1

National Library
of Canada

Bibliothèque nationale
du Canada

Canadian Theses Service

Service des thèses canadiennes

Ottawa, Canada
K1A 0N4

NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Previously copyrighted materials (journal articles, published tests, etc.) are not filmed.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

Les documents qui font déjà l'objet d'un droit d'auteur (articles de revue, tests publiés, etc.) ne sont pas microfilmés.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, S.R.C. 1970, c. C-30.

Modeling of a Wave Generator
and the
Design of An Optimal Estimator

by

P.H. Laurich

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

Ottawa-Carleton Institute for Electrical Engineering
Faculty of Engineering
Department of Systems and Computer Engineering
Carleton University

March, 1988

Copyright © 1988, Peter Hermann Laurich

Permission has been granted to the National Library of Canada to microfilm this thesis and to lend or sell copies of the film.

The author (copyright owner) has reserved other publication rights, and neither the thesis nor extensive extracts from it may be printed or otherwise reproduced without his/her written permission.

L'autorisation a été accordée à la Bibliothèque nationale du Canada de microfilmer cette thèse et de prêter ou de vendre des exemplaires du film.

L'auteur (titulaire du droit d'auteur) se réserve les autres droits de publication; ni la thèse ni de longs extraits de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation écrite.

ISBN 0-315-40645-3

The undersigned hereby recommends to the Faculty of Graduate
Studies and Research acceptance of the thesis,

Modeling of a Wave Generator
and the
Design of An Optimal Estimator

submitted by
Peter H. Laurich

in partial fulfillment of the
requirements for the degree of Master of Engineering

Thesis Supervisor

Chairman,
Department of Systems and
Computer Engineering

Department of Systems and Computer Engineering
Faculty of Engineering
Carleton University

April, 1988

ABSTRACT

This thesis deals with the development of a mathematical model of a wave generator and the design of an optimal estimator derived from the model. The mathematical model is developed by a graph theoretical analysis of the wave generator. Some results comparing the measured response of the system to the response predicted by the model are presented.

An optimal estimator based on the steady state solution of the Kalman filter is outlined and used to provide improved estimates of the states of the system. The measured responses of the system are also compared with the estimated states to demonstrate the performance of the estimator. Finally, the real-time implementation of an optimal controller based on the optimal estimator is discussed. Although the motivation for developing the mathematical model and the estimator is to allow an optimal controller to be designed, the design of an optimal controller is only discussed in general terms.

ACKNOWLEDGEMENTS

I would like to express my thanks to my supervisor, Dr. A.U.H Sheikh, for his guidance, thoughtful advice and encouragement. I would also like to thank the Hydraulics Laboratory of the National Research Council of Canada for the use of their exceptional facilities and for granting me the time to work on my thesis. This project would not have been possible without the above assistance.

I would also like to thank Mr. M.D. Miles of the Hydraulics Laboratory for his pioneering work in the development of the kinematic equations of the wave generator and assistance throughout many of the phases of this work.

TABLE OF CONTENTS

Abstract	i
Acknowledgements	ii
Table Of Contents	iii
List Of Figures	iv
List Of Tables	v
List Of Appendices	vi
List Of SYMBOLS	vii
INTRODUCTION	1
1.1 Wave Generation	1
1.2 Hydraulics Laboratory	4
1.3 Software	5
MATHEMATICAL MODEL DEVELOPMENT	7
2.1 Introduction	7
2.2 Wave Generator Kinematics	10
2.3 Derivation of Differential Equations	15
2.3.1 Servo Valve	15
2.3.2 Hydraulic Actuator	16
2.3.2.1 Hydraulic Subcomponent	18
2.3.2.2 Mechanical Subcomponent	21
2.3.3 Mechanical Subsystem	23
2.3.3.1 Lever Arm	24
2.3.3.2 Spring	26
2.3.3.3 Mass	26
2.3.3.4 Damper	27
2.3.3.5 Model of the Mechanical Subsystem	27
2.4 Continuous Time Equations	29
2.5 Model Parameter Values	33
2.5.1 Servo Valve	34
2.5.2 Hydraulic Actuator	35
2.5.3 Mechanical Subsystem	37
2.5.3.1 Lever Arm	37
2.5.3.2 Spring	38
2.5.3.3 Mass	38
2.5.3.4 Damper	40
2.6 Discrete Time Equations	40
MODEL VERIFICATION AND REFINEMENT	46
3.1 Test Data	46
3.2 Processing of Measured Data	51
3.2.1 Accelerometer Compensation	52
3.3 Test Results	55
ESTIMATOR DEVELOPMENT	65
4.1 Introduction	65
4.2 Optimal Control	67
4.3 Optimal Estimation	74
4.4 Solution of the Steady State Optimal Estimator	78
4.5 Estimator and Model Refinement	82
CONCLUSIONS AND RECOMMENDATIONS FOR FUTURE WORK	102
5.1 Summary	102
5.2 Implementation	103
5.3 Future Work	106
REFERENCES	108

LIST OF FIGURES

Figure 2.1 Wave Generator	8
Figure 2.2 Kinematics of Wave Generator	11
Figure 2.3 Hydraulic Actuator	17
Figure 2.4 Hydraulic Subcomponent of the Actuator	17
Figure 2.5 Mechanical Subcomponent of the Actuator	17
Figure 2.6 Simplified Wave Generator Model	25
Figure 2.7 Ideal Lever Arm	25
Figure 2.8 Terminal Graph of Simplified Wave Generator Model	25
Figure 2.9 Displacement Amplification Factor	39
Figure 2.10 Force Amplification Factor	39
Figure 2.11 System and Sampling Operations	40
Figure 3.1 Compensation of the Actuator Accelerometer	53
Figure 3.2 Compensation of the Glider Accelerometer	53
Figure 3.3 Actuator Acceleration for Irregular Wave Drive Signal	59
Figure 3.4 Glider Acceleration for Irregular Wave Drive Signal	60
Figure 3.5 Actuator Acceleration for Square Wave Drive Signal	61
Figure 3.6 Glider Acceleration for Square Wave Drive Signal	62
Figure 3.7 Actuator Acceleration for 0.5 Hz Sinusoidal Drive Signal	63
Figure 3.8 Glider Acceleration for 0.5 Hz Sinusoidal Drive Signal	64
Figure 4.1 Block Diagram of System with Estimator and Control Law	79
Figure 4.2 Actuator Position Comparison for System Model One	83
Figure 4.3 Actuator Velocity Comparison for System Model One	84
Figure 4.4 Glider Position Comparison for System Model One	85
Figure 4.5 Glider Velocity Comparison for System Model One	86
Figure 4.6 Differential Pressure Comparison for System Model One	87
Figure 4.7 Actuator Acceleration Comparison for System Model One	88
Figure 4.8 Glider Acceleration Comparison for System Model One	89
Figure 4.9 Actuator Position Comparison for Square Wave Test	93
Figure 4.10 Actuator Velocity Comparison for Square Wave Test	94
Figure 4.11 Glider Position Comparison for Square Wave Test	95
Figure 4.12 Glider Velocity Comparison for Square Wave Test	96
Figure 4.13 Differential Pressure Comparison for Square Wave Test	97
Figure 4.14 Actuator Acceleration Comparison for Square Wave Test	98
Figure 4.15 Glider Acceleration Comparison for Square Wave Test	99
Figure 4.16 Actuator Acceleration Comparison (Open Loop)	100
Figure 4.17 Glider Acceleration Comparison (Open Loop)	101

LIST OF TABLES

Table 2.1 Kinematic Data	12
Table 2.2 Eigenvalues of the Sixth Order System	44
Table 3.1 Drive Signal File Characteristics	47
Table 3.2 Results Of Model Verification Tests	57
Table 4.1 Control and Estimation Duality	77
Table 4.2 Estimator Performance Tests	81
Table 4.3 Characteristics of System Models Tested	91

LIST OF APPENDICES

Appendix A	Software for the Conversion of the Continuous Time System Model to the Equivalent Discrete Time System Model	A-1
Appendix B	Software for the Calculation of the Steady State Optimal Estimator	B-1

LIST OF SYMBOLS

A	cross-sectional area of the piston (excluding the area of the piston rod)
B	damping coefficient
C	hydraulic fluid compressibility constant
C_L	actuator leakage coefficient
D_A	lever arm displacement amplification factor
f	force
F	continuous time system matrix
F_A	lever arm force amplification factor
G	continuous time input matrix
H	output matrix
H_C	control Hamiltonian
H_e	estimation Hamiltonian
J	cost of control errors
K	feedback gain matrix
k_q	servo valve no load flow gain
L	estimator gain matrix
M	mass
p	differential pressure across the piston
P_s	hydraulic fluid supply pressure
q	flow through the servo valve
Q_1, Q_2	weighting matrices for state error and control effort
$R_1, R_2, R_6, R_7,$ R_{13}, R_{67}	lengths in the wave generator kinematics figure
R_{ACT}	length of the actuator with the piston centred
R_{EXT}	extension of the piston in the actuator
T	sampling period
u	system input
V	velocity
x	state variable
$\hat{x}$	updated estimate of the state variables
$\bar{x}$	estimated state variables
y	output measurement
$\alpha_1, \alpha_2, \alpha_3, \alpha_4$	angles in the wave generator kinematics figure
β	angle between the X axis and the axis of the actuator
Γ	discrete time input matrix
γ_G	angles in the compensation of the glider accelerometer
δ	displacement
ϵ, σ	angles in the compensation of the actuator accelerometer
θ	lever arm angle
θ_0	lever arm angle when actuator piston is centred
ρ	angle between the glider push rod and the x axis
τ	servo valve time constant
ϕ	angle between the Y axis and the face of the wave board
Φ	discrete time system matrix
ω	frequency

CHAPTER 1

INTRODUCTION

1.1 WAVE GENERATION

Wave generation systems are used extensively by research institutions throughout the world investigating the performance of offshore and coastal structures in simulated sea states. As structures such as oil exploration and production platforms are moved into deeper water and more difficult sea conditions, it becomes increasingly important to know the response of the structure before it is constructed and put into active use. The response of a structure is typically determined by subjecting a scale physical model of the structure to simulated sea conditions designed to reproduce the severe storm conditions to which the structure could be exposed. Scale model breakwaters must also be tested to ensure that they will survive the environment into which they are to be placed and to ensure that the surrounding coastal areas are not damaged by the existence of the breakwater. Recent failures of offshore structures and breakwaters emphasize the importance of this testing.

Waves are generated by moving a wave board through the water along a predefined trajectory. The interaction between the wave board and the water causes waves to be produced. These waves will have the desired characteristics if the trajectory is specified such that the dynamic

affects of the wave generator, the control system and the physical process relating the wave board motion to wave heights are all correctly accounted for.

A sea state is initially defined by a time series of wave heights in the test area. Linear wave theory is used to compute the velocity of the wave board which will result in the desired waves being produced. A model of the wave generator and the control system is used to transform the wave board velocity specifications into a time series of positions of the actuator which drives the wave board. The time series of actuator positions, known as a drive signal, is input into the control system and the resulting waves in the test area are measured and compared with the desired waves. The drive signal is modified until an appropriate reproduction is achieved.

The demands on the accuracy to which waves must be reproduced increases as more information on naturally occurring sea states becomes available and as researchers discover the characteristics of seas which are important in determining the response of structures. These demands are responsible for the appearance of short crested wave generation facilities and research into areas such as improved control and active wave absorption. Short crested wave facilities consist of many wave generators independently controlled and actuated so that the generated waves more closely resemble the three dimensional short crested waves observed in nature. Active wave absorption is concerned with the absorption of waves directed towards the wave generator while generating

desired waves. Reflections from the structure or from the walls of the test basin which reach the wave generator will be reflected back into the test area if they are not absorbed. These reflections, which do not occur in nature, corrupt the desired sea state.

The requirement for more accurate reproduction of sea states ultimately defines the specifications of the control system which controls the wave generator. Traditional methods used to control the position of the actuator involve using simple analog or digital controllers. These controllers use some type of position feedback and possibly another stabilization feedback to reduce the error between the drive signal and the measured actuator position. The availability of low cost yet powerful computers makes it possible to consider more complex control strategies. An optimal controller is an example of a controller which could result in improved wave generation capabilities.

Optimal controllers consist of a control law and an estimator. The control law relates the control signal to a linear combination of the system states which are made available by the estimator. Both the control law and the estimator are based on a mathematical model of the system. The emphasis of this report is on the development of the mathematical model and the estimator.

1.2 HYDRAULICS LABORATORY

The Hydraulics Laboratory of the National Research Council of Canada (NRC) is involved in research requiring the generation of waves. There are several wave generators in use in the laboratory including the short crested wave generation facility which consists of sixty four wave generators. Each of the wave generators in the short crested facility is individually controlled by a direct digital control system. It is the only short crested facility currently in existence which uses digital control.

The control system allows the feedback signals, the drive signal, the control signal and intermediate calculations in the control loop to be recorded. These data can be recorded and transferred to a general purpose computer for further analysis. The data used to verify the mathematical model of the wave generator and to demonstrate the suitability of the estimator were obtained from the control system.

The test facilities used were a test flume, 2 wave generators from the short crested wave generation facility, a digital control system and a data analysis computer system. The 2 meter high by 1 meter wide by 30 meter long flume was equipped with a wave absorbing beach at the end opposite to the wave generators. The digital control system consisted of all of the computer equipment, transducers and signal conditioning circuitry necessary for control. All of the analysis of acquired data was

done on the general purpose computer system in the laboratory. The testing was carried out in a water depth of 1.5 meters.

1.3 SOFTWARE

Software was used for data collection, data analysis and for the design and analysis of the estimator. The data collection software is part of the digital control system, the details of which are not relevant to this report. The data analysis software used is part of the general purpose software system, GEDAP, which is used in the NRC Hydraulics Laboratory [ref. 3]. The GEDAP system consists of a series of programs which operate on data files. Each program is designed for a specific function and generally inputs a data file, operates on the data and outputs a modified data file. The main GEDAP programs used include low pass Kaiser digital filtering, multiple integration and differentiation, non-linear least squared error fitting and plotting software. The software developed for the design and analysis of the estimator is described in this report.

The remainder of this report details the development of the mathematical model and the estimator. This starts with a kinematical analysis in which the major components of the wave generator are identified and the equations defining the geometry of the components are developed. Using the kinematic data and information on the actuator used to drive the wave board, a mathematical model describing the system is developed. Tests done on the actual wave generator are documented and the

use of the data to refine the mathematical model is described. The development of an estimator based on the final mathematical model is described and some results obtained using the estimator are included. Finally, some of the future work required to implement the estimator in real-time and to use it for optimal control is outlined.

CHAPTER 2

MATHEMATICAL MODEL DEVELOPMENT

2.1 INTRODUCTION

The wave generator to be modeled is shown in Figure 2.1. The major components identified in the figure are the hydraulic actuator, lever arm, glider, wave board, glider push rod, and wave board push rod. The actuator provides the force required to move the wave board and glider. A digital control system controls the motion of the piston in the actuator by controlling the flow of pressurized hydraulic fluid through the actuator. The wave generator was designed by the NRC Hydraulics Laboratory and Davis Engineering Ltd. of Ottawa, and was built by Davis Engineering.

The purpose of the mathematical model of the wave generator is to adequately describe the response of the system to known inputs. The model must be as simple as possible to allow for future implementation in real-time control algorithms, yet include sufficient detail to provide a representative model of the actual system.

The system under study is a high order nonlinear system. In order to reach the desired form with a manageable size model, several assumptions and simplifications were made. Each of the assumptions and simplifications is documented below. The validity of the assumptions and

Figure 2.1 Wave Generator

simplifications are ultimately confirmed by comparing the measured response of the system to known inputs with the estimated response using the same input conditions.

The desired form for the model of the system is a set of linear, constant coefficient discrete time equations of the form:

$$x(n+1) = \Phi x(n) + \Gamma u(n) \quad (2.1)$$

$$y(n) = Hx(n) \quad (2.2)$$

where $x(n)$ is a vector of state variables, Φ is the discrete system matrix, Γ is the discrete control input matrix, $u(n)$ is the control input, $y(n)$ is the output measurement and H is the discrete system output matrix. The system will be treated as a lumped parameter system in which the physical characteristics of the components are assumed to exist at discrete points. Such systems can be described in continuous time using ordinary differential equations.

The six steps involved in developing the desired model are:

- 1) derivation of the kinematic equations describing the geometry of the system
- 2) derivation of the differential equations describing the response of the system
- 3) identification of the state variables, formation of the continuous time system matrix, formation of the continuous time

control input matrix and formation of the continuous time output measurement matrix

- 4) specification of the values for the parameters in the continuous time model
- 5) conversion from continuous time to discrete time
- 6) verification of the model.

The first five of these steps will be discussed in detail in this section. Discussion of the model verification process is deferred until Chapter 3.

2.2 WAVE GENERATOR KINEMATICS

The kinematic equations define the position of the major components of the wave generator with respect to the lever arm angle. The main use of these equations is in the data analysis phase where significant measurements, such as the position of the actuator, are derived from the measured lever arm angle. The kinematic equations are derived from the mechanical drawings of the wave generator.

A simplified view of the wave generator is shown in Figure 2.2. The major points of interest, relevant angles and the coordinate system defined are indicated in the figure. Table 2.1 describes the positions, lengths and angles defined in the drawing and indicates the actual value of any measurements, where possible. The length of the glider push rod and the length of the actuator with the piston centred, are indicated as

Figure 2.2 Kinematics of Wave Generator

Table 2.1 Kinematic Data

Variable	Value	Description
0 (x_0, y_0)	(0.0, 0.0)	centre of rotation of the lever arm
1 (x_1, y_1)	(calculated)	point of attachment of the glider push rod to the lever arm
2 (x_2, y_2)	(calculated)	point of attachment of the wave board push rod to the lever arm
3 (x_3, y_3)	(calculated, -0.7724 m)	point of attachment of the glider push rod to the glider
4 (x_4, y_4)	(calculated)	point of attachment of the wave board push rod to the wave board
5 (x_5, y_5)	(calculated, -2.6147 m)	point of attachment of the wave board to the glider
6 (x_6, y_6)	(calculated)	point of attachment of the actuator to the lever arm
7 (x_7, y_7)	(1.04216 m, 0.06426 m)	point of attachment of the actuator to the machine frame
R_1	1.1603 m	distance from point 0 to point 1
R_2	1.2129 m	distance from point 0 to point 2
R_6	0.2873 m	distance from point 0 to point 6
R_7	1.0441 m	distance from point 0 to point 7
R_{13}	calculated	distance from point 1 to point 3
R_{67}	calculated	distance from point 6 to point 7
R_{ACT}	calculated	length of the actuator with piston centred
R_{EXT}	calculated	extension of the piston in the actuator
α_1	0.2830 rad	angle between the line joining points 0 and 1, and the negative extension of the line joining points 0 and 6
α_2	0.3045 rad	angle between the line joining points 0 and 2, and the negative extension of the line joining points 0 and 6
α_3	0.0932 rad	angle between the line joining points 4 and 5, and the face of the wave board
α_4	0.0616 rad	angle between the X axis and the line joining points 0 and 7
θ	calculated	angle between the line joining points 0 and 6 for any arbitrary extension of the actuator, and the line joining points 0 and 6 when the actuator piston is centred
θ_0	calculated	angle between the Y axis and the line joining points 0 and 6 when the actuator piston is centred
β	calculated	angle between the X axis and the extension of the line joining points 6 and 7
ϕ	calculated	angle between the Y axis and the front of the wave board
ρ	calculated	angle between the glider push rod and the x axis

being calculated. This is necessary since the actual measurements can vary depending on how the parts are assembled. The methods used to calculate the actual lengths are described later in the report.

The equations to calculate the angle β , the extension of the actuator, the position of the glider, and the ratio of the force in the x direction measured at the glider to the force applied along the axis of the actuator, are given below. The equation for the angle of the wave board with respect to the y axis is not presented since the motion of the top of the wave board with respect to the hinge point of the wave board is minimal and assumed to be zero. The length of the wave board push rod has been calculated to make the wave board angle, ϕ , zero when the lever arm angle is zero.

The angle β can be found by solving the equation:

$$\beta = \tan^{-1} \frac{R_6 \cos(\theta + \theta_0) - R_7 \sin \alpha_4}{R_6 \sin(\theta + \theta_0) + R_7 \cos \alpha_4} \quad (2.3)$$

where R_6 and R_7 are known, θ is measured and θ_0 is a constant to be specified.

The extension of the actuator, R_{EXT} can be found by solving the equation:

$$R_{EXT} = [(-R_0 \cos(\theta + \theta_0) - x_7)^2 + (R_0 \cos(\theta + \theta_0) - y_7)^2]^{1/2} - R_{ACT} \quad (2.4)$$

where R_0 , x_7 and y_7 are known, θ is measured, and θ_0 and R_{ACT} are constants to be specified.

The x coordinate of the top of the glider, x_3 , can be calculated by solving the equation:

$$x_3 = [R_{13}^2 - (y_3 + R_1 \cos(\theta + \theta_0 + \alpha_1))^2]^{1/2} + R_1 \sin(\theta + \theta_0 + \alpha_1) \quad (2.5)$$

where R_1 , y_3 and α_1 are known, θ is measured, and θ_0 and R_{13} are constants to be specified.

The ratio of the force measured in the x direction at the glider to the force applied along the axis of the actuator can be found by solving the equation:

$$\text{force ratio} = \frac{R_0 \cos(\theta + \theta_0 + \beta) \cos(\rho)}{R_1 \cos(\theta + \theta_0 + \alpha_1 - \rho)} \quad (2.6)$$

where R_0 and R_1 are known, θ is measured, θ_0 is a constant to be specified, β is calculated using equation 2.3 and ρ is calculated from the equation:

$$\rho = \tan^{-1} \frac{y_3 - y_1}{x_3 - x_1} \quad (2.7)$$

In the above equations θ_0 , R_{ACT} and R_{13} are not immediately known since the lengths R_{ACT} and R_{13} vary depending on the depth of insertion of the bearings placed at the ends of the actuator and the glider push rod. The variation in the lengths is small but significant when used to compensate the acceleration measurements made using accelerometers. The least squared error procedure used to determine the unknown length

provides best estimates for the angle θ_0 and the length R_{1s} . The length R_{ACT} can be calculated from equation 2.4 using the best estimate value for θ_0 and setting both θ and R_{EXT} to zero.

