DEPARTMENT OF AEROSPACE ENGINEERING
COLLEGE OF ENGINEERING & TECHNOLOGY
OLD DOMINION UNIVERSITY
NORFOLK, VIRGINIA 23529

EXTENSION TO THE DYNAMIC MODELING OF THE LARGE ANGLE MAGNETIC SUSPENSION TEST FIXTURE

By

Lucas E. Foster

and

Dr. Colin P. Britcher, Principal Investigator

Progress Report
For the period November 1, 1994 through April 30, 1995

Prepared for
National Aeronautics and Space Administration
Langley Research Center
Hampton, VA 23681-0001

Under
Research Grant NAG-1-1056
Dr. Nelson J. Groom, Technical Monitor
GCD-Spacecraft Controls Branch

Submitted by the
Old Dominion University Research Foundation
P.O. Box 6369
Norfolk, VA 23518

May 1995

Extension to the Dynamic Modeling of the Large Angle Magnetic Suspension Test Fixture

By

Lucas E. Foster

B.S. May 1992, Old Dominion University

A thesis submitted to the faculty of Old Dominion University in partial fulfillment of the requirement for the degree of

Master of Science

in

Mechanical Engineering

Old Dominion University

May 1995

Approved By:

Colin P. Britcher

Brett Newman

Jenkuang Huang

ABSTRACT

Extension to the Dynamic Modeling of the

Large Angle Magnetic Suspension Test Fixture

Lucas E. Foster

Old Dominion University

The Large Angle Magnetic Suspension Test Fixture (LAMSTF) is a laboratory scale proof-of-concept system. The configuration is unique in that the electromagnets are mounted in a circular planar array. A mathematical model of the system had previously been developed, but was shown to have inaccuracies. These inaccuracies showed up in the step responses. Eddy currents were found to be the major cause of the modeling errors. In the original system, eddy currents existed in the aluminum baseplate, iron cores, and the sensor support frame. An attempt to include the eddy current dynamics in the system model is presented. The dynamics of a dummy sensor ring were added to the system. Adding the eddy current dynamics to the simulation improves the way it compares to the actual experiment. Also presented is a new method of determining the yaw angle of the suspended element. From the coil currents the yaw angle can be determined and the controller can be updated to suspend at the new current. This method has been used to demonstrate a 360 degree yaw angle rotation.

ACKNOWLEDGMENTS

The author of this paper would like to thank Nelson J. Groom of NASA Langley who is the technical monitor for the LAMSTF project Grant NAG-1-1056, Dr. Colin Britcher of Old Dominion University, and Tom Britton of Lockheed, for their assistance in the completion of this thesis.

TABLE OF CONTENTS

LIST OF TABLES

TABLE

LIST OF FIGURES

Figure

Figures Continued

Nomenclature

A, B, C, D	state equation matrices
A_{yaw}, B_{yaw}, C_{yaw}	coefficients for the yaw function
a, b	parameters in the experimental equation
α	parameter relating L to L_e
β	parameter relating L_m to the other inductances
$\vec{B}$	field vector
$[\delta B]$	field gradient matrix
$\vec{F}$	force vector
g	acceleration of gravity
$\vec{H}$	magnetic field
$\vec{I}$	coil currents
$\vec{I}_o$	initial offset currents
I_{max}	maximum coil current
[K]	field coefficients
L_m	mutual inductances
L	self inductance
$\vec{M}$	magnetic vector
M	mixer matrix
m_c	mass of the core
R	resistance of the eddy current loop
ρ	resistitivity
S2P	sensor to position matrix
s	laplace variable
$\vec{T}$	torque vector

$\vec{T}_m$	coordinate transformation matrix
V	volume of the core
$\vec{V}$	velocity vector
W	weighting matrix
$[\Omega]$	angular rotation rate matrix
∇	gradient operator
ϵ	variable in the yaw function
ϵ_{max}	correction factor for the yaw angle
μ_r	permeability of the material being used
μ_o	permeability of air
σ	resistance of the material being used
ω	field frequency
ψ	initial guess for yaw angle
ϕ	flux of the magnetic field
θ_y , θ_z	angular position of the core
θ_n	angular position of the coils
x , y , z	position of the core

subscripts

x, y, z	components along the relative axes
e	dealing with the eddy currents

CHAPTER ONE

Introduction to Magnetic Suspension Technology

Magnetic suspension is a relatively new science. While it is currently being used in some industrial and scientific applications, its use will increase dramatically in the future when the science is perfected.

Magnetic suspension applications can be divided into two groups according to the dimension of the air-gap, which is the distance between the object being suspended and the suspending magnets, as compared to the overall scale of the system. These groups are large-gap and small-gap. The difference in gap size is important because the assumptions made on a given problem will be different. An example of an assumption that can be used in small-gap system that is invalid for a large-gap system is that the fields are uniform across the air gap and have no gradients [1]. This assumption makes small-gap systems easier to analyze.

In a typical small-gap suspension system the gap can be as small as a fraction of a millimeter. Magnetic bearings and very accurate small object positioning systems are examples of small-gap systems. Magnetic bearings serve the same purpose as regular bearings, which is to hold a rotating shaft in place, yet they have several advantages over conventional bearings. The parts in the bearing do not touch so wear is minimal and friction is eliminated. These characteristics translate into longer life with less down-time, and higher

efficiency. Magnetic bearings are currently being used in a wide range of applications such as in pump stations along pipelines[2]. They are of great advantage because they may be in a remote location that makes performing routine maintenance costly. Magnetic bearings should not require any maintenance, and they have no reason to ever wear out.

One known application of small gap magnetic levitation is the magnetically levitated train [3,4]. They have several advantages over typical trains that run on rails. They are faster, quieter, more efficient, and smoother. These characteristics are all the result of removing the friction associated with wheels and the rumblings of all the mechanical parts in an ordinary train. They are typically faster because of two reasons. The first reason is that the train only has to overcome the force of air resistance. The second reason is in the design of the track. The track can be designed in such a way as to be safer with respect to derailing as compared to conventional railroad tracks, and the maglev track can be sloped to make banked turns allowing higher speeds than would be wise in a conventional train. Another advantage of a magnetically levitated train is that the train is powered magnetically allowing the power to come from an energized track. This cuts the weight of the train by eliminating the engine, a substantial amount of weight, making it even more efficient.

Other types of small gap suspension systems include accurate moving systems that are used in the manufacture of integrated circuits [5]. In these applications being able to move a circuit board a few Angstroms with accuracy is important. Another advantage in using magnetic levitation is the ability to isolate the board from machine vibration

Large-gap magnetic suspension systems are extremely varied in their applications. For a system to be classified as a large gap system, the gap may

be as little as 0.1 meters, for a small system, or much larger. Some of the uses for large gap systems include pointing, vibration isolation, and wind tunnel model support applications.

Pointing systems have a suspended element that can be controlled about different axes to aim a payload [6]. Pointing can be used by telescopes mounted on space stations that are in orbit. The telescopes need to be able to track a star across the sky, and by suspending the telescope it can track the star smoothly and accurately.

Vibration isolation systems are used in microgravity experiments where vibrations from equipment not related to the experiment would invalidate the experiment [7]. Vibration can be transmitted to the experiment through the support structure and umbilicals, or the experiment itself may create vibration. Using levitation the experiment no longer touches the vibrating platform and the other sources of vibrations can be actively eliminated.

Magnetic Suspension and Balance Systems (MSBS) are used in wind tunnels. In most ordinary wind tunnels, the models are suspended by fixing them to the walls of the tunnel through the use of a model mount called a sting. The problem is that the sting can alter the airflow around the model making the experiment invalid [8]. The requirement of the model to be altered to accept the sting may also make the experiment invalid. However, by suspending the model magnetically, the need for a sting is eliminated along with a major source of errors. The amount of force on the model can be derived from measuring the currents going through the coils and subtracting them from the currents used when the model is being suspended without any external force. A computer, which is most likely controlling the wind tunnel anyway, can determine how much force, such as drag or lift, is on the model.

There are some magnetic bearings and large gap suspension systems that are configured so they are stable without any type of active controls [9]. However, most practical applications of magnetic suspension technology require an active controller. An active controller measures the position of the object being suspended and develops an error signal indicating how far from the datum position it is. Based on this error signal the controller changes the currents in the coils to develop forces on the suspended object to push or pull it back to its datum position. Depending on the configuration the controller may have to update the currents fifty times per second or more.

Magnetic suspension can be achieved in a number of different ways. The method used by magnetic bearings relies on the fact that ferromagnetic material is attracted to a magnet. The shaft is made from a ferromagnetic material like iron or steel, and it is surrounded by electromagnets and position sensors at the location of the bearing. The suspension is achieved by attractive forces pulling the shaft back to the center.

Other suspension systems, the Large Angle Magnetic Suspension Test Fixture (LAMSTF) in particular, rely on magnetic interaction between a permanent magnet that is the suspended element and the electromagnetic coils. LAMSTF is a set of five electromagnets mounted in a flat planar array and will be described in more detail in chapter two. The suspended element is a magnet with its north and south poles at the ends, and the suspension is maintained by producing an electromagnetic south pole under the south pole and an electromagnetic north pole under the north pole of the suspended element. The coils are used to change the fields and the gradients of the field in subtle ways to keep the suspended element from falling out of suspension. For LAMSTF the forces are all repulsive, but this is not true in general. In most other applications

electromagnets are placed overhead to provide attractive forces as well as repulsive forces.

Chapter three develops the analytical model of the system, but this model has been shown to have inaccuracies causing the dynamics of the suspended element to be different from what is expected. A large portion of the system errors are due to the eddy currents in the system. Chapter four attempts to correct the errors by developing a way to mathematically describe the eddy currents and add them to the system.

CHAPTER TWO

The Large Angle Magnetic Suspension Test Fixture

The LAMSTF was constructed as a proof-of-concept system for a unique hardware configuration. The hardware configuration is unique for several reasons. The coil arrangement, five coils of the same size set in a horizontal circular arrangement, suspends the element using only repulsive forces. This form of suspension is called levitation, but the terms suspension and levitation will be used interchangeably. Most other large gap systems use a combination of attractive and repulsive forces to suspend. The LAMSTF system configuration is also unique from a controls standpoint because there is a lack of natural decoupling. In most other systems the coils are arranged so that each coil affects only one or two degrees-of-freedom, but in LAMSTF each coil has an effect on all of the degrees-of-freedom. This aspect makes control more difficult because the controller must decouple the different degrees-of-freedom. More information covering all aspects of LAMSTF including the controls, modeling and potential variations of the configuration can be found in [10]-[14].

The hardware used for the LAMSTF can be broken down into three different sections. The first section is a description of the plant. The plant includes the magnets, amplifiers, and the suspended element. The plant consists of the elements of the system that determine the dynamics. The other two systems are the position feedback sensors and the computer that is used for

control. These systems are interconnected by means of voltages. The sensors produce six voltages that are related to the position of the suspended element. These electrical voltages are used by the computer to find a set of five voltages that are related to the desired currents in the five electromagnets.

2.1 The Plant Description

The system consists of five electromagnets arranged in a planar array in the form of a circle under the suspended element. The coils are arranged symmetrically at a radius of 13.77cm. This configuration is shown in Figure 2.1. The coils have an outer diameter of 15 cm, and an inner diameter of 8.5 cm. They are wet wound using epoxy on bakelite forms with 509 turns of AWG 10 wire. In order to keep them stable while they are running a coating of epoxy has also been applied to the outside of the coils. In the center of each coil is an iron core. The purpose of the core is to boost the output of the coils so they can suspend the element using reasonable currents. The iron core is as long as the electromagnet with an outer diameter of 7.5 cm. The core is not laminated, which is common practice to reduce eddy currents, and this poses problems that will be explained in the section on eddy current modeling.

The electromagnets are driven by amplifiers that take a voltage between $\pm$ 10 volts. They amplify the voltage by -3, and they have a output current between $\pm$ 30 amperes. For safety reasons the amplifiers are connected to temperature probes located inside the coils. If a coil reaches 160 degrees all of the amplifiers are shut off automatically. This is to reduce the risk of damaging the coils, and it takes about 30 minutes of solid operation to produce that temperature.

Under the current configuration, the system can control the model in five degrees-of-freedom, and they are x, y, z, translations and pitch and yaw rotations as shown in Figure 2.2. The system has no control in the roll degree-of-freedom. As a result of this, during suspension the element will rock freely until it is stopped by physically constraining the rolling motion. Another system is under development that has more coils for the purpose of controlling the 6^{th} degree-of-freedom. This system is also a prototype proof-of-concept system that uses repulsive forces.

2.2 The Position Sensors

The sensors that are currently used on LAMSTF are light blocking position sensors. They have two components, a light emitting component, and a receiving component. The emitter discharges an infrared laser that has a rectangular cross-section area that is about .5x.5 inches. This beam has a uniform amount of light spread out across its cross section. The detector translates the amount of light reaching it to a voltage. When the suspended element is floating at its center or zero position it will block half of the light beam. Then, when the model moves around, the amount of light will change and hence the amount of voltage coming from the detector will change. This voltage typically changes from one for a fully blocked beam to five volts for a clear beam.

In an area of high magnetic fields that are constantly changing, such as when the element is being suspended, the noise induced in a long voltage run can become significant. This problem is solved by placing a current loop in the voltage line. This setup allows a large portion of the run to be a current based signal which is less prone to noise caused by magnetic field interference. Where

a voltage signal is required, the wires are twisted together to reduce the noise.

In LAMSTF, measurements from six sensors are used in locating the suspended element along the five degrees-of-freedom. The sensors are arranged in a circle in a planar array above the magnets (see Figure 2.3). Sensors 1 and 2 are used to find z position and pitch. Sensors 3, 4, 5, and 6 are coupled to find the x, y, and yaw position. After the voltages are brought into the computer via the analog-to-digital converter board, they are multiplied by the sensor decomposition matrix S2P in the following equation.

$$\text{Position} = \text{S2P} * \text{Sensor Voltages}$$

$$\text{S2P} = \begin{bmatrix} 0 & 0 & .353 & -.353 & -.353 & .353 \\ 0 & 0 & -.353 & .353 & -.353 & .353 \\ .5 & .5 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -.5 & -.5 & .5 & .5 \end{bmatrix}$$

After the multiplication is performed, there are five numbers which represent the five degrees-of-freedom x, y, z, pitch, and yaw. It is these five numbers which are used by the controller as inputs.

Currently the sensors are the limiting factor in large angle control of the model. The sensor range depends on the size of the model. The sensors measure one diameter of the model in translation. This is because if the model is high enough to allow light to show under the model, as in Figure 2.4, the sensors will indicate the wrong position, and the controller will not respond correctly, dropping the suspended element out of suspension.

2.3 The Computer's Role as a Controller

The computer's responsibility is to read in the sensor voltages and produce current output voltages. As the model moves about, the current in the coils must be constantly updated in order to maintain stable suspension. The rate at which the computer updates the currents can vary, but for a typical run 500 Hz is used. One of the additional goals of the project is to develop a controller that can run at 40 Hz, but this is difficult because it is very slow compared to the fastest unstable mode of oscillation. The computer uses three boards: a counter-timer board, an analog-to-digital converter board, and a digital-to-analog converter board.

For the controller routine to work it has to be run at a constant and known frequency. However, the program used to control the element has a graphical interface to the user which is very slow, and the amount of time it takes to run varies because it updates different amounts of the screen depending on the different settings. In order to make sure that the control routine is being run at a constant rate the program uses a hardware interrupt. A hardware interrupt is used by other things like the printer and mouse. In all, there are 256 different interrupts possible in the computer. Only a small portion of them, eight to be exact, can be used by a programmer. The purpose of the counter-timer board is to generate the interrupt, and the board can be set up from the program in an initialization routine. The program sets the board to generate a square wave signal at the desired frequency. This signal is fed into the interrupt pin on the board. Whenever a rising edge is sensed by the interrupt controller chip it stops whatever the computer is doing and runs the interrupt service

routine. For the purposes of the magnetic suspension the interrupt service routine is set to be the controller routine. After the controller has been run a signal is sent to the interrupt controller chip to tell it that the routine is finished and it can return to whatever the computer was doing before the interrupt occurred. This method allows the graphics, which are slow, to proceed at their own rate while the controller routine interrupts it at the known frequency needed by the controller.

The other boards are used to translate the voltages used by the amplifiers and the voltages from the sensors to numbers that can be used by the computer. The analog to digital converter takes the 0 to 10 volts output by the sensors and converts it to a number from 0 to 4095. This number is an integer that is proportional to the voltage. The digital-to-analog board repeats the process starting with a number and converting it to a voltage that is used by the amplifiers.

The interrupt service routine is the controller. It uses the six sensor voltages to produce the five positions. The controller dynamically compensates the five position signals with lead or lag to generate the five coil currents. For the controller used in this thesis, the coil currents are updated 500 times a second.

