CONTROL OF AN HYDRAULIC SUBSEA ROBOT

YIFEI ZHANG

Control of an Hydraulic Subsea Robot

by

©Yifei Zhang, B.Eng.

A thesis submitted to the School of Graduate Studies
in partial fulfillment of the requirements for the degree of
Master of Engineering

Faculty of Engineering and Applied Science
Memorial University of Newfoundland
August, 1994

St. John's Newfoundland Canada

ISBN 0-612-0[illegible]33-0

Abstract

Remotely operated vehicles (ROVs) and underwater robots have been greatly involved in scientific related research and commercial utilization of underwater resources for the past few decades. They are operated at scenes where conditions are beyond the physical limitation of human divers or are too hazardous for them.

To gain experience with the design, fabrication, control and operation of subsea robot, an hydraulic subsea robot is being developed at Faculty of Engineering and Applied Science of Memorial University of Newfoundland. This uses a plunger/waterjet device to control depth. When operating with the plunger, variation of displaced volume generates a positive or a negative buoyancy and thus controls the robot's vertical movement and depth. When operating with waterjet, movement is controlled by the combination of the buoyancy and the thrust generated from the waterjet squeeze.

This thesis explains in detail the development of the control system of the robot, from simulation to test in a water tank, and reviews the systems stability, time response and frequency response performance.

Acknowledgments

I would like to express my thanks to my supervisor, Dr. Michael Hinchey, for allowing me to get involved in this project, helping me at every stage of the development, and for his constructive comments and valuable suggestions.

Special thanks to Mr. Austin Bursey and other technical staff for their concrete help and contribution in many of my laboratory experiments. I also wish to express my thanks to my friend and colleague, Mr. Zhongqun Lu, for his encouragement and continuous support.

Finally, I want to thank Dr. Michael Hinchey and the Faculty of Engineering and Applied Science for providing the financial assistance needed to complete this project.

Contents

List of Figures

Chapter 1

Introduction

1.1 Problem Description

The ocean covers two-thirds of the planet and contains a vast amount of mineral resources, oil and gas reserves. Unlike outer space, the ocean still remains a poorly explored realm. One reason is the environment is hostile and complex when compared to space. The medium is highly corrosive and the pressures can exceed many thousands of pounds per square inch. Agents operating underwater must contend with currents, thermoclines, unknown obstacles, and changing bottom topography.

Currently, remotely operated vehicles (ROVs) and remotely operated underwater robots are being used in ocean exploration. However, it is expected that autonomous underwater vehicles (AUVs) and untethered unmanned vehicles (UUVs) will soon compete for this role. Their communication with the mother

ships does not rely on an umbilical cable. Thus, they have overcome some of the limitations of ROVs and underwater robots. Their working space is expanded and they are not restricted by the length of cables connecting them with their mother ships. They require little or no support from surface vessels, and have no tether to become entangled or to break(Blidberg, 1991). The autonomy of operation has been greatly increased.

With respect to the level of autonomy, most existing ROVs and underwater robots are classified as tethered supervised vehicles (Yuh, 1989), which indicates that they are connected to a mother ship by a cable through which all the communication, data and power are transmitted. These vehicles can replace human divers in many cases in shallow and deep water where environments are too hazardous for them. They perform an enormous variety of tasks such as scientific research, inspection and repair of offshore platforms, pipeline burial, mine disposal, find and identify wrecks and items lost or placed on the sea floor, burying, inspection and recovery of telecommunication cables.

Several ROVs or underwater robots were developed in recent years. *EAVE*, an unmanned, untethered vehicle developed by University of New Hampshire, is a test bed for the development of autonomous control concepts. *DOLPHIN 3K*, a deep tethered ROV, developed by Japan Marine Science and Technology Center (JAMSTEC) is used for ocean bottom surveys. *HYDRO 150*, a tethered vehicle developed by Hydro Products Inc., and *SUPER C'CAT*, a tethered underwater vehicle, developed by USAL Inc. are both used for inspection (Yuh, 1989).

Due to the reason that the ocean is a non-uniform, unpredictable environment with great uncertainty and complex disturbances from the ocean's multidi-

rectional currents and tethers, we cannot adapt the high technologies developed for on-land vehicles or robots directly to the ROVs and underwater robots. Many ROVs and underwater robots have unstreamlined features, their hydrodynamic coefficients are difficult to ascertain, and the dynamic behavior of the vehicles is highly nonlinear. Those make precise motion and positioning more difficult. To improve performance, it is the requirement that ROVs and underwater robots have a reliable propulsion and a robust control system.

Most ROVs and underwater robots use propellers or thrusters for propulsion. They are ducted to improve performance whose characteristics are generally defined by their open-water efficiency. They operate with high efficiency only if they are well designed and operate under favorable conditions. Any significant departure from this operating point leads to a corresponding drop in efficiency. Usually a free-swimming ROV has 4 to 8 propellers in order to gain six degrees of freedom motion.

Ducted propellers or thrusters are currently being utilized by most underwater robots and ROVs, but people usually underestimate their defects.

- They are expensive and physically complex, so their numbers are always kept to minimum (Farbrother and Stacey, 1993).
- Thrusters are subject to the thruster/vehicle interaction, meaning the blockage of the thrusters region of influence by other structural components (Farbrother and Stacey, 1993).
- Extra mechanical complexity can introduce the potential for a greater degree of nonlinearity in the thruster response (Farbrother and Stacey, 1993).

- Significant velocity differential when fluid enters or leaves the thruster ducting will induce a change in momentum that is felt as an additional drag force (Farbrother and Stacey, 1993).

- They are subject to serious degradation due to axial and cross-flow effects that make them not the ideal force-producing actuators (Yoerger, *et al* 1990).

- The thruster dynamics are dependent on propeller speed and the effect of thruster dynamics to the vehicle motion becomes significant at low velocity (Yoerger, *et al* 1990).

- When hovering, nonlinear thruster response causes the vehicles to oscillate about the desired position with a very repeatable frequency and magnitude (Yoerger, 1990).

- Their performance has been limited by cavitation.

Besides the propulsion system, the control system is also one of the most critical subsystems of ROVs and underwater robots. ROV and robot dynamics are nonlinear due to inertia, buoyancy and hydrodynamic effects. Thus, the control of such vehicles is difficult. With the increased utilization of the vehicles in subsea applications, the development of autonomous underwater vehicles becomes highly desirable to enhance operator efficiency. ROVs and robots are always unstreamlined and adding more sensors and other components can cause significant dynamic change. A good control system can greatly reduce the human pilot's workload, adapt the vehicle in the environment when parameters change and enhance the vehicle's autonomy. The current control systems of ROVs and underwater robots are

usually not robust and reliable enough which make underwater operations more difficult to handle.

1.2 Literature Review

Vehicles must maintain acceptable performance in the face of parametric uncertainty and unmodelled disturbance. To achieve the required robustness of the control system, adaptive elements have to be incorporated into the control system, so the controller is capable of learning about a process on-line and modify its own behaviors with the goal of enhancing system performance (Farbrother and Stacey, 1993). Sliding mode, fuzzy logic and neural network control have the characteristics of adaptivity, and their usage in underwater vehicles control is expanding.

A lot of work has been conducted in recent years and many literatures published have addressed the usage of more advanced control strategies together with the classical control strategies in ROVs and underwater robots to enhance propulsion system performance and vehicle autonomy (Hinchey, 1994 and Muggeridge, 1994).

Yoerger, *et al* (1990) tested two conventional linear controllers and adaptive sliding switching controller in a computer simulation of thruster dynamics. Results showed that the adaptive sliding controller was effective over the entire operating range and can compensate for uncertainties and degradation of the thruster. Yoerger (1990) solved the nonlinear thruster response of a vehicle using small thrusters for fine maneuvering. Yuh and Gonugunta (1992) designed a neural

net controller for an underwater robotic vehicle(URV) using a parallel recursive adaptation algorithm. Computer simulation showed that the control system can provide high performance in the presence of the effect of thruster dynamics and saturation.

Yoerger and Slotine (1985) proposed three single-input and single-output(SISO) controllers using sliding mode control methodology. The influence of parametric uncertainty and the use of simplified models were demonstrated in computer simulation using a nonlinear model of the University of New Hampshire's Experimental Autonomous Vehicle(EAVE). Yoerger, *et al* (1986) demonstrated two important elements of the supervisory control system: the close-loop control of vehicle translation in pool tests with a prototype vehicle, and the interactive trajectory in simulation trials. The control technique they used was sliding mode control, and results showed vehicle operations were made more efficient and simpler. Yoerger and Newman (1989) tested both basic automatic control and interactive supervisory control techniques for hovering and tracking of the vehicle *JASON ROV*. Results showed that the vehicle had better performance to interact with the environment with the supervisory control techniques than with the conventional control technique. Yuh (1990) proposed a multilayer neural network controller for a URV, and error-back propagation method was used as the training algorithm. Simulation results showed that the control system could respond to changes in the vehicle and its environment, and the network weights could be adjusted to provide the proper control signal. Yoerger, *et al* (1991) applied adaptive sliding control to the autonomous benthic explorer (ABE). The tests showed that the control system could detect changes in control gain on multiple degrees of freedom, automatically refine the vehicle model until performance achieved a level corresponding to a

prescribed level of model uncertainty, then automatically revert to a nonadaptive configuration. Zheng (1992) implemented layered control, on the *ARCS* vehicle and *THESEUS* vehicle. Test and validation of the control system showed that layered control is a simple and powerful concept. The quickness with which it can be implemented makes it a good candidate for practical AUV application. Yuh and Lakshmi (1993) developed a multilayered neural network controller using a critic equation for an ROV. Three learning algorithms: error back propagation algorithm(EBP), parallel recursive prediction error algorithm(PRPE) and modified parallel recursive prediction error algorithm(MPRPE) were tested. Effects of the number of layers in the network, the number of neurons in the hidden layer, initial weights for the network and the critic coefficient were investigated by computer simulation. Results showed that the neural net controller can be effective in the presence of parametric uncertainty and random noise. Yuh and Choi (1993) have developed two controllers for an omni-directional underwater robotic vehicle: a nonlinear controller and a discrete-time adaptive controller. Since the hydrodynamic coefficients were poorly known, the Parameter Adaptation Algorithm(PAA) was applied in the adaptive controller.

All of the control strategies outlined in this section have been shown to possess considerable robustness to parametric uncertainty and are capable of dealing with nonlinearities. With regard to underwater vehicles, they have been tested only in computer simulations. The much harder physical implementation has yet to be carried out.

1.3 Scope of the Work

At Memorial University of Newfoundland (MUN), an hydraulic subsea robot is being developed. The project was initiated by Dr. M. Hinchey and one of his students K. Muggeridge three years ago. K. Muggeridge designed the mechanical setup, as shown on page 14 in next chapter. Due to the lack of complete software and hardware in the control system, this rig could not be tested in a water tank at that time. After I got involved in this project, I designed the control system, and tested this system together with the rig in a small water tank.

Right now, the robot does not have a manipulator and can not do any kind of work in the water. In the present work we focused only on depth control. The robot can be used later on to gather data (pressure, temperature, conductivity, salinity) in shallow water if proper sensors are implemented. The purpose of the project was to gain experience with the design, fabrication, control and operation of a subsea robot.

In designing its propulsion system, Muggeridge (1994) deviated from the ordinary motor driven propellers or thrusters and used an hydraulic system, which includes an accumulator, a storage chamber, an hydraulic actuator and a servovalve together with a plunger/duct system. The plunger contacts the duct tightly by three O'rings as shown on page 17. The plunger is screwed on the actuator piston rod and moved inside this duct whose diameter is 6 inches. For depth control this movement can change the buoyancy of the robot or create a strong waterjet. The long term goal of the project is to compare the buoyancy and waterjet control, and the goal of my work was to check the buoyancy and have a preliminary look at

waterjet.

Compared with propellers, our duct or waterjet propulsion system has the following advantages and disadvantages,

Advantages

- It is physically much simpler and easier to attach on ROVs and robots than propellers.
- Its power supply is hydraulic oil. There is no need for DC motor. The size of the tether can be reduced, together with the nonlinear effects caused by the tether.
- When operating, it has much smaller vibration than propellers.
- Degradation of performance caused by the cavitation is greatly reduced.
- Cleaning ocean appendages is much easier for the duct than the propellers.
- When operating, it is safer than propellers in shallow water.

Disadvantage

- The robot at this moment can only move vertically in the water, thus its maneuverability is limited.
- Its efficiency is usually lower than propellers. The thrust generated by propellers is always higher than other marine propulsion systems.
- In our project, the amount of oil in the accumulator determines the task duration. Presently this is much shorter than propeller operated systems.

- Due to the high pressure in deep water, the velocity of waterjet can be reduced. Thus the device in deep water will not perform as efficiently as in shallow water.

As for the control system of the robot, we used conventional control strategies. These strategies give good performance and have been widely used in industries for decades. The stability of the control system was checked using Nyquist stability criterion and nonlinear describing function. The transient response and frequency response performance was investigated in simulation and test.

1.4 Outline of This Thesis

This thesis gives a detailed description of the control system for our robot including its computer simulation, stability analysis and real time control and test in a water tank. It contains five chapters including the introduction.

Chapter two introduces the dynamics equations of the robot and demonstrates classical control strategies such as proportional-integral-derivative (PID) and proportional control of buoyancy in computer simulation. The stability is analyzed using the nonlinear describing function with the Nyquist stability criterion for buoyancy and waterjet devices.

Chapter three details the components of the robot and the equipment needed to test it in a water tank. Procedures for testing the hydraulic system using proportional control action and simulation are presented next. Stability is examined by Bode diagram, and the buoyancy and waterjet tests in the water tank are described

respectively.

Chapter four discusses the advantages and disadvantages of the system based on the results we got from the computer simulation and the experiment in the water tank. Potential modifications to the setup to lessen the disadvantages are elucidated.

Chapter five gives the conclusion from the project, and illustrates the experience we have gained through this project, as well as our future works.

Chapter 2

System Simulation and Stability Analysis

2.1 Robot Description

The configuration of the robot is a cuboid shape with mainly a supporting frame, two cylinders, an actuator, a servovalve and a duct/waterjet device as depicted in Figure 2.1. The robot can be expanded by installing additional components. The supporting frame is welded together, and is made of aluminum. The dimension of the rig is 0.6m in length, 0.45m in width and 1.7m in height, and the weight is approximately 160 kg.

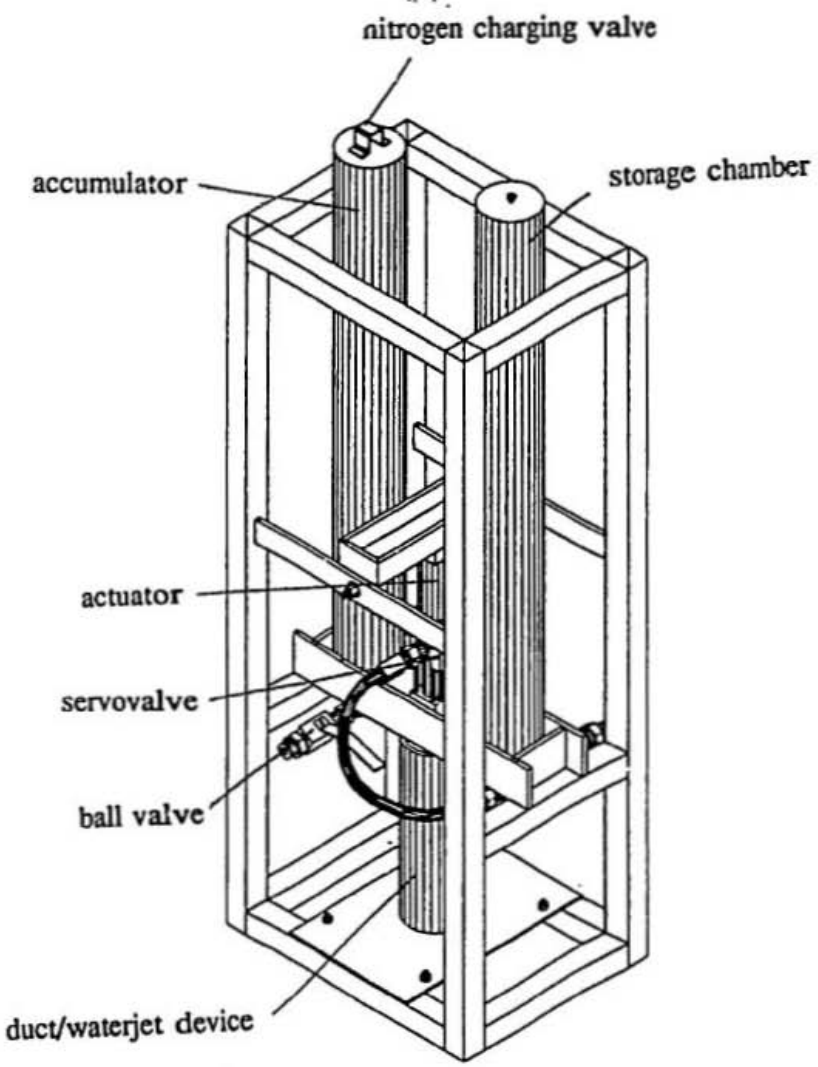

Figure 2.1: Configuration of the robot

The two cylinders, the actuator and the servovalve make up the hydraulic system. The shorter one of the two cylinders is an MTS accumulator. Inside, when fully charged, it has 3,000 psi high pressure oil under nitrogen gas. The nitrogen is charged through a small pneumatic valve on top of the accumulator. The oil is used to activate the piston rod of the actuator. The velocity and position of the piston rod are controlled by the servovalve. After the oil goes through the actuator and servovalve, it goes to the longer cylinder. This cylinder is just an empty chamber whose purpose is to store the low pressure oil which has passed through the actuator and servovalve. We call this cylinder the storage chamber. There is a 10mm diameter hole with thread on top of the storage chamber. It is usually kept closed by a bolt except when the oil is drained from the chamber, in which case the bolt is taken off so that the air pressure inside this chamber is the same as the air pressure outside and the oil can be easily released. As more oil passes through the servovalve, the pressure of the oil left inside the accumulator goes down. When the minimum of 1000psi operating pressure is reached, the system stops to work. The oil inside the storage chamber needs to be drained, and accumulator needs to be recharged.

The actuator is placed between the accumulator and the storage chamber, with the servovalve attached on it. The oil from the accumulator to the actuator and from the actuator to the storage chamber passes through two rubber pipes and each pipe is switched by a ball valve. These two ball valves are only opened when the experiment is in process, so the oil can be circulated from the accumulator to the storage chamber via the servovalve. Otherwise, the two valves are usually closed. There are another two ball valves connected to the bottoms of the accumulator and the storage chamber respectively. When the ball valve connected

to the accumulator is open, oil can be pressed into the accumulator from a pump and after the pressure of oil reaches 3,000 psi, it has to be kept closed all the time. The ball valve connected to the storage chamber is kept closed unless oil in the chamber needs to be drained. All the four ball valves are designed to hold 3,000 psi oil pressure.

A plastic duct is assembled on a flat plate by several small bolts and stuck to it by the sealent *Silicon*. The flat plate is made of aluminum and is fixed by four connecting bolts at the bottom of the supporting frame. Its inner diameter is 6 inches and the length is 15 inches. The plunger is 2 inches high with a diameter slightly less than 6 inches. It is also made of plastic. The plunger is tightly contacted with the duct by three O'rings, and it is driven to move inside the duct by the piston rod. The plunger separates the plastic duct into two parts. Above the plunger is the upper part which opens to the water. Below the plunger is the lower part and only contains air. For buoyancy control tests, tygon tubing attached to a small vent hole is used to connect this air to atmosphere. When the piston rod moves, the amount of air and water inside the duct changes, and so does the buoyancy. With the variation of the buoyancy, the robot can move vertically in the water.

For waterjet tests, a 1.5″ diameter hole was drilled into the bottom plate. When the plunger moves inside the duct, the robot can be driven by the thrust of the waterjet through this hole. The duct with the plunger is shown in Figure 2.2.

