBOT DYNAMICS

The main difficulties in designing the controller for the subsea robot dynamics originate from the unknown model with high nonlinearity. The design restrictions, such as energy saving which is critical for a subsea robot with long term missions, make the design process tougher. In this chapter and Chapters 5 and 6, several control strategies for the dynamics of NOMAD are studied and compared. The most suitable one is picked for implementation on the robot. This chapter discusses two classical control strategies, PID control, and Sliding Mode Control.

Since NOMAD only has one degree of freedom, the discussion of its dynamics will concentrate on depth control.

§ 4.1 Conventional PID Control Scheme

The proportional, integral and derivative control strategy, often referred to as PID Control, is considered purely error-driven. Control outputs are generated based on the difference between what should be happening versus what is actually happening. A block diagram of a typical PID controller for the depth control is shown in Figure 4.1.1.

The analog PID controllers have been used successfully in many industrial applications for over half a century. The control law is:

$$h_d(t) = K_p\left[e_z(t) + \frac{1}{T_i}\int_0^t e_z(t)dt + T_d\frac{de_z(t)}{dt}\right] \qquad \textit{(Equation 4.1.1)}$$

or the transfer function is

$$\frac{H_d(s)}{E_x(s)} = K_p(1 + T_d s + \frac{1}{T_i s}) \quad , \qquad \textit{(Equation 4.1.2)}$$

where,

$e_x(t)=x_d(t)-x(t)$ is the difference between the desired depth and actual depth;

x_d	is the desired depth sent to the depth controller from a higher level controller;
$h_d(t)$	is the output of the PID controller, which is also the reference input to the inner control loop of the ballast tank;
K_p	is the proportional gain;
T_i	is the integral time constant;
T_d	is the derivative time constant.

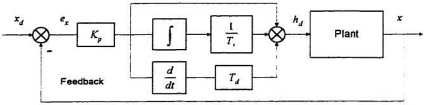


Figure 4.1.1 PID Control Scheme

To implement the PID control law by computer, transformation of Equation 4.1.1 to a discrete form by the Backward Difference Method (Ogata, 1987) gives

$$h_d(kT) = K_p\left[e_x(kT) + \frac{T}{T_i}\sum_{i=0}^{k} e_x(iT) + \frac{e_x(kT) - e_x(k-1)T}{T}T_d\right]. \qquad (Equation\ 4.1.3)$$

The increment of h_d between the time kT and $(k-1)T$ is

$$\Delta h_d(kT) = h_d(kT) - h_d(k-1)T = \alpha_1 e_x(k) + \alpha_2 e_x(k-1) + \alpha_3 e_x(k-2) \qquad (Equation\ 4.1.4)$$

where,

$$\alpha_1 = K_p\left(1 + \frac{T}{T_i} + \frac{T_d}{T}\right)$$
$$\alpha_2 = -K_p\left(1 + \frac{2T_d}{T}\right)$$
$$\alpha_3 = K_p\frac{T_d}{T}$$

and T is the sampling period.

Equation 4.1.4 gives the velocity form of the PID control scheme

$$h_d(k) = h_d(k-1) + \Delta h_d(k).$$

Compared with the position form in Equation 4.1.3, the velocity form exhibits better response characteristics, when there are sudden large changes in the set point, or at the start of an operating process.

There are only three independent parameters in the PID control law. They are highly model dependent. The procedure for determining α_1, α_2, α_3 is establishing the plant model, linearizing the nonlinear parts at the working point, and calculating α_1, α_2, α_3 in the frequency domain based on the criteria for stability, steady state error and settle time. Usually the parameters α_1, α_2, α_3 have to be adjusted while the controller is working. It is hard to set them up in dry-land tests.

The disadvantage of a basic PID controller for subsea robot dynamics is its lack of robustness and lack of ability to deal with the model uncertainty and nonlinearity which are common in the ocean environment. Thus, the basic PID controller would not be the most suitable candidate for subsea tasks.

§ 4.2 Sliding Mode Control Scheme

The sliding mode control strategy combines the computed load technique (Development Group, 1994) with a switching type of error-driven compensation. Switching helps counteract uncertainties and is said to make control more robust (Slotine and Coetsee, 1986). It ensures that there is always some measure of control when uncertainties have known upper bounds.

Sliding mode control is based on the concept of the " sliding control " requiring all trajectories to converge on a time-varying sliding surface, $S(t)$, as shown in Figure 4.2.1. The sliding surface is often referred to as the switching surface and is an idealized state trajectory. A state trajectory is a quantitative description of the condition or state of the system through a period of time (Slotine and Sastry, 1983). High speed switching feedback control is used in an attempt to make the state trajectory lie along the switching surface, $S = 0$.

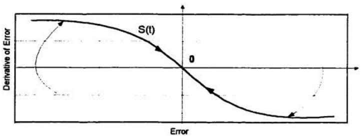


Figure 4.2.1 The Switch Surface S(t) in the Sliding Mode Control

In the case of depth control of NOMAD, one starts by defining the sliding line, as shown in Figure 4.2.1:

$$S = \frac{de_x}{dt} + \lambda\, e_x \qquad \textit{(Equation 4.2.1)}$$

where,

S is the measure of the algebraic distance to the sliding surface, S(t);

e_x is the difference between desired and actual depth;

λ is the bandwidth.

As is generally the case, the sliding equation is one order lower than the system order. The goal of this control strategy is to make S zero. If S, which drives e_x, can be made zero, then e_x must decay and the actual robot depth will match the desired position.

Sliding mode control uses the Lyapunov stability method, specifically Lyapunov's direct method, to ensure system stability. When the control inputs satisfy the " sliding condition" the control algorithm is said to be a true Lyapunov function.

Lyapunov's theory is based on the fact that for the system to remain stable the system energy must decrease with time until it comes to rest at its equilibrium state or position. Lyapunov introduced a fictitious energy function, named the Lyapunov function, since there is no simple way of defining an energy term mathematically. A Lyapunov function, denoted as V_l, is a scalar function satisfying Lyapunov's stability theorems and is based on the state variables for the system. State variables are a set of variables which completely define the system at any instant in time. The time derivative of the Lyapunov function establishes the stability of the system. If the time derivative, dV_l/dt, is negative the system is stable. This method determines system stability without solving state equations, which is quite advantageous for nonlinear and time-varying systems that have complex state equations.

In the case of simple depth positioning of NOMAD, the Lyapunov function, V_l, and its time derivative may be defined as

$$\begin{aligned} V_l &= \frac{M + a_2}{2} S^2 \\ \frac{dV_l}{dt} &= S\,(M + a_2)\frac{dS}{dt} \end{aligned} \qquad \textit{(Equation 4.2.2)}$$

The time derivative of the Lyapunov function must be negative to ensure system stability. This condition which constrains trajectories to be directed towards the sliding line is called the sliding condition:

$$\frac{dV_l}{dt} \leq \zeta |S| \qquad \textit{(Equation 4.2.3)}$$

where, ζ is a negative constant.

In order to determine the value for dV_l/dt which will ensure system stability, S and dS/dt must be defined in terms of the state variables and substituted into Equation 4.2.2.

Substituting the tracking error, which is normally defined as the desired position of the robot minus the actual position, $e_x = x_d - x$, into the sliding equation, Equation 4.2.1, gives

$$S = \frac{dx_d}{dt} - \frac{dx}{dt} + \lambda\, x_d - \lambda\, x \qquad \textit{(Equation 4.2.4)}$$

Its time derivative is

$$\frac{dS}{dt} = \frac{d^2x_d}{dt^2} - \frac{d^2x}{dt^2} + \lambda \frac{dx_d}{dt} - \lambda \frac{dx}{dt} \qquad \textit{(Equation 4.2.5)}$$

where,

x_d is the desired robot depth;

x is the actual robot depth.

Rearranging the equation of motion (Equation 3.1.2) and including a term of disturbance force F_{dis} gives

$$\frac{d^2x}{dt^2} = \frac{1}{M + a_2}\left(F_p + F_{dis} - a_1 \frac{dx}{dt}\left|\frac{dx}{dt}\right| \right) \qquad \textit{(Equation 4.2.6)}$$

Substitution of Equation 4.2.6 into Equation 4.2.5 gives

$$\frac{dS}{dt}=\frac{1}{M+a_2}\left(a_1\frac{dx}{dt}\left|\frac{dx}{dt}\right|-F_p-F_{dis}\right)+\frac{d^2x_d}{dt^2}+\lambda\frac{dx_d}{dt}-\lambda\frac{dx}{dt} \qquad \textit{(Equation 4.2.7)}$$

Equation 4.2.7 gives the value of *dS/dt* in terms of the state variables, but unfortunately, the parameters $M+a_2$ and a_1 and the disturbance, F_{dis}, are unknown. Let $M+\hat{a}_2$, $\hat{a}_1$ and $\hat{F}_{dis}$ be the best estimates of uncertain parameters $M+a_2$ and a_1 and disturbance F_{dis}. If these values for $M+a_2$, a_1, and F_{dis} were exact, then the force that would keep S fixed is

$$\hat{F}_p=\hat{a}_1\frac{dx}{dt}\left|\frac{dx}{dt}\right|-\hat{F}_{dis}+\left(M+\hat{a}_2\right)\left(\frac{d^2x_d}{dt^2}+\lambda\frac{dx_d}{dt}-\lambda\frac{dx}{dt}\right) \qquad \textit{(Equation 4.2.8)}$$

In order to satisfy the sliding condition despite uncertainty, a discontinuous control term is added to the continuous control law. The lowest level sliding mode control strategy uses ideal relay compensation and lets

$$F_p=\hat{F}_p+K_R\,sign(S), \qquad \textit{(Equation 4.2.9)}$$

where, K_R is the relay gain, and

$$sign(S)=+1 \quad if\ S>0$$

$$sign\,(S) = -1 \quad if\, S < 0$$

Through substitution of Equation 4.2.8 into Equation 4.2.9 and substitution of the resultant equation for F_p into Equation 4.2.7, the dS/dt equation becomes

$$\frac{dS}{dt} = \frac{1}{M+a_2}\left[(a_1-\hat{a}_1)\frac{dx}{dt}\left|\frac{dx}{dt}\right| - (F_{dis}-\hat{F}_{dis}) + (a_2-\hat{a}_2)\left(\frac{d^2x_d}{dt^2}+\lambda\frac{dx_d}{dt}-\lambda\frac{dx}{dt}\right) - K_R sign(S)\right]$$

(Equation 4.2.10)

Equation 4.2.10 with exact values for $M+a_2$, a_1 and F_{dis} reduces to

$$\frac{dS}{dt} = -\frac{K_R}{M+a_2} sign(S)$$

In this case, S would go to zero and cause e_x to decay. With inexact $M+a_2$, a_1 and F_{dis}, a sliding mode control strategy tries to adjust the relay gain, K_R, so that the time derivative of V_1 is always more negative than $\zeta|S|$.

Substitution of Equation 4.2.10 into Equation 4.2.2 gives a general form for dV_1/dt,

$$\frac{dV_1}{dt} = \left[(a_1-\hat{a}_1)\frac{dx}{dt}\left|\frac{dx}{dt}\right| - (F_{dis}-\hat{F}_{dis}) + (a_2-\hat{a}_2)\left(\frac{d^2x_d}{dt^2}+\lambda\frac{dx_d}{dt}-\lambda\frac{dx}{dt}\right) - K_R sign(S)\right]S$$
$$= \left[(P-\hat{P}) - K_R sign(S)\right]S$$

(Equation 4.2.11)

where, $P-\hat{P}$ is the error in the estimated values for $M+a_2$, a_1 and F_{dis} of the system. Comparison of Equation 4.2.11 with Equation 4.2.3 gives

$$K_R \geq \left| P - \hat{P} \right| - \zeta .$$

The value for the relay gain, K_R, must be large enough to ensure stabilization. A good estimate of $|P - \hat{P}|$ is necessary for sliding mode control to function properly. Unfortunately, it is very difficult to estimate values for the uncertain parameters $M+a_2$ and a_1 and the disturbance, F_{dis} , and thus obtaining a good estimate of $|P - \hat{P}|$ is close to impossible (Muggeridge, 1994).

Even though there are some modifications to the classical sliding mode control, such as using a Fuzzy Logic Controller to change the slope of the sliding line (Xu and Smith, 1995), it still requires a high rate switching in the control output to force the state variables to a desired trajectory (Cristi, Papoulias and Healey, 1990). Every switching in the control output, as described in Equation 4.2.9 means a change of flow direction in or out of the ballast tank, which consumes the compressed air stored in NOMAD. A high frequency of switching will drain the energy source quickly. Therefore, based on energy conservation considerations, Sliding Mode Control would not be an appropriate strategy for NOMAD.

CHAPTER 5 NEURAL NETWORK BASED ADAPTIVE CONTROL SCHEME

Control of an underwater robot is difficult because its dynamics is highly nonlinear and its operating environment is very unpredictable. Classical control strategies exhibit their shortcomings in such applications (Hinchey and Muggeridge, 1994). A new control scheme using neural networks to deal with the difficulties in the depth control of NOMAD is developed in this PhD project. It has been simulated with a virtual plant of NOMAD. This neural network based control scheme consists of three main parts:

1. A neural net based Identifier (NNI) to identify the dynamic behavior.
2. A neural net based Adjuster (NNA) in the forward path to generate the signal to control the depth.
3. A second order Optimized Linear Reference or OLR model.

This scheme theoretically does not need a physical model of the subsea robot. However, such models could be used in the early stages of the development of a neural network based controller to speed up the training process. In this Chapter, a digital simulation of the neural network scheme is developed and results of the simulation are presented.

§ 5.1 Configuration for the Neural Network Controller

1. Problem Description

Consider a discrete-time multiple-input multiple-output nonlinear plant described by the following set of difference equations

$$\text{Plant} \quad \begin{cases} X_s(k+1) = f(X_s(k), U(k)) \\ \xi(k) = h(X_s(k), U(k)) \end{cases} \qquad \textit{(Equation 5.1.1)}$$

where, in terms of the real number field R, $X_s \in R^n$, $\xi \in R^l$ and $U \in R^m$ are state, output and input vectors respectively, and $f: R^n \times R^m \rightarrow R^n$ and $h: R^n \times R^m \rightarrow R^l$ are differentiable nonlinear functions. The plant assumes that functions f and h are unknown and the state vector $X_s(k)$ is accessible at time k. In the case of the robot NOMAD, the state variables of the robot body model are the depth and vertical velocity, if the model is simplified to a second order system. They are physically measurable at any time by the depth sensor and a differential device.

2. Feed Forward Neural Network

Neural networks exhibit great potential in handling the nonlinearity and uncertainty existing in the dynamic control of subsea robots (Hinchey, 1994).

Much research has been carried out on the static neural network, which is powerful in pattern recognition and function reconstruction (Yuh, 1990). The basic structure of this type of neural network (Lu, 1993). is shown as in Figure 5.1.1.

It is known that a small feedback system is equivalent to a large and possibly infinite static system (Hush and Horne, 1993). This is because an infinite number of static logic gates would be needed to emulate a machine with arbitrary finite states (Jin and Gupta, 1995).

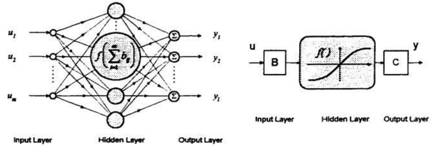


Figure 5.1.1 Schematic Representation of Static Neural Network

3. Dynamic Neural Network

To establish the input-output relations for a dynamic system, where the input and output are both time dependent variables, state feedback has to be introduced into the static neural network. The general idea is that the values of state variables in the next step is predicted using the current values of the state variables. Then the state variables are fed back to the neural network input. A general expression of this type of neural network with state feedback is given by the following equation:

$$\begin{cases} Z(k+1) = -\lambda\, Z(k) + A\sigma\,(Z(k)) + BU(k) \\ X(k) = CZ(k) \end{cases} \qquad \text{(Equation 5.1.2)}$$

where,

$Z \in R^n$, $X \in R^l$ and $U \in R^m$	are neural state, output and input vectors;
$A \in R^{n \times n}$, $B \in R^{n \times m}$, $C \in R^{l \times n}$	are connection weight matrices linking the neural state,

output and input vectors;

λ is the feedback gain for controlling state variable $Z(k)$ decaying, which is chosen between 0 and 1.

$\sigma: R^n \rightarrow (-1,1)^n$ is a sigmoidal function vector which is chosen as $\sigma(input) = tan(input)$.

The structure of a dynamic neural network is shown in Figure 5.1.2. Note that the z^{-1} term in Figure 5.1.2 comes from a z transformation or a Laplace transformation of a sampled data system.

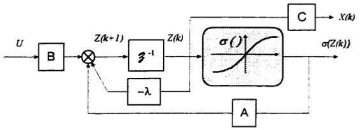


Figure 5.1.2 Schematic Representation of Dynamic Neural Network

The Basic Approximation Theorem (Funahashi and Nakamura, 1993) indicates that: assuming φ to be a continuous vector function defined on $R^l \mapsto R^l$, consider a set of difference equations which could be regarded as a state transition process of a physical system:

$$\xi(k+1) = \varphi(\xi(k)) \qquad \xi \in R^l \qquad \textit{(Equation 5.1.3)}$$

and a dynamic neural network with zero input

$$Z(k+1) = -\lambda Z(k) + A\sigma(Z(k)) \qquad \textit{(Equation 5.1.4)}$$
$$X(k) = CZ(k)$$

where, $Z \in R^n$, $X \in R^l$. With an appropriate initial state, there exists an inequality with an arbitrary $\varepsilon > 0$ as follows:

$$\max_k \| X(k) - \xi(k) \| < \varepsilon \qquad 0 \le k \qquad \textit{(Equation 5.1.5)}$$

The learning process of the neural network is to update the weight matrices, so that the output error between the transition process of the physical system and the output of the dynamic neural network converges to zero as $k \to \infty$. Thus the identification for the unknown dynamics is reached.

If the weights of a neural network are taken into account as the unknown parameters of a nonlinear system, the weight learning problem of the neural network can be phrased as a parameter identification problem for the dynamic nonlinear system. The truth of the above statement is held for a neural network based dynamic controller. The back-propagation algorithm for the multiple-layered feed forward neural networks (MFNN) has many successful applications (Miyamoto, Kawato, Setoyama and Suzuki, 1988). A simple and natural extension of this training algorithm is suitable to train dynamic neural networks, which was first studied by Williams and Zipser (Williams and Zipser, 1989) and Narendra and Parthasarthy (Narendra and Parthasarthy, 1991).

Given a transition process $\xi(k)$ described in Equation 5.1.1, which a dynamic neural network (Equation 5.1.2) is supposed to follow, a sum-squared error function could be defined as follows:

$$e(k) = \| X(k) - \xi(k) \|^2 \qquad \textit{(Equation 5.1.6)}$$

Partition of the matrices A, B and C in Equation 5.1.2 gives

$$A = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} \quad B = \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix} \quad C = \begin{bmatrix} c_1 \\ \vdots \\ c_l \end{bmatrix}$$

Using the gradient descent technique, the weight updating formulations for the purpose of minimizing the error function can be obtained as

$$\begin{aligned} a_i(k+1) &= a_i(k) - \eta \frac{\partial E(k)}{\partial a_i}\Big|_k \\ b_i(k+1) &= b_i(k) - \eta \frac{\partial E(k)}{\partial b_i}\Big|_k \\ c_i(k+1) &= c_i(k) - \eta \frac{\partial E(k)}{\partial c_i}\Big|_k \end{aligned} \qquad \textit{(Equation 5.1.7)}$$

where, $E(k) = [\, e_1(k), \ \dots \ , e_d(k) \,]^T = [\, x_1(k)-\xi_1(k), \ \dots \ , x_d(k)-\xi_d(k) \,]^T$ and $\eta > 0$ is a learning rate. The partial derivatives on the right-hand side of the above equations are computed at time instant *k*.

As the weight matrices *A, B, C* are updated, $e(k)$ converges to its minimum, $\lim_{k\to\infty} e(k) = 0$.

4. Architecture of the Neural Network Controller for Depth Control

The neural network controller (NNC) for the depth control of NOMAD is divided into three parts as shown in Figure 5.1.3.

The Identifier (NNI) for the plant is actually a state estimator which represents a model of the unknown nonlinear plant. It consists of a three layer neural network.

The hidden layer has 8 neurons with a *tan( )* type sigmoidal mapping function. The output layer has 2 neurons with a linear mapping function. When the network of the Identifier is being trained, the error E_m is used to update the weight matrices in the Identifier and force the sum-squared error $\| E_m \|^2$ to fall into a minimum, which allows the output of the Identifier X_m to follow the variation of the state variables X_p in the real world plant.

A desired second order linear model is chosen as a reference for the plant. It is not difficult to design such an optimized model by the Linear Quadratic Regulator LQR method for a second order system. This reference model has a satisfactory response to command inputs for the depth.

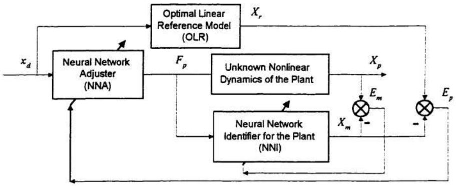

Optimal Linear Reference Model (OLR)
X_r
x_d
Neural Network Adjuster (NNA)
F_p
Unknown Nonlinear Dynamics of the Plant
X_p
E_m
E_p
Neural Network Identifier for the Plant (NNI)
X_m
-
-

Figure 5.1.3 Three Main Parts of the Neural Network Based Controller : NNA, NNI, OLR

The Adjuster (NNA), which is the main controller in the forward path, is also a three layer neural net with a hidden layer of 6 neurons. The weights in the Adjuster are updated by a back propagation algorithm. The error E_p, the difference between the output of the reference model and the output of the Identifier, is back propagated to the Adjuster and the weights in the Adjuster are modified to force the sum-squared error $\|E_p\|^2$ to a minimum. For each net of NNI and NNA, a trial and error procedure was used to find an adequate number of hidden neurons.

In terms of the input-output relationship, the dynamics of the Identifier could be expressed as

$$\textit{Identifier} \quad \begin{cases} Z_m(k+1) = -\lambda_m Z_m(k) + A_m \sigma(Z_m(k)) + B_m F_p(k) \\ X_m(k) = C_m Z_m(k) \end{cases} \qquad \textit{(Equation 5.1.8)}$$

where,

$F_p(k)$ is the driving input to the Identifier, the same as to the plant;

$Z_m(k)$ is the intermediate variable in the Network NNI;

$X_m(k)$ is the output of the Identifier;

A_m, B_m, C_m are the weights of the neural net in the Identifier;

λ_m is the feedback gain for controlling state variable $Z_m(k)$ decaying, which is chosen between 0 and 1.

The dynamics of the Adjuster could be expressed as

$$\textit{Adjuster} \quad \begin{cases} Z_a(k+1) = -\lambda_a Z_a(k) + A_a \sigma(Z_a(k)) + B_a x_d(k) \\ F_p(k) = C_a Z_a(k) \end{cases} \qquad \textit{(Equation 5.1.9)}$$

where,

$Z_a(k)$ is the intermediate variable in the network NNA;

$x_d(k)$ is the set point for the depth control system, which is also sent to the optimized reference model;

$F_p(k)$ is the output of the Adjuster, which is proportional to the set point for the water level in the ballast tank, as well as the input to the Identifier;

A_a, B_a, C_a are the weights of the neural net in the Adjuster;

λ_a is the feedback gain for controlling state variable *$Z_a(k)$* decaying, which is chosen between 0 and 1.

The error function for training the dynamic neural network NNI is

$$e_m(k) = \left\| X_p(k) - X_m(k) \right\|^2 = \sum_{i=1}^{n} (x_{pi}(k) - x_{mi}(k))^2 \qquad \text{(Equation 5.1.10)}$$

where,

$X_p(k)$ is the state vector in the real world plant;

$X_m(k)$ is the output of the Identifier;

n is the number of the state variables in the real world plant,

and

$$e_p(k) = \left\| X_r(k) - X_m(k) \right\|^2 = \sum_{i=1}^{n} (x_{ri}(k) - x_{mi}(k))^2 \qquad \text{(Equation 5.1.11)}$$

where,

$X_r(k)$ is the state variable of the Optimized Linear Reference (OLR) model.