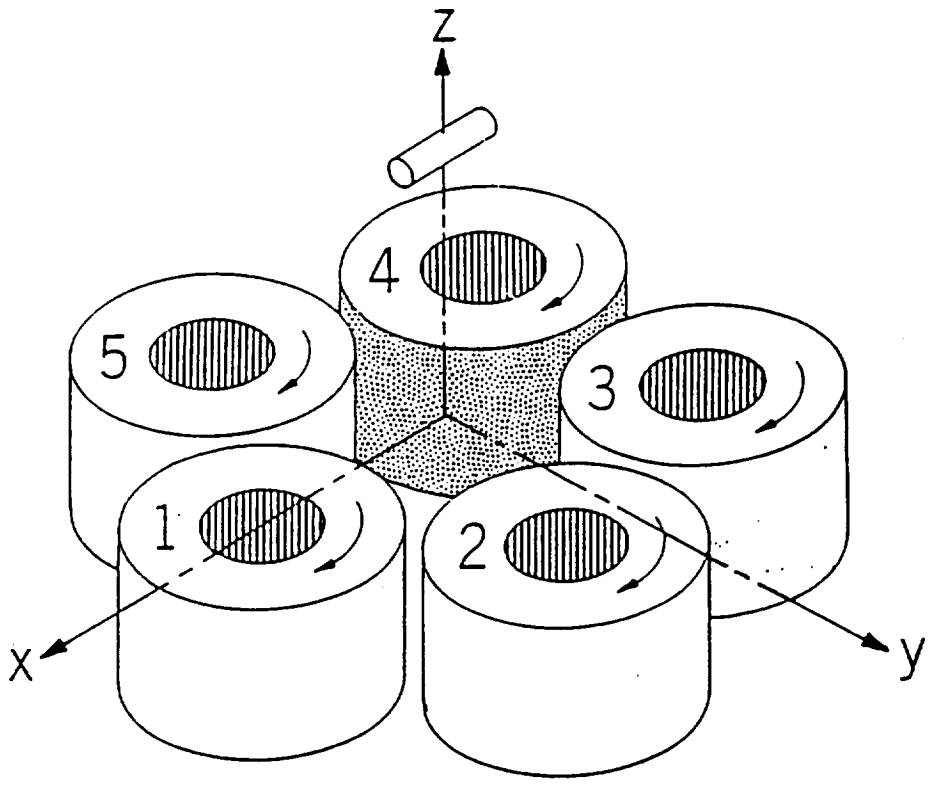


Figure 2.1 Coil Configuration

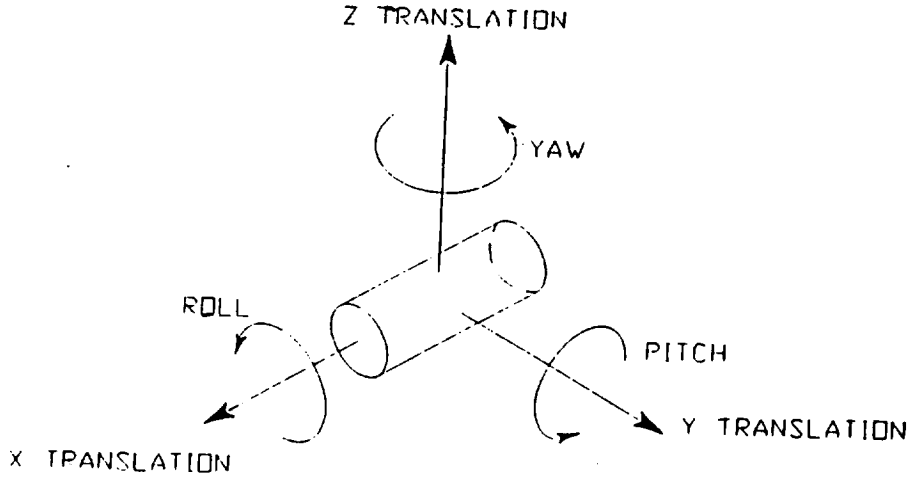


Figure 2.2 The Axis System

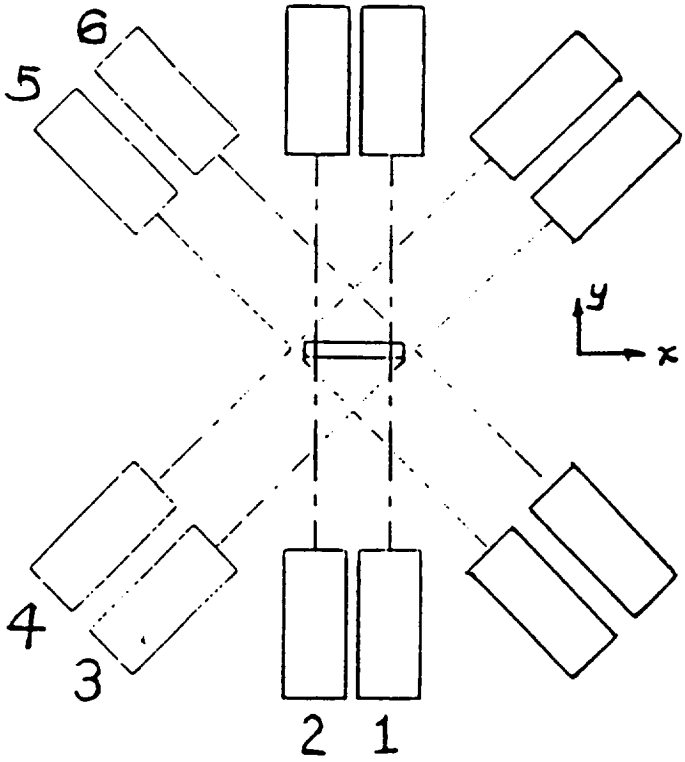


Figure 2.3 Sensor Configuration

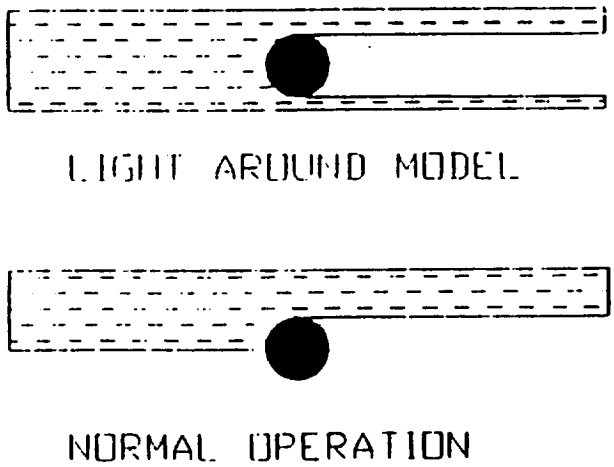


Figure 2.4 Sensor Saturation

CHAPTER THREE

General Dynamic Modeling

An accurate mathematical model of the system is necessary to create a control law that will be successful in suspending the element. Utilizing the mathematical model, a potential controller response is simulated to access the closed-loop stability and performance. An iterative design process is used to eliminate any deficiencies uncovered in the simulation. The system that is being modeled is the plant as described in Chapter 2.

The general procedure for developing the system model starts with the basic equations. These basic equations are linearized and put into a useful form to be used in simulation.

3.1 The Coordinates

There are two different coordinate systems used to make the analysis simpler. Both of the coordinate systems are typical right handed orthogonal systems. The first system is called the inertial coordinates. They are fixed with respect to the coils, and the origin is at the zero position of the suspended element. The second system is called core coordinates, and they are allowed to move around as the suspended element moves. The x axis is along the length of the suspended element with the y axis horizontal and the z axis vertical.

When the model does not show any translation or rotation from its zero position the two coordinate systems are as shown in Figure 3.1. Then as the suspended element moves around the core coordinates track the motion. The core coordinates are used to find the torques and forces on the model that result from the different fields and gradients. The inertial coordinates are used to find the fields and gradients. In actual use the field and gradients will be expressed in inertial coordinates and transformed to core coordinates. Transposing from one coordinate system to the other is a simple matter of developing a transformation matrix, based on the location of the model. In general operation, the coordinate system will be very close together due to the fact that all of the step responses will be kept to small angles and translations. In this situation, transformation will not be necessary. However, one degree of freedom will undergo large angular displacement and it is the yaw angle. The system has demonstrated a yaw angle of 360 degrees. If the z axis remains coincident with the inertial z axis then the transformation matrix will look like:

$$T_m = \begin{bmatrix} \cos(\theta_Z) & \sin(\theta_Z) & 0 \\ -\sin(\theta_Z) & \cos(\theta_Z) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

where θ_Z is the yaw angle. This transformation matrix will transform field vectors and velocity vectors. Transforming a derivative between the inertial coordinate system and the core coordinate system is not as easy. This is because the core coordinates are allowed to accelerate. The equation for transforming the force and torques, the derivatives of linear and angular momentum, is as follows:

$$\vec{F} = m_c \frac{d\vec{V}}{dt} + \vec{\Omega} \times m_c \vec{V}$$

where $\vec{V}$ is the velocity of the core , m_c is the mass of the core, and $\vec{\Omega}$ is the angular rotation rate of the moving core coordinates with respect to the inertial coordinates and it is expressed in matrix form as

$$[\Omega] = \begin{bmatrix} 0 & -\Omega_{\vec{z}} & \Omega_{\vec{y}} \\ \Omega_{\vec{z}} & 0 & -\Omega_{\vec{x}} \\ -\Omega_{\vec{y}} & \Omega_{\vec{x}} & 0 \end{bmatrix}$$

This is the typical procedure for developing equations in an accelerating coordinate system such as the core coordinates. For the LAMSTF system the accelerations will, under most circumstances, be small enough to be neglected. This assumption is made to greatly simplify the development of the governing equation.

3.2 The Basic Equations

The torques and forces due to the magnet fields are:

$$\vec{T} = \int_V (\vec{M} \times \vec{B})\, dV$$

$$\vec{F} = \int_V (\vec{M} \times \nabla)\, \vec{B}\, dV$$

These two equations are used to solve for the linear accelerations and the angular acceleration. $\vec{T}$ and $\vec{F}$ are the torque and force vectors, $\vec{M}$ is the

magnetic vector, and $\vec{B}$ is the field vector. The integrals are evaluated over the volume of the core. The first assumption is that the equations can be rewritten as:

$$\vec{T} = V\ (\vec{M} \times \vec{B})$$

$$\vec{F} = V\dot{}(\vec{M} \bullet \nabla)\ \vec{B}$$

This assumption treats the field like a constant and is possible because throughout the core the fields are symmetric functions about the suspension point. The modeling errors introduced by the previous equations are constant values, and they are taken care of in the $V\vec{M}$ term, which is not capable of being measured with accuracy.

At this point the fields are in the inertial coordinates and the magnetic vector is in core coordinates. This is corrected by using the transformation matrix, as described earlier, in the equations as follows:

$$\vec{T} = V\ \vec{M}\ [T_m]\ \vec{B^T}$$

$$\vec{F} = V\ [T_m]\ [\partial B]\ [T_m]^{-1}\ \vec{M}$$

These equations are used to solve for the linear and rotational accelerations as shown below.

$$\dot{\vec{\Omega}} = \frac{1}{[I_c]}\ V\ \vec{M}\ [T_m]\ \vec{B}$$

$$\dot{\vec{V}} = \frac{1}{m_c}\ V\ [T_m]\ [\partial B]\ [T_m]^{-1}\ \vec{M}$$

These equation will be used to develop the A and B matrices of the linear state space representation of the system. The equation for velocity in x will be developed here. All of the other degrees of freedom are developed in the same fashion with a more complete development found in Reference [16]. The equation for $\dot{V}_x$ is as follows.

$$\dot{V}_x = \frac{1}{m_c} V M_{\vec{x}} \left(B_{xx} + 2\,\theta_z\, B_{xy} - 2\,\theta_y\, B_{xz} \right) + \theta_y\, g$$

A Taylor series expansion is performed on V_x resulting in:

$$\dot{V}_x = \dot{V}_x \mid_o + \frac{\partial \dot{V}_x}{\partial X} \mid_o \delta X$$

where X is the state variable in the state space representation.

$$X = \left(\theta_y \;\; \theta_z \;\; x \;\; y \;\; z \;\; \Omega_y \;\; \Omega_z \; V_x \; V_y \;\; V_z \right)^T$$

A linear approximation for the change in velocity along x is then:

$$\delta V_x = \frac{\partial V_x}{\partial X} \; \delta X$$

When taking the partial of the equation, the field terms $\vec{B}$ are functions of x, y, z, and the coil currents. Two other assumptions are made to get to the next step. The first is that the θ_y and θ_z are small angular displacements. This assumption is reasonable due to the fact that the sensors can only sense a small angle. The second assumption is that products of small numbers, such as $\theta_y \delta x$, are negligible. The resulting δV_x is the equation actually modeled in the state space equation below.

$$\delta\dot{X} = A*\delta X + B*\delta I$$

Where δI is the input currents that are used to create the fields. The B is not to be confused with the fields and gradients $\vec{B}$ which is denoted as a vector or with a direction subscript. The A matrix in the previous equation is:

$$A = W * \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & -B_{xz} & 0 & -B_{xz} & -B_{yz} & -B_{zz} \\ 0 & 0 & 0 & 0 & 0 & 0 & -B_x & B_{yz} & -B_{yy} & B_{yz} \\ 0 & 0 & 0 & 0 & 0 & \left(\frac{m_c g}{VM_x} 2B_{xx}\right) & 2B_{xy} & B_{xxx} & B_{xxy} & B_{xxz} \\ 0 & 0 & 0 & 0 & 0 & B_{yz} & (B_{yy}-B_{xx}) & B_{xyx} & B_{xyy} & B_{xyz} \\ 0 & 0 & 0 & 0 & 0 & (B_{xx}-B_{zz}) & B_{yz} & B_{xzx} & B_{xzy} & B_{xzz} \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Where W contains the constants from the equations along its diagonal.

$$W = \begin{bmatrix} \frac{V\,M_x}{I_c} & 0 & \cdots & & & & \cdots & 0 \\ 0 & \frac{V\,M_x}{I_c} & \ddots & & & & & \vdots \\ \vdots & \ddots & \frac{V\,M_x}{m_c} & & & & & \\ & & & \frac{V\,M_x}{m_c} & & & & \\ & & & & \frac{V\,M_x}{m_c} & & & \\ & & & & & 1 & & \\ & & & & & & 1 & \\ & & & & & & & 1 \\ \vdots & & & & & & \ddots & 1 \\ 0 & \cdots & & & & & \cdots & 0 \end{bmatrix} \begin{matrix} \\ \\ \\ \\ \\ \\ \\ \vdots \\ 0 \\ 1 \end{matrix}$$

The B matrix is made up of terms that are the inputs of the system. However, before the equation can be used the current from the coils must be introduced into the field inputs, $\vec{B}$. This is done by finding the K values in the following equation.

$$\vec{B} = [K]\,\vec{I}$$

K is in units of gauss per amp. A K value exists for each different field B_x, B_y, and B_z, and all of the gradients. This group of K values will become the B matrix in the state space equations. Following the same logic as before, a Taylor series expansion for the velocity can be written as:

$$V_x = V_x + \frac{\partial V_x}{\partial I}\,\delta I$$

Where δV_X is:

$$\delta V_X = \frac{\partial V_X}{\partial I} \delta I$$

The δV_X is the final equation that is in the B matrix. The B matrix is also multiplied by the W matrix that holds the constants. The B matrix is as follows:

$$B = W * \begin{bmatrix} -K_{z_1} & -K_{z_2} & -K_{z_3} & -K_{z_4} & -K_{z_5} \\ K_{y_1} & K_{y_2} & K_{y_3} & K_{y_4} & K_{y_5} \\ K_{xx_1} & K_{xx_2} & K_{xx_3} & K_{xx_4} & K_{xx_5} \\ K_{xy_1} & K_{xy_2} & K_{xy_3} & K_{xy_4} & K_{xy_5} \\ K_{xz_1} & K_{xz_2} & K_{xz_3} & K_{xz_4} & K_{xz_5} \\ & \cdots & 0 & \cdots & \\ & \cdots & 0 & \cdots & \\ & \cdots & 0 & \cdots & \\ & \cdots & 0 & \cdots & \\ & \cdots & 0 & \cdots & \end{bmatrix}$$

The system that has been developed is a pure analytical model. This model has inaccuracies in it, and these inaccuracies show up in the step response of pitch. The damping of the step response in pitch is less in experiment than the simulated results predicted. A method of improvement for this problem is presented in Chapter 4.

3.3 System Analysis

Now that the model has been developed the next step is to analyze the system to find out what modes exist and which modes are stable or unstable. The values for K are calculated from OPERA, which is a software package used to numerically evaluate magnetic fields. The method that OPERA uses to calculate the fields is described in Chapter 5. That chapter will also cover how the field gradients are calculated. The numeric values for the A and B matrices for the LAMSTF are as follows.

$$A = W * \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 7.816e\text{-}3 & 0 & -9.25e\text{-}2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 7.816e\text{-}3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -9.25e\text{-}2 & 0 & 4.710e\text{-}1 & 0 & -2.366e\text{-}4 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 9.015e\text{-}1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -2.366e\text{-}4 & 0 & -8.614e\text{-}3 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$B = W * \begin{bmatrix} 4.071e1 & 4.071e1 & 4.071e1 & 4.071e1 & 4.071e1 \\ 0 & 9.527e1 & 5.8899e1 & -5.8899e1 & -9.5278e1 \\ 2.3010e\text{-}1 & -1.5935e\text{-}1 & 8.1846e\text{-}2 & 8.1846e\text{-}2 & -1.593e\text{-}1 \\ 0 & 1.268e\text{-}1 & -2.0525e\text{-}1 & 2.0525e\text{-}1 & -1.268e\text{-}1 \\ -2.8869e\text{-}1 & 8.9161e\text{-}2 & 2.3356e\text{-}1 & 2.3356e\text{-}1 & -8.9161e\text{-}2 \\ & \cdots & 0 & \cdots & \\ & \cdots & 0 & \cdots & \\ & \cdots & 0 & \cdots & \\ & \cdots & 0 & \cdots & \\ & \cdots & 0 & \cdots & \end{bmatrix}$$

$$W = \begin{bmatrix} 4.3308e5 & 0 & \cdots & & & & & \cdots & 0 \\ 0 & 4.3308e5 & \ddots & & & & & & \vdots \\ \vdots & \ddots & 1.0602e2 & & & & & & \\ & & & 1.0602e2 & & & & & \\ & & & & 1.0602e2 & & & & \\ & & & & & 1 & & & \\ & & & & & & 1 & & \\ & & & & & & & 1 & \ddots & \vdots \\ \vdots & & & & & & & \ddots & 1 & 0 \\ 0 & \cdots & & & & & & \cdots & 0 & 1 \end{bmatrix}$$

Using these matrices the eigenvalues for the plant model are:

$$\text{Eigenvalues} = \begin{bmatrix} 59.2596+0i \\ -59.2596+0i \\ 0+7.9717i \\ 0-7.9717i \\ 0+.9556i \\ 0-.9556i \\ 58.2942+0i \\ -58.2942+0i \\ 9.7762+0i \\ -9.7762+0i \end{bmatrix}$$

There are five different pairs of eigenvalues corresponding to five different modes of motion. Two of the motions which lie on the $j\omega$ axis are neutrally stable. The other three modes are very unstable and they will throw the suspended element out of suspension. Table 3.1 has all of the eigenvalues and what degrees-of-freedom they correspond to. The two most unstable modes are called the compass needle modes which include yaw and x. The reason for the highly unstable behavior is because when the model is suspending, there is a large B_X field component that is in the opposite sign as the magnetic vector in the suspended element. The B_X component is not used to control any degree-of-freedom, but it exist when the other field and gradient are created. The desire of the suspended element is to align itself with the B_X field causing the high frequency of the compass needle modes. Figure 3.2 has a diagram of the different modes of the suspended element as they relate to the motions of the model.

3.4 Initial conditions

The actual currents through the coils will be the offset current and the control current. Finding the initial condition is important for knowing what the offset current through the coils should be. The initial currents can be found using statics and the original two equations that the model was built upon. The only force on the model will be the force required to offset gravity. By setting all of the other forces and torques to zero the fields and gradients necessary to support the model can be found.

$$B_y = B_z = B_{xx} = B_{xy} = 0$$

$$B_{xz} = \frac{m_c\, g}{V\, M_x}$$

Using the field equation form before:

$$\vec{B} = [K]\, \vec{I}$$

The initial currents will be:

$$\vec{I}_o = [K]^{-1}\, \vec{B}^t$$

The initial coil currents are listed in Table 3.2. The currents in coils 2 and 5 are the same and the current in coil 3 and 4 are the same. This is due to

the symmetry of the circular planar configuration. Coils 1, 2, and 5 are all negative while 3 and 4 are positive. This is due to the fact that the magnetic vector of the suspended element is along the x axis.