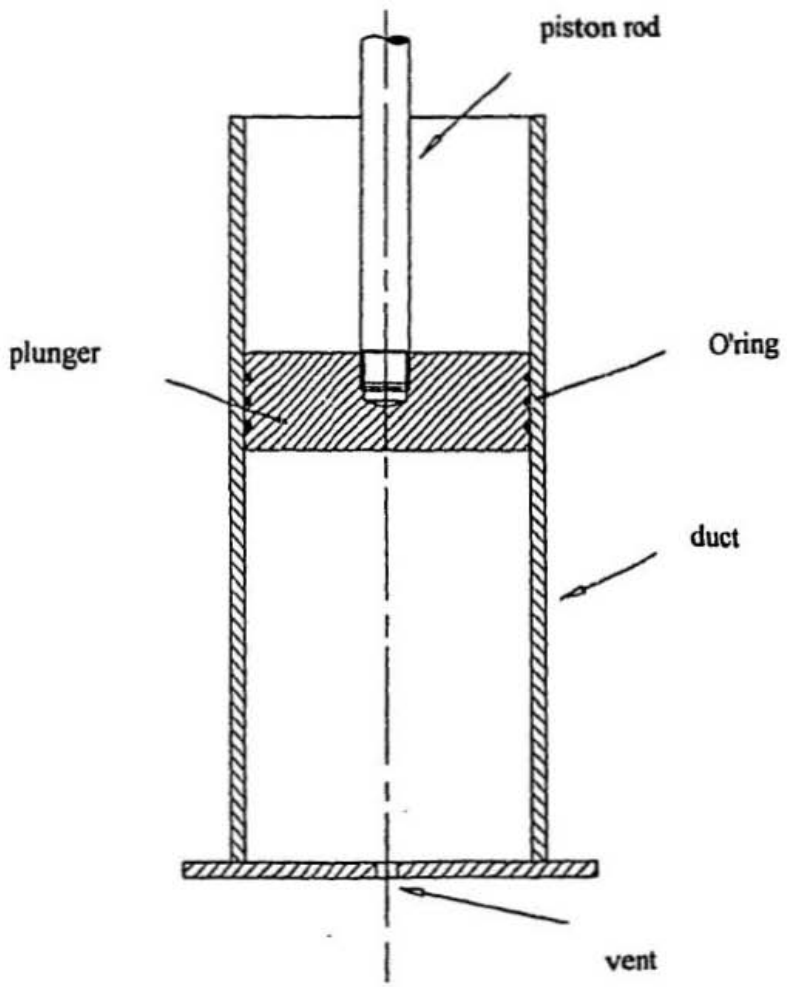

Figure 2.2: Duct and the plunger

2.2 Computer Simulation

Testing an ROV or an underwater robot in the ocean or even in a water tank is difficult and expensive. Simulation is a versatile method that provides for diagnosis and correction of system faults. In order to ensure complete reliability, the control strategies and software need to be tested in computer simulation before the robot is deployed in water. The contents of this section demonstrate the use of simulation as an approach for designing, developing, evaluating and comparing different control strategies and software.

During deployment of the robot in a water tank or seawater, the robot should be in a neutrally buoyant state, before any action takes place. For the buoyancy control tests this was made to be so when the piston rod of the actuator was at its middle position (half way in and half way out). When the piston rod stretches out from its middle position, the plunger moves downwards so the volume of the upper part increases while that of the lower part decreases. Part of the air inside the duct is pressed out to the surface by the plunger and more water comes into the upper part. So the buoyancy force becomes negative and the robot sinks. When the piston rod moves in opposite way, volume of the lower part expands and more air is drawn from the surface and some water in the upper part is expelled out. So, buoyancy force becomes positive, and the robot floats towards the surface. The buoyant force is changed due to the movement of the piston rod. To hover at a certain depth, the piston rod must stay at the middle position again. For the waterjet tests, a trial and error procedure was used to get neutral buoyancy.

Newton's Second Law was assumed to govern the vertical movement of the

robot.

$$\begin{aligned}
\frac{d^2}{dt^2}X &= (F_B + F_J - F_D)/M \\
\frac{d}{dt}X &= V \\
\frac{d}{dt}V &= (F_B + F_J - F_D)/M
\end{aligned}$$

where X is the depth of the robot, V is the velocity, F_B is the buoyancy force, F_J is the jet force and F_D is the drag force, M is the mass. In the setup, F_B and F_J were not active at same time. F_B, F_J and F_D can be expressed by:

$$\begin{aligned}
F_B &= \rho \cdot A_{duct} \cdot L_{rod} \cdot g \\
F_J &= \dot{M} \cdot U \\
F_D &= \frac{1}{2}\rho \cdot C_D \cdot A_{robot} \cdot V^2 \cdot sign(V)
\end{aligned}$$

A_{duct} is the duct inner cross area, here we assume is the same as the plunger area, L_{rod} is the piston rod position, which being measured relative to the neutral piston rod position, A_{robot} is the area of the robot facing the water in the moving direction, ρ is the water density, g is the gravitational acceleration, and $\dot{M}$ and U are the mass flow rate and velocity of the jet respectively.

The simplest numerical integration procedure was used to march the simulation forward in time (Muggeridge and Hinchey 1992). For a time step Δt:

$$X(t + \Delta t) = X(t) + \Delta t \cdot V(t)$$

$$V(t+\Delta t) = V(t) + \Delta t \cdot \frac{d}{dt}V(t)$$
$$\frac{d}{dt}V(t) = \left[F_B(t) + F_J(t) - \frac{1}{2}\rho \cdot C_D \cdot A_{robot} \cdot V \cdot |V|\right]/M \qquad (2.1)$$

In the following part of this chapter, we are going to use several conventional control strategies and compare their effects. From these, we can choose one which gives the best performance.

2.2.1 Proportional Buoyancy Control Scheme

We first use the proportional control scheme to evaluate the robot performance in the water. For a controller with proportional control action, the relationship between the output of the controller $m(t)$ and the actuating error $e(t)$ is

$$m(t) = K_p e(t)$$

or, in Laplace-transformed quantities,

$$\frac{M(s)}{E(s)} = K_p$$

where K_p is termed the proportional sensitivity or the gain. The block diagram of a typical proportional controller is shown in Figure 2.3.

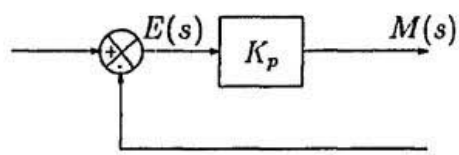

Figure 2.3: Block diagram of a proportional controller

In our case, the controller output is the servovalve signal, and the error signal is the difference between the command position where the robot is supposed

to hover and the robot's actual depth. The block diagram of the whole system with proportional control scheme is shown in Figure 2.4.

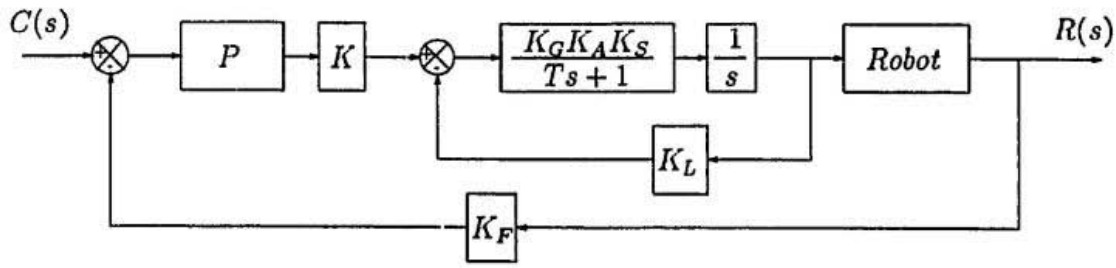

Figure 2.4: Block diagram of the system with proportional control scheme

In Figure 2.4, the inner control loop is the block diagram of the hydraulic system which positions the piston rod. This rod positioning control scheme will be discussed in next chapter. Here, we can simply consider it as a gain. K is the overall gain of the whole system, K_F is the feedback gain.

Before the simulation, we assume the viscous drag coefficient C_D is 0.5 in our program. All the other parameters such as mass M, plunger diameter, piston rod stroke and cross area A_{robot} can be measured from the robot. We assume the robot starts at a certain depth and moves towards another depth, and stops there. We call these two depths the start depth and the end depth (in our program the start depth is $2m$ and end depth is $5m$). We put all these data in a file, together with the proportional gain which we need to change to evaluate the performance. When the program executes, it first reads all the input data from this file, after computation, it writes its output (depth, velocity and time) to another file. From this file, we

can observe the trajectory of how the robot reaches the end depth, and the time it needs to reach this final spot. When it reaches this spot, the robot should be fully motionless. This may result endless computer computation and makes the output file fairly large. In our program, when the difference of the robot's position and final spot is less than $0.05m$ and robot's velocity is less than $0.02m/s$, we assume the robot reaches the final spot and is motionless. The simulation program with proportional control scheme is in Appendix A.

Following are the plots of velocity versus depth and the depth versus time. The proportional gain we used was 1.0. In program, we set time step Δt to 0.001s, which in real time test, is not exactly this value.

Euler's method was used in the simulation program to solve the differential equations. The method does not have good accuracy and stability. To have better accuracy, we choose step size Δt a very small value, 0.001s. Although the accuracy would be lost over a long time period, it will not affect the simulation results, different control schemes could still be compared.

From Figure 2.5 and Figure 2.6, we notice that the robot oscillates at the $5m$ depth, with a reducing amplitude until it stops at the $5m$. The deepest spot it reaches is $5.75m$, $0.75m$ deeper than the final depth, we name this value the overshoot. The robot needs about 11.4s to reach the command spot, we name this time the setup time. From Figure 2.5, we find the maximum speed it reaches is $0.4m/s$. By increasing the proportional gain to different value, the time for the robot to reach the end depth becomes shorter until the proportional gain equals to 20, the number of oscillation and the setup time is minimum, $9.5s$, as shown in Figure 2.7 and Figure 2.8.

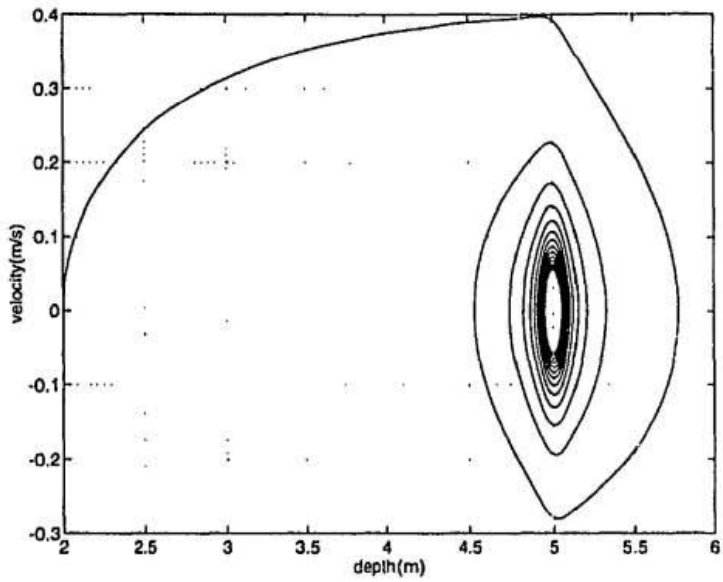

Figure 2.5: Variation of robot velocity when proportional gain is 1

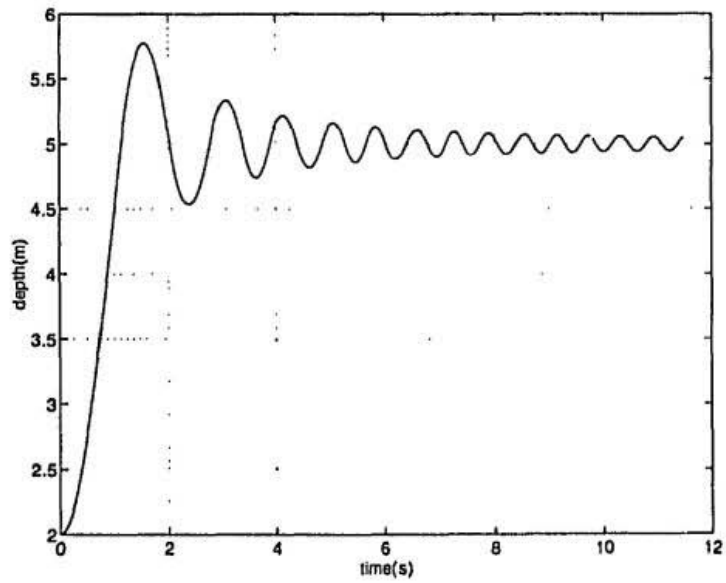

Figure 2.6: Trajectory of robot when proportional gain is 1

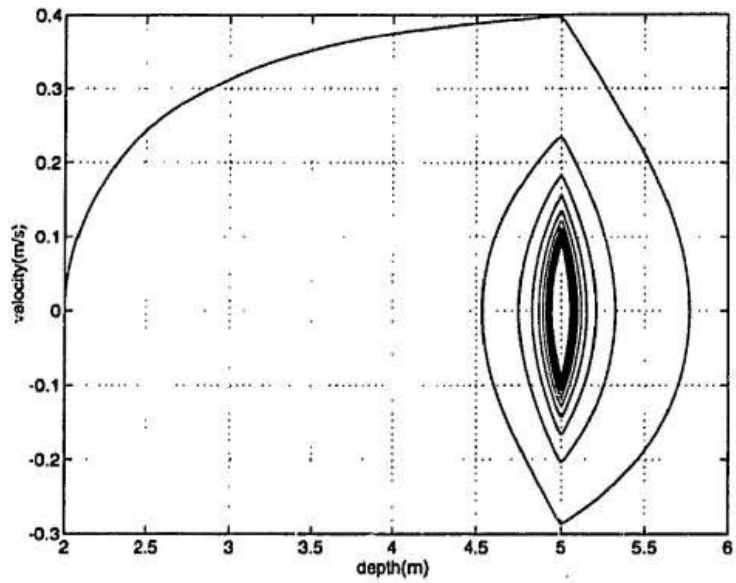

Figure 2.7: Variation of robot velocity when proportional gain is 20

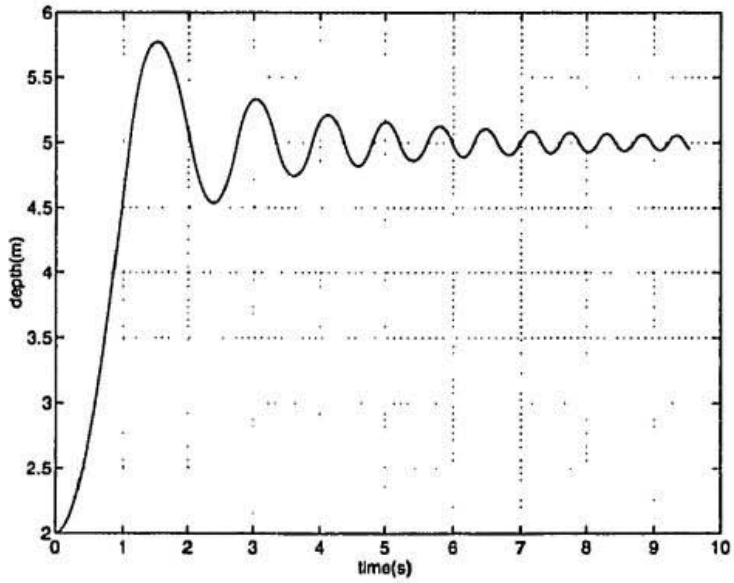

Figure 2.8: Trajectory of robot when proportional gain is 20

We notice from the above plots, that no matter what the proportional gain is, the robot always reaches $5.75m$ and then comes back. So the proportional gain does not affect the overshoot. In order to reduce the oscillation at the end depth, or to reduce the overshoot to minimum, we are going to try another control strategy, the PID control scheme, in the following section.

2.2.2 PID Buoyancy Control Scheme

The analog PID control system has been used successfully in many industrial control systems for over half a century. It is the combination of proportional control action, derivative control action and integral control action. The combined action has the advantages of each of the three individual control actions. The equation of a controller with this combined action is given by

$$m(t) = K_p \left[e(t) + \frac{1}{T_i} \int_0^t e(t)dt + T_d \frac{de(t)}{dt} \right]$$

or the transfer function is

$$\frac{M(s)}{E(s)} = K_p(1 + T_d s + \frac{1}{T_i s})$$

where $e(t)$ is the input to the controller (the actuating error signal), $m(t)$ is the output of the controller (the manipulating signal), K_p is the proportional gain, T_i is the integral time (or reset time), and T_d is the derivative time (or rate time). The block diagram of a PID controller is shown in Figure 2.9.

We use velocity form PID control scheme in our simulation program (Ogata 1987). To derive the velocity form PID control equation, consider the backward

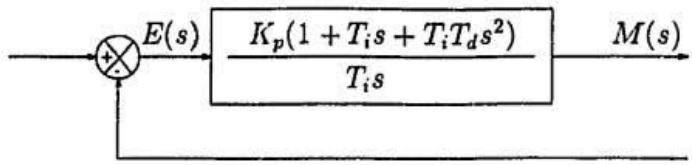

Figure 2.9: Block diagram of PID controller

difference in $m(kT)$, that is, the difference between $m(kT)$ and $m((k-1)T)$:

$$\begin{aligned}
\Delta m(kT) &= m(kT) - m((k-1)T) \\
&= K\{e(kT) - e((k-1)T) + \frac{T}{2T_i}[e(kT) + e((k-1)T)] \\
&\quad + \frac{T_d}{T}[e(kT) - 2e((k-1)T) + e((k-2)T)]\} \\
&= K_P[e(kT) - e((k-1)T)] + K_I e(kT) \\
&\quad + K_D[e(kT) - 2e((k-1)T) + e((k-2)T)]
\end{aligned}$$

where

$$K_P = K - \frac{KT}{2T_i} = K - \frac{K_I}{2} = \text{proportional gain}$$

$$K_I = \frac{KT}{T_i} = \text{integral gain}$$

$$K_D = \frac{KT_d}{T} = \text{derivative gain}$$

The simulation program of PID control scheme is in Appendix A. We add another two parameters in the input data file, the integral gain K_i and derivative gain K_d. First, we randomly choose three values for three gains, 10 for proportional gain, 1 for integral gain and 10 for derivative gain, and observe the robot's performance in the water. This time, it still moves from $2m$ to $5m$. The velocity vs. depth is in Figure 2.11 and depth vs. time plot is in Figure 2.12.

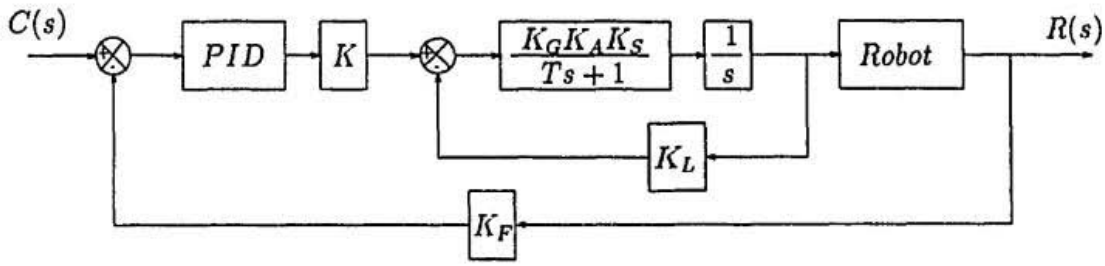

Figure 2.10: Block diagram of the system with PID control scheme

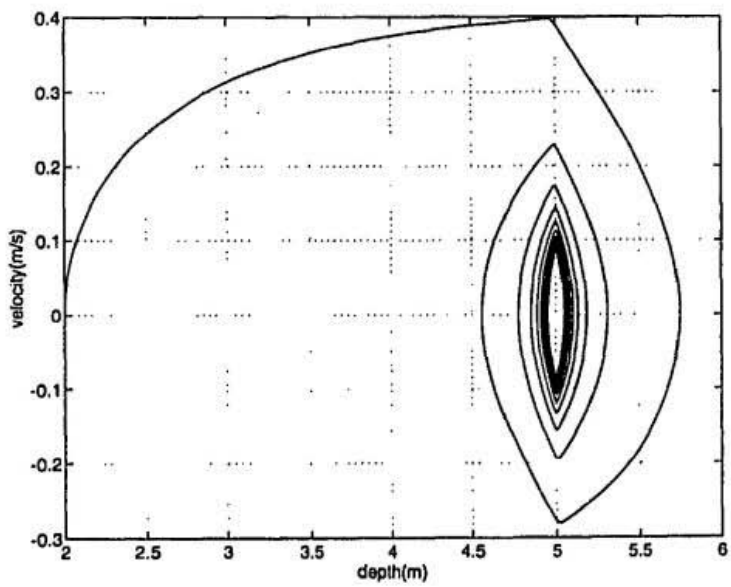

Figure 2.11: Variation of robot velocity when $K_p = 10$, $K_i = 1$, $K_d = 10$

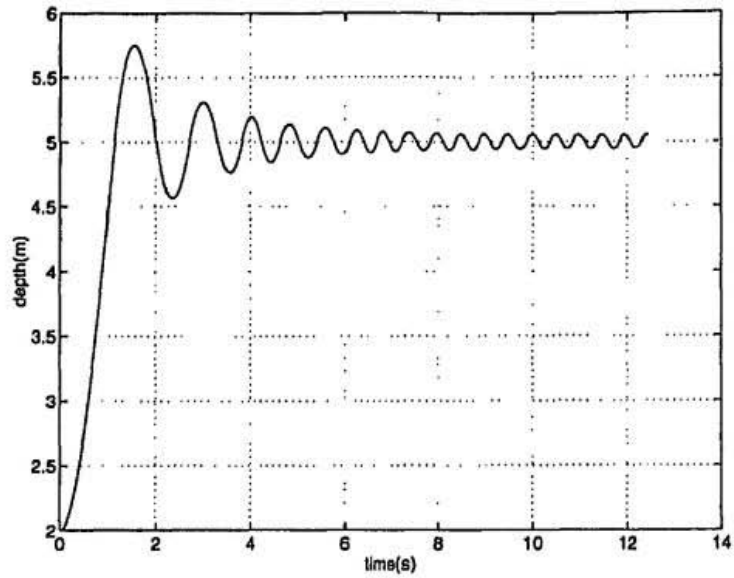

Figure 2.12: Trajectory of robot when $K_p = 10$, $K_i = 1$, $K_d = 10$

From Figure 2.11 and 2.12, we find that the setup time is $12.5s$ and the number of oscillation has not been reduced. Thus we need to adjust the three gains to see if a better performance can be achieved.