The error vector $E_p(k) = [\, x_{r1}(k)-x_{m1}(k),\ \ldots,\ x_{rn}(k)-x_{mn}(k)]^T$ is used to train the Adjuster in the forward path of the system, while $E_m(k)=[x_{p1}(k)-x_{m1}(k),\ \ldots,\ x_{pn}(k)-x_{mn}(k)\,]^T$ is chosen to train the Identifier as a dynamic model of the unknown plant. The main updating formulations for the weights are given as follows:

For the weights in the Adjuster

$$a_{ai}(k+1) = a_{ai}(k) - \eta \frac{\partial E_p(k)}{\partial a_{ai}}\Big|_k$$
$$b_{ai}(k+1) = b_{ai}(k) - \eta \frac{\partial E_p(k)}{\partial b_{ai}}\Big|_k \qquad \textit{(Equation 5.1.12)}$$
$$c_{ai}(k+1) = c_{ai}(k) - \eta \frac{\partial E_p(k)}{\partial c_{ai}}\Big|_k$$

and for the weights in the Identifier

$$a_{mi}(k+1) = a_{mi}(k) - \eta \frac{\partial E_m(k)}{\partial a_{mi}}\Big|_k$$
$$b_{mi}(k+1) = b_{mi}(k) - \eta \frac{\partial E_m(k)}{\partial b_{mi}}\Big|_k \qquad \textit{(Equation 5.1.13)}$$
$$c_{mi}(k+1) = c_{mi}(k) - \eta \frac{\partial E_m(k)}{\partial c_{mi}}\Big|_k$$

The output $X_m(k)$ of the Identifier is forced to home in on the real state vector $X_p(k)$ of the plant which is assumed to be available at the time instant k. If the unknown plant, as shown in Figure 3.2.1, is perfectly identified by the Identifier (NNI) after the learning process, the following relationship could be established:

$$\lim_{k\to\infty} \| X_p(k) - X_m(k) \| = 0 \qquad \textit{(Equation 5.1.14)}$$

Next, the Identifier could be treated as a known model of the plant. By the Equation 5.1.12, the error $E_p(k)$, which is the difference between the outputs of the OLR model and the Identifier, is forced to zero after a number of iterations or epochs.

$$\lim_{k\to\infty} \| X_r(k) - X_m(k) \| = 0. \qquad \textit{(Equation 5.1.15)}$$

Equating Equation 5.1.14 with 5.1.15 gives

$$\lim_{k\to\infty}\left\|X_r(k) - X_p(k)\right\| = 0\,, \qquad \textit{(Equation 5.1.16)}$$

which makes the state variable $X_p(k)$ approach the state variable $X_r(k)$ in the OLR model. The overall performance of this control scheme depends upon both the Adjuster Neural Net (NNA) and Identifier Neural Net (NNI): the performance of NNA is closely related with that of NNI. So a higher convergence requirement should be imposed on the training process for the NNI.

The identification process of the plant could be carried out either on-line or off-line. If on-line identification was adopted, a high speed Digital Signal Processor (DPS) for NNI training or a chip with a hardware based neural network would be necessary. In the research for the control system of NOMAD, only the off-line training scheme was studied.

§ 5.2 Optimized Design for the Linear Reference Model

The most effective and widely used technique for optimal design of a linear control system is the method of linear quadratic regulator (LQR) (Williams and Zipser, 1989).

Considering a linear system

$$\frac{dX_r(t)}{dt} = A_r X_r(t) + B_r U_r(t)\,, \qquad \textit{(Equation 5.2.1)}$$

the goal of the optimization is to find a control input $U_r(t)$ which minimizes the quadratic cost function

$$J = \int_0^\infty (X_r^T(t)QX_r(t) + U_r^T(t)RU_r(t))dt \qquad \textit{(Equation 5.2.2)}$$

where Q is a symmetric positive semidefinite n×n matrix and R is a symmetric positive definite m×m matrix. They are used to put the emphasis on minimizing one of the terms

$$\int_0^\infty X_r^T(t)X_r(t)dt \quad or \quad \int_0^\infty U_r^T(t)U_r(t)dt.$$

The problem can be simplified by taking $Q = \phi I$ and $R=I$ to get

$$J = \int_0^\infty \left[\phi\, X_r^T(t)X_r(t) + U_r^T(t)U(t)\right]dt \qquad \textit{(Equation 5.2.3)}$$

where ϕ is the weight coefficient used to put emphasis on response time or on energy saving while the objective function J is minimized.

For the linearized model of the robot at the regular working speed $x_0 = 0.1\ m/s$, as shown in Figure 3.4.2, the open loop transfer function of the robot body has a form of a second order system

$$G(s) = \frac{x(s)}{F_p(s)} = \frac{0.13}{s(6.67s+1)} = \frac{0.13 \times 0.15}{s(s+0.15)} \qquad \textit{(Equation 5.2.4)}$$

Comparison of Equation 5.2.4 with the transfer function of a standard second order system

$$G(s) = \frac{\omega_{nl}^2}{s(s+2\xi_l\,\omega_{nl})} \qquad \textit{(Equation 5.2.5)}$$

gives the natural frequency of the robot $\omega_{nl}=0.14$ and damping ratio $\xi_l= 0.5$. From Figure 3.4.1 we notice that the drag force will increase as the speed increases. Therefore, the damping ratio of the robot will change with speed due to the nonlinearity. Equations 3.4.1 to 3.4.6 together with Equation 5.2.5 indicate that ξ_l will increase when the speed x_o increases. Obviously the dynamics of the robot would not be satisfactory.

We assume that the Optimized Linear Reference (OLR) model results from the linearized robot body model with an optimized state feedback as shown in Figure 5.2.1.

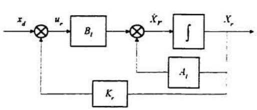


Figure 5.2.1 Structure of The Optimal Linear Reference Model

The state equation of the linearized robot body model with a state feedback is

$$\begin{aligned} \dot{X}_r(t) &= A_l X_r(t) + B_l u_r(t) \\ u_r(t) &= K_r X_r(t) + x_d(t) \end{aligned} \qquad \textit{(Equation 5.2.6)}$$

where

$$X_r(t) = [x_{r1}(t),\ \ x_{r2}(t)]^T$$

$$A_l = \begin{bmatrix} -2\xi_l\omega_{nl} & 0 \\ 1 & 0 \end{bmatrix}$$

$$B_l = \begin{bmatrix} \omega_{nl}^2 \\ 0 \end{bmatrix}$$

$$K_r = [k_{r1} \quad k_{r2}]$$

We assume that the Optimal Linear Reference (OLR) model has a form of

$$\dot{X}_r(t) = A_r X_r(t) + B_r x_d(t) \qquad \textit{(Equation 5.2.7)}$$

Substituting the second equation in Equation 5.2.6 into the first equation, and comparing with Equation 5.2.7 gives

$$\begin{aligned} A_r &= A_l + B_l K_r \\ B_r &= B_l \end{aligned} \qquad \textit{(Equation 5.2.8)}$$

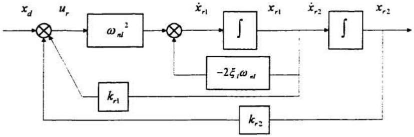


Figure 5.2.2 Detailed Structure of The Optimal Linear Reference Model

The detailed structure of the OLR model is shown in Figure 5.2.2. To determine the gain matrix K_r, the most effective and widely used technique is the linear quadratic method (LQ). The goal is to find a control input $u_r(t)=K_rX_r(t)$ which minimizes the quadratic cost function in Equation 5.2.2 or Equation 5.2.3 when system input $x_d(t)=0$.

There is a well-developed tool box in Matlab for design of such a state feedback gain K_r. In the depth control problem of NOMAD, the energy consumption should be emphasized. A control input $u_r(t)$ with longer period or larger amplitude is not preferred. Thus ϕ is picked in the range 0.3 to 0.5 for the design of the feedback gain matrix K_r. After a few trials and iterations, the following was obtained:

$$K_r = [k_{r1}, k_{r2}] = [-7.14, -2]$$

This gives the linear reference model parameters:

$$A_r = \begin{bmatrix} -2\xi_l\omega_{nl} + \omega_{nl}^2 k_{r1}, & +\omega_{nl}^2 k_{r2} \\ 1 & 0 \end{bmatrix} \quad . \qquad \textit{(Equation 5.2.9)}$$
$$B_r = B_l$$

Since the state feedback does not change the system order, the reference model is still a second order system (Brogan, 1991). The characteristic equation of the reference model has the following form:

$$\Delta_{ref}(s) = s^2 + 2\xi_r\omega_{nr}s + \omega_{nr}^2 \qquad \textit{(Equation 5.2.10)}$$

Comparison of the polynomial $|sI-A_r|$ with Equation 5.2.10 gives

$$\omega_{nr} = \omega_{nl}\sqrt{-k_{r2}} = 0.2$$
$$\xi_r = \frac{1}{2\omega_{nr}}(2\xi_l\omega_{nl} - \omega_{nl}^2 k_{r1}) = 0.707 \qquad \textit{(Equation 5.2.11)}$$

The location of the eigenvalues of the OLR model on the complex plane is shown in Figure 5.2.3. It is noticed that the reference model has a good dynamic performance. The overshoot σ_p of the system when it responds to a step input is

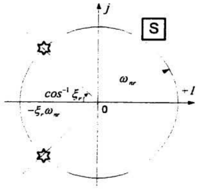


Figure 5.2.3 Location of the Eigenvalues of the Optimal Linear Reference Model

$$\sigma_p = e^{-\frac{\xi_r \pi}{\sqrt{1-\xi_r^2}}} \le 5\%$$

and the settle time is

$$t_s = \frac{3}{\xi_r \omega_{nr}} = \frac{3}{0.707 \times 0.2} \approx 21\,\text{sec}.$$

The step response of the OLR model is shown in Figure 5.2.4.

Theoretically speaking, a state feedback can locate the eigenvalues to any desired positions and make the settle time as small as wanted if the system can provide a control input $U(t)$ with a big enough amplitude (Lu, Tian and Chai 1991). In the case of NOMAD, the maximum control input to the plant is proportional to the maximum buoyancy and weight the ballast tank can provide. It is restricted by the capacity of the physical driving device. Since the objective function J in Equation 5.2.2 and 5.2.3 takes the control effect $U^T(t)RU(t)$ into account, the Linear Quadratic (LQ) design method gives an optimized state feedback gain K_r under this constraint.

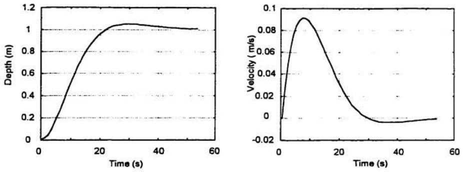


Figure 5.2.4 Response of the Optimal Linear Reference Model to a Unit Step Input

§ 5.3 Numeric Simulation For The Neural Network Based Adaptive Control .

Since the actual model of the subsea robot is highly nonlinear and uncertain, dynamic neural networks are used to identify the actual robot model and force the output of the actual robot to approach that of the OLR model. In this case the unknown nonlinear plant could have approximately the same dynamic performance as the OLR model.

1. Model Description

Two files contain the model information. The file *.rob* contains the actual model of the robot. The parameters in this model are isolated from the control system. The input and output of the model described by the file *.rob* are treated as the actual input and output of the subsea robot. The actual nonlinear model is

$$\begin{bmatrix}\dot{x}_{p1}\\ \dot{x}_{p2}\end{bmatrix}=\begin{bmatrix}x_{p2}\\ \frac{1}{M+a_2}\left(-a_1|x_{p2}|x_{p2}\right)\end{bmatrix}+\begin{bmatrix}0\\ \frac{1}{M+a_2}\end{bmatrix}F_p \qquad \textit{(Equation 5.3.1)}$$

where x_{p1}, x_{p2} are the state variables of the actual robot.

According to the conclusion in §3.2 $a_1=37.5$, $a_2=15$, $M=35$.

The file *.ref* contains the Optimal Linear Reference or OLR model, which has the desired dynamic performance. Substitution of Equation 5.2.9 into Equation 5.2.7 gives:

$$\begin{bmatrix}\dot{x}_{r1}\\ \dot{x}_{r2}\end{bmatrix}=\begin{bmatrix}-2\xi_l\omega_{nl}+\omega_{nl}^2k_{r1} & \omega_{nl}^2k_{r2}\\ 1 & 0\end{bmatrix}\begin{bmatrix}x_{r1}\\ x_{r2}\end{bmatrix}+\begin{bmatrix}\omega_{nl}^2\\ 0\end{bmatrix}x_d \qquad \textit{(Equation 5.3.2)}$$

where $\xi_l=0.5$, $\omega_{nl}=0.14$, $k_{r1}=-7.14$, $k_{r2}=-2$ are as calculated in section § 5.2.

A third order Runge-Kutta Algorithm is applied to solve Equation 5.3.1 and 5.3.2 and calculate the state variables which correspond to the inputs. Specifically the state X_r is the response of the reference model to the command input x_d, while the state X_p is the response of the actual robot model to the control input F_p, which is also the output of the neural network Adjuster (NNA).

2. Training the Neural Network Identifier (NNI)

To identify the actual model of the robot, a back propagation algorithm (IEEE, 1992a) is applied to train the Identifier (NNI), which has three layers. The hidden layer consists of 8 neurons with tangent sigmoidal functions. The output layer consists of 2 neurons with linear functions. The number of neurons in the output layer should equal to the number of outputs of the neural network.

The training procedure is shown in Figure 5.3.1, where:

T_s is the sample period, $T_s=0.1s$.

$x_{p1}(k)$ is the state variable, actual speed at the time kT_s.

$x_{p2}(k)$ is the state variable, actual depth at the time kT_s.

$F_p(k)$ is the force created by the ballast tank. When positive, it represents the gravity force, when negative it represents the buoyancy force.

$\Delta x_{p1}(k)$, $\Delta x_{p2}(k)$ are the actual increments of $x_{p1}(k)$, $x_{p2}(k)$ at the time kT_s.

$\Delta x_{m1}(k)$, $\Delta x_{m2}(k)$ are the estimated increments of $x_{p1}(k)$, $x_{p2}(k)$ at the time kT_s.

$e_{m1}(k)$, $e_{m2}(k)$ are the estimation error at the time kT_s.

$x_{m1}(k+1)$, $x_{m2}(k+1)$ are the estimated speed and depth at the $(k+1)T_s$.

If the neural network based controller were applied to the subsea robot, the values of the actual state variables could be directly obtained from the sensors. In the numerical simulation they are generated in the following way:

(1) Define the ranges of the input and state variables

$$x_{p1} \in [-speed\ max,\ speed\ max];$$

$$x_{p2} \in [\ depth\ min,\ \ depth\ max];$$

$$F_p \in [-force\ max,\ +force\ max].$$

(2) Divide each input and state variable respectively into some intervals, for example, divide the range of x_{p1} into n_1 intervals, the range of x_{p2} into n_2 intervals and the range of F_p into n_p intervals. The divisions do not have to be even. The

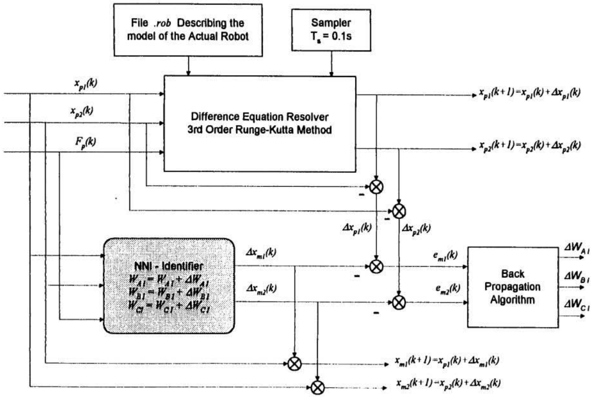

File .rob Describing the model of the Actual Robot
Sampler $T_s = 0.1s$
$x_{p1}(k)$
$x_{p2}(k)$
$F_p(k)$
Difference Equation Resolver 3rd Order Runge-Kutta Method
$x_{p1}(k+1)=x_{p1}(k)+\Delta x_{p1}(k)$
$x_{p2}(k+1)=x_{p2}(k)+\Delta x_{p2}(k)$
$\Delta x_{p1}(k)$
$\Delta x_{p2}(k)$
NNI - Identifier
$W_{A1}=W_{A1}+\Delta W_{A1}$
$W_{B1}=W_{B1}+\Delta W_{B1}$
$W_{C1}=W_{C1}+\Delta W_{C1}$
$\Delta x_{m1}(k)$
$\Delta x_{m2}(k)$
$e_{m1}(k)$
$e_{m2}(k)$
Back Propagation Algorithm
ΔW_{A1}
ΔW_{B1}
ΔW_{C1}
$x_{m1}(k+1)=x_{p1}(k)+\Delta x_{m1}(k)$
$x_{m2}(k+1)=x_{p2}(k)+\Delta x_{m2}(k)$

Figure 5.3.1 Digital Simulation Scheme of the Training Procedure for the Neural Network Identifier (NNI)

simulation indicates that smaller intervals around the equilibrium point, such as $x_{p1}=0$, $F_p=0$, will give a more precise neural network. For example, in Figure 5.3.2, x_{p2} is evenly divided, but x_{p1} and F_p are not. The intervals of x_{p1} around the point $x_{p1}=0$ and that of F_p around $F_p=0$ are much smaller.

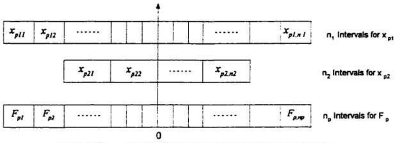


Figure 5.3.2 Divisions for the Ranges of the Training Variables

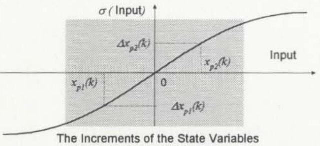
(3) Combining each interval in every input and state variable to simulate all the possible circumstances the subsea robot could encounter gives the training input for NNI.

$$NNI.TR_{input}=\begin{bmatrix} x_{p11} & x_{p11} \cdots x_{p11} & x_{p11} & x_{p11} \cdots x_{p11} & \cdots & x_{p1n_1} & x_{p1n_1} & \cdots x_{p1n_1} \\ x_{p21} & x_{p21} \cdots & x_{p21} & x_{p2j} & x_{p2j} \cdots x_{p2j} & \cdots & x_{p2n_2} & x_{p2n_2} & \cdots x_{p2n_2} \\ F_{p1} & F_{p2} \cdots F_{pn_p} & F_{p1} & F_{p2} \cdots F_{pn_p} & \cdots & F_{p1} & F_{p2} & \cdots F_{pn_p} \end{bmatrix}_{3\times(n_1\times n_2\times n_p)}$$

(Equation 5.3.3)

which is a $3\times(n_1\times n_2\times n_p)$ matrix.

To avoid wildly fluctuating training data, incremental changes in state variables Δx_{p1}, Δx_{p2} are used instead of the absolute values of x_{p1}, x_{p2} as the training target. Then the data would drop into the flat ranges of sigmoidal functions as shown in Figure 5.3.3, which accelerates convergence of the training process.



The Increments of the State Variables most likely drop in the Shadow Area

Figure 5.3.3 Flat Portion of the Sigmoid Function (Shadow Area)

The solution of the difference equation of the actual robot gives the training target:

$$NNI.TR_{Target} = \begin{bmatrix} \Delta x_{p1,1} \cdots \Delta x_{p1,n_p} & \cdots \Delta x_{p1,n_p \times n_2} & \cdots \Delta x_{p1,n_1 \times n_2 \times n_p} \\ \Delta x_{p2,1} \cdots \Delta x_{p2,n_p} & \cdots \Delta x_{p2,n_p \times n_2} & \cdots \Delta x_{p2,n_1 \times n_2 \times n_p} \end{bmatrix}_{2 \times (n_1 \times n_2 \times n_p)}$$

(Equation 5.3.4)

When the Difference Equation Resolver and the Identifier NNI are fed with the same training input, their responses $\Delta X_p(k)$ and $\Delta X_m(k)$ are compared. The errors e_{1m} and e_{2m} are used in the back propagation algorithm to calculate the modification matrices ΔW_{AI}, ΔW_{BI}, ΔW_{CI} to update the weight matrices in the Identifier

$$\begin{aligned} W_{AInew} &= W_{AIold} + \Delta W_{AI} \\ W_{BInew} &= W_{BIold} + \Delta W_{BI} \\ W_{CInew} &= W_{CIold} + \Delta W_{CI} \end{aligned}$$

(Equation 5.3.5)

In the numerical simulation, the training pairs $NNI.TR_{input}$ and $NNI.TR_{target}$ are organized in batch form. A training performance index is defined as the sum of squared errors:

$$SSE_{INN} = \sum_{j=1}^{2} \sum_{k=1}^{n_1 \cdot n_2 \cdot n_p} e_{mj}^2(k) \qquad \textit{(Equation 5.3.6)}$$

where $e_{mj}(k)$ is the error in the estimation of the jth state variable for the kth pair of training data in the matrices $NNI.TR_{input}$ and $NNI.TR_{target}$.

When the training process is ongoing, SSE should decrease. As soon as it falls down below a satisfactory level, the training process is terminated.

In the simulation, the following parameters are specified for training the Identifier NNI:

(1) Speed range $x_{p1}(k) = [-2, 2]$ m/s, $n_1 = 11$;

(2) Depth range $x_{p2}(k) = [0, 8]$ m, $n_2 = 8$;

(3) Driving Force $F_p(k) = [-60, 60]$ N, $n_p = 17$;

(4) Sampling period $T_s = 0.1s$;

Figure 5.3.4 shows SSE as it drops down very quickly at the start of training and slows down later. The training process stops when SSE is without significant decrease.

When the neural network NNI is well trained, it should have precisely the same time response as the actual robot, no matter if the input force or state variables are in the training pair $NNI.TR_{Input}$ and $NNI.TR_{Target}$ or not. Figure 5.3.5 shows typical time responses of the actual robot model and the neural network NNI model. They show that the neural network NNI has stored the dynamic information of the actual robot in the weight matrices W_{AI}, W_{BI} and W_{CI}.

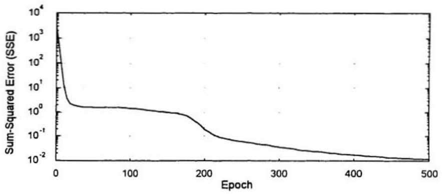


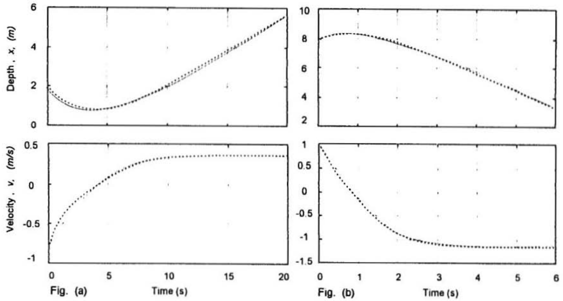
Figure 5.3.4 Sum-Squared Error in The Training of The Neural Network Identifier

3. Training the Neural Network Adjuster (NNA)

The neural network based Adjuster NNA also has three layers. The hidden layer consists of 6 neurons. The output layer has only one neuron, since there is only one output F_p from the Adjuster. Tangent sigmoidal functions and linear functions are adopted for the hidden layer and output layer respectively. The same sampling period as in the NNI $T_s=0.1$ is adopted for training NNA. The training scheme for NNA is shown in Figure 5.3.6, where:

$x_{r1}(k)$, $x_{r2}(k)$ are the state variables in the Optimal Linear Reference or OLR model at the time kT_s;

$x_{r1}(k+1)$, $x_{r2}(k+1)$ are the state variables in OLR at the time $(k+1)T_s$ responding to the



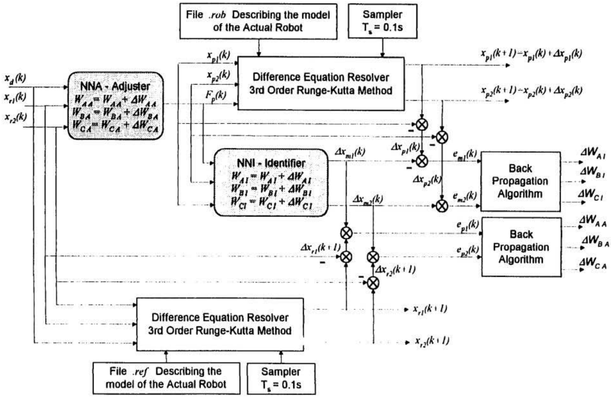
(a) Under the Constant Driving Force F_d= *5N* and Initial Condition x(0)=2m and v(0)= -0.8 m/s

(b) Under the Constant Driving Force F_d= *50N* and Initial Condition x(0)=8m and v(0)=1 m/s

Responses of the Actual Robot Model

Output of the Neural Network Identifier After Well Trained

Figure 5.3.5 Comparison of the Outputs of the Actual Robot Model and That of the Neural Network Identifier (NNI)



File .rob Describing the model of the Actual Robot
Sampler $T_s = 0.1s$
$x_d(k)$
$x_{r1}(k)$
$x_{r2}(k)$
NNA - Adjuster
$W_{AA} = W_{AA} + \Delta W_{AA}$
$W_{BA} = W_{BA} + \Delta W_{BA}$
$W_{CA} = W_{CA} + \Delta W_{CA}$
$x_{p1}(k)$
$x_{p2}(k)$
$F_p(k)$
Difference Equation Resolver 3rd Order Runge-Kutta Method
$x_{p1}(k+1) = x_{p1}(k) + \Delta x_{p1}(k)$
$x_{p2}(k+1) = x_{p2}(k) + \Delta x_{p2}(k)$
NNI - Identifier
$W_{AI} = W_{AI} + \Delta W_{AI}$
$W_{BI} = W_{BI} + \Delta W_{BI}$
$W_{CI} = W_{CI} + \Delta W_{CI}$
$\Delta x_{m1}(k)$
$\Delta x_{m2}(k)$
$\Delta x_{p1}(k)$
$\Delta x_{p2}(k)$
$e_{m1}(k)$
$e_{m2}(k)$
$e_{p1}(k)$
$e_{p2}(k)$
Back Propagation Algorithm
ΔW_{AI}
ΔW_{BI}
ΔW_{CI}
Back Propagation Algorithm
ΔW_{AA}
ΔW_{BA}
ΔW_{CA}
$\Delta x_{r1}(k+1)$
$\Delta x_{r2}(k+1)$
$x_{r1}(k+1)$
$x_{r2}(k+1)$
Difference Equation Resolver 3rd Order Runge-Kutta Method
File .ref Describing the model of the Actual Robot
Sampler $T_s = 0.1s$

Figure 5.3.6 Digital Simulation Scheme of the Training Procedure for the Neural Network Adjuster (NNA)

state variables $x_{r1}(k)$, $x_{r2}(k)$ and depth set point $x_d(k)$ at the time kT_s. (They are calculated by the Runge-Kutta method based on the OLR model contained in the file *.ref*);

$\Delta x_{r1}(k)$, $\Delta x_{r2}(k)$ are the increments of state variable $x_{r1}(k)$ and $x_{r2}(k)$ at time kT_s.