Mode	Eigenvalue	Stability	Degrees-of-freedom
1	± 59.26 rad/s	Unstable	x, θ_y (Axial, Pitch)
2	± 7.972 i rad/s	Stable Oscillatory	x, θ_y (Axial, Pitch)
3	± 58.29 rad/s	Unstable	θ_z (Yaw)
4	± .0956 i rad/s	Stable Oscillatory	z (Vertical)
5	± 9.776 rad/s	Unstable	y (Lateral)

Table 3.1 The Eigenvalues of the System

Coil	Current (Amps)
1	-14.17
2	-4.38
3	11.47
4	11.47
5	-4.38

Table 3.2 Coil Currents

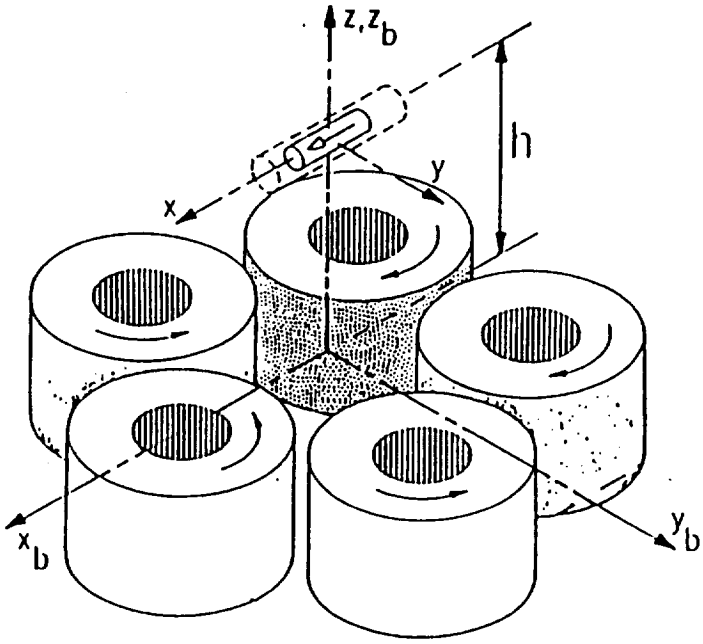


Figure 3.1 The Axis System

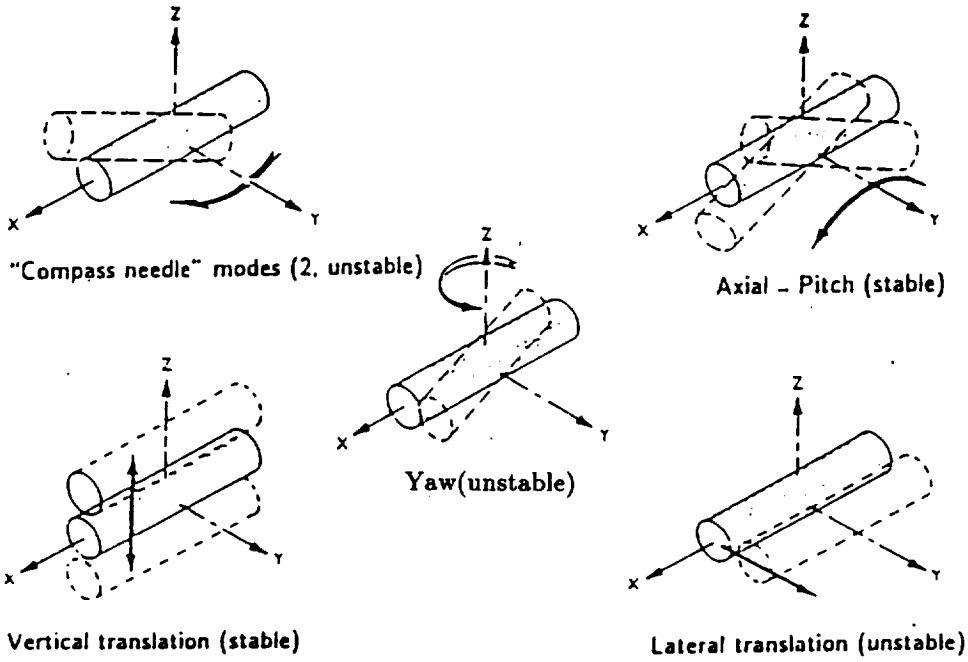


Figure 3.2 Modes of Motion

CHAPTER FOUR

Eddy Current Analysis

The discrepancy in the step response in pitch was believed to be due to eddy currents. In the original design of the system, aluminum was used as a base plate and as the support structure for the sensors. Iron cores that are not laminated were used as the core of the electromagnets in order to produce stronger fields with less current. All of this metal permits eddy currents to flow which affects the system plant giving the model errors.

Eddy currents are produced anytime a conducting material is placed in a fluctuating magnetic field. As the magnetic field changes, currents are developed in the metal that resist the change. The fields calculated for a given set of currents through the coils is accurate for a steady-state condition. However, when the current going through the coils is changing the field response is not immediate. The transient field at the suspension point is reduced by the eddy currents in the system. In the original system dynamic model the eddy currents were ignored. However, the model was sufficiently close to the actual system dynamics to allow an implemented controller design to suspend the model. Discrepancies in the transient behavior between the analytical model and the physical system still exist though, and correcting these deficiencies is the main focus of the research. By developing the eddy current analysis, the theoretical simulation will move closer to the actual responses.

In LAMSTF the eddy currents arise from 3 different places. It was suspected that the greatest effect came from the sensor support ring. This effect shows up most prominently when a step response in pitch is performed (see Figure 4.1). The aluminum base plate that the coils are mounted on has an eddy current associated with it, but it is believed that its effect is reasonably small. The iron cores are the most challenging to model accurately. The reason for this is explained later in the theoretical analysis. A figure of all the eddy currents is shown in Figure 4.2.

In most other magnetic suspension systems, eddy currents are minimized by design. Eddy currents need a closed path to travel or they cannot exist. The effects of the eddy currents can be reduced by electrically cutting the eddy current loops. Loops can be cut by using laminated cores and non-conducting inserts in the frame. Eddy currents can be avoided entirely by replacing the metal with a nonconductive material such as wood or fiberglass.

As was stated before, it was suspected that the most prominent eddy currents come from the sensor support frame. The sensors are mounted on a large aluminum ring that is placed over all of the magnets. The reason for pitch to be particularly affected is when a step response in pitch is performed, the amount of field passing through the ring has a large net change. Step responses in the other degrees-of-freedom are not as sensitive to the sensor ring eddy currents. This is due to the fact that when the step responses are performed the field is changed in such a way as to leave the net amount of field passing through the ring the same. For these reasons the step response in pitch will be used as a comparison between the original model and the updated model.

4.1 Theoretical Outline

The eddy currents will be modeled as a transformer with the secondary coil shorted out. This method will give a good first-order approximation of most eddy currents as long as they satisfy a few assumptions. The first assumption is that the skin depth, the depth that the eddy currents penetrate into the material, is much greater than the thickness of the metal. This also implies that the eddy currents are uniformly distributed throughout the metal. The skin depth varies according to the frequency that the field is being driven. As the frequency is increased the skin depth decreases. The frequency that the skin depth is calculated at in order to check this assumption is the highest frequency that will affect the system; for the case of LAMSTF the highest unstable open loop pole is about 20 Hz, to allow for a safety factor 100 Hz is used. The equation for skin depth is as follows[15]:

$$\text{skin depth} = \sqrt{\frac{2}{\mu_0 \, \mu_r \, \sigma \, \omega}}$$

where μ_r is the permeability of the material, μ_0 is the permeability of free space which is a constant, σ is the resistance of the material, and ω is the field frequency.

The large skin depth assumption is used throughout the following theoretical calculations, but considering the case of the iron cores as the frequency is increased the assumption loses its validity. The model's inaccuracies at high frequency may cause the theory to be off slightly at the frequencies of interest. The reason for the deviation at high frequencies is that

as the frequency is increased, the magnetic fields and eddy currents are forced to the outside of the core leaving the middle of the core free of magnetic fields and currents. This makes the uniform current assumption invalid. As the frequency gets very high the currents are reduced to surface currents along the outside of the core. This reduction in the amount of material that the eddy currents are flowing through increases the resistance of the loop, one of the parameters that was assumed to be constant in the equation for the eddy current dynamics that will be developed.

Figure 4.3 is used as a general reference for the analysis. All of the eddy currents use the same equation with different parameters as will be demonstrated later.

For the driven coil, the voltage across the terminals is:

$$V = IR + L\frac{dI}{dt} + L_{m_1}\frac{dI_{e1}}{dt} + L_{m_2}\frac{dI_{e2}}{dt} + \ldots$$

The first two terms deal with the primary coil itself without any eddy currents. Each of the other individual terms deal with one eddy current. L_{m_n} is the mutual inductance between the n^{th} eddy current loop and the driving coil, and I_{e_n} is the amount of current in the loop. The mutual inductance L_{m_n} is a constant. For each eddy current loop, the following equation can be used to eliminate I_{e_n} from the previous equation. The following equation does not include the effect of the other eddy currents on the eddy current being considered. If it were desired the equation could be amended to reflect the other eddy current loops, but for the sake of simplicity the effects of eddy currents on eddy currents has been assumed to be negligible.

For the eddy current loop the following equation applies:

$$0 = I_{e_1} R_{e_1} + L_{e_1} \frac{dI_{e_1}}{dt} + L_m \frac{dI}{dt}$$

where all terms noted with an e are the values in the eddy current loop itself. The last term is the mutual inductance and the current in the driving coil. By taking the Laplace transform of both equations and solving:

$$\frac{I}{V} = \left(\frac{1}{(R + Ls) - \frac{(L_{m_1} s)^2}{(R_{e_1} + L_{e_1} s)} - \ldots} \right)$$

Now:

$$\frac{I}{V} = \frac{1}{(R + Ls)}$$

is the normal transfer function for the terminal characteristics of the primary coil without any eddy currents. The other terms are the effects of the eddy currents.

To show that this is correct a special case is considered. The special case comes about in two different ways; the first is when the frequency is allowed to go to zero, and the second is the case where the resistance becomes very small. To make these points clear, two substitutions are made:

$$L = \alpha L_{e_1} \qquad \text{and} \qquad L_{m_1} = \beta \sqrt{L_{e_1} L}$$

The result of which is:

$$\frac{I}{V} = \left(\frac{1}{(R + Ls) - \frac{\beta^2 \alpha (L_{e_1} s)^2}{(R_{e_1} + L_{e_1} s)} - \ldots} \right)$$

When R_{e_1} is allowed to go to zero or as s becomes very large:

$$\frac{I}{V} = \left(\frac{1}{R + Ls\,(1-\beta^2)} \right)$$

This indicates that the secondary loop effectively "turns off" part of the primary coil if the secondary is reactive(non-dissipative).

The field at any point from a coil can be calculated by:

$$B_j = \sum_{z=1}^{q} K_{zj} I_z$$

where K_j is the field coefficient and q is the number of coils to be added together. Eddy currents are added in the same manner.

$$B_j = \sum_{z=1}^{q} K_{zj} I_z + \sum_{n=1}^{p} K_{e_{nj}} I_{e_n}$$

Taking the Laplace transform for one coil and eddy currents:

$$B_j = K_j I \left(1 - \sum_{n=1}^{p} \frac{K_{e_n} L_{m_n} s}{K_j (R_{e_n} + L_{e_n} s)} \right)$$

Where j is the direction of the field desired and p is the number of eddy currents that are to be included in the equation.

Now that the form of the equation has been developed the problem is to find values for the variables of a given eddy current loop. These variables can be calculated or variants of them can be derived from experimental method. The program used to calculate the variables is called VF/GFUN which is part of OPERA and it is distributed by Vector Fields Inc. VF/GFUN is a static field calculation program, but it is sufficient to find the variables that are needed in the above equations.

Finding the resistance of a current loop is a relative simple task that involves finding the area of the loop, its length, and its resistivity which is a property of the material. These properties are used in the following equation.

$$R = \rho\left(\frac{\text{Length}}{\text{Area}}\right)$$

Where R is the resistance, Length is the circumference of the loop, Area is the area of the cross-section of the loop.

For the purposes of finding the other parameters, the eddy current loops are treated like a small coil of the same size as the assumed loop size. The K_e value can be found from the equation

$$B_Z = K_Z\, I$$

where B_Z is found from OPERA when one coil or loop is turned on to one amp. For self-inductances, the eddy current loop is turned on to one amp and the field

going through the loop is integrated. The equation governing the voltage in the loop is:

$$V = L \frac{dI}{dt} = \frac{d\phi}{dt}$$

Therefore:

$$L = \frac{d\phi}{dI} = \frac{\phi}{I} = \frac{\int_0^\infty B_z \, dA}{I}$$

With the current at one volt, the integral becomes the inductance. For mutual inductances the driven coil will be the actual coil in the system.

The accuracy of the results obtained by this method varies greatly depending on the geometry of the eddy current. If the eddy current is of a known geometry such as the sensor ring, then the results will be reasonably accurate to use in the model. Unfortunately, when the geometry of the eddy current is not of an exact or known size then the accuracy of the results are degraded according to the amount of simplification necessary. For the plate the diameter of the eddy current loop is not known, but it is assumed to be approximately the size of the inner diameter of the driving coil. No assumption is required concerning the height of the eddy current coil because the height is equal to the thickness of the plate. For the iron core the true eddy current geometry changes with different frequencies with the thickness and the height being the unknowns. For the core, the calculations serve as a gross magnitude approximation for selecting the values from the experimental results.

There are magnetic field software tools that use numerical solution techniques, similar to OPERA, that will find where the eddy currents are flowing

in a piece of metal. ELEKTRA is one such program which is also marketed by Vector Fields, the same company producing OPERA.

The preceding section explained how the parameters are evaluated from pure mathematical modeling. The next section will explain how to find the parameters from the measured transfer functions. A comparison of the two methods will be at the end of the next section.

4.2 Experimental Procedure

Experimental determination of the various transfer functions is necessary, not only to validate the theoretical calculations, but to improve the variables in the equations that will be added to the system. The transfer function can be determined from the experimental Bode plots.

In order to get the experimental Bode plots, a Schlumberger transfer function analysis machine, model number 1250 was used. This analyzer drives a system with a sine wave, and the output of the system is fed back to the analyzer. In doing this, one point on the Bode plot is created. By stepping the sine wave through a given range of values, the frequency region of interest on the Bode plot can be generated. This data is stored in the machine as a file, which is then transferred to the computer via a GPIB connection. MATLAB is a mathematical analysis program that will be used to analyze the data.

In order to get the desired Bode plot from the system that is being tested, several instruments are used. The amplifiers used for the system are the same ones used for all of the tests. The sine wave is fed directly into the amplifier driving one coil. The configuration of the metal and the driving coil is changed in the test apparatus to isolate the effects of one eddy current. The field is

measured at a point with a F.W.Bell model number 9903 gaussmeter. The meter gives the field in one direction at one point using a Hall-effect sensor. The output of the Bell gaussmeter is an analog signal that is proportional to the amount of field at the probe. This signal is fed back to the transfer function analysis machine to form the input to output dynamic model.

The desired Bode plot has an input of coil current and an output of field. In reality, it is not possible to get the exact transfer function. Every piece of equipment adds its own unique transfer function to the final Bode plot. To deal with this problem, a baseline measurement is taken that includes all of the different transfer functions. This baseline is subtracted from all other data taken, leaving behind only the desired Bode plot. The baseline configuration is an air-cored coil with the field probe lined up along the axis of the coil. The probe was placed at the suspension height for all tests including the baseline. A plot of the baseline is shown in the Figure 4.4.

There are three different configurations that will be calculated and compared to experimental values. These three configurations are the iron core, the aluminum base plate, and the sensor ring. The configurations used for calculating and testing purposes are designed to isolate and identify the effects of one eddy current loop at a time. Two of the configurations, the aluminum base plate and the iron core, are exact representations of the corresponding eddy currents in the real system. However, the sensor frame is a very complex shape with many possible paths for eddy currents to travel. This will be modeled in the experiments by a horizontal aluminum ring. A picture of the system with the dummy sensor ring in place is shown in Figure 4.5. It is believed that this ring has the greatest effect on the system, and it is the easiest eddy current loop to calculate.

Using the experimental Bode plot and knowing the general form of the transfer function, it is possible to determine the parameters. This is done in MATLAB using a trial and error method. The general form of the transfer function that is used to get the parameters is as follows:

$$B_Z = K_j I\left(1 - \frac{a\,s}{1 + b\,s}\right)$$

The variables a and b are varied to get the resonant frequency and the phase shift at the resonant frequency correct. The variables for the experiment and the theory are given in appendix B. The a variable will shift the phase difference while the b variable will change the resonant frequency. Finding the variables will not give the same parameters that are in the theoretical outline, but they will give the transfer function that will be used. The figures comparing the transfer functions will determine the validity of the approximations invoked throughout the theory. For the case of the core, the assumption of large skin depth becomes inaccurate as the frequency becomes large. At large frequency, the phase of the core starts to roll off whereas the theory returns to zero. To take care of this problem a new theory will have to be developed.

The transfer functions given by the test data can be used to update the mathematical model of the system. In Figure 4.6 comparison of the test data, the theoretical modeling, and the experimental approximation are shown.

4.3 Comparison of Experimental Results

The results of the previous section are presented in Figure 4.5 through 4.8. The figures contain two plots, the first one is of magnitude vs. frequency

and the second one is the phase plotted against the frequency. Figure 4.6 shows the response of the eddy currents in the dummy sensor ring. This plot contains the curves for the experiment, the original theory, and the estimated theory. This plot is presented to show how close the theory is to the actual experiment. For the other plots the experimental and the estimated theoretical curves are shown without the original theory. This is because the differences between experiment and original theory are due to inaccuracies in the estimation in parameters such as resistance, and the shape of the eddy current. Figure 4.7 and Figure 4.8 are the effects of the eddy currents in the aluminum plate and the iron core. The estimated theory and the experiment agree reasonably well in the case of the aluminum plate and the dummy sensor ring. However, in the case of the iron core the agreement between estimated theory and experiment falls apart at the higher frequencies. The theory involved is not sufficient to explain these differences, but another theory is being developed that will better approximate the core response. That theory is still in its development stage and is beyond the scope of this report. Figure 4.9 is a response of all of the eddy currents in the system. Most of the error that exists in the plots is due to the discrepancy in the iron core modeling. This is given as an example of how several different eddy currents can be added together.

4.4 Incorporation into Dynamic Model

In the current system the aluminum plate has been replaced with a wood base plate, and the sensor frame has also been replaced with a wooden version. The only eddy currents left in the system are the currents in the iron cores, and these are necessary for the operation of the system. No attempt will be made to model the iron core due to the fact that the effect on the system is small, and a

limited number of eddy currents can be incorporated in the system using the method outlined here. For the purposes of this demonstration the dummy sensor ring will be the eddy current that is being modeled. This is because it has the most noticeable effect on the system. The eddy current causes the step response in pitch to be under damped.

To show this effect a new controller has been designed, based on a linear quadratic regulator controller, that is sensitive to the effects of the eddy current. This controller is used for the experimental pitch step responses, and it is also used for the simulated pitch step response. By using the same controller an amount of uncertainty is eliminated. Figure 4.10 shows a comparison between the experimental step in pitch with the dummy sensor ring in place and without the ring. By adding the ring the damping of the step response has been greatly reduced.

Figure 4.11 is the same step in pitch, but for this plot the responses were simulated using the MATLAB script file located in Appendix C. This script file contains a block titled eddy current effects, and that block adds the dynamics of the eddy currents to the plant model. The transfer function found to best describe the eddy current has been changed to a state-space representation by using the MATLAB tf2ss command. The dynamics of this state-space representation have been added together, using the MATLAB append command, to get a model with five inputs and five output. The five inputs represent the five coil currents from the controller, and the five outputs represent the five coil currents that have been updated with the eddy current dynamics. The updated eddy currents are fed to the plant model to find the simulated pitch response.