We fix the integral gain K_i and derivative gain K_d, and increase K_p step by step and we watch the performance. When it reaches 200, we find that as Figure 2.13 and 2.14 show, the overshoot is zero and setup time the shortest, $1.32s$. So the robot will not need to oscillate around the end depth and the time needed to reach the final spot is much reduced.

We fix the proportional gain K_p and the derivative gain K_d, change the integral gain K_i, what we got was either longer setup time or larger overshoot. We then fix the proportional gain K_p and integral gain K_i, and change the derivative gain K_d, to 1, we found that the K_d does not have much effect in the robot performance. The plots when $K_p = 200$, $K_i = 1$ and $K_d = 1$ are in Figure 2.15

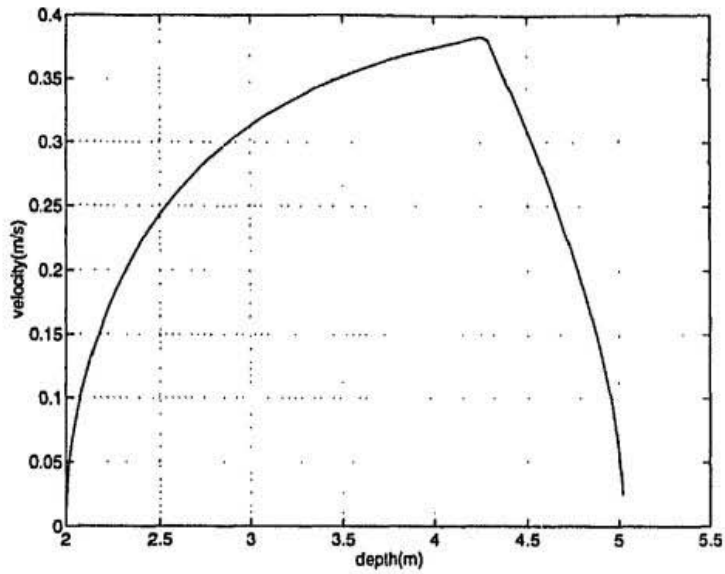

Figure 2.13: Variation of robot velocity when $K_p = 200$, $K_i = 1$, $K_d = 10$

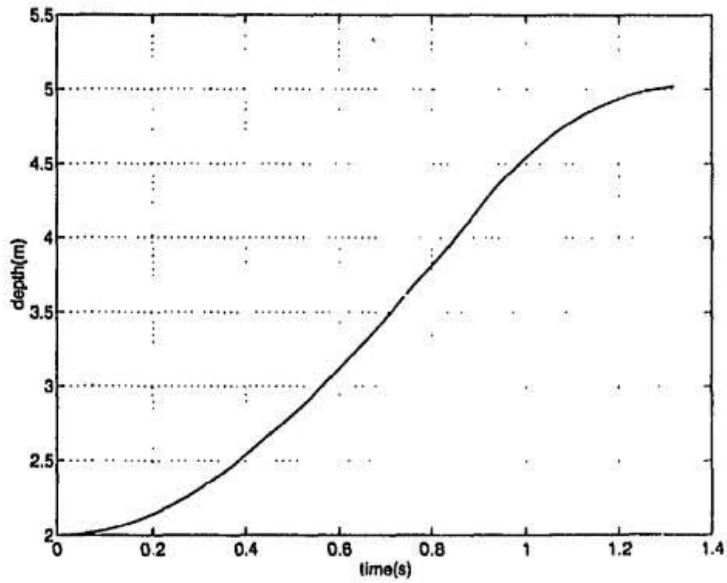

Figure 2.14: Trajectory of robot when $K_p = 200$, $K_i = 1$, $K_d = 10$

and 2.16.

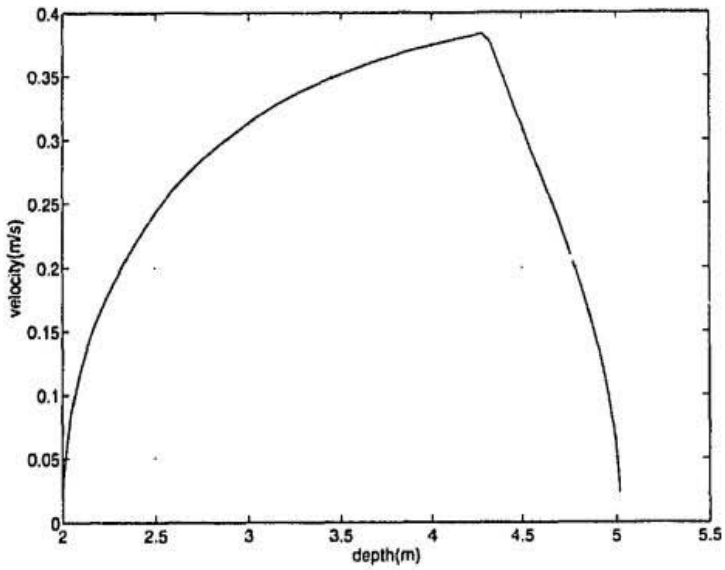

Figure 2.15: Variation of robot velocity when $K_p = 200$, $K_i = 1$, $K_d = 1$

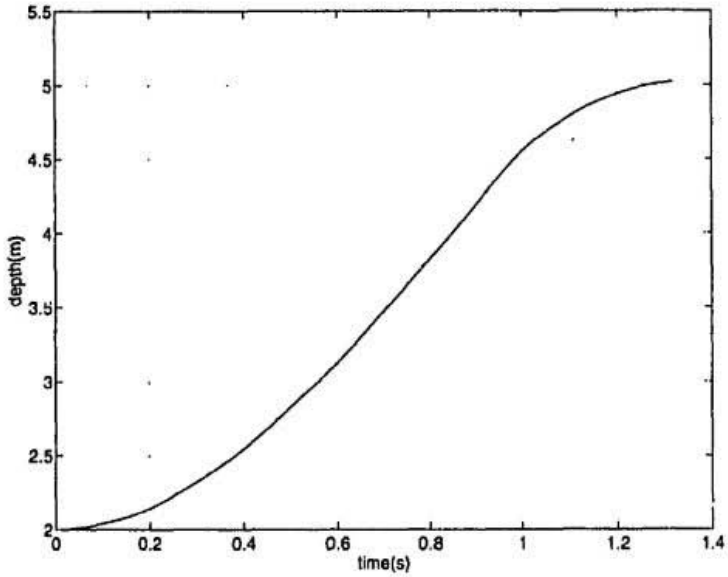

Figure 2.16: Trajectory of robot when $K_p = 200$, $K_i = 1$, $K_d = 1$

From the above observation of performance, we can conclude that the PID

control strategy outperforms the proportional control strategy. The performance in PID control depends on the proper choice of the three gains, K_p, K_i and K_d. Following section will discuss the system's stability.

2.3 Buoyancy Control System Stability Analysis

The simulation in the previous section is just like different kind of tests. From the results of the simulation, we know the control system is stable. Since we want to analyze its stability from its frequency response performance, we use the Nyquist stability criterion.

2.3.1 Nyquist Stability Criterion

For stability analysis of the system, we use Nyquist stability criterion stated below. This criterion is useful in control engineering because the absolute stability of the close-loop system can be determined graphically from open-loop frequency response curves and there is no need for actually determining the close-loop poles.

Nyquist stability criterion states that in the system shown in Figure 2.17, if the open-loop transfer function $G(s)H(s)$ has k poles in the right-half s plane, then for stability the $G(j\omega)H(j\omega)$ locus, as ω varies from $-\infty$ to ∞, must encircle the $-1 + j0$ point k times in the counterclockwise direction.

To analyze the stability of the system, we first plot the system block diagram

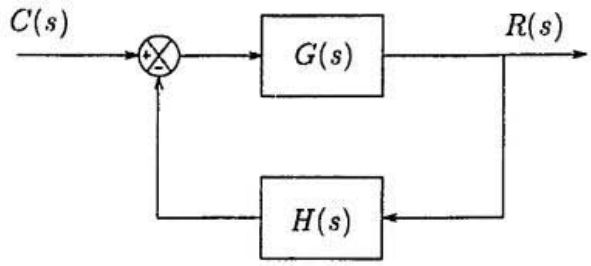

Figure 2.17: Closed-loop system

which is the same as Figure 2.4 on page 21. But here we consider the robot as a damped second order system. Its dynamic equation is

$$m\ddot{R} + b\dot{R} = F$$

where m is robot mass, R is the actual depth here, b is a cycled averaged damping coefficient and F is force. Very rough calculations give $b = 25\frac{N}{m/s}$. After the Laplace transformation, it becomes

$$(ms^2 + bs)R = F$$

The block diagram of the whole system is shown in Figure 2.18. The inner loop is the piston rod positioning system.

In Figure 2.18, K_G is the proportional gain, K_L is the LVDT gain (these two gains are given in *depth.h* file in Appendix B), K_A is the amplifier gain and K_S is the gain of the servovalve, T is the setup time. The product of K_G, K_A and K_S can be expressed as $\tilde{K}$. This subsystem of the rod positioning system can be reduced to its transfer function.

$$\frac{G}{1+GH} = \frac{\tilde{K}}{s(Ts+1)} / (1 + \frac{\tilde{K}K_L}{s(Ts+1)}) = \frac{\tilde{K}}{s(Ts+1) + \tilde{K}K_L}$$

The subblock *BUOY* in Figure 2.18 converts actuator displacement to buoy-

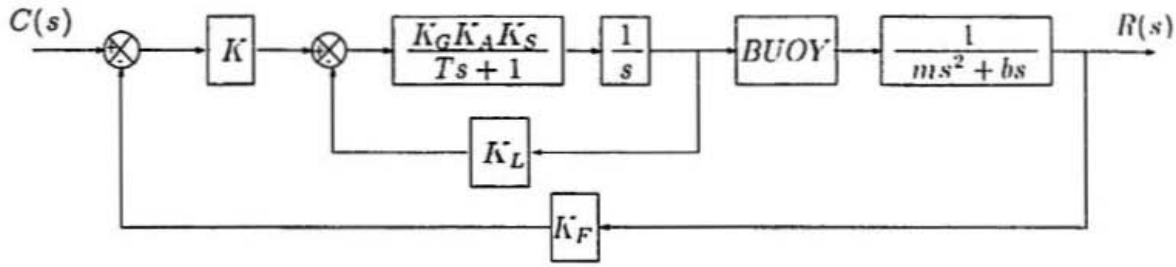

Figure 2.18: Block diagram of the buoyancy system for stability analysis

ancy force. It can be considered as a gain which is the product of the water density ρ, plunger area $A_{plunger}$ and gravitational acceleration g. We can obtain its value from the following equation.

$$BUOY = \rho \cdot A_{plunger} \cdot g = 1000kg/m^3 \times 0.01824m^2 \times 9.81N/kg = 179N/m$$

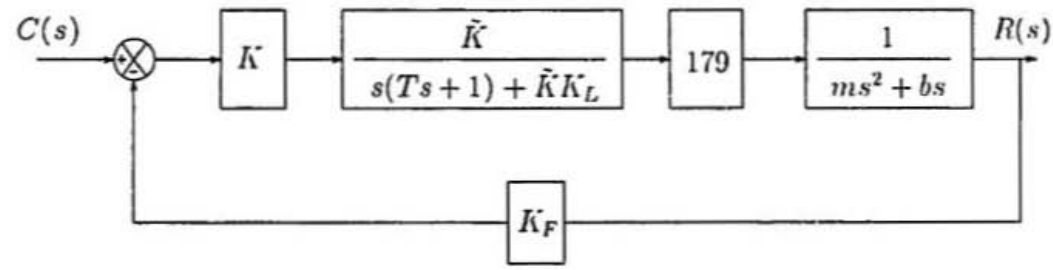

Figure 2.19: Reduced block diagram of the system

The block diagram of the system can be reduced as shown is Figure 2.19. Its open-loop transfer function $G(s)H(s)$ is shown in Equation 2.2.

$$G(s)H(s) = \frac{179 \times K\tilde{K}K_F}{s(ms+b)(Ts^2+s+\tilde{K}K_L)}$$

$$= \frac{179 \times K\tilde{K}K_F}{s(Tms^3 + (m+Tb)s^2 + (m\tilde{K}K_L + b)s + b\tilde{K}K_L)} \qquad (2.2)$$

By replacing s with $j\omega$, we get the following equation

$$G(j\omega)H(j\omega) = \frac{179 \times K\tilde{K}K_F}{j\omega(\left[(m\tilde{K}K_L + b) - Tm\omega^2\right] j\omega + \left[b\tilde{K}K_L - (m+Tb)\omega^2\right])}$$

When $b\tilde{K}K_L - (m+Tb)\omega^2 = 0$, $G(j\omega)H(j\omega)$ locus has real axis cross-over, and it occurs when

$$\omega^2 = \frac{b\tilde{K}K_L}{m+Tb}$$

So $G(j\omega)H(j\omega)$ becomes

$$G(j\omega)H(j\omega) = \frac{179 \times K\tilde{K}K_F}{-\omega^2\left[(m\tilde{K}K_L + b) - Tm \times \frac{b\tilde{K}K_L}{m+Tb}\right]}$$

Since

$$m\tilde{K}K_L + b - Tm \times \frac{b\tilde{K}K_L}{m+Tb} = \frac{m^2\tilde{K}K_L + bm + Tb^2}{m+Tb}$$

is positive, so the cross-over occurs to negative real axis.

We now evaluate the values of the parameters in Equation 2.2. We first assume the overall gain K is 0.1, and by experience, setup time T is 0.2s and b is 25. $\tilde{K}$ is the product of K_G, K_A and K_S, and it is given a value 0.2 on page 66 next chapter. K_F is the gain of the system feedback, which can be obtained from the experimental setup. The value is $0.33m/V$m. m is the mass of the robot. Because of inertia of surrounding water, robot appears heavier than it actually is which we call it added mass. The added mass here is estimated to be one half of the actual mass 158kg, so m is 237kg. K_L is the LVDT gain, which is $1/0.0254 = 39.4$V/m. From the above value, the open-loop transfer function $G(s)H(s)$ becomes

$$G(s)H(s) = \frac{1.1}{s(48s^3 + 242s^2 + 1892s + 197)}$$

The transfer function has four poles, one is at the origin, the other three are at -0.1055, $-2.468 + 5.7275j$ and $-2.468 - 5.7275j$ respectively. Since there are no poles on the right-half of the $G(s)H(s)$ plane, the $G(j\omega)H(j\omega)$ locus should not encircle the $-1 + j0$ point if the system is stable.

By examining the $G(j\omega)H(j\omega)$ locus, we use the Control Toolbox in Matlab, a numerical computation package, to plot the Nyquist curves. Giving different range of frequencies ω, we get four $G(j\omega)H(j\omega)$ loci in Figure 2.20

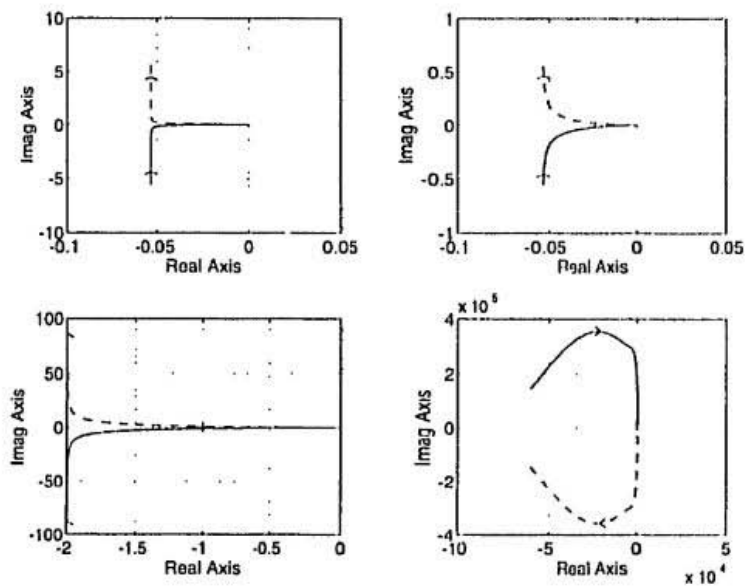

Figure 2.20: $G(j\omega)H(j\omega)$ loci in the GH plane when $G(s)H(s) = 1.1/s(48s^3 + 242s^2 + 1892s + 197)$

From Figure 2.20, we find that the $G(j\omega)H(j\omega)$ locus does not encircle the $-1 + j0$ point. The system is stable for the open-loop function of $G(s)H(s) = 1.1/s(48s^3 + 242s^2 + 1892s + 197)$. Since the overall gain K is estimated, the result may not be accurate. Now we check the stability be giving a different valve of K.

This time we increase the gain from 0.1 to 100, and get another $G(j\omega)H(j\omega)$ locus in Figure 2.21. The $G(j\omega)H(j\omega)$ locus does not encircle the $-1+j0$ point, the system at this time is still stable.

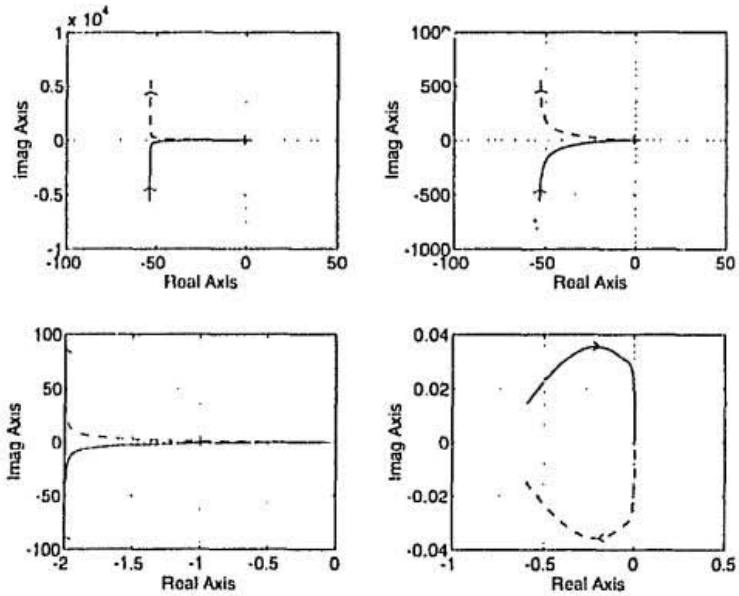

Figure 2.21: $G(j\omega)H(j\omega)$ loci in the GH plane when $G(s)H(s) = 1100/s(48s^3 + 242s^2 + 1892s + 197)$

Now we change the overall gain to 200, and the $G(j\omega)H(j\omega)$ locus is shown in Figure 2.22. From this plot, we find the locus encircles the $-1+j0$ point twice clockwise, so the system becomes unstable at large gain.

In order to make the system to be stable, the gain factor in the open-loop transfer function $G(s)H(s)$ must maintain at low value.

We use Routh's stability criterion to check the result. The characteristic equation is $48s^4 + 242s^3 + 1892s^2 + 197s + 11 = 0$. All the coefficients are positive. The array of coefficients is

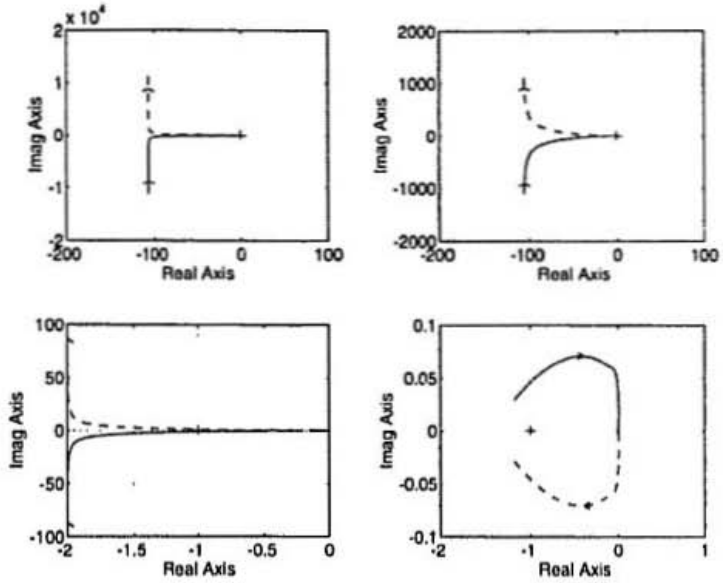

Figure 2.22: $G(j\omega)H(j\omega)$ loci in the GH plane when $G(s)H(s) = 2220/s(48s^3 + 242s^2 + 1892s + 197)$

s^4	48	1892	11
s^3	242	197	
s^2	1853	11	
s^1	240		
s^0	11		

Routh's stability criterion states that the number of roots of the characteristic equation with positive real parts is equal to the number of changes in sign of the coefficients of the first column of the array. Since there is no sign change in the first column in our coefficient array, thus there is no close-loop pole in the left-half s plane. So the system is stable.