2.3 DERIVATION OF DIFFERENTIAL EQUATIONS

The system to be modeled consists of an electrohydraulic flow control servo valve, an hydraulic actuator and the mechanical subsystem of the wave generator. An unlimited supply of pressurized hydraulic fluid is assumed to be available for use. The water surrounding the wave generator is treated as an added mass and added viscous damping to the structural components.

The mathematical model was developed by reducing the system into a network of simpler idealized components. Each component is represented by a terminal graph and one or more terminal equations. The terminal graph identifies the points at which the component is allowed to interact with other components; the terminal equations provide the relationship between the variables defined in the terminal graph of the component [ref. 5]. The analysis of each of the three main components and the equations derived from the analysis follow below.

2.3.1 Servo Valve

The servo valve is an electrohydraulic device which controls the flow of pressurized hydraulic fluid such that the flow rate is proportional to the electrical current input. Technical information

supplied by Moog [ref 7], indicates that the series 62 servo valve used in the system can be described by a first order linear differential equation of the form:

$$\frac{dq(t)}{dt} = -\frac{1}{\tau} q(t) + \frac{k_q}{\tau} i(t) \quad (2.8)$$

where $q(t)$ is the flow through the servo valve, $i(t)$ is the drive current input to the servo valve, k_q is the no load flow gain and τ is the servo valve time constant.

2.3.2 Hydraulic Actuator

A simplified drawing of the actuator is shown in Figure 2.3. The two input ports, the piston and the piston rod are shown in the figure. The input ports are connected to the servo valve. High pressure hydraulic fluid flows through the servo valve and into the actuator through one of the input ports. Low pressure fluid returns from the actuator through the servo valve from the other input port. The pressure differential across the piston causes the piston to move in the direction of the pressure gradient. The direction of the pressure gradient is controlled by the servo valve.

The actuator terminal graph and terminal equations are derived by decomposing the actuator into an hydraulic subcomponent and a mechanical subcomponent. The desired terminal graph of the hydraulic subcomponent is a two point graph with a terminal equation which specifies the relationship between the hydraulic variables and the velocity of the

Figure 2.3 Hydraulic Actuator

Figure 2.4 Hydraulic Subcomponent of the Actuator

Figure 2.5 Mechanical Subcomponent of the Actuator

actuator piston. The terminal graph and terminal equation for the mechanical subcomponent gives the relationship between the mechanical variables and the force available from the pressure of the hydraulic fluid acting on the actuator.

2.3.2.1 Hydraulic Subcomponent

The hydraulic variables used in the model are the differential pressure and differential flow measured across the input ports of the actuator. The differential flow is assumed to consist of the flow caused by the motion of the piston, the flow due to the compressibility of the hydraulic fluid and the flow due to the leakage of fluid from one chamber to the other. The terminal graph of the desired model of the hydraulic subcomponent of the actuator is shown in Figure 2.4a.

The derivation of the terminal equations for the terminal graph in Figure 2.4a is a three step process. In the first step, the relationship between the modeled flows and pressures and the driver flows and pressures are developed from the terminal graph shown in Figure 2.4c. A driver is used to provide a set of desired test conditions for which the response of the component can be measured. The second step is to use the terminal graph shown in Figure 2.4b to compute the equations linking the driver flows and pressures and those of the desired model. In the final step the driver flows and pressures are eliminated from the terminal equations for Figure 2.4c using the terminal equations from Figure 2.4b. The resulting set of equations are the desired terminal equations. The

complete procedure as applied to the hydraulic subcomponent is shown below.

In Figure 2.4c, the graph edge labelled 3 represents the flow of fluid caused by a leakage from one side of the piston to the other. Edge 4 in the figure represents the flow of fluid caused by the piston moving. The flow due to the compressibility of the hydraulic fluid is represented by edge 5. The terminal equations for the edges are given below.

$$q_3 = C_L p_3 \quad (2.9)$$

$$q_4 = A \frac{d\delta}{dt} \quad (2.10)$$

$$q_5 = - \frac{C}{2} \frac{dp_5}{dt} \quad (2.11)$$

where C_L is the leakage coefficient, A is the cross sectional area of the piston excluding the area of the piston rod, δ is the displacement of the piston and C is the compressibility constant defined by the volume of oil under pressure divided by the bulk modulus of the oil.

The concepts of a tree, a fundamental cutset and a fundamental circuit are important in a graph theoretical analysis of a system or component. A tree is a subset of edges such that all nodes in a graph are connected to the tree and there are no circuits in the tree. In Figure 2.4c any one of the edges in the graph could be considered as a tree since both nodes are connected by any edge. A fundamental cutset is the set of edges of a graph which are cut by drawing a closed line through the graph cutting exactly one edge of the tree. A fundamental cutset for

the graph in Figure 2.4 would be defined by a closed line around either of the nodes. All edges in the graph including the single edge in the tree would be cut by the closed line. Any circuit in a graph which contains exactly one edge which is not in the tree is a fundamental circuit. Using the driver as the tree in the graph shown in Figure 2.4, any edge plus the driver is a fundamental circuit.

The fundamental cutset equation states that the sum of the flows into a cutset must be zero. The fundamental circuit equation states that the sum of the across variables around a fundamental circuit must be zero. Kirchhoff's current and voltage laws are equivalent to the fundamental cutset and fundamental circuit equations for electrical networks.

Using the definitions above and designating the driver as the tree in Figure 2.4c, the fundamental cutset and fundamental circuit equations are:

$$q_a + q_3 + q_4 + q_5 = 0 \quad (2.12)$$

$$\text{and } p_a - p_3 - p_4 - p_5 = 0 \quad (2.13)$$

Substituting the terminal equations for the edges into the cutset equation and using the circuit equation to express all of the pressures in terms of the driver pressures (p_a), the equation for the driver becomes:

$$q_a = -C_L p_a - \frac{C}{2} \frac{dp_a}{dt} - A \frac{d\delta}{dt} \quad (2.14)$$

To relate the above equation to the terminal equation of the desired model the relations

$$q_1 = -q_a \quad (2.15)$$

$$\text{and } p_1 = p_a \quad (2.16)$$

from Figure 2.4b are used to eliminate q_a and p_a from equation 2.14 resulting in the final terminal equation:

$$q_1 = C_L p_1 + \frac{C}{2} \frac{dp_1}{dt} + A \frac{d\delta}{dt} \quad (2.17)$$

2.3.2.2 Mechanical Subcomponent

The mechanical variables in the model are the force available from the piston and the displacement of the piston. The force and displacement are measured with respect to a fixed point of reference, g , with the orientation as shown in Figure 2.3. The force available from the actuator is assumed to be affected by the force due to the differential pressure across the piston, the inertial force of the piston mass and the damping force caused by the piston seal friction. The terminal graph of the desired model for the mechanical subcomponent of the actuator is shown in Figure 2.5a.

The terminal equations for the terminal graph are developed using the same procedure as was used for the hydraulic subcomponent terminal equations. Starting with the graph shown in Figure 2.5c the relationship between the driver forces and displacements is developed. The equations

defining the relationship between the driver forces and displacements and those of the desired model are derived from the terminal graph shown in Figure 2.5b. The final desired terminal equations are obtained by eliminating the driver forces and displacements from the terminal equations for the terminal graph in Figure 2.5c. The steps leading to the final terminal equations are shown below.

In Figure 2.5c, the graph edge labelled 6 represents the hydraulic force caused by a differential pressure across the piston. Edge 7 represents the inertial force required to accelerate the mass, and the force due to the viscous damping caused by the piston seal friction is represented by edge 8. The terminal equations for the edges are given below.

$$f_6 = AP \quad (2.18)$$

$$f_7 = M_p \frac{d^2 \delta_7}{dt^2} \quad (2.19)$$

$$f_8 = B_p \frac{d\delta_8}{dt} \quad (2.20)$$

where A is the cross sectional area of the piston excluding the area of the piston rod, P is the differential pressure across the piston, M_p is the mass of the piston, δ_7 and δ_8 are the displacement of the piston, and B_p is the seal friction.

Using the driver as the tree, the fundamental cutset and circuit equations are:

$$f_b - f_6 + f_7 + f_8 = 0 \quad (2.21)$$

and

$$\delta_b = \delta_7 = \delta_8 \quad (2.22)$$

Substituting the terminal equations for the edges into the cutset equation and using the circuit equation to express all displacements in terms of the driver displacements (δ_b), the equation for the driver becomes:

$$f_b = AP - M_p \frac{d^2 \delta_b}{dt^2} - B_p \frac{d\delta_b}{dt} \quad (2.23)$$

To relate the above equation to the terminal equation of the desired model, the relations

$$f_2 = -f_b \quad (2.24)$$

$$\text{and } \delta_2 = \delta_b \quad (2.25)$$

from Figure 2.5b are used to eliminate f_b and δ_b resulting in the desired final equation:

$$f_2 = -AP + M_p \frac{d^2 \delta_2}{dt^2} + B_p \frac{d\delta_2}{dt} \quad (2.26)$$

2.3.3 Mechanical Subsystem

The mechanical subsystem of the wave generator is modeled as a combination of a lever arm, a spring, a mass and a damper. The lever arm connects the actuator to the glider and wave board push rods. The lever arm in the model is assumed to have no mass, no friction and to be of infinite stiffness. The elasticity of the actual lever arm and push rods is represented by a spring attached to the lower end of the lever arm.

The combined effect of the mass of the glider, the wave board and the added mass of the water affected by the wave board is modeled as a single mass. The added hydrodynamic damping of the water and any additional friction in the system is modeled as a single viscous damper. The simplified, idealized model is shown in Figure 2.6. The terminal equations for the mechanical components are developed below.

2.3.3.1 Lever Arm

The schematic and terminal graph for a lever are shown in Figure 2.7. The idealized model for a massless, frictionless lever corresponding to the figure has terminal equations given by:

$$\begin{bmatrix} f_1 \\ \delta_2 \end{bmatrix} = \begin{bmatrix} 0 & l_2/l_1 \\ -l_2/l_1 & 0 \end{bmatrix} \begin{bmatrix} \delta_1 \\ f_2 \end{bmatrix} \quad (2.27)$$

In the wave generator, the lever arm terminal equations must specify the relationship between the force and displacement of the actuator piston, measured along the axis of the piston, to the force and displacement of the glider/wave board combination, measured along the x axis specified in Figure 2.2. Due to the geometry of the system, this relationship is not as simple as that specified in the idealized model of the lever arm equation above. However, it is assumed that a linear model of the lever arm can be developed with terminal equations of the form:

$$\begin{bmatrix} f_1 \\ \delta_2 \end{bmatrix} = \begin{bmatrix} 0 & F_A \\ -D_A & 0 \end{bmatrix} \begin{bmatrix} \delta_1 \\ f_2 \end{bmatrix} \quad (2.28)$$

Figure 2.6 Simplified Wave Generator Model

(a) SCHEMATIC

(b) TERMINAL GRAPH

Figure 2.7 Ideal Lever Arm

Figure 2.8 Terminal Graph of Simplified Wave Generator Model

where δ_1 is the extension of the actuator, f_1 is the force applied along the axis of the actuator, δ_2 is the displacement of the glider relative to its position when the lever arm angle is zero degrees, f_2 is the force applied to the glider as measured in the x direction, F_A is the force amplification factor and D_A is the displacement amplification factor.

2.3.3.2 Spring

The lever arm and push rods deform under large loads. This deformation is modeled as a linear spring attached to the bottom of the lever arm. The terminal equation for the spring is:

$$f_s = k_s \delta_s \quad (2.29)$$

where f_s is the force required to deform a spring with a spring constant of k_s by an amount δ_s .

2.3.3.3 Mass

The mass shown in Figure 2.6 represents the combined mass of the glider, the wave board and any water affected by the action of the wave board. The mass is assumed to be constant even though the added mass of the water due to the wave board motion is a function of the water depth and of the frequency of the generated waves. The terminal equation for the mass is:

$$f_m = M \frac{d^2 \delta}{dt^2} \quad (2.30)$$

where f_m is the force required to accelerate a mass of M by $d^2 \delta / dt^2$.

2.3.3.4 Damper

The damper shown in Figure 2.6 represents the combined viscous damping of the water affected by the action of the wave board and any other damping in the mechanical subsystem of the wave generator. As with the mass, the damping is assumed to be constant even though the added damping of the water due to the wave board motion is a function of the frequency of the generated waves. The terminal equation for the damper is:

$$f_b = B \frac{d\delta}{dt} \quad (2.31)$$

where f_b is the damping force experienced while moving at a velocity of $d\delta/dt$ in a medium with a viscous damping coefficient B .

2.3.3.5 Model of the Mechanical Subsystem

The terminal graph corresponding to the schematic of mechanical subsystem shown in Figure 2.6 is shown in Figure 2.8. Edge 1 in the figure represents the terminal equation for the mechanical components in the actuator. The lever arm is represented by graph edges 2 and 3, edge 4 represents the spring, edge 5 represents the load mass and the load damping is represented by edge 6. The tree defined in the graph is shown using heavier lines in the figure.

The tree identifies the primary and secondary variables in the graph. Primary variables are tree across variables (displacement) and cotree through variables (force); secondary variables are cotree across

variables and tree through variables. Using this definition the primary variables are δ_1 , δ_3 , δ_5 , f_2 , f_4 and f_6 ; the secondary variables are δ_2 , δ_4 , δ_6 , f_1 , f_3 and f_5 . The fundamental cutset equations, shown below, define the relationship between secondary through variables and primary through variables. In matrix form, the fundamental cutset equations are:

$$\begin{bmatrix} f_1 \\ f_3 \\ f_5 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} f_2 \\ f_4 \\ f_6 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \quad (2.32)$$

Similarly, the fundamental circuit equations, shown below, define the relationship between secondary across variables and primary across variables.

$$\begin{bmatrix} \delta_2 \\ \delta_4 \\ \delta_6 \end{bmatrix} + \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 1 \\ 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} \delta_1 \\ \delta_3 \\ \delta_5 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \quad (2.33)$$

The terminal equations for the edges are:

$$\text{actuator:} \quad f_1 = -AP + M_a \frac{d^2 \delta_1}{dt^2} + B_a \frac{d\delta_1}{dt} \quad (2.34)$$

$$\text{lever arm:} \quad \begin{bmatrix} f_2 \\ \delta_3 \end{bmatrix} = \begin{bmatrix} 0 & F_A \\ -D_A & 0 \end{bmatrix} \begin{bmatrix} \delta_2 \\ f_3 \end{bmatrix} \quad (2.35)$$

$$\text{spring:} \quad f_4 = k_s \delta_4 \quad (2.36)$$

$$\text{load mass:} \quad f_5 = M_l \frac{d^2 \delta_5}{dt^2} \quad (2.37)$$

$$\text{load damping:} \quad f_6 = B_l \frac{d\delta_5}{dt} \quad (2.38)$$

In determining the complete set of equations for the mechanical components, the fundamental cutset and circuit equations, and the terminal equations are used to eliminate all through and across variables other than the across variables appearing as derivatives in the terminal equations. The hydraulic pressure force driver (AP) is also allowed to remain in the final equations.

For the actuator mechanical component terminal equation the process of elimination of all forces and displacements other than AP, δ_1 and δ_5 proceeds as follows.

$$\begin{aligned}
 -AP + M_a \frac{d^2 \delta_1}{dt^2} + B_a \frac{d\delta_1}{dt} &= f_1 \\
 &= -F_A f_3 \\
 &= -F_A f_4 \\
 &= -F_A k_s \delta_4 \\
 &= -F_A k_s (\delta_3 - \delta_5) \\
 &= -F_A k_s (-D_A \delta_2 - \delta_5) \\
 &= -F_A k_s D_A \delta_1 - F_A k_s \delta_5
 \end{aligned}$$

$$\rightarrow M_a \frac{d^2 \delta_1}{dt^2} = AP - B_a \frac{d\delta_1}{dt} - F_A k_s D_A \delta_1 - F_A k_s \delta_5 \quad (2.39)$$

A similar elimination procedure applied to the terminal equation for the load mass results in the equation:

$$M_l \frac{d^2 \delta_5}{dt^2} = -B_l \frac{d\delta_5}{dt} - k_s \delta_5 - D_A k_s \delta_1 \quad (2.40)$$

2.4 CONTINUOUS TIME EQUATIONS

The set of differential equations describing the system can be combined into a set of linear, constant coefficient continuous time equations of the form:

$$\frac{dx(t)}{dt} = Fx(t) + Gu(t) \quad (2.41)$$

$$y(t) = Hx(t) \quad (2.42)$$

where $x(t)$ is a vector of state variables, F is the continuous system matrix, G is the continuous control input matrix, $u(t)$ is the control input, $y(t)$ is the output measurement and H is the continuous system output matrix. State variables are the variables which contain sufficient information about the state of the system to determine the state of the system at any point in time providing that the inputs to the system are known. In the differential equation form shown above, the state variables are those variables which require an initial value in order to integrate to find the value of the variable at some future time t .

The complete set of differential equations derived above are:

$$\frac{dq(t)}{dt} = -\frac{1}{r} q(t) + \frac{k_q}{r} i(t), \quad (2.43)$$

$$\frac{dp(t)}{dt} = -\frac{2}{C} q(t) - \frac{2C_L}{C} p(t) - \frac{2A}{C} \frac{d\delta_a(t)}{dt}, \quad (2.44)$$

$$\frac{d^2\delta_a(t)}{dt^2} = -\frac{A}{M_a} p(t) - \frac{B_a}{M_a} \frac{d\delta_a(t)}{dt} - \frac{F_A k_s D_A}{M_a} \delta_a(t) - \frac{F_A k_s}{M_a} \delta_l(t), \quad (2.45)$$

$$\frac{d^2\delta_l(t)}{dt^2} = -\frac{B_l}{M_l} \frac{d\delta_l(t)}{dt} - \frac{k_s}{M_l} \delta_l(t) - \frac{D_A k_s}{M_l} \delta_a(t). \quad (2.46)$$

where $q(t)$ is the flow through the servo valve, $i(t)$ is the drive current input to the servo valve, k_q is the no load flow gain, r is the servo valve time constant, $p(t)$ is the differential pressure across the

actuator piston, C_L is the coefficient of leakage within the actuator, A is the cross sectional area of the piston excluding the piston rod area, δ_a is the displacement of the actuator piston, C is the compressibility constant of the oil, B_a is the viscous damping coefficient for the actuator, M_a is the mass of the piston in the actuator, D_A is the displacement amplification factor, F_A is the force amplification factor, k_s is the spring constant of the lever arm, δ_l is the displacement of the load, B_l is the load damping and M_l is the mass of the load.

By making the substitutions:

$$\frac{dV_a(t)}{dt} = \frac{d^2\delta_a(t)}{dt^2}, \quad (2.47)$$

$$V_a(t) = \frac{d\delta_a(t)}{dt}, \quad (2.48)$$

$$\frac{dV_l(t)}{dt} = \frac{d^2\delta_l(t)}{dt^2}, \quad (2.49)$$

and

$$V_l(t) = \frac{d\delta_l(t)}{dt} \quad (2.50)$$

where $V_a(t)$ is the velocity of the piston, δ_a is the displacement of the piston, V_l is the velocity of the load and δ_l is the displacement of the load, and reformatting the equations into matrix form the final system of equations is:

$$\begin{bmatrix} \dot{q}(t) \\ \dot{p}(t) \\ \dot{v}_a(t) \\ \dot{\delta}_a(t) \\ \dot{v}_l(t) \\ \dot{\delta}_l(t) \end{bmatrix} = \begin{bmatrix} -\frac{1}{r} & 0 & 0 & 0 & 0 & 0 \\ \frac{2}{C} & -\frac{2C_L}{C} & -\frac{2A}{C} & 0 & 0 & 0 \\ 0 & \frac{A}{M_a} & -\frac{B_a}{M_a} & -\frac{D_A k_s}{F_A M_a} & 0 & -\frac{D_A k_s}{M_a} \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -\frac{D_A k_s}{M_l} & -\frac{B_l}{M_l} & -\frac{k_s}{M_l} \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} q(t) \\ p(t) \\ v_a(t) \\ \delta_a(t) \\ v_l(t) \\ \delta_l(t) \end{bmatrix} + \begin{bmatrix} \frac{k_q}{r} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} i(t) \quad (2.51)$$

The desired vector of state variables, continuous system matrix, continuous control input matrix and control input can be readily identified. By inspection the system is a single input system, with the input being the drive current input to the servo valve.

The outputs from the system are measured using transducers. These transducers output a voltage proportional to the value of the quantity being measured. The transducers used in the system include a differential pressure transducer, used to measure the differential pressure across the actuator piston, and a rotary variable differential transformer (RVDT), used to measure the angular position of the lever arm. The differential pressure transducer directly measures the differential pressure state variable; the RVDT measures lever arm angle which is not a state variable. However, if the piston in the actuator and the portion of the lever arm above the point of rotation of the lever arm are assumed to have infinite stiffness, then the lever arm angle can be considered to be a measurement of the piston position. The continuous time system output matrix is, therefore, given by:

$$\begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} q(t) \\ p(t) \\ V_a(t) \\ \delta_a(t) \\ V_\ell(t) \\ \delta_\ell(t) \end{bmatrix} \quad (2.52)$$

The elements in the output matrix which are set to 1 imply that the associated state variable measurement can be made exactly. This is not the case. The output from the transducers pass through a filter network to remove high frequency noise from the signal. These filter networks introduce some phase distortion and amplitude attenuation at frequencies in the range of interest.