The change in step response for pitch, between the metal and non metal case, shown in Figure 4.11 is not as large a difference as in the case of the

experimental step responses, but the trend of making the step response less damped with the addition of the dummy sensor ring does appear. The difference in the magnitude of the change in damping is not very surprising. The theoretical model still has errors, and Figure 4.12 shows the step responses with out the dummy sensor ring for the simulation and the experiment. The step responses should be exactly the same. Their differences demonstrate the errors that remain in the model of the system. One of the known sources of these errors are the iron cores which were not added to the dynamics of the plant for the purposes of the simulation.

Gains can be incorporated in the model to make the experiment and simulation agree better. The improved simulated step response vs. the experimental step response is shown in Figure 4.13, again this plot is created without the dummy sensor ring in place. Using this model, which has less initial errors than the model that was used originally, the step response was simulated for both with and without the dummy sensor ring. This plot is shown in Figure 4.14, and it agrees better with the experimental measurements shown in Figure 4.10.

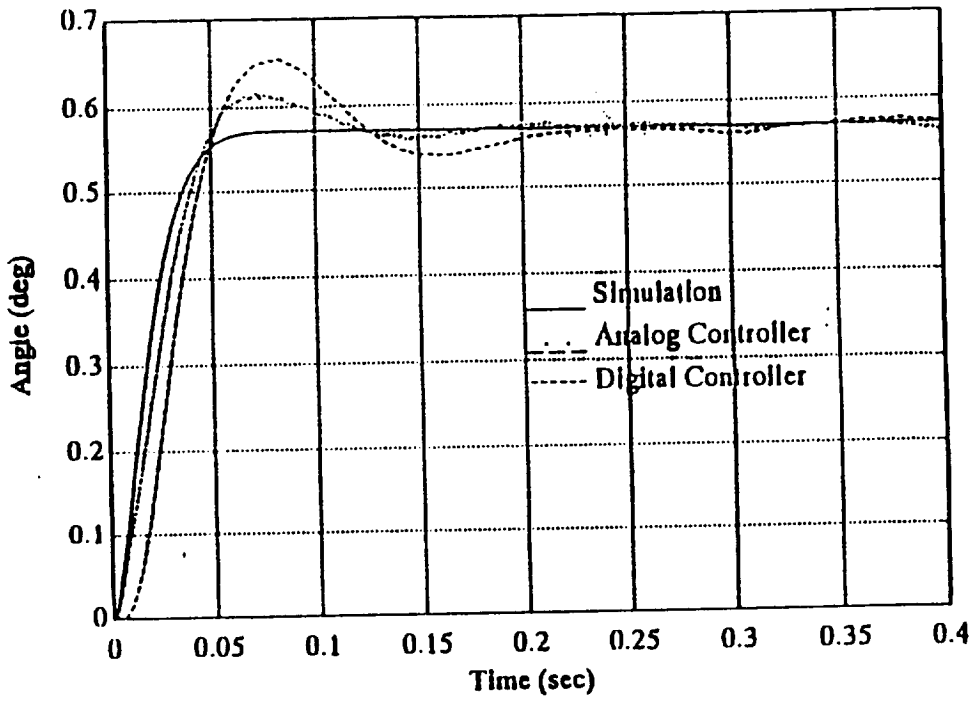


Figure 4.1 Step Response in Pitch

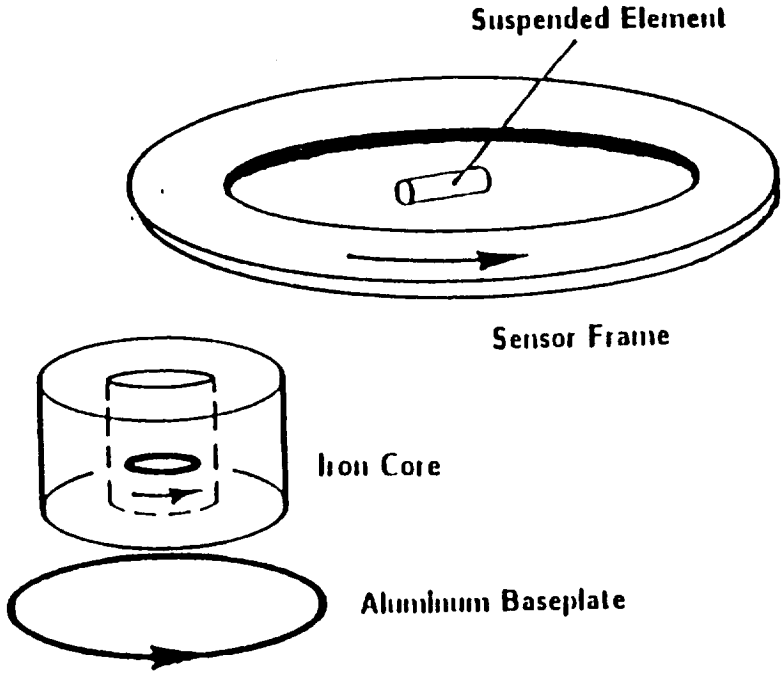


Figure 4.2 The Eddy Current

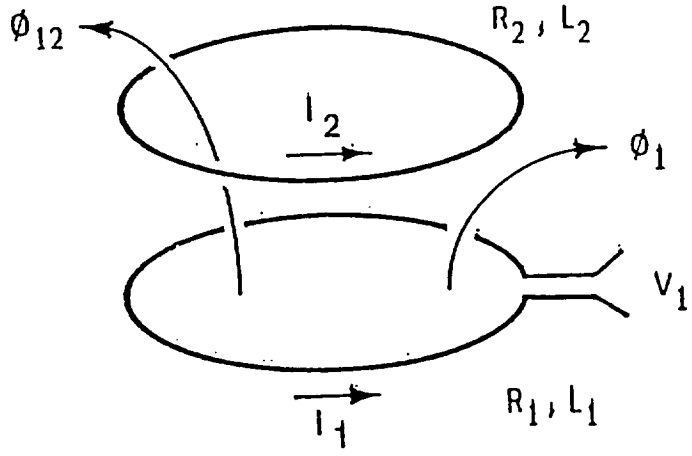


Figure 4.3 The Circuit Model

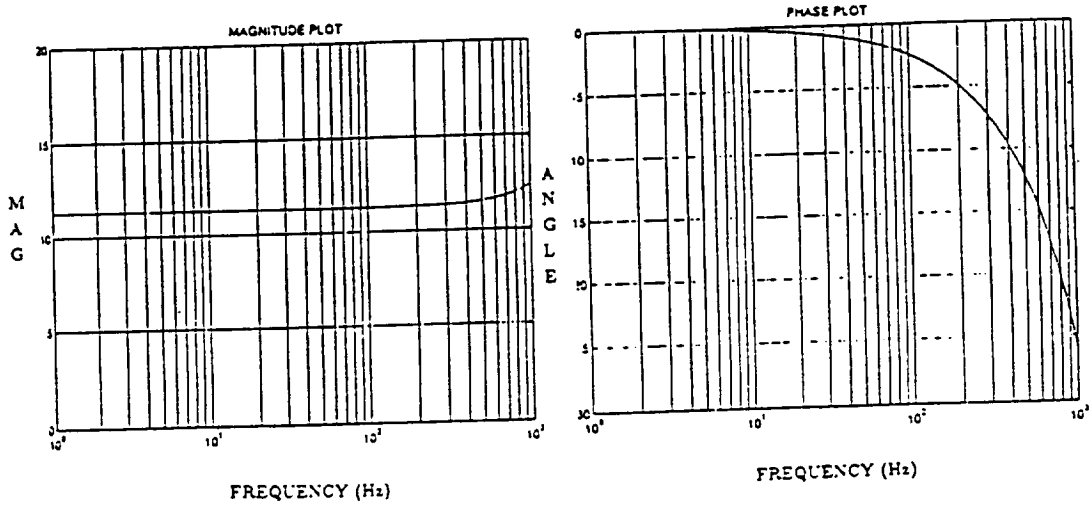


Figure 4.4 Probe Baseline Tests

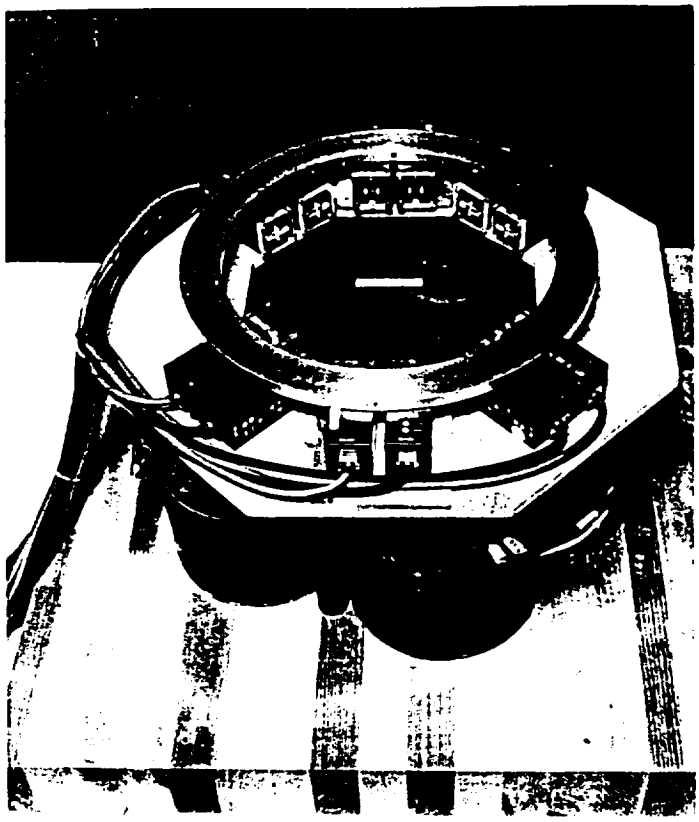


Figure 4.5 Dummy Sensor Ring