2.3.2 Nonlinear Describing Function

We usually use linear differential equations to simplify the systems. Actually, all physical systems are nonlinear and have time-varying parameters in some degree. Therefore, we have to use nonlinear control stability criterion together with the Nyquist criterion.

The important difference between nonlinear control systems and linear control system is that in linear control system, the shape of the time response is independent of the size of the input or initial condition, and stability, or the lack of it, is a property of the system. In a nonlinear system, on the other hand, the nature of the time response, and in fact stability, is usually dependent on the input or initial condition. New frequencies, harmonics and subharmonics, of the input frequencies and limit cycles which are periodic oscillations of fixed frequency and amplitude, are generated by nonlinear components (Ogata 1970).

One way to analyze a particular group of nonlinear control systems, in which the degree of nonlinearity is small, is to use equivalent linearization techniques and to solve the resulting linearized problem. The describing function method is one of the equivalent linearization methods.

Nonlinearities usually refer to saturation, hysteresis, dead zone, backlash and static or coulomb friction, etc. In our case, robot buoyant force is proportional to the piston rod position within the piston rod stroke, 4 inches. The piston rod position is proportional to the signal we send from the computer. Any signal which moves the piston rod beyond its stroke will not be achieved, and will only result in its stroke. So the command signal we send from the computer is saturated.

Thus the system is subject to saturation nonlinearity.

A typical input-output characteristic curve for the saturation nonlinearity is shown in Figure 2.23. For small input signals, the output of a saturation element is proportional to the input. For larger input signals the output will not increase proportionally, and finally for very large input signals the output is constant. Figure 2.24 depicts the input and output waveforms for the saturation nonlinearity.

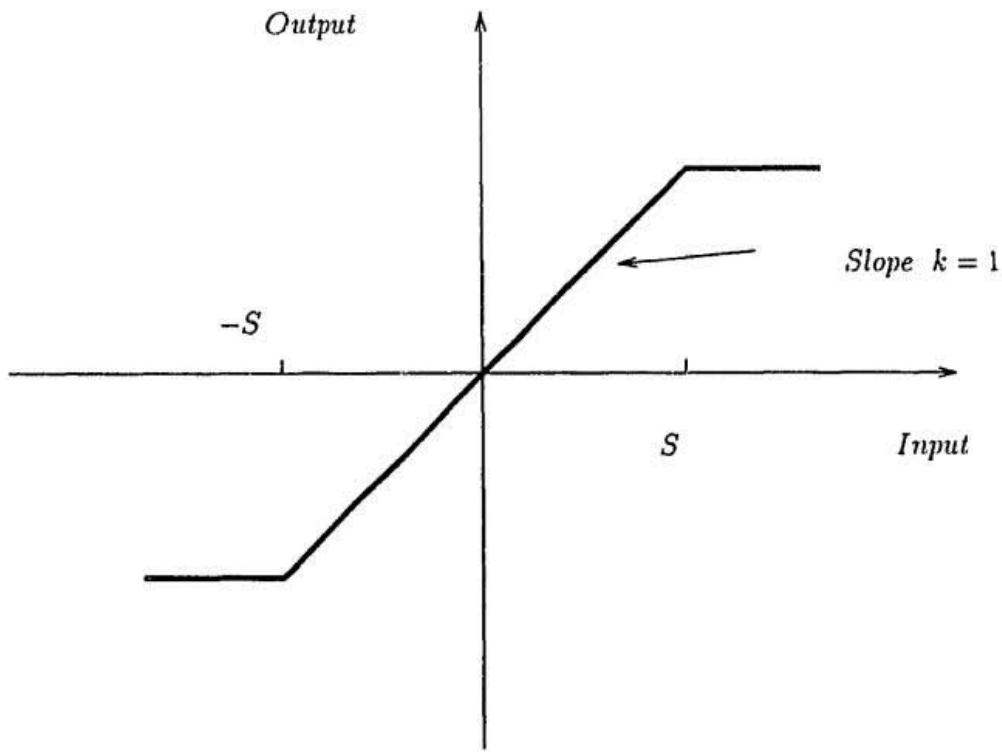

Figure 2.23: Input-output characteristic curve for the saturation nonlinearity

In Figure 2.24, the input to the system is sinusoidal. The output is not sinusoidal. In the describing function analysis, only the fundamental harmonic component is considered important. The reason is that higher harmonics often have

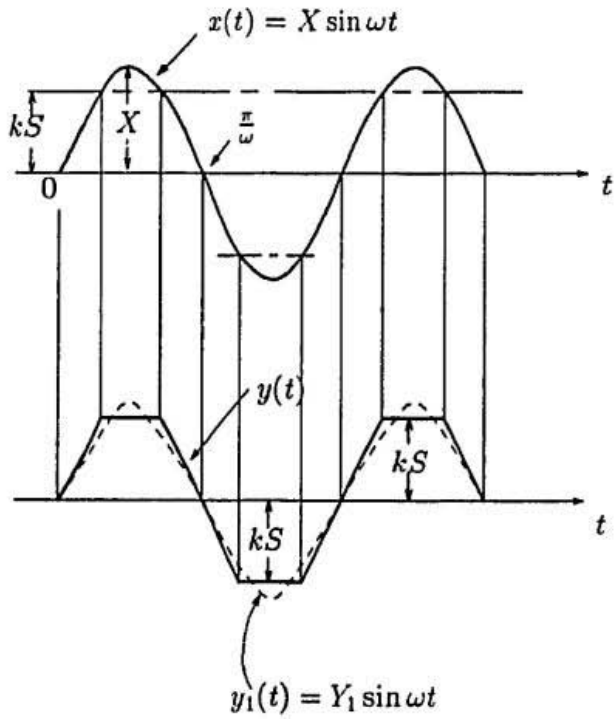

Figure 2.24: Input and output waveforms for the saturation nonlinearity

smaller amplitude than that of the fundamental harmonic component. Also most control systems are low-pass filters, with the result that the higher harmonics are very much attenuated. The describing function or sinusoidal describing function of a nonlinear element is defined to be the complex ratio of the fundamental harmonic component of the output to the input. That is

$$N = \frac{Y_1}{X}\angle\phi_1$$

where

N = describing function

X = amplitude of input sinusoid

Y_1 = amplitude of the fundamental harmonic component of output

ϕ_1 = phase shift of the fundamental harmonic component of output

In calculating the describing function for a given nonlinear element, we need to find the fundamental harmonic component of the output. For the sinusoidal input $x(t) = X\sin\omega t$ to the nonlinear element, the output $y(t)$ may be expressed as a Fourier series as follows:

$$\begin{aligned} y(t) &= A_0 + \sum_{n=1}^{\infty}(A_n\cos n\omega t + B_n\sin n\omega t) \\ &= A_0 + \sum_{n=1}^{\infty} Y_n\sin(n\omega t + \phi_n) \end{aligned}$$

where

$$A_n = \frac{1}{\pi}\int_o^{2\pi} y(t)\cos n\omega t d(\omega t)$$

$$B_n = \frac{1}{\pi}\int_o^{2\pi} y(t)\sin n\omega t d(\omega t)$$

$$Y_n = \sqrt{A_n^2 + B_n^2}$$

$$\phi_n = \tan^{-1}(\frac{A_n}{B_n})$$

If the nonlinearity is symmetric, then $A_0 = 0$. In Figure 2.24, the output is an odd function. For any odd function, we have $A_n = 0(n = 0, 1, 2, \cdots)$. Hence, the fundamental component of $y(t)$ is

$$y_1(t) = B_1 \sin \omega t$$

where

$$B_1 = \frac{1}{\pi}\int_o^{2\pi} y(t)\sin\omega t d(\omega t) = \frac{2}{\pi}\int_o^{\pi} y(t)\sin\omega t d(\omega t)$$

The describing function for an element with saturation can be obtained as

$$N = \frac{B_1}{X}\angle 0^0 = \frac{2k}{\pi}\left[\sin^{-1}(\frac{S}{X}) + \frac{S}{X}\sqrt{1-(\frac{S}{X})^2}\right] \tag{2.3}$$

for $X > S$ and $N = k$ for $X <= S$, where X is the amplitude of the sinusoidal input, k and S can be found in Figure 2.23.

The describing function can be used for stability analysis of nonlinear control system. The system block diagram with saturation nonlinearity is shown in Figure 2.25.

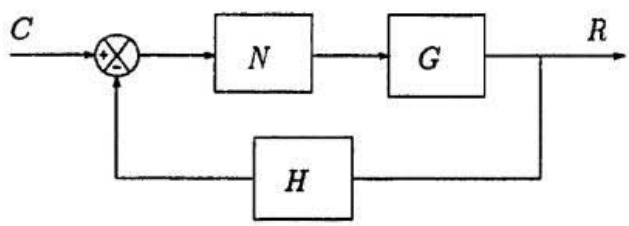

Figure 2.25: System with nonlinearity

where N denotes the describing function of the saturation nonlinearity, GH is the open-loop transfer function. Then the closed-loop frequency response becomes

$$\frac{R(j\omega)}{C(j\omega)} = \frac{NG(j\omega)H(j\omega)}{1 + NG(j\omega)H(j\omega)}$$

The characteristic equation is

$$1 + NG(j\omega)H(j\omega) = 0$$

or

$$G(j\omega)H(j\omega) = -\frac{1}{N}$$

In the describing function analysis, the conventional frequency response analysis is modified so that the entire $-1/N$ locus becomes a locus of critical points. Thus, the relative location of the $-1/N$ locus and $G(j\omega)H(j\omega)$ locus will provide the stability information.

The criterion for stability is that if the $-1/N$ locus is not enclosed by the $G(j\omega)H(j\omega)$ locus, then the system is stable, or there is no limit cycle at steady state. If the $-1/N$ locus in enclosed by the $G(j\omega)H(j\omega)$ locus, then the system is unstable. If the $-1/N$ locus and the $G(j\omega)H(j\omega)$ locus intersect, the system output may exhibit a sustained oscillation, or limit cycle.

To analyze the stability, we first sketch the $-1/N$ locus from Equation 2.3. Since the values of slope k and S are constant, $-1/N$ is function of only the amplitude of sinusoidal input X. When $X <= S$, the $-1/N$ is 1. When $X > S$, and we let X change from S to ∞, we find that N in Equation 2.3 is from k to 0, so the describing function $-1/N$ is from $-1/k$ to $-\infty$. As we can make the slope k in Figure 2.23 to be 1, thus, the $-1/N$ is from -1 to $-\infty$ on Re axis in

$G(j\omega)H(j\omega)$ plane when input X is from 0 to ∞.

We plot $-1/N$ locus as X changes from 0 to ∞ in Figure 2.26. At the same time, we also plot the $G(j\omega)H(j\omega)$ loci we got from Figure 2.21, and use the frequency range of $-\infty$ to ∞.

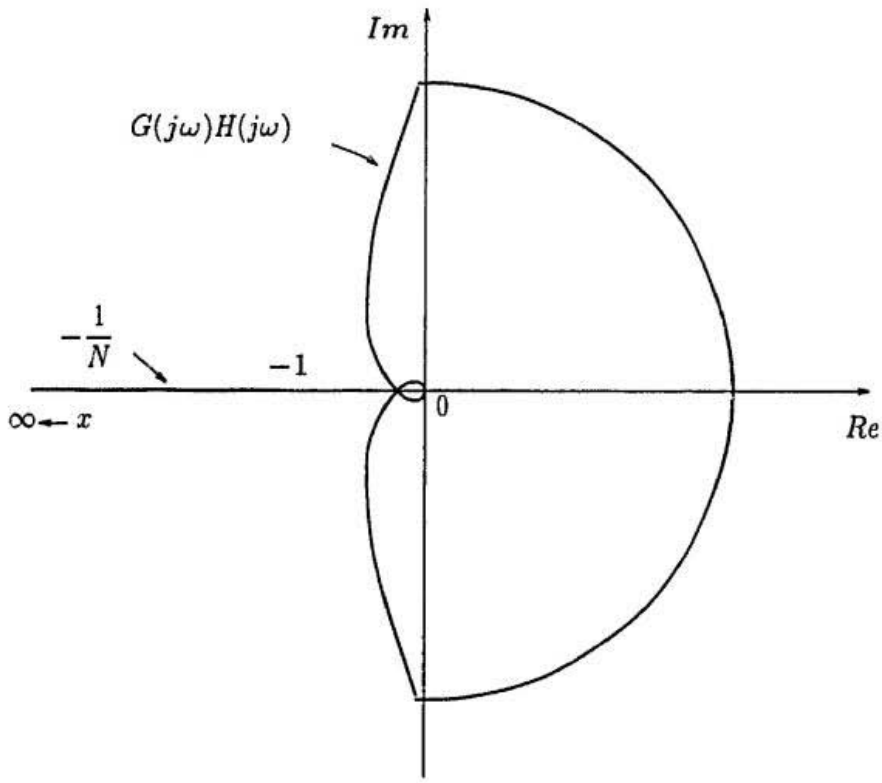

Figure 2.26: Plot of $-1/N$ and $G(j\omega)H(j\omega)$ for stability analysis of buoyancy system

From Figure 2.26, we find that the $-1/N$ locus is to the left of the $G(j\omega)H(j\omega)$ locus, so the $G(j\omega)H(j\omega)$ locus does not enclose the $-1/N$ locus or intersect with the $-1/N$ locus. As for X in range 0 to S, $-1/N = -1$, the Nyquist plot does not include $-1/N$ locus, and thus does not encircle $-1 + j0$ point. So, if linear system is stable, saturation will not cause instability or limit cycle. We can conclude from

Figure 2.26 that the system is stable, and any oscillations which may occur in the system output as a result of disturbances will die out and no sustained oscillation will exist at steady state.

2.4 Waterjet Stability Analysis

For the waterjet case, the block diagram is as shown in Figure 2.27. The J block converts rod motion to force:

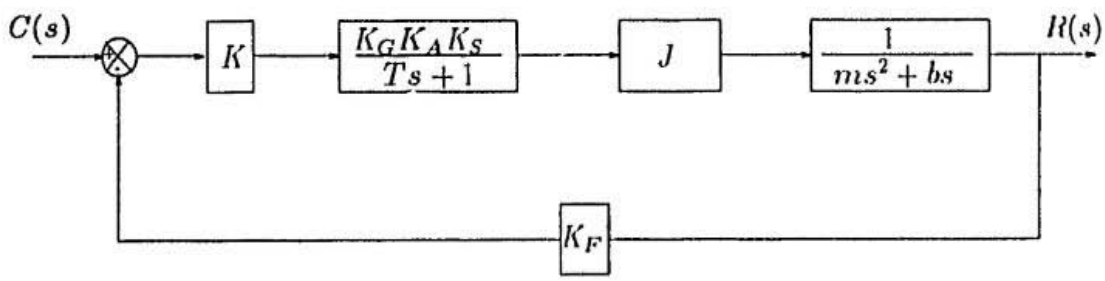

Figure 2.27: Block diagram of jet system for stability analysis

where J is a constant. For linear motion about a command position, $J = 0$. Procedures such as Nyquist show that in this case, the system is borderline stable. Using a nonlinear controller, one could get a describing function N or effective J which in nonzero. By rough computation and experience, we estimate the value of J to be 50. The open-loop transfer function $G(s)H(s)$ can be expressed as:

$$G(s)H(s) = \frac{J \times K\tilde{K}K_F}{s(ms+b)(Ts+1)} = \frac{1.65}{237s^3 + 1210s^2 + 125s} \quad (2.4)$$

The Nyquist plot of the transfer function 2.4 is shown in Figure 2.28. The poles of the open-loop transfer function $G(s)H(s)$ are at the origin, $-0.1055 + j0$ and $-5 + j0$ points, there are no poles in the right half of s plane. Since the Nyquist plot does not include the $-1 + j0$ point, the system is stable.

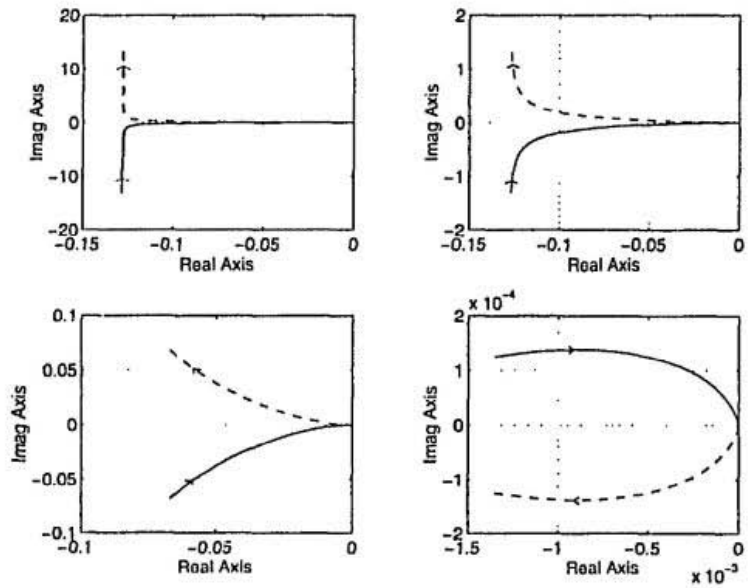

Figure 2.28: $G(j\omega)H(j\omega)$ loci in the GH plane when $G(s)H(s) = 1.65/237s^3 + 1210s^2 + 125s$

When the overall gain K and J are chosen to be very big, the Nyquist plot would look like Figure 2.29. The plot encircles the $-1 + j0$ point twice clockwise, so the system is unstable at very large gain.

Like the buoyancy system, the waterjet system also has saturation nonlin-

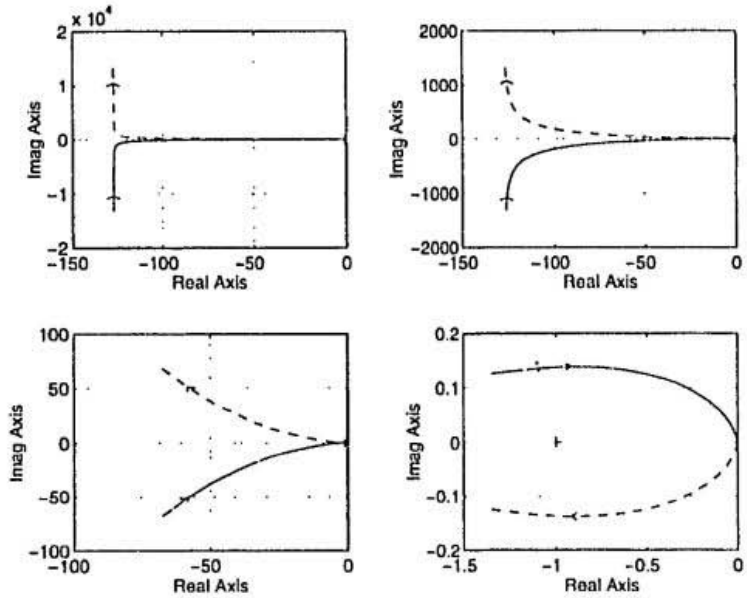

Figure 2.29: $G(j\omega)H(j\omega)$ loci in the GH plane when $G(s)H(s) = 1650/237s^3 + 1210s^2 + 125s$

earity characteristics. The describing function N is still the same as the buoyancy one on Page 42.

$$N = \frac{B_1}{X}\angle 0^0 = \frac{2k}{\pi}\left[\sin^{-1}(\frac{S}{X}) + \frac{S}{X}\sqrt{1-(\frac{S}{X})^2}\right] \tag{2.5}$$

The $-1/N$ locus still changes from -1 point to $-\infty$ as amplitude X changed from 0 to ∞. Since the Nyquist plot does not encircle the $-1+j0$ point, it will not include $-1/N$ locus. If the linear system is stable, then saturation will not cause any instability or limit cycle. The $-1/N$ locus with the Nyquist plot are shown in Figure 2.30.