The differential pressure signal passes through a 3rd order Butterworth filter with a corner frequency of 500 Hz. The RVDT also passes through a 3rd order Butterworth filter, but the corner frequency is 50 Hz. For a 2.5 Hz signal the phase lag introduced by the filters are 0.6 degrees for the differential pressure signal filter and 5.7 degrees for the RVDT filter. The amplitude attenuations of both filters is zero at 2.5 Hz. These effects will be considered small enough to assume that they can be ignored and, therefore, the output matrix shown in equation 2.52 will be used.

2.5 MODEL PARAMETER VALUES

All measurements and parameter values used in the mathematical model will be in the SI system of units. Pressures will be specified in

kN/m² rather than N/m². The values for each of the parameters in the continuous time system model presented in the previous section are defined below.

2.5.1 Servo Valve

The Moog model 62-110 servo valve used in the wave generator has a rated flow of 10 US GPM at a pressure of 1000 psi. The electrical coils in the servo valve are arranged in series allowing the input current in the range of +/- 50 ma.

According to the technical specifications given by MOOG [ref. 6], the parameter k_q is defined by the equation:

$$K_q = \frac{(\text{rated flow at 1000 psi})}{(\text{rated current})} \sqrt{\frac{P_s}{1000}} \quad (2.53)$$

where P_s is the supply pressure. Using the rated flow of 10 US GPM at a supply pressure of 1000 psi, a rated current of 50 ma and an operating supply pressure of 3000 psi, the value for K_q is 0.3464 GPM/ma or 2.185×10^{-5} m³/sec/ma.

The data sheet also indicates that the equivalent first order time constant for the servo valve is 0.024 sec at a supply pressure of 2000 psi. Moog technical data sheet 622 482 [ref. 8], presents a figure which indicates that the 90 degree phase shift frequency at 3000 psi is approximately 1.11 times the 90 degree phase shift frequency at 2000 psi. Using the phase angle equation:

$$\phi = -\tan^{-1}(r\omega) \quad (2.54)$$

the relationship between the time constant at 3000 psi and the time constant at 2000 psi is

$$\begin{aligned} \tau_{3000} &= \tau_{2000} \omega_{2000} / \omega_{3000} \\ &= 0.024 / 1.11 \\ &= 0.022 \end{aligned} \quad (2.55)$$

The time constant of $\tau = 0.022$ seconds will be used in the model.

2.5.2 Hydraulic Actuator

The parameter values to be specified for the actuator are the cross sectional area of the piston, A , the compressibility of the oil, C , the coefficient of leakage, C_L , the mass of the piston, M_a , and the piston seal friction, B_a . Each of these parameters is discussed below.

The cross sectional area of the piston is the cross sectional area of the actuator bore minus the cross sectional area of the piston rod. The bore diameter is 2.5 inches and the diameter of the piston rod is 1.375 inches. Using these dimensions the cross sectional area parameter A is 3.4238 in² or 2.209×10^{-3} m².

The compressibility of the oil is the volume of oil in the actuator divided by the bulk modulus of the oil. Using a cross sectional area of 3.4238 in² and a stroke of +/- 4.25 inches, the volume of oil in the actuator is 29.103 in³ or 4.769×10^{-4} m³. Since the volume of oil under pressure changes, one half of the volume of the actuator is used as the volume of oil under pressure. This is a simplification which will be true

on average only, since as the piston moves the volume of oil under pressure changes. The bulk modulus of the hydraulic fluid used is approximately 1.25×10^5 psi or 8.613×10^5 kN/m². The value for the compressibility constant C is 2.769×10^{-10} m⁵/kN.

The coefficient of leakage represents the resistance to the flow of hydraulic fluid from the high pressure port to the low pressure port. The leakage flow given in the manufacturer's specifications is the leakage in the servo valve. This leakage of 0.45 US GPM for a supply pressure of 1000 psi is used as the leakage in the actuator. Using these figures the leakage coefficient is 4.12×10^{-9} m⁵/kNsec.

The mass of the actuator piston used in the wave generator is not given in the technical specifications of the actuator. However, a similar sized actuator with a shorter piston had a stated mass of 1.75 kg. The combined mass of the piston and the piston rod is, therefore, assumed to be at least 5.0 kg. A more appropriate figure for the mass of the piston will be determined during the process of refining the mathematical model. This figure will likely be higher as a portion of the mass of the lever arm may have to be included in the mass of the piston.

The damping coefficient of the actuator being used was not available. However, a similar actuator with a cylinder bore of 2.5 inches had a stated piston friction force of 1200 lbf at a supply pressure of 2000 psi. Given that this frictional force is observed with no load, it is assumed that the piston was traveling at the maximum velocity.

possible. For a piston cross sectional area of $2.209 \times 10^{-3} \text{ m}^2$ and a maximum flow rate of $6.309 \times 10^{-4} \text{ m}^3/\text{sec}$ the maximum velocity of the piston is 0.2856 m/sec . For the stated frictional force of $5.3376 \times 10^3 \text{ N}$ the damping coefficient is $1.8689 \times 10^4 \text{ Nsec/m}$.

2.5.3 Mechanical Subsystem

The parameters to be specified for the mechanical subsystem of the wave generator are the displacement and force amplification factors for the lever arm, the spring constant of the spring, the mass of the load and the damping of the load. Since the mass and damping caused by the water are functions of water depth and the frequency of generated waves, a water depth of 1.5 meters and a frequency of 0.45 Hz is assumed. The water depth and frequency both represent typical values used in the generation of waves in the NRC Hydraulics Laboratory.

2.5.3.1 Lever Arm

The displacement amplification factor, D_A , was calculated as the average ratio of the displacement of the glider to the displacement of the actuator piston for various lever arm angles. The curve of the displacement amplification factor versus lever arm angle is shown in Figure 2.9. The force amplification factor, F_A , was assumed to be the average ratio of the horizontal force at the glider to the axial force at the actuator for a range of lever arm angles. The force amplification factor versus lever arm angle curve is shown in Figure 2.10. Although, both curves are nonlinear, a constant displacement amplification factor

of 4.1093, was used. The average force amplification factor from Figure 2.10 is 0.2438, but since D_A and F_A must be inverses of each other a value of $F_A = 1/D_A = 0.24335$ is used.

2.5.3.2 Spring

The spring constant in the mathematical model represents the stiffness of the structural components between the top of the lever arm and the wave board. A finite element model of the mechanical components of the wave generator done by Davis Engineering Ltd. determined that the spring constant is approximately 12219 lbf/inch or 2.14×10^6 N/m.

2.5.3.3 Mass

The mass term represents the combined mass of at least part of the lever arm, the glider, the wave board and the added mass of the water. The combined mass of the glider and the wave board is 350 kg. The added mass caused by the generation of a 0.45 Hz wave in a water depth of 1.5 m is approximately 50 kg [ref. 4]. Therefore, the value used for the load mass in the mathematical model is at least 400 kg. Other values for mass will be tested during the verification of the model.

Figure 2.9 Displacement Amplification Factor

Figure 2.10 Force Amplification Factor

2.5.3.4 Damper

The damping term represents the combined viscous damping in moving the lever arm and the glider, as well as the added damping of the water. The added damping caused by the generation of a 0.45 Hz wave in a water depth of 1.5 m is approximately 4500 Nsec/m [ref. 4]. The viscous damping due to friction in the glider and in the lever arm is not known. The combined effect of all viscous damping sources is above 4500 Nsec/m. Other values above 4500 will be tested during the verification of the model.

2.6 DISCRETE TIME EQUATIONS

The actual system to be modeled includes the continuous time model developed above, a discrete time input to the system and the discrete time sampled output from the system. The system input is the output from a digital to analog converter (D/A) which is updated at discrete points in time. The input comes from the finite set of inputs to the system available from the D/A converter. The outputs from the system are the sampled continuous time outputs. The process is shown in Figure 2.11.

Figure 2.11 System and Sampling Operations

The output from the D/A converter is constant between update times. This is modeled as a zero order hold circuit in which the output is constant from time nT to time $nT+T$, where T is the time between samples. The time taken to change the output of the D/A converter is considered to be zero. The output of the system is measured using A/D converters which sample the output at discrete points in time. It is assumed that the samples can be measured instantaneously and that the input to the system and the sampling of the system outputs take place simultaneously.

The solution to the continuous time equations:

$$\frac{dx(t)}{dt} = Fx(t) + Gu(t) \quad (2.56)$$

is given by:

$$x(t) = e^{F(t-t_0)}x(t_0) + \int_{t_0}^t e^{F(t-\varphi)}Gu(\varphi)d\varphi \quad (2.57)$$

By substituting $nT + T$ for t and nT for t_0 the equation becomes:

$$x(nT+T) = e^{FT}x(nT) + \int_{nT}^{nT+T} e^{F(nT+T-\varphi)}Gu(\varphi)d\varphi \quad (2.58)$$

Realizing that the input to the system:

$$u(\varphi) = u(nT) \text{ for } nT \leq \varphi \leq nT + T \quad (2.59)$$

and making the substitution:

$$\eta = nT + T - \varphi \quad (2.60)$$

in equation 2.58 leads to the equation:

$$x(nT+T) = e^{FT}x(nT) + \int_0^T e^{F\eta}Gu(nT)d\eta \quad (2.61)$$

$$\rightarrow x(nT+T) = \Phi x(nT) + \Gamma u(nT) \quad (2.62)$$

where

$$\Phi = e^{FT} \quad (2.63)$$

and

$$\Gamma = \int_0^T e^{F\eta}Gd\eta \quad (2.64)$$

The sampling speed for the wave generator control system is 100 Hz. The input D/A converters and the output A/D converters both have a 12 bit resolution. Any affect on the system due to the resolution of the converters or the filtering of the signal from the D/A converter is ignored. The 100 Hz small amplitude dither signal added to the servo valve drive signal to prevent silting in the servo valve is also ignored.

Assuming the model parameter values defined above, the continuous system matrix is:

$$\begin{bmatrix} -.455e+02 & .000e+00 & .000e+00 & .000e+00 & .000e+00 & .000e+00 \\ .722e+10 & -.298e+02 & -.160e+08 & .000e+00 & .000e+00 & .000e+00 \\ .000e+00 & .442e+00 & -.374e+04 & -.723e+07 & .000e+00 & -.176e+07 \\ .000e+00 & .000e+00 & .100e+01 & .000e+00 & .000e+00 & .000e+00 \\ .000e+00 & .000e+00 & .000e+00 & -.220e+05 & -.113e+02 & -.535e+04 \\ .000e+00 & .000e+00 & .000e+00 & .000e+00 & .100e+01 & .000e+00 \end{bmatrix} \quad (2.65)$$

and the continuous control input matrix is:

$$\begin{bmatrix} .9932e-03 \\ .0000e+00 \\ .0000e+00 \\ .0000e+00 \\ .0000e+00 \\ .0000e+00 \end{bmatrix} \quad (2.66)$$

The system matrix of the discrete equivalent is:

$$\begin{bmatrix} .635e+00 & .000e+00 & .000e+00 & .000e+00 & .000e+00 & .000e+00 \\ .267e+08 & .334e+00 & .187e+04 & .514e+07 & .158e+05 & .193e+07 \\ .106e+03 & .445e-04 & .641e+00 & -.252e+03 & -.890e-01 & .266e+03 \\ .168e+01 & .330e-07 & -.115e-03 & .548e+00 & -.115e-02 & -.153e+00 \\ -.180e+03 & -.546e-05 & -.108e-02 & -.937e+02 & .773e+00 & -.226e+02 \\ -.624e+00 & -.289e-07 & -.148e-04 & -.516e+00 & .903e-02 & .874e+00 \end{bmatrix} \quad (2.67)$$

and the discrete control input matrix is:

$$\begin{bmatrix} .7981e-05 \\ .1526e+03 \\ .1669e-02 \\ .8832e-05 \\ .6199e-03 \\ .1566e-05 \end{bmatrix} \quad (2.68)$$

The data in equations 2.65 to 2.68 were generated by the procedure DDISCR_6. The procedure, which is included in Appendix A, uses the method outlined in this section to compute the discrete time system from the continuous time system.

The eigenvalues of the continuous time system matrix, given in equation 2.65, are shown in Table 2.2. The eigenvalues show that the system has two oscillatory modes and two non-oscillatory modes. The frequency of oscillation of the first pair of eigenvalues is very high

and the large real part of the eigenvalues indicate that the oscillations will die out quickly. It is, therefore, logical to consider reducing the order of the system model to ignore the high frequency oscillatory mode. A simplified fourth order system can be constructed, without the high frequency mode, by ignoring the mass and damping in the actuator. The compliance of the oil column is modeled, but is included with the spring in the mechanical subsystem. Unfortunately, the simplified system is not observable. In order to maintain observability, as is required to construct an estimator, the sixth order model was selected over the fourth order model.

Table 2.2 Eigenvalues of the Sixth Order System

-0.1877e+4 - 0.3288e+4i
-0.1877e+4 + 0.3288e+4i
-0.4546e+2 + 0.01
0.4589e-3 + 0.01
-0.1348e+2 - 0.5158e+2i
-0.1348e+2 + 0.5158e+2i

Many assumptions were made during the development of the mathematical model. A list of the assumptions in the order in which they appear in the report is given below.

- 1) There is no motion of the wave board relative to the glider.
- 2) The system is a linear, lumped parameter system which can be described using ordinary differential equations.

- 3) The servo valve can be modeled using a first order linear differential equation.
- 4) The compressibility of the hydraulic fluid is constant at all piston positions.
- 5) There is no drop in pressure caused by the servo valve.
- 6) An infinite supply of pressurized hydraulic fluid is available.
- 7) The lever arm can be modeled as a simple lever even though the displacement amplification and force amplification vary with respect to the lever arm angle.
- 8) The added mass and added damping of the water are constant.
- 9) Differential pressure and angular position can be measured instantaneously and without error.
- 10) The pressure drop across the piston is equal to the supply pressure.
- 11) The coefficient of leakage is constant and the value assumed for the coefficient is correct.
- 12) The combined mass of the piston and piston rod is at least 5.0 kg.
- 13) The damping coefficient of the actuator used is the same as that specified for a similar actuator.
- 14) The spring constant of the mechanical subsystem is 2.14×10^6 N/m.
- 15) The control output and sampling operations are performed simultaneously and instantaneously.

CHAPTER 3

MODEL VERIFICATION AND REFINEMENT

3.1 TEST DATA

The data used to verify and refine the numerical model were obtained by taking measurements while waves were being generated. The data were acquired at the control update rate of 100 Hz using the data acquisition capability of the digital control system. Each test consisted of specifying the time series of desired lever arm angles, known as a drive signal, transferring the drive signal to the control computer, selecting the measurements to be acquired and recording the data for the duration of the test. Following each test the data were transferred to a general purpose computer for future analysis.

Data from tests using three types of drive signals were collected. These were regular, irregular and square wave drive signals. Regular drive signals were used to produce sinusoidal waves at a single frequency. Irregular drive signals contain energy over a range of frequencies in order to produce waves which have desired spectral and time domain characteristics. The irregular wave drive signals were of the type typically used at the NRC Hydraulics Laboratory for testing offshore structures. The most frequently used irregular wave drive signal was one designed to generate waves with a significant wave height of 20 meters in a water depth of 60 meters with a peak frequency of 0.08 Hz at a model

length scale of 40:1. The square wave drive signal commanded the lever arm angle to follow a square wave pattern. This drive signal was used to create worst case conditions to test the performance of the model. The characteristics of the drive signal files that were used are shown in Table 3.1. The table indicates that the square wave drive signal DQ101 had an amplitude of 5 degrees. However, when this drive signal was used an external span control was used to reduce the amplitude to 1 degree or 3.75 degrees.

Table 3.1 Drive Signal File Characteristics

DRIVE SIGNAL FILE	CHARACTERISTICS
DS201	Freq: 0.1 Hz; Amp: 20 deg; Type: sinusoidal
DS601	Freq: 0.2 Hz; Amp: 20 deg; Type: sinusoidal
DS901	Freq: 0.4 Hz; Amp: 15 deg; Type: sinusoidal
DSA01	Freq: 0.5 Hz; Amp: 12.5 deg; Type: sinusoidal
DSC01	Freq: 0.75 Hz; Amp: 8.5 deg; Type: sinusoidal
DSD01	Freq: 1.0 Hz; Amp: 8.5 deg; Type: sinusoidal
DI201	irregular, peak frequency 0.08 Hz, 20 meter significant wave height
DQ101	Freq: 0.1 Hz; Amp: 5 deg; Type: square wave

In order to measure the response of the system completely, all state variables should have been measured. This would have meant using additional transducers to measure the flow of fluid through the servo valve, the velocity of the actuator, and the position and velocity of the wave board. The instrumentation and signal conditioning circuitry for all of these measurements was not available. However, accelerations of the

actuator and the glider were measured using accelerometers. An accelerometer was not mounted on the wave board due to the small relative motion between the glider and the wave board which resulted in a non zero angle between the wave board and the vertical. This angle, although small, affected the component of gravity measured by the accelerometer. The acceleration of the glider was assumed to be the acceleration of the load mass.

The variables measured were:

- 1) lever arm angle
- 2) differential pressure
- 3) actuator accelerometer output
- 4) glider accelerometer output
- 5) desired lever arm angle
- 6) control signal.

A Penny and Giles Rotary Variable Differential Transformer (RVDT) was used to measure the lever arm angle. The output from the RVDT was passed through a filter and gain network such that the conditioned signal varied between ± 6 volts for ± 20 degree rotations. The sampling resolution of this signal using a 12 bit A/D converter was approximately ± 0.016 degrees.

A differential pressure transducer, built in to the actuator, was used to provide a voltage proportional to the pressure drop across the

actuator piston. After the filter and gain network the voltage signal varied ± 6 volts for ± 20670 kN/m² pressure providing a sampling resolution of ± 16.8 kN/m².

Sundstrand model QA-900 and QA-1400 accelerometers were used to measure the acceleration. The slightly higher resolution QA-1400 model accelerometers were used on the actuators where the accelerations were thought to be smaller. The filter and gain network were set up such that the a ± 10 volt signal would be generated for ± 0.4 g acceleration. The resolution after the A/D converter was $\pm 2.0 \times 10^{-4}$ g or $\pm 1.9 \times 10^{-3}$ m/sec².

The desired lever arm positions at 0.01 second intervals were calculated by interpolating the user defined drive signal which was specified at 0.1 second intervals. Lever arm angles were stored in the control computer in thousandths of a degree.

The control signal is a set of values between ± 5 volts computed by the control computer and output via a D/A converter. The output voltage was passed through a filter and gain network which converted the ± 5 volt signal to ± 50 milliamps of drive current input to the servo valve.

The output of each transducer was carefully calibrated to determine the slope and offset of the best fit straight line describing the relationship between the physical variable and the output from the

transducer, as measured by the A/D converters. The RVDTs were calibrated using a slotted vane and an optical limit switch. As the lever arm, RVDT shaft and slotted vane were rotated, the slots in the vane either broke a beam of infrared light or allowed the beam to be made. The occurrence of breaking a light beam was detected by external circuitry and was used to provide an indication that the lever arm was at one of a number of known angles. The calibration curve was calculated from the resulting set of known angles and measured voltages. The best fit straight line procedure applied to 16 to 20 data points typically showed maximum linearity errors of less than 0.3 degrees and average absolute errors of less than 0.16 degrees.

Differential pressure transducers were calibrated by moving the actuator piston to one of the end stops in the actuator and forcing the servo valve to open fully in the direction forcing the piston into the end stop. Since the piston could not move and the servo valve was fully open, the pressure drop across the piston was the supply pressure of 3000 psi. Repeating the procedure at the other end stop provided two data points from which the best fit straight line was calculated. Since there were only two points used to calculate the line, no linearity errors could be exposed by the calibration.

The accelerometers were calibrated using a machine shop dividing head to precisely set the angle of the accelerometers with respect to the vertical. The component of gravity measured by the accelerometers and the associated voltages were used to determine the calibration curve. To

ensure that any errors due to the base of the dividing head not being perfectly level were eliminated, readings for each angle were repeated after rotating the base of the dividing head by 90, 180, 270 and 360 degrees. Using this technique positive angle errors due to a base tilt were canceled by equal and opposite negative angle errors. The best fit straight line procedure based on 12 angles typically gave maximum errors of less than 1.84×10^{-2} m/sec² and average absolute errors of less than 1.35×10^{-2} m/sec².

3.2 PROCESSING OF MEASURED DATA

The data files acquired from each of the tests were passed through various GEDAP procedures to extract the measured acceleration from the accelerometer output. This process involved low pass filtering the raw accelerometer data to remove high frequency accelerations due to the vibration of the wave generator frame, compensating the accelerometer data and post processing the extracted acceleration data.

The time series of estimated states were computed using equation 2.1 and the measured control input. Estimates of the accelerations were generated by applying a frequency domain differentiation procedure to the estimated velocity time series. The differentiation was limited to frequencies below 10 Hz to ensure that the accelerations were not corrupted by high frequency noise. Post processing of the acceleration data by a low pass filter further reduced the noise content in the differentiated time series.

Before comparison, the measured and estimated accelerations were refiltered using a low pass filter with a cut off frequency of 7.5 Hz. This was done to limit the accelerations to frequencies in the range of interest. Edge effects caused by the finite impulse response filter were eliminated by removing the beginning and ending transients from the data files.