$e_{p1}(k)$, $e_{p2}(k)$ are the network error used by back propagation algorithms to calculate ΔW_{AA}, ΔW_{BA}, and ΔW_{CA}, which update the weight matrices W_{AA}, W_{BA} and W_{CA}:

$$\begin{aligned} W_{AAnew} &= W_{AAold} - \Delta W_{AA}; \\ W_{AAnew} &= W_{BAnew} - \Delta W_{BA}; \\ W_{CAnew} &= W_{CAold} - \Delta W_{CA}. \end{aligned} \qquad \textit{(Equation 5.3.7)}$$

The pairs of inputs and targets for training the Adjuster are organized in a batch form in the simulation.

The training ranges and the number of divisions for each are shown in Table 5.3.1.

Table 5.3.1 The Training Ranges and Divisions

Variables	Range	Number of Divisions
Speed $x_{r1}(k)$, $x_{p1}(k)$	$[-2,\ 2]$ m/s	$n_1 = 11$
Depth $x_{r2}(k)$, $x_{p2}(k)$	$[0,\ 8]$ m	$n_2 = 8$
Set point for Depth $x_d(k)$	$[1,\ 7]$ m	$n_d = 12$

Combination of each interval of $x_{r1}(k)$, $x_{r2}(k)$ and $x_d(k)$ gives the input matrix for training NNA:

$$NNA.TR_{Input} = \begin{bmatrix} x_{r11} \cdots x_{r11} & \cdots & x_{r11} \cdots x_{r11} & \cdots & x_{r1n_1} \cdots x_{r1n_1} \\ x_{r21} \cdots x_{r21} & \cdots & x_{r2j} \cdots x_{r2j} & \cdots & x_{r2n_2} \cdots x_{r2n_2} \\ x_{d1} \cdots x_{dn_d} & \cdots & x_{d1} \cdots x_{dn_d} & \cdots & x_{d1} \cdots x_{dn_d} \end{bmatrix}_{3 \cdot (n_1 \cdot n_2 \cdot n_d)}$$

(Equation 5.3.8)

The solution of the difference equation of the reference model gives the target matrix for training the neural network Adjuster (NNA)

$$NNA.TR_{Target} = \begin{bmatrix} \Delta x_{r11} \cdots\cdots\cdots \Delta x_{r1.(n_1 \cdot n_2 \cdot n_d)} \\ \Delta x_{r21} \cdots\cdots\cdots \Delta x_{r2.(n_1 \cdot n_2 \cdot n_d)} \end{bmatrix}_{2 \cdot (n_1 \cdot n_2 \cdot n_d)}$$ *(Equation 5.3.9)*

As in the training for the Identifier (NNI), an SSE index is defined to indicate the training performance for the Adjuster NNA

$$SSE_{NNA} = \sum_{i=1}^{2} \sum_{k=1}^{n_1 \cdot n_2 \cdot n_d} e_{pi}(k)$$ *(Equation 5.3.10)*

Figure 5.3.7 shows SSE_{NNA} versus the training epochs. It is noticed that the convergence speed in training the Adjuster Network (NNA) is slower than in training the Identifier Network (NNI). The process of training the Adjuster (NNA) is based on the trained Identifier (NNI): the neural network model error in NNI greatly influences the speed and accuracy in training NNA.

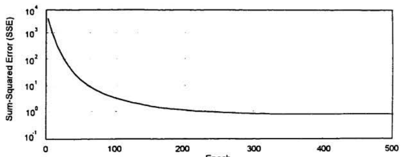
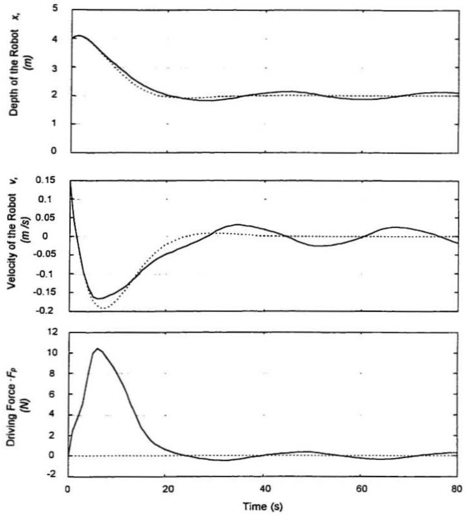


Figure 5.3.7 Sum-Squared Error in The Training of The Neural Network Adjuster

Typical time responses of the neural network based depth control system and the Optimal Linear Reference or OLR model to a step depth set point input are shown in Figures 5.3.8 and 5.3.9. It is noticed that the output of the actual robot is approximately the same as that of the OLR model. The minor oscillation with an amplitude 0.15 m in the steady state is caused by the training error, mainly in the process for training the net of NNA. If more data were integrated into the training pairs, $NNA.TR_{Input}$ and $NNA.TR_{Target}$, and if more neurons were adopted into the hidden layer of the net NNA, the amplitude of the oscillation would be much smaller. However the training time would increase greatly. Figure 5.3.10 shows the time responses of the actual robot and the OLR model to a sine wave input.

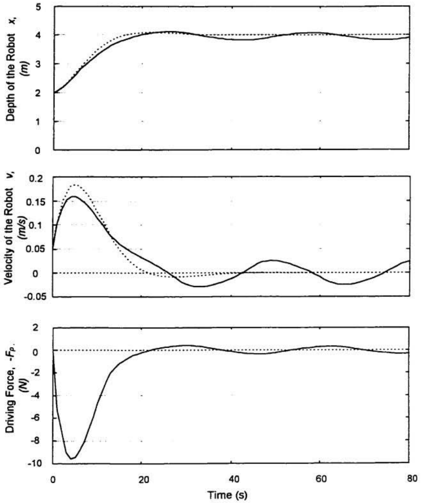
The work for the neural network controller which was done in this PhD project is based on off-line numerical simulation. Results indicate that the Neural Network Based Adaptive Controller has great potential to treat unknown nonlinear models such as that of a subsea robot. It drives the unknown nonlinear model to approach the dynamic performance of an Optimal Linear Reference model. However, there is still a lot of further work involved in applying this technology to the subsea robot control. The most important thing is to significantly increase the training speed when neural networks are applied to on-line model identification and network training. Some specially designed VLSI chips using hardware logic to implement the back propagation training algorithm would greatly increase the computing speed, which indicates that practical applications of the neural network based controllers are possible. However, it would increase the cost of the robot, and therefore, go against the objective of a low cost robot design.



Responses of the Optimal Linear Reference Model

Responses of the Subsea Robot with the Neural Network Based Controller

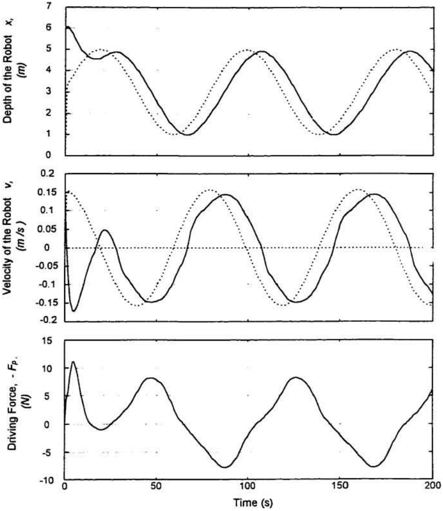
Figure 5.3.8 Step Responses and Driving Force of the Subsea Robot with the Neural Network Based Controller Under the Initial Condition x(0)=4, v(0)=0.15



······ Responses of the Optimal Linear Reference Model

——— Responses of the Subsea Robot with the Neural Network Based Controller

Figure 5.3.9 Step Responses and Driving Force of the Subsea Robot with the Neural Network Based Controller Under the Initial Condition x(0)=2, v(0)=0.05



┈┈┈ Sine Form Reference Input to the Subsea Robot, $x_d = 3 + 2\sin(0.025\pi t)$

——— Responses of the Subsea Robot with the Neural Network Based Controller

Figure 5.3.10 Responses and Driving Force of the Subsea Robot with the Neural Network Based Controller to a Sine Form Reference Input Under the Initial Condition x(0)=6, v(0)=0.15

CHAPTER 6 FUZZY VARIABLE STRUCTURE SWITCHING CONTROL SCHEME

Fuzzy logic is presently being developed extensively for control for plants with complex and often not precisely known dynamics (Lee, 1990). In almost all cases, the basis of a fuzzy control law is heuristic knowledge about a suitable control strategy. This knowledge is usually acquired by interviewing operators who have a lot of experience in the manual control of the plant. However, when this is not available, it can also come from numerical simulation (Togai, 1991).

Because there are some inherent defects in classical control strategies and the Neural Network Based Control System is computationally expensive, a more reliable and economical control scheme, Fuzzy Variable Structure Switching (FVSS) Control, is put forward and detailed in this Chapter. This strategy causes the robot to undergo a self-excited oscillation known as a limit cycle. To enhance the system stability and diminish the amplitude of the limit cycle, hysteresis and phase lead compensation are introduced into this control strategy.

Digital simulation in the time domain and the describing function method of analysis in the frequency domain are applied to the fuzzy logic controller with and without hysteresis and phase lead compensation. The study indicated that this control scheme is not only stable and robust but also energy saving. Because of this, it was then adapted to be physically installed on the robot.

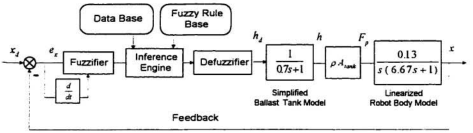


Figure 6.1.1 Scheme of the Fuzzy Logic Controller

§ 6.1 General Description for the Fuzzy Control Scheme

For the depth control of the subsea robot NOMAD, a possible fuzzy controller is shown in Figure 6.1.1. Moving the block of coefficient ρA_{tank} backward and including it in the main controller, the equivalent block diagram could be drawn as shown in Figure 6.1.2, where A_{Tank} is the cross sectional area of the ballast tank and ρ is the density of sea water.

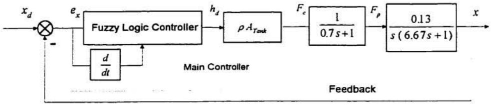


Figure 6.1.2 Equivalent Block Diagram After Moving the Gain ρA_{Tank} Backward

The fuzzy controller has two inputs:

(1) the error of the depth $e_x(k) = x_d(k) - x(k)$;

(2) the derivative of the error $\dot{e}(k) = \dot{x}_d(k) - \dot{x}(k)$;

and one output, the desired driving force $F_c(k)$. It is a buoyancy force when $F_c(k) < 0$, and a weight force when $F_c(k) > 0$. Obviously when $F_c(k) = 0$, the robot should maintain a neutral buoyancy status.

Fuzzy sets are defined for each of these input and output variables. Usually crisp, triangular, trapezoidal or quadratic membership functions are used to indicate the degree of membership to which a variable belongs to a fuzzy set, as shown in Figure 6.1.3.

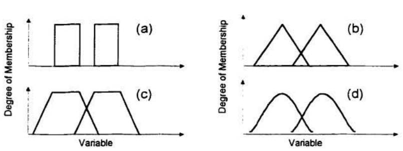


Figure 6.1.3 Different Definitions of the Membership Function

Note that for trapezoidal fuzzy sets, increasing the slope in the membership function (Figure (c)) causes the fuzzy set to approximate a Boolean or crisp set. Triangular fuzzy sets have proven popular with fuzzy logic practitioners. A reason for this is that the extra smoothness introduced by higher order fuzzy sets is not strongly reflected in the output quality of a fuzzy model. In many circumstances, when the states of the controlled plant are far away from their equilibrium, the system does not exhibit significant difference with the different definitions for the fuzzy membership function as shown in Figure 6.1.3. This indifference makes it possible to simplify the process of fuzzification and defuzzification, and even to simplify driving devices. For the depth control of NOMAD, a fuzzy logic controller with triangular fuzzy membership functions for each input and output variable was studied using numerical simulation. Following this the fuzzy membership function for the output variable of the controller was changed from a triangular shape to a trapezoidal shape and then approximated by a Boolean or crisp membership function. The results of digital simulation indicate that it is possible to use a discontinuous or crisp driving device such as solenoid valve to substitute for a continuous driving device such as servo valve if a proper structure of the controller is installed.

§ 6.2 Fuzzy Depth Control with Triangular Membership Function

Figure 6.2.1 shows the range and the definition of fuzzy sets for each input and output variable. There are five fuzzy sets, NB, NS, Z, PS, PB defined for the error, $e_x(k) = x_d(k)-x(k)$; three for the speed $v_{rob}(k)$ and five for the output $F_c(k)$. The range for error $e_x(k)$ is $(-10, 10)m$; for speed $v_{rob}(k)$ is $(-2, 2)m/s$; for force $F_c(k)$ is $(-60, 60)N$. A fuzzy set is a linguistic description of a variable. Here, for example, NB means "negative big", NS means "negative small", Z means "zero" and PS means "positive small" and so on.

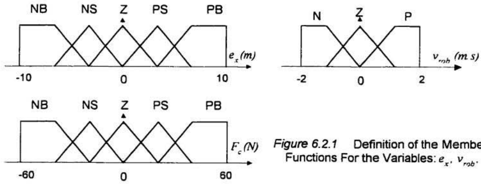


Figure 6.2.1 Definition of the Membership Functions For the Variables: e_x, v_{rob}, F_c.

The degree of the membership, which is represented by the vertical axis with a range from 0 to 1, is calculated according the given value of the input variable and the membership function. It could be denoted as:

$$\mu_{LV}(\, e_x(k)\,) \in [0,1]$$

where LV is the name of the fuzzy set and $e_x(k)$ is the value of the input variable. $\mu_{LV}(\, e_x(k)$ is the degree to which the value of the variable $e_x(k)$ belongs to the fuzzy set "LV". For example when $e_x(k)=1.5$, we can either say the error is "positive small" or "positive big". So this value of $e_x(k)$ belongs to two fuzzy sets: "PS" , "PB". However the degrees of membership are different. In the case of $e_x=e_1$, as shown in Figure 6.2.2, we have

$$\mu_{PB}[e_x(k)=e_1]=0.3 \;;$$

$$\mu_{PS}[e_x(k)=e_1]=0.9\,.$$

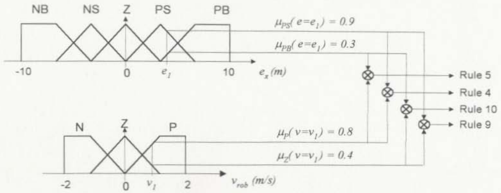


Figure 6.2.2 The Reasoning Process used in the Depth Controller of NOMAD Robot

Because of the overlap of the fuzzy membership functions, a variable with a given value often belongs to more than one fuzzy sets.

For the numerical simulation of the NOMAD, a rule base was built as shown in Table 6.2.1.

Table 6.2.1 Fuzzy Knowledge Rule Base for Depth Control

F_c		e_x				
		NB	NS	Z	PS	PB
v_{rob}	P	[1] NB	[2] NB	[3] NS	[4] PS	[5] PS
	Z	[6] NB	[7] NS	[8] Z	[9] PS	[10] PB
	N	[11] NS	[12] NS	[13] PS	[14] PB	[15] PB

The rule base consists of 15 IF (... AND ...) THEN statements.

Rule 1	IF	*error*	IS	*negative big*
	AND	*speed*	IS	*positive*
	THEN	*force*	IS	*negative big*
......				
Rule 8	IF	*error*	IS	*zero*
	AND	*speed*	IS	*zero*

	THEN	*force*	IS	*zero*
......				
Rule 15	IF	*error*	IS	*positive big*
	AND	*speed*	IS	*negative*
	THEN	*force*	IS	*positive big*

With triangular fuzzy membership functions, the rule base can be described by the 3D plot shown in Figure 6.2.3.

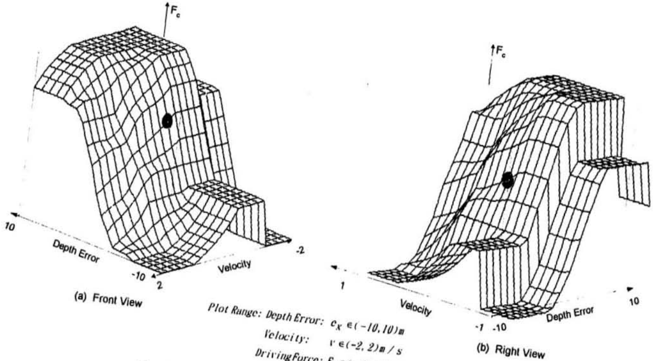
There are usually three basic fuzzy logic operators: *and, or, not.* They are defined as follows:

$$\mu_{A\ and\ B}(e(k)=e_o)=\min\{\mu_A(e(k)=e_o),\ \mu_B(e(k)=e_o)\}$$
$$\mu_{A\ or\ B}(e(k)=e_o)=\max\{\mu_A(e(k)=e_o),\ \mu_B(e(k)=e_o)\}$$
$$\mu_{\bar{A}}(e(k)=e_o)=1-\mu_A(e(k)=e_o)$$

where,

max, min	are the maximum and minimum operators;
$\mu_{A\ and\ B}(e(k)=e_o)$	is the degree of membership to which $e(k)=e_o$ belongs to the intersection of fuzzy sets A and B ;
$\mu_{A\ or\ B}(e(k)=e_o)$	is the degree of membership to which $e(k)=e_o$ belongs to the union of fuzzy sets A and B ;
$\mu_{\bar{A}}(e(k)=e_o)$	is the degree of membership to which $e(k)=e_o$ does not belong to the fuzzy set A.

Control processes are usually associated with system inputs and outputs. In a fuzzy system, input/output mappings are made from one universe of discourse U to another Y. For the depth control of NOMAD, U has two dimensions: one is depth



Plot Range: Depth Error: $e_X \in (-10, 10)m$
Velocity: $v \in (-2, 2)m/s$
Driving Force: $F_c \in (-60, 60)N$

Figure 6.2.3 3D Plot of the Fuzzy Logic Rule Base of the Depth Controller.

error $e_x(k)$, the other is speed $v_{rob}(k)$. Y has only one dimension, that is, the desired driving force $F_c(k)$, which is proportional to the water level in the ballast tank.

The above three operators are used to quantify the imprecision of the inputs to the control process. Assuming that $\mu_{A_i}(u_i)$ ($i=1,\dots,n$) is the degree to which u_i belongs to the fuzzy set A_i, the rules map the logic combination of these fuzzy sets A_i to the output universe Y, according to:

Rule i:	IF	(A_1 *and* A_2 ... *and* A_n)	THEN C_i
Rule j:	IF	(A_1 *or* A_2 ... *or* A_n)	THEN C_j
Rule k:	IF	(*not* A_1)	THEN C_k

where C_i, C_j, C_k are the fuzzy sets in the universe Y. Similar to the definition of degree of membership, which describes the degree to which the value of a variable belongs to a fuzzy set, a real number $\mu_{LV}^{o}(R_i) \in [\ 0,\ 1\ ]$ is defined as the belief level at which the mapping result of the rule R_i belongs to the fuzzy set LV in the output universe Y.

The belief levels of the mapping results of rules i, j, k can be calculated as follows (Bielawski and Lewand, 1991):

$$\mu_{c_i}^{o}(R_i) = \min\{\mu_{A_1}(u_1), \mu_{A_2}(u_2) \cdots \mu_{A_n}(u_n)\}$$
$$\mu_{c_j}^{o}(R_j) = \max\{\mu_{A_1}(u_1), \mu_{A_2}(u_2) \cdots \mu_{A_n}(u_n)\}$$
$$\mu_{c_k}^{o}(R_k) = 1 - \mu_{A_1}(u_1)$$

The output of a fuzzy controller y is calculated in the defuzzification process. Several different approaches are used in practice (Berenji, 1992).

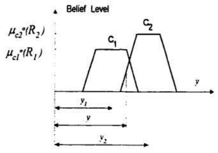


Figure 6.2.4 The Output of the Weight Average Defuzzifier

The weighted average defuzzification procedure takes the centroid of area (COA) of the aggregated fuzzy set C_1 and C_2 as the output of the controller. This is given by

$$y = \frac{\sum s_i y_i}{\sum s_i}$$

where as shown in Figure 6.2.4, s_i is the area of the cropped trapezoid associated with the fuzzy set C_i; y_i is the centroid of the area s_i; y is the centroid of the aggregated areas.

The reasoning process of the fuzzy controller used for the depth control of NOMAD is illustrated in Figure 6.2.2. Assume that at the time $t=kT$, we know from sensor readings $e_x(k)=e_1$ and $v_{rob}(k)=v_1$. Then from Figure 6.2.2 it follows that:

$\mu_{ps}(e_x=e_1)=0.9$ is the degree of membership to which e_1 belongs to "PS" ;
$\mu_{PB}(e_x=e_1)=0.3$ is the degree of membership to which e_1 belongs to "PB" ;
$\mu_p(v=v_1)=0.8$ is the degree of membership to which v_1 belongs to "P" ;
$\mu_z(v=v_1)=0.4$ is the degree of membership to which v_1 belongs to "Z" .

These cause four rules to fire:

Rule 4:	IF (*error* IS *positive small*) AND (*speed is positive*) THEN *force* IS *positive small* ;
Rule 9:	IF (*error* IS *positive small*) AND (*speed is zero*) THEN *force* IS *positive small* ;
Rule 5:	IF (*error* IS *positive big*) AND (*speed is positive*) THEN *force* IS *positive small* ;
Rule 10:	IF (*error* IS *positive big*) AND (*speed is zero*) THEN *force* IS *positive big* .

From Figure 6.2.2 belief levels for these rules are as follows:

Rule 4:	$\mu^o_{PS}(R_4) = \min(\mu_{PS}(e_z = e_1), \mu_P(v = v_1)) = 0.8$;
Rule 5:	$\mu^o_{PS}(R_5) = \min(\mu_{PB}(e_z = e_1), \mu_P(v = v_1)) = 0.3$;
Rule 9:	$\mu^o_{PS}(R_9) = \min(\mu_{PS}(e_z = e_1), \mu_Z(v = v_1)) = 0.4$;
Rule10:	$\mu^o_{PB}(R_{10}) = \min(\mu_{PB}(e_z = e_1), \mu_Z(v = v_1)) = 0.3$.

The output of the fuzzy controller, the driving force F_c, follows from:

$$F_c = \frac{\sum_{i=4,5,9,10} s_i y_i}{\sum_{i=4,5,9,10} s_i}$$

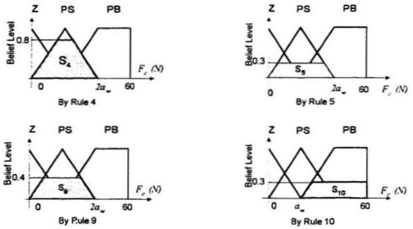


Figure 6.2.5 Mapping Results of the Rules of 4, 5, 9, 10.