This chapter has given a brief introduction of the robot configuration and has discussed buoyancy depth control with the proportional control scheme and PID

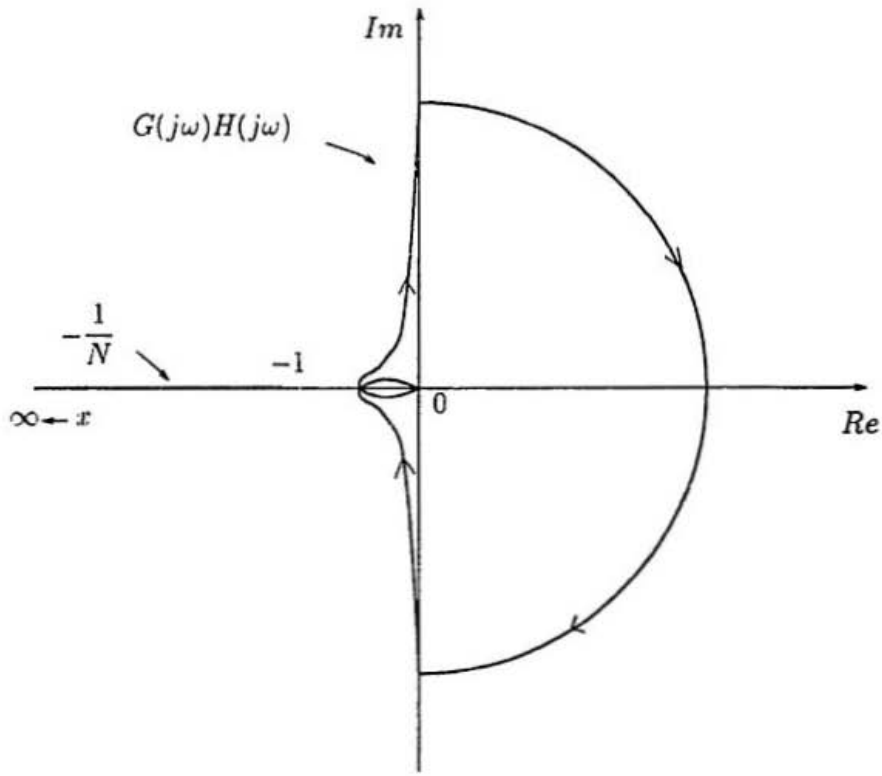

Figure 2.30: Plot of $-1/N$ and $G(j\omega)H(j\omega)$ for stability analysis of waterjet system

control scheme in computer simulation. The results show that the PID scheme outperforms the proportional control scheme in our robot control system. With PID control, the robot reaches the final spot faster and has no oscillation. The system stability for both buoyancy and waterjet have been evaluated with consideration of nonlinearity using describing function with the Nyquist stability criterion. In the next chapter we are going to introduce the procedures and results of the test of the hydraulic system with proportional scheme, and the test in water tank for both buoyancy and waterjet systems. Since proportional scheme is proved later in the test that it worked effectively in the hydraulic system to position the piston rod, we did not apply PID control.

Chapter 3

System Tests

3.1 Components Description

This section is going to describe all the equipment or components needed in the experimental tests.

3.1.1 Transducers

Transducers are important in feedback control systems. They can largely enhance the control hardware and software functions if well designed and on the contrary, they can also deteriorate the control system capabilities.

In our project, the robot was implemented with two transducers. One was used in the hydraulic system control, the LVDT (Linear Variable Differential Trans-

former). It was mounted inside the MTS actuator. Another one was an potentiometer, used to measure the depth of the robot in the water tank test.

LVDT

The LVDT produces an electrical output proportional to the displacement of a separate movable core. Motion of the noncontacting magnetic core varies the mutual inductance of each secondary relative to the primary, which determines the voltage induced from the primary to each secondary. LVDT has the following characteristics (Allocca and Stuart 1984):

- Because there is no physical contact between core and coil, the mechanical component of LVDT do not wear out or deteriorate.
- LVDT has good resolution and small hysteresis.
- It has excellent linearity within its nominal linear range.
- It has good vibration and environment sensitivity.
- It has good response and dynamic characteristics.

LVDT is used here to monitor the displacement of the piston rod of the actuator, and thus the position of the plunger in the duct The output of the LVDT is demodulated by *Daytronic 201C* transducer exciter demodulator, which also supplies regulated excitation to the LVDT. The demodulator produces a filtered DC output signal which is proportional to the mechanical input, over the full plus and minus range of the LVDT, and feeds this regulated signal to the A/D converter

in the data acquisition and control system *Keithley 570*. The excitation supplied by the *Daytronic 201C* is 3 volts at 3 kHz; The output level of *Daytronic 201C* is limited by transducer characteristics and range, typically 10 mV per .001" of core deflection, and the power requirement is 105 – 125 volts, 50 – 400 Hz, 10 watts.

Potentiometer

We used a potentiometer as the depth sensor to measure the depth of the robot in water tank test. The reason we chose the potentiometer as our depth sensor instead of other pressure transducers was it is physically simple and its output is DC voltage which can easily be read by multimeter or *Keithley 570* with computer, therefore, there is no need for extra hardware, such as demodulator or amplifier.

The potentiometer is basically a conductive resistance with a wiper. It is designed to provide an electrical signal (voltage) proportional to angular displacement of its shaft. The output can be read out on a digital voltmeter. In our case, we fed the output to the *Keithley 570* system and read by the computer.

In the water tank test, what we were interested in was the robot's depth which is a linear displacement. To transform the angular displacement to linear one, we put a pulley whose diameter is 105mm on the shaft of the potentiometer and mounted them together on a beam as shown in Figure 3.1.

The beam was used in the test to support the potentiometer and pulley above the water level when the test was conducted in a water tank. The pulley was winded by a wire. One end of the wire was connected to the robot, and the other end was connected to a weight. So when robot descended in the tank, the weight moved

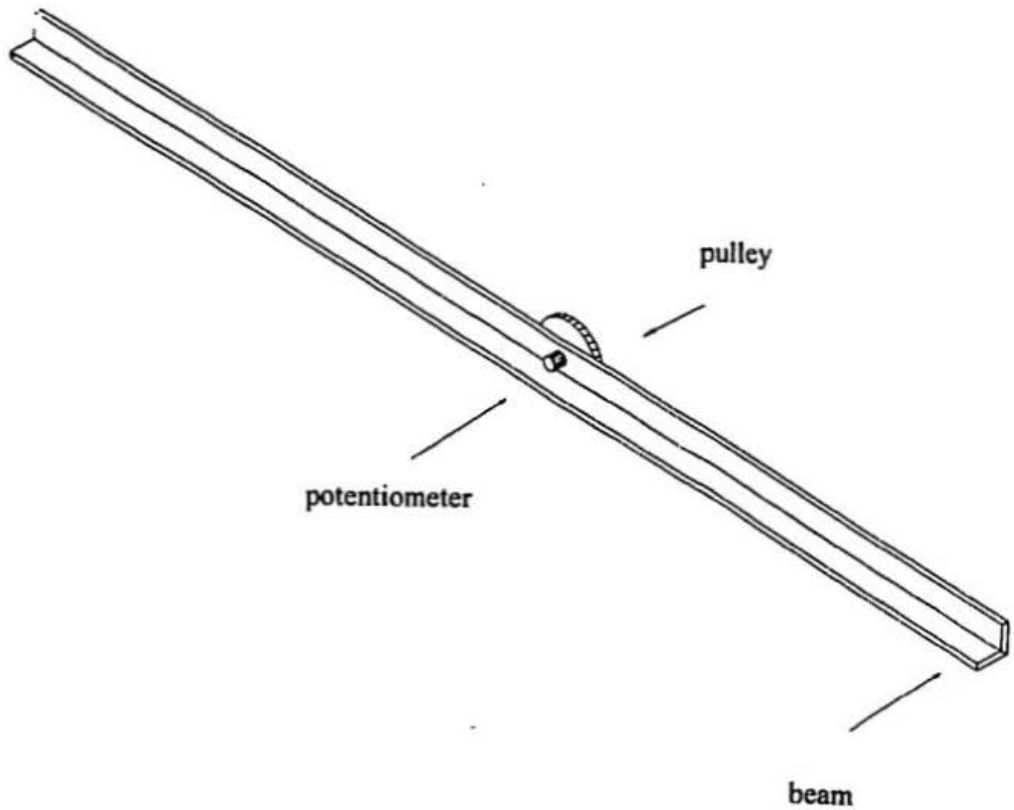

Figure 3.1: Potentiometer in the water tank test

upwards and when the robot ascended, the weight moved downwards. Since we did not want the weight to move in the water, we put another smaller pulley at the end of the beam as shown in Figure 3.2, so the weight moved outside the water tank.

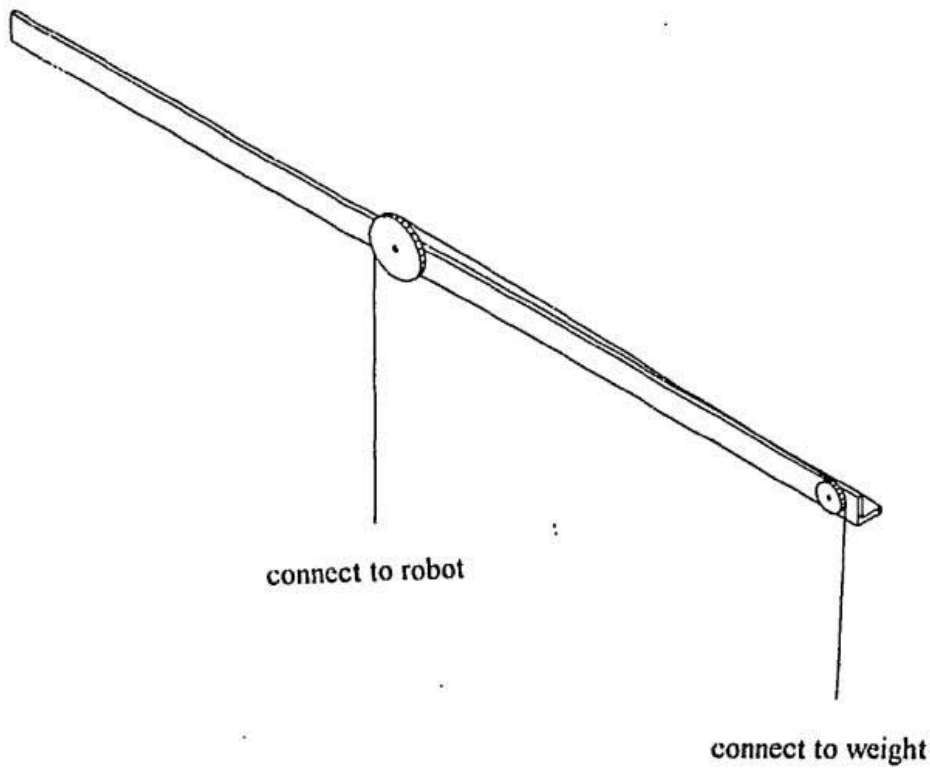

Figure 3.2: Potentiometer and pulleys

When the robot moved in the tank, it pulled the wire so the pulley rotated the shaft of potentiometer, and the signal was fed to the *Keithley 570*. A DC power supply was used to provide power for the potentiometer. The potentiometer shaft could rotate ten times, so the maximum linear displacement we could measure was

ten times the circumference of the pulley, that is 3.3m. That means the maximum depth we can measure is 3.3m. We calibrated the output voltage of the potentiometer by adjusting the power supply output, so that the potentiometer output range was from 0 to 10 volts in its ten round rotation. Any linear displacement of the robot in the water tank corresponded a voltage reading from the potentiometer.

3.1.2 Data Acquisition System

The data acquisition system we used was *Keithley 570* by Keithley Data Acquisition & Control Inc.. It functioned as the interface between computers and other components which we needed to collect signals from or send signals to, such as the servovalve, LVDT and the potentiometer in our case.

The *Keithley 570* requires the host computer to have at least $256K$ bytes of RAM, with 8088 or 80286 CPU. The *Keithley 570* came with another interface card mounted in one of the slots in computer and connects with the *Keithley 570* using a 15 foot shielded interface cable. The *Keithley 570* has inbuilt A/D and D/A converters with 32 single-ended, or 16 differential analog input channels, 2 analog output channels, 16 digital input and 16 digital output channels. The A/D converter input range in *Keithley 570* can be switched to $+/-10$ volts. Because A/D converter has 12 bits resolution, there are 4096 possible voltage levels, specified with digital values $0-4096$. If the input range is $+/-10$ volts, then voltage of each step is 20/4096 or 0.004882, and 0 represents -10 volts, and 4095 represents 10 volts, any voltage between -10 to 10 volts can be computed from this equation, $V = -10 + (D \times 0.004882)$, where D is the digital value from

the A/D converter, and V is the corresponding voltage. *Keithley 570* has inbuilt resistance so that we can also measure currents.

3.1.3 The Hydraulic System

The hydraulic system has four main components, the accumulator, the storage chamber, the actuator and the servovalve, together with other accessories, such as the ball valves and the rubber pipes.

The accumulator is an *MTS* series 111 accumulator. Its length is 1.135m and diameter is 0.128m. It is a piston-type, nitrogen-gas-charged accumulator. It acts as an energy source for short term tests. In this case, it is used to store the high pressure hydraulic oil for the operation of the actuator. The maximum pressure of oil it can store is 3,000 psi. Its size limits the test duration. On top of it is a small valve, it is usually closed except the nitrogen inside leaks and the pressure of it goes below 1,000 psi and we need to charge extra nitrogen into it.

The storage chamber is a hollow cylinder manufactured at MUN with 1.18m in length and 0.125m in diameter. It is used to store the oil which has passed through the servovalve operating the actuator. The small vent on top of it is usually kept closing except when oil needs to be released. When it is open, the pressure inside is the same as the pressure outside so the oil can be easily drained out.

The hydraulic actuator is *MTS* series 242 linear actuator. It operates under precise servovalve control in the control loop. It has a piston rod on one end and a pedestal base on the other. Hydraulic fluid flow is regulated by a servovalve which

is attached on the actuator. Piston rod is moved by differential pressure across the piston faces. The velocity at which the piston rod moves is determined by the flow rate in the servovalve. An internally-mounted LVDT provides a displacement indication of the actuator piston rod. The stroke of the piston rod is 4 inches.

The servovalve is *MTS* series 252. It is one of the key elements in the hydraulic system. The performance of the whole hydraulic system highly depends on the performance of the servovalve. It precisely controls the hydraulic flow to the actuator to achieve the desired loads or displacement. The servovalve itself has a closed loop system electronic servo-controller which provides hydraulic fluid flow control. Within a specific frequency and amplitude range, the flow rate through the servovalve is directly proportional to the electrical input current to the valve. The electrical input is determined by the controller in response to the relationship between the desired excitation and actual excitation.

3.1.4 Buoyancy Control System Design

The reliability of the control system is as crucial as the sensors, the hardware and the mechanical system. It must meet a rigorous requirement. Both hardware and software should be expandable to accommodate more complex sensors. To simplify the setup of the hydraulic rig, we put the controller external from it.

The central point of the robot is the hydraulic system which was controlled by an external IBM PC computer with a 80286 CPU and a 80287 math co-processor. The computer was part of the controller and was used to send the command signal to the servovalve, and collected the feedback signals from the potentiometer and

LVDT through *Keithley 570*. During a mission the computer periodically reads the signal from the potentiometer. If the robot was not at the expected spot, it would send the difference as the command signal to the D/A converter in *Keithley 570*. This difference was the error signal that was acted on the proportional controller in the hydraulic system control. The reason we used the proportional controller instead of the PID controller in the hydraulic system was that the proportional controller gave satisfactory performance in the test. The signal was amplified by the amplifier and deflected the needle in the servovalve, piston rod of actuator was moved, so the buoyancy force was changed. The computer also received signals from LVDT, these signals were demodulated by *Daytronic 201C* demodulator and digitized by A/D converter in *Keithley 570* before they were fed to the computer. As the robot moved towards the spot, the difference between the current depth and final depth tended to zero, so the error decreased. When the robot reached the final spot, the error was zero and the piston rod would be in the middle position, and there was no buoyancy force. This control system contained two loops, the inner loop controls the piston rod movement using proportional scheme, and the outer one controls the depth using PID scheme. The block diagram of the system is Figure 2.10 on page 27.

The control software was comprised of two parts. The first part was a rod positioning algorithm, used initially during deployment to position the rod at the neutrally buoyant position and later to change the volume of the upper and lower parts of the duct for the robot to descend and ascend. After reaching the final spot, the piston rod stayed at the middle position again for the robot to hover. This part of control software used proportional control action with a closed-loop feedback. The second part of the software was to control the depth of the robot

in the water. It combined with the first part and used a PID control action in a close-loop feedback. The operations such as reading the values of LVDT or potentiometer and sending voltage to the servovalve via A/D or D/A converters in *Keithley 570* were written in functions and were called by the main program. The direction of signals flow in the control system is shown in Figure 3.3.

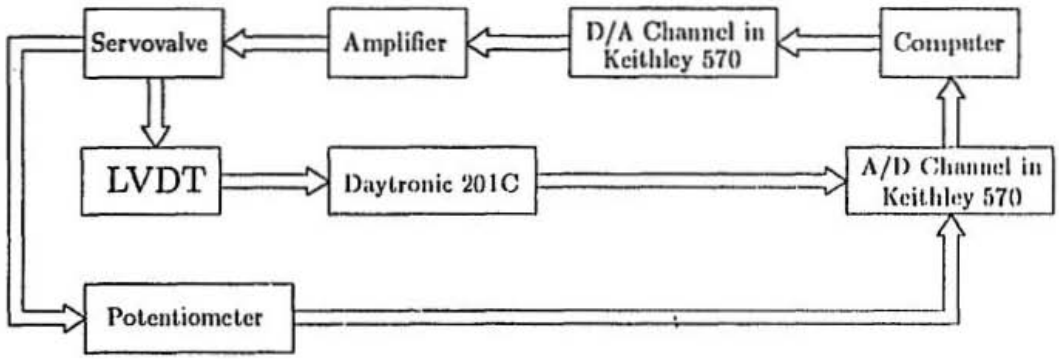

Figure 3.3: Direction of signals in the control system

The software was implemented on the surface computer. It was written in C programming language. We chose C language because it is a high level language which allows a degree of control over the hardware which can only be matched by assembly language. It is more productive than assembly language and supports all primitive data types(floating point, signed and unsigned integers, booleans, BCD and bit fields). It gives extremely rapid response to the external events which is essential in this case in the real time control of the robot. All the control programs together with the functions are available in Appendix B.

3.1.5 Power

There were two umbilical cables which connected the robot and the external controller. One was used to transmit the command signals from the computer to the servovalve and the other was used to return the feedback signals from the LVDT to the computer. Because there was no DC motor in the robot propulsion system, there was no electrical power through the cables for the driving motor.

The power to activate the actuator piston rod was the hydraulic oil in the accumulator. The command signal from the external computer moved the spool inside the servovalve and opened the ports so the oil in the accumulator passed through the servovalve and activated the piston rod, then went to the storage chamber. When the oil inside the accumulator ran out, the robot had to be lifted outside the tank for draining the oil from the storage chamber and recharging the accumulator.

3.2 Buoyancy Control System Test

The control system consisted rod positioning algorithm using proportional control action and robot's depth cont ·l algorithm using PID control action. The first part can be tested in air while the second part can only be tested in the water.

3.2.1 Rod Positioning Test

The rod positioning control is a second-order control system. The block diagram of this system is given in Figure 3.4 the one which we have mentioned in the previous chapter.

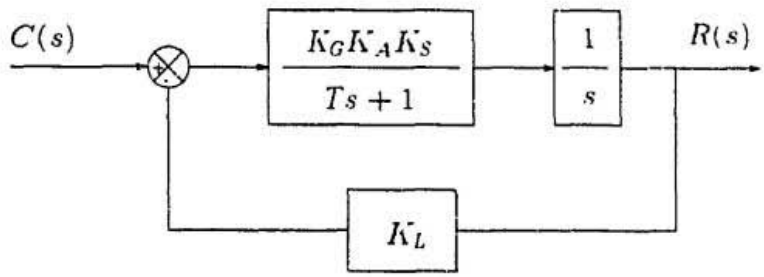

Figure 3.4: Block diagram of the control system

We used proportional control scheme in the rod positioning control. The input command was voltage corresponded to the expected position of the rod, and the output was the current position of the piston rod. The software first checked the piston rod position by reading the voltage from the LVDT, if the current position was not the expected one, their difference was multiplied by a fixed proportional gain, which was set in the software, and sent to the channel 1 of the D/A converter in *Keithley 570* and then amplified by the amplifier, it opened the servovalve and the flow rate in the servovalve was proportional to the voltage we sent from the computer. The velocity of the piston rod and its displacement varied with the flow rates. The displacement of the rod was recorded by the LVDT inside the actuator. The voltage value from the LVDT for a specific displacement was fed back through the 6th channel of D/A converter in *Keithley 570* and demodulated by *Daytronic 201C* and read by the computer. The computer again compared the command signals with the actual signals fed back from the LVDT, the new difference was

multiplied by the proportional gain, then sent back to the servovalve via *Keithley 570*. The operation continued inside this close loop until the piston rod reached the desired position. By moving the piston rod, we could change the air volume in the duct, so the buoyancy of the robot could be changed.