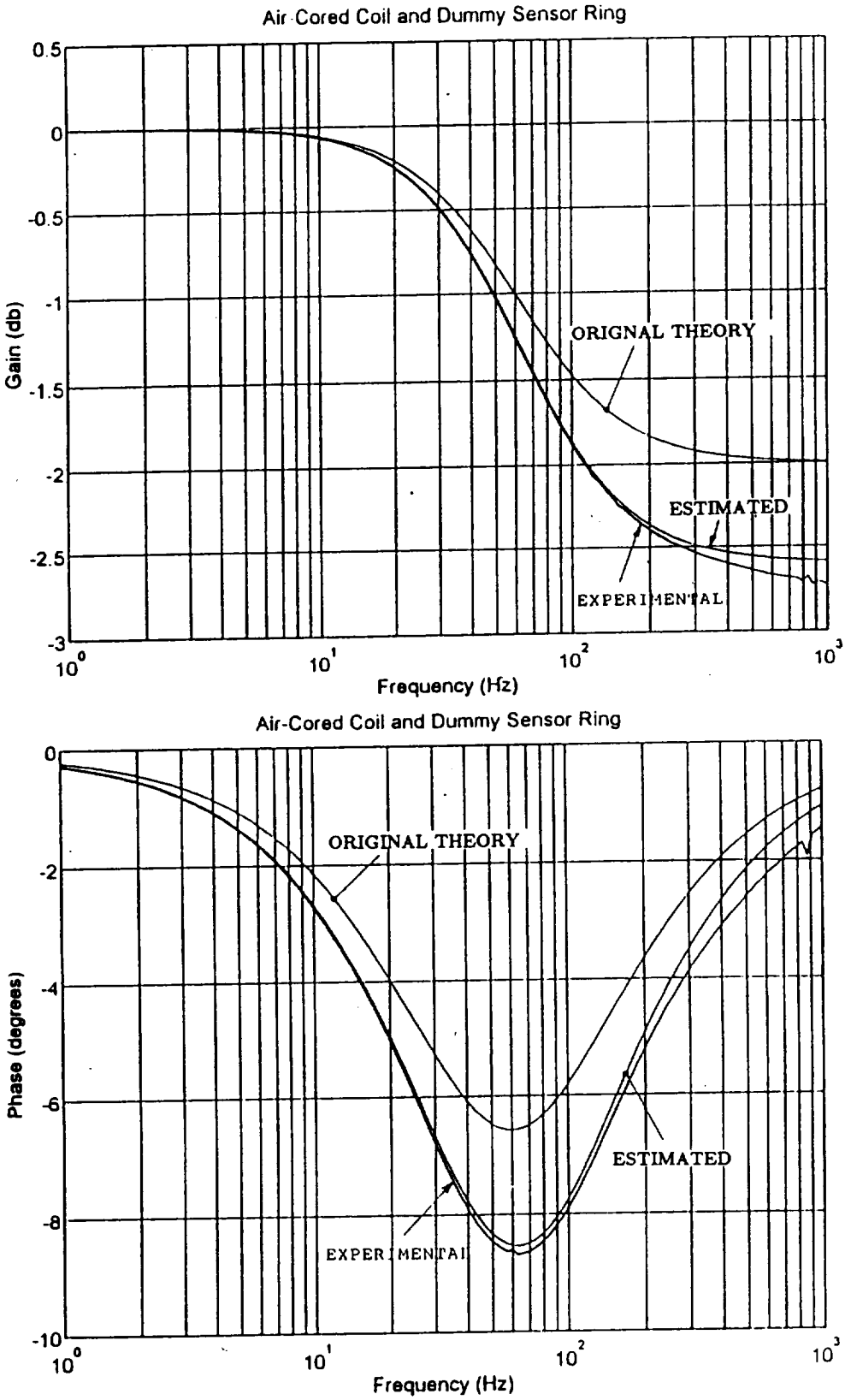


Figure 4.6 Response of the Dummy Sensor Ring

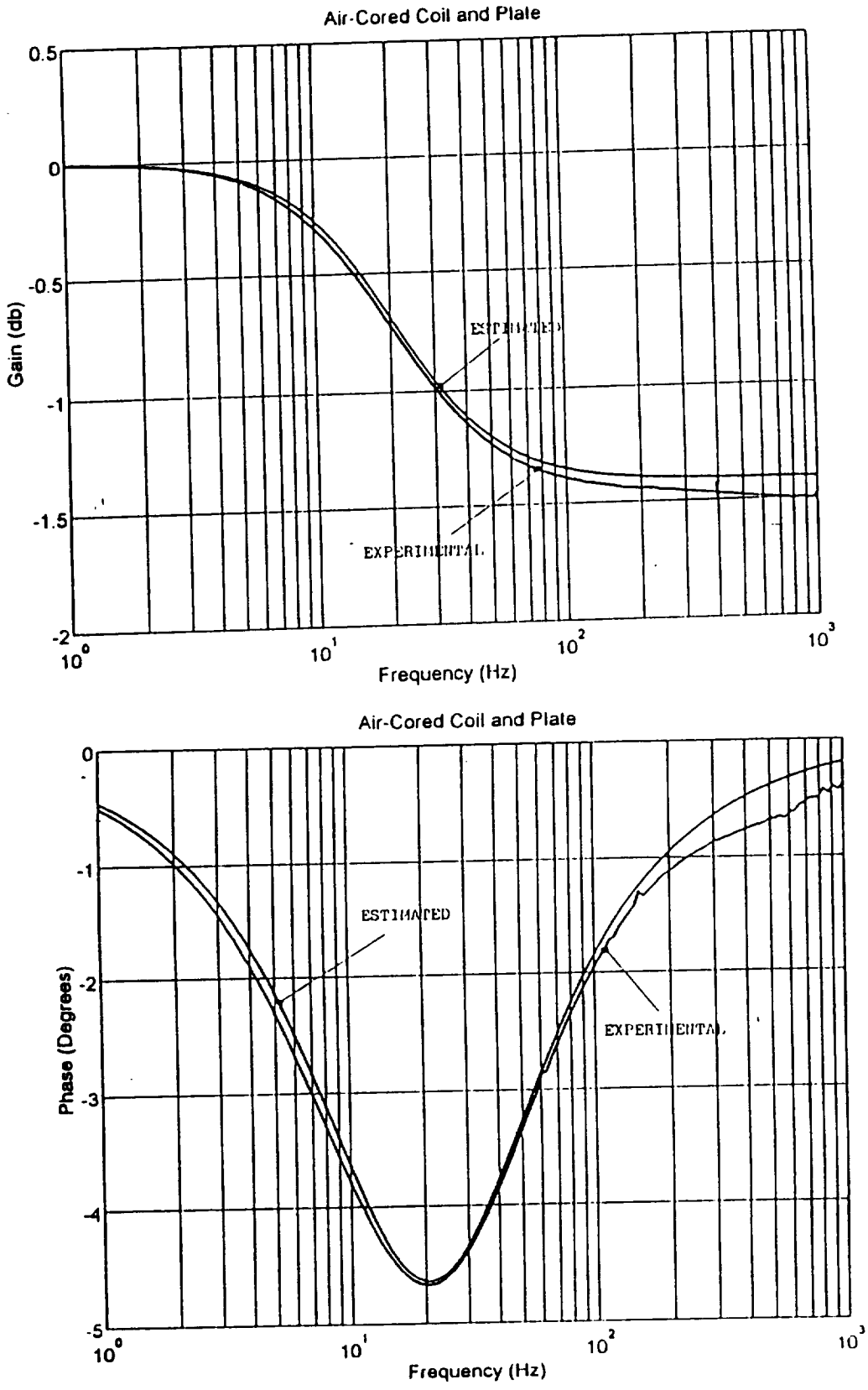


Figure 4.7 Response of the Aluminum Plate

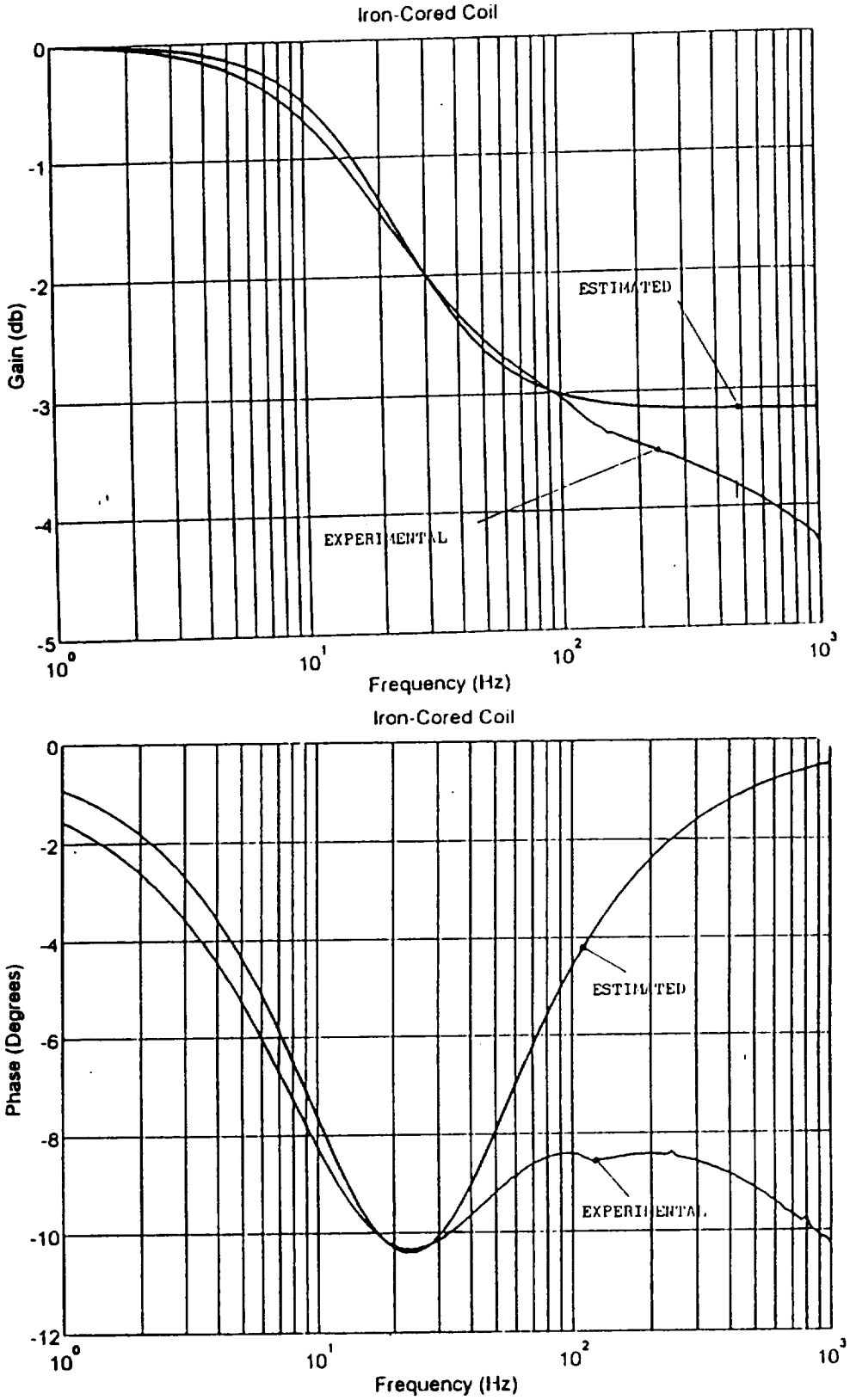


Figure 4.8 Response of the Iron Core

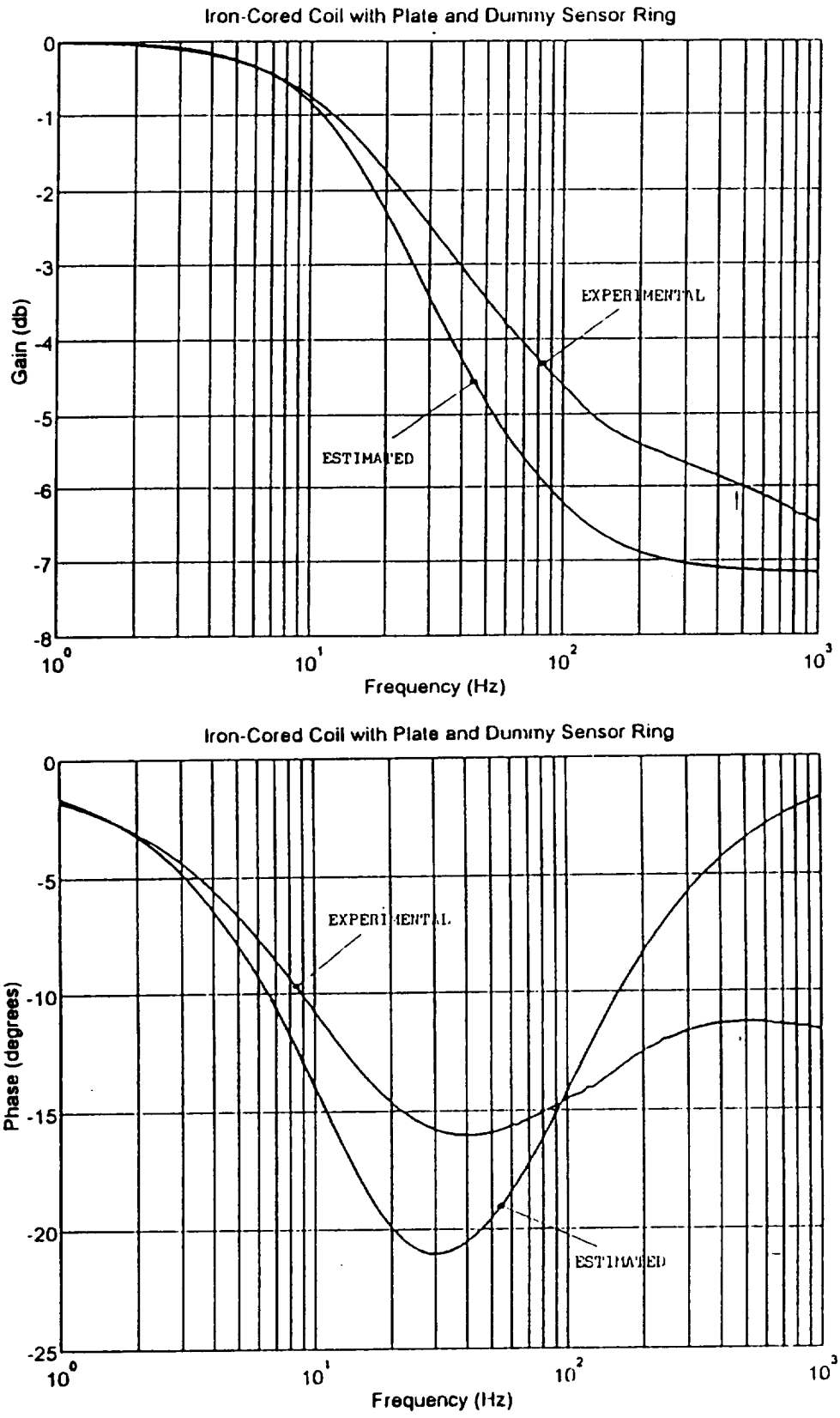


Figure 4.9 Response of all of the Eddy Currents in the System

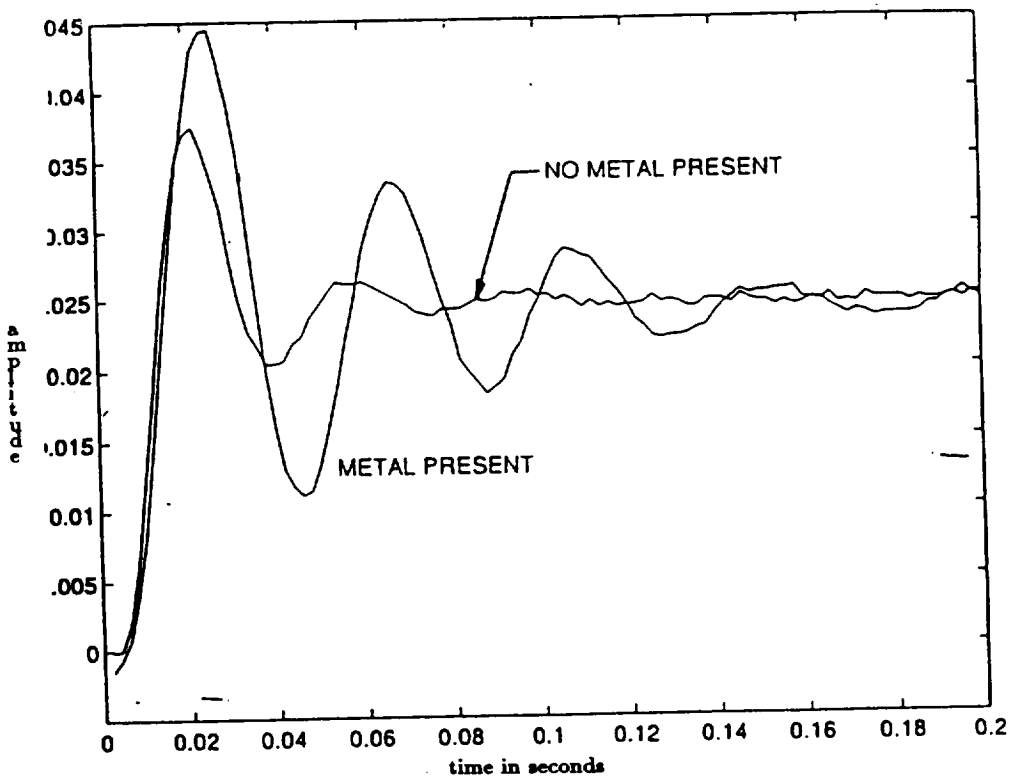


Figure 4.10 Experimental Step Response in Pitch

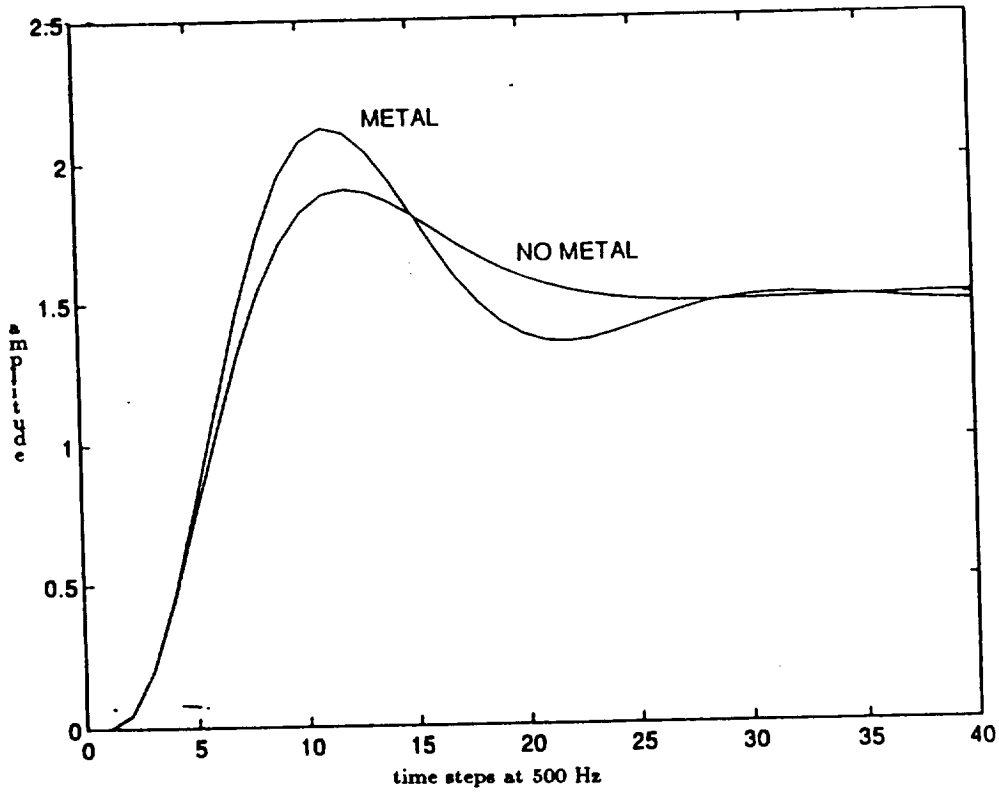


Figure 4.11 Simulated Step Response in Pitch

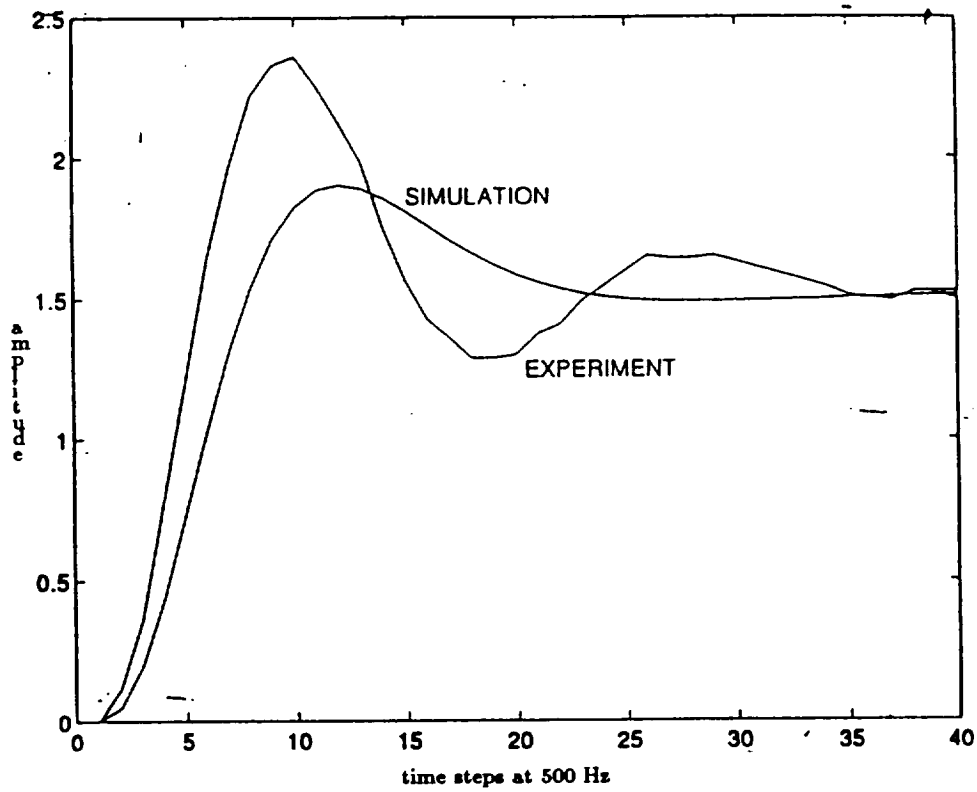


Figure 4.12 Simulated and Experimental Step Response

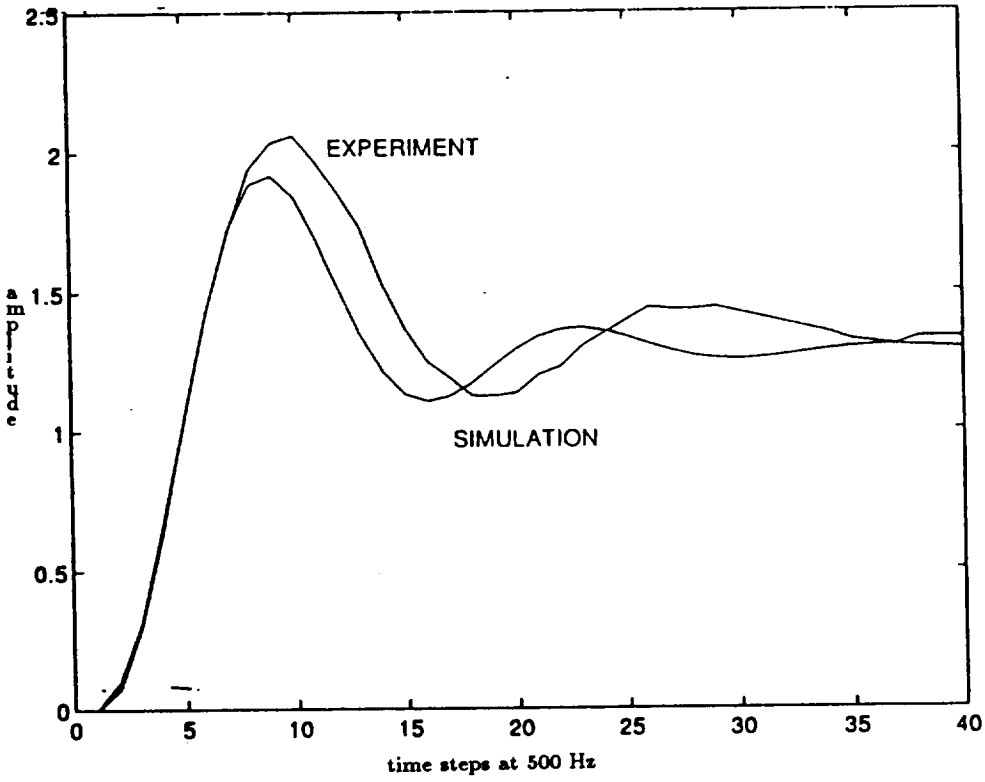


Figure 4.13 Updated Simulation vs. Experiment Without Dummy Sensor Ring

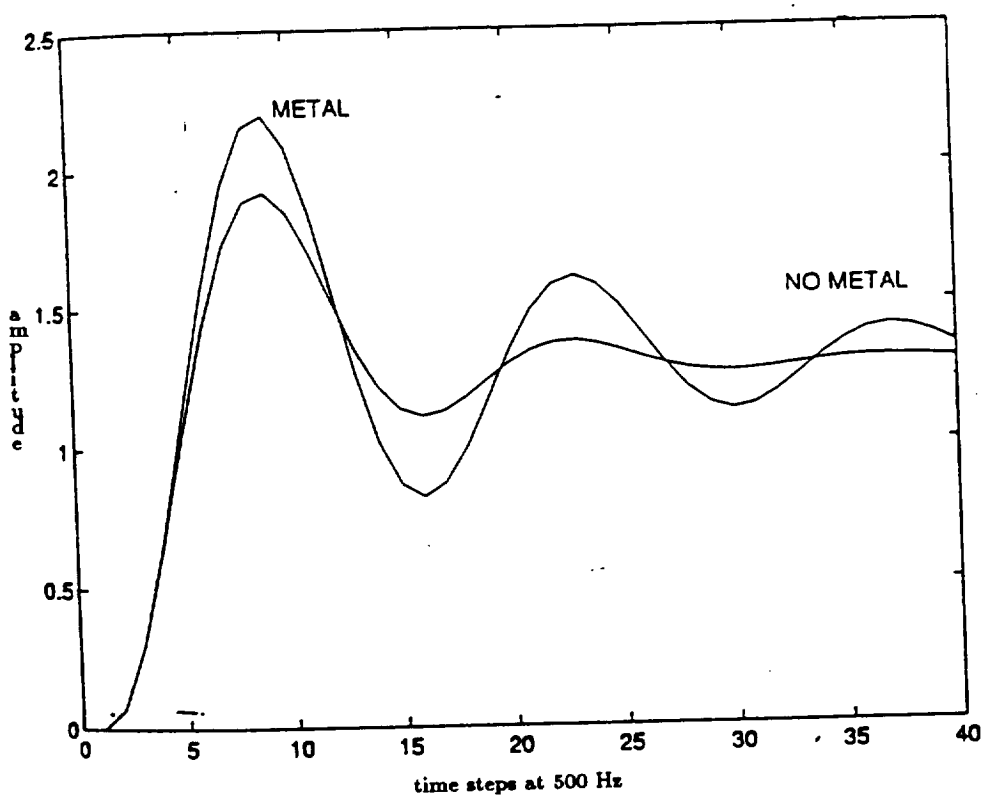


Figure 4.14 Updated Simulation for Metal vs. No Metal

CHAPTER FIVE

Magnetic Field Calculation

5.1 OPERA Calculation Theory

OPERA is a well established environment for analyzing magnetic problems. From this environment a number of different magnetic field solving programs, such as GFUN, Elektra, and Soprano, can be run. GFUN was developed originally by the Rutherford Appleton Laboratory in England, and its current developer and distributor is Vector Fields Inc. it performs static magnetic field calculations. It is capable of calculating the effects of a non-linear material on the magnetic fields. Most of the theoretical variable calculations did not need GFUN for the magnetic field calculation. When the work consisted of electromagnets the field calculations could be done in the OPERA post processor environment. The only case were GFUN was required is when the iron cores are added to the system.