3.2.1 Accelerometer Compensation

Accelerometer data required compensation to remove effects such as acceleration due to gravity, angular acceleration and errors due to improper alignment of the accelerometers. To compute the acceleration of the actuator from the accelerometer output values, the angles σ , β and ϵ shown in Figure 3.1 were required. The angle σ represents the error due to the improper alignment of the axis of the accelerometer with the axis of the actuator; ϵ is the angle between the x axis of the defined coordinate system and the horizontal; β is the angle between the actuator and the x axis and is calculated using equation 2.3. The angle β depends on the angle θ_0 which is not known since the exact length of the actuator is unknown. Assuming that all of σ , ϵ and β are known, the equation relating the accelerometer output to the acceleration of the actuator is given by:

$$\frac{d\delta_a^2}{dt^2} = (\text{accelerometer output} - 9.806\sin(\beta+\sigma-\epsilon))/\cos(\sigma) \quad (3.1)$$

Figure 3.1 Compensation of the Actuator Accelerometer

Figure 3.2 Compensation of the Glider Accelerometer

The equation does not account for the acceleration due to the rate of change of the angle β which is measured by the accelerometer. This acceleration was assumed to be sufficiently small to be ignored.

Figure 3.2 defines the angle γ_G which is the difference between the axis of the accelerometer and the horizontal axis. The equation relating the accelerometer output to the horizontal acceleration of the glider is given by:

$$\frac{d\delta_l^2}{dt^2} = (\text{accelerometer output} + 9.806\sin(\gamma_G))/\cos(\gamma_G) \quad (3.2)$$

Equations 3.1 and 3.2 can not be solved unless values for σ , ϵ , θ_0 and γ_G are known.

To determine the unknown angles, data from an additional test were acquired and analyzed. These data were acquired while the wave generator control system was tracking a low frequency regular drive signal with an amplitude of 20 degrees. The time series of recorded lever arm positions were low pass filtered and the amplitude, phase angle and frequency of the sine wave which best fit the filtered lever arm angle data was computed. The kinematic equations for the calculation of the actuator extension (equation 2.4) and the calculation of the glider position (equation 2.5) were differentiated with respect to time to compute the theoretical acceleration of the actuator and the glider. The time derivatives of the best fit sine wave were used in the theoretical acceleration equations to ensure that any noise in the measured lever arm angles did not corrupt the theoretical accelerations.

Equations 3.1 and 3.2 were rearranged into equations of the error between the theoretical accelerations and the compensated accelerometer output. The partial derivatives of the error equations with respect to the unknown angles were formed and used in a least squared error procedure. The procedure provided values for the unknown angles which were used in the extraction of the acceleration data of all subsequent tests.

3.3 TEST RESULTS

Attempts to compare the estimated actuator position to the position calculated using the measured lever arm angle and equation 2.4 proved to be unsuccessful. The estimated position was found to drift towards infinity. The comparison of the differential pressures was also unsatisfactory. These problems were due to the inaccuracy of the servo valve model. All comparisons between the model output and the measured data were, therefore, limited to the comparison of the actuator accelerations and the glider accelerations.

The difficulty with the component model used for the servo valve is that it does not account for the effects of hysteresis or the non zero control input, called null bias current, required to set the control flow to zero. The control algorithm accounts for these affects by adjusting the input applied to the servo valve. Applying the recorded control signals, which include an offset to compensate for the null bias current,

to the servo valve model causes the resulting flow of hydraulic fluid in the model to be different than the actual flow. This difference in flow causes the piston to move at a velocity which is slightly higher, or lower, than the actual piston velocity. Over time this velocity error causes the actuator position to drift towards infinity. Accelerations are unaffected by the null bias current and the hysteresis since the constant velocity drift disappears when the estimated velocities are differentiated to compute accelerations. The effect of the null bias current was further reduced by subtracting an estimate of the bias from the recorded input before using the input to drive the model. The estimate of the null bias current for each test was taken as the average input current to the servo valve over the duration of the test.

The analysis of the system carried out in the model development section provided fixed values for some of the model parameters and estimates of the others. Initial estimates for the actuator mass, actuator damping, load mass and load damping were 5 kg, 18700 Nsec/m, 400 kg and 4500 Nsec/m respectively. The system matrix and the control input matrix for the discrete time model derived using these values are shown in equations 2.67 and 2.68. The results obtained when driving this model with the inputs recorded for the test using irregular wave drive signal DI201 showed that the model was unstable. Increasing the actuator mass to 10 kg stabilized the model and resulted in an acceptable match between the measured and estimated accelerations.

In order to determine the sensitivity of the model to changes in the mass and damping, various tests were performed. The basic procedure followed was to change values of various parameters in the continuous time mathematical model, convert the model to the discrete time equivalent and compare the output of the model to the measured output of the system. The input used to drive the models was the input recorded while using the irregular wave drive signal. Table 3.2 shows the standard deviations of the differences between the measured and estimated accelerations as well as the values used for the masses and damping terms for the test cases. The results show that the variation in the fit between the measured and estimated accelerations is minimal providing the model value for the actuator mass is at least 10 kg and the load damping is at least 4500 Nsec/m. Typical plots for the actuator accelerations and glider accelerations are shown in Figures 3.3 and 3.4.

Table 3.2 Results Of Model Verification Tests

TEST CASE	ACTUATOR		LOAD		ERROR STANDARD DEVIATION	
	MASS (kg)	DAMPING (Nsec/m)	MASS (kg)	DAMPING (Nsec/m)	ACTUATOR (m/sec ²)	GLIDER (m/sec ²)
1	5	18700	400	4500	unstable	unstable
2	10	18700	400	4500	0.0927	0.4178
3	10	9500	400	4500	0.0940	0.4258
4	10	30000	400	4500	0.0916	0.4146
5	20	9500	400	4500	0.0940	0.4250
6	20	18700	400	4500	0.0927	0.4187
7	10	18700	400	2250	oscillates	oscillates
8	10	18700	400	9000	0.0869	0.3827
9	10	18700	800	4500	0.0911	0.4222
10	10	18700	800	9000	0.0922	0.3772

As an additional test of the model, the inputs recorded for a square wave drive signal were used to drive the discrete time model derived using parameter values shown in test case 8. The plots of the acceleration comparison plots are shown in Figures 3.5 and 3.6. The accelerations comparisons using data from a 0.5 Hz sinusoidal drive signal test are shown in Figures 3.7 and 3.8.

There are an infinite number of possible models which can be obtained by varying the values of the model parameters. The tests done above investigate the performance of the model for only a small number of the possible masses and damping coefficients. The affects of changing other key parameters such as the servo valve time constant could also be investigated. However, it was felt that the match between the open loop estimations of accelerations and the measured accelerations indicated that the mathematical model adequately described the actual system. The development of the estimator can, therefore, proceed using the discrete time model. The specific values used for the actuator mass, actuator damping, load mass and load damping are 10 kg, 18700 Nsec/m, 400 kg and 9000 Nsec/m.

Figure 3.3 Actuator Acceleration for Irregular Wave Drive Signal

Figure 3.4 Glider Acceleration for Irregular Wave Drive Signal

Figure 3.5 Actuator Acceleration for Square Wave Drive Signal

Figure 3.6 Glider Acceleration for Square Wave Drive Signal

Figure 3.7 Actuator Acceleration for 0.5 Hz Sinusoidal Drive Signal

Figure 3.8 Glider Acceleration for 0.5 Hz Sinusoidal Drive Signal

CHAPTER 4

ESTIMATOR DEVELOPMENT4.1 INTRODUCTION

The objective for developing the system model in the previous section was to enable the actual system to be controlled more accurately than would be possible without the model. The improved controller used to control the system would consist of a control law and an estimator. The control law provides the relationship between the system states and the control input used to drive the system. The control law can be described by the equation:

$$u(n) = -Kx(n) \quad (4.1)$$

where $u(n)$ is the control input, K is the feedback gain matrix and $x(n)$ is the vector of state variables. The gain matrix is computed to meet the desired performance specifications as closely as possible.

The control law requires the complete state vector $x(n)$. These states are usually not all available by direct measurement since providing full state feedback by direct measurement is usually not possible or is too costly. For the wave generator being studied only two of the six states identified in the model are being measured. The cost of measuring the other four states for each of the sixty-four wave generators which make up the short crested wave generation facility makes it impossible to even consider direct measurement of all of the states.

For these reasons estimators are used to provide estimates of the state vector which can be used in the control law. Estimators use a model of the system and the state measurements available to form the state vector. The simplest form of an estimator can be described by the equation:

$$\hat{x}(n+1) = \Phi \hat{x}(n) + \Gamma u(n) + L[y(n) - H\hat{x}(n)] \quad (4.2)$$

where $\hat{x}(n)$ is an estimate of the state vector, Φ is the system matrix, Γ is the control input matrix, L is the estimator gain matrix, $y(n)$ is the vector of state measurements and H is the output matrix. Examination of the estimator equation shows that the estimated states are the sum of the open loop estimation and the weighted error between the measured states and the estimated states. The rate at which the error between the actual states and the estimated states decay is determined by the gain matrix L .

The controller implemented using equations 4.1 and 4.2 would use the estimated states $\hat{x}(n+1)$ to compute the control signal $u(n+1)$ which is output at the start of the $n+1$ st time step. All of the information necessary to compute $u(n+1)$ is available at some point during the n th time step. This allows the control signal to be computed before the start of each control interval. This technique has the disadvantage of not using the measurements made during the time step $n+1$ to affect the control signal $u(n+1)$. An alternative estimator which uses the most recent measurements is given by the equation:

$$\hat{x}(n) = \bar{x}(n) + L[y(n) - H\bar{x}(n)] \quad (4.3)$$

$$\text{where } \bar{x}(n) = \Phi\hat{x}(n-1) + \Gamma u(n-1) \quad (4.4)$$

Equation 4.3, known as the measurement update equation, uses the most recent measurement, $y(n)$, to update the estimate of the states generated by the time update equation, equation 4.4, before the measurement $y(n)$ was made. The updated estimate $\hat{x}(n)$ is used in the control law to compute the control signal $u(n)$.

There are various solution techniques available to compute the feedback gain matrix K and the estimator gain matrix L . These techniques typically start with a set of desired pole locations, specified by the designer, and solve for the gain matrix which moves the poles to the desired locations. These techniques are only suitable for single input and single output systems. The feedback and estimation gain matrices for multi-input and multi-output systems are computed using optimization techniques.

4.2 OPTIMAL CONTROL

The solution of the optimal control problem and the optimal estimation problem are very similar in form. The solution of the optimal controller is typically developed first, with the optimal estimator solved by analogy. This approach was adopted in the solution of the steady state estimator.

Optimal controllers are developed by minimizing the value of a 'cost equation'. The cost equation provides a convenient means to define the relative importance of achieving the desired system states and the control effort required to achieve the states. A typical cost equation is given by:

$$J = \sum_{n=0}^N [x(n)^T Q_1 x(n) + u(n)^T Q_2 u(n)] \quad (4.5)$$

where Q_1 is the weighting matrix for the states and Q_2 is the weighting matrix for the control input. If the value of Q_1 relative to Q_2 is large, then the control system designer has placed a large cost on not achieving the desired states and the response of the controller is likely to be quick; large values of Q_2 relative to Q_1 result in a controller which responds more slowly to state errors since the relative cost of control input is high. The weighting matrices Q_1 and Q_2 must be real, positive and symmetric.

Minimization of the cost equation defines the time varying feedback gain matrix $K(n)$ used in the control law:

$$u(n) = -K(n)x(n) \quad (4.6)$$

The feedback gain matrix is computed from the recursive relations:

$$K(n) = R^{-1}(n+1)\Gamma^T S(n+1)\Phi \quad (4.7)$$

$$S(n) = \Phi^T M(n+1)\Phi + Q_1 \quad (4.8)$$

where

$$R(n+1) = Q_2 + \Gamma^T S(n+1)\Gamma \quad (4.9)$$

$$M(n+1) = S(n+1) - S(n+1)\Gamma R^{-1}(n+1)\Gamma^T S(n+1) \quad (4.10)$$

and where the S matrix is subject to the boundary condition

$$S(N) = Q_1 \quad (4.11)$$

Equation 4.8 is solved backward to $n=0$ before equation 4.7 can be used to compute the feedback gain matrix.

Plots of the time varying feedback gains show that they are constant over most of the optimization period N . In many implementations the time varying gain matrix can be replaced by the steady state matrix without significantly affecting the performance of the controller.

The solution of the steady state optimal controller involves diagonalizing the control Hamiltonian:

$$H_c = \begin{bmatrix} \Phi + \Gamma Q_2^{-1} \Gamma^T \Phi^{-T} Q_1 & -\Gamma Q_2^{-1} \Gamma^T \Phi^{-T} \\ -\Phi^{-T} Q_1 & \Phi^{-T} \end{bmatrix} \quad (4.12)$$

The entries along the diagonal of the diagonalized Hamiltonian are the eigenvalues of the original Hamiltonian. These eigenvalues occur in pairs, with the property that the members of each pair are reciprocals of each other. Because of this property one eigenvalue of each pair is a stable root ($|z| < 1$) and the other is an unstable root ($|z| > 1$). The diagonalization of the Hamiltonian is such that the eigenvalues corresponding to the stable roots are placed in the upper left of the diagonal matrix and the eigenvalues corresponding to the unstable roots are placed in the lower right. The Hamiltonian is diagonalized using the matrix W in the similarity transform:

$$H_c^* = W^{-1} H_c W \quad (4.13)$$

The transform matrix W is given by:

$$W = \begin{bmatrix} X_I & X_O \\ \Lambda_I & \Lambda_O \end{bmatrix} \quad (4.14)$$

where

$$\begin{bmatrix} X_I \\ \Lambda_I \end{bmatrix} \quad (4.15)$$

is the matrix of eigenvectors associated with the stable eigenvalues (inside the unit circle) and

$$\begin{bmatrix} X_O \\ \Lambda_O \end{bmatrix} \quad (4.16)$$

is the matrix of eigenvectors associated with the unstable eigenvalues (outside the unit circle).

The constant feedback gain matrix is given by

$$K = (Q_2 + \Gamma^T S_\infty \Gamma)^{-1} \Gamma^T S_\infty \Phi \quad (4.17)$$

where

$$S_\infty = X_I \Lambda_I^{-1} \quad (4.18)$$

See Franklin and Powell [ref. 2] for the complete derivation leading to equations 4.7, 4.8, 4.17 and 4.18.

The procedure SOLVE_SS_OPT_C_EQNS, shown in Appendix B, is used to compute the steady state feedback gain matrix. The procedure forms the control Hamiltonian from the input matrices and computes all of its eigenvalues. The user is asked to select the stable eigenvalues for which the corresponding eigenvectors are computed. The matrix of eigenvectors

and equations 4.17 and 4.18 are used in the computation of the gain matrix K_{∞} .

The complex eigenvalues of the non-symmetric control Hamiltonian are computed by the procedure C_EVAL. This procedure calls BALANC to balance the input matrix, ELMHES to convert the balanced matrix into upper Hessenberg form and HQR to diagonalize the matrix. Balancing uses similarity transforms to reduce the sensitivity of the eigenvalues to rounding errors. The eigenvalues of a matrix are not affected by similarity transforms. The matrix in Hessenberg form is passed to HQR and diagonalized using the QR algorithm. The procedures BALANC, ELMHES and HQR were taken from the Numerical Recipes software package [réf. 9] and modified to perform computations in double precision. The procedures C_EVAL and C_EVECT are included in the program file DPEIGEN.FOR shown in Appendix B.

Once the eigenvalues have been determined, the eigenvectors corresponding to each of the stable eigenvalues is computed by the procedure C_EVECT. This procedure uses the inverse iteration method to compute the eigenvectors. In this method the equation:

$$(A - \lambda I)y = x_i \quad (4.19)$$

is used to iteratively solve for the unknown eigenvector x_i , where A is the input matrix, λ is the estimate of an eigenvalue, x_i is the previous best estimate of the eigenvector and the solution y is the next estimate of the eigenvector. At each iteration x_i is replaced by the new normalized approximation of the eigenvector given by:

$$x_{i+1} = y/|y| \quad (4.20)$$

The procedure is terminated when subsequent approximations of the eigenvector converge.

The solution vector, y , in equation 4.19 is obtained by performing an LU decomposition of the matrix $(A - \lambda I)$ and using back substitution to compute y . The procedures LUDCMP and LUBKSB are used to perform the LU decomposition and back substitution operations, respectively. Both procedures are double precision implementations of the procedures found in the Numerical Recipes software package [ref. 9].

The eigenvectors and eigenvalues of the Hamiltonian may be complex. Since the procedures LUDCMP and LUBKSB can not handle complex numbers the matrix $(A - \lambda I)$ is replaced by:

$$\begin{bmatrix} A - \lambda_r I & \lambda_i I \\ -\lambda_i I & A - \lambda_r I \end{bmatrix} \quad (4.21)$$

and the x_i vector is replaced by:

$$\begin{bmatrix} x_{r1} \\ x_{i1} \end{bmatrix} \quad (4.22)$$

where λ_r and λ_i are the real and imaginary components of λ , and x_{r1} and x_{i1} are the real and imaginary components of x_i . The order of the modified system is twice that of the original, but can be solved using procedures designed for real computations.

Tests done on the procedure showed that the estimates of the eigenvector converged within five iterations. The procedure C_EVECT executes five iterations rather than using the difference between subsequent iterations as the stopping condition.

The real components of the eigenvectors are placed in the first NS columns of the eigenvector matrix and the imaginary components of the eigenvector are placed in the second NS columns, where NS is the order of the system. The eigenvectors are stored in the order specified by the user when the stable eigenvalues were selected. The final form of the eigenvector matrix is:

$$\begin{bmatrix} X_I \\ \Lambda_I \end{bmatrix} = \begin{bmatrix} X_{IR} & X_{II} \\ \Lambda_{IR} & \Lambda_{II} \end{bmatrix} \quad (4.23)$$

where X_{IR} and Λ_{IR} are the real components of the eigenvectors, and X_{II} and Λ_{II} are the imaginary components.

If the eigenvector matrix is transposed the matrices:

$$X_I^T = \begin{bmatrix} X_{IR}^T \\ X_{II}^T \end{bmatrix} \quad (4.24)$$

and

$$\Lambda_I^T = \begin{bmatrix} \Lambda_{IR}^T \\ \Lambda_{II}^T \end{bmatrix} \quad (4.25)$$

can be formed. Since the matrices K_∞ and S_∞ are real, the eigenvectors must appear in complex conjugate pairs. Because of this and the technique of storing the real and imaginary components of the eigenvectors separately, many of the rows of X_I^T and Λ_I^T will be linearly dependent or zero. These matrices will also be singular making the solution of S_∞ using equation 4.18 impossible. The matrix S_∞ is determined by solving the equation:

$$X_I^T S_\infty^T - \Lambda_I^T \quad (4.26)$$

for S_∞^T instead. Where X_I^T and Λ_I^T are of the order $2NS \times NS$ and S_∞^T is of the order $NS \times NS$, where NS is the order of the system. Equation 4.26 is solved by performing single valued decomposition of X_I^T and using back substitution to compute the columns of S_∞^T which are mapped into the corresponding columns of Λ_I^T . The procedures used to perform single valued decomposition and back substitution are SVDCMP and SVBKS, respectively. These procedures are taken from the software package Numerical Recipes [ref. 9].

4.3 OPTIMAL ESTIMATION

Optimization techniques can also be used to develop estimators that are optimal in some sense. One such optimal estimation solution is known as the Kalman filter.

The equations for a discrete system including process noise and measurement noise are:

$$x(n+1) = \Phi x(n) + \Gamma u(n) + \Gamma_1 w(n) \quad (4.27)$$

$$y(n) = Hx(n) + v(n) \quad (4.28)$$

where $w(n)$ is the process noise and $v(n)$ is the measurement noise. Given certain assumptions about the noise, the Kalman filter provides estimates of the states which are optimal. The assumptions concerning the noise random sequences are that they are uncorrelated, have zero autocorrelation and have a mean of zero.

For the system described by equations 4.27 and 4.28 and with the assumptions given above for the noise, the Kalman filter equations are:

$$\hat{x}(n) = \bar{x}(n) + L(n)[y(n) - H\bar{x}(n)] \quad (4.29)$$

$$P(n) = M(n) - L(n)HM(n) \quad (4.30)$$

$$\bar{x}(n+1) = \Phi \hat{x}(n) + \Gamma u(n) \quad (4.31)$$

$$M(n+1) = \Phi P(n) \Phi^T + \Gamma_1 R_w \Gamma_1^T \quad (4.32)$$

$$\text{where } L(n) = M(n)H^T(HM(n)H^T + R_v)^{-1} \quad (4.33)$$

In the above equations $L(n)$ is the time varying estimator gain matrix, $\hat{x}(n)$ is the updated estimate of the states, $\bar{x}(n)$ is the initial state estimate, $y(n)$ is the vector of measured states, H is the output matrix, $P(n)$ is the time varying matrix of the covariance of the estimation errors, $M(n)$ is the time varying matrix of the covariance of the estimated state errors, Φ is the system matrix, Γ is the input matrix, and Γ_1 is the process noise input matrix. The covariance matrices R_w , R_v , M and P are defined by the relations:

$$R_w = E(w w^T), \quad (4.34)$$

$$R_v = E(v v^T), \quad (4.35)$$

$$M = E((x - \bar{x})(x - \bar{x})^T) \quad (4.36)$$

and

$$P = E((\hat{x} - x)(\hat{x} - x)^T) \quad (4.37)$$

Equations 4.31 and 4.32 are known as the time update relations since they are used to update information for the next time step. Equations 4.29 and 4.30 are the measurement update relations used to modify information during the current time step.

Plots of the time varying estimator gains show that after a few time steps the gains approach steady state values. The steady state gain values are usually used instead of the time varying gains in order to reduce the computational effort involved in implementing optimal estimators. The steady state values will be used for the optimal estimator used for the wave generator.