The range of the voltage from LVDT is from -2 to 2 volts, and the stroke of the piston rod is 4 inches. If we assumed the midpoint of the rod is zero point, the two extreme ends were -2 inches and 2 inches. The rod moved within this range. If the rod stayed at 2 inches, the LVDT would feedback a 2V and if the rod was at -2 inches, what we got from the LVDT was -2V. We sent different values of voltage between -2V to 2V from the computer, and measured different piston position and feedback voltage from LVDT and got the following table.

Command Volt(V)	Piston position(Inch)	Feedback Volt from LVDT(V)
2	2	1.98
1.5	1.4	1.49
1	0.9	0.98
0.5	0.4	0.47
0	0	0.02
-0.5	-0.4	-0.49
-1	-0.9	-0.97
-1.5	-1.4	-1.47
-2	-2	-1.98

The plot of the feedback voltage versus the piston position is shown in Figure 3.5.

From the above table and Figure 3.5, we could conclude the feedback gain

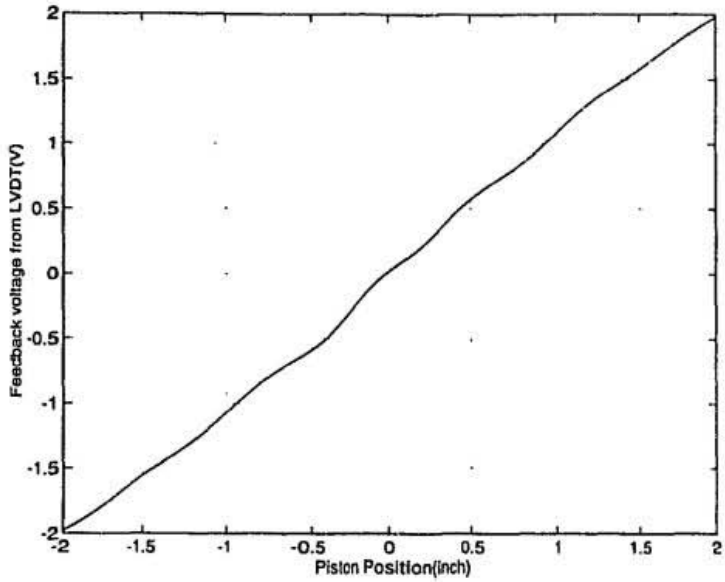

Figure 3.5: Feedback voltage vs. piston position

was 1V/inch, which in dimension of V/m is 39.7. The transfer function $G(s)H(s)$ in the feedforward path is

$$G(s)H(s) = \frac{K_G K_A K_S K_L}{s(Ts+1)}$$

The close-loop transfer function is

$$\frac{R(s)}{C(s)} = \frac{\frac{K_G K_A K_S}{s(Ts+1)}}{1 + \frac{K_G K_A K_S K_L}{s(Ts+1)}}$$

The transfer function becomes

$$\frac{R(s)}{C(s)} = \frac{K_G K_A K_S}{Ts^2 + s + K_G K_A K_S K_L} = \frac{\frac{K_G K_A K_S}{T}}{s^2 + \frac{1}{T}s + \frac{K_G K_A K_S K_L}{T}}$$

In transient-response analysis, the close-loop transfer function $R(s)/C(s)$ can be written as

$$\frac{R(s)}{C(s)} = \frac{\omega_n^2/K_L}{s^2 + 2\xi\omega_n s + \omega_n^2}$$

ω_n is called undamped natural frequency, and ξ is called damping ratio of the system. So in this system, they are

$$\frac{K_G K_A K_S K_L}{T} = \omega_n^2 \quad and \quad \frac{1}{T} = 2\xi\omega_n$$

The maximum overshoot at the peak time M_p and the settling time t_s when 5% criterion is used are given in equation 3.1.

$$M_p = e^{-(\xi/\sqrt{1-\xi^2})\pi} \quad and \quad t_s = \frac{3}{\xi\omega_n} \tag{3.1}$$

We analyzed the system performance in transient response and frequency response using simulation and experiment.

3.2.2 Simulation

We used SIMULINK, a dynamic system simulation software and an extension to MATLAB.

Before the simulation, we have to get the values of K_A, K_S and K_G in the transfer function.

To get the amplifier gain K_A, we sent different voltage to the *Keithley 570* D/A converter, and measured the voltage coming out of the amplifier. At the same time, we measured the maximum voltage we could send out from the computer which would not cause the current in the servovalve exceeded 25mA, the current limit which would not damage the servovalve. We got the following table.

Command Volt(V)	Amplifier Volt(V)	Servovalve Current(mA)
7	4.09	25.8
6	3.51	22.8
5	2.92	18.5
4	2.33	14.8
3	1.74	11.0
2	1.15	7.3
1	0.57	3.6
0	0	0
-1	-0.57	-3.6
-2	-1.15	-7.3
-3	-1.75	-11.1
-4	-2.34	-14.8
-5	-2.92	-18.6
-6	-3.51	-22.2
-7	-4.09	-25.9

The plot of the input voltage vs. amplifier output voltage is shown in Figure 3.6. From the above table and Figure 3.6, we know that the amplifier's output is proportional to the input, and gain K_A is approximately 0.57. Besides that, the voltage we input from the computer could not exceed 7V. Otherwise, the maximum current limit of the servovalve, 25mA, would be exceeded.

The proportional gain K_G is set to 3 in the file *depth.h* in Appendix B. The servovalve gain K_S is hard to achieve because the flow rate in the servovalve drops with the drop of pressure in the accumulator. Since the full-flow rating for 1,000 psi across the servovalve is $0.24m^3/s$ and we assumed the pressure in the accumulator

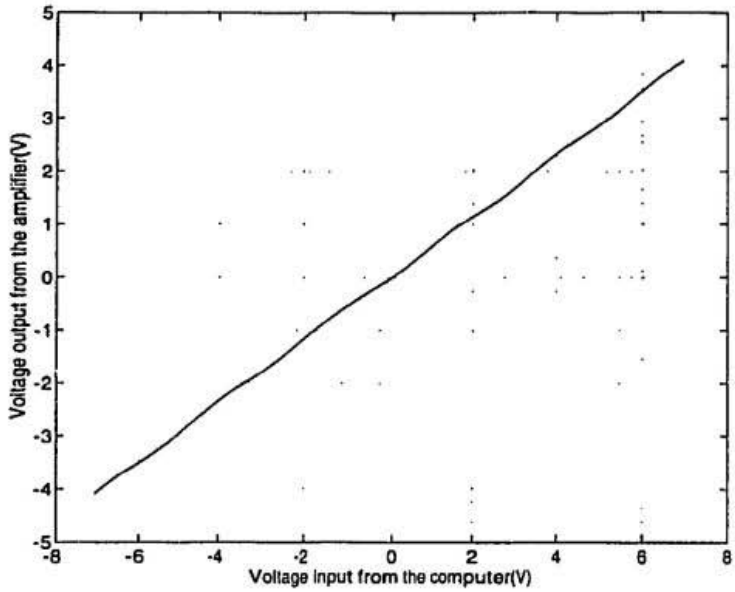

Figure 3.6: Input voltage vs. amplifier output voltage

remained constant at 3,000 psi, so when sending 7V from the computer command, the flow rating is $3 \times 0.24 = 0.72m^3/s$ will be achieved. Therefore, the servovalve gain is roughly estimated as $0.72/7 = 0.1m^3/sV$. The setup time T is estimated as 0.2 by experience. So the feedforward gain $\tilde{K} = K_G \times K_A \times K_S \approx 0.2$. Thus, the block diagram of the system becomes Figure 3.7.

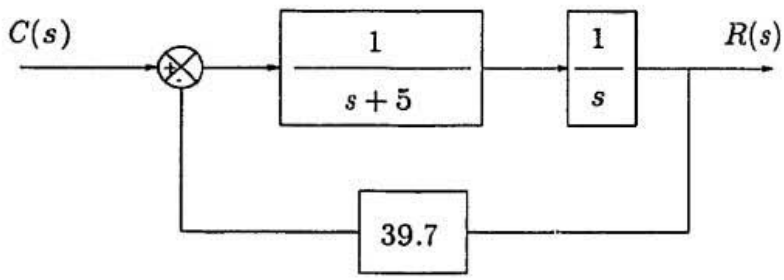

Figure 3.7: Rod positioning control system

To perform the simulation in the SIMULINK, we need to build a model. The block diagram of the model is shown in Figure 3.8. All the symbols in this

figure are from the SIMULINK library. The input is a unit-step function. The time vector t is stored by feeding a *Clock* block into the block *To Workspace1*. The output is stored in the *To Workspace* named y. When the simulation is running, the *Scope* block is a graphic representation of an oscilloscope, it provides a means of observing the signal(s) while a simulation is in progress.

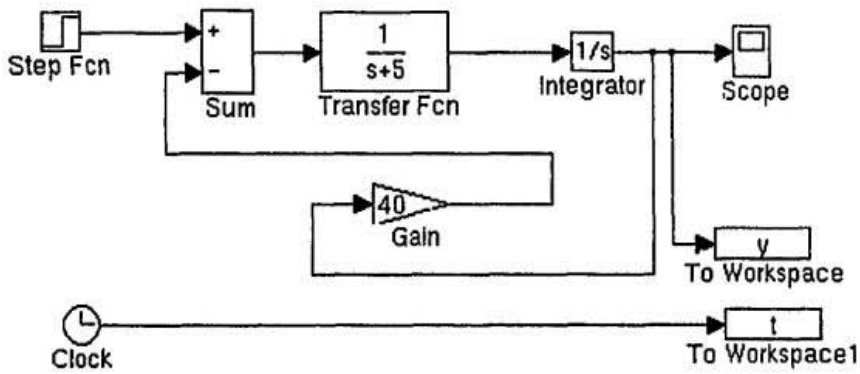

Figure 3.8: Block diagram of the model

When the input is a unit-step function, we plot the output vector y versus time vector t, and got the transient response performance in Figure 3.9. In this figure, we noticed that overshoot is 0.0065, thus $M_p = 26\%$, and setup time t_s is about 3 seconds.

To analyze the frequency response performance, we used a sinusoidal function as the input, and the function in the Control Toolbox in MATLAB to plot the bode diagram which is shown in Figure 3.10. In this figure, the phase margin and gain margin are both positive, thus the system is stable.

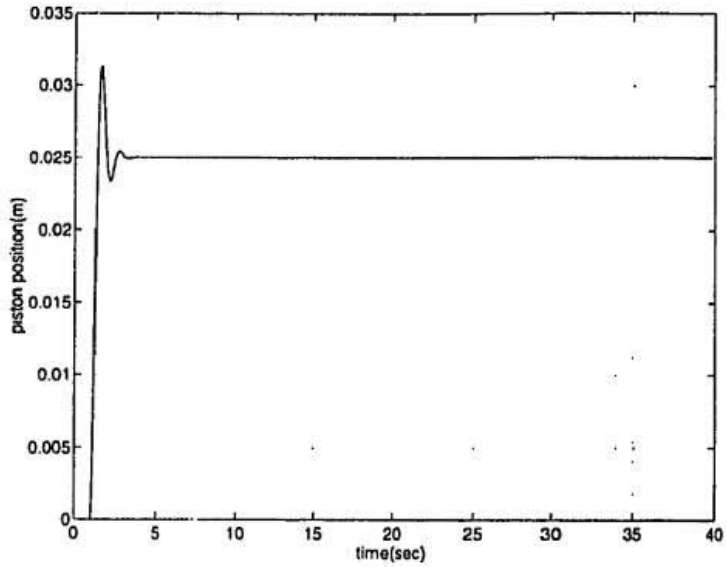

Figure 3.9: Transient time response of the system

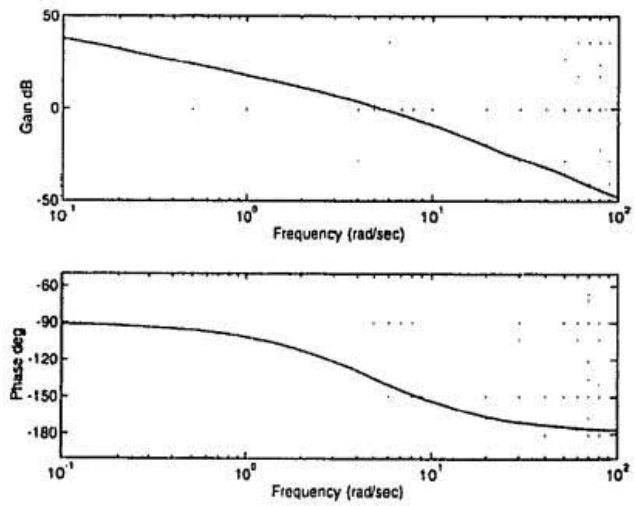

Figure 3.10: Bode diagram of the system

3.2.3 System Performance Test

To analyze the hydraulic system performance through test, and determine the undamped natural frequency ω_n and damping ratio ξ, we executed the piston rod positioning program, but this time, we sent the square wave command to the servovalve from the computer, and sampled the excitation signals from the LVDT, and later subtracted these signals from the square wave commands we sent from the computer and sampled the difference, we called the difference the errors signals.

The reason we used square wave form signals was that they could be considered unit-step input, and the transient response of a unit-step input is always used to specify the performance characteristics of a control system. Another reason was because it is easy to generate and is sufficiently drastic. The wave form of the excitation voltage and the error voltage were displayed and saved by *Tektronix 2212* digital storage & analog oscilloscope. Another PC computer was used to download the data from the oscilloscope and saved them in data files. The square wave amplitudes we sent from the computer were 1 volt and 2 volts respectively. When sending 1 volt, the excitation wave form measured from LVDT is shown in Figure 3.11.

The error signals are the excitation signals subtracted from the command reference(amplitude) values. The wave form we got from the oscilloscope when command amplitude was 1 volt is in Figure 3.12.

From Figure 3.11 and Figure 3.12, we notice that the system exhibited transient response. By looking at the range of 4.2s to 4.8s in the time axis in Figure 3.12, we get Figure 3.13 from which we can clearly analyze the performance

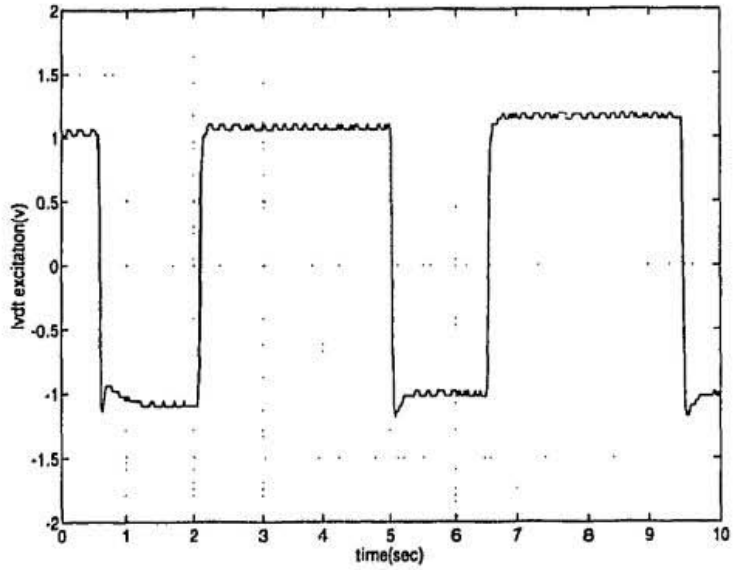

Figure 3.11: Excitation wave from LVDT when the amplitude was 1 volt

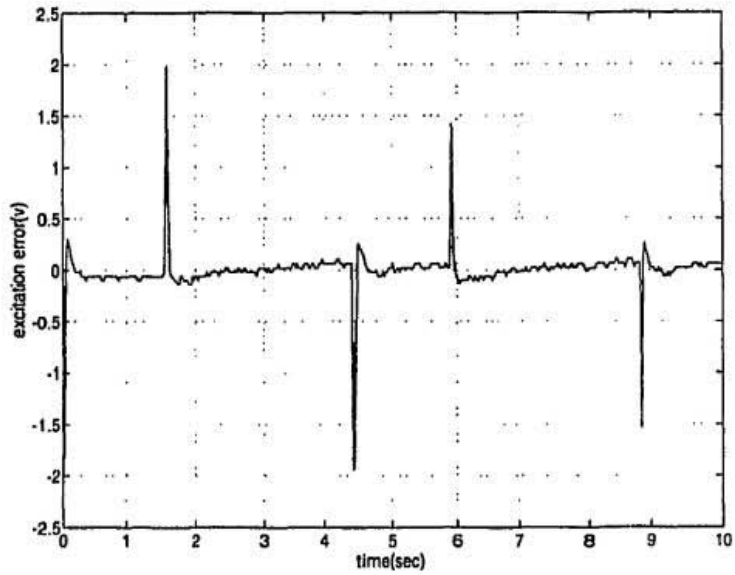

Figure 3.12: Error signal when amplitude was 1 volt

characteristics, that means the maximum overshoot M_p and settling time t_s can be evaluated from this plot.

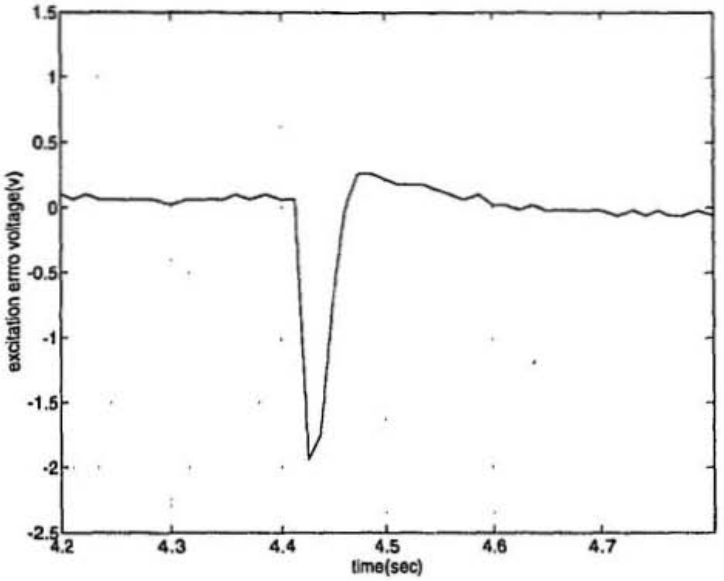

Figure 3.13: Amplified error signal of 1 volt

Since the response curve is not measured from unity in Figure 3.13, the maximum overshoot M_p is more clearly termed as the maximum per cent overshoot, which is defined as $\frac{c(t_p) - c(\infty)}{c(\infty)} \times 100\%$, where t_p is the peak time. In our case, the maximum per cent overshoot $M_p = \frac{2.25 - 2}{2} \times 100\% = 12.5\%$, where 2.25 is the peak height of the signal measured along the y axis in Figure 3.13, and 2 is the height of the signal in steady state. The settling time t_s is about 0.2s. From equation 3.1, we have

$$\begin{aligned} 0.125 &= e^{-(\xi/\sqrt{1-\xi^2})\pi} \\ 0.2 &= \frac{3}{\xi\omega_n} \end{aligned} \tag{3.2}$$

From equation 3.2, we get $\xi = 0.552$ and $\omega_n = 27.2$. From the value of ξ, we can tell that the system is underdamped which complies with the wave form we got from the test.

When the input square wave amplitude from the computer was 2 volts, the excitation signal from the oscilloscope is shown in Figure 3.14, the error signal we measured from the *Keithley 570* is shown in Figure 3.15. When we choose the time range from 5.6s to 6.3s, we get Figure 3.16.

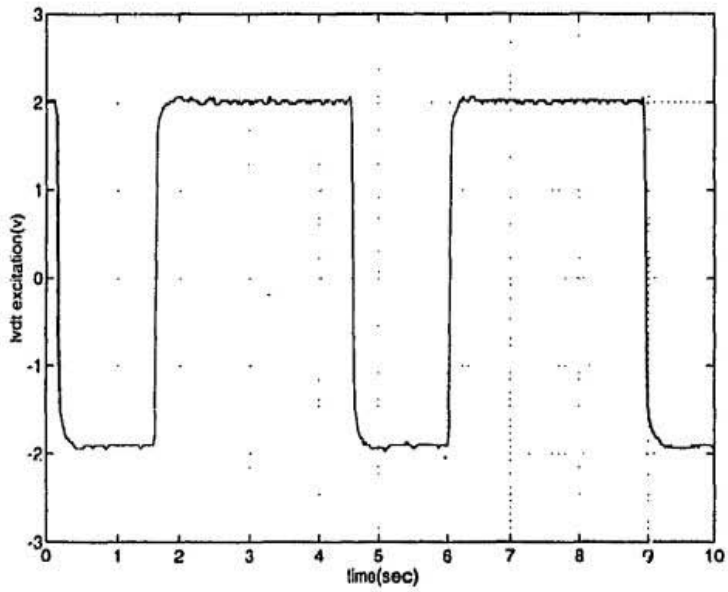

Figure 3.14: Excitation wave when the amplitude was 2 volts

From Figure 3.14 to Figure 3.16, we notice that the system did not have the maximum per cent overshoot and the settling time t_s, which is $0.17s$, is shorter than the settling time t_s, which is $0.2s$, when the amplitude voltage is 1V. Thus, the rod positioning system exhibits better transient response performance when the input has larger step than that of smaller step.