The fields are calculated in two parts. The first part of the field is from the iron cores, and the second part is from the electromagnetic coils as shown below.

$$\vec{H} = \vec{H}_{iron} + \vec{H}_{coils}$$

The field from the coils is calculated using Biot Savart law which is integrated over the volume of the coil.

$$\vec{H} = \int \frac{\vec{J} \times \vec{R}}{|\vec{R}|^3} \, d\Omega_m$$

The fields from the iron is not as easy to calculate as the coil's fields. The iron is broken down into small pieces that are modeled like dipole magnets. The fields are calculated using the equation below:

$$\vec{H}_m = \nabla \int \vec{M} \bullet \nabla\left(\frac{1}{|\vec{R}|}\right) d\Omega_m$$

In this equation both $\vec{H}$ and $\vec{M}$ are unknowns, but they are related through the equation below.

$$\vec{M} = \mu_o (\mu_r - 1) \vec{H}$$

The previous two equations are evaluated at each of the control points in the finite element grid. This will produce a set of n equations with n unknowns. The equations are solved to find the magnetization of each element. Once the strengths of the dipole magnets are known finding the field at any given point is a matter of summing the effects of the dipoles together.

5.2 Field Gradients

Calculating the field gradients utilizes the grid command from OPERA. The grid command is the only command, in OPERA, that will produce an output file that is easily readable by other codes. However, there is no provision

in OPERA for finding the field gradients. In order to find the gradients the x, y, and z field component must be known along several lines. The configuration of these lines is shown in Figure 5.1. Each of these lines has 10 points in it, and they are centered around the point of interest. This is the information necessary to find all of the field gradients. After the information is retrieved from OPERA a mathematical analysis program such as MATLAB is utilized to find the actual gradients. For second order gradients, such as B_{xx}, a best fit 2^{nd} order polynomial is generated for the plot of B_x vs. x.

$$B_x = A\,x^2 + B\,x + C$$

The first derivative is

$$B_{xx} = 2\,A\,x + B$$

evaluated at x=0. For B_{yx} the field values for B_y along the x axis line as use for the best fit polynomial. The second order gradients, such as B_{xxx}, are found from the same best fit polynomial.

$$B_{xxx} = 2A$$

All of the field gradients can be found from the three on axis lines except for the B_{xyz} gradient. The three lines in the x direction are use for this purpose. The value of B_{xy} is found on all three lines by the method outlined above. Then a 2^{nd} order polynomial is fitted to the three values of B_{xy} giving the B_{xyz} field gradient. A listing of the MATLAB script file that is used to find the field gradients is listed in appendix B.

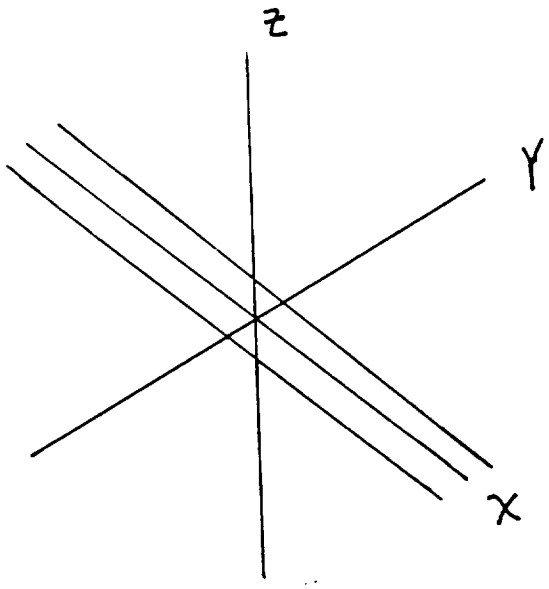


Figure 5.1 Lines for Field Gradient Calculations

CHAPTER SIX

Yaw Angle Interpolation

Finding the yaw angle of the suspended element with respect to the coils is necessary to demonstrate 360 degrees of yaw control. This is due to the fact that the computer varies a mixing matrix that determines the amount of current each coil should receive in order to maintain suspension.

$$\vec{I} = M * \vec{F}$$

Where $\vec{I}$ is the coil currents, and $\vec{F}$ is a vector of the required forces and torques. The mixer matrix M changes according to the yaw angle. The problem of finding the yaw angle is in the sensors. When the suspended element leaves the range of the sensors they do not feedback the correct position, and they have a short range. In order to keep the suspended element in sensor range during a 360 degree rotation the sensors are rotated by hand with respect to the electromagnets. This means that the computer needs information as to how far the sensors have been rotated. A rotary encoder, or similar piece of equipment, could have been used, but a desire not to add any more new hardware to the system has led to the development of other methods to detect the yaw angle.

Two methods have been developed that use the same amount of information that the computer was already receiving.

In the first method to be implemented, the angle is determined from the yaw error [17]. As the sensors are rotated, the mixing matrix, which is not being updated at this point, develops inaccuracies. These inaccuracies cause the model to be suspended with a yaw angle error. When this error is large enough not to be noise the controller updates the mixer matrix in such a way as to drive the error back to zero. The method has been used to demonstrate 360 degrees of yaw, and it is surprisingly reliable. However, the motion is not smooth because when the mixer matrix is updated the suspended element "snaps" to the new yaw angle.

The second method reads the time-averaged current going to the coils during suspension. For each yaw angle there is a unique current distribution required to maintain suspension. The currents in the coils vary like five cosine waves shifted in phase as the suspended element is rotated 360 degrees (see Figure 6.1). This allows each yaw angle to have a unique set of coil currents.

By reading the currents an algorithm can determine the angle of yaw and evaluate a new mixer matrix. The has the advantages of continuously updating the matrix allowing a smooth rotation.

6.1 The Method of Calculation

In theory the currents in the coils follow the equation:

$$I_n = I_{max} * \cos(\theta_n - \phi)$$

Where I_n is the current in the n^{th} coil, I_{max} is the maximum current needed in any coil for the purposes of suspension, θ_n is the angular position of the coil itself, and ϕ is the position of the suspended element. Both of the angles are measured with respect to the first coil. In practice the currents are updated at high frequency, to maintain suspension, which causes noise. This noise must be eliminated to get an error-free result. The noise is reduced by passing the current values through a low pass filter with a break frequency on the order of one Hz.

The yaw function that is generated is as follows:

$$\text{Yaw Function} = \sum_{n=1}^{5} I_n * \cos(\theta_n - (\psi + \epsilon))$$

Where ψ is the previous yaw angle, and ϵ varies between $\pm$ five degrees. The yaw function is calculated for each ϵ. When the yaw function is plotted versus ϵ, its maximum is at the yaw angle (as shown in Figure 6.2 for a yaw angle of 5 degrees and a guessed angles of 3 degrees).

The distance d is the correction factor to be added to the old value of the yaw angle. The distance is calculated be fitting a second order polynomial to the yaw function as in the equation below.

$$\text{Yaw Function} = A_{yaw} * \epsilon^2 + B_{yaw} * \epsilon + C_{yaw}$$

Then the correction factor and yaw angle are:

$$\epsilon_{max} = \frac{-B_{yaw}}{(2 * A_{yaw})}$$

$$\text{Yaw Angle} = \psi + \epsilon_{max}$$

During initialization of the computer, mixer matrices are stored for six degree increments around the first 72 degrees. From these matrices the mixer matrix for any angle can be interpolated. Due to the unique symmetry of the system the numbers inside the mixer matrix repeat every 72 degrees. The only thing that changes is the numbering convention used for the coils. As the rotation passes the 72 degree mark coil 2 becomes coil 1 for the purposes of calculating the mixer matrix.

6.2 Experimental Results

Using the current interpolation program to control the gain scheduling has resulted in reliable suspension through a large yaw angle. This program has also demonstrated rotations larger than 360 degrees. As the angle passes 360 degrees the program resets the angle back to 0 degrees. The rotation using the interpolation program for gain scheduling is smoother than when the gain scheduling was performed using the yaw angle error filtering routine. Figure 6.3 shows the suspended element undergoing a large angle rotation.

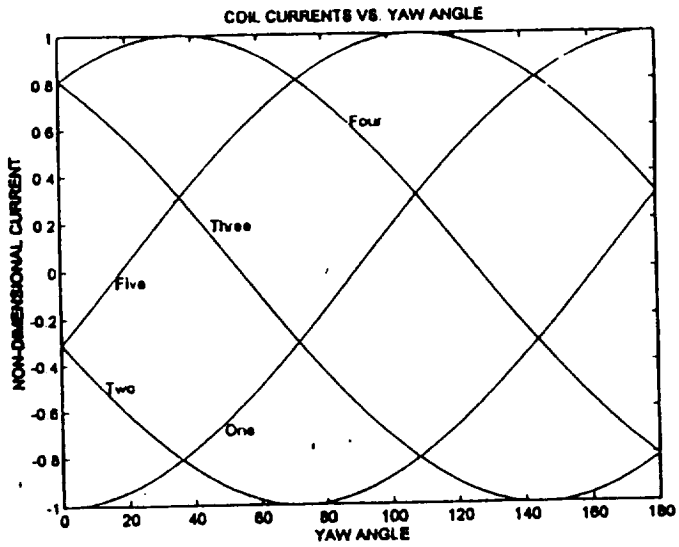


Figure 6.1 Current vs. Yaw Angle

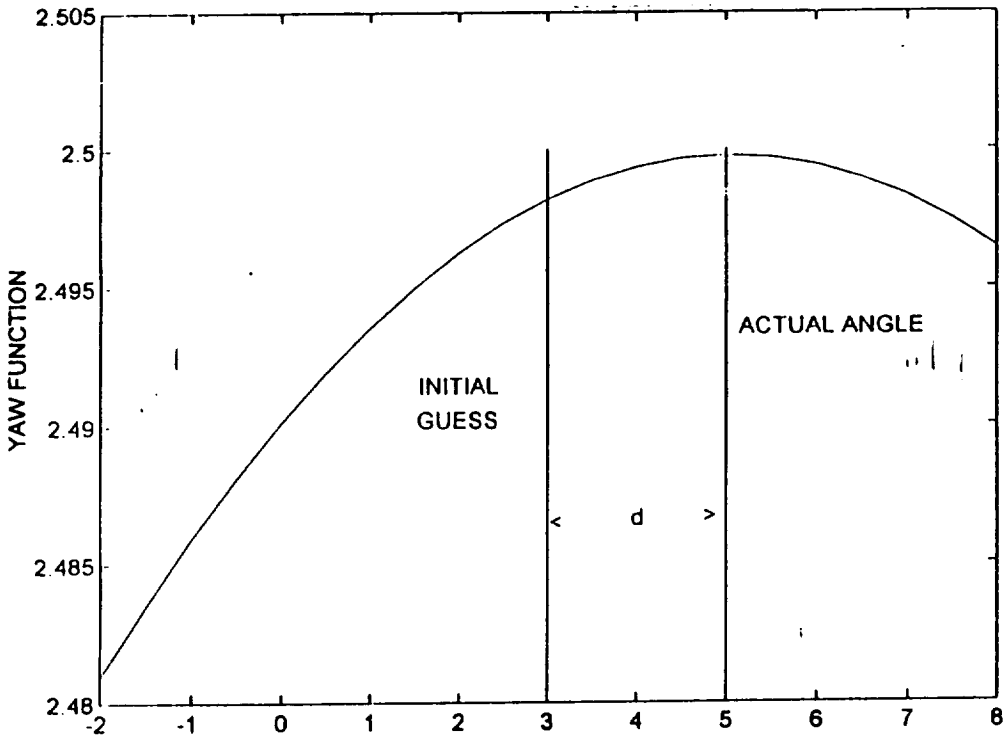


Figure 6.2 Yaw Function vs. Yaw Angle

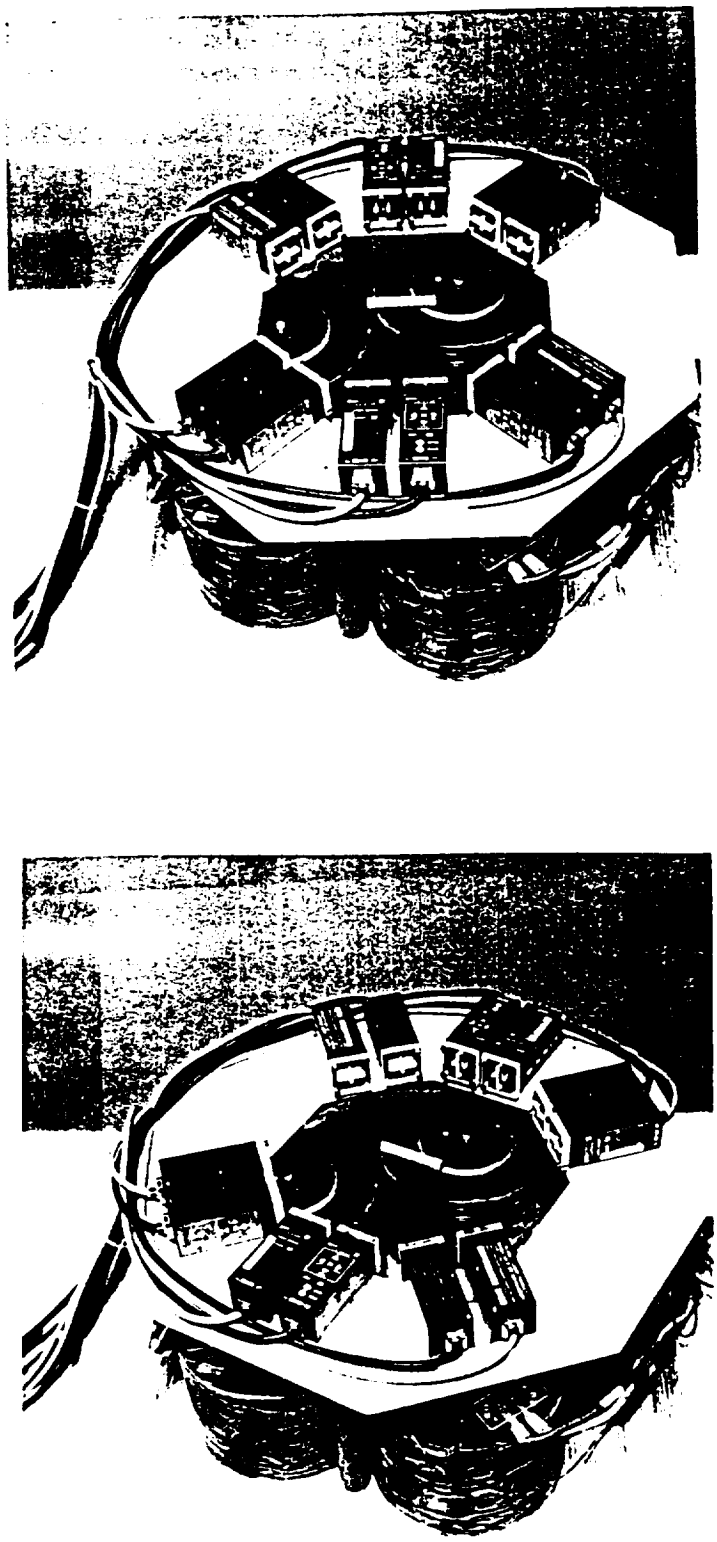
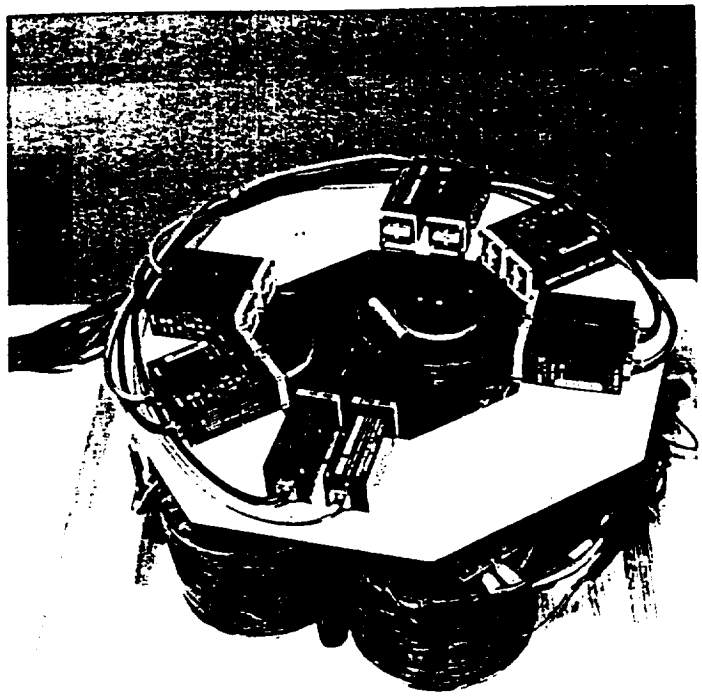


Figure 6.3 Yaw angle rotation 360 Degrees



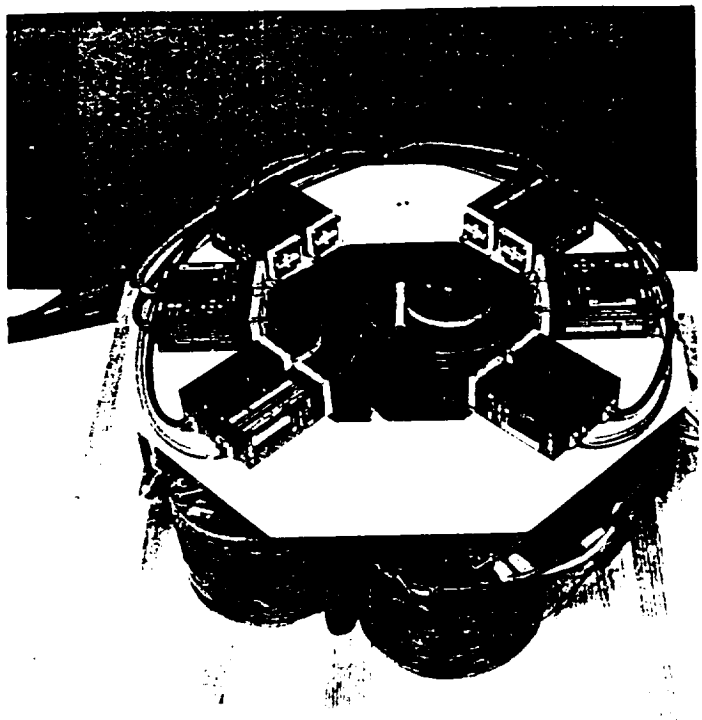
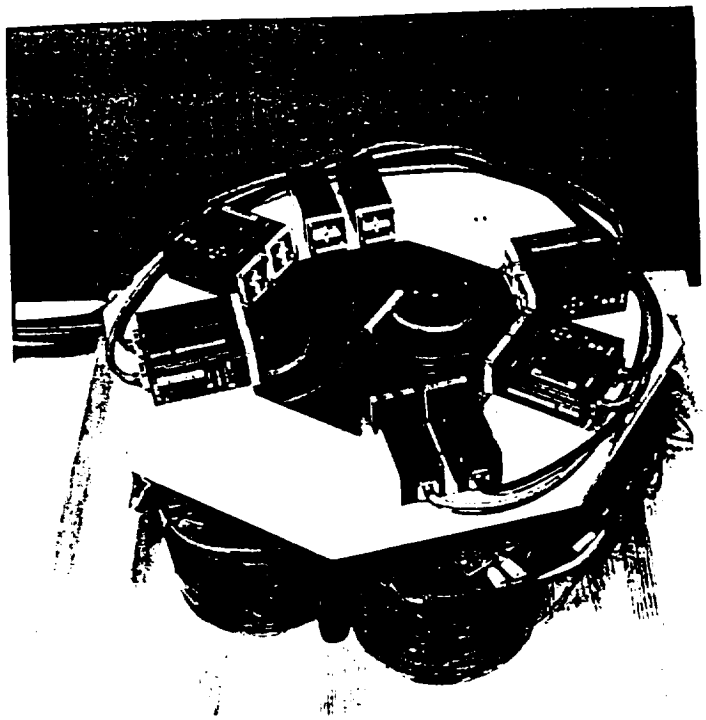