The enclosed areas s_i and centroids of these areas y_i are shown in Figure 6.2.5.

$$
\begin{aligned}
s_i &= \left[(1-\mu^o_{PS}(R_i))\,a_w + a_w\right]\mu^o_{PSi}(R_i) && i = 4, 5, 9 \\
s_{10} &= \left(b_w - \left(\frac{1}{2}a_w\,\mu^o_{PS}(R_{10}) + a_w\right)\right)\mu^o_{PS}(R_{10}) \\
y_i &= a_w && i = 4, 5, 9 \\
y_{10} &= \frac{1}{2}\left(b_w + \left(\frac{1}{2}a_w\,\mu^o_{PS}(R_{10}) + a_w\right)\right)
\end{aligned}
$$

where, a_w is the half width of triangle which defines the membership function for output variable F_c. In simulation, a_w was tried out between 15 ~ 25; b_w is the maximum output of F_c, which was set at 60.

The program for simulating the depth control of the subsea robot NOMAD was implemented using the Fuzzy Programming Language (FPL), which was developed by the Togai InfraLogic Inc (Togai, 1993). The controller exhibits a good dynamic behavior. Typical step responses are shown in Figure 6.2.6. As can be seen, the settle time, the overshoot and steady state error are all satisfactory.

§ 6.3 Fuzzy Variable Structure Switching Controller With a Discontinuous Output

Besides reliability, another important object sought in the design of the subsea robot NOMAD is low cost. As mentioned in Chapter 2, a water level detector is installed in the ballast tank. The voltage output from this is used in a feedback loop to control the water level in the ballast tank.

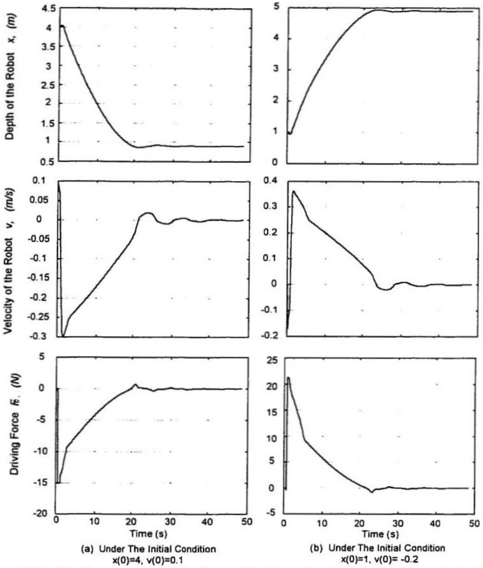


Figure 6.2.6 **Step Responses and Driving Force of the Subsea Robot With the Fuzzy Logic Controller Which Has Triangular Membership Functions and A Continuous Driving Force**

Instead of adopting a continuous water level detector, to enhance the reliability and keep low cost, a column of float micro switches was installed in the ballast tank as shown in Figure 6.3.1. The switches indicate 10 water levels. These are SWION, SWIOFF, SW2ON, SW2OFF, ..., SW5ON, SW5OFF. Efforts were made to modify the fuzzy controller discussed in Section §6.2 to meet the design specifications of this low cost and reliable setup.

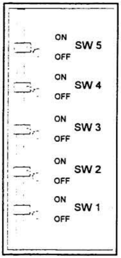


Figure 6.3.1 Float Micro Switches Installed in the Ballast Tank

§ 6.3.1 Problems Raised by the Discontinuous Output of the Fuzzy Controller

For the depth control of the robot NOMAD, the output variable $F_c(k)$ of the fuzzy controller has five linguistic grades: NB, NS, Z, PS, PB. These linguistic grades are assigned to the five micro switches from the top to the bottom of the ballast tank as shown in Figure 6.3.1 and Figure 6.3.2. The sets defined in Figure 6.3.2 form a group of crisp sets. The width for each is the dead zone of the micro switch.

For these sets, the "Maxima" defuzzification scheme is adapted (Cox, 1992):

$$F_c(k) = LV \left|\left\{\mu^o_{LV}(R) = \max(\mu^o_{LV_i}(R_i))\right\}\right| \qquad i = 1 \cdots 15$$

where, LV is the linguistic value of the output variable $F_c(k)$. The candidates are NB, NS, Z, PS, PB; R_i is the mapping result by the *ith* rule.

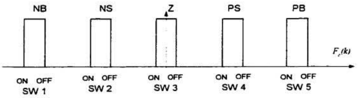


Figure 6.3.2 The Crisp Sets of the Output Variable $F_c(k)$

When there are several rules mapping to a same result, that is,

$$LV_i = LV_{i-1} = \dots = LV_{i-l}$$

the belief level for this result is given by

$$\mu^o_{LV}(R) = 1 - \prod_{k=i}^{i-l} (1 - \mu^o_{LV_k}(R_k))$$

For example, for the case shown in Figure 6.2.4, the belief level of the mapping results from rules 4, 5, 9 is

$$\begin{aligned}\mu^o_{PS}(R) &= 1 - (1 - \mu^o_{PS}(R_4))(1 - \mu^o_{PS}(R_5))(1 - \mu^o_{PS}(R_9)) \\ &= 1 - (1 - 0.3)(1 - 0.8)(1 - 0.4) \\ &= 0.916\end{aligned}$$

The linguistic value of the output of the fuzzy controller is

$$F_c = LV\left|\left\{\mu^o_{LV}(R) = \max(\ \mu^o_{PS}(R) = 0.916,\ \mu^o_{PB}(R_{10}) = 0.3\ )\right\}\right.$$
$$LV = \text{PS}, \quad \textit{that is } F_c = \text{PS}.$$

The defuzzification process in a traditional fuzzy controller smoothens the output of the controller. However, due to the adoption of micro switches for water level detection in the ballast tank, crisp sets are defined for the output variable $F_c(k)$. The driving force acting on the body of the robot becomes a discrete variable with only 5 values: namely NB, NS, Z, PS, PB as shown in Figure 6.3.3.

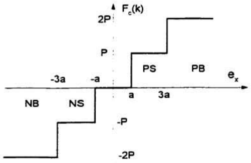


Figure 6.3.3 The Stair Case Nonlinearity Introduced by the Discontinuous Output of the Controller

Obviously the controller is highly nonlinear. Jumps like those shown in Figure 6.3.3 tend to make systems oscillate. This degrades system stability. Describing Functions and the Nyquist Procedure can be used to analyze such a system (Hinchey and Goteti, 1991). When a nonlinear system is unstable, nonlinearities usually limit the amplitude of oscillations to some finite level. The system is said to undergo a self excited oscillation or limit cycle. Suppose that the system is undergoing such an oscillation and suppose also that the signal being fed back around the control loop is a pure sinusoid. When the command signal is zero, the signal into the controller could be expressed as

$$x_{in}(t) = A \sin\omega t \qquad \textit{(Equation 6.3.1)}$$

where A is the amplitude of the oscillation and ω is the frequency of the oscillation.

The output from the controller could be very complex. However, by Fourier Series analysis, it can be broken down into a fundamental component and an infinite number of higher harmonics:

$$\begin{aligned} x_{out}(t) &= \sum_{k=1}^{\infty} (A_k \cos k\omega t + B_k \sin k\omega t) \\ &= A_1 \cos\omega t + B_1 \sin\omega t + \textit{Higher Harmonics} \\ &\approx C_1 \sin(\omega t + \theta_1) \end{aligned} \qquad \textit{(Equation 6.3.2)}$$

where

$$C_1 = \sqrt{A_1^2 + B_1^2} \qquad A_k = \frac{2}{\pi}\int_o^{\pi} x_2(t)\cos k\omega t \, d(\omega t)$$
$$\theta_1 = tg^{-1}\frac{A_1}{B_1} \qquad B_k = \frac{2}{\pi}\int_o^{\pi} x_2(t)\sin k\omega t \, d(\omega t) \qquad \textit{(Equation 6.3.3)}$$

Using complex notations, Equations 6.3.1 and 6.3.2 can be written as:

$$\begin{aligned} X_{in}(A,\ \omega) &= A \\ X_{out}(A,\ \omega) &= C_1 e^{j\theta 1} \end{aligned} \qquad \textit{(Equation 6.3.4)}$$

This gives the transfer function $N(A,\omega)$,

$$N(A,\omega) = \frac{X_{out}(A,\omega)}{X_{in}(A,\omega)} = \frac{C_1}{A}e^{j\theta_1} = \frac{\sqrt{A_1^2 + B_1^2}}{A}\angle tan^{-1}\frac{A_1}{B_1} \qquad \textit{(Equation 6.3.5)}$$

which is defined as the describing function of the nonlinearity.

The assumption that the signal fed back is a pure sinusoid is usually a good one because the linear parts of the system which follow the nonlinear controller normally act as a low pass filter and remove the higher harmonics.

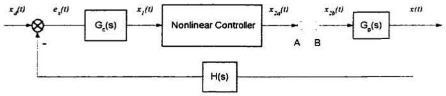


Figure 6.3.4 A System With a Nonlinear Controller

Now consider the system shown in Figure 6.3.4. Assume $x_d(t)=0$ and let the input to the plant $G_p(s)$ be:

$$x_{2b}(t)=A_{in}\sin\omega t \qquad \textit{(Equation 6.3.6)}$$

The signal fed back to the input of the nonlinear controller would be:

$$x_1(t)=-|G(j\omega)|A_{in}\sin[\omega t+\angle G(j\omega)] \qquad \textit{(Equation 6.3.7)}$$

where,

$$\begin{aligned}|G(j\omega)| &= |G_p(j\omega)G_c(j\omega)H(j\omega)|\\ \angle G(j\omega) &= \angle G_p(j\omega)+\angle G_c(j\omega)+\angle H(j\omega)\end{aligned}$$

If the describing function of the nonlinear controller is

$$N(A,\omega)=|N(A,\omega)|\,e^{j\angle N(A,\omega)} \qquad \textit{(Equation 6.3.8)}$$

then the output of the controller is

$$x_{2a}(t)=-|N(A,\omega)||G(j\omega)|A_{in}\sin[\omega t+\angle G(j\omega)+\angle N(A,\omega)] \qquad \textit{(Equation 6.3.9)}$$

If $x_{2a}(t)=x_{2b}(t)$, the system will oscillate. Comparison of Equation 6.3.6 with Equation 6.3.9 gives the conditions for existence of such an oscillation

$$\begin{cases} |N(A,\omega)||G(j\omega)| = 1 \\ \angle N(A,\omega) + \angle G(j\omega) = (2n+1)\pi \end{cases} \qquad \textit{(Equation 6.3.10)}$$

where n is an integer. Equation 6.3.10 could be written as

$$G(j\omega) = -\frac{1}{N(A,\omega)} \qquad \textit{(Equation 6.3.11)}$$

which indicates that the intersection of $G(j\omega)$ and $-\frac{1}{N(A,\omega)}$ on the complex plane determines a possible oscillation with frequency ω and amplitude A.

According to Nyquist theory, if the chart of $-\frac{1}{N(A,\omega)}$ on the complex plane is not enclosed by that of $G(j\omega)$, the system is stable and without oscillation in the steady state.

The describing function of the stair case nonlinearity shown in Figure 6.3.3 is purely real and is as follows (Atherton, 1975).

$$N(A,\omega) = \frac{4P}{\pi A}\sum_{i=1}^{n} \sqrt{1-(2i-1)^2\left(\frac{a}{A}\right)} \quad \angle 0^\circ \qquad \textit{(Equation 6.3.12)}$$

where P is the step height of the driving force; a is the step width; n is the number of steps of the stair case. For the depth control of NOMAD, $P=30\ N$, $a=3m$, $n=2$. Note that $N(A,\omega)$ does not depend on ω in the case of nonlinearity shown in Figure 6.3.3.

Figure 6.3.5 shows that the maximum value of real part of $-\frac{1}{N(A,\omega)}$ is:

$$\mathrm{Re}\left[-\frac{1}{N(A,\omega)}\right]\Big|_{A=10.5} = -0.19 \qquad \textit{(Equation 6.3.13)}$$

The imaginary part of $-\frac{1}{N(A,\omega)}$ is

$$I_m\left[-\frac{1}{N(A,\omega)}\right] \equiv 0 \qquad \textit{(Equation 6.3.14)}$$

Figure 6.3.5 shows that $-\frac{1}{N(A,\omega)}$ equals -0.255 at the start point $A=9$, reaches a maximum of -0.19 when $A = 10.5$, and then decreases to $-\infty$ as A tend to $+\infty$.

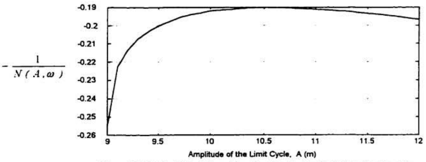


Figure 6.3.5 The Plot of Describing Function for the Stair Case Nonlinearity versus the Amplitude of the Limit Cycle

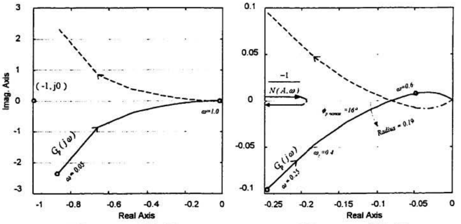
The linearized transfer function of the plant $G_p(s)$ consists of two parts: the ballast tank and the body of the subsea robot,

$$G_p(s) = \frac{X(s)}{F_c(s)} = \frac{0.13}{(0.7s+1)(6.67s+1)s} \qquad \textit{(Equation 6.3.15)}$$

Its Nyquist chart is plotted in Figure 6.3.6(a) and expanded in Figure 6.3.6(b) for the most interesting frequency range $\omega \in (0.25,\ 0.6)$. As can be seen this passes through the negative real axis at $\omega_g \approx 0.55$, where

$$\mathrm{Re}\left[G_P(j\omega)\right]\Big|_{\omega_g=0.55} = -0.08$$

Since the $G_p(j\omega)$ chart neither intersects the $-\dfrac{1}{N(A,\omega)}$ chart nor encloses it, the system will not exhibit a limit cycle oscillation and transients will die away. However, such a system is still far from being satisfactory. The main reasons lie in the following two considerations:



(a) Frequency ω: 0.05 ~ 1.0 (b) Frequency ω: 0.25 ~ 0.6

Figure 6.3.6 Nyquist Chart of $G_p(s)$ and the Describing Function of the Stair Case Nonlinearity

1. Stability Consideration: The Nonoscillation Margins are Too Small.

Here we define two quantities that measure the degree to which the system won't exhibit oscillation. As shown in Figure 6.3.7, the gain margin for nonoscillation is defined as:

$$Mg_{nonos}(db) = 20\lg\left|\frac{\sup[-1/N(A,\omega)]}{G_p(j\omega_g)}\right| \qquad \textit{(Equation 6.3.16)}$$

where ω_g is the phase-pass-through frequency.

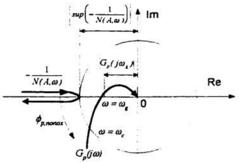


Figure 6.3.7 The Definitions of Nonoscillation Margins

Note that $Mg_{nonos}(db)$ is the factor by which the gain of the whole system can be raised before oscillation results.

For the depth control of NOMAD, we have the following from Figure 6.3.6(b).

$$\begin{aligned} &\omega_g = 0.55; \\ &|G_p(j\omega_g)| = +0.08; \\ &\left|\sup[\frac{-1}{N(A,\omega)}]\right| = 0.19 \end{aligned} \qquad \textit{(Equation 6.3.17)}$$

Substitution of Equation 6.3.17 into Equation 6.3.16 gives

$$Mg_{nonos} = 20\lg\left|\frac{0.19}{0.08}\right| = 7.4db \qquad (Equation\ 6.3.18)$$

The phase margin for nonoscillation is defined as

$$\phi_{p.nonos} = \angle G_p(j\omega_c) - 180^o \qquad (Equation\ 6.3.19)$$

where ω_c is the amplitude-pass-through frequency where the $G_p(j\omega)$ chart passes through the circle with radius $\left|\sup\left[\frac{-1}{N(A,\omega)}\right]\right|$; $\phi_{p.nonos}$ is the amount of degrees by which the phase angle of $G_p(j\omega_c)$ could decrease before oscillation results at which point

$$|G_p(j\omega_c)| = \left|\sup\left[-\frac{1}{N(A,\omega)}\right]\right| \qquad (Equation\ 6.3.20)$$

For the depth control of NOMAD, we have from Figure 6.3.6

$$\omega_c = 0.4, \quad \angle G_p(j\omega_c) = tan^{-1}\frac{0.055}{0.19} + 180^o \qquad (Equation\ 6.3.21)$$

Substitution in to Equation 6.3.19 gives

$$\phi_{p.nonos} \approx 16.14^o. \qquad (Equation\ 6.3.22)$$

Usually, to get adequate performance $\phi_{p.nonos}$ should be greater than 30^o and Mg_{nonos} greater than $8db$ (Phillips and Harbor, 1991). Obviously, the current nonoscillation margins are not sufficient.

If $G_p(j\omega)$ passes through the chart $-\frac{1}{N(A,\omega)}$, as shown in Figure 6.3.8, a severe oscillation will happen. Since $G_p(j\omega)$ passes through the negative real axis at the frequency ω_g and

$$\left|G_p(j\omega_g)\right| > \left|\sup\left(\frac{-1}{N(A,\omega)}\right)\right| = 0.19 \quad , \qquad \textit{(Equation 6.3.23)}$$

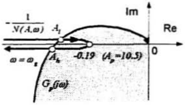


Figure 6.3.8
Intersections A_l , A_h Indicate the Oscillations Caused by the Stair Case Nonlinearily

there are two intersections and thus two possible limit cycles with the amplitudes of A_l and A_h .

Assume that the system exhibits an oscillation with amplitude A_l and frequency ω_g. If there is a disturbance driving the amplitude of the oscillation bigger (right forward along the chart of $-\frac{1}{N(A,\omega)}$ in Figure 6.3.8) the system enters the unstable range indicated by shade where $G_p(j\omega)$ encloses $-\frac{1}{N(A,\omega)}$. The amplitude of the oscillation increases in this range due to its unstable character until the amplitude arrives at A_h. On the other hand, if the disturbance drives the amplitude of the oscillation smaller (left forward from the point A_l along the chart $-\frac{1}{N(A,\omega)}$), the system enters the stable range where $G_p(j\omega)$ does not enclose $-\frac{1}{N(A,\omega)}$ and the amplitude of the oscillation decreases in this range until it arrives at the minimum value A_o. It can be concluded that the oscillation with amplitude $A=A_l$ and frequency $\omega=\omega_g$ is not a stable oscillation, because trajectories near it diverge from it. On the other hand, the oscillation with amplitude $A=A_h$ and $\omega=\omega_g$ is a stable oscillation because trajectories near it converge on it. Since the current system has only small gain and phase margins for nonoscillation, a model variation could expand $G_p(s)$ and cause intersections and an external disturbance could put A near A_h. This means an oscillation with an amplitude greater than 10.5m is possible any time. Even though it is a stable oscillation, its large amplitude is unacceptable. To avoid such situations, we need larger gain and phase margins for nonoscillation.

2. Accuracy Consideration : Steady State Error is Too Big

When the speed of the robot $v_{rob}(t)=0$ and error is between $(-a, a)$, as shown in Figure 6.3.3, there is no control output. Such a wide range of stable equilibrium states allows the system to have a large steady state error, which is not acceptable according to the design specification.

§ 6.3.2 Solutions to the Problems

To solve the two problems described in the last section, a phase lead compensator is introduced and a variable control structure is adopted.

1. A phase lead compensator is introduced to increase the gain and phase margin for nonoscillation.

The transfer function of such a compensator has the form

$$G_c(s) = k_c \frac{\alpha Ts + 1}{Ts + 1} \qquad \textit{(Equation 6.3.24)}$$

where,

T is the time constant of the compensator, α is the time lead coefficient and k_c is gain of the compensator.

The Bode plot of a phase lead compensator with $k_c = 1$ is shown in Figure 6.3.9. The maximum compensated phase angle is

$$\phi_m = \sin^{-1}\frac{\alpha - 1}{\alpha + 1} \qquad \textit{(Equation 6.3.25)}$$

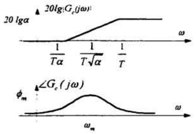


Figure 6.3.9
Frequency Character of
a Phase Lead Compensator

To increase the nonoscillation phase margin $\phi_{p,nonos}$ to 45°, ϕ_m is set to be

$$\phi_m = \phi_{p,nonos,\ target} - \phi_{p,nonos,current}. \qquad \textit{(Equation 6.3.26)}$$

where $\phi_{p,nonos,target} = 45^o$. Substitution into Equation 6.3.26 gives $\phi_m = 45 - 16.14^o \approx 28^o$. From Equation 6.3.25, we get $\alpha = 2.77$. To make a maximum phase angle compensation, the middle frequency of the phase lead compensator

$$\omega_m = \frac{1}{\sqrt{\alpha}T} \qquad \textit{(Equation 6.3.27)}$$

should be set to a frequency a little higher than ω_c, say $\omega_m \approx \omega_c + 0.3$. Then from Equation 6.3.27 we have $T = 0.866$. To increase the nonoscillation gain margin, k_c is set to $k_c = 0.4$. Then the transfer function of the compensator becomes

$$G_c(s) = \frac{0.4(2.4s+1)}{0.866s+1} \quad . \qquad \textit{(Equation 6.3.28)}$$

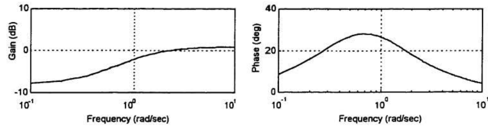


Figure 6.3.10 **Bode Plot of the Phase Lead Compensator** $G_c(s)$

Figure 6.3.10 shows the Bode plot of $G_c(s)$. With the phase lead compensator, we draw the Nyquist chart of the plant:

$$G_p(s)G_c(s) = \frac{0.13}{s(0.7s+1)(6.67s+1)}\,\frac{0.4(2.4s+1)}{(0.866s+1)} \qquad \textit{(Equation 6.3.29)}$$

as shown in Figure 6.3.11. This indicates a new phase-pass-through frequency $\omega_g=1.0$, which gives $|G_pG_c(j\omega_g)| = -0.01$, and a new amplitude-pass-through frequency $\omega_c=0.45$. which gives

$$\angle G_pG_c(j\omega_c) = tg^{-1}\frac{I_m[G_pG_c(j\omega_c)]}{R_e[G_pG_c(j\omega_c)]} + 180^\circ = 229^\circ . \qquad \textit{(Equation 6.3.30)}$$

So the new phase and gain margins for nonoscillation after compensation are

$$\phi_{p,nonos} = \angle G_p G_c(j\omega_c) - 180^\circ = 49^\circ$$

(Equation 6.3.31)

$$Mg_{nonos}(db) = 20\lg\left[\frac{\left|\sup[-1/N(A,\omega)]\right|}{\left|G_p G_c(j\omega_g)\right|}\right] = 20\lg\left|\frac{-0.19}{-0.01}\right| = 25.57\ db$$

This essentially guarantees that the system won't oscillate due to an intersection of $G_pG_c(j\omega)$ and $-\dfrac{1}{N(A,\omega)}$.

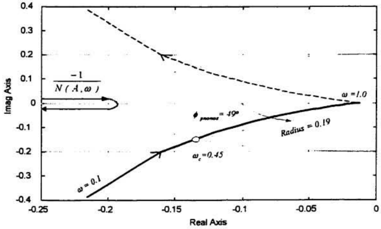


Figure 6.3.11 Nyquist Chart of $G_p(s)G_c(s)$ and the Describing Function of the Stair Case Nonlinearity

2. A Variable Structure Fit Into the Fuzzy Controller to Get Rid of the Steady State Error

As mentioned in section §3.3.1, to increase the system reliability and decrease the cost, micro switches were used instead of a continuous sensor to detect the water level in the ballast tank. So the fuzzy controller with discontinuous output exhibits not only a high nonlinearity but also a " dumb" character when the error falls into the dead zone. This causes a steady state error which depends on both the width of the dead zone $2a$, and the output jump P of the stair case.

If we decrease the width of the dead zone $2a$ and at the same time try to keep a sufficient step jump P to ensure quick transition from one depth to another depth when depth error is big, the turn around point of $-\frac{1}{N(A,\omega)}$ will move to the right and the chances of an intersection with $G_p(j\omega)$ will increase. Such an intersection will result in a limit cycle with a higher frequency. As mentioned in Chapter 3, every change of the direction of the control force associated with a limit cycle consumes a certain amount of compressed air, the main energy source of the robot NOMAD. So any increase of the oscillation frequency would cause the stored energy to be used up at a faster rate.