The reason we did the test of the hydraulic system was to check its stability

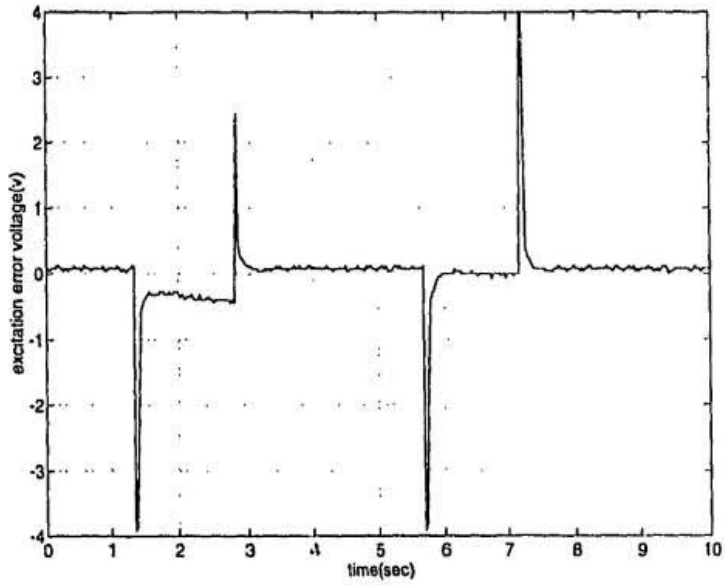

Figure 3.15: Error signal when amplitude was 2 volts

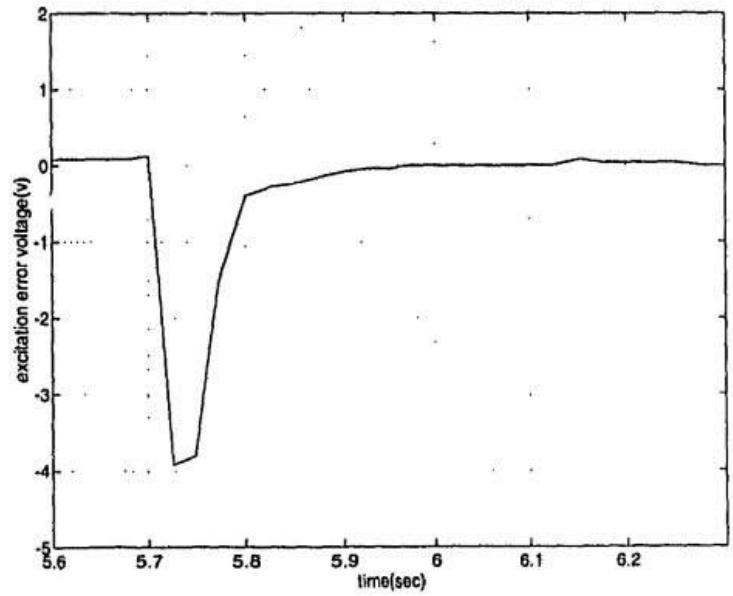

Figure 3.16: Amplified error signal when the amplitude was 2 volts

and transient response performance. The result from the test showed that the hydraulic system was quite stable and had good transient response performance.

3.3 Test in the Water Tank

3.3.1 Preparations

Before deploying the robot into the water tank to test, several preparations had to be done.

There are two umbilical cables that transmit signals between the surface computer and the robot. One is connected to the servovalve by a 9 pin connection and the other is connected to the hydraulic actuator for the LVDT feedback by a 5 pin connection. Both connections are not waterproof. To prevent leakage of electricity from the connections and oil from the servovalve as well into the water, we sealed the two connections and the whole servovalve with seal gasket wax. The reason we chose this kind of wax instead of other sealent like *Silicon* is, that it will not dry out, harden or crack, and most important the components it seals can be reused after being cleaned. These haracteristics of the wax make it the suitable sealent in underwater operation.

When we tested the piston rod positioning program, we did not connect the rod to the plunger and put it into the duct. Before the robot was put into water, these three things had to be assembled together. To do the assemblage, we put the plunger into the duct by hand, since the plunger had tight contact with the

duct, it only stayed on the top of the duct if we did not push it too hard. Then we screwed the plunger on the piston rod. The duct was also rotated when screwing. Later, we screwed the duct on the aluminum plate with eight small bolts. After screwing, we connected the plate to the robot supporting frame with four bolts, but we left the four bolts loose until we executed the rod positioning program to push the plunger into the duct. If we tightened the four bolts before the plunger was pushed, the force generated by the duct would not be applied on the plunger evenly, this would cause air or water leakage between the two parts of the duct.

The robot is 160kg in weight, which gave us difficulty in transporting and deploying it. We used a bridge crane to lift the robot and put it into the tank.

The robot had to be neutrally buoyant in the water tank before any operation could take place. To get it in this state we first added styrofoam to make it approximately neutrally buoyant and we moved the piston rod to the middle position. To see if it was neutrally buoyant, we put it into the water tank. If the robot sank, we lifted it out of the tank and added some more styrofoam to increase the buoyancy, and then put it back into the tank. If it floated at the water surface, we took some styrofoam off it. Such operations were repeated several times until the robot was in a slightly positive state of buoyancy. Then we put it into the tank and added several weights to make it precisely neutral. During the trial, we also found the steel sling added some extra force on the robot. To prevent this unexpected force, we used rope sling instead of steel sling. The rope sling was neutrally buoyant.

3.3.2 Buoyancy Control Test

A couple of wave tank facilities are available, one in the Faculty of Engineering and Applied Science of MUN and the other in the Institute for Marine Dynamics (IMD) of National Research Council (NRC). Since we did not want oil to spill in the wave tanks, we chose another water tank for our test. The water tank was a cylindrical tank with approximately 1.5m in diameter and 2.4m in depth which is shown in Figure 3.17. With this tank, we did not need to worry about the oil spill and also it is easier to fill and drain the water than the wave tank. The water was filled up to 2.1m in the tank. For the depth measurement, a beam with the pulleys was put across the water tank after the robot was immersed into the water.

We tested the robot working with buoyancy without using depth feedback. The purpose of this test was just to see if the movement of the plunger would cause the buoyancy of the robot to change. A potentiometer was used to record the depth of the robot in the tank.

We used a DC power supply to provide power to the potentiometer. Its positive and negative leads were connected to the two leads of the potentiometer by long wires. The other two leads of the potentiometer were connected to channel 5 of the A/D converter in *Keithley 570* for computer to read the voltage value. We adjusted the voltage range of the power supply from 0V to 10V, so the voltage from the potentiometer was from 0V to 10V, it could be read in single-ended mode by *Keithley 570*. Since *Keithley 570* A/D converter had already been set in differential mode for reading feedback signals from LVDT, so we used differential mode for the potentiometer.

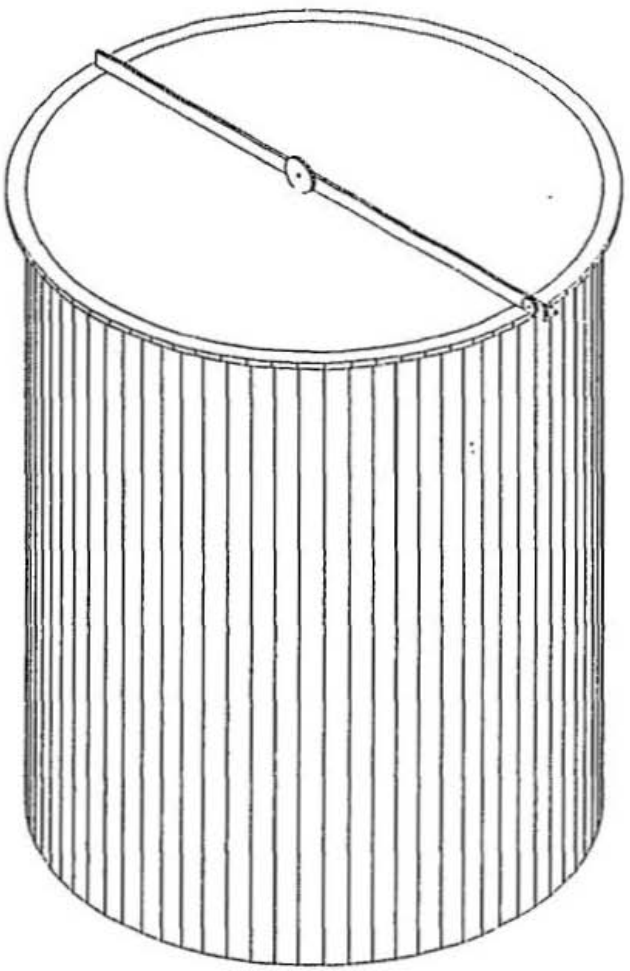

Figure 3.17: Water tank in the test

We operated the crane to lift the robot and put it into the tank. Before it reached the water surface, we opened the ball valve in the hydraulic system, so the oil could move from the accumulator to the storage chamber. When robot was completely immersed into the water, we put the beam with the string on the pulley across the tank for recording the vertical movement of the robot. Both ends of the string were tied on weights. One weight was put on the top of the robot and the other was hung outside the water tank. The robot first stayed in the middle of the water, when the position rod moved upwards, the buoyancy increased and the robot floated towards the surface. We then stretched the piston rod, the buoyancy decreased and the robot sank to the bottom. We noticed from this test and from the previous rod positioning tests that because the oil in the hydraulic system can not be circulated, the duration of the test was dependent on the amount of oil stored in the accumulator. Thus when the oil in the accumulator ran out, we had to lift the robot out of the water and refill the oil to the accumulator. The whole test took about 4 or 5 minutes. The change of the depth was recorded by the computer and was stored in a data file. From this file, we plotted the trajectory of the robot moving in the tank as shown in Figure 3.18. In this plot, we noticed that the robot oscillated at a certain depth. The oscillation might be caused by the instability voltage jump of the potentiometer or by the slip out of the string from the pulley during the test. The test had to be operated in a very favorable condition, meaning the robot had to be moved exactly beneath the pulley, otherwise, the string was easily to slip out from the pulley, and the result from the test was not precise. To enable retrieval at the end of a deployment, the rod of the hydraulic actuator was moved to the position of maximum buoyancy. But usually we used the crane to lift the robot out, because the oil in the accumulator easily ran out.

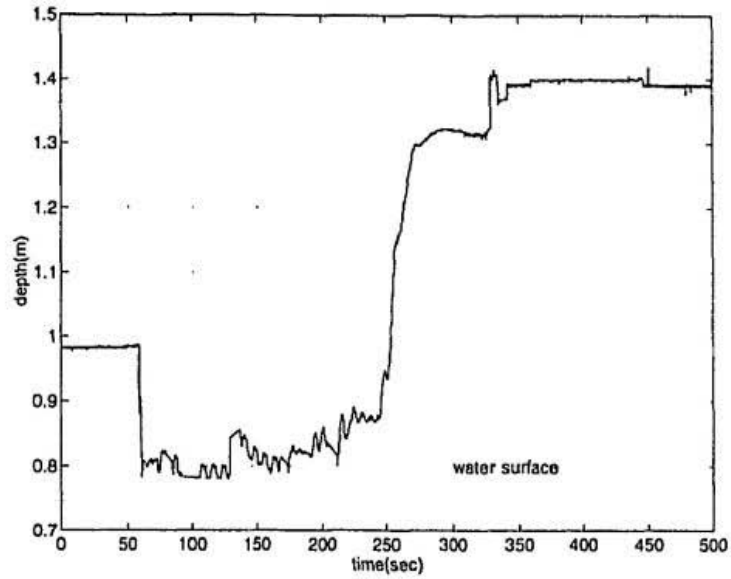

Figure 3.18: Robot trajectory in the buoyancy test

The test of the robot in buoyancy mode with the depth feedback PID close-loop control scheme should be our next test. Unfortunately, because of problems with the data acquisition system and potentiometer, this has been left for future work. Here, I will outline how the close-loop system is supposed to work.

The block diagram of the close-loop control has already been given in Section 2.2.2 on page 27 which is shown again here in Figure 3.19.

The control program for this test is *depth.c* in Appendix B. The command syntax is *depth 3* (if you want the robot to be hover at 3 meters depth), where *depth* is the executable file name and 3 is its argument, here is the reference depth.

The program first checks the robot current depth by reading the voltage from channel 5 of D/A converter in *Keithley 570*, this is the voltage fed back from the potentiometer. The voltage is multiplied by a gain to convert to the depth. If it is

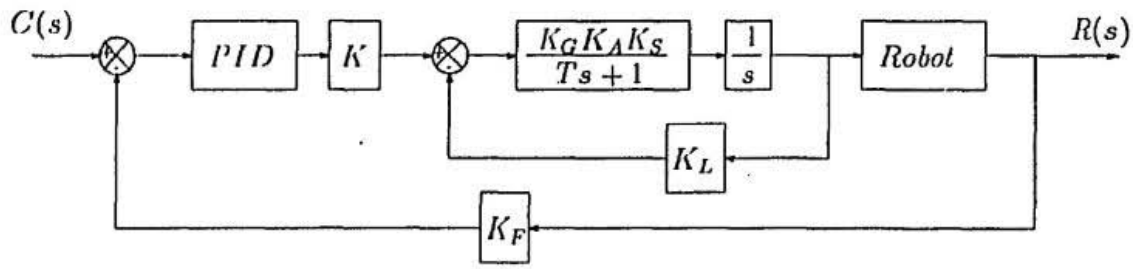

Figure 3.19: Block diagram of the system with PID control scheme

not the referenced depth given in the command line, the difference is sent to the PID controller as the error. From PID scheme, the program computes the piston rod position as the function of this error. The program then checks the piston rod position, by reading channel 6 of D/A converter in *Keithley 570*. If this position is not the one the program computes from the PID scheme, it then executes the piston rod positioning program by sending the rod position error multiplied by the proportional gain through channel 1 of A/D converter in *Keithley 570*. The piston rod moves, so does the robot towards the reference depth. The computation and execution inside the loop continues until the robot reaches the expected depth.

Since the voltage supplied to the potentiometer was from 0V to 10V, and 10 times of the circumference of the pulley is 3.33m, the gain for reading the depth in the program is 0.333m/V. So, if the reading from the potentiometer is 2V, then the depth is at 0.66m.

3.3.3 Waterjet Test

For a preliminary look at the waterjet device, we took the tygon tubing off the bottom of the duct and enlarged the vent on the plate to a 1.5 inches in diameter hole. Another small vent was drilled near the upper edge of the duct but below the plunger when it was at the uppermost position. This vent allowed the air in the lower part of the duct to come out when the robot was immersed into the water.

The movement of the robot is caused by the thrust of the waterjet generated by the plunger kicks. To let the robot move to the surface, the plunger kicks downward must be faster than those upward. We used a signal generator to generate a biased sinusoidal waveform, so the downward kick was given higher speed than the upward ones, and the robot could move towards the surface. To let it move towards the bottom, we just simply changed the two leads of the generator, and a suction thrust was generated which caused the robot to move towards the bottom. The trajectory of the robot work in the waterjet mode is shown in Figure 3.20.

This chapter has given a description of the components in the control and hydraulic systems. The procedure for testing the system in a water tank in buoyancy mode and waterjet mode with open-loop control were also outlined. Results showed that the hydraulic system has good response and both buoyancy mode and waterjet mode devices are applicable in underwater operation as the propulsion system.

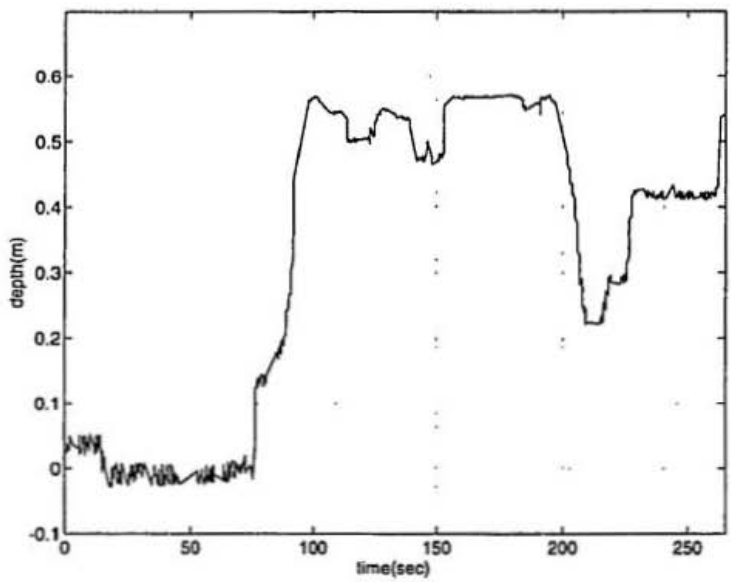

Figure 3.20: Robot trajectory in the waterjet test

Chapter 4

Discussions

In this project, we have tested robot depth control using an hydraulic system both in computer simulation and water tank test. We found that the system worked well as the propulsion system, the conventional control strategies are easy to implement and the system is quite stable.

Since this is our first attempt to design the underwater robot and operate it, there are some inevitable drawbacks in our mechanical setups and some defects in control system which caused the robot not to work efficiently and which we must improve in our future work.

Firstly, the robot power supply came from the oil stored in the accumulator. Since the duration of operation under water highly depended on the amount of oil inside the accumulator, a small amount of oil storage inside it will largely limit the operation of the robot and degrade its performance. In our case, the accumulator's

dimension is 1.18m long with an outer diameter of 0.15m, which is not compact from the view point of its weight and cost. Unfortunately, increasing its volume will not only make the robot heavier and more clumsy but will also make it more costly.

In our future works, we plan to incorporate a small pump with the accumulator and the storage chamber as part of the hydraulic system, so the oil can be circulated between the accumulator and storage chamber. This needs extra cable for the pump's power supply from the surface. Although the cable will cause some side effect as far as robot autonomy and maneuverability are concerned, robot duration under water will be much longer and there will be no need to retrieve the robot to drain the storage chamber and recharge the accumulator in short period. Therefore, the robot will have enough time to handle some more sophisticated task and the technical staff workload will be much relieved.

Secondly, the current robot propulsion system is the duct device. This restricts the robot to move in only one dimension. The maneuverability is largely reduced. As the underwater operations demands smarter or more sophisticated robots or vehicles to deal with the changing environment and to fully replace the human divers in the hazardous conditions, the robots or vehicles must have abilities to move freely in the water.

Thirdly, the current conventional control algorithm cannot compensate for the uncertainty of the dynamics of the robot and unexpected environment changes. Automatically compensating for the unexpected change of the environment is beyond the robot's control system capabilities. This makes the robot not able to adjust its parameters during the everchanging surrounding environment. When

unpredictable change comes, the robot has to wait for better conditions (for example, till the current subsides).

Control system of the underwater robot and ROVs nowadays must be robust enough to be capable of maintaining acceptable performance levels in the face of parametric uncertainty and unmodelled disturbances. More advanced control strategies have been developed and applied in the underwater robots or ROVs to compensate for the unexpected environment changes. Examples are the sliding mode control, neural networks and fuzzy logic control which have adaptive abilities. Future control system of our project must possess the characteristics of the advanced control strategies and be able to deal with the unpredictable ocean environment.

Other modifications or improvement must be made in our future works. The robot will be equipped with a depth sensor or pressure transducer, the current one is not accurate enough and cannot be applied in further test in wave tank or seawater. The robot may need some more sensors, such as speed sensor. When operating in deep water conducting complex assignment, an image producing system (like ultra light camera) is compulsory. Moreover, the robot must have self recovery capability in case that when anything in the control system, hydraulic system, sensor or mechanical system fails during the mission, the robot can surface by itself using emergency battery power.

Our robot, as well as the most underwater robots and vehicles currently being developed, connect to the controllers or supporting ships on the surface with cables, through which communication, data and power are transmitted. With these cables and their limited length, operating range of the robot and vehicles is

quite restricted. When deployed in the sea, the supporting ship, where the human operators stay, must accompany the robot or vehicles during the operation. AUVs and UUVs will dominate the underwater operation in the next one or two decades, some have been developed and tested in the world. We must consider replacing the current cable with some new communication technique like ultrasonic link, acoustic link or optical spectrum. In that case, the autonomy of the robot can be enhanced. This is the trend of developing underwater vehicles and a lot of researchers around the world are putting their effort into it.

Various researches are being conducted to increase the autonomy of the vehicle. People expect to develop an autonomous vehicle which collects all the information from its sensors and the data base, evaluates the information and makes the changes to comply with the requirement of the environment with minimum or without supervision. However, this expectation is not easy to achieve. The reason is that current technologies can not be adapted directly to the underwater robot due to the unpredictable, unstructured and rigorous seawater environment.