Comparing equation 4.30 with 4.10 and equation 4.32 with 4.8 the duality table shown in Table 4.1 can be derived. Using the substitutions defined in Table 4.1 in the control Hamiltonian the estimation Hamiltonian given below can be created.

$$H_e = \begin{bmatrix} \Phi^T + H^T R_v^{-1} H \Phi^{-1} \Gamma_1 R_w \Gamma_1^T & -H^T R_v^{-1} H \Phi^{-1} \\ -\Phi^{-1} \Gamma_1 R_w \Gamma_1^T & \Phi^{-1} \end{bmatrix} \quad (4.38)$$

By comparison with equation 4.18 the steady state value of M , denoted as M_∞ , is given by:

$$M_\infty = X_I \Lambda_I^{-1} \quad (4.39)$$

and by comparison with equation 4.17 the steady state estimator gain matrix, denoted as L_∞ , is given by:

$$L_\infty = M_\infty H^T (H M_\infty H^T + R_v)^{-1} \quad (4.40)$$

Table 4.1 Control and Estimation Duality

Control	Estimation
Φ	Φ^T
M	P
S	M
Γ	H^T
Q_1	$\Gamma_1 R_w \Gamma_1^T$
Q_2	R_v

The mainline driver for the steady state optimal estimator, called SS_OPT_0, is included in Appendix B. The estimator mainline makes the substitutions indicated in Table 4.1, calls the procedure SOLVE_SS_OPT_C_EQNS to compute the constant feedback gain matrix and then transposes this matrix to form the constant estimator gain matrix desired:

In the steady state estimator the constant gain matrix, L_∞ , is used in the measurement update equation rather than the time varying gain matrix $L(n)$. At the start of a control interval new measurements, $y(n)$, are made. This information along with the estimates of the states from

the previous time step ($\bar{x}(n)$) are used in the constant gain measurement update equation to compute the updated estimates of the system states ($\hat{x}(n)$). The updated estimates are then used in the control law to compute the control signal, $u(n)$, which is output to the system. Immediately following the output of the control signal, the time update equation is used to estimate the system states for the next time step ($\bar{x}(n+1)$). A block diagram of the process is shown in Figure 4.1. The delay element in the figure represents the passing of a control interval.

4.4 SOLUTION OF THE STEADY STATE OPTIMAL ESTIMATOR

The specific solution of the constant estimator gain matrix requires that the system matrix, Φ , the system output matrix, H , the measurement noise covariance matrix, R_v , and the process noise covariance matrix, $\Gamma_1 R_w \Gamma_1^T$ be known. The system matrix and the output matrix were defined in the previous section. Measurements made using the differential pressure transducer and the RVDT showed that the measurement noise covariance for the differential pressure was approximately $1000 \text{ kN}^2/\text{m}^4$ and the noise covariance for the actuator position measurement was approximately 0.0003 m^2 . The matrix R_v is assumed to be diagonal with the differential pressure covariance in the (1,1) entry and the actuator position covariance in the (2,2) entry. The process noise covariance matrix given by $\Gamma_1 R_w \Gamma_1^T$ is more difficult to determine. It is assumed that this matrix is also diagonal with the noise covariances for the states entered along the diagonal. Different values for the process noise covariances were tested to find an acceptable set of values.

Figure 4.1 Block Diagram of System with Estimator and Control Law

The procedure used to determine the covariance matrices used in the estimator was to use the data acquired during the irregular wave test in 1.5 meters of water to compare the performance of the estimator with the measured data for various covariance values. For each estimator, the steady state form of the measurement update equation (equation 4.29) and the time update equation (equation 4.31) were used to estimate the state vector for each time step. The average absolute value of the error between the measured states and the estimated states were computed using the acquired test data set. The estimator with the smallest average absolute errors was assumed to be the best. Table 4.2 shows the error data for the different covariances tested. All of the estimators were developed with the same system model. This model used an actuator mass of 10 kg, actuator damping of 18700 N.sec/m², a load mass of 400 kg and load damping of 9000 N.sec/m².

The data in Table 4.2 show that the performance of the estimator is relatively insensitive to minor changes in the process noise and measurement noise covariances. As the noise covariance for the differential pressure measurement is decreased the average absolute error of the differential pressure estimation decreases, but does so at the expense of an increased actuator position estimation error. The values selected for the covariances based on the above tests were 10 for the process noise covariances, 500 for the differential pressure measurement covariance and 0.0003 for the actuator position measurement covariance.

Figure 4.2 shows the measured and estimated actuator position for the first five seconds of the irregular wave test. The actuator velocity curves shown in Figure 4.3 do not match as well as the position curves. The estimated velocity is compared with the ideal velocity since the actual velocity was not measured. The ideal velocity was obtained by differentiating the measured actuator position time series.

Table 4.2 Estimator Performance Tests

TEST CASE	Process Noise Covariances	Measurement Noise Covariances	Average Absolute Errors	
			Diff Press (kN/m ²)	Actuator Pos. (mm)
1	1,1,1,1,1,1	1000, 0.0003	3233	0.3654
2	10,10,10,10,10,10	1000, 0.0003	3233	0.3653
3	1,1,1,1,1,1	5000, 0.0003	3233	0.3654
4	1,1,1,1,1,1	5000, 0.0003	3233	0.3654
5	10,10,10,10,10,10	500, 0.0003	3233	0.3653
6	1,1,1,1,1,1	500, 0.001	3232	0.3658
7	10,10,10,10,10,10	500, 0.001	3233	0.3653
8	1,1,1,1,1,1	100, 0.001	3232	0.3658
9	1,1,1,1,1,1	100, 0.01	3226	0.3702
10	1,1,1,1,1,1	10, 0.1	3226	0.3702
11	1,1,1,1,1,1	10, 0.1	3160	0.4142
12	1,1,1,1,1,1	1, 0.1	3160	0.4142
13	1,1,1,1,1,1	1, 0.1	2696	0.8090
14	1,1,1,1,1,1	0.1, 0.1	3160	0.4142
15	1,1,1,1,1,1	0.01, 0.01	3226	0.3702
16	1,2,3,4,5,6	500, 0.0003	3345	0.3707
17	10,200,1,5,2,5	500, 0.0003	3223	0.3654

The glider position and velocity curves are shown in Figures 4.4 and 4.5, respectively. In both cases the state variables were not measured and, therefore, the ideal curves are presented. The ideal glider position is calculated from the lever arm angle using only the kinematic equations, ignoring the dynamic affects of the system. The ideal glider

velocity was obtained by differentiating the time series of ideal glider positions.

The differential pressure curves presented in Figure 4.6 demonstrate the estimators inability to accurately estimate the differential pressure.

The measured and estimated accelerations of the actuator and the glider are presented in Figures 4.7 and 4.8, respectively. The estimated accelerations were obtained by differentiating the estimated velocities. The figures show a good match between the estimated and measured accelerations, but the matches achieved using the closed loop estimator are not significantly better than those achieved using the open loop estimator. The acceleration curves for the open loop estimator were presented in Figures 3.3 and 3.4.

4.5 ESTIMATOR AND MODEL REFINEMENT

The motivation for using an estimator is to provide estimates of the state variables which match the actual state variables as closely as possible. Since the performance of the estimator depends largely on the system model, it seems reasonable to consider changing the model to improve the fit of the closed loop estimations. Therefore, alternate system models were considered in an attempt to improve the performance of the estimator. Any changes in the model which improved the estimates were considered to be justified.

Figure 4.2 Actuator Position Comparison for System Model One

Figure 4.3 Actuator Velocity Comparison for System Model One

Figure 4.4 Glider Position Comparison for System Model One

Figure 4.5 Glider Velocity Comparison for System Model One

2

of/de

2

Figure 4.6 Differential Pressure Comparison for System Model One

Figure 4.7 Actuator Acceleration Comparison for System Model One

Figure 4.8 Glider Acceleration Comparison for System Model One

The system model and estimator developed in the previous sections were used as starting conditions in the refinement process. The parameter values for the system model were: an actuator mass of 10 kg, actuator damping of 18700 N.sec/m, a load mass of 400 kg and load damping of 9000 N.sec/m. The diagonal entries of the process noise covariance matrix were all set to 10 and the measurement covariances used for the differential pressure and actuator position were $500 \text{ kN}^2/\text{m}^4$ and 0.0003 m^2 , respectively. All of the estimators developed with modified system models used the covariance matrices with diagonal entries as listed above.

The characteristics of the different system models for which the performance of the estimators were tested are shown in Table 4.3. The estimators were compared on their ability to minimize the difference between the estimated states and the measured states for the data collected from the irregular wave test. For the actuator velocity, glider position and glider velocity state variables the states derived assuming ideal conditions were used as the measured states. The derivatives of the estimated actuator velocity and estimated glider velocity were also compared against the measured accelerations as another measure of the performance of the estimators.

Table 4.3 Characteristics of System Models Tested

SYSTEM MODEL	ACTUATOR		LOAD	
	MASS (kg)	DAMPING (Nsec/m)	MASS (kg)	DAMPING (Nsec/m)
1	10	18700	400	9000
2	10	12000	400	9000
3	10	7500	400	9000
5	40	18700	400	9000
6	30	18700	400	9000
7	40	30000	400	9000
8	20	18700	400	9000
9	20	18700	800	9000
10	20	9000	800	9000

The data showed that estimators based on system models 1, 2, 7, 8, 9 and 10 were all nearly equal in their ability to match the measured data. The estimator based on system model 5 severely overestimated the actuator velocity and acceleration. Estimators based on models 3 and 6 also over estimated the actuator accelerations, but by less than that observed using model 5. Of the six acceptable estimators, the one based on system model 8 was selected as the best. Its performance, however, was nearly identical to that of the estimator based on model 1 and, therefore, Figures 4.2 to 4.8 which were generated using system model 1 can be used as an indication of the match achieved using system model 8.

The selected estimator was applied to all of the test data acquired for the drive signals listed in Table 3.1. For all of the tests the comparison plots exhibited the following similarities:

- 1) the estimated actuator position matched the measured position extremely well.
- 2) the estimated glider position matched the ideal glider position, calculated using the measured actuator position and the kinematic equations, extremely well.
- 3) the estimated actuator velocity is consistently less than the derivative of the measured actuator position.
- 4) the estimated glider velocity matched the ideal glider velocity very well.
- 5) the estimated differential pressures are typically higher than those measured.
- 6) the derivative of the estimated actuator velocity matches the measured actuator acceleration to within acceptable limits.
- 7) the derivative of the estimated glider velocity matches the measured glider acceleration.

The comparison plots for the worst case test are shown in Figures 4.9 to 4.15. The worst case test conditions were obtained using a square wave drive signal with an amplitude of 3.75 degrees. The accelerations of the actuator and the glider based on the open loop estimator used in section 4 are shown in Figures 4.16 and 4.17. The improvement in the ability to match the measured accelerations when the closed loop estimator is used can be seen by comparing Figure 4.14 with 4.16 and Figure 4.15 with 4.17. The measured accelerations of the glider shown in Figures 4.15 and 4.17 were clipped at $\pm 4 \text{ m/sec}^2$ by the accelerometer signal conditioning circuitry.

Figure 4.9 Actuator Position Comparison for Square Wave Test

Figure 4.10 Actuator Velocity Comparison for Square Wave Test

Figure 4.11 Glider Position Comparison for Square Wave Test

Figure 4.12 Glider Velocity Comparison for Square Wave Test

Figure 4.13 Differential Pressure Comparison for Square Wave Test

Figure 4.14 Actuator Acceleration Comparison for Square Wave Test

Figure 4.15 Glider Acceleration Comparison for Square Wave Test

Figure 4.16 Actuator Acceleration Comparison (Open Loop)

Figure 4.17 Glider Acceleration Comparison (Open Loop)

CHAPTER 5

CONCLUSIONS AND RECOMMENDATIONS FOR FUTURE WORK

5.1 SUMMARY

The system model and estimator developed in the previous sections were shown to provide representative estimates of the system states. An optimal controller based on the system model and using the state feedback made available by the estimator would likely result in improved control of the wave generator.

There are undoubtedly more representative models of the system which could be uncovered. Development of an estimator and a control law based on a better model would further enhance the performance of the controller. However, the trial and error method used to develop the final form of the model and estimator would not be suitable for testing enough cases to find the 'best' model. An alternative method would be to use system identification techniques to construct a system model from experimental data.

Full state information made available by an estimator enables the development of control laws which would be capable of controlling any of the system states, regardless of whether they are measured directly. This capability is particularly attractive in the wave generation field since the experimenter establishing a desired wave field actually wishes to

control the height of the waves. This would be possible if the velocity of the water particles along the face of the wave board could be controlled. Direct control over the velocity of the wave board would make this possible. If the wave height at the wave board was also measured, changes in the water particle velocity due to waves reflected from beaches or offshore structures could be detected and this information could be used to change the velocity of the wave board. Such a control system would be capable of active wave absorption while generating the desired wave climate.

Another advantage in using estimated states for control is that the effect of noise in a feedback signal can be reduced, providing the noise has the characteristics assumed in the development of the Kalman filter. For the wave generator, for example, the filter used in the RVDT signal conditioning circuitry could be reduced in order or modified for a higher corner frequency. This would increase the noise in the measurement of the lever arm angle, but the estimated actuator position available as output from the estimator could be relatively free of the noise. The decrease in phase lag caused by the new filter would increase the phase margin of the system.

5.2 IMPLEMENTATION

Both the estimator and any controller based on the estimator are of little value if they can not be implemented in real-time. Using the

control update rate of 100 Hz, which is used in the current digital controller, this allows a maximum of 10 milliseconds. During each control interval the following steps must be performed.

- 1) measure feedback signals from the RVDT and differential pressure transducer
- 2) use the measured signals and the estimated states in the measurement update equation to update the estimates of the state vector
- 3) use the updated states in the control law to calculate the control signal
- 4) output the control signal
- 5) use the control signal and the updated state vector in the time update equation to estimate the states for the next time step
- 6) error checks and general housekeeping functions.

Steps 1 to 4 should be completed before a significant portion of the control interval passes.

Calculation of the updated states using the measurement update equation involves 12 multiplications and 14 additions. The control law can be implemented in 6 multiplications and 5 additions while the time update equation requires 42 multiplications and 36 additions. The total minimum number of computations for the system model and estimator in the form developed is 60 multiplications and 55 additions. These figures include the savings in computations made possible by the simple form of

the output matrix H which has only two non zero elements, both of which are 1.0. If these operations were to be executed using an 8 Mhz 80287 numeric coprocessor, the multiplications would take approximately 0.72 milliseconds and the additions would take approximately 0.55 milliseconds. These times do not include the time taken to load data into the 80287 or to move the results back into memory. Based on these data it appears as if the controller could be implemented in real-time.

The computational effort could be reduced if the system model was transformed into an alternate form. If the current state variables $x(n)$ were replaced by the new state variables $x_n(n)$ according to the equation:

$$x_n(n) = Tx(n) \quad (7.1)$$

where T is a non-singular matrix then the equivalent system model using the new state variables would be defined by the equations:

$$x_n(n+1) = T^{-1} \Phi T x_n(n) + T^{-1} \Gamma u(n) \quad (7.2)$$

$$y(n) = H T x_n(n) \quad (7.3)$$

The matrix T could be selected such that the similarity transform $T^{-1} \Phi T$ would result in a new system matrix which is diagonal. This would significantly reduce the number of computations in the time update equation. Using the new form of the system, the number of calculations would be 24 multiplications and 24 additions for the measurement update equation, 6 multiplications and 5 additions for the control law, and 12 multiplications and 6 additions for the time update equation. The new total minimum number of computations is 42 multiplications and 35 additions. These computations could be executed in as little as 0.85 milliseconds using an 8 Mhz 80287 numeric coprocessor.

5.3 FUTURE WORK

Additional research work on the wave generator control system should be carried out to fully explore the possibility of optimal control using an optimal estimator. This work should be concentrated in two areas. These are the refinement of the system model and the estimator, and development of an optimal controller.

Direct measurement of the glider position is necessary to improve the system model and the estimator. This information would allow system identification techniques to be used to find an optimum system model. Tests should be carried out in various water depths to determine the effect of water depth on the system model. Once the improved model is developed the estimator can be refined. The direct measurement of the position of the glider would provide improved data against which to compare the estimates of the glider position and glider velocity.

The estimator and system model work carried out indicate that they are sufficiently accurate to warrant implementation. A control law to control the wave generator lever arm angle using the estimated states should be developed and compared with the current control system. If this test proves to be successful, a new controller should be developed to directly control the position of the glider and then the velocity of the glider.

To ease the development and testing of the controllers, they should be initially implemented using a powerful CPU with a numeric coprocessor. Once successful controllers have been designed and tested they could be converted to use fixed point arithmetic. This would decrease the computation time and allow less expensive computer hardware to be used to control the wave generator.

REFERENCES

- 1) Brogan, W.L. 'Modern Control Theory - Second Edition', Prentice-Hall Inc., Englewood Cliffs, New Jersey 07632, 1985
- 2) Franklin, G.F. and Powell, J.D. 'Digital Control of Dynamic Systems', Addison-Wesley Publishing Co., Reading, Massachusetts, 1980
- 3) Funke, E.R., Crookshank, N.L., Wingham M. 'An Introduction to GEDAP - An Integrated Software System for Experimental Control, Data Acquisition and Data Analysis', National Research Council Canada, Hydraulics Laboratory Technical Report, LTR-HY-75, 1980
- 4) Gilbert, G., Thompson, D.M. and Brewer, A.J. 'Design Curves for Regular and Random Wave Generators', Journal of Hydraulic Research, Vol. 9, No. 2, 1971
- 5) Martens, H.R. and Allen, D.R. 'Introduction to Systems Theory', Charles E. Merril Publishing Co., Columbus, Ohio, 1969
- 6) Moog 'Moog A085 Servoactuators', Data Sheet No. 850 184, Moog Inc., Industrial Division, East Aurora, New York 14052
- 7) Moog 'Moog A86 Servoactuators', Data Sheet No. 860 284, Moog Inc., Industrial Division, East Aurora, New York 14052

8) Moog 'Moog Series 62 Low Cost Flow Control Servovalves', Catalog No. 622 482, Moog Inc., Industrial Division, East Aurora, New York 14052

9) Press, W.H. et al 'NUMERICAL RECIPES - The Art of Scientific Computing', Cambridge University Press, Cambridge, 1986

APPENDIX A

Software for the Conversion of the Continuous Time System Model to the
Equivalent Discrete Time System Model

The procedure DDISCR_6 is used to convert the continuous time system model to the equivalent discrete time model. The continuous time model is created from the user specifications of the model parameters. This model is then converted to discrete time and two output files are created. The list file contains the attributes of the continuous time model, both the continuous time model and the discrete time equivalent as well as the error matrices in the conversion process. The control output file contains both the continuous time system and the discrete time system. This file is used as input to other control system analysis procedures developed.

The procedure calls matrix manipulation procedures which are also listed. Computations are generally performed in double precision and the procedures are coded in Microsoft FORTRAN.

PROGRAM DDISCR_6

```

C
C File:DDISCR_6.FOR
C
C*****
C
C   Programmed by P.H. Laurich
C               NRC Hydraulics Lab
C               07-Oct-87
C   Version 1   P.H. Laurich/NRC 07-Oct-87 Modified the code in version
C               4 of DDISCR for the sixth order model of the wave
C               generator.
C   Version 2   P.H. Laurich/NRC 09-Oct-87 Allow user input to specify
C               the mass of the actuator and the damping of the actuator.
C   Version 3   P.H. Laurich/NRC 14-Oct-87 Set the coefficient of
C               leakage to zero.
C   Version 4   P.H. Laurich/NRC 26-Nov-87 Changed to a two output
C               system.
C   Version 5   P.H. Laurich/NRC 27-Nov-87 Added code to allow the user
C               to specify TOU, the servo valve time constant.
C   Version 6   P.H. Laurich/NRC 01-Feb-88 Modified parameter values
C               used in the system matrix to correspond to those used in
C               the thesis.
C   Version 7   P.H. Laurich/NRC 10-Feb-88 Remodified parameter values
C               used in the system matrix to correspond to those used in
C               the thesis.
C
C   This program performs its calculation using double precision reals.
C   The matrix manipulation procedures can be found in the file DPMAT.FOR
C
C   Program to convert the state equations of a continuous, linear,
C   constant coefficient system of differential equations given in the
C   form:
C
C       
$$\frac{dx(t)}{dt} = -Fx(t) + Gu(t)$$

C
C       
$$y(t) = Hx(t)$$

C
C   into the discrete equivalent:
C
C       
$$x(n+1) = PHIx(n) + GAMMAu(n)$$

C       
$$y(n) = Hx(n)$$

C
C   The sampling interval, T, must be given. The discrete representation
C   is obtained from the continuous by assuming that u(t) is input to the
C   system in question through a zero order hold circuit such that
C    $u(t) = u(nT)$  for  $nT \leq t < nT+T$ . It is further assumed that y(nT)
C   is a sampled version of y(t). See chapter 6 in Digital Control of
C   Dynamic Systems by Franklin and Powell for more info.