Figure 6.3 Continued



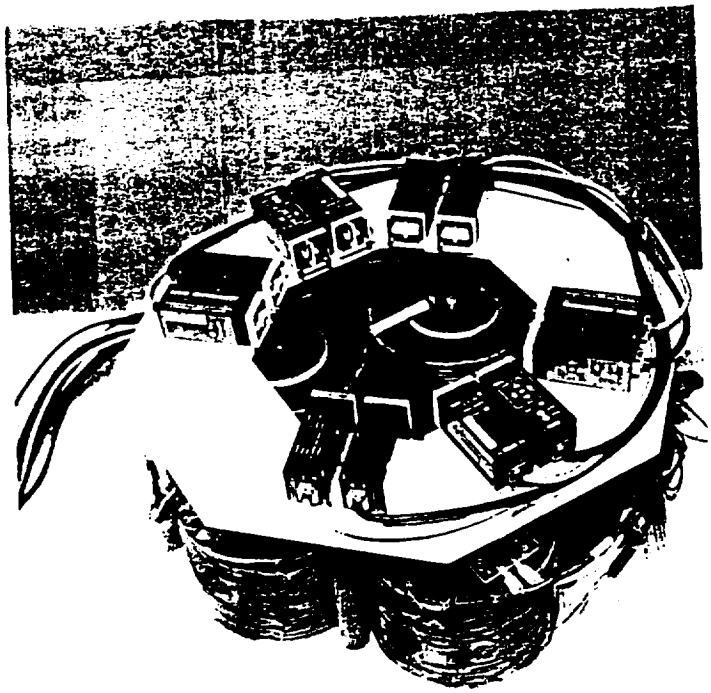


Figure 6.3 Continued

Figure 6.3 Continued

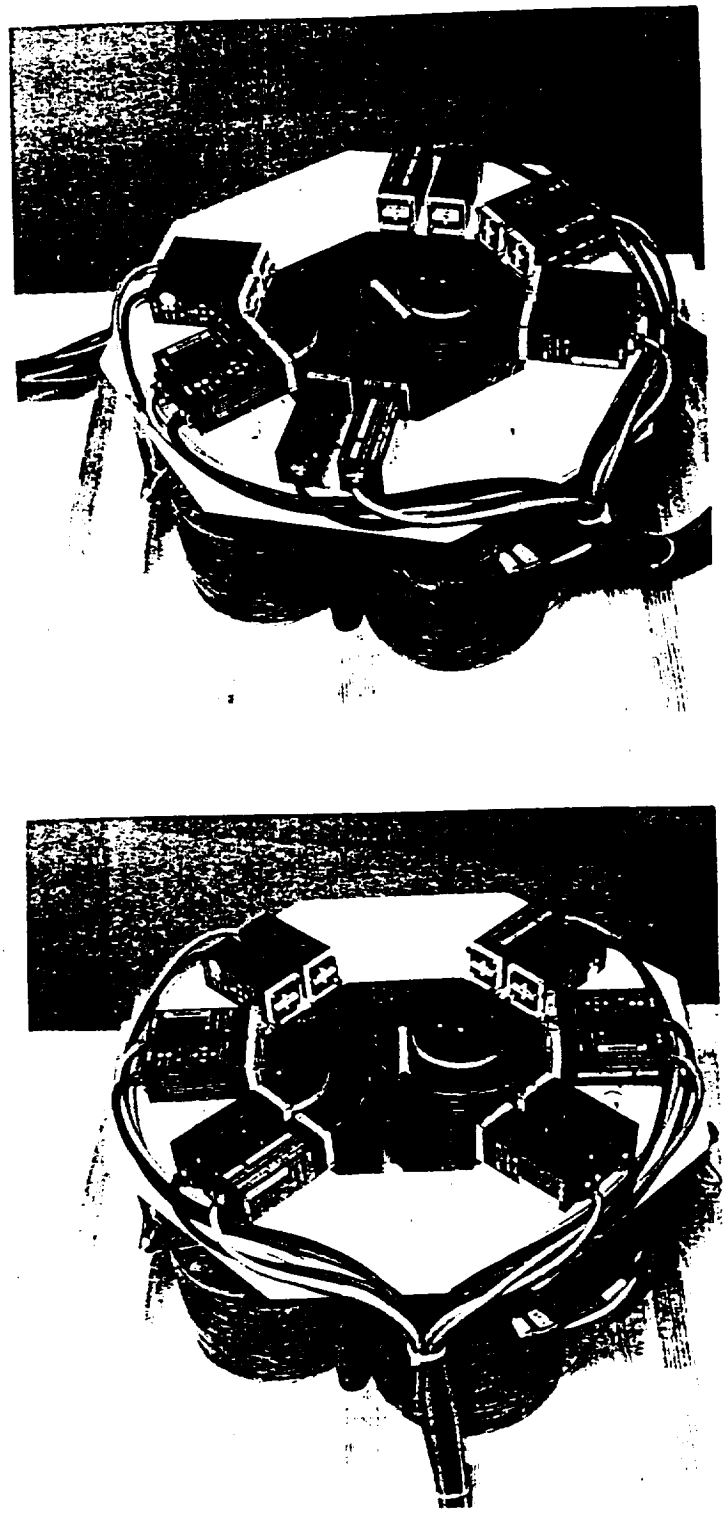


Figure 6.3 Continued

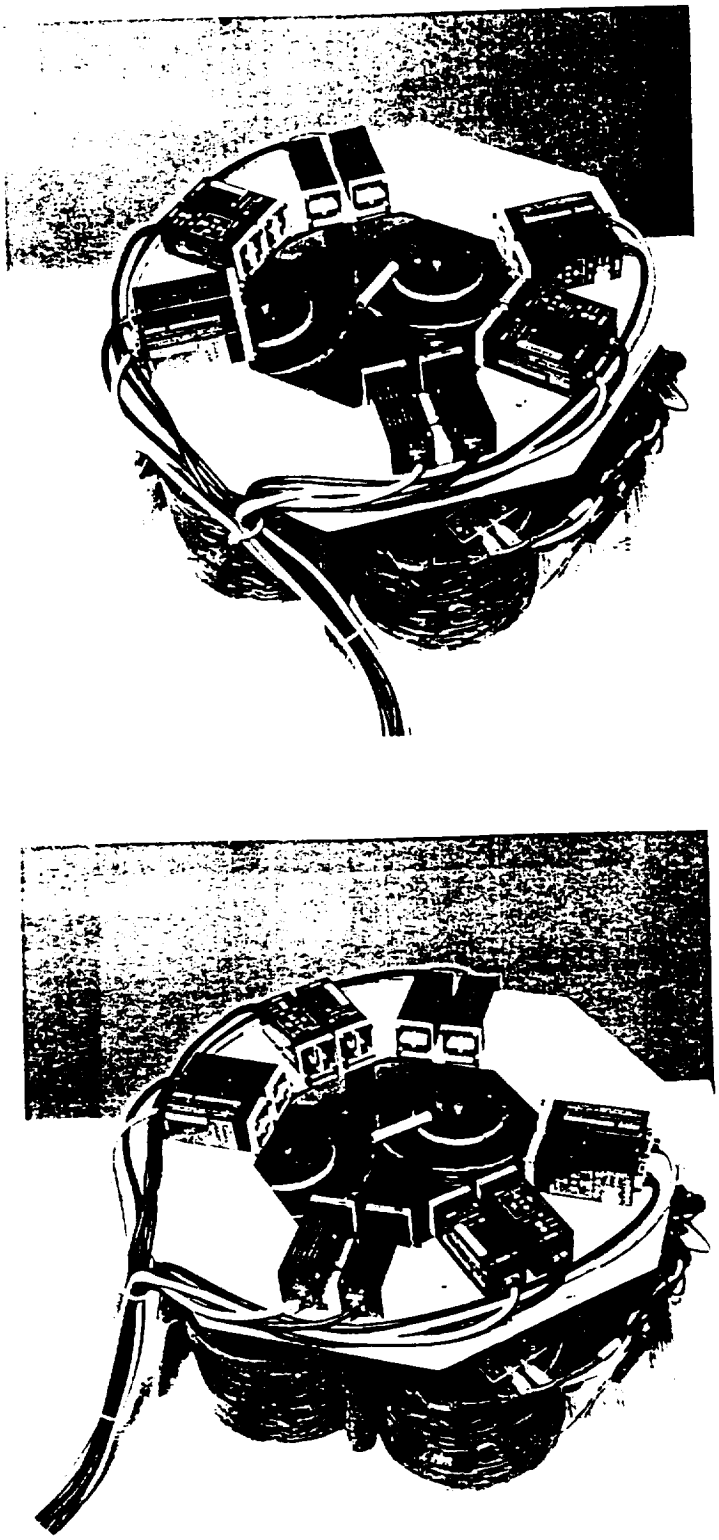


Figure 6.3 Continued

CHAPTER SEVEN

Discussion

The procedure of taking experimental frequency response data and using it to update parameters in a theoretical formula can be applied to a variety of systems such as wind tunnels and magnetic bearings. How accurate the mathematical model adjustment is will be determined, in part, by some characteristics of the eddy currents involved. When the mathematical model of LAMSTF was updated for the aluminum sensor ring the results were very good. However, when the model was updated for the iron cores, the results were not as accurate, particularly at the higher frequencies. In the case of LAMSTF the effect of the cores is not as pronounced as the effects of the sensor ring. Therefore, representation of the dominant eddy current leads to significant improvements in the model fidelity.

When applying this procedure to other systems, the eddy current with the largest effect must be identified. Once this is done, determination of the useful benefit depends on the eddy current itself and the frequencies of interest from the controls stand point. Some points to consider about the eddy current itself are how well the shape is known or constrained, and if the shape is stable with respect to the frequency.

In the case of the iron core the shape of the eddy current changes as the frequency of the driving magnetic field is increased. This is due to the change of

the skin depth, which has otherwise been assumed to be so large that the eddy currents are distributed evenly throughout the piece of metal. As the shape changes, the resistance, which was assumed to be constant, is drastically altered. Checking the skin depth requirement at high frequency will give an indication of the applicability of the technique to other eddy currents.

Finding the shape of the eddy current is important to establish variables that can be used as a starting point for estimating the parameters that will then be used experimentally to update the system's mathematical model. The more that is known about the shape of the eddy current the more accurate the starting point will be. If the eddy current is highly constrained, as was the case of the sensor ring, the initial guess will be fairly close to the actual answer. In the case of the aluminum plate, the eddy current's diameter was unknown, and this made the first approximation worse as compared to the first approximation of the the sensor ring. The same thought holds true for multiple eddy currents. If the eddy currents can be isolated, and then added together the theory will work. If, however, the eddy current can not be isolated the effects of a single eddy current will be lost when combined with all of the other eddy currents. This was the case with the original sensor support frame where one piece of metal could be part of 3 or 4 different eddy current loops.

The last point to consider is the frequency range of interest. The mathematical model does not have to be accurate out to a high frequency if it is not used. The range for frequencies can be determined by the highest open loop unstable pole of the system. This pole will determine how fast a controller must be run to achieve suspension. If the analysis corresponds to the experimentally determined transfer function then the updates to the model will be helpful.

CHAPTER EIGHT

Conclusions

When the system was under construction, metal was used extensively for support structures, etc.. It was known that eddy currents would be present in the system, and the purpose of using so much metal was to study the effects of these eddy currents. The eddy currents did not effect the system to the point where the system could not be made operational. They did, however, have a pronounced effect on the performance of the system. This effect showed up in the step responses of the different degrees of freedom most notably in the pitch response. After the general effect was understood, the system was refined to study the individual eddy currents. This was done by eliminating the eddy currents using traditional techniques such as replacing the metal with a nonconductive material. This step was necessary to isolate the different eddy currents. Before the system was altered it was not clear which eddy currents were altering the system dynamics. After the different eddy currents were sorted out the sensor frame was shown to have the most prominent effect on the system followed by the iron cores, and the aluminum baseplate effect was shown to be very small. A means to add the dynamics of the sensor ring to the system was developed, and it is outlined in this thesis. The theory best applies to a dummy sensor ring. That case was used to compare the updated model to the experimental system. By adding the ring's dynamics to the system the step response in pitch became less damped, and this was the effect observed in the experiments on the system.

The method outlined for analyzing the eddy currents to the system has several drawbacks. One eddy current, the dummy sensor ring, has been added to the system. The results agreed better with the experiment, but errors still exist in part due to the other eddy currents. These other eddy currents were not added to the system for two reasons. The theory did not cover the iron cores as well as the sensor ring, and the method by which the eddy currents are added is questionable when a significant amount of eddy current need to be added to the system. Adding together several different eddy currents is necessary to improve a system with a lot of metal and multiple eddy currents in the structure. A better method is needed for adding multiple eddy currents to the system dynamics.

Another draw back to the theory is the problem of describing the eddy currents present in the iron cores. A theory is under development that can explain the dynamics of the iron cores. One of the unique problems of describing the eddy currents present in the core is that the gain roll off is 10 db/dec, or half of a single pole's rolloff. This is explained in the proposed theory by using an infinite pole zero arrangement. This will be difficult to add to the system due to the fact that even when truncating to just 5 or 6 poles the method currently being used to add the dynamics to the system becomes questionable.

REFERENCES

[1] Britcher, C. P.: "Large Gap Magnetic Suspension System", Proceedings of the International Symposium on Magnetic Suspension Technology, pp. 33-50, Aug 19-23, 1991

[2] Moulton, D. H. and Eakins, P. S.: "Application of Magnetic Bearings to Solar C304 Natural Gas Pipeline Compressor", Proceedings of the Mag 92 Magnetic Bearings, Magnetic Drives and Dry Gas Seals, pp. 203-217, July 29-31, 1992.

[3] Proise, M.: "System Concept Definition of the Grumman Superconducting Electromagnetic Suspension (EMS) Maglev Design", Proceedings of the 2nd International Symposium on Magnetic Suspension Technology, pp. 69-76, August 11-13, 1993.

[4] Luerken, R. F.: "Transrapid-The First High-Speed Maglev Train System Certified "Ready for Application": Development Status and Prospects for Deployment", Proceedings of the 2nd International Symposium on Magnetic Suspension Technology, pp. 77-92, August 11-13, 1993.

[5] Arling, R.W. and Kohler, S. M.: "Six Degree of Freedom Fine Motion Positioning Stage Based in Magnetic Levitation" Proceedings of the 2nd International Symposium on Magnetic Suspension Technology, pp. 641-652, August 11-13, 1993.

[6] Britcher, C.P. and Groom, N.J.: "Current and Future Development if the Annular Suspension and Pointing System", Proceedings of the Fourth International Symposium on Magnetic Bearings, pp. 559-564, August 23-26, 1994.

[7] Fenn, R. C., Gerver, M., and Johnson, B.: "A Six Degree-of-freedom Lorentz Force Vibration Isolator with Nonlinear Controls", Proceedings of the International Symposium on Magnetic Suspension Technology, pp. 219-246, August 19-23, 1994.

[8] Britcher, C. P. and Alcorn, C.W.: "Interference-Free Measurement of the Subsonic Aerodynamic of Slanted-Base Ogive-Cylinder", AIAA Journal, April 1991, vol. 29, no. 4, pp. 520-525.

[9] Genta, G. et all: "Analytical and Experimental Investigation of a Magnetic Radial Damper", Proceedings of 3rd International Symposium on Magnetic Bearings, pp. 255-266, July 29-31, 1992.

[10] Cox, D. E. and Groom, N. J.: "Implementation of a decoupled Controller for a Magnetic Suspension System Using Electromagnets Mounted in a Planar Array", Proceedings of the 2nd International Symposium on Magnetic Suspension Technology, pp. 257-274, August 11-13, 1994.

[11] Britcher, C. P., and Foster, L. E.: "Some Further Developments in the Dynamic Modeling and Control of the Large Angle Magnetic Suspension test Fixture", Proceedings of the 2nd International Symposium on Magnetic Suspension Technology, pp. 367-390, August 11-13, 1993.

[12] Groom, N.J.: "Analytical Model of a Five Degree of Freedom Magnetic Suspension and Positioning System", NASA TM-10067, March 1989.

[13] Allaire, P.E. Ed.: "Proceeding of the Third International Symposium on Magnetic Bearings", Technomic Publishing Co., Lancaster, PA, 1992.

[14] Tuttle, M. H., Moore, D. L., and Kilgore, R. A.: "Magnetic Suspension and Balance Systems, A Comprehensive, Annotated Bibliography", NASA TM-4318.

[15] Stoll, R. L.: "The Analysis of Eddy Currents", Clarendon, 1974

[16] Groom, N.J. and Britcher, C.P. : "Open Loop Characteristics of Magnetic Suspension Systems Using Electromagnets Mounted in a Planar Array", NASA TP-3229, November 1992.

[17] Ghofrani, M. and Britcher, C. P.: "Approaches to Control of the Large Angle Magnetic Test Fixture", Masters Thesis for Old Dominion University Norfolk, December 1992.