ROVs and underwater robots, especially those AUVs and UUVs, are still in the stage of trial rather than being applied extensively. Fully replacing the human divers and conducting sophisticated assignment in the ocean water will not be very soon.

Chapter 5

Conclusion

In this thesis, the feasibility of using the hydraulic system and the procedure of testing its control system using classical control strategies in the water tank have been demonstrated. The hydraulic system control using proportional control scheme and the open-loop control system have been test in the water tank in both buoyancy and waterjet modes. Buoyancy depth control using proportional-integral-derivative(PID) control scheme has been examined in simulation. Nonlinearity of the hydraulic system has been considered and stability analysis has been investigated by nonlinear describing function with Nyquist stability criterion.

Throughout the project, we have gained experience with propulsion and control systems of underwater robots. We also have confidence that in the long run we will be able to apply more advanced control strategies to the robot so that the robot will become a reliable, self-learning, self-adaptive, self-retrievable robot and can work in more rigorous environment.

The following conclusions on the project can be drawn on the basis of the performance analysis from the previous chapters:

- The buoyancy and waterjet can both be used as robot propulsion system when only vertical movement in the water is concerned.
- Closed-loop control system has promising performance in the underwater control operation.
- From the test and simulation of the hydraulic system, the system is a second-order, undamped system and has good transient response and frequency response.
- The hydraulic system is stable since the describing function $-1/N$ is not enclosed by the Nyquist plot in the $G(j\omega)H(j\omega)$ plane.
- The PID control scheme outperforms the proportional control scheme as shown by the results of the simulation in Chapter 2.

In the underwater operations these days, the task complexity and difficulty is increasing all the time, as is the requirement for higher performance. With this comes the requirement for more flexible and sophisticated control strategies. Our future works must be focused on increasing the adaptivity and learning abilities of the control system.

The design of the robot mechanical setup and control system with its verification in the water tank was a challenge. Our goal to test different propulsion system devices and implement more advanced control strategies on this robot will hopefully contribute to the state of the art.

References

Allocca, J. A. and Stuart, A. 1984 "Transducers Theory and Applications", *Prentice-Hall Company.*

Blidberg, D. R. 1991 "Autonomous Underwater Vehicles: A Tool for the Ocean", *Unmanned System, Spring 1991*, pp.10-15.

Farbrother, H. N. R. and Stacey, B. A. 1993 "Aspects of Remotely Operated Vehicle Control – A Review", *Underwater Technology*, Vol. 19, no. 1, pp24-36.

Hinchey, M. J. 1994 "Potential for Subsea Robot Control", *Ocean Engineering Journal*, Pergamon Press.

Muggeridge, K. 1994 'Control of Subsea Robot", M.Eng. Thesis, Memorial University of Newfoundland.

Muggeridge, K. and Hinchey, M. J. 1992 "A New Jet Propulsion Device for Small Subsea Robot", *Proc. 1992 Sym. on Autonomous Underwater Vehicle Tech.*, pp.112-115.

Ogata, K. 1970 "Modern Control Engineering", *Prentice-Hall, Inc.*.

Ogata, K. 1987 "Discrete-Time Control Systems", *Prentice-Hall, Inc.*.

Yoerger, D. R. and Slotine, J.-J. E. 1985 "Robust Trajectory Control of Underwater Vehicles", *IEEE J. Ocean. Eng.*, Vol. 10, pp.462-470.

Yoerger, D. R., Newman, J. B. and Slotine, J.-J. E. 1986 "Supervisory Control System for the JASON ROV", *IEEE J. Ocean. Eng.*, Vol. 11, pp.392-400.

Yoerger, D. R. and Newman, J. B. 1989 "Control of Remotely Operated Vehicles for Precise Survey", *Proc. ROV'89 Conf.*, pp.123-127.

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.

Yoerger, D. R. 1990 "Precise Control of Underwater Robots: Why and How", *First Workshop on Mobile Robots for Subsea Environment 1990*, pp.113-117.

Yoerger, D. R., Bradley, A. M. and Walden, B. 1991 "The Autonomous Benthic Explorer", *Unmanned System, Spring, 1991*, pp.17-23.

Yuh, J. 1989 "Overview of an Interlligent System for Underwater Robotic Vehicles", *13th ROBOTS Conf. 1989*, pp.6.15-6.22.

Yuh, J. 1990 "A Neural Net Controller For Underwater Robotic Vehicles", *IEEE J. Ocean. Eng.*, Vol. 15, pp.161-166.

Yuh, J. and Gonugunta, D. V. 1992 "Effect of Thruster Dynamics and Advanced Control of Underwater Robotic Vehicles using Neural Network", *International Sym. on Robotics and Manufacturing 1992*, pp.735-740.

Yuh, J. and Lakshmi, R. 1993 "An Intelligent Control System for Remotely Operated Vehicles", *IEEE J. Ocean. Eng.*, Vol. 18, pp.55-62.

Yuh, J. and Choi, S. K. 1993 "Design of An Omni-Directional Underwater Robotic Vehicle and Its Coordinated Motion Control", *Proc. American Control Conf.*, pp.570-574.

Zheng, X. 1992 "Layered Control of a Practical AUV", *Proc. 1992 Sym. on Autonomous Underwater Vehicle Tech.* pp.142-147.

Appendix A

Simulation Programs

```
/* This program is for the simulation of the underwater robot with
buoyancy from the duct as its propulsion device. */

#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#define  DENSITY    1000            /* density of water */
#define  PI         3.1415926       /* pi */
#define  G          9.81            /* gravitational acceleration */

main(){

  FILE *in, *out;
  float mass,                        /* mass of the rig */
        drag_coe,                    /* drag coefficient */
        dia_plunger,                 /* diameter of the plunger*/
        stroke,                      /* rod stroke */
        pro_gain,                    /* proportional gain */
        set_dep,                     /* command or set depth */
        start_dep,                   /* starting depth */
        start_vel,                   /* starting velocity */
        delt,                        /* duration of step in time */
        area,                        /* frontal area of robot */
        area_plunger, buoy, relay_height, error, yor, drag, accel,
        time_scale, time = 0.0, rod_pos = 0.0;
   int j = 0;

  if((in=fopen("plunger.in", "r")) == NULL){
      puts("cannot open file.\n");
      exit(1);
      }

  if((out=fopen("plunger.out", "w")) == NULL){
      exit(1);
      }


  fscanf(in, "%f%f%f%f%f%f%f%f%f%f%f%f%f%f",
             &time_scale, &mass, &drag_coe, &delt,
             &relay_height, &pro_gain,  &stroke, &dia_plunger,
```

```
            &start_dep, &set_dep, &start_vel, &area);

   printf("  time_scale:%-6.2f     mass:%-6.2f   drag_coe:%-6.2f delt:%-6.3f\n\
relay_height:%-6.2f pro_gain:%-6.2f     stroke:%-6.3f dia_plunger:%-6.2f\n\
   start_dep:%-6.2f  set_dep:%-6.2f  start_vel:%-6.2f area:%-6.2f\n",
          time_scale, mass, drag_coe, delt,
          relay_height, pro_gain,  stroke, dia_plunger,
          start_dep, set_dep, start_vel, area);

  area_plunger= PI * dia_plunger* dia_plunger/4.0;

   do{
       error = set_dep - start_dep;
       rod_pos = pro_gain * error;

       if(error) rod_pos += relay_height * error/fabs((double)error);

       if(rod_pos > stroke) rod_pos = stroke;
       if(rod_pos < -stroke) rod_pos = -stroke;

       buoy = rod_pos * area_plunger * DENSITY * G;
       drag = drag_coe * DENSITY * fabs(start_vel) * start_vel * area
/ 2.0;
       accel = (-drag + buoy)/mass;
       start_vel += delt * accel;
       start_dep += delt * start_vel;
       yor = time * time_scale;
       j++; if(!(j%10)){
       fprintf(out, "%12.4f%12.4f%12.4f\n", start_vel, start_dep, yor);}
       time += delt;
       if(fabs(set_dep-start_dep)<=0.05 && fabs(start_vel) <= 0.02) break;
  }while(1);

  fclose(in);
  fclose(out);
}
```

```
/* This program is for the simulation of the underwater robot with
buoyancy from the duct as its propulsion device. The control action is
the PID control scheme*/

#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#define  DENSITY    1000            /* density of water */
#define  PI         3.1415926       /* pi */
#define  G          9.81            /* gravitational acceleration */

main(){
  FILE *in, *out;
  float mass,                       /* mass of the rig */
        drag_coe,                   /* drag coefficient */
        dia_plunger,                /* diameter of the plunger */
        stroke,                     /* rod stroke */
        set_dep,                    /* command or set depth */
        start_dep,                  /* starting depth */
        start_vel,                  /* starting velocity */
        delt,                       /* duration of step in time */
        area,                       /* frontal area of robot */
        kp,                         /* proportional gain */
        ki,                         /* integral gain */
        kd,                         /* derivative gain */
        area_plunger, buoy, relay_height, error, yor, drag, accel,
        time_scale, time = 0.0, Rod_pos = 0.0, rod_pos,
        error_1 = 0.0, error_2 = 0.0;
   int  j = 0;

  if((in=fopen("pid.in", "r")) == NULL){
      puts("cannot open file.\n");
      exit(1);
      }

  if((out=fopen("pid.out", "w")) == NULL){
      exit(1);
      }

  fscanf(in, "%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f",
```

```
            &time_scale, &mass, &drag_coe, &delt,
            &relay_height, &stroke, &dia_plunger,
            &start_dep, &set_dep, &start_vel, &area,
            &kp, &ki, &kd);

    printf("time_scale:%-6.2f  mass:%-6.2f  drag_coe:%-6.2f delt:%-6.2f\n\
            relay_height:%-6.2f  stroke:%-6.2f dia_plunger:%-6.2f\n\
            start_dep:%-6.2f  set_dep:%-6.2f  start_vel:%-6.2f area:%-6.2f\n\
            kp:%-6.2f ki:%-6.2f kd:%-6.2f\n",
            time_scale, mass, drag_coe, delt,
            relay_height, stroke, dia_plunger,
            start_dep, set_dep, start_vel, area,
            kp, ki, kd);

    area_plunger = PI * dia_plunger * dia_plunger/4.0;
    do{
        error = set_dep - start_dep;
        rod_pos=kp*(error-error_1)+ki*error+kd*(error-2*error_1+error_2);
        Rod_pos += rod_pos;
        if(error) Rod_pos += relay_height * error/fabs((double)error);
        if(Rod_pos > stroke) Rod_pos = stroke;
        if(Rod_pos < -stroke) Rod_pos = -stroke;
        buoy = Rod_pos * area_plunger * DENSITY * G;
        drag = drag_coe * DENSITY * fabs(start_vel) * start_vel * area
/ 2.0;
        accel = (-drag + buoy)/mass;
        start_vel += delt * accel;
        start_dep += delt * start_vel;
        yor = time * time_scale;
        j++; if(!(j%10)){
        fprintf(out, "%12.4f%12.4f%12.4f\n", start_vel, start_dep, yor);}
        time += delt;
        if(fabs(set_dep-start_dep) <=0.05 && fabs(start_vel) <= 0.02) break;
        error_2 = error_1;
        error_1 = error;
    }while(1);

    fclose(in);
    fclose(out);
}
```

Appendix B

Control Programs

```
/* This is the header file for the all the buoyancy depth control programs. */

/*********************************
 * include file for depth control *
 *********************************/
#include <stdio.h>
#include <math.h>
#include <conio.h>
#include <stdlib.h>

/*****************************************
 * defining the address of the bios timer *
 *****************************************/
#define     BIOS_DATA_SEG       0x40
#define     TIMER_DATA          0x6c

/*********************************************
 * defining segment address for the Keithley *
 *********************************************/
#define     KEITHLEY_SEG        0xcff0      /* segment address for keithley */

/*****************************
 * digital to analog control *
 *****************************/
#define     DA_CONTROL          0x84        /* d/a control, indicate which
                                               channal, must precede d/a data
*/
#define     CHAN0_LOW           0x00        /* channal 0 low byte */
#define     CHAN0_HI            0x01        /* channal 0 high byte */
#define     CHAN1_LOW           0x02        /* channal 1 low byte */
#define     CHAN1_HI            0x03        /* channal 1 high byte */
#define     DA_DATA             0x85        /* data value into d/a converter
*/
#define     DA_STROBE           0x9d        /* d/a strobe location */
#define     ENABLE              0x40        /* enable strobe */
#define     DISABLE             0x80        /* disable strobe */
#define     ISSUE               0x01        /* output data in latch */

/*****************************
 * analog to digital control *
 *****************************/
```

```
#define     SEL_CHANNEL          0x8a      /* select the channel for a/d */
#define     SEL_SLOT             0x81      /* select the slot for a/d */
#define     GLOBAL_GAIN          0x9a      /* analog input gain */
#define     GAIN1                0x00      /* analog input gain is 1 */
#define     GAIN2                0x01      /* analog input gain is 2 */
#define     GAIN5                0x02      /* analog input gain is 5 */
#define     GAIN10               0x10      /* analog input gain is 10 */
#define     AD_STATUS            0x98      /* a/d start/status */
#define     AD_LOW               0x82      /* a/d low data */
#define     AD_HIGH              0x83      /* a/d high data */
#define     SLOT6                6         /* a/d slot always is 6 */

/***********************
 * keithley input gain *
 ***********************/
#define     GAIN                 4         /* gain of servo valve */
#define     KP                   60        /* proportional gain in PID*/
#define     KI                   1         /* integral gain in PID */
#define     KD                   10        /* derivative gain in PID */

/*****************************
 * analog to digital channel *
 *****************************/
#define     CHANNEL_LVDT         6         /* da channel for LVDT is 6 */
#define     CHANNEL_DEP          5         /* da channel for depth sensor is
5 */
#define     CHANNEL_VEL          8         /* da channel for velo sensor is
8 */

/********************
 * stroke and gains *
 ********************/
#define     K_LVDT               0.0254    /* return 1 volt, rod move 2.5cm
*/
#define     K_DEP                0.333     /* relationship between depth and
                                              voltage from depth sensor */
#define     K_VEL                          /* relationship between velocity
                                              and voltage from velocity sen-
sor */
#define     STROKE               0.0625    /* rod stroke is 0.0625 m */
#define     K_L                  39.7      /* LVDT gain */
```

```
#define    K_F                3        /* depth control feedback gain */

/*********************
 * global variables *
 *********************/
unsigned char  _far *select_slot, _far *select_channel, _far *ad_status,
          _far *global_gain, _far *da_control, _far *da_data,
          _far *da_strobe, _far *ad_low, _far *ad_high;

/************************
 * function declaration *
 ************************/
void da_output0(double);
void da_output1(double);
void point_pointers(void);
void delay(void);
double get_LVDT(unsigned char channel);
double get_DEPTH(unsigned char channel);
double get_VELOCITY(unsigned char channel);
void initialize(void);
```

```
/* This program controls the depth of robot in the water tank using
   buoyancy, the argument you give in the command is the depth which
   you want the robot to hover. usage: depth 3, means you want the
   robot to hover at 3 meters under the surface. The control scheme in
   this program is PID.*/

#include "depth.h"

main(int argc, char *argv[])
{
  float set_dep, start_dep, error, error_1, error_2, rod_pos, Rod_pos,
        lvdt_pos, output_val, start_vel;

  point_pointers();

  if(argc < 2){
     printf("%s\n%s\n",
            "You must enter the depth on the command line.",
            "Usage: depth 3");
       exit(0);
       }

  set_dep = atof(argv[1]); /* get the set depth from the command line */

  start_dep = get_DEPTH(CHANNEL_DEP)*K_DEP - 0.2;   /* get the current depth
                                from the depth sensor */
      if(set_dep == start_dep){
      printf("The robot is at the depth you want it to hover.");
      exit(0);
      }

  else{
    do{
      error = set_dep - start_dep;
      rod_pos = KP*(error-error_1)+KI*error+KD*(error-2*error_1+error_2);
      Rod_pos += rod_pos;                    /* PID control scheme */

      if(Rod_pos > STROKE) Rod_pos = STROKE;
      if(Rod_pos < -STROKE) Rod_pos = -STROKE;

      lvdt_pos=get_LVDT(CHANNEL_LVDT)*K_LVDT; /* get the position of the rod
```

```
*/
      output_val = GAIN * (Rod_pos - lvdt_pos);
      da_output1(output_val);    /* send the error signal to servo valve */
      start_dep = K_DEP * get_DEPTH(CHANNEL_DEP) - 0.2;
      start_vel = K_VEL * get_VELOCITY(CHANNEL_VEL); /* get the current vel
*/
      if(fabs(set_depth-start_dep) ≤ 0.05 && fabs(start_vel) ≤0.02) break;
      }while(1);
     }
   printf("The robot is at the %f meters depth.\n", set_depth);
   da_output1(0.0);              /* close the servo valve */
}
```

```
/* this problem sets up the pointers */

#include "depth.h"

void point_pointers()
{
 FP_SEG(bios_ticks) = BIOS_DATA_SEG;
 FP_OFF(bios_ticks) = TIMER_DATA;

 FP_SEG(select_slot) = KEITHLEY_SEG;
 FP_OFF(select_slot) = SEL_SLOT;

 FP_SEG(select_channel) = KEITHLEY_SEG;
 FP_OFF(select_channel) = SEL_CHANNEL;

 FP_SEG(ad_status) = KEITHLEY_SEG;
 FP_OFF(ad_status) = AD_STATUS;

 FP_SEG(global_gain) = KEITHLEY_SEG;
 FP_OFF(global_gain) = GLOBAL_GAIN;

 FP_SEG(ad_low) = KEITHLEY_SEG;
 FP_OFF(ad_low) = AD_LOW;

 FP_SEG(ad_high) = KEITHLEY_SEG;
 FP_OFF(ad_high) = AD_HIGH;

 FP_SEG(da_control) = KEITHLEY_SEG;
 FP_OFF(da_control) = DA_CONTROL;

 FP_SEG(da_data) = KEITHLEY_SEG;
 FP_OFF(da_data) = DA_DATA;

 FP_SEG(da_strobe) = KEITHLEY_SEG;
 FP_OFF(da_strobe) = DA_STROBE;
}
```

```
/* this program send a voltage to channel 1 */

#include "depth.h"

void da_output1(double voltage)  /* channel 1 */
{
  int setting;
  setting = (int)(voltage * (2048.0/10.0) + 2048);

  if(setting > 4095) setting = 4095;
  if(setting < 0) setting = 0;

  *da_strobe = ENABLE;
  *da_control = CHAN1_LOW;      /* low byte channel one */
  *da_data = (unsigned char)(setting & 0x00ff);
  *da_control = CHAN1_HI;       /* high nibble channel one */
  *da_data = (unsigned char)((setting>>8) & 0x000f);
  *da_strobe = ISSUE;
}
```

```
/* This program is to get the voltage reading from the LVDT */

#include "depth.h"

void delay(){
  int i; for(i=0; i<5; i++);
}

double get_LVDT(unsigned char channel)
{
 double vol_LVDT;
 unsigned char lowbyte, highbyte;
 unsigned int value;

 point_pointers();

 *select_channel = channel;      /* differential input through "channel" */
 *select_slot = SLOT6;           /* input from slot 6 */
 *global_gain = GAIN1;           /* set the global gain to 1 */

 if(*ad_status == 0xff){         /* check if ad converter is busy */
      do{
        ;
        }while(*ad_status == 0xff); /* wait till conversion is finish */
     delay(); delay();
     *ad_status = 0x00;      /* trigger the ad converter */
     }
  else {delay(); delay(); *ad_status = 0x00;}

  lowbyte = *ad_low;             /* read the converter low byte */
  highbyte = *ad_high;           /* read the converter high byte */
  value = (unsigned int)highbyte;
  value = (value << 8) & 0x0f00;
  value += (unsigned int)lowbyte;
  vol_LVDT = (double)(-10.0 + ((float)value * 10.0/2048.0));

 return vol_LVDT;
}
```

```
/* This program is to get the robot depth from the depth sensor */

#include "depth.h"

double get_DEPTH(unsigned char channel)
{
 double vol_DEPTH;
 unsigned char lowbyte, highbyte;
 unsigned int value;

 pointer_pointers();

 *select_channel = channel;      /* differential input through "channel" */
 *select_slot = SLOT6;           /* input from slot 6 */
 *global_gain = GAIN1;           /* set the global gain to 1 */

 if(*ad_status == 0xff){         /* check if ad converter is busy */
      do{
        ;
        }while(*ad_status == 0xff); /* wait till conversion finish */
    delay(); delay();
        *ad_status = 0x00;       /* trigger the ad converter */
    }
 else {delay(); delay(); *ad_status = 0x00;}

 lowbyte = *ad_low;
 highbyte = *ad_high;
 value = (unsigned int)highbyte;
 value = (value << 8) & 0x0f00;
 value += (unsigned int)lowbyte;
 vol_DEPTH = (double)(-10.0 + ((float)value * 10.0/2048.0));

 return vol_DEPTH;
}
```