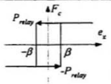
F_c e_z v	NB	NS	Z	PS	PB
P	NB	NB	NS	PS	PS
Z	NB	NS	 F_c P_{relay} e_z $-\beta$ β $-P_{relay}$	PS	PB
N	NS	NS	PS	PB	PB

Figure 6.3.12 The Rule Base Of The Fuzzy Logic Controller With A Variable Structure

To solve this problem, a variable structure is introduced into the depth control strategy for the robot NOMAD. When the error drops into the dead zone of the stair case nonlinearity, which is indicated by rule 8: " Zero" Control Effort in Table 6.2.1, the control law shifts from fuzzy control to relay control. The relay control has a small hysteresis loop instead of a dead band with a minor step jump in control output, as shown in Figure 6.3.12. As will be shown below, with a proper choice of β and P_{relay}, it is possible to drive the system into a stable limit cycle with a lower frequency and a much smaller amplitude. We named this control scheme Fuzzy Variable Structure Switching (FVSS) Control.

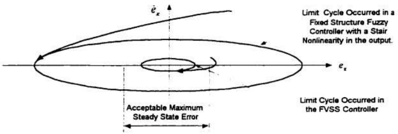

$\dot{e}_z$
e_z
Limit Cycle Occurred in a Fixed Structure Fuzzy Controller with a Stair Nonlinearity in the output.
Limit Cycle Occurred in the FVSS Controller
Acceptable Maximum Steady State Error

Figure 6.3.13 Comparison of the Limit Cycle Occurred in a Fixed Structure Fuzzy Controller of Discontinuous Outputs with that in the FVSS Controller

Figure 6.3.13 gives the comparison of the oscillation caused by the stair case nonlinearity with amplitude $A_h \approx 10.5m$ and frequency $\omega_g \approx 0.55$ and the oscillation caused by the Fuzzy Variable Structure Switching (FVSS) controller. The latter has a much smaller amplitude and lower frequency. The maximum amplitude of the limit cycle caused by the FVSS controller can be made to sit within the band of required steady state error. So the system is stable in the Lyapunov sense. A lower frequency of limit cycle means a smaller number of changes in the direction of control force, and this means less energy consumption.

For the depth control of NOMAD, we let $\beta = 0.15m$, $P_{relay} = 3N$. The describing function for the relay with hysteresis nonlinearity is

$$N(A,\omega) = \frac{4P_{relay}}{\pi A} e^{-j\beta} \qquad \textit{(Equation 6.3.32)}$$

and its negative reciprocal is

$$\begin{aligned} \frac{-1}{N(A,\omega)} &= -\frac{\pi}{4P_{relay}}\sqrt{A^2-\beta^2} - j\frac{\pi\beta}{4P_{relay}} \qquad (A \geq \beta) \\ &= -0.26\sqrt{A^2-0.0225} - j0.039 \end{aligned} \qquad \textit{(Equation 6.3.33)}$$

The plot of this $-\frac{1}{N(A,\omega)}$ on the $G_p(j\omega)G_c(j\omega)$ complex plane is shown on the Figure 6.3.14. For $A \geq \beta$, it is a horizontal line, which starts at $(0, -j0.039)$ with $A = \beta$ and extends to $(-\infty, -j0.039)$ as A increases to infinity. The chart of $-\frac{1}{N(A,\omega)}$ meets the Nyquist chart of $G_pG_c(j\omega)$ at the point which satisfies:

$$\frac{-1}{N(A,\omega)} = G_pG_c(j\omega) \qquad \textit{(Equation 6.3.34)}$$

From Figure 6.3.14, for this point, $Re_{inter} = -0.06$, $Im_{inter} = -0.039$.

The frequency of the intersection should satisfy:

$$|G_P(j\omega)\,G_c(j\omega)| = \left| \frac{0.13}{(0.7j\omega+1)(6.67j\omega+1)j\omega} \quad \frac{(2.4j\omega+1)}{2.5(0.866j\omega+1)} \right| \qquad \textit{(Equation 6.3.35)}$$
$$= \sqrt{0.06^2+0.039^2} = \sqrt{\mathrm{Re}_{inter}^2+\mathrm{Im}_{inter}^2}$$

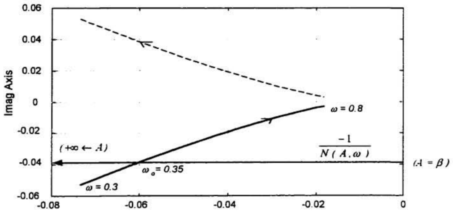


Figure 6.3.14 Nyquist Chart of $G_p(s)G_c(s)$ and the Describing Function of the Relay Nonlinearity with a Small Hysteresis Loop

Solution of Equation 6.3.35 gives the limit cycle frequency $\omega_o \approx 0.35$.

The amplitude of the limit cycle corresponding to this intersection point is solved from

$$\mathrm{Re}\left[-\frac{1}{N(A,\omega)}\right] = \mathrm{Re}_{inter} \qquad \textit{(Equation 6.3.36)}$$

The real part of equation 6.3.33 gives

$$-\frac{\pi}{4P_{relay}}\sqrt{A^2-\beta^2}=\mathrm{Re}_{inter} \qquad \textit{(Equation 6.3.37)}$$

where,

$P_{relay}=3N$ is the height of the hysteresis of the relay and $\beta=0.15$ is the width of the hysteresis of the relay. Solution of Equation 6.3.37 gives for the FVSS controller the limit cycle amplitude:

$$A_o=0.27m \qquad \textit{(Equation 6.3.38)}$$

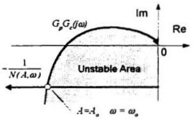


Figure 6.3.15
The Stable Oscillation (A_o, ω_o) Existing in the FVSS Controller

Since the area where the Nyquist Chart $G_pG_c(j\omega)$ encloses the $-\frac{1}{N(A,\omega)}$ is the system unstable area, any oscillation with an amplitude smaller than $A_o=0.27m$ will grow until its amplitude equals A_o as shown in Figure 6.3.15. On the other hand the area outside of the shaded is the stable area, so that any oscillation with an amplitude larger than A_o will decay until its amplitude equals A_o.

It can be concluded that the oscillation with amplitude $A_o=0.27m$ and frequency $\omega_o=0.35$ determined by the intersection of $G_pG_c(j\omega)$ and $-\frac{1}{N(A,\omega)}$ is a stable oscillation in the Lyapunov sense.

Figure 6.3.16 and Figure 6.3.17 show the results of numerical simulation of the whole system with the Fuzzy Variable Structure Switching (FVSS) controller under the different initial conditions. Note that the system exhibits only the limit cycle oscillation due to the relay nonlinearity shown in Figure 6.3.12 but not the oscillation due to the stair case nonlinearity shown in Figure 6.3.3. Additionally the variable structure of the fuzzy controller makes it possible to confine the oscillation within the required maximum steady state error. The variable structure passes the control to a narrow hysteresis relay nonlinearity when the error arrives at the dead zone of the stair nonlinearity. To take into account the effect of asymmetry of the flow rate into and out of the ballast tank and other minor nonlinearities possessed by the system, a detailed simulation around the equilibrium point on the mechanical component level was carried out before this control scheme was implemented by the onboard micro computer of the robot NOMAD. The physical realization for the Fuzzy Variable Structure Switching Control scheme and its in-water test will be detailed in following chapters.

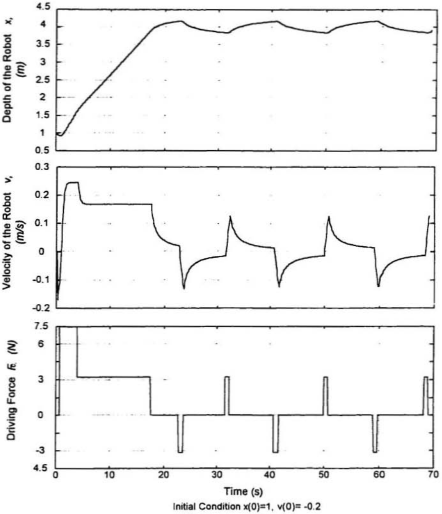


Figure 6.3.16
Depth Response and Driving Force of the Subsea Robot with the Fuzzy Variable Structure Switching Controller Which Has A Crisp Membership Function and Discontinuous Wave Form in the Driving Force

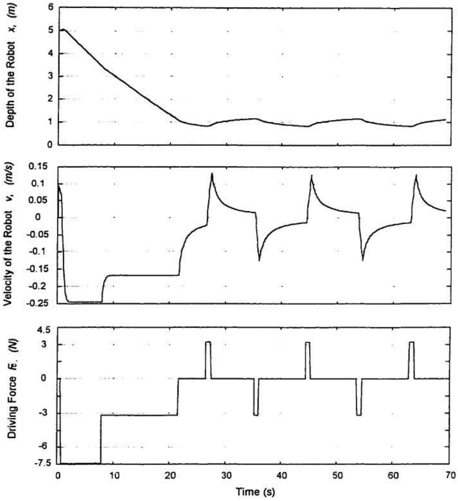


Figure 6.3.17
Depth Response and Driving Force of the Subsea Robot with the FVSS Controller Which Has A Crisp Membership Function and Discontinuous Wave Form in the Driving Force

CHAPTER 7 SOFTWARE ARCHITECTURE FOR THE CONTROL SYSTEM

This chapter deals with the software architecture. Because of the complexity of the control process, a multiple-task, multiple-layer software structure was developed using Dynamic C and a micro computer with a Z180 CPU. Here the main aspects of the software developed for NOMAD are discussed, including Real Time Kernel for implementation of the software structure; Interruption for time management of computing capacity; Space Management of the memory capacity.

§ 7.1 Context Relationship in the Software Structure

Intelligent control of underwater robots is a demanding task for which no completely satisfactory approach has yet been developed (Yuh and Lakshmi, 1993). For underwater robots that are designed to remain on-site for significant periods of time (Curtin and Bellingham, 1993) or those whose missions are otherwise of long duration, the task is more difficult. Not only must the controller create and follow a plan to accomplish mission objectives, it must also pay careful attention to resource management. Unanticipated events, which can occur during missions of any duration, become much more likely the longer the mission. For an underwater robot far from base, and hence far from human intervention, handling events intelligently becomes more important. Such control activities are generally referred to as high level control. On the other hand, during operation, the dynamic behavior of the robot, such as moving from one spot to another or hovering at a specified depth, should be closely observed and controlled,. These are usually referred to as low level control. The highest level would make decisions based on

(1) information fed from the lower level controller such as current dynamic states, surrounding circumstances or failure detection;

(2) pre-stored task agenda and directives;

(3) renewed instructions from communication links between the subsea robot and the shore station.

Usually the last one is not a real time command. In other words, it would not directly lead to low level control actions or affect the dynamic behavior of the robot. Human interference could be used to modify pre-stored messages, cancel or add task routines, or change the parameters in some tasks. Information from human interference would be first " digested" by the highest control level. Then the decisions made by this level would be propagated to the controllers in the lower levels. These decisions could change the set points, algorithms or parameters in the lower level controllers (Hall and Adams, 1992).

The lowest level controllers in the hierarchical architecture usually deal with the hardware interfaces. They receive signals from different sensors, put data from sampling devices into the memory and send driving commands to different actuators, for example, the valves in the robot NOMAD. Each low level controller would have its own control algorithm and its own feedback control loop. These control algorithms are real time algorithms. They work at the rate around 25 ms in NOMAD. Those in higher levels work at a much slower rate. The corresponding period can be minutes or hours. They scan for messages from human interface, hardware and database, based on a regular period or on an interruption request. These messages start sections of code that generate tasks for lower level controllers.

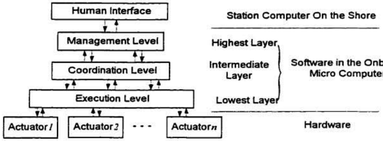


Figure 7.1.1 Hierarchical Software Architecture for the Micro Computer Based Onboard Controller

The highest level in the hierarchical architecture could be referred to as the "management level", while the lowest level could be referred to as the "execution level". The level between these two levels, an intermediate level, could be referred to as the "coordination level" as shown in Figure 7.1.1.

§ 7.2 Real Time Kernel (RTK) - Implementation of the Software Structure

To implement the multiple-layered control strategy described in Section §7.1 on the NOMAD micro computer, a Real Time Kernel (RTK) software structure was developed. The RTK allows the control program to be divided into prioritized tasks.

These tasks can each be treated as separate programs running independently of one another. Execution of these tasks is interleaved in time. Two principal advantages of this real time kernel are:

(1) More urgent tasks like depth control and failure detection can be performed in preference to less urgent tasks. The priority order could be changed by the management control level. That is, in different missions, the priority order could be quite different.

(2) It is easier to write and organize a program when separate tasks or sequences of events can be handled as if they were isolated from one another. Special events such as urgent alarms could generate asynchronous interruption request signals. If the CPU acknowledges such a request, it will suspend the current routine and wake the higher level control. Various control functions are organized in different tasks. Only the input and output data for each task are put into a common data exchange area to set up the link of the hierarchical structure.

A real time clock is required for the Real Time Kernel (RTK) to operate. A programmable counter (PRT) on the CPU Z180 creates periodic synchro interruptions that can be used for this purpose. The period of synchro interruptions can be run at rates between 20 and 500 times per second. In the controller for NOMAD, the period of synchro interruptions or the time interval between ticks is *25 ms*, that is, at a frequency of *40Hz*. This time is small relative to the limit cycle period, which is around 20s, and to the ballast tank time constant, $T_{tank}=0.7sec$, which is the smallest time constant in the system. According to Shannon's Sampling theorem

(Embree and Kimble, 1991)., a 25 ms real time click is fast enough for this system.

To reduce the computing overhead, a number of time functions are installed in the control programs for NOMAD to make sure a task is invoked only when it is needed. Following are some time functions used in the high level and low level control of NOMAD.

(1) The program can request that a task be invoked at a particular time [36 bit BCD(Binary Coded Decimal) code of ticks] which covers a time range of $10^{\frac{36}{4}} \times 25ms = 289\ days$. In the management level, different missions are listed by the execution time in the agenda, for example,

run_at (*mission number, int *time*) causes the mission specified by the *mission number* to be requested when the time is greater than or equal to the time specified by the pointer *int *time*. The time pointer points to a 36 bit number (9 digits of BCD code), which is the number of ticks since RTK was called.

(2) A mission could also be called a certain time (24 bit BCD code of ticks) after a specified task was invoked, for example,

run_after(*mission number, long delay*) requests the mission specified by *mission number* to run *long delay* (a long integer) ticks after the "*run_after*"function was executed. The maximum delay is $10^{\frac{24}{4}} \times 25ms = 6.9\ hours$.

(3) Lower level tasks are invoked on a regular basis:

run_every(*task number, unsign period*) causes the specified task of *task number* to be requested once every *unsign period* of ticks. The maximum length of the period is

$2^{16} \times 25ms = 27$ *minutes.*

The multiple-layer and multiple-task hierarchical architecture for control was implemented by the Real Time Kernel (RTK). Special care was taken to ensure that the control program contained well organized mission and task descriptions. The program was written in Dynamic C. The flow chart is shown in Figure 7.2.1. Its basic structure is as follows:

```
/*  Real Time Kernel Control Structure  */
int mission 1( ), ... mission n( ), task 1( ), ... task m ( ), background ( );
int (*Ftask[ ]) ( )= {mission 1, ..., mission n, task 1, ..., task m, background };
......
constant definition
......
variable definition

root main ( )
{
     /*  T & P Area: Time and Parameters Table for Missions  */
     run_at (mission_number_1,  int * point_to_time_1) ;
       ......
     run_at (mission_number_n,  int *point_to_time_n) ;
       ......
     run_every (task_number_1,  unsign  period_1) ;
       ......
     run-every (task_number_m,  unsign period_n) ;
     system initialization ;
}

mission 1( )
{
     /*  Invoke the time schedule and parameters for mission 1
          in the T&P Area in the root   main ( ).   */
     Description for mission 1 ;
     Exchange the data with the INFOEX area in background ( ) ;
}
.......
```

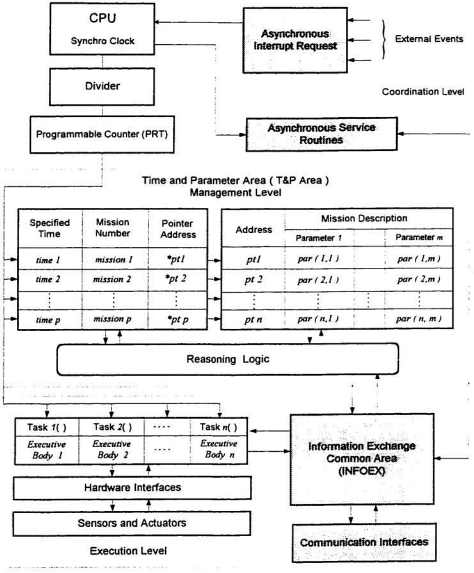


Figure 7.2.1 Detail of the Software Structure of the Onboard Controller

```
mission n ()
{
    /*  Invoke the time schedule and parameters for mission n
        in the T&P Area in the root   main ().   */
    Description for mission n ;
    Exchange the data with the INFOEX area in background () ;
}

task 1 ()
{
    Description for task 1 ;
    Exchange the input output data with the INFOEX area in background() ;
    begin of execution for task 1 () ;
    .........
    end of execution for task 1 () ;
}
......

task n ()
{
    Description for task n ;
    Exchange the input output data with the INFOEX area in background() ;
    begin of execution for task n () ;
    .........
    end of execution for task n () ;
}

background ()
{
    /*  INFOEX area: common data storage area for the data exchange
        between missions and tasks.     */
    Coordination for  the tasks to form the layered structure ;
    Communication with the station computer ;
}
```

It is quite possible that the routine for a mission would need to be changed by the operators working at the station computer, when the subsea robot was already at the working site. That means that the content in the T&P area would need to be modified through some way. A common exchange database INFOEX was set up for this purpose. This database is accessible by the T&P area and every task function and communication interface. There are two means of communication for the subsea robot NOMAD: one is the cable communication in the Remote Control Mode while the other is the radio communication in the Autonomous Control Mode. The latter one is only available when the robot emerges to the water surface.

§7.3 Interruption - Time Management of Computing Capacity

Z180 is the CPU chip of the onboard micro computer carried by the robot NOMAD. Besides 7 internal interruption request inputs to the CPU, there are three maskable external interruption request inputs to the CPU and one non-maskable interruption request input to the CPU. For each interruption request, except for the non-maskable request, an interruption vector is assigned, which points to the entrance address of the corresponding interruption service routine (ISR). Table 7.2.1 shows the functions of interruption inputs and the addresses for their vectors.

Instead of loading the entrance address of the ISR from the interruption vector, the nonmaskable interruption input **NMI** causes a jump directly to the entrance address of the ISR. For example: 0×66 is not the address of the interruption vector, but it is directly the entrance address of the nonmaskable power-failure interruption service routine. That usually deals with emergency events,

Table 7.2.1 Interruptions and Their Pointers Addresses

Name	Pointer Address	Description
INT0	0x38	External interruption terminal
INT1	0x00	External interruption terminal
INT2	0x02	Maskable External interruption terminal
NMI	0x66*	Nonmaskable external interruption terminal
PRT0	0x04	Programmable counter channel 0, internal interruption request
PRT1	0x06	Programmable counter channel 1, internal interruption request
DMA0	0x08	Direct Memory Access Ch0 internal, interruption request
DMA1	0x0A	Direct Memory Access Ch1 internal, interruption request
CSI0	0x0C	Clocked Serial Input/Output internal interruption request
SER0	0x0E	Serial Port Channel 0 internal interruption request
SER1	0x10	Serial Port Channel 1 internal interruption request

such as critical element failures. The three external interruption request terminals are used to deal with external events: INT1 and INT2 are used in radio communication while INT0 is a reserved interruption request terminal for the instruments in the payload chamber, in case they need to send an interruption request to the CPU of the onboard computer. SER0 and SER1 are used to handle the RS232 cable communication between the onboard micro computer and the station computer. DMA0 and DMA1 offer a direct connection between recorded data in the Extended Memory (XMEM) and the communication interface. The recorded data could be the results of marine observations such as temperature and salinity profiles versus depth. When the robot emerges to the surface, the radio transmitter would send these data to the station computer. To save CPU time, the DMA interruption service routine bypasses the CPU and transfers the data in the Extended Memory (XMEM) directly to the transmitter. DMA0 and DMA1 are internal interruption requests, which can be triggered by the void functions:

void DMA0_count (int count) or *void DMA1_count (int count)*

When the DMA0 count function is executed, an interruption request from DMA channel 0 is sent to the CPU and the flag DMA0 is set to 1, which enables the interruption. As soon as the CPU acknowledges the DMA interruption request, the interruption service routine dumps the data in XMEM to the communication interface.

The priorities of interruption requests in Table 7.2.1 are organized as shown in Table 7.2.2.

Highest *Table 7.2.2* Priorities of Interruptions Lowest

NM1	INT0	INT1	INT2	PRT0	PRT1	DMA0	DMA1	CSI0	SER0	SER1

Interruptions with lower priorities can be interrupted by an interruption request with a higher priority, as shown in Figure 7. 3.1.

(0) The CPU is executing the main program.

(1) The CPU receives an interruption request from a source with lower priority.

(2) The CPU acknowledges the interruption request with lower priority.

(3) The CPU is executing the interruption service routine with lower priority.

(4) The CPU finishes the service routine and returns to the main program.

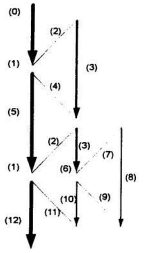


Figure 7.3.1 The Priority in Interruption

(5) The CPU is continuously executing the main program.

(6) While the CPU is executing an interruption service routine with lower priority, another interruption request with higher priority is sent to the CPU.

(7) The CPU stops executing the current interruption service routine with lower priority and acknowledges the interruption request with higher priority.

(8) The CPU is executing the interruption service routine with higher priority.

(9) The CPU finishes the service routine with higher priority and return to the service routine with lower priority.

(10) The CPU is continuously executing the remaining part of the service routine with lower priority.

(11) The CPU finishes the service routine with lower priority and return to the main program.

(12) The CPU continues the main program.

Interruption requests are scanned by CPU at the end of every Directive Period. If the Interruption Flag is ON (IFF register), the CPU will acknowledge the request and push the content of all registers into a stack (to reserve the current work space) and redirect the routine to a specified Interruption Service Routine (ISR) to execute the "Body Code"of ISR. As soon as it is finished, the work space stored in the stack will be popped out to the registers and the routine that was executing before the interruption is continued. Following is an example of the interruption service routines used in the depth control program of the robot NOMAD.

```
interrupt reti service ( )
{
    EI( );              *    Re-enable interruptions. Allows a higher interruption request
                             to interrupt the current ISR.    *
    Begin of the interruption service:
        ...
    Bode of code:
        ...
    End of the interruption service:
    DI ( );             *    Disable interruption. Avoid data loss when the data
                             in memory and in registers are being exchanged.    *
    Resume registers contents to the contents before interruption:
    Return:
}
```

In the control system of NOMAD, the real time clock or synchro interruptions are sent by the Programmable Counter Channel in PRT. They have a medium interruption priority in Table 7.2.2. The Interruption Service Routines caused by PRTs are actually missions and tasks as shown in Figure 7.2.1. The interruption requests INTO and INT1 and INT2 caused by external events have higher priorities. They can interrupt the ISR caused by PRT. Since INTO and INT1 and INT2 are maskable interruptions, these interruption requests can be disabled by resetting (set to " 0") the corresponding registers IFFO and IFF1 and IFF2.

Disabling external interruptions is often done at the beginning of a specified *task( )* or *mission( )* function and enabling external interruptions again is done at the end. In this way, some specified important task would not be interrupted by external interruptions even though the latter may have a higher priority.

The interruptions of SER0 and SER1 have the lowest priority. They are used for cable communication running in the background. When the CPU is free from other ISRs, the onboard micro computer communicates with the station computer by the RS232 link.