```

SOME EQUATIONS USED:*****

$$1) \text{ XSI} \approx I + \frac{FT}{2} + \frac{FT^2}{3} + \frac{FT^3}{4} + \dots + \frac{FT^{NTERM-1}}{NTERM}$$

$$2) \text{ GAMMA} = (XSI)(T)(G)$$

$$3) \text{ PHI} = I + (F)(T)(XSI)$$

$$4) \text{ XSIerror} = \text{sum from } NTERM+1 \text{ to infinity of } \frac{(FT)^{i-1}}{i!}$$

$$\text{The first 2 terms of the error} = \frac{(FT)^{NTERM} * [(NTERM+2)*I + FT]}{(NTERM + 2)!}$$

$$5) \text{ The first 2 terms of XSI} = I + FT/2$$

ALGORITHM USED:*****

Step 1) Select NTERM

- 1.1 - calculate XSI for NTERM = 2
- 1.2 - ask user for NTERM
- 1.3 - calculate the sum of the first error terms
- 1.4 - display XSI and the error
- 1.5 - is NTERM ok? If not goto 1.2

Step 2) Calculate the other matrices and print the results

- 2.1 - XSI = I
- 2.2 - K = NTERM
- 2.3 - if K = 1 goto 2.7
- 2.4 - XSI = I + $\frac{FT}{K}$ * XSI
- 2.5 - K = K-1
- 2.6 - goto 2.3
- 2.7 - GAMMA = (T)(XSI)(G)
- 2.8 - GAMMAerror > (T)(XSIerror)(G)
- 2.9 - PHI = I + (F)(T)(XSI)
- 2.10 - PHI error > (F)(T)(XSIerror)
- 2.11 - print F, G, PHI, PHIerror, GAMMA, GAMMAerror

MATRICES REQUIRED:*****

NS - system order
NI - number of inputs
NO - number of outputs

IDEN(NS,NS), F(NS,NS), G(NS,NI), FT(NS,NS), XSI(NS,NS),
XSIER(NS,NS), PHI(NS,NS), PHIER(NS,NS), GAMMA(NS,NI), GAMER(NS,NI),
TEMP(NS,NS), TEMP2(NS,NS), TEMP3(NS,NS), TCOL(NS,NI)

C*****

C

C Set the parameter MO to the maximum order allowed for the system.
C The number of outputs and the number of inputs to the system must
C be less than or equal to the order of the system.

C

```

parameter (MO = 10)
REAL*8 IDEN(MO,MO), F(MO,MO), G(MO,MO), FT(MO,MO), XSI(MO,MO),
1      XSIER(MO,MO), PHI(MO,MO), PHIER(MO,MO), GAMMA(MO,MO),
2      GAMER(MO,MO), TEMP(MO,MO), TEMP2(MO,MO), TEMP3(MO,MO),
3      TCOL(MO,MO), H(MO,MO)
REAL*8 FACT, SCALE, T, SPRING_K, LOAD_MASS, LOAD_DAMPING,
1      TOU, FLOW_K, PISTON_AREA, DA, FA, COEF_LEAK, C_OIL,
2      ACT_MASS, ACT_DAMPING
INTEGER I, J, NTERM, IRESP, NS, NI, NO, CNTRL, LIST, IHR,
1      IMIN, ISEC, IDSEC, IYR, IYEAR, IMON, IDAY
character*27 CNTRL_FILE, LIST_FILE

```

C

C Record the time and date

C

```

call GETTIM(IHR, IMIN, ISEC, IDSEC)
call GETDAT(IYEAR, IMON, IDAY)

```

C

C Initialize matrices and other variables

C

C Obtain the output file name

C

```

write (*,6000)
6000 format(' Enter the name of the control output file ',\ )
read (*,6005) CNTRL_FILE
6005 format(A27)
open(UNIT = 6, file = CNTRL_FILE, status = 'UNKNOWN')
CNTRL = 6
write (*,6010)
6010 format(' Enter the name of the list file ',\ )
read (*,6005) LIST_FILE
open(UNIT = 7, file = LIST_FILE, status = 'UNKNOWN')
LIST = 7

```

C

C Set the system order to 6, the number of inputs to 1 and the number of
C outputs to 2

C

```

NS = 6
NI = 1
NO = 2
T = .01D0
TOU = .022D0
FLOW_K = 2.185D-5
PISTON_AREA = 2.209D-3
C_OIL = 2.769D-10
COEF_LEAK = 4.12D-9

```

```

ACT_MASS = 1.0D2
ACT_DAMPING = 18.689D3
DA = 4.1093
FA = 1.0/DA
SPRING_K = 2.14D6
LOAD_MASS = 350.0D0
LOAD_DAMPING = 4.5D3
do 10 I = 1,NS
    do 20 J = 1,NS
        IDEN(I,J) = 0.D0
        F(I,J) = 0.D0
        H(I,J) = 0.D0
20    continue
        IDEN(I,I) = 1.D0
        G(I,1) = 0.D0
10    continue
C
C Set the non zero terms of F, G and H
C
    write (*,604)
604    format(' enter the servo valve time constant ',\ )
    read (*,*) TOU
    write (*,606)
606    format(' enter the mass of the actuator ',\ )
    read (*,*) ACT_MASS
    write (*,607)
607    format(' enter the damping of the actuator ',\ )
    read (*,*) ACT_DAMPING
    F(1,1) = -1.0D0/TOU
    write (*,600)
600    format(' enter the mass of the load ',\ )
    read (*,*) LOAD_MASS
    write (*,605)
605    format(' enter the damping of the load ',\ )
    read (*,*) LOAD_DAMPING
    F(2,1) = 2.0D0/C_OIL
    F(2,2) = -2.0D0*COEF_LEAK/C_OIL
    F(2,3) = -2.0D0*PISTON_AREA/C_OIL
    F(3,2) = PISTON_AREA/ACT_MASS*1000.0D0
    F(3,3) = -ACT_DAMPING/ACT_MASS
    F(3,4) = -DA/FA*SPRING_K/ACT_MASS
    F(3,6) = -DA*SPRING_K/ACT_MASS
    F(4,3) = 1.D0
    F(5,4) = -DA*SPRING_K/LOAD_MASS
    F(5,5) = -LOAD_DAMPING/LOAD_MASS
    F(5,6) = -SPRING_K/LOAD_MASS
    F(6,5) = 1.0D0
    G(1,1) = FLOW_K/TOU
    H(1,2) = 1.D0
    H(2,4) = 1.D0

```

C

C Store the file header

C

```

      IYR = IYEAR - (IYEAR/100)*100
      write(CNTRL, 6045) IDAY, IMON, IYR, IHR, IMIN, ISEC
6045 format(' *',/,
*      ' *      ',I2,'-',I2.2,'-',I2.5X,I2.2,':',I2.2,':',I2.2)
      write(CNTRL, 6050) NS, NI, NO, T
6050 format(' *',/,
*      ' *****',/,
*      ' * Convert the continuous time system given by:',/,
*      ' *      ',/,
*      ' *      X = FX + Gu',/,
*      ' *      Y = HX',/,
*      ' *',/,
*      ' * to the equivalent discrete time system given by ',/,
*      ' *',/,
*      ' *      x(n+1) = PHIx(n) + GAMMAu(n)',/,
*      ' *      y(n) = Hx(n)',/,
*      ' *',/,
*      ' *****',/,
*      ' *',/,
*      ' * SYSTEM ORDER      : ',I2,/,
*      ' * NUMBER OF INPUTS   : ',I2,/,
*      ' * NUMBER OF OUTPUTS  : ',I2,/,
*      ' * TIME STEP         : ',F6.3)

```

C

C FORM FT

C

CALL MSCLS(F, MO, MO, FT, MO, MO, NS, NS, T)

C

C Calculate XSI using first 2 terms

C

```

      CALL MSCLS(FT, MO, MO, TEMP, MO, MO, NS, NS, 0.5D0)
      CALL MADDS(TEMP, MO, MO, IDEN, MO, MO, XSI, MO, MO, NS, NS)

```

C

C Calculate the sum of the first 2 error terms using the user
C specified NTERM

C

C NTERM

C 1) form (FT)

C

C (NTERM+2)!

C 2) multiply this by ((NTERM+2)*IDEN + FT)

C

NTERM = 125

CALL MMULS(FT, MO, MO, NS, NS, FT, MO, MO, NS, NS, TEMP, MO, MO)

IF (NTERM .GT. 2) THEN

DO 100 I = 2, NTERM-1

CALL MMULS(TEMP, MO, MO, NS, NS, FT, MO, MO, NS, NS,

* TEMP2, MO, MO)

CALL MCOPY(TEMP2, MO, MO, TEMP, MO, MO, NS, NS)

```

100     CONTINUE
      ENDIF
      FACT = 1.D0
      DO 110 I = 1, NTERM+2
        FACT = FACT*DFLOAT(I)
110    CONTINUE
      SCALE = 1.D0/FACT
      CALL MSCLS(IDEN, MO, MO, TEMP2, MO, MO, NS, NS, DFLOAT(NTERM+2))
      CALL MADDS(TEMP2, MO, MO, FT, MO, MO, TEMP3, MO, MO, NS, NS)
      CALL MMULS(TEMP, MO, MO, NS, NS, TEMP3, MO, MO, NS, NS,
*          XSIER, MO, MO)
      CALL MSCLS(XSIER, MO, MO, TEMP, MO, MO, NS, NS, SCALE)
      CALL MCOPY(TEMP, MO, MO, XSIER, MO, MO, NS, NS)

C
C   Print XSI and error if NTERM terms are used
C
      WRITE(*, 1010) NTERM
1010  FORMAT(' If the matrix XSI is formed using', I3, ' terms', '/',
1      ' then the XSI matrix will be approximately:', '/')
      WRITE(*, 1020) ((XSI(I,J), J = 1, NS), I=1, NS)
1020  FORMAT(6(6(D11.4, 2X)/))
      WRITE(*, 1030) ((XSIER(I,J), J=1, NS), I=1, NS)
1030  FORMAT(' and the error will be approximately: ', '/',
1      ' 6(6(D11.4, 2X)/))

C
C   Step 2. Calculate the actual values of the matrices XSI, PHI and GAMM
C
      CALL MCOPY(IDEN, MO, MO, XSI, MO, MO, NS, NS)
      K = NTERM
200   CONTINUE
      IF (K .EQ. 1) GOTO 300
      CALL MMULS(FT, MO, MO, NS, NS, XSI, MO, MO, NS, NS, TEMP, MO, MO)
      CALL MSCLS(TEMP, MO, MO, TEMP2, MO, MO, NS, NS, 1.D0/DFLOAT(K))
      CALL MADDS(IDEN, MO, MO, TEMP2, MO, MO, XSI, MO, MO, NS, NS)
      K = K-1
      GOTO 200
300   CONTINUE

C
C   Use XSI to calculate GAMMA and GAMMA error
C
      CALL MMULS(XSI, MO, MO, NS, NS, G, MO, MO, NS, NI, TCOL, MQ, MO)
      CALL MSCLS(TCOL, MO, MO, GAMMA, MO, MO, NS, NI, T)
      CALL MMULS(XSIER, MO, MO, NS, NS, G, MO, MO, NS, NI, TCOL, MO, MO)
      CALL MSCLS(TCOL, MO, MO, GAMER, MO, MO, NS, NI, T)

C
C   Use XSI and XSIER to calculate PHI and PHIER
C
      CALL MMULS(FT, MO, MO, NS, NS, XSI, MO, MO, NS, NS,
*          TEMP, MO, MO)
      CALL MADDS(TEMP, MO, MO, IDEN, MO, MO, PHI, MO, MO, NS, NS)
      CALL MMULS(FT, MO, MO, NS, NS, XSIER, MO, MO, NS, NS,

```

```

      *          PHIER, MO, MO)
C
C Print the final results to the list file
C
      write(LIST, 2000) NS, TOU, FLOW_K, DA, FA, LOAD_MASS,
      *          LOAD_DAMPING, SPRING_K, PISTON_AREA,
      *          ACT_MASS, ACT_DAMPING, C_OIL, COEF_LEAK
2000 format(' The ', I2, 'th order continuous time system with the '
      *          ' following characteristics: ', /,
      *          ' TOU', D11.5, /,
      *          ' FLOW GAIN', D11.5, ' m**3/sec.mamp', /,
      *          ' DISPLACEMENT AMPLIFICATION', D11.5, /,
      *          ' FORCE AMPLIFICATION', D11.5, /,
      *          ' LOAD MASS', D11.5, ' kg', /,
      *          ' LOAD DAMPING', D11.5, ' N.sec/m', /,
      *          ' SPRING CONSTANT', D11.5, ' N/m', /,
      *          ' PISTON AREA', D11.5, ' m**2', /,
      *          ' ACTUATOR MASS', D11.5, ' kg', /,
      *          ' ACTUATOR DAMPING', D11.5, ' N.sec/m', /,
      *          ' OIL VOL/BULK MOD', D11.5, ' m**5/KN', /,
      *          ' COEFFICIENT OF LEAKAGE', D11.5, ' m**5/KN', /,
      *          ' has the continuous time equations : ', /)
      DO 400 I = 1, NS
      WRITE(LIST, 2010) (F(I, J), J = 1, NS), G(I, 1)
400 CONTINUE
2010 FORMAT(' ', 6(D10.4, 1X), '!', ' ', D10.4)
      WRITE(LIST, 2020)
2020 FORMAT(' ', /, ' can be approximated by the discrete equivalent: ')
      DO 410 I = 1, NS
      WRITE(LIST, 2010) (PHI(I, J), J = 1, NS), GAMMA(I, 1)
410 CONTINUE
      WRITE(LIST, 2040) NTERM
2040 FORMAT(' ', /, ' The above approximation uses ', I3, ' terms ',
      *          ' for XSI and has the following ', /, ' error matrices ', /)
      WRITE(LIST, 2050) ((XSIER(I, J), J = 1, NS), I = 1, NS)
2050 FORMAT(' XSI ERROR: ', /, 6(2X, 6(D11.5, 2X)))
      WRITE(LIST, 2060) ((PHIER(I, J), J = 1, NS), I = 1, NS)
2060 FORMAT(' PHI ERROR: ', /, 6(2X, 6(D11.5, 2X)))
      WRITE(LIST, 2070) (GAMER(I, 1), I = 1, NS)
2070 FORMAT(' GAMMA ERROR: ', /, 10(2X, D11.5))
C
C Print the final results to the control output file
C
      write(CNTRL, 6060) NS, NS
6060 format(' * F MATRIX: (', I2, 'x', I2, ')')
      DO 618 I = 1, NS
      WRITE(CNTRL, 6070) (F(I, J), J = 1, NS)
618 CONTINUE
6070 FORMAT(' ', 10(D11.5, 2X))
      write(CNTRL, 6080) NS, NI
6080 format(' * G MATRIX: (', I2, 'x', I2, ')')

```

```

DO 619 I = 1, NS
    WRITE(CNTRL, 6090) G(I, 1)
619  CONTINUE
6090 FORMAT(' ', (D11.5, 2X))
    write(CNTRL, 6100) NO, NS
6100 format(' * H MATRIX:  (' , I2, 'x' , I2, ')')
    DO 621 I = 1, NO
        WRITE(CNTRL, 6070) (H(I, J), J = 1, NS)
621  CONTINUE
    write(CNTRL, 6110) NTERM
6110 format(' * NUMBER OF TERMS IN SUMMATION: ', I5)
    write(CNTRL, 6120) NS, NS
6120 format(' * PHI MATRIX:  (' , I2, 'x' , I2, ')')
    DO 622 I = 1, NS
        WRITE(CNTRL, 6070) (PHI(I, J), J = 1, NS)
622  CONTINUE
    write(CNTRL, 6130) NS, NI
6130 format(' * GAMMA MATRIX:  (' , I2, 'x' , I2, ')')
    DO 623 I = 1, NS
        WRITE(CNTRL, 6090) GAMMA(I, 1)
623  CONTINUE
C
C Rewrite the H matrix into the control output file
C
    write(CNTRL, 6100) NO, NS
    DO 624 I = 1, NO
        WRITE(CNTRL, 6070) (H(I, J), J = 1, NS)
624  CONTINUE
    END

```

C
C File: DPMAT.FOR
C

C-----
C
C Routines for matrix manipulation using double precision
C

C-----
C
C Version 1 P.H. Laurich/NRC 24-Sept-87 Copied the code from SPMAT
C and converted it to use double precision.
C Version 2 P.H. Laurich/NRC 28-Sept-87 Modified the code to allow
C the input matrices to be of arbitrary dimension with only
C part of the matrix being used to store data. This
C requires that the dimension of the matrix and the used
C area of the matrix get passed to these subroutines.
C Version 3 P.H. Laurich/NRC 05-Oct-87 Added the procedure GAUSSJ
C which is used to obtain the solution to a set of linear
C equations using Gauss-Jordan elimination. The single
C precision procedure was taken from Numerical Recipes.
C Version 4 P.H. Laurich/NRC 30-Oct-87 Added the procedure SVDCMP
C which is used to perform a singular value decomposition
C of a matrix. The single precision procedure was taken
C from Numerical Recipes.
C Version 5 P.H. Laurich/NRC 02-Nov-87 Added the procedure MINV
C which is used to compute the inverse of a matrix. The
C procedure uses SVDCMP.
C Version 6 P.H. Laurich/NRC 04-Nov-87 Removed all of the
C subroutines not dealing with matrix manipulation. The
C subroutines dealing with the solution of linear equations
C were placed in DPLIN.FOR. The subroutines dealing with
C the solution of eigensystems were placed in DPEIGEN.FOR
C

C*****
C*****
C

SUBROUTINE MMULS(A, NRA_DIM, NCA_DIM, NRA_USED, NCA_USED,
* B, NRB_DIM, NCB_DIM, NRB_USED, NCB_USED,
* C, NRC_DIM, NCC_DIM)

C
C-----
C Subroutine MMULS multiplies an MxN matrix by an NxP matrix to
C produce an MxP result. The dimensioned size of the various arrays
C is indicated in the **_DIM variables and the used area of the arrays
C is indicated by the **_USED variables.
C-----
C

C
C
C INTEGER NRA_DIM, NCA_DIM, NRA_USED, NCA_USED,
* NRB_DIM, NCB_DIM, NRB_USED, NCB_USED,
* NRC_DIM, NCC_DIM, I, J, K

```

      REAL*8 A(NRA_DIM, NCA_DIM), B(NRB_DIM, NCB_DIM),
      *      C(NRC_DIM, NCC_DIM)
      if (NCA_USED .ne. NRB_USED) then
        WRITE(*, 100)
100      FORMAT(' ERROR MMULS: matrices can not be multiplied as',
        1      ' dimensioned')
        return
      else if ((NRA_USED .gt. NRC_DIM)
      *      .or. (NCB_USED .gt. NCC_DIM)) then
        WRITE(*, 105)
105      FORMAT(' ERROR MMULS: matrix not large enough for result')
        return
      endif
      DO 10 I = 1, NRA_USED
        DO 10 J = 1, NCB_USED
          C(I,J) = 0.0D0
          DO 10 K = 1, NRB_USED
            C(I,J) = C(I,J) + A(I,K)*B(K,J)
10      CONTINUE
      RETURN
      END

```

```

C
C*****
C*****
C

```

```

      SUBROUTINE MADDS(A, NRA_DIM, NCA_DIM, B, NRB_DIM, NCB_DIM,
      *      C, NRC_DIM, NCC_DIM, NROW, NCOL)

```

```

C
C-----
C  Subroutine MADDS adds an MxN matrix to another MxN matrix to
C  produce an MxN result. The dimensioned size of the arrays are
C  passed in the **_DIM variables; the used area of the arrays are
C  indicated by the NROW and NCOL variables.
C-----
C
C

```

```

      INTEGER NRA_DIM, NCA_DIM, NRB_DIM, NCB_DIM, NRC_DIM, NCC_DIM,
      *      NROW, NCOL, I, J
      REAL*8 A(NRA_DIM, NCA_DIM), B(NRB_DIM, NCB_DIM),
      *      C(NRC_DIM, NCC_DIM)
      DO 10 I = 1, NROW
        DO 10 J = 1, NCOL
          C(I,J) = A(I,J) + B(I,J)
10      CONTINUE
      RETURN
      END

```

```

C
C*****
C*****
C

```

```

      SUBROUTINE MSCLS(A, NRA_DIM, NCA_DIM, B, NRB_DIM, NCB_DIM,

```

```

      *                               NROW, NCOL, SCALE)
C
C-----
C  Subroutine MSCLS multiplies an MxN matrix by a scalar, SCALE, to
C  produce an MxN result. The dimensions of the arrays are passed in
C  the **_DIM variables; NROW and NCOL indicate the area of the arrays
C  in use.
C-----
C
C
C      INTEGER NRA_DIM, NCA_DIM, NRB_DIM, NCB_DIM, NROW, NCOL, I
C      REAL*8 A(NRA_DIM, NCA_DIM), B(NRB_DIM, NCB_DIM), SCALE
C      do 10 I = 1, NROW
C          do 10 J = 1, NCOL
C              B(I,J) = A(I,J)*SCALE.
10  continue
      RETURN
      END
C
C*****
C*****
C
C      SUBROUTINE MTRNS(A, NRA_DIM, NCA_DIM, B, NRB_DIM, NCB_DIM,
C      *                NROW, NCOL)
C
C-----
C  Subroutine MTRNS transposes an MxN matrix to produce an NxM result.
C  The dimensioned size of the arrays are passed in the **_DIM
C  variables while the used areas of the arrays are indicated by NROW
C  and NCOL.
C-----
C
C
C      INTEGER NRA_DIM, NCA_DIM, NRB_DIM, NCB_DIM, NROW, NCOL, I, J
C      REAL*8 A(NRA_DIM, NCA_DIM), B(NRB_DIM, NCB_DIM)
C      DO 10 I = 1, NROW
C          do 10 J = 1, NCOL
C              B(J,I) = A(I,J)
10  CONTINUE
      RETURN
      END.
C
C*****
C*****
C
C      SUBROUTINE MCOPY(A, NRA_DIM, NCA_DIM, B, NRB_DIM, NCB_DIM,
C      *                NROW, NCOL)
C
C-----
C  Subroutine MCOPY copies an MxN matrix into another MxN matrix.
C-----

```

C
C

```
INTEGER NRA_DIM, NCA_DIM, NRB_DIM, NCB_DIM, NROW, NCOL, I, J
REAL*8 A(NRA_DIM, NCA_DIM), B(NRB_DIM, NCB_DIM)
DO 10 I = 1, NROW
  DO 10 J = 1, NCOL
    B(I,J) = A(I,J)
10 CONTINUE
RETURN
END
```


OUTPUT LIST FILE

The 6th order continuous time system with the following characteristics:

TOU	.22000D-01	
FLOW GAIN	.21850D-04	m**3/sec.mamp
DISPLACEMENT AMPLIFICATION	.41093D+01	
FORCE AMPLIFICATION	.24335D+00	
LOAD MASS	.40000D+03	kg
LOAD DAMPING	.45000D+04	N.sec/m
SPRING CONSTANT	.21400D+07	N/m
PISTON AREA	.22090D-02	m**2
ACTUATOR MASS	.50000D+01	kg
ACTUATOR DAMPING	.18700D+05	N.sec/m
OIL VOL/BULK MOD	.27690D-09	m**5/KN
COEFFICIENT OF LEAKAGE	.41200D-08	m**5/KN

has the continuous time equations :

-.4545D+02	.0000D+00	.0000D+00	.0000D+00	.0000D+00	.0000D+00	!	.9932D-03
.7223D+10	-.2976D+02	-.1596D+08	.0000D+00	.0000D+00	.0000D+00	!	.0000D+00
.0000D+00	.4418D+00	-.3740D+04	-.7227D+07	.0000D+00	-.1759D+07	!	.0000D+00
.0000D+00	.0000D+00	.1000D+01	.0000D+00	.0000D+00	.0000D+00	!	.0000D+00
.0000D+00	.0000D+00	.0000D+00	-.2198D+05	-.1125D+02	-.5350D+04	!	.0000D+00
.0000D+00	.0000D+00	.0000D+00	.0000D+00	.1000D+01	.0000D+00	!	.0000D+00

can be approximated by the discrete equivalent:

.6347D+00	.0000D+00	.0000D+00	.0000D+00	.0000D+00	.0000D+00	!	.7981D-05
.2666D+08	.3343D+00	.1867D+04	.5142D+07	.1580D+05	.1930D+07	!	.1526D+03
.1057D+03	.4449D-04	.6406D+00	-.2523D+03	-.8896D-01	.2659D+03	!	.1669D-02
.1680D+01	.3298D-07	-.1154D-03	.5478D+00	-.1146D-02	-.1529D+00	!	.8832D-05
-.1801D+03	-.5464D-05	-.1079D-02	-.9374D+02	.7731D+00	-.2256D+02	!	-.6199D-03
-.6241D+00	-.2885D-07	-.1476D-04	-.5156D+00	.9028D-02	.8744D+00	!	-.1566D-05

The above approximation uses 125 terms for XSI and has the following error matrices

XSI ERROR:

-.6614D-254	.00000D+00	.00000D+00	.00000D+00	.00000D+00	.00000D+00
.20777D-08	-.38584D-14	.22907D-10	.62991D-07	-.11961D-11	.15329D-07
.17487D-11	-.63430D-18	-.24795D-14	.10257D-10	-.95936D-15	.24961D-11
-.13284D-15	.24127D-21	-.14195D-17	-.39388D-14	.76375D-19	-.95850D-15
.19252D-14	.41398D-21	-.11992D-16	-.69084D-14	-.10361D-17	-.16812D-14
-.70685D-18	.26210D-24	.95469D-21	-.42396D-17	.38789D-21	-.10317D-17

PHI ERROR:

.30063D-254	.00000D+00	.00000D+00	.00000D+00	.00000D+00	.00000D+00
-.27963D-06	.10235D-12	.38880D-09	-.16553D-05	.15342D-09	-.40282D-06
-.46609D-10	-.10766D-16	.29651D-12	.17941D-09	.25069D-13	.43661D-10
.17487D-13	-.63430D-20	-.24795D-16	.10257D-12	-.95936D-17	.24961D-13

.29026D-13 -.53104D-19 .31337D-15 .86693D-12 -.16695D-16 .21097D-12
.19252D-16 .41398D-23 -.11992D-18 -.69084D-16 -.10361D-19 -.16812D-16

GAMMA ERROR:

-.65689-259
.20635D-13
.17368D-16
-.13194D-20
.19121D-19
-.70203D-23

OUTPUT CONTROL FILE

```

*
*   1-03-88   . 12:12:02
*
*****
* Convert the continuous time system given by:
*
*    $X = FX + Gu$ 
*    $Y = HX$ 
*
* to the equivalent discrete time system given by
*
*    $x(n+1) = PHIx(n) + GAMMAu(n)$ 
*    $y(n) = Hx(n)$ 
*
*****
*
* SYSTEM ORDER      : 6
* NUMBER OF INPUTS  : 1
* NUMBER OF OUTPUTS : 2
* TIME STEP         : .010
* F MATRIX: ( 6x 6)
.45455D+02 .00000D+00 .00000D+00 .00000D+00 .00000D+00 .00000D+00
.72228D+10 -.29758D+02 -.15955D+08 .00000D+00 .00000D+00 .00000D+00
.00000D+00 .44180D+00 -.37400D+04 -.72274D+07 .00000D+00 -.17588D+07
.00000D+00 .00000D+00 .10000D+01 .00000D+00 .00000D+00 .00000D+00
.00000D+00 .00000D+00 .00000D+00 -.21985D+05 -.11250D+02 -.53500D+04
.00000D+00 .00000D+00 .00000D+00 .00000D+00 .10000D+01 .00000D+00
* G MATRIX: ( 6x 1)
.99318D-03
.00000D+00
.00000D+00
.00000D+00
.00000D+00
.00000D+00
* H MATRIX: ( 2x 6)
.00000D+00 .10000D+01 .00000D+00 .00000D+00 .00000D+00 .00000D+00
.00000D+00 .00000D+00 .00000D+00 .10000D+01 .00000D+00 .00000D+00
* NUMBER OF TERMS IN SUMMATION: 125
* PHI MATRIX: ( 6x 6)
.63474D+00 .00000D+00 .00000D+00 .00000D+00 .00000D+00 .00000D+00
.26664D+08 .33435D+00 .18669D+04 .51424D+07 .15801D+05 .19300D+07
.10565D+03 .44488D-04 .64061D+00 -.25225D+03 -.88955D-01 .26592D+03
.16800D+01 .32978D-07 -.11539D-03 .54775D+00 -.11459D-02 -.15293D+00
-.18005D+03 -.54643D-05 -.10788D-02 -.93742D+02 .77305D+00 -.22556D+02
-.62411D+00 -.28854D-07 -.14762D-04 -.51560D+00 .90278D-02 .87439D+00
* GAMMA MATRIX: ( 6x 1)
.79810D-05
.15261D+03
.16685D-02

```

.88323D-05

-.61986D-03

-.15662D-05

* H MATRIX: (2x 6)

.00000D+00	.10000D+01	.00000D+00	.00000D+00	.00000D+00	.00000D+00
.00000D+00	.00000D+00	.00000D+00	.10000D+01	.00000D+00	.00000D+00

APPENDIX B

Software for the Calculation of the Steady State Optimal Estimator

The software to compute the steady state optimal estimator gain matrix consists of the mainline procedure SS_OPT_0 and other procedures in the files DPMAT.FOR, DPLIN.FOR, DPEIGEN.FOR and CON_SUB.FOR. The file DPMAT.FOR contains double precision matrix manipulation routines and is included in Appendix A. Any software dealing with the solution of linear equations are stored in the file DPLIN.FOR. These procedures are generally modifications of procedures taken from the Numerical Recipes software package. Most of the procedures for solving eigensystems are also based on procedures taken from Numerical Recipes. These procedures make up the file DPEIGEN.FOR. Any procedures used to read matrices or other information from the control output file created by DDISCR_6 or the estimator output file created by SS_OPT_0 are included in the file CON_SUB.FOR.

The procedures, SVDCMP, SVBKSB, LUDCMP and LUBKSB, which deal with the solution of linear equations can be found in the Numerical Recipes software package. The procedures provided with the package must, however, be modified to use double precision. The Numerical Recipes eigensystem analysis procedures: BALANCE, ELMHES and HQR, are also not listed in this appendix, but must also be modified to use double precision.

The mainline procedure SS_OPT_0 solves the steady state optimal estimator problem by reformating the problem into steady state optimal control form. The procedure inputs the control file created by DDISCR_6, requests covariance information from the user and outputs an estimator output file. The estimator output file can contain multiple estimator gain matrices for different covariance specifications.

The estimator output file is then used as input to the procedure KAL_SIM which simulates the steady state Kalman filter. The procedure inputs a file of measured data, acquired during a wave generator test, to compute the estimated state vector and the error between the estimated states and the measured states for each time step. A list file is produced which shows the system model, the estimator gain matrix and the average absolute error between the estimated states and the measured states. If there is more than one estimator gain matrix in the estimator output file input to the procedure, the output list file contains the results from the simulation of all of the estimators.

The software makes reference to estimators and observers. Both names refer to the same process.

PROGRAM SS_OPT_O

```

C
C*****
C
C   Programmed by P.H. Laurich
C               NRC Hydraulics Lab
C               11-Nov-87
C   Version 1   P.H. Laurich/NRC 11-Nov-87 Created program to compute
C               the steady state optimal observer gain matrix. The method
C               is described in the Digital Control text by Franklin and,
C               Powell. See chapter 9 section 9.3.2.
C   Version 2   P.H. Laurich/NRC 12-Nov-87 Added code to create a file
C               containing the covariances specified and the resulting
C               observer gain matrix. The user is also allowed to loop
C               through the program and specify as many different options
C               as desired.
C
C   This program uses double precision reals.
C   The subroutines used can be found in the file CON_SUB.FOR.
C
C   Program to compute the gain matrix L required to achieve the desired
C   observer response for a constant coefficient system given in the
C   form:
C
C        $x(n+1) = \text{PHI}x(n) + \text{GAMMA}u(n)$ 
C        $y(n) = Hx(n)$ 
C
C   The matrices PHI and H, and information on the order of the system
C   and the number of outputs is read from the control file input.
C
C   ALGORITHM USED:*****
C
C   Step 1) Read the required information from the control input file
C           specified by the user.
C   Step 2) ask the user to specify the covariance matrix for the
C           process noise and the covariance of the measurement noise.
C           The matrices are known as RW and RV respectively.
C   Step 3) solve the steady state optimal observer equations by making
C           the required modifications to the matrices so that
C           subroutine SOLVE_SS_OPT_C_EQNS can be used.
C   Step 4) print the results
C
C   NS - system order
C   NI - number of inputs
C   NO - number of outputs
C
C*****
C
C   Set the parameter MO to the maximum order allowed for the system.
C   The number of outputs and the number of inputs to the system must
C   be less than or equal to the order of the system.

```

```

C
parameter (MO = 6, MO2 = MO*MO)
REAL*8 PHI(MO,MO), H(MO,MO), RW(MO,MO), RV(MO,MO),
*      OGAIN(MO,MO), PHI_T(MO,MO), H_T(MO,MO), OGAIN_T(MO,MO), T,
*      GAMMA(MO,MO)
INTEGER I, J, K, NS, NI, NO, CNTRL, OBS, IHR, IMIN,
1      ISEC, IDSEC, IYR, IYEAR, IMON, IDAY, IPRINT
character*27 CNTRL_FILE, OBS_FILE
character*1 IRESP
data RW /MO2*0.0D0/
data RV /MO2*0.0D0/

C
C Record the time and date
C
call GETTIM(IHR, IMIN, ISEC, IDSEC)
call GETDAT(IYEAR, IMON, IDAY)

C
C Get the name of the control input file and the observer gain output
C list file. Read the required input from the control file and then
C write the header and system information into the output file.
C
write (*,6000)
6000 format(' Enter the name of the control input file ',\ )
read (*,6005) CNTRL_FILE
6005 format(A27)
open(UNIT = 6, file = CNTRL_FILE, status = 'OLD', ERR = 6010)
goto 6015
6010 write (*,6012)
6012 format(' Error in reading the file specified. Check to ensure',
*          ' that the file exists')
goto 999
6015 continue
CNTRL = 6
write (*,6020)
6020 format(' Enter the name of the observer gain output file ',\ )
read (*,6005) OBS_FILE
open(UNIT = 7, file = OBS_FILE, status = 'UNKNOWN')
OBS = 7

C
C Read the system order and the PHI matrix. Print an error message
C if any of the records were not in the file.
C
call READ_PHI(MO, CNTRL_FILE, PHI, NS, IERR)
if (IERR .ne. 0) then
write(*, 5000) IERR
5000 format(' Error in reading the PHI matrix and system order.',
*          ' Error code: ',I3)
goto 999
endif

C
C Read the number of inputs and the GAMMA matrix. Print an error message

```


C if any of the records were not in the file.

C

call READ_GAMMA(MO, CNTRL_FILE, GAMMA, NS, NI, IERR)

if (IERR .ne. 0) then

write(*, 5005) IERR

5005 format(' Error in reading the GAMMA matrix and the number.',

* ' of inputs. Error code: ',I3)

goto 999

endif

C

C Read the number of outputs and the H matrix. Print an error message

C if any of the records were not in the file.

C

call READ_H(MO, CNTRL_FILE, H, NS, NO, IERR)

if (IERR .ne. 0) then

write(*, 5010) IERR

5010 format(' Error in reading the H matrix and the number',

* ' of outputs. Error code: ',I3)

goto 999

endif

C

C Read the time step size. Print an error message if the record was ,

C not in the file.

C

call READ_T(CNTRL_FILE, T, IERR)

if (IERR .ne. 0) then

write(*, 5015) IERR

5015 format(' Error in reading the time step size. Error code: ',I3)

goto 999

endif

C

C Store the file header

C

IYR = IYEAR - (IYEAR/100)*100

write(OBS, 7000) IDAY, IMON, IYR, IHR, IMIN, ISEC

7000 format(' **',/,

* ' * ',I2,'-',I2.2,'-',I2.5X,I2.2,':',I2.2,':',I2.2)

write(OBS, 7005) NS, NI, NO, T

7005 format(' **',/,

* ' *****',

* ' *****',/,

* ' * The discrete time system given by ',/,

* ' * ',/,

* ' * $x(n+1) = PHIx(n) + GAMMAu(n)$ ',/,

* ' * $y(n) = Hx(n)$ ',/,

* ' * ',/,

* ' * has the covariance matrices and observer gain',

* ' matrices given below',/,

* ' * ',/,

* ' *****',

* ' *****',/,

```

*      ' *',/,
*      ' * SYSTEM ORDER      : ',I2,/,
*      ' * NUMBER OF INPUTS   : ',I2,/,
*      ' * NUMBER OF OUTPUTS  : ',I2,/,
*      ' * TIME STEP          : ',F6.3)

```

C

C Copy the system matrices to the observer gain output file

C

```

      write(OBS,7010) NS, NS
7010 format(' * PHI MATRIX: (',I2,'x',I2,')')
      DO 700 I = 1,NS
          WRITE(OBS,7015) (PHI(I,J), J = 1,NS)
7015 FORMAT(' ',10(D11.5,2X))
700 CONTINUE
      write(OBS,7020) NS, NI
7020 format(' * GAMMA MATRIX: (',I2,'x',I2,')')
      DO 705 I = 1,NS
          WRITE(OBS,7015) (GAMMA(I,J), J = 1,NI)
705 CONTINUE
      write(OBS,7025) NO, NS
7025 format(' * H MATRIX: (',I2,'x',I2,')')
      DO 710 I = 1,NO
          WRITE(OBS,7015) (H(I,J), J = 1,NS)
710 CONTINUE

```

C

C Print the PHI and H matrices

C

```

      write (*,*) 'PHI MATRIX'
      DO 410 I = 1,NS
          WRITE(*,2030) (PHI(I,J), J = 1,NS)
410. CONTINUE
2030 FORMAT(' ',6(D10.4,1X))
      write (*,*) 'H MATRIX'
      DO 415 I = 1,NO
          WRITE(*,2030) (H(I,J), J = 1,NS)
415. CONTINUE
      write(OBS,7027)
7027 format(' *** END OF SYSTEM DEFINITION PORTION ***')
9000 continue

```

C

C Get the RW and RV matrices from the user. RW is the covariance of the
 C process noise and RV is the covariance of the measurement noise. Copy
 C these matrices into the output observer gain files after they are
 C input.

C

```

      write (*,2050) NS
2050 format(' Enter the covariances of the process noise for the '
*      ',I2,' states: '\)
      read(*,*) (RW(K,K), K = 1,NS)
      write(OBS,7030)
7030 format(' * PROCESS NOISE COVARIANCES:')

```

```

write(OBS,7015) (RW(J,J), J = 1,NS)
write (*,2055) NO
2055 format(' Enter the covariance of the ',I2,' measurements: '\)
      read(*,*) (RV(K,K), K = 1,NO)
      write(OBS,7035)
7035 format(' * MEASUREMENT NOISE COVARIANCES:')
      write(OBS,7015) (RV(J,J), J = 1,NO)
C
C Make the required transformations so that the steady state optimal
C control equations can be used
C
      call MTRNS(PHI, MO, MO, PHI_T, MO, MO, NS, NS)
      call MTRNS(H, MO, MO, H_T, MQ, MO, NO, NS)
C
C Solve the steady state optimal control equations. Set IPRINT to 0 to
C suppress printing of the eigenvectors and eigenvalues.
C
      IPRINT = 0
      call SOLVE_SS_OPT_C_EQNS(PHI_T, H_T, NS, NO, MO, RW, RV, OGAIN_T,
*                               IPRINT)
C
C Print the results
C
      call MTRNS(OGAIN_T, MO, MO, OGAIN, MO, MO, NO, NS)
      write(*,*) 'The observer gain matrix: '
      do 515 I = 1, NS
        write(*,2115) (OGAIN(I,J), J = 1,NO)
2115    format(' ',15(D10.4,1X))
515    continue
      write(OBS,7040) NS, NO
7040 format(' * OBSERVER GAIN: ('I2,'x',I2,')')
      DO 715 I = 1,NS
        WRITE(OBS,7015) (OGAIN(I,J), J = 1,NO)
715    CONTINUE
      write(*,2120)
2120 format(' Another case? (Y/N): '\)
      read(*,2125) IRESP
2125 format(A1)
      if ((IRESP .eq. 'Y') .or. (IRESP .eq. 'y')) goto 9000
999    continue
      END

```

```

C
C   File: DPEIGEN.FOR
C
C-----
C
C   Routines for dealing with eigensystems using double precision.
C   These routines are generally based on programs from Numerical Recipes.
C
C-----
C
C   Version 1   P.H. Laurich/NRC   04-Nov-87   Copied the code from DPMAT
C               and deleted all subroutines not dealing with the solution
C               of eigensystems.
C
C*****
C*****
C
C       SUBROUTINE BALANC(A,N,NP)
C
C-----
C   Subroutine BALANC replaces an NxN matrix A, stored in an array with
C   physical dimensions NPxNP, by a balanced array with identical
C   eigenvalues. The parameter RADIX should be the machine's floating
C   point radix.
C-----
C
C   **** SEE NUMERICAL RECIPES SOFTWARE PACKAGE ****
C
C*****
C*****
C
C       SUBROUTINE ELMHES(A,N,NP)
C
C-----
C   Subroutine ELMHES reduces an NxN real, nonsymmetric matrix A into
C   Hessenberg form by the elimination method. The matrix A is stored in
C   an array with physical dimensions NPxNP. On output, the Hessenberg
C   matrix is in elements A(I,J) with I<=J+1. The other elements of A are
C   to be considered as random values. The subroutine BALANC should be
C   called before calling this subroutine.
C-----
C
C   **** SEE NUMERICAL RECIPES SOFTWARE PACKAGE ****
C
C*****
C*****
C
C       SUBROUTINE HQR(A,N,NP,WR,WI)
C
C-----
C   Subroutine HQR finds all eigenvalues of an NxN upper Hessenberg

```

```

C   matrix A stored in an NPxNP array. The A matrix can be of the form
C   created by ELMHES on input. A is destroyed on output. The real and
C   imaginary parts of the eigenvalues are returned in WR and WI
C   respectively.
C-----
C
C   **** SEE NUMERICAL RECIPES SOFTWARE PACKAGE ****
C
C
C*****
C*****
C
C       SUBROUTINE C_EVECT(A, WR, WI, MOI, NP, EVECTR, EVECTI, IERR)
C
C-----
C   Subroutine C_EVECT computes the complex eigenvector corresponding to
C   the complex eigenvalue passed to it.
C*****
C   INPUT DATA:
C
C   A       - the A array in the eigenvector calculation (REAL*8)
C   WR      - the real part of the eigenvector           (REAL*8)
C   WI      - the imaginary part of the eigenvector      (REAL*8)
C   MOI     - the dimensioned size of the A array
C   NP      - the part of the A array used
C
C*****
C   OUTPUT DATA:
C
C   EVECTR  - the real part of the eigenvector. The array has MOI
C             elements.
C   EVECTI  - the imaginary part of the eigenvector. The array has MOI
C             elements.
C   IERR    - an error code: set to 0 if no errors occurred
C             set to -1 if there were errors
C
C*****
C   PROCEDURE:
C   1) Initialize the column vector bCOL. Normalize the vector to 1.0
C   2) form the matrix (A-LAMBDA*I) using a special form if the
C      eigenvalue is complex
C   3) solve for x in (A-LAMBDA*I)x = b
C   4) substitute x for b and goto step 3 unless |x-b| < EPS
C-----
C
C       parameter (MO - 24)
C       integer IERR, MOI, NP
C       integer*4 IDUM
C       real*8 A(MOI,MOI), EVECTR(MOI), EVECTI(MOI), WI, WR
C       real*8 A_LU(MO,MO), BCOL(MO), BC_TEMP(MO), AT(MO,MO),

```

```

*      BNEW(MO), C(MO,MO), P, SUM, BC2(MO), BC3(MO)
integer INDX(MO)
real*4 XRAND
C
  if ((NP .gt. MO) .or. ((2*NP .gt. MO) .and. (WI .ne. 0.0D0)))then
    write(*,*) ' The input array is too large'
    IERR = -1
    return
  endif
C
C Initialize the column vector BCOL
C
  IDUM = -1
  SUM = 0.0D0
  do 10 I = 1, NP
    XRAND = RANO(IDUM)
    BCOL(I) = dble(XRAND)
    SUM = SUM + BCOL(I)*BCOL(I)
10  continue
  do 15 I = 1, NP
    BCOL(I) = BCOL(I)/sqrt(SUM)
15  continue
C
C form the matrix A - LAMBDA*I
C
  do 20 I = 1, NP
    do 25 J = 1, NP
      A_LU(I,J) = A(I,J)
25  continue
    A_LU(I,I) = A(I,I) - WR
20  continue
  N = NP
  if (WI .ne. 0.0D0) then
    do 30 I = 1, NP
      do 35 J = 1, NP
        A_LU(I,J+NP) = 0.0D0
        A_LU(I+NP,J) = 0.0D0
        A_LU(I+NP,J+NP) = A_LU(I,J)
35  continue
        A_LU(I,I+NP) = WI
        A_LU(I+NP,I) = -WI
        BCOL(I+NP) = 0.0D0
30  continue
      N = NP*2
    endif
C
C Copy A_LU into AT for future use
C
  call MCOPY(A_LU, MO, MO, AT, MO, MO, N, N)
C
C Perform LU decomposition on the matrix A-LAMDA*I

```

```

C      call LUDCMP(A_LU, N, MO, INDX, P)
C
C      Iterate to find the eigenvector
C
C      do 60 J = 1,5
C          call MCOPY(BCOL, MO, 1, BC_TEMP, MO, 1, N, 1)
C          call LUBKSB(A_LU, N, MO, INDX, BCOL)
C
C      Normalize the result
C
C          SUM = 0.0D0
C          do 65 I = 1, N
C              SUM = SUM + BCOL(I)*BCOL(I)
65      continue
C          do 70 I = 1,N
C              BCOL(I) = BCOL(I)/sqrt(SUM)
70      continue
60      continue
C
C      Copy the eigenvector into the appropriate output vectors
C
C          do 125 I = 1, NP
C              EVECTR(I) = BCOL(I)
C              if (WI .ne. 0.0D0) then
C                  EVECTI(I) = BCOL(I+NP)
C              else
C                  EVECTI(I) = 0.0D0
C              endif
125      continue
C
C
C      Check the accuracy of the eigenvector and eigenvalue by forming
C      A*x - LAMDA*x. If both are correct the result should be zero.
C
C
C      Form the real error term:
C      A*real(eigenvector) - real(eigenvalue)*real(eigenvector)
C      + imaginary(eigenvector)*imaginary(eigenvalue)
C
C      call MMULS(A, MOI, MOI, NP, NP, EVECTR, MOI, 1, NP, 1,
C      *      BCOL, MO, 1)
C      call MSCLS(EVECTR, MOI, 1, BC_TEMP, MO, 1, NP, 1, -1.0D0*WR)
C      call MADDS(BCOL, MO, 1, BC_TEMP, MO, 1, BC2, MO, 1, NP, 1)
C      call MSCLS(EVECTI, MOI, 1, BC3, MO, 1, NP, 1, WI)
C      call MADDS(BC2, MO, 1, BC3, MO, 1, BC_TEMP, MO, 1, NP, 1)
C      if (WI .ne. 0.0D0) then
C
C      Form the imaginary error term:
C      A*imaginary(eigenvector) - imaginary(eigenvalue)*real(eigenvector)
C      - real(eigenvector)*imaginary(eigenvalue)

```

```

C
      call MMULS(A, MOI, MOI, NP, NP, EVECTI, MOI, 1, NP, 1,
*          BCOL, MO, 1)
      call MSCLS(EVECTR, MOI, 1, BC_TEMP, MO, 1, NP, 1, -WI)
      call MADDS(BCOL, MO, 1, BC_TEMP, MO, 1, BC2, MO, 1, NP, 1)
      call MSCLS(EVECTI, MOI, 1, BC3, MO, 1, NP, 1, -WR)
      call MADDS(BC2, MO, 1, BC3, MO, 1, BC_TEMP, MO, 1, NP, 1)
    endif
    return
  end

C
C*****
C*****
C
      SUBROUTINE C_EVAL(A, MOI, NP, EVALR, EVALI)
C
C-----
C      Subroutine C EVAL computes the complex eigenvalues of the array A.
C*****
C      INPUT DATA:
C
C      A      - the A array in the eigenvector calculation (REAL*8)
C      MOI    - the dimensioned size of the A array
C      NP     - the part of the A array used
C
C*****
C      OUTPUT DATA:
C
C      EVALR - the real part of the eigenvalues. The array has MOI elements.
C      EVALI - the imaginary part of the eigenvalues. The array has MOI
C              elements.
C      IERR  - an error code: set to 0 if no errors occurred
C              set to -1 if there were errors
C*****
C
C      PROCEDURE:
C      1) Copy the array into a temporary array AT
C      2) Balance the matrix using the subroutine BALANCE
C      3) Convert the balanced matrix into Hessenberg form
C      4) Compute the eigenvalues of the matrix which is in Hessenberg form
C-----
C
      parameter (MO = 15)
      integer IERR, MOI, NP
      real*8 A(MOI,MOI), EVALR(MOI), EVALI(MOI)
      real*8 AT(MO,MO)

C
      if (NP .gt. MO) then
        write(*,*) 'The input array is too large'
        IERR = -1

```



```
    return  
endif
```

C

```
call MCOPY(A, MOI, MOI, AT, MO, MO, NP, NP)  
call BALANC(AT, NP, MO)  
call ELMHES(AT, NP, MO)  
call HQR(AT, NP, MO, EVALR, EVALI)  
return  
end
```

```

C
C*****
C*****
C
      subroutine SOLVE_SS_OPT_C_EQNS(PHI, GAMMA, NS, NI, MOI,
*                                Q1, Q2, KGAIN, IPRINT)
C
C*****
C
C   Programmed by P.H. Laurich
C   NRC Hydraulics Lab
C   11-Nov-87
C
C   This program performs its calculation using double precision reals.
C   The matrix manipulation procedures can be found in the file
C   DPMAT.FOR, the procedures for solving systems of linear equations can
C   be found in DPLIN.FOR and the procedures for solving eigensystems can
C   be found in DPEIGEN.FOR
C
C   Program to compute the gain matrix K required to achieve the desired
C   system response for a constant coefficient system given in the form:
C
C       x(n+1) = PHIx(n) + GAMMAu(n)
C       y(n)    = Hx(n)
C
C   Procedure:
C
C   Step 1) Compute the Hamiltonian matrix which is given by:
C
C       
$$H_c = \begin{bmatrix} \Phi + \Gamma Q_2^{-1} \Gamma^T \Phi^{-T} Q_1 & -\Gamma Q_2^{-1} \Gamma^T \Phi^{-T} \\ -\Phi^{-T} Q_1 & \Phi^{-T} \end{bmatrix}$$

C
C   Step 2) find the eigenvalues of the hamiltonian matrix
C   Step 3) find the eigenvectors corresponding to the stable
C            eigenvalues
C   Step 4) compute  $S_\infty$  the eigenvalues and eigenvectors
C   Step 5) compute  $K_\infty$ 
C   Step 6) compute the open loop and closed loop eigenvalues
C
C*****
C   INPUT DATA:
C
C   PHI   - the PHI array
C   GAMMA - the GAMMA array
C   NS    - the order of the system
C   NI    - the number of inputs
C   MOI   - the dimensioned size of the arrays passed as input
C   Q1    - the weighting matrix for the system states
C   Q2    - the weighting matrix for the input

```

```

C   IPRINT- flag set to 1 if the eigenvectors are to be printed
C
C*****
C   OUTPUT DATA:
C
C   KGAIN - the steady state feedback gain values.
C*****
C   Set the parameter MO to the maximum order allowed for the system.
C   The number of outputs and the number of inputs to the system must
C   be less than or equal to the order of the system.
C
C   parameter (MO = 6, MO2 = 2*MO)
C   REAL*8 PHI(MOI,MOI), GAMMA(MOI,MOI), Q1(MOI,MOI), Q2(MOI,MOI)
C   REAL*8 PHI_INV(MO,MO), PHI_IT(MO,MO), GAMMA_T(MO,MO),
C   1      Q2_INV(MO,MO), T1(MO,MO), HC(MO2,MO2),
C   3      WR(MO2), WI(MO2), EVECTR(MO2), EVEC TI(MO2)
C   real*8 SINF(MO,MO), S1(MO,MO), S2(MO,MO), S3(MO,MO), XI(MO,MO2),
C   1      XI_T(MO2,MO), GI(MO,MO2), GI_T(MO2,MO), V(MO,MO),
C   2      W(MO), COL(MO), KGAIN(MO,MO), WRO(MO), WIO(MO),
C   3      WRC(MO), WIC(MO)
C   INTEGER I, J, K, NS, NI, NO, EORDER(MO), IPRINT
C
C   Ensure that the matrices passed fit in the matrices defined in
C   this subroutine.
C
C   if (NS .gt. MO) then
C       write(*,*) 'Error SOLVE_SS_OPT_C_EQNS: ',
C   *           'Input arrays are too large'
C       return
C   endif
C
C   Form the hamiltonian matrix HC. The form of the matrix is given
C   above. The matrices is formed by by concatenating the four corner
C   matrices which are formed first.
C
C   TOP LEFT - compute the inverse of Q2 and the transpose of the inverse
C               of PHI. The negative of the top right matrix is computed
C               in the process.
C
C   call MCOPY(Q2, MOI, MOI, T1, MO, MO, NI, NI)
C   call MINV(T1, Q2_INV, MO, NI)
C   call MCOPY(PHI, MOI, MOI, T1, MO, MO, NS, NS)
C   call MINV(T1, PHI_INV, MO, NS)
C   call MTRNS(PHI_INV, MO, MO, PHI_IT, MO, MO, NS, NS)
C   call MMULS(GAMMA, MOI, MOI, NS, NI, Q2_INV, MO, MO, NI, NI,
C   *           S3, MO, MO)
C   call MTRNS(GAMMA, MOI, MOI, GAMMA_T, MO, MO, NS, NI)
C   call MMULS(S3, MO, MO, NS, NI, GAMMA_T, MO, MO, NI, NS,
C   *           T1, MO, MO)
C   call MMULS(T1, MO, MO, NS, NS, PHI_IT, MO, MO, NS, NS,

```

```

      *          S3, MO, MO)
      call MMULS(S3, MO, MO, NS, NS, Q1, MOI, MOI, NS, NS,
      *          T1, MO, MO)
      call MADDS(T1, MO, MO, PHI, MOI, MOI, S2, MO, MO, NS, NS)
C
C TOP RIGHT -
C
      call MSCLS(S3, MO, MO, T1, MO, MO, NS, NS, -1.0D0)
      call MCOPY(T1, MO, MO, S3, MO, MO, NS, NS)
C
C BOTTOM LEFT -
C
      call MMULS(PHI_IT, MO, MO, NS, NS, Q1, MOI, MOI, NS, NS,
      *          T1, MO, MO)
      call MSCLS(T1, MO, MO, S1, MO, MO, NS, NS, -1.0D0)
C
C Put the Hamiltonian matrix together
C
      do 200 I = 1, NS
        do 205 J = 1, NS
          HC(I,J) = S2(I,J)
          HC(I,J+NS) = S3(I,J)
          HC(I+NS,J) = S1(I,J)
          HC(I+NS,J+NS) = PHI_IT(I,J)
205      continue
200      continue
C
C Compute the eigenvalues of the hamiltonian matrix.
C
      NS2 = 2*NS
      call C_EVAL(HC, MO2, NS2, WR, WI)
      do 500 I = 1, NS2
        write(*,2100) I, WR(I), WI(I)
2100      format(' Eigenvalue : ',I2,2(D11.4,2X))
500      continue
C
C Get the order of the eigenvalues
C
      write(*,*) 'Enter the order for the stable eigenvalues'
      read(*,*) (EORDER(I), I = 1, NS)
C
C Select the eigenvalues for which the eigenvectors are to be computed.
C
      do 505 I = 1, NS
        IND = EORDER(I)
        call C_EVECT(HC, WR(IND), WI(IND), MO2, NS2, EVECTR, EVECTI, IERR)
C
C Fill the S infinity matrix as the user specifies the eigenvalues
C
C
C Figure out where to store this eigenvector

```

```

C
      do 507 J = 1, NS2
        if(EORDER(J) .ne. IND) goto 507
        ICOL = J
        goto 508
507      continue
508      continue
C
C Place the eigenvector into the appropriate columns of XI and GI. The
C real part of the eigenvector is placed in column ICOL and the
C imaginary part is placed in column ICOL + NS, where NS is the system
C order.
C
      do 510 J = 1, NS
        XI(J,ICOL) = EVECTR(J)
        GI(J,ICOL) = EVECTR(J+NS)
        if (WI(IND) .ne. 0.0D0) then
          XI(J,ICOL+NS) = ERECTI(J)
          GI(J,ICOL+NS) = ERECTI(J+NS)
        else
          XI(J,ICOL+NS) = 0.0D0
          GI(J,ICOL+NS) = 0.0D0
        endif
510      continue
505      continue
C
C Print the eigenvector matrix
C
      if (IPRINT .eq. 1) then
        write(*,*) 'The eigenvector matrix'
        do 667 I = 1, NS
          write(*,6666) (XI(I,J), J = 1,NS2)
6666        format(' ',10(D11.4,1X))
667      continue
          do 668 I = 1, NS
            write(*,6666) (GI(I,J), J = 1,NS2)
668      continue
          endif
C
C Compute the S infinity matrix by using singular value decomposition to
C solve
C
      T      T      T
      XI SINF = GI
C
      call MTRNS(XI, MO, MO2, XI_T, MO2, MO, NS, NS2)
      call MTRNS(GI, MO, MO2, GI_T, MO2, MO, NS, NS2)
      call SVDGMP(XI_T, NS2, NS, MO2, MO, W, V)
      do 600 I = 1, NS
        call SVBKS(XI_T, W, V, NS2, NS, MO2, MO, GI_T(1,I), COL)
C
C Place the solution vector (COL) into the appropriate row of

```

```

C   SINF. This will form the transpose of SINF transpose at the
C   same time that the array is stored.
C
      do 605 J = 1, NS
        SINF(I,J) = COL(J)
605    continue
600  continue
C   write(*,*) 'S infinity matrix'
C   do 670 I = 1, NS
C     write(*,6666) (SINF(I,J), J = 1,NS)
C670  continue
C
C   Compute the gain matrix K infinity
C
      call MMULS(GAMMA_T, MO, MO, NI, NS, SINF, MO, MO, NS, NS,
*           ' S1, MO, MO)
      call MMULS(S1, MO, MO, NI, NS, GAMMA, MOI, MOI, NS, NI,
*           ' S2, MO, MO)
      call MMULS(S1, MO, MO, NI, NS, PHI, MOI, MOI, NS, NS,
*           ' S3, MO, MO)
      call MADDS(Q2, MOI, MOI, S2, MO, MO, S1, MO, MO, NI, NI)
      call MINV(S1, S2, MO, NI)
      call MMULS(S2, MO, MO, NI, NI, S3, MO, MO, NI, NS,
*           ' KGAIN, MO, MO)
C
C   Compute the eigenvalues of the closed loop poles (PHI - GAMMA*KGAIN).
C   First form the matrix PHI - GAMMA*KGAIN then calculate the
C   eigenvalues. Also compute the open loop eigenvalues for comparison.
C   This is only done if the user wishes to see them (IPRINT = 1).
C
      if (IPRINT .eq. 1) then
        call MMULS(GAMMA, MOI, MOI, NS, NI, KGAIN, MO, MO, NI, NS,
*           ' T1, MO, MO)
        call MSCLS(T1, MO, MO, S1, MO, MO, NS, NS, -1.0D0)
        call MADDS(PHI, MOI, MOI, S1, MO, MO, S2, MO, MO, NS, NS)
C
        call C_EVAL(PHI, MOI, NS, WRO, WIO)
        call C_EVAL(S2, MO, NS, WRC, WIC)
        write(*,*) '      Open Loop Eigenvalues      Closed Loop',
*           ' Eigenvalues'
        do 550 I = 1, NS
          write(*,2150) I, WRO(I), WIO(I), WRC(I), WIC(I)
2150      format(' ',I2,T5,2(D11.4,1X),3X,2(D11.4,1X))
550    continue
        endif
      return
    end

```

ESTIMATOR OUTPUT FILE

```

*
*      1-03-88      13:30:13
*
*****
* The discrete time system given by
*
*       $x(n+1) = \Phi x(n) + \Gamma u(n)$ 
*       $y(n) = Hx(n)$ 
*
* has the covariance matrices and observer gain matrices given below
*
*****
*
* SYSTEM ORDER      : 6
* NUMBER OF INPUTS  : 1
* NUMBER OF OUTPUTS : 2
* TIME STEP         : .010
* PHI MATRIX: ( 6x 6)
.63474D+00 .00000D+00 .00000D+00 .00000D+00 .00000D+00 .00000D+00
.26449D+08 .38321D+00 -.20085D+02 .60707D+07 .15182D+05 .14773D+07
.13988D+03 .27761D-06 .40885D-03 .50527D+01 -.99498D-01 .12282D+01
.16929D+01 .30021D-07 .53030D-05 .48931D+00 -.11013D-02 -.12428D+00
-.17320D+03 -.52550D-05 -.24874D-02 -.88410D+02 .68667D+00 -.21515D+02
-.60499D+00 -.27787D-07 -.27532D-04 -.49732D+00 .85470D-02 .87898D+00
* GAMMA MATRIX: ( 6x 1)
.79810D-05
.15292D+03
.16813D-02
.88095D-05
-.60086D-03
-.15203D-05
* H MATRIX: ( 2x 6)
.00000D+00 .10000D+01 .00000D+00 .00000D+00 .00000D+00 .00000D+00
.00000D+00 .00000D+00 .00000D+00 .10000D+01 .00000D+00 .00000D+00
*** END OF SYSTEM DEFINITION PORTION ***
* PROCESS NOISE COVARIANCES:
.10000D+01 .20000D+01 .30000D+01 .40000D+01 .50000D+01 .60000D+01
* MEASUREMENT NOISE COVARIANCES:
.50000D+03 .30000D-03
* OBSERVER GAIN: ( 6x 2)
.13967D-07 .14645D-01
.89824D+00 .59438D+07
.41063D-05 .66178D+01
.71664D-07 .59198D+00
-.14617D-04 -.73172D+02
-.77543D-07 -.95424D+00
* PROCESS NOISE COVARIANCES:
.10000D+02 .20000D+03 .10000D+01 .50000D+01 .20000D+01 .50000D+01
* MEASUREMENT NOISE COVARIANCES:
.50000D+03 .30000D-03

```

* OBSERVER GAIN: (6x 2)

.14491D-07	.11123D-01
.89134D+00	.59739D+07
.41789D-05	.62439D+01
.75508D-07	.56732D+00
-.14188D-04	-.76853D+02
-.95219D-07	-.84456D+00

KALMAN FILTER SIMULATION OUTPUT FILE

```

*
*      1-03-88      13:30:13
*
*****
* The discrete time system given by
*
*       $x(n+1) = \Phi x(n) + \Gamma u(n)$ 
*       $y(n) = Hx(n)$ 
*
* has the covariance matrices and observer gain matrices given below
*
*****
*
* SYSTEM ORDER      : 6
* NUMBER OF INPUTS  : 1
* NUMBER OF OUTPUTS : 2
* TIME STEP         : .010
* PHI MATRIX: ( 6x 6)
.63474D+00 .00000D+00 .00000D+00 .00000D+00 .00000D+00 .00000D+00
.26449D+08 .38321D+00 -.20085D+02 .60707D+07 .15182D+05 .14773D+07
.13988D+03 .27761D-06 .40885D-03 .50527D+01 -.99498D-01 .12282D+01
.16929D+01 .30021D-07 .53030D-05 .48931D+00 -.11013D-02 -.12428D+00
-.17320D+03 -.52550D-05 -.24874D-02 -.88410D+02 .68667D+00 -.21515D+02
-.60499D+00 -.27787D-07 -.27532D-04 -.49732D+00 .85470D-02 .87898D+00
* GAMMA MATRIX: ( 6x 1)
.79810D-05
.15292D+03
.16813D-02
.88095D-05
-.60086D-03
-.15203D-05
* H MATRIX: ( 2x 6)
.00000D+00 .10000D+01 .00000D+00 .00000D+00 .00000D+00 .00000D+00
.00000D+00 .00000D+00 .00000D+00 .10000D+01 .00000D+00 .00000D+00
*** END OF SYSTEM DEFINITION PORTION ***
* PROCESS NOISE COVARIANCES:
.10000D+01 .20000D+01 .30000D+01 .40000D+01 .50000D+01 .60000D+01
* MEASUREMENT NOISE COVARIANCES:
.50000D+03 .30000D-03
* OBSERVER GAIN: ( 6x 2)
.13967D-07 .14645D-01
.89824D+00 .59438D+07
.41063D-05 .66178D+01
.71664D-07 .59198D+00
-.14617D-04 -.73172D+02
-.77543D-07 -.95424D+00
AVERAGE CONTROL SIGNAL:
.11575901D+01
AVERAGE ABSOLUTE ERRORS:
.33448871D+04 .37066548D-03

```

* PROCESS NOISE COVARIANCES:

.10000D+02 .20000D+03 .10000D+01 .50000D+01 .20000D+01 .50000D+01

* MEASUREMENT NOISE COVARIANCES:

.50000D+03 .30000D-03

* OBSERVER GAIN: (6x 2)

.14491D-07 .11123D-01

.89134D+00 .59739D+07

.41789D-05 .62439D+01

.75508D-07 .56732D+00

-.14188D-04 -.76853D+02

-.95219D-07 -.84456D+00

AVERAGE CONTROL SIGNAL:

.11575901D+01

AVERAGE ABSOLUTE ERRORS:

.32332754D+04

.36537460D-03

END

09-08-88

FIN