Two of the three maskable external interruptions, INT1 and INT2, are assigned to serve as the " hand shaking" functions for radio wave communication. The INT1 interruption request is triggered at the end of data transmission between the onboard micro computer and the station computer. When the station computer receives all incoming data and confirms there is no error in the parity check (odd-even check), it sends a signal to the onboard micro computer of the robot to trigger the interruption request. The interruption service routine for INT1 stops recursive transmission and turns off the power supply to the transmitters. The radio transmitter would have been turned on in a pre-programmed mission and it periodically repeats the data transmission until the ISR for INT1 terminates it. The other maskable interruption is for receiving the signal from the station computer to modify the contents of the T&P area of the onboard micro computer. After the onboard transmitter sends a hand shaking signal to the station computer in a specified mission, the radio receiver on the subsea robot is turned on and waits for the acknowledgment. As soon as the radio receiver receives the acknowledgment signal from the station computer, the second interruption request INT2 is triggered. As soon as the received data are transferred into the serial buffer of the receiver, the ISR for INT2 writes them to proper addresses in the T&P area where the time tables and parameters for missions are stored. An overview of the communication system is shown in Figure 7.3.2.

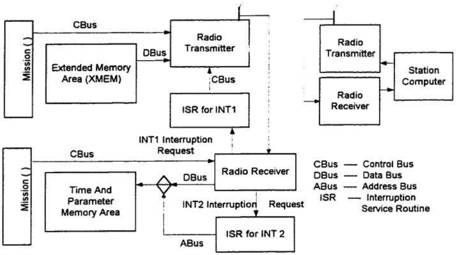


Figure 7.3.2 Proposed and Lab Tested Radio Communication System for the Subsea Robot NOMAD

§ 7.4 Space Management of the Memory Capacity

When the robot NOMAD is carrying out a marine observation mission, recorded data would be stored in the battery backed RAM (Random Access Memory). These data would then be transferred in batch to the station computer for processing. A big capacity of memory in the onboard micro computer would be required for these raw temporary data storage. Unfortunately, the Z180, the CPU chip in the onboard micro computer, has only 16 bit addresses, giving a logical address space of *64k (2^{16}=65536).* For long term observation missions, this 64k memory is too small for storage of executable machine codes and recorded data, and for the working space needed by control algorithms.

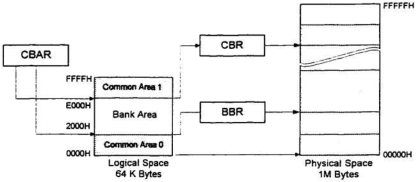


Figure 7.4.1 Registers CBAR, CBR, BBR Used to Map Logical Space to Physical Space

The manufacturer of the Z180 chip offers a possibility to expand the memory capacity of the chip by using on-chip registers. These registers could form a memory management unit (MMU) to translate 16-bit addresses to 20-bit addresses. The maximum memory capacity after expansion could be 1 megabyte (2^{20}=1048576). Now we name three 8-bit registers as follows:

CBAR	*Common/Bank Area Register*
CBR	*Common Base Register*

BBR *Bank Base Register*

They can be used to divide the 64k logical space into three sections and map each section into 1 Mbytes physical memory as shown in Figure 7.4.1.

The logical address space is partitioned on 4 Kbytes boundaries. Given a 16-bit address, the Z180 CPU determines, using the register CBAR, whether the address is in common area 1, common area 0, or the bank area. If the address is in common area 1, the Z180 uses the register CBR as the base to calculate the 1 Mbytes physical address. If the address is in the bank area, the Z180 uses the BBR. If the address is in common area 0, the Z180 uses a base of 00_{HEX}. The physical address is calculated by:

(Base Register: CBR, BBR) << *12 + Logical Address*

as shown in Figure 7.4.2, where << is the left shift operator. Table 7.4.1 describes the basic functions of the three sections in the 64 kbyte logical space.

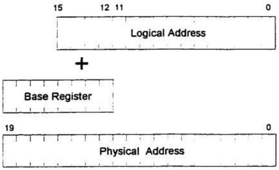


Figure 7.4.2 Calculation for the Physical Address

Table 7.4.1 Basic Functions of 64 Kbyte Logical Space

NAME	AREA	SIZE	DESCRIPTION
BIOS	Common Area 0	8K	Basic Input/Output System. The BIOS contains the power-up code, the communication kernel and important system features. The BIOS is always mapped to address 0000_{HEX} of ROM in the physical memory space corresponding to common area 0.
ROOT root code root data root buffer	Bank Area	48K	The Root, which corresponds to the Bank Area, resides in a fixed portion of physical memory between the BIOS and XMEM. Root code grows upward in logical space from address 2000_{HEX} and root data grow down from $E000_{HEX}$.
XMEM	Common Area 1	8K	Extended Memory. XMEM is essentially an 8 Kbytes "window" into physical memory. XMEM can map to any part of physical memory simply by changing the CBR register. XMEM corresponds to Common Area 1.

The sections of BIOS in the logical space have 1:1 mappings to the physical space. The section of Root has the option to map either to the EPROM or to the Static RAM. However, the XMEM section has many mappings in the physical space, as shown in Figure 7.4.3.

The executable machine codes of the control system of NOMAD resided as

Root Code in the Bank Area when the system was under development. The Root Code in the logical space was mapped to the RAM of the physical space, as shown in the path A of Figure 7.4.3. When well developed, the Root Code was burned in the EPROM of the physical space, as shown in the path B of Figure 7.4.3.

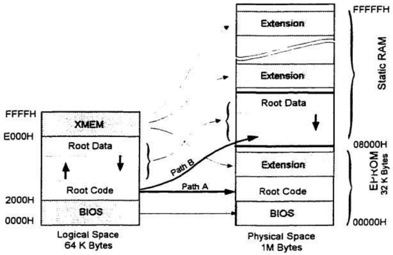


Figure 7.4.3 Correspondence Between the Logical Space and the Physical Space

The Root Data in the Bank Area is the working space for the control algorithms, which is always mapped to the RAM of the physical space. The marine observation data, recorded by different sensors in the payload chamber, are sent to the Root Data area in the logical space first and are then transferred to the Extended Memory Segments in the physical space by 8 Kbytes each page.

The following two functions, written in Z180 assembly language, are used to transport the marine observation data between the Root Data area in the logical space and the Extended Memory Segments in the physical space.

```
xmem2root ( long source,   void *destination, int n );
root2xmem ( void *source,   long destination, int n );
```

where, *long source* and *long destination* are 32 bit integers indicating 20 bit physical addresses; *void *destination* and *void *source* are 16 bit pointers indicating 16 bit logical addresses; *int n* is the number of the bytes that will be transferred.

The following program is used to transfer the marine observation data in the root area to the extended memory in a batch of 8 Kbytes corresponding to 2048 floating point data points.

```
int i;
float rtmem [2048];                                   *   4 bytes for a float point datum    *
xdata xtmem [2048];
for ( i=1; i <= 2048; i ++ )                          *   transfer 2048 data                 *
{ root2xmem (rtmem + i * 4, xtmem + i * 4, 4); }        *   4 bytes for each transmission *'
```

where, *rtmem* is an array in root area containing 2048 data recorded by a sensor, each four bytes long; *xtmem* is the array in extended memory segments, occupying 8 Kbytes extended memory area.

For the control system of the robot NOMAD, an extended memory with 256 Kbytes capacity was installed, and this was used to store data profiles versus depth and other observation records.

§ 7.5 Functional Configuration of the Control Software

A typical mission for the subsea robot would be to get temperature and salinity profiles at specific times. The robot is supposed to wake up from the sea floor and scan the depth range between the pre-programmed upper and lower limits of the depth. When the mission is completed, the robot goes back to the sea floor continuing its sleep. Assuming the scientists are only interested in these profiles where the water is relatively calm, the scan range would not be fixed. Usually the sea becomes rougher closer to the surface. In that case, the robot itself would change the upper limit on depth based on readings from its wave action sensor which indicates the roughness of the sea state.

Some failure detection mechanisms would be adopted during missions. For example, after hours of sleeping on the sea floor, it is possible for the robot to get stuck in the bottom sediment. In that case, the speed of the robot would be still zero after an "up movement" directive was issued. To break free, the robot could use a strong water/air jet to help the robot move up. The strength of such a jet depends upon the pressure difference between inside and outside of the ballast tank. This could be specified in the T&P area and be controlled by a jet intensity loop consisting a pressure sensor in the ballast tank and a charging valve as shown below.

```
root main ( )
{
    /*T&P Area: time table and parameters are stored in the structure array T&P */
        float up_bound, low_bound, wave_scale, other_par1, other_par2, other_par3;
        int scan_number, wave_check, acting_time, return_sleep, record_temp,record_salty;
    struct T&P
    {
        float Up_bound;
        float Low_bound;
        int Scan_number;
        float Wave_scale;
        int Wave_check;
        int Acting_time;
        float Other_par1;
        float Other_par2;
        float Other_par3;
    }   mission_des [5];

    /*  The dimension of the structure array is 5. Descriptions for five missions
        are stored.  Following is the descriptions for the first mission */
    mission_des[1]. Up_bound = UP_BOUND
    mission_des[1]. Low_bound = LOW_BOUND
    mission_des[1]. Scan_number = SCAN_NUMBER
    mission_des[1]. Wave_scale = WAVE_SCALE
    mission_des[1]. Acting_time = ACTING_TIME
    mission_des[1]. Other_par1 = JET_INTENSITY
        .....
    run_at (mission 1, DD_HH_MM)
        ......
}

mission1( )
{
    up_bound = mission_des[1].Up_bound ;
    low_bound = mission_des[1].Low_bound;
    scan_number = mission_des[1].Scan_number ;
    wave_check = "ON" ;
    wave_scale = mission_des[1].Wave_scale ;
    acting_time = mission_des[1].Acting_time ;
    stick_check = "ON" ;
    jet_intensity = mission_des[1].Other_par1 ;
    return_sleep = "ON" ;
    record_temp = "ON" ;
    record_salty = "ON" ;
}
```

Activities on the execution level of the robot are controlled by a higher layer. For example, tasks dealing with the failure situation of "*failure due to stuck, can not move*" and the decision "*to make a jet*" are implemented by *task8()*. It is executed only when the flag of *stick_check* is "ON". This flag is set by the management layer.

```
task8( )
{ if    ( (F_p = "NS" or F_p = "NB")&                    /* failure detection */
        (time_count > acting_time)&
        (abs(spd) < 0.1)&
        (stick_check='ON" ) )
  {     up_setout (5.0); up_setout (4.1) ;               * close both top and bottom plugs*
        up_setout (2.0); up_setout (1.1) ;
        if      (inside_pressure - outside_pressure   jet_intensity)
                up_setout (3. 1);                        * charge to the ballast tank until the
        else                                             pressure difference equals
        {       up_setout (3. 0);                        jet_intensity */
                up_setout (2. 1);
                up_setout (1. 0);                        /* open the bottom plug to
        }                                                generate the jet */
  }
}
```

Tasks for "*wave scale check*" and "*update the up_bound of the scan range*" are implemented by *task9( )*. This is executed on a regular basis if the flag of *wave_check* is "ON".

```
task 9 ( )
{       if((wave_sensor > wave_scale)&(wave_check="ON"))
        {       up_bound = up_bound * 1.2;
                wave_check = "OFF";
        }
}
```

The wave intensity is measured by the wave action sensor. When it is larger than a preset value *WAVE_SCALE*, the set point of the *up_bound* of the scan range is increased by 20%. Then the wave check flag is disabled, so that the *up_bound* won't be increased any more until an interference from a higher control level resets the variable *wave_check* to "*ON*" again.

The main flow chart of the control process of NOMAD is shown in Figure 7.5.1. The robot has two working modes: Remote Mode and Autonomous Mode. Most simple missions could be completed by the Remote Mode, while long term missions should be carried out in the Autonomous Mode. In the Remote Mode, there are two sub-modes: Remote Operating Mode and Remote Automatic Mode. Either of them could be activated through the keyboard. In the Remote Operating Sub-mode, the robot follows the keyboard commands using the means of buoyancy/weight and water/air jet to move up and down. In the Remote Automatic Sub-mode, the depth set point and jet pressure set point are specified by the station computer first, then the robot moves between the different depth set points. In the Autonomous Mode, missions and mission related parameters are stored in the agenda (T&P Memory Area). The agenda is visited when the mission is invoked by the function call of *run_at(mission number, time)*. There are two special missions: the " *Communication Mission*" and the " *End Mission*". Before an invoked mission is carried out, it is judged by the Management Layer of the control system first to see if it is one of these two special missions. If the mission is a " *Communication Mission*", the subsea robot moves up to the surface to transmit or receive signals from the station computer. If it is an " *End Mission*", the robot emerges to the surface and transmits a Collection Request to the station computer and then Self Location Pulse Signals to help a collector such as a helicopter locate the robot and bring it home. If the mission is a regular working mission, the mission related parameters are propagated to the Coordination and Execution layers of the control system and then the mission is carried out. When the mission is finished, the robot goes to the default state, which is sleeping on the sea floor and waiting for the next mission call.

The Remote Mode with the Remote Operating Sub-mode and the Remote Automatic Sub-mode were tested on the physical subsea robot in the Deep Tank of Memorial University. The Autonomous Mode was tested on the same robot in a simulated environment in the laboratory. These tests will be detailed in the next chapter.

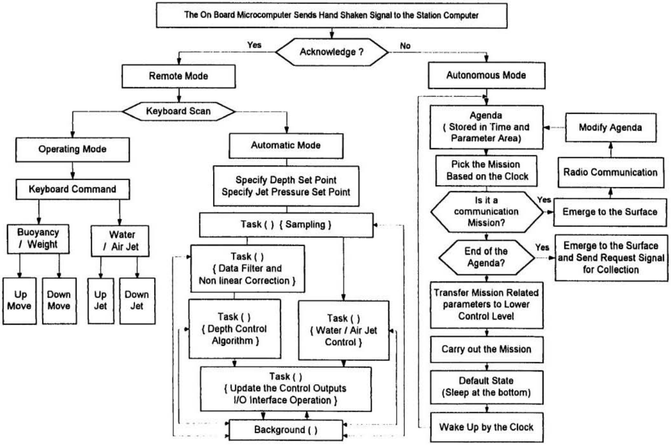

The On Board Microcomputer Sends Hand Shaken Signal to the Station Computer
Acknowledge ?
Yes
No
Remote Mode
Autonomous Mode
Keyboard Scan
Operating Mode
Keyboard Command
Buoyancy / Weight
Water / Air Jet
Up Move
Down Move
Up Jet
Down Jet
Automatic Mode
Specify Depth Set Point
Specify Jet Pressure Set Point
Task ( ) { Sampling }
Task ( ) { Data Filter and Non linear Correction }
Task ( ) { Depth Control Algorithm }
Task ( ) { Water / Air Jet Control }
Task ( ) { Update the Control Outputs I/O Interface Operation }
Background ( )
Agenda (Stored in Time and Parameter Area)
Modify Agenda
Pick the Mission Based on the Clock
Radio Communication
Is it a communication Mission?
Yes
Emerge to the Surface
End of the Agenda?
Yes
Emerge to the Surface and Send Request Signal for Collection
Transfer Mission Related parameters to Lower Control Level
Carry out the Mission
Default State (Sleep at the bottom)
Wake Up by the Clock

Figure 7.5.1 Main Flow Chart of Control Process of the Subsea Robot NOMAD

CHAPTER 8 PRACTICAL DEVELOPMENT AND PERFORMANCE TEST

The concept of the Distributed Marine Observation System involves multiple disciplines such as control theory, computer hardware and software, communications and mechanical design. However, the physical tests in this PhD project focused on the main part of the system, the microcomputer based control system for the depth movement of the subsea robot NOMAD. Other functions were tested in a simulated environment in the laboratory.

In this chapter, a detailed digital simulation at the mechanical component level and a digital analog hybrid simulation at the system management level were carried out to confirm the feasibility of the design of the control system. In these two simulations, identical parameters and computer hardware and software as that mounted on the robot were used. In addition, in this chapter, the setup and some of the results from dry-land tests are presented. At the end of this chapter, physical tests in the deep tank are described. The dynamic performance of the robot NOMAD was recorded under different test conditions. The data records from the physical tests confirm results from theoretical analysis and calculations. Typical data records are also presented in this chapter.

§ 8.1 Detailed Digital Simulation at Mechanical Component Level

Before actually constructing the robot, a detailed digital simulation at mechanical component level written in C^{++} was conducted to make sure the depth control system of the mechanical setup (which was also completed in this PhD project) worked properly. This digital simulation modeled every mechanical action of the control process, such as the flow rate of compressed air, the dimension of the ballast tank, the cross sectional area of the plugs as well as energy consumption. Like the theoretical analysis in Chapter 6, the digital simulation focused on the dynamic behavior of the robot near its equilibrium state.

The flow chart of the digital simulation is shown in Figure 8.1.1. Names beginning with a capital are parameters read from a data file; names with all letters capital are constants; names with all letters lowercase are variables. The physical meaning of each name is as follows:

speedin	is the speed of water particles flowing into the ballast tank;
flowin	is the volume flow rate of water flowing into the ballast tank;
speedout,	is similar to *speedin* ;
flowout	is similar to *flowin* ;
Flowout	is the volume flow rate set by the throttle as shown in Figure 2.3.1;
Tank	is the half height of the ballast tank;
high	is the water level in the ballast tank;
Hole	is the area of the ballast tank holes through which water flows;
Dcom	is the command depth;

delt	is the sampling period;
dnew	is the current depth;
err	is the difference between the command depth and current depth;
err1	is the error one period before the last sampling;
err2	is the error two periods before the last sampling;
merr	is the output of the discrete phase lead compensator;
dmerr	is the increase of *merr* ;
A1,A2,A3	are the parameters for the phase lead compensator;
Tor	is the half-width of the error band;
Dis	is the height of relay control in the variable structure controller;
pulse	is the force on the robot due to air/water flow into or out of the ballast tank;
DENSITY	is the water density;
Base	is the cross sectional area of the ballast tank;
G	is the Gravitational Acceleration;
buoy	is the buoyancy force;
Size	is the maximum buoyancy or weight the ballast tank can provide;
drag	is the hydrodynamic drag force exerted on the robot body;
Wake	is the coefficient of drag force;
Area	is the frontal area of the robot;
uold	is the current speed of the robot;
udot	is the acceleration of the robot.

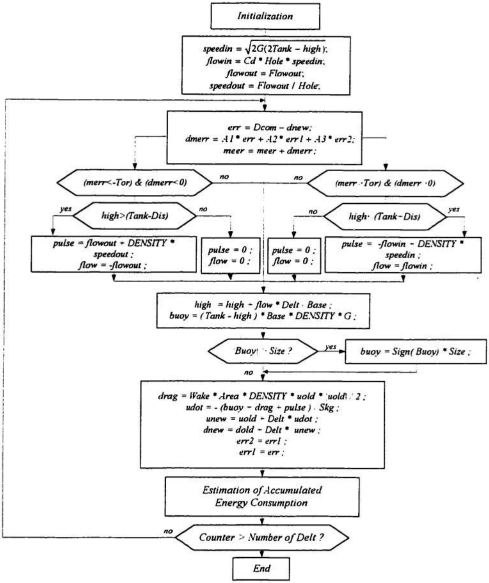
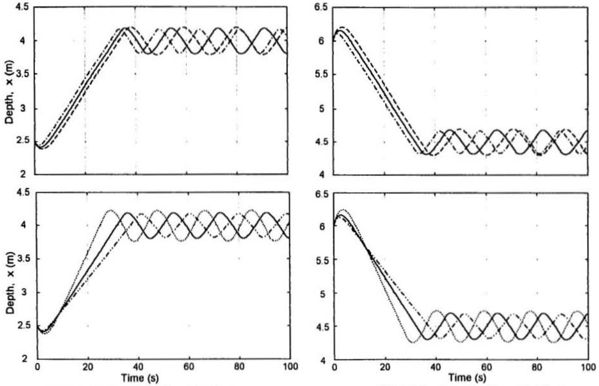

Initialization
speedin = √2G(2Tank – high);
flowin = Cd * Hole * speedin;
flowout = Flowout;
speedout = Flowout / Hole;
err = Dcom – dnew;
dmerr = A1 * err + A2 * err1 + A3 * err2;
meer = meer + dmerr;
(merr<-Tor) & (dmerr<0)
no
no
(merr>Tor) & (dmerr>0)
yes
high>(Tank-Dis)
no
no
high<(Tank-Dis)
yes
pulse = flowout + DENSITY * speedout ;
flow = -flowout ;
pulse = 0 ;
flow = 0 ;
pulse = 0 ;
flow = 0 ;
pulse = -flowin – DENSITY * speedin ;
flow = flowin ;
high = high + flow * Delt / Base ;
buoy = (Tank - high) * Base * DENSITY * G ;
|Buoy| > Size ?
yes
buoy = Sign(Buoy) * Size ;
no
drag = Wake * Area * DENSITY * uold * |uold| / 2 ;
udot = - (buoy – drag + pulse) / Skg ;
unew = uold + Delt * udot ;
dnew = dold + Delt * unew ;
err2 = err1 ;
err1 = err ;
Estimation of Accumulated Energy Consumption
no
Counter > Number of Delt ?
End

Figure 8.1.1 **Flow Chart of Digital Simulation Around the Equilibrium State at the Component Level**

Newton's Second Law gives the plant equation

$$(M + a_2)\ddot{x} = -a_1|\dot{x}|\dot{x} + F_p \qquad \textit{(Equation 8.1.1)}$$

where the equivalent total mass of the subsea robot $(M-a_2)$ and the drag coefficient a_1 are two important parameters. To study the influence of model variation, three groups of different parameters were used in the simulation. The equivalent total mass $(M+a_2)$ varied from 35 kg to 65 kg and drag coefficient a_1 varied from 20 to 60. Time resoonses of the robot under different initial conditions are shown in Figure 8.1.2. These results show the robustness of the controller: system stability is maintained even though the model varies significantly. The results also support the analysis in Section §6.3.2 of Chapter 6: the system only exhibits a small limit cycle oscillation with a period of 20 seconds and an amplitude of 0.25 m. Limit cycles on the plane of $x - \dot{x}$ are shown in Figure 8.1.3. A typical energy consumption chart for these runs is shown in Figure 8.1.4. As mentioned in Chapter 6, the energy consumption only occurs when the control force F_p changes its direction. This can be seen in Figure 8.1.4. Obviously, when scanning the water vertically over a large range, the driving force F_p only changes at the ends of the range. There is no compressed air consumption during movement from one end to the other.

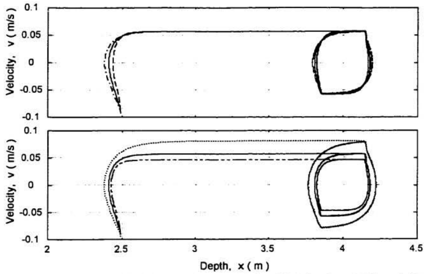


(a) Initial Condition: x(0)=2.5m; v(0)= -0.1m/s

(b) Initial Condition: x(0)=6.0m; v(0)=0.2 m/s

Simulation Based on the Calculated Model as shown in Figure 3.4.3
Based on The Model With 30% Increase of Mass $M+a_2$
Based on The Model With 30% decrease of Mass $M+a_2$
Based on The Model With 50% Increase of Drag Coefficient C_d
Based on The Model With 50% decrease of Drag Coefficient C_d

Figure 8.1.2 Depth Responses Under Different Initial Conditions With Parameter Variations



———— Simulation Based on the Calculated Model as shown in Figure 3.4.3
- - - - - Based on The Model With 30% Increase of Mass $M+a_2$
· · · · · Based on The Model With 30% decrease of Mass $M+a_2$
-·-·-·- Based on The Model With 50% Increase of Drag Coefficient C_d
Based on The Model With 50% decrease of Drag Coefficient C_d
Initial Condition : x(0)=2.5m; v(0) = -0.1 m/s, Commanded Depth: 4.0m

Figure 8.1.3 Small Limit Cycle Oscillation in the D-V Phase Plane With Parameter Variations

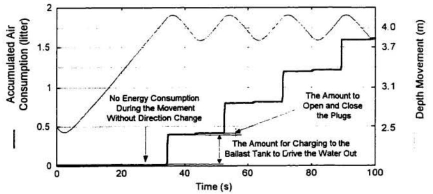


Figure 8.1.4 The Accumulated Energy Consumption Under the Working Pressure

§ 8.2 Real Time Digital Analog Hybrid Simulation

Due to the difficulty of the control task and the complexity of the software structure, substantial laboratory tests on the computing logic and the interaction between external events and the control process were conducted using digital analog hybrid simulation before any in-water tests. A digital analog hybrid simulation is quite different from a pure digital simulation (Lu, 1987). The hybrid one is carried out in the real world. So it is referred to as on-line real time simulation. The computer hardware and the programs used in this simulation were the ones used in the actual robot.

The main purpose of the real time simulation here was not only to study the dynamic performance, but more importantly to check the logic, the memory organization, and the computing time of the software for the onboard micro computer. The author wanted to see if there were any software faults and run time conflicts when the system ran on the real time clock.

The real time hybrid simulation consisted of two major parts:

1. An Analog Computer for Realization of the Low Level Physical Model

An analog computer with three linear operational amplifiers was developed to simulate the dynamic model of the subsea robot as shown in Figure 8.2.1. G_I is an inertial device with a variable gain (when the sliding pointer of R_a arrives at the top

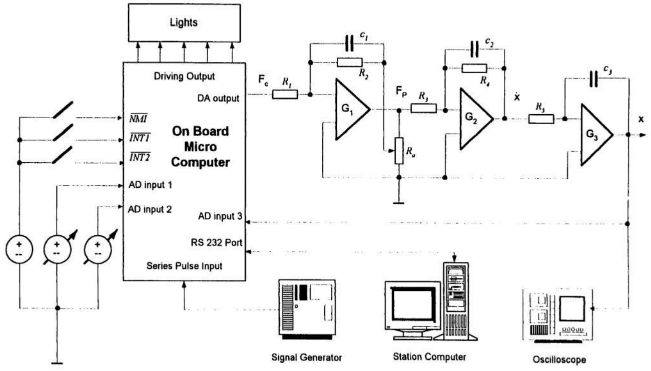


Figure 8.2.1 Real Time Digital Analog Hybrid Simulation and Test for the Onboard Controller

end, the gain equals $\frac{R_2}{R_1}$, while at the ground the gain equals ∞). The transfer function of G_1 is

$$G_1(s) = \frac{R_2 K_w}{R_1(R_2 C_1 s + 1)} = \frac{K_1}{(T_1 s + 1)} \qquad \textit{(Equation 8.2.1)}$$

where,

$$K_1 = \frac{R_2}{R_1} K_w, \qquad K_w \in (1, +\infty); \qquad T_1 = R_2 C_1$$

To make $G_1(s)$ approach the model of the ballast tank, let

$$C_1 = 10\mu f; \qquad [illegible] = 70K\Omega; \quad R_1 = 140K\Omega$$

then

$$G_1(s) = \frac{0.5 \times K_w}{0.7s + 1} \qquad \textit{(Equation 8.2.2)}$$

G_2 is also an inertial device with the transfer function

$$G_2(s) = \frac{K_2}{T_2 s + 1} \qquad \textit{(Equation 8.2.3)}$$

where, $T_2 = R_4 C_2$, $K_2 = R_4 / R_3$.

G_3 is an integral device whose transfer function is

$$G_3(s) = \frac{1}{T_3 s} \qquad \textit{(Equation 8.2.4)}$$

where $T_3 = R_5 C_3$.

To make $G_2(s)G_3(s)$ approach the model of the body of the subsea robot, let

$$C_2 = 20\mu f \qquad R_4 = 330K\Omega \qquad R_3 = 2.5M\Omega$$
$$R_5 = 1M\Omega \qquad C_3 = 1\mu f$$

Then we have

$$G_2(s)G_3(s) = \frac{0.13}{s(6.67s+1)} \qquad \textit{(Equation 8.2.5)}$$

So the analog computer has the transfer function:

$$G(s) = G_1(s)G_2(s)G_3(s) = \frac{0.13}{s(6.67s+1)(0.7s+1)} \qquad \textit{(Equation 8.2.6)}$$

which is the linearized model for the depth control of the subsea robot NOMAD.

2. The Onboard Micro Computer for Realization of the Low Level and High Level Control of the Robot.

The output of the integral device, x, of the analog computer is connected to the A/D input of the micro computer. The voltage mimics the depth of the robot. A signal generator is connected to the serial input of the micro computer. Square waves, whose frequency and pulse width are adjustable, mimic the output of the wave action sensor. Three micro switches are connected respectively to the three interruption request inputs on the Z180 chip, $\overline{INT1}, \overline{INT2}, \overline{NMI}$, to mimic the following external events:

(1) the station computer acknowledges the Hand Shake Request signal from the subsea robot and informs the onboard micro computer to start communication;

(2) the station computer has received all messages without error in the Parity Check and informs the onboard micro computer of the subsea robot to finish sending

data;

(3) the onboard micro computer detects a critical energy failure. For example, the pressure inside of the gas bottle falls below a specified level and this triggers a pressure switch mounted on the input of the second stage pressure reducer.

Five lights and two oscilloscopes are connected to the control input and output terminals of the micro computer, to indicate if the program responds to each interruption request and carries out a proper interruption service routine. After logic reasoning in the high control level, decisions are made to either directly affect the output of the controller or only just change set points and parameters in the T&P Memory Area, in which case the effect will show up later.

The following control functions in high level were tested in the laboratory environment.

(1) Real Time Kernel Control Structure.

(2) The onboard micro computer responds to external event interruption requests.

(a) The onboard computer starts communication upon receiving an acknowledgment from the station computer which responds to the Hand Shake Request signal sent by the subsea robot.

(b) The onboard computer stops sending data upon receiving the signal from the station computer which confirms that the station computer has received all data without error.

(c) The onboard computer starts an Information Protection Routine upon receiving a warning of power exhaustion.

(3) The onboard computer changes the depth set point based on a decision made in a higher control level.

(4) The onboard computer creates a strong water/air jet after detecting a failure to move vertically.

Correct functioning of these tasks was confirmed by indication lights, actions of valves, and flags set in test programs. The results were recorded by oscilloscopes and output files.

§8.3 Tests for Subsystems in a Simulated Environment and Identification for the Body Model

To avoid damage due to subsystem failure in real world tests, many dry-land tests of the subsystems were conducted during the development of the robot NOMAD. To confirm the validity of the robot body model, in-water tests on the actual robot were conducted before the model was used in the simulation and the design for the control algorithm in the final depth control tests.

1. Water Proof Safety Test for the Micro Computer and Electronic Parts Carried by the Subsea Robot

As shown in Figure 2.4.1, there are four water tight boxes attached to the main frame of the robot. These are the computer box, the electronics/sensors box, the driving circuit box, and the communications box. They are connected to each other by cables.

These boxes were designed in this PhD project. Before putting the electronic devices into them, a severe pressure test was conducted as shown in Figure 8.3.1. Two boxes containing dry powder were connected by a communication cable through water-tight connectors. To avoid being crushed by high pressure some supports were put inside these boxes. A high pressure test container containing both boxes was filled with 1m^3 water; the boxes were fully submerged. Internal pressure of the container set by a pressure valve was increased from 1 bar to 8 bar and was allowed to sit at the maximum pressure for 12 hours. After the duration test, the boxes were opened and checked to see if the dry powder was wet or dry.

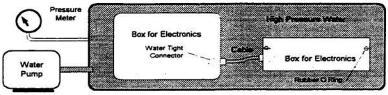


Figure 8.3.1 Durable High Water Pressure Test For the Electronic Boxes and Cable Connectors

After several improvements to the design, the powder remained dry under the pressure of 8 bar. Because of the limit of the test equipment, higher pressure tests were not conducted.

2. Working Pressure Constancy Test

Two pressure reducers are installed on the robot. The first brings the air pressure down from the bottle pressure to 13 bar while the second takes it down from 13 bar to 3 bar, which is the working pressure of the solenoid valves of the robot. Note that the working pressure here is the pressure difference between the air supply from the second pressure reducer and the surrounding water.

Ideally the two pressure reducers should keep the working pressure constant at 3 bar when the flow rate greatly changes or when the stored compressed air is being used up. Also, as the robot moves into a deeper water, where the hydrostatic pressure is higher, the working pressure should still be maintained at 3 bar relative to the static pressure of the environment .

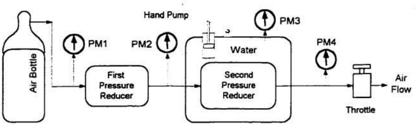


Figure 8.3.2 Test For Maintaining a Constant Working Pressure

The equipment setup for the working pressure constancy test is shown in Figure 8.3.2. It contained four pressure meters: PM_1, PM_2, PM_3, PM_4.

PM_1 read the pressure inside the gas bottle which varies from 13 bar ~ 250 bar;

PM_2 read the middle pressure input to the second reducer. It should be 13 bar relative to the environment.

PM_3 read the simulated environment pressure generated by a hand pump from

0 bar ~ 5 bar corresponding to 0 m ~ 50 m depth;

PM_4 read the output pressure of the second reducer, 3 bar ~ 8 bar, adjustable by the second pressure reducer;

The working pressure is the difference of PM_4 and PM_3. The second pressure reducer was put inside a pressurized container to simulate the environment of deep water. The throttle was used to adjusts the flow rate from *0 liter/min ~500 liter/min.*

Figure 8.3.3 shows that the working pressure relative to the environment held at approximately 3 bar under input and output disturbances.

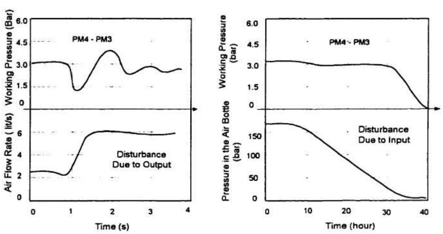


Figure 8.3.3 Variation of the Working Pressure Due to Input and Output Disturbances

3. Ballast Tank Function Test

Devices attached to the ballast tank have the following functions:

(1) To open and close the top and bottom plugs;

(2) To purge the tank using flow from the gas bottle, and take in water from the surroundings;

(3) To create a strong water/air jet by building up pressure inside the tank and then suddenly releasing it;

(4) To read the water level inside the ballast tank;

(5) To read the pressure inside the tank.

(6) To release inside pressure when it exceeds a set value relative to surroundings.

The ballast tank and its components were also designed in this PhD project. One electrical cable runs through the wall of the tank to collect signals from water level sensors inside and pass them on to the micro computer box. Six air tubes pass through the wall of the tank: one for charging the tank with compressed air; four for driving the actuators which open and close the plugs on the top and bottom of the tank; and another for measuring the inside pressure. A pressure release valve is installed at the top of the tank. The ballast tank is pressure tight and electrically isolated.

The test setup is shown in Figure 8.3.4. The following tests were conducted to make sure the tank functioned properly:

(1) Set the Threshold Level of the Pressure Release Valve:

A command from the computer was issued to close both of the plugs on top and bottom. Then the compressed air charge valve was turned on. Then while reading the air pressure inside the ballast tank, the screw on the release valve was adjusted to make release occur at around 3.2 bar.

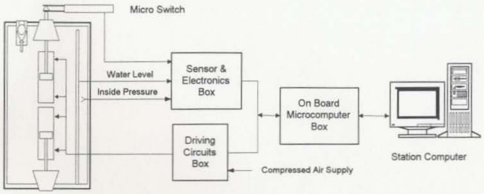


Figure 8.3.4. Setup for The Ballast Tank Function Test

(2) Check the Sealing of Plugs and Connectors:

The ballast tank was charged until its pressure reached approximately *3 bar*. Visual inspection was used to check if there was any leak from the tank. After 3 hours, the pressure inside the ballast tank was still approximately *3 bar*.

(3) Measure Electrical Insulation Resistance:

The ballast tank was fully filled with water and the charging valve was turned on until the pressure inside the ballast tank reached 2 bar. The resistance between the output cable from the water level sensors and the frame of the robot was measured. It was greater than 20 MΩ, which means there were no electrical leaks or shorts.

(4) Measure the Acting Time Delay of the Plugs:

A pulse signal from the station computer was sent to the actuator of the plug, as shown in Figure 8.3.4. The pulse and the response from the micro switch was compared. The time delay of the action of the mechanical driving system was read from the screen of the station computer as shown in Figure 8.3.5. It is almost 40 ms. Compared with the large time constants in the model of the ballast tank (T_{tank} = *0.7sec.*, Equation 3.3.3) and in the model of the robot body (T_{body} = *6.67sec.*, Equation 3.4.6), this time delay could be ignored in the control algorithm.

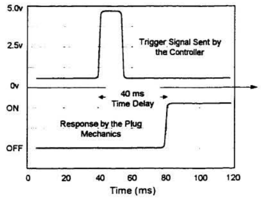


Figure 8.3.5 Test for the Time Delay of the Mechanics

(5) Set the Time Constants t_{out} and T_{tank} :

t_{out} is an important time constant as shown in Equation 3.3.1. It is the time needed to drain about 45% of full ballast tank of water. T_{tank} is the equivalent time constant used in the approximate linear model of the ballast tank. Under the control of the onboard micro computer, the ballast tank was filled until the water level, as indicated by the middle water level sensor, arrived at the half height of the ballast tank. Then the station computer sent the " drain" command to the robot. This command opened the plug at the bottom of the ballast tank and the valve which controlled the charging of compressed air to the ballast tank. The out-going flow rate of the water is roughly equal to the charging flow rate of the coming-in compressed air. The latter can be adjusted by the throttle shown in Figure 2.3.1. By adjusting the throttle and watching output of the water level sensors on the screen of the station computer, t_{out} was set to a value of $2.3sec$. From Section §3.3, this implies that the equivalent time constant T_{tank} is about $0.7sec.$.

4 Identification of the Robot Body Model Parameters by In-water Tests

A nonlinear model for vertical motion of the robot was described in Equation 3.1.2. The model contains two important parameters. In Section §3.1, they were roughly estimated to be $a_1=35\sim350$ and $a_2=17.5$ under some assumptions. To confirm the validity of the model, in-water tests on the actual robot were conducted to get more accurate estimates of a_1 and a_2. Even though the controller discussed in Chapter 5 and 6 should be robust to parameter variation, a well estimated model allows one to put the working point of the controller closer to the center of the possible variation range. In addition, many studies conducted in this project used digital simulations, and, in each case, the model of the controlled object in the real world was described by a set of differential equations with estimated parameters. Having well estimated parameters in these equations made the conclusions more believable.

The in-water tests were carried out in the deep water tank at Memorial

University. A weight with a mass of *3.06 kg* was put on the robot to make it neutrally buoyant. Then the robot was sunk to the bottom of the deep water tank by another extra force. When the onboard computer was ready to record test data, the weight and the extra force were removed. The robot then moved upward. In this case, the driving force F_{p1} was around *-30 N*. The recorded depth and velocity are shown in Figure 8.3.6(a). By picking points around time $t_1 = 13s$, the following estimates of velocity and acceleration were obtained:

$$x_{21}(t_1) = -0.75m/s, \quad \dot{x}_{21}(t_1) = -0.175m \quad s^2 \qquad \textit{(Equation 8.3.1)}$$

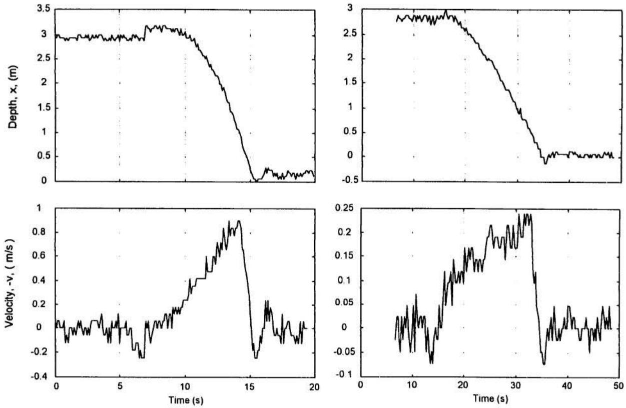
The test was repeated using a smaller weight with a mass of 0.22 kg, which generated 2.1N driving force $F_{p2} \approx -2.1N$. The recorded depth and velocity chart for this case are shown in Figure 8.3.6 (b). In this case, points around time $t_2 = 18s$ gave the following estimates of the velocity and acceleration:

$$x_{22}(t_2) = -0.09m/s \quad \dot{x}_{22}(t_2) = -0.036m/s^2 \qquad \textit{(Equation 8.3.2)}$$

Substitution of $x_{21}(t_1), \dot{x}_{21}(t_1), x_{22}(t_2), \dot{x}_{22}(t_2), F_{p1}, F_{p2}$ into Equations 3.2.3 and 3.2.4 gives

$$\begin{vmatrix} \dot{x}_{21}(t_1) & x_{21}(t_1)|x_{21}(t_1)| \\ \dot{x}_{22}(t_2) & x_{22}(t_2)|x_{22}(t_2)| \end{vmatrix} = \begin{vmatrix} -0.175 & -(0.75)^2 \\ -0.036 & -(0.09)^2 \end{vmatrix} = -0.0188 \neq 0 \qquad \textit{(Equation 8.3.3)}$$

$$\begin{vmatrix} F_{p1} & x_{21}(t_1)|x_{21}(t_1)| \\ F_{p2} & x_{22}(t_2)|x_{22}(t_2)| \end{vmatrix} = \begin{vmatrix} -30 & -(0.75)^2 \\ -2.1 & -(0.09)^2 \end{vmatrix} = -0.94 \qquad \textit{(Equation 8.3.4)}$$



(a) Driving Force F_p = -30N;   Initial Condition x(0)=3m, v(0)=0 m/s   (b) Driving Force F_p = -2.1N;   Initial Condition x(0)=3m, v(0)=0 m/s

Figure 8.3.6   The Records of In water Test For Estimating the Parameters in The Robot Body Model

$$\begin{vmatrix} \dot{x}_{21}(t_1) & F_{p1} \\ \dot{x}_{22}(t_2) & F_{p2} \end{vmatrix} = \begin{vmatrix} -0.175 & -30 \\ -0.036 & -2.1 \end{vmatrix} = -0.7 \qquad \textit{(Equation 8.3.5)}$$

$$\begin{aligned} M + \hat{a}_2 &= -0.94 / (-0.0188) = 50 \\ \hat{a}_1 &= -0.71 / (-0.0188) = 37.5 \end{aligned} \qquad \textit{(Equation 8.3.6)}$$

Since the basic mass of the robot is known to be $M = 35\ kg$ then $\hat{a}_2 \approx 15$. These values of $\hat{a}_1$ and $\hat{a}_2$ roughly match those obtained from the theoretical calculation.

§8.4 In-Water Tests For The Dynamic Performance

1. In-Water Tests in a Shallow Tank

In-water tests were first carried out in a shallow cylindrical tank which was approximately 1.5 m in diameter and 2.5 m in depth. Since the robot NOMAD has a height of 1.2m, the effective depth of the tank was only 1.3m. The goals here were:

(1) To check if there were any leaks in the micro computer box, the electronic sensors box, the driving circuit box, the ballast tank, the first and second stage pressure reducers and all pneumatic and water proof connectors.

(2) To check the insulation of all the electrical connectors.

(3) To study the effect of static electricity on the micro computer. In the test, the cylindrical tank was not connected to ground. So static electricity tended to

accumulate on the robot as it moved up and down. This test made sure the robot would not have trouble when it worked in the Remote Operating Mode. In this mode, the robot could have a ground electrical potential which was different from that of the spot the operator worked at.

(4) To see if in such a small test tank the robot could hold itself in the middle of the tank without touching the bottom or rising to the surface.

(5) To estimate the energy consumption, when the robot was hovering at a specified depth.

(6) To test the reliability of the *38 Kbit/s* communication rate for a RS232 cable immersed in water.

(7) To check the integration of the subsystems of the subsea robot, such as the mechanical subsystem, the pneumatic subsystem, the hydraulic subsystem, the electronic subsystem, including the sensors, amplifiers, micro computer, electrical valves and communication links.

The system checked out OK. Typical data sampled by the depth sensor and recorded by the onboard microcomputer are shown in Figure 8.4.1. They indicate that the robot tended to undergo a limit cycle oscillation with an amplitude around 0.45m and a period around *20sec.*.

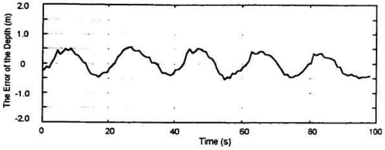


Figure 8.4.1 The Error of the Depth Using the FVSS Controller Without a Phase Lead Compensation Recorded In the Shallow Tank Tests

The control strategy used in the shallow tank tests was the fuzzy logic scheme with a variable structure (staircase plus relay with hysteresis) discussed in Chapter 6, but without phase lead compensation. As discussed in Section §6.3, the linearized transfer function of the robot body without phase lead compensation is:

$$G_p(s) = \frac{0.13}{(0.7s+1)(6.67s+1)s} \qquad \textit{(Equation 8.4.1)}$$

As discussed in Section §6.3.2, the describing function of the relay nonlinearity used in the Fuzzy Variable Structure Switching (FVSS) controller is

$$\frac{-1}{N(A,\omega)} = \frac{-\pi}{4P_{relay}}\sqrt{A^2-\beta^2} - j\frac{\pi\beta}{4P_{relay}}(A \ge \beta) \qquad \textit{(Equation 8.4.2)}$$

where,

P_{relay} is the half height of the hysteresis loop of the relay nonlinearity $P_{relay}=3N$;

β is the half width of the hysteresis loop of the relay nonlinearity $\beta=0.15m$;

A is the amplitude of the possible limit cycle oscillation ;

ω is the frequency of the possible limit cycle oscillation.

As shown in Figure 8.4.2, the intersection of $G_p(s) = -1/N(A, \omega)$ gives $Re_{inter} = -0.17$ and $Im_{inter} = -0.039$ together with a frequency $\omega_o \approx 0.34\ rd/sec$. Equating the real part of Equation 8.4.2 and Re_{inter} gives

$$\frac{-\pi}{4P_{relay}}\sqrt{A^2 - \beta^2} = Re_{inter} \qquad \textit{(Equation 8.4.3)}$$

Substitution of the numbers into Equation 8.4.3 gives $A_o \approx 0.6m$. This is the predicted amplitude of the limit cycle oscillation generated by the fuzzy variable structure switching controller without a phase lead compensation.

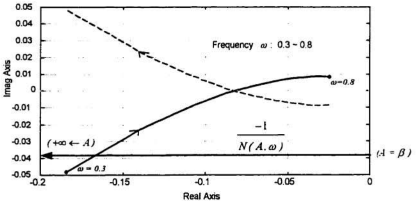


Figure 8.4.2 Nyquist Chart of $G_p(s)$ and the Describing Function of the Relay Nonlinearity with Small Hysteresis

Note that the result derived from Figure 6.3.14 and Equation 6.3.38 gives A_o =

0.27m with a phase lead compensation. Thus the amplitude of limit cycle oscillation can be greatly diminished by the phase lead compensation suggested in Section §6.3.2. Even though the system without the phase lead compensation is still stable in the Lyapunov sense, the amplitude of the limit cycle oscillation exceeds the design specification $A \leq 0.4m$. One should also note that the frequency of limit cycle oscillation without compensation is basically the same as that with compensation. Finally, the physical tests in the shallow tank where there was no phase lead compensation suggest that such compensation is necessary. We did not check this in the shallow water tank.

2. In-Water Tests in the Deep Tank at MUN

After successful tests in the shallow tank, except for the unacceptable large amplitude of limit cycle oscillation, the test site was moved to the deep tank at Memorial University. This tank is *4.3m* deep and has a cross sectional area $5m \times 5m$.

The deep tank was equipped by 2 underwater lights and an underwater video camera. The tests were recorded by the video camera.

In the deep tank tests, the machine code of the control software was downloaded to the memory of the onboard micro computer. Since the micro computer carried by the robot has its own CPU, memory, A/D input and driving output, the RS232 cable which connected the robot to the station computer was used only to send commands through the keyboard of the station computer telling the robot to do things like shifting between Remote Operating Mode and Remote

Automatic Mode. In the Remote Operating Mode, every acting valve is accessible to the keyboard on the station computer through the RS232 communication, while in the Remote Automatic Mode the acting valves are insulated from the station computer. However the parameters in higher control level, such as the set points for the desired depth and the up and low bounds of the scanning range, can be changed through the keyboard of the station computer. A 486 PC computer with MS-Windows worked as the station computer. Several windows were opened to monitor the control process.

Before each in-water test of the subsea robot, its ballast tank was emptied of water and its plugs were both closed. When it was thrown into the water in this state, it floated. The top of the robot stuck about *10 cm* above the water surface. Then, the initialization program started to work. It sent a series of pulses to the valves which opened the plugs for short periods of time allowing a bit of water to flow into ballast tank at each pulse. This process was continued until the subsea robot just started to sink.

(1) Tests for the Remote Operating Mode

In the Remote Operating Mode, data such as the depth, the speed of the robot, the pressure in the ballast tank, the water level in the ballast tank and the status of the acting valves are sent back to the station computer in real time and displayed in a "feedback window" on the screen of the station computer. The operator can make a decision based on feedback window information to command the subsea robot to move up and down, go to a specified depth and then hover at it, or let the robot make strong jets. The direction and intensity of the jets can be set by the operator through the keyboard.

Figure 8.4.3 shows the depth charts generated when the subsea robot was commanded to move up from the bottom of the deep tank to the surface and then back down to bottom again at different speeds: Figure 8.4.3 (a) is a full speed move up; (b) is a slow speed move up; Figure 8.4.3 (c) is a full speed move down; (d) is a slow speed move down.

(2) Tests in the Remote Automatic Mode

The target depth at which the robot is going to hover is specified through a keyboard command by the remote operator. The set point for the depth is then transferred to the micro computer on the robot. After comparing the target depth and current depth obtained from the depth sensor, the micro computer controls the different valves to drive robot to the target depth.

When the robot approaches the target depth, the micro computer shifts the fuzzy control law to relay control with a very small hysteresis loop. Then the robot undergoes a stable limit cycle oscillation with a small amplitude and low frequency. If the oscillation does not exceed the design specification, the system is said to be stable in the Lyapunov Sense.

To diminish the amplitude of the oscillation, a phase lead compensation algorithm was put into the program. The transfer function for this in the continuous frequency domain was given in Equation 6.3.28 and is:

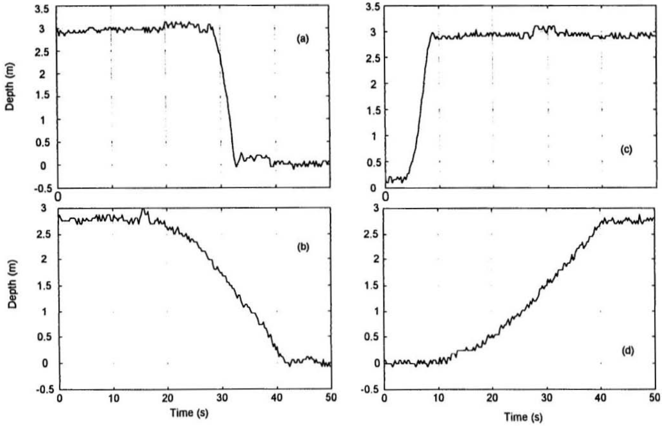


Figure 8.4.3 Recorded Depth When The Subsea Robot NOMAD Moves Up and Down at Different Speeds In the Deep Tank Tests

$$G_c(s) = \frac{Y(s)}{X(s)} = \frac{0.4(2.4s+1)}{0.866s+1} \qquad \textit{(Equation 8.4.4)}$$

To implement phase lead compensation in a digital control loop, Equation 8.4.4 needs to be transferred to a discrete form. Since the control period is much smaller than the time constants of the compensator, a simple backward difference method can be applied to $G_c(s)$ (Frank and Powell, 1994).

The differential equation in the continuous time domain corresponding to Equation 8.4.4 is

$$0.4(2.4\dot{x}(t) + x(t)) = 0.866\dot{y}(t) + y(t) \qquad \textit{(Equation 8.4.5)}$$

Applying the backward difference discrete method to both sides of Equation 8.4.5 gives the difference equation in discrete time domain:

$$\frac{0.866}{T_c}(y(k) - y(k-1)) + y(k) = 0.4\left(\frac{2.4}{T_c}(x(k) - x(k-1)) + x(k)\right) \qquad \textit{(Equation 8.4.6)}$$

Reorganization of Equation 8.4.6 gives:

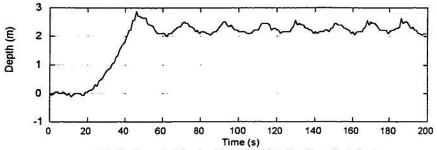
$$y(k) = \frac{T_c + 0.866}{0.866}\left[\left(0.4 + \frac{0.96}{T_c}\right)x(k) - \frac{0.96}{T_c}x(k-1) + \frac{0.866}{T_c}y(k-1)\right] \qquad \textit{(Equation 8.4.7)}$$

where, T_c is the control period. In subsea robot NOMAD, the sampling period is T_s = *25 ms*. The control period is T_c = *100 ms*, that is, the controller makes one modification to the control output once every 4 sampling periods.

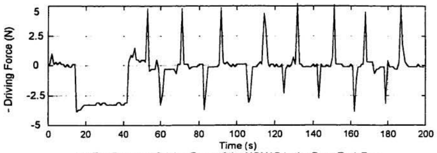
Figure 8.4.4(a) shows depth charts for the case where the phase lead compensation algorithm was added to the control loop. In the charts the robot was commanded to go from the water surface to a depth of 2.25m. As can be seen the robot undergoes a limit cycle oscillation with an amplitude around *0.2 m* which is about half the amplitude (0.45m) without compensation. The chart in Figure 8.4.4(b) shows the control action, the driving force of the robot. Figure 8.4.4(c) shows the recorded Phase Plane plot, which clearly indicates the small limit cycle around the depth of *2.25m*. Figure 8.4.4(b) also indicates the water flow only occurred before each direction change of the movement.

Figure 8.4.5(a) shows a case where the robot was commanded to move up from the bottom of the deep tank to a specified depth of *1.25m* and hover there. Figure 8.4.5(b) shows a case where the robot was commanded to hover at a depth of *1.25m* for *80s* and then go to a depth of *2.75m* and stay there for *15s* and then return to surface at a high speed. Figure 8.4.5(c) shows a case where the robot started at the depth of *1.25m* and was commanded to slowly scan the water body from a depth of *3m* to the surface and then go back to the former depth of *1.25m* and stay there.

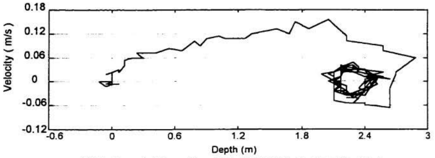
Some functions in the Autonomous Mode were also tested in the deep tank. Figure 8.4.6 shows a case where the robot woke from sleeping and scanned the water from the bottom to a specified depth and then hovered at that depth.



(a) The Recorded Depth of the NOMAD In The Deep Tank Test
When It Moves From Surface to 2.25 m.

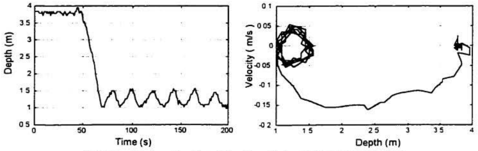


(b) The Recorded Driving Force of the NOMAD in the Deep Tank Test
When It Moves From Surface to 2.25 m.

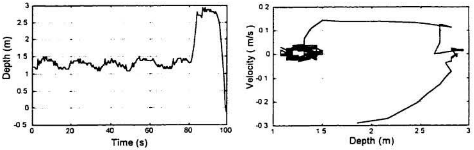


(c) The Recorded Phase Plane Plot of NOMAD in the Deep Tank Test
When It Moves From Surface to the Depth of 2.25m

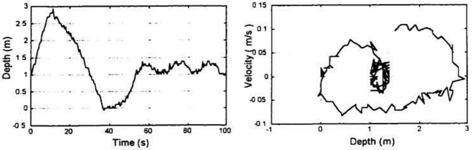
Figure 8.4.4 Deep Tank Test For the Subsea Robot NOMAD
Using the Fuzzy Variable Structure Switching Controller with a Phase Lead Compensation



(a) Waked Up from the Floor of the Deep Tank and Moved to a Specified Depth Then Hovered There



(b) Hovered at a Specified Depth for a Pre-set Period Then Scanned the Water Body and Returned to the Surface



(c) Started at a Specified Depth and Scanned the Water Body Then Hovered at Another Specified Depth

Figure 8.4.5 Deep Tank Tests For NOMAD With Multiple Layered Control System Containing Different Mission Descriptions

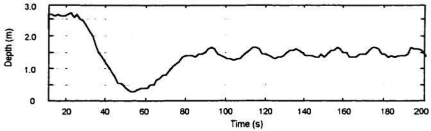


Figure 8.4.6 The robot was woken from sleeping on the sea floor, then scanned the water body, then hovered at a specified depth.

As discussed in Section §7.5 the subsea robot can make a strong water/air jet when it gets stuck in the sediment on the sea floor. The jet would blow away the stuck sediment as well as give the robot an extra initial driving force to make it move upward. The jets could be activated when a higher control level detected a failure to move vertically. Within the multiple layered control system on NOMAD the following steps would be taken to active such a jet:

(1) The failure to move vertically is identified using the Failure Identify Algorithm (FIA).

(2) The FIA sends an interruption request to CPU.

(3) The CPU acknowledges the request and activates the corresponding interruption service routine.

(4) This routine charges the ballast tank to a preset pressure.

(5) The bottom hole of the ballast tank is opened to create the jet.

(6) If successful, the flag of FIA is reset and CPU resumes the previous routine. If there is still a failure to move up, FIA send another interruption request.

The management level of the control system would make a decision whether or not the robot should continue to create the water/air jet, based on the remaining storage of the compressed air and the emergency of the on-going mission.

Figure 8.4.7 shows a case where a water/air jet was made by the subsea robot. At point "a" the robot detected a failure to move vertically. There was no movement after a move up command was issued and there were no other error messages such as power failure. Then the controller assumed that the robot was stuck in the sediment, so that a decision was made to create a strong jet to help the robot break free. The ballast tank started to be charged at point "b" shown in the second chart. At the moment of "c", the ballast tank had been charged to the preset pressure and the robot was ready to make the jet. At point "d" the controller opened the plug at the bottom of the ballast tank. The robot moved up fast, as shown in the first chart, and the pressure inside the ballast tank decreased rapidly.

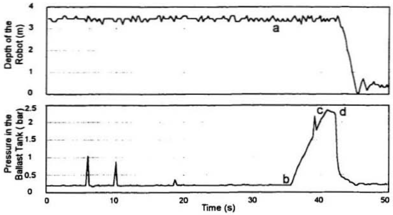


Figure 8.4.7 Water /Air Jet Helped the Robot to Break Away From Being Stuck to the Sea Floor

CHAPTER 9 CONCLUSIONS AND RECOMMENDATIONS

§ 9.1 Conclusions

The physical tests of the robot NOMAD in the deep water tank at Memorial University were quite successful. The recorded data indicated that the goal of this PhD project, to design and develop a prototype of an inexpensive subsea robot for survey type missions has been reached.

To facilitate the mission requirements, great effort was made to develop a high performance onboard micro computer based intelligent control system. To deal with the uncertainty and nonlinearity of the robot model, investigations were conducted to check the potential of strategies based on Neural Networks and Fuzzy Logic. The results of digital simulation indicated that a Neural Network Based Adaptive Controller has great potential to treat unknown nonlinear models. However, there are still many difficulties that have to be overcome before this can be physically applied to subsea robot control. The most important one among these is the computing speed of the onboard micro computer.

As the result of these investigations and studies, a Fuzzy Variable Structure Switching (FVSS) control scheme was developed and fully studied through digital simulations and theoretical analysis. This scheme with phase lead compensation was implemented on the onboard micro computer of NOMAD.

A Real Time Kernel software structure was installed on the onboard micro computer to implement a multiple layered control strategy. Some functions in high level control of NOMAD, such as " Failure Detection" and " Break Free from the Sea Floor", were tested in the deep tank. Others, such as radio communications between the onboard micro computer and the station computer were tested in a simulated environment. These experiments demonstrated that it is feasible to realize artificial intelligence on an inexpensive small subsea robot. It greatly helps the robot to successfully carry out on-going missions and survive in the rough ocean environment.

The feasibility of using an air/water ballast tank instead of a battery/motor system to drive the robot vertically was confirmed. This system is energy saving, easy to maintain, and responds fast. The water flow through the ballast tank only occurs when the direction of motion of the robot changes. It was estimated that about 0.02 standard liters of compressed air would be consumed for each change of direction from upward movement to downward movement and about 0.35 standard liters for each change of direction from downward movement to upward movement. There is almost no energy consumption when the robot is moving in one direction. So NOMAD is quite energy efficient especially for missions where there is a heavy payload or a wide vertical scanning range.

The vast recorded data showed that NOMAD worked quite reliably. All the tests were repeatable and the data closely resembled the results of theoretical analysis. This project fused knowledge from multiple disciplines into an integrated system. Besides the new concept of pneumatic driving system and micro computer based intelligent control system, many technologies and techniques from different areas were introduced and applied in the system development. The subsea robot NOMAD shows great potential for industrial application.

§ 9.2 Recommendations

Future work should focus on industrial applications of NOMAD. These would require a more robust NOMAD. For this it would be better to adopt a hardware structure for the onboard computer control system that uses single chips with their own CPUs and memories to deal with individual low level control tasks. That would allow the main CPU more computing time to work on the management role in high level control tasks.

In most applications, the small limit cycle oscillation like that observed in the tank tests of NOMAD would be tolerable. However, if more precise depth control is required, servo valves and pistons could be used to control the flow areas at the top and bottom of the ballast tank. This would allow much finer adjustment of the water flow into or out of the ballast tank. In this case, a continuous water level detector could also be used in the ballast tank to obtain more precise feedback to the buoyancy/weight controller.

In the past decade, H^∞ Optimal Control has become an effective tool for robust control design. It is, however, too conservative in dealing with parameter uncertainty. To better quantify robustness, a new method, μ-Analysis has been put forward, which is based on a defined structured singular value (Doyle, 1982). The research done by NASA on the HiMAT(Highly Maneuverable Aircraft Technology) remotely piloted vehicle (RPV) indicates that the μ-Analysis and μ-Synthesis Methods provide an effective way to treat the uncertainty in the vehicle system. Subsea robot control faces a similar challenge with regard to model and environment uncertainty. Future work on control could begin by transplanting some of the existing high technologies in the aircraft control area to subsea robot control.

In the current design of NOMAD, the acceleration is not included in the depth control. Using an accelerometer will help to get a better depth control. It could also be used to predict the water level in the ballast tank. To obtain a finer depth adjustment, extra small holes could be installed on the top and bottom of the ballast tank in addition to the existing holes. When the robot reaches the vicinity of the equilibrium, the small holes are used to draw in or drain out a little bit of water, which will keep the robot hover at the specified depth.

In the current design of the NOMAD radio communication system, the robot has to emerge to the surface to communicate with the station. In future designs, one could use acoustic waves to carry out communication under water and then relay data to the shore station from an acoustic-radio converter floating on the water surface.

The design and development of the robot NOMAD with its micro computer based control system was a challenge. The work done in this PhD project will hopefully contribute to industrial applications and academic research.

So far NOMAD has not undergone a real sea test. A more robust prototype would have to be constructed for such tests. The subsea robot group at Memorial University is presently seeking funds to develop such a prototype.

REFERENCES

Allmendinger, E.E. (1990). " Submersible Vehicle Systems Design," The Society of Naval Architects and Marine Engineers, Jersey City, NJ., USA, pp. 5-31.

Allocca, J. A. and Stuart, A. (1984). "Transducer Theory and Application," Prentice-Hall Company.

Atherton, D. P. (1975). " Nonlinear Control Engineering," Van Nostrand Reinhold Inc. New York, NY, USA.

Berenji, H.R. (1992). " Fuzzy Logic and Neural Networks for control Systems," IEEE Press, ISBN 0-7803-0335-0, Piscataway, NJ, USA.

Bielawski, L. and Lewand, R. (1991). " Intelligent Systems Design, Integrating Expert Systems," John Wiley and Sons, Inc., New York, NY, USA.

Brogan, W.L. (1991), " Modern Control Theory," The 5th Edition, Prentice Hall, Englewood Cliffs, New Jersey. pp. 443-498.

Busby, F., and Vadus, J. R. (1990). "Autonomous Underwater Vehicle R&D Trends," Sea Technology, Vol. 31, No. 5, pp. 65-73.

Cox, E. (1992). "Fuzzy Fundamentals," IEEE Spectrum, Vol. 29, No. 3, pp. 58-61.

Cristi, R., Papoulias, F. A., and Healey, A. J. (1990). "Adaptive Sliding Mode Control of Autonomous Underwater Vehicles in the Dive Plane," IEEE Journal of Oceanic Engineering, Vol. 15, No. 3, pp. 152-160.

Curtin, T., and Bellingham, J. (1993). " Autonomous Oceanographic Sampling Networks," Oceanography, Vol. 6, No.3.

Development Group, (1994). " Development of a Small Manned Space Pod," Faculty of Engineering, Memorial University of Newfoundland, St.John's, NF, Canada.

DeYong, M., and Polson, J. (1992)." Fuzzy and Adaptive Control Simulation for a Walking Machine," IEEE Control Systems Magazine, June 1992.

Doyle, J. C. (1982). " Analysis of Feedback Systems with Structured Uncertainties," Proc. IEE, Pt D, Vol.129, pp.242-250.

Embree, P.M. and Kimble, B. (1991). " C Language Algorithms for Signal Processing," Prentice-Hall International (UK) Limited, London UK.

Frank G. F. and Powell, J.D. (1994) " Feedback Control of Dynamic Systems," The 3rd edition, Addison Wesley Publishing Company, Reading, MA, USA.

Funahashi, K., and Nakamura, Y. (1993). " Approximation of Dynamical Systems by Continuous Time Recurrent Neural Networks." Neural Networks Magazine, Vol 6, pp. 801-806.

Hall, W., and Adams, M. (1992). "Autonomous Vehicle Software Taxonomy," Proceedings of the 1992 Symposium on Autonomous Underwater Vehicle Technology,

Henrickson, Mark F., and Dzielski, John E. (1995). " Development of a System Architecture for an Underwater Autonomous Testbed," Proceedings of the 9th International Symposium on Unmanned Untethered Submersible Technology, NH., USA, pp. 343-349.

Hinchey, M. (1994). " Supervisory Control Strategies for Subsea Robots," The Proceedings of the 21th Annual Technical Symposium and Exhibition of the Association for Unmanned Vehicle Systems, Detroit, MI, USA, pp. 403-410.

Hinchey, M. J., Muggeridge, K. J. (1994). "Potential for Subsea Robot Control", Ocean Engineering Journal, Pergamon Press. Vol. 22, No.2, ISSN 0029-8018

Hinchey, M. and Goteti, R. (1991). " Submergence Depth Control of a Free-fall Submersible Escape Capsule for an Offshore Oil Rig," Proceedings of the Ocean Engineering Conference 91, Seattle, Washington, USA

Hinchey, M. J., Muggeridge, K. J. and Rivera C. (1992). "Subsea Robotics Research at OERC/MUN," Proceedings of the Third Newfoundland Electrical and Computer Engineering Conference, sponsored by the Newfoundland and Labrador Section of IEEE, St. John's, Newfoundland.

http://www.hiberniabank.com/, (1995). " Base Line Study at Hibernia Successfully Completed," Hibernia Management and Development Company Ltd., St.John's, NF, Canada.

http://www.hiberniabank.com/, (1995). "Assessment Methods for Benthic Impacts of Salmon Aquaculture," Hibernia Management and Development Company Ltd., St.John's, NF, Canada.

Hush, D.R. and Horne, B.G. (1993). " Progress in Supervised Neural Networks," IEEE Signal Processing Magazine, No.1, pp.8-39.

IEEE, (1992a). Neural Networks: Theoretical Foundations and Analysis, ed. C. Lau, IEEE Press, Piscataway, New Jersey.

Ishibuchi, H. (1993). " Neural Networks that Learn from Fuzzy IF-Then Rules," IEEE Transactions on Fuzzy Systems, Vol.1, No.2, pp. 85-95.

Jin, L., and Gupta, M.M. (1995). " Intelligent Control for Nonlinear Systems using dynamic Neural Networks with Robotic Applications," Intelligent Automation and Soft Computing Journal, ISSN 1079-8587, Volume 1, Number 2, pp. 121-143.

Lee, C. C. (1990). "Fuzzy Logic in Control Systems: Fuzzy Logic Controller - Part I and II," IEEE Transactions on Systems, Man, Cybernetics, Vol. 20, No. 2, pp. 404-435.

Leonard, J.J. (1995). "Bottom Following for Survey-class Autonomous Underwater Vehicles," Proceedings of the 9th International Symposium on Unmanned Untethered Submersible Technology, NH., USA. pp. 327.

Lu, Z. (1987), " Micro Computer Aided Analysis of Control Systems", Electrical Automation Magazine, ISSN 1000-3886, Vol. 2, pp. 6-12.

Lu, Z. and Tian, Z. and Chai, G. (1991) " Theory and Design of Control Systems in Engineering," Publishing House of Shanghai Jiao Tong University, University Text Book ISBN7-313-00795/TB.1, pp. 305-355.

Lu, Z. (1993). " Development of a Subsea Robot with Intelligent Control Systems," the Proposal for Doctoral Research, Faculty of Engineering, Memorial University of Newfoundland, St.John's, NF, Canada.

Lu, Z. and Hinchey, M. and Friis, D. (1994) " Development of a Pneumatic Subsea Robot," The Proceedings of the 21th Annual Technical Symposium and Exhibition of the Association for Unmanned Vehicle Systems, Detroit, MI, USA, pp. 369-374

Miyamoto, H., Kawato, M., Setoyama, T., and Suzuki, R. (1988). "Feedback-Error-Learning Neural Network for Trajectory Control of a Robotic Manipulator," Neural Networks, Vol. 1, pp. 251-265.

Muggeridge, K. (1994). "Control of Subsea Robot", M.Eng. Thesis, Memorial University of Newfoundland.

Muggeridge, K. J., and Hinchey, M. J. (1992). "A New Jet Propulsion Device for Small Subsea Robots," Proceedings of the 1992 Symposium on Autonomous Underwater Vehicle Technology, sponsored by the Oceanic Engineering Society of IEEE, Washington, DC.

Narendra, K. S., and Parthasarthy, K. (1991). " Gradient Methods for the Optimization of Dynamical Systems Containing Neural Networks,' IEEE Trans. on Neural Networks, Vol.2, No.2, pp. 4-27.

Nomoto, J., and Hattori, M. (1986). "A Deep ROV Dolphin 3K: Design and Performance Analysis," IEEE Journal of Oceanic Engineering, Vol. OE-11, pp. 373-391

Ogata, Katsuhiko (1987). " Discrete Time Control Systems," Prentice-Hall International (UK) Limited, London, UK.

Phillips C. L. and Harbor, R. D. (1991). " Feedback Control Systems," Second Edition, Prentice-Hall International (UK) Limited, London, UK.

Sarpkaya, T., and Isaacson, M. (1984) " Mechanics of Wave Forces on Offshore Structures, " Van Nostrand Reinhold Company, New York, NY., USA.

Self, K. (1990). "Designing with Fuzzy Logic," IEEE Spectrum, Vol. 27, No. 11, pp. 42-44, 105.

Slotine, J. J. E. (1983). "Tracking Control of Nonlinear Systems Using Sliding Surfaces," Ph.D. Thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts.

Slotine, J. J. E. (1984). "Sliding Controller Design for Nonlinear Systems," International Journal of Control, Vol. 40, No. 2, pp. 421-434.

Slotine, J. J. E. and Coetsee, J. A. (1986). "Adaptive Sliding Controller Synthesis for Nonlinear Systems," International Journal of Control, Vol. 43, No. 6, pp. 1631-1651.

Slotine, J. J. E., and Sastry, S. S. (1983). "Tracking Control of Nonlinear Systems Using Sliding Surfaces, with Application to Robot Manipulators," International Journal of Control, Vol. 38, No. 2, pp. 465-492.

Togai InfraLogic, Inc. (1993). " TIL Shell User's Manual," Version 3.0.0., Irvine, CA, USA.

Togai, M. (1991). "Fuzzy Logic: Applications and Perspectives," The 41st Video Conference Seminar via Satellite, produced by IEEE, Inc., Piscataway, New Jersey, pp. 1-50.

White, Frank M. (1986). "Fluid Mechanics," McGraw-Hill Book Company, New York, NY., USA.

Williams,R., and Zipser, D. (1989). " A Learning Algorithm for Continually Running Fully Recurrent Neural Networks," Neural Computation Magazine, Vol 1, pp. 270-280.

Xu, Min, and Smith, S. E. (1995). " A Fuzzy Sliding Controller for autonomous Underwater Vehicles," Proceedings of the 9th International Symposium on Unmanned Untethered Submersible Technology, NH., USA. pp.178-185.

Yoerger, D. R., Bradley, A. M. and Walden, B. (1991). " The Autonomous Benthic Explorer", Unmanned System, Spring, 1991, pp. 17-23.

Yoerger, D. R., Cooke, J. G. and Slotine, J. -J. E. (1990). "The Influence of Thruster Dynamics on Underwater Vehicle Behavior and Their Incorporation Into Control System Design", IEEE J. Ocean Eng., Vol. 15, pp. 167-178.

Yuh, J. (1990). "A Neural Net Controller for Underwater Robotic Vehicles" IEEE Journal of Oceanic Engineering, Vol. 15, No. 3, pp. 161-166.

Yuh, J. and Lakshmi, R. (1993). "An Intelligent Control System for Remotely Operated Vehicles", IEEE J. Ocean Eng., Vol. 18, pp. 55-62.

Zworld Engineering, (1992). " Integrated C Development Environment for Embedded Systems," Z-World Inc., Davis, CA, USA.