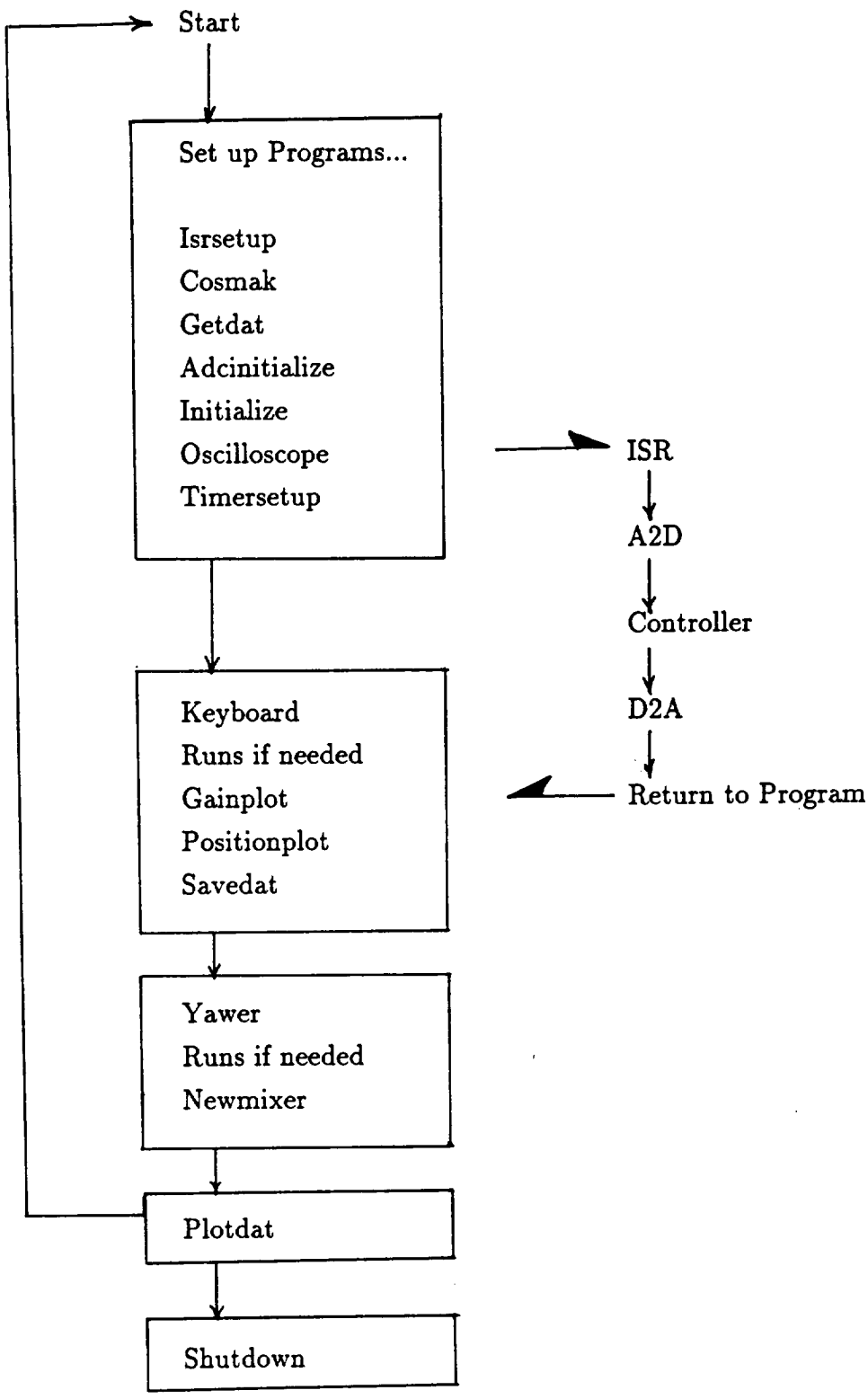
APPENDICES

APPENDIX A

Code Development

The code was developed based on a code that was developed for the original system. The original code is discussed in more detail in reference 17. Several modifications were performed on the original code due to updates in the system. These updates include changing the boards used, cleaning up the code and installing the yaw angle interpolation subroutines.

Updating the program is necessary when a different board is used because every board has different commands that serve the different input-output ports. To help facilitate the updating and correction of the program it has been moduralized. This procedure starts with the original program that was 26 pages of code and breaks it into several different files each of which control a single aspect of the program's performance. This makes the program easier to update leaving the bulk of the program intact. A flow chart for when the different files are used is presented followed by a listing of the program.


Start
Set up Programs...
Isrsetup
Cosmak
Getdat
Adcinitialize
Initialize
Oscilloscope
Timersetup
ISR
A2D
Controller
D2A
Return to Program
Keyboard
Runs if needed
Gainplot
Positionplot
Savedat
Yawer
Runs if needed
Newmixer
Plotdat
Shutdown

Header file to declare variables and subroutines.

```
#define TIMER0x210   /* Timer board Address */
#define DDAbase 0x320   /* D/A base address  */
#define UP 72
#define DOWN 80
#define LEFT 75
#define RIGHT 77
#define ENTER 28
#define ESC 27
#define F1 59
#define F2 60
#define F4 62
#define F5 63
#define F6 64
#define F9 67
#define F10 68
#define pi 3.14159
#define On 1
#define Off 0
#define MAXCUR 8.36
#pragma intrinsic(outp,inp)
#ifndef WHERE
  #define WHERE extern
  WHERE double theta[5];
  WHERE unsigned int chan;
  WHERE int rota;
  WHERE int chans[5];
  WHERE int intnum;
  WHERE int gains[5];
  WHERE int plotcolor,datcolor;
  WHERE int zoom,ang;
  WHERE int tmp[5],ix;
  WHERE double offset[5];
  WHERE double gain[5],gainstep,positionstep;
  WHERE double mix[5][5];
```

```
  WHERE double yawangle,yawanglestep,autoyaw;
  WHERE int chd;
  WHERE char plotmode,ch2;
#else
  double theta[5]={0.0,72.0,144.0,216.0,288.0};
  int chd=1;
  char plotmode='a',ch2='z';
  unsigned int chan=10;
  int rota=0;
  int chans[5]={0,1,2,3,4};
  int intnum=0x0d; /* interrupt vector number */
  int gains[5]={0,0,0,0,0};
  int plotcolor=0,datcolor=15;
  int zoom=1,ang=0;
  int tmp[5]={53,101,149,197,245},ix=25;
  double offset[5]={8.36,2.713,-6.965,-6.965,2.713};
  double gain[5]={1,1,1,.5,.5},gainstep=.01,positionstep=.1;
  double mix[5][5]={{1.0,0.0,-1.0,0.625,0.0},
   {-0.809,0.618,-0.309,1.0,1.0},
   { 0.309,-1.0,0.809,0.768,0.618},
   { 0.309,1.0,0.809,0.768,-0.618},
   {-0.809,-0.618,-0.309,1.0,-1.0}};
  double yawangle=0,yawanglestep=0.5,autoyaw=0;
#endif
WHERE int mode,flag,params[10];
WHERE unsigned int *DMA_buf,*bufoffset;
WHERE int far *chans_ptr;
WHERE int far *gains_ptr;
WHERE double mixall[5][5][61],filt[5],position[5];
WHERE unsigned int adcsr;
WHERE int kv,dec,sampler;
WHERE int y11,x1,samplrate,diviser;
WHERE int input,k,j,num;
WHERE int ii,i,l,x,olddat[500][5];
WHERE int positionsign,gainupdown,trig;
WHERE double a11,a,b,c,d,e,f,current[5],savdat[5][500];
```

```
WHERE double v,v1[25],v2[25],vsensor[6];
WHERE double cosdat[800];
WHERE int maxnum,yawerrorlimit;
WHERE char ch,ch1,loop,chin;
WHERE long backc;
WHERE double y,xy,x2y;
WHERE double correction,testangle,function,ag;
  void (_interrupt _far *oldnum)(void);
  void _interrupt _far isr(void);
  extern MSCL_DAS40(int*,int*,int*);

  void initialize(void);
  void oscilloscope(void);
  void timerset(void);
  void adcinitialize(void);
  void plotdat(void);
  void keyboard(void);
  void outdat(void);
  void gainplot(void);
  void positionplot(void);
  void compensator(void);
  void timersetup(void);
  void acknowledge(void);
  void savedat(void);
  void getdat(void);
  void newmixer(void);
  void cosmak(void);
  void shutdown(void);
  void isrsetup(void);
  void yawer(void);
```

Interrupt Service routine (controller)

```
#include <graph.h>
#include <stdio.h>
#include <conio.h>
#include <math.h>
#include <float.h>
#include <dos.h>
#include <bios.h>
#include <signal.h>
#include <stdlib.h>
#include <malloc.h>
#include "suspend.h"

/* Intrrupt service routine  */
/* datin() obtains data from ADC in the array      */
void _interrupt _far isr(void)
{
   unsigned int y11,Outport;
   int k,df,tmp;
  for(k=0;k<=5;k++)
{
     mode = 7;
     flag = 0;
     params[0] = k;
     params[1] = 0;
     MSCL_DAS40 (&mode, params, &flag);
     if (flag != 0) ch=ESC;
     tmp=params[0];
     vsensor[k]=(float)(tmp/409.5);
   }
_enable();

/*************************************************************************
*/
/*    decoupler       */
```

```
    v1[2]=((vsensor[2]-vsensor[3])+(vsensor[5]-vsensor[4]))/(4*.707);
    v1[5]=((-vsensor[2]+vsensor[3])+(vsensor[5]-vsensor[4]))/(4*.707);
    v1[8]=(vsensor[0]+vsensor[1])/2;
    v1[11]=(vsensor[1]-vsensor[0]);
    v1[14]=((-vsensor[2]-vsensor[3])+(vsensor[4]+vsensor[5]))/2;

    if(sampler<=200)
    {
for(df=0;df<=4;df++)
savdat[df][sampler]=v1[2+df*3];
sampler++;
    }
 /* compensator: dual phase advance using backward difference method */
 /*   v1[2+3*l]=v1[2+3*l]+position[l]*4;   */
     for(l=0;l<=4;l++)
{
 v2[2+3*l]=a11*(gain[l]*(b*(v1[2+3*l]+position[l])
    +c*v1[1+3*l]+d*v1[3*l])-
e*v2[1+3*l]-f*v2[3*l]);
 }
/*************************************************************************
*/
/* current formation*/
    for(l=0;l<=4;l++)
      {
  current[l]=v2[2]*mix[l][0]+v2[5]*mix[l][1]+v2[8]*mix[l][2]+
     v2[11]*mix[l][3]+v2[14]*mix[l][4]+offset[l];
/*****************************************************************/
/* Data outputs to the five D/As   */
if(fabs(current[l])<10.0)
  y11=(unsigned int)(204.75*(current[l]+10.0));
 Outport=DDAbase+l*2;
_disable();
   _asm
{
   push ax;
```

```
  push dx;
  mov dx,Outport;
  mov ax,y11;
  out dx,ax;
  pop dx;
  pop ax;
 }
_enable();
   v1[3*l]=v1[1+3*l];
   v1[1+3*l]=v1[2+3*l];
   v2[3*l]=v2[1+3*l];
   v2[1+3*l]=v2[2+3*l];
     }
  _disable();
  _asm
     {
   push ax
   push dx
    mov dx,TIMER     ;Load the ADC board address
   add dx,6     ;point to adc high byte register
   or al,al     ; clear al
   in al,dx     ;output command
   mov dx,20h       ; load 8259 address
   mov al,20h       ; set normal priority, EOI=1, (65h)
   out dx,al       ; output command
   pop dx
   pop ax
}
   _enable();
}
```

Main Program

```
#define WHERE
#include <graph.h>
#include <stdio.h>
#include <conio.h>
#include <math.h>
#include <float.h>
#include <dos.h>
#include <bios.h>
#include <signal.h>
#include <stdlib.h>
#include <malloc.h>
#include "suspend.h"

void main(void)
{
/* Start up and initilization routines */
    isrsetup();/* sets up the interrupt vector */
    cosmak();/* calculate cosine values for every .5 deg */
    getdat();/* Get mixer matrix data */
    adcinitialize();/* Initialize the A/D converter board */
    initialize();/* Compute The compensators parameters */
    oscilloscope();/* Display the Controller Screen */
    timersetup();/* Set timer to interrupt at sample-rate */

/* This is the controller loop */
ch=ESC;
while(ch1!=ESC)
{
   _setcolor(4);
   _rectangle(_GFILLINTERIOR,550,420,637,477);
   _settextposition(28,71);
   printf("RUNNING");
   _settextposition(29,71);
   printf("ESC- quit");
```

```
   outp(TIMER+1,0x61);/*Starts the interrupt by loading and arming timers*/
   while(ch!=ESC)
{
keyboard();      /* Check for user input */
yawer();      /* This does the gain scheduling */
for(j=0;j<=4;++j)
{
   v=v1[2+3*j];
   if(sampler<=2)
 {
  _setcolor(14);
  _rectangle(_GFILLINTERIOR,600,330,620,325);
 }
   else
 if(sampler>200)
  {
   _setcolor(1);
   _rectangle(_GFILLINTERIOR,600,330,620,325);
  }
   if(chd!=0)
  {
   if(!trig || ix<474)
  {
    if(plotmode=='a') plotdat();
    if(plotmode=='s')
      {
 switch(ch2)
   {
     case 'x': if(j==0) plotdat();
     break;
     case 'y': if(j==1) plotdat();
     break;
     case 'z': if(j==2) plotdat();
     break;
     case 'm': if(j==3) plotdat();
     break;
```

```
      case 'n': if(j==4) plotdat();
      break;
    }
      }
  }
      }
     }
   if((trig==0)&&(ix>=475)) ix=25;/* puts oscope counter back at start */
   else ix++;   /* or advances it */
}
   _setcolor(2);
   _rectangle(_GFILLINTERIOR,550,420,637,477);
   _settextposition(28,71);
   _outtext("R-Run");
   _settextposition(29,71);
   _outtext("ESC-exit");
   outp(TIMER+1,0xdf);  /* shut down interrupt counter  */
   for(ii=0;ii<=4;++ii)
     {
     outp(DDAbase+ii*2,0);   /*  0000000 for low byte at zero  */
     outp(DDAbase+ii*2+1,8); /* 1000 for high byte at zero */
     }
   ch='.';
   ch1=(char)getch();
}
   _setvideomode(_DEFAULTMODE);
   _dos_setvect(intnum,oldnum);  /* reset interrupt vector table */
   system("d40 u");
}
```


C-2

```
#include <graph.h>
#include <stdio.h>
#include <conio.h>
#include <math.h>
#include <float.h>
#include <dos.h>
#include <bios.h>
#include <signal.h>
#include <stdlib.h>
#include <malloc.h>
#include "suspend.h"


void isrsetup(void)
{
   oldnum=_dos_getvect(intnum);      /* 0x0d IRQ-5 */
   _disable();       /* Disable all interrupts */
   _dos_setvect(intnum,isr);      /* Set new interrupt vector table */
   _enable();      /* Enable all interrupts */
   outp(0x21,0x00);      /* Send acknowledgement signal */
   outp(0x20,0x20);      /* To the interrrupt handler */
}
```

```
#include <graph.h>
#include <stdio.h>
#include <conio.h>
#include <math.h>
#include <float.h>
#include <dos.h>
#include <bios.h>
#include <signal.h>
#include <stdlib.h>
#include <malloc.h>
#include "suspend.h"


/* cosmak : calculates cos values for angles 0..360  */
void cosmak(void)
{    int i;
     double radang;

    maxnum=(int)(360.0/yawanglestep);
    yawangle=0;
    for(i=0;i<=maxnum;i++)
{
 radang=(yawangle/180.0)*pi;
 cosdat[i]=cos(radang);
 yawangle=yawangle+yawanglestep;
}
  yawangle=0;
}
```

Retrives the Mixer Matrix Data

```
#include <graph.h>
#include <stdio.h>
#include <conio.h>
#include <math.h>
#include <float.h>
#include <dos.h>
#include <bios.h>
#include <signal.h>
#include <stdlib.h>
#include <malloc.h>
#include "suspend.h"

/* getdat()  loads mixing matrix data*/
void getdat(void)
{
    int i,j,k,dim;
    FILE *fp;
    printf("Getting Data From MIX.DAT\n");
    if((fp=fopen("mix.dat","r+"))!=NULL)
    { fscanf(fp,"%d",&dim);
 printf("Number of matrices are: %d \n",dim);
 for(k=0;k<=dim-1;k++)
 {
   for(i=0;i<=4;i++) for(j=0;j<=4;j++) fscanf(fp,"%lf",&mixall[i][j][k]);
  }
printf("Data was successfully loaded\n");
 fclose(fp);
    }
  else
  fprintf(stderr,"Oops file error");
}
```

ADC board initilizer

```
#include <graph.h>
#include <stdio.h>
#include <conio.h>
#include <math.h>
#include <float.h>
#include <dos.h>
#include <bios.h>
#include <signal.h>
#include <stdlib.h>
#include <malloc.h>
#include "suspend.h"


/*********************************************************************/
void adcinitialize(void)
{
   /* DAS-40 Initialization  */
   system("d40");
   mode=0;
   flag=0;
   params[0]=0;
   MSCL_DAS40(&mode,params,&flag);
   if(flag !=0)
   {
    printf ("**** Error %u detected in mode %u ", flag & 0xff, ((flag & 0xff00)
>> 8));
    exit(1);
   }

}
```

Initilizes the controller parameters

```
#include <graph.h>
#include <stdio.h>
#include <conio.h>
#include <math.h>
#include <float.h>
#include <dos.h>
#include <bios.h>
#include <signal.h>
#include <stdlib.h>
#include <malloc.h>
#include "suspend.h"


/* initialize: setting the parameters of the dual phase advance compensator */
  void initialize(void)
{
  double dt,T,n;
  int breakfreq;
  printf("\nSample rate : 500\n");
  samplrate=500;    /* Get the sampling rate */
  printf("Compensator's Break frequency is 170hz\n");
  breakfreq=170;   /* Get the break frequency */
  dt=1/((double) samplrate);    /* period of the sampling */
  T=1/(2.0*pi*(double)(breakfreq));    /* 1/ break frequency*/
  printf("\nT= %9.6f\n\n",T);
  n=6.0;   /* ratio of lead to lag break frequency */

  a11=1/((dt*dt)+4*T*dt+4*(T*T));
  b=(dt*dt)+4*n*dt*T+4*n*n*T*T;
  c=2*dt*dt-8*n*n*T*T;
  d=(dt*dt)-4*n*dt*T+4*n*n*T*T;
  e=2*dt*dt-8*T*T;
  f=(dt*dt)-4*dt*T+4*T*T;
  printf("The compensator parameters are:\n");
```

```
 printf("a=%9.6f\n",a11);
 printf("b=%9.6f\n",b);
 printf("c=%9.6f\n",c);
 printf("d=%9.6f\n",d);
 printf("e=%9.6f\n",e);
 printf("f=%9.6f\n",f);

 getch();

}
```

```
#include <graph.h>
#include <stdio.h>
#include <conio.h>
#include <math.h>
#include <float.h>
#include <dos.h>
#include <bios.h>
#include <signal.h>
#include <stdlib.h>
#include <malloc.h>
#include "suspend.h"

/* oscilloscope: signal display screen setup   */
void oscilloscope(void)
{
  int ll,jj,gainstart,posibar;
  _setvideomode(_VRES16COLOR);
/* initialization of the screen clear array */
   for(ll=0;ll<=5;++ll)
    {
for(jj=0;jj<=500;++jj)
olddat[jj][ll]=20;
     }
/* display of the oscilloscopescreen     */
    _clearscreen(_GCLEARSCREEN);
_setcolor(13);
_rectangle(_GBORDER,0,0,639,479);
_setcolor(plotcolor);
_rectangle(_GFILLINTERIOR,20,5,480,290);
_settextposition(4,2);
printf("X");
_settextposition(7,2);
printf("Y");
_settextposition(10,2);
```

```
printf("Z");
_settextposition(13,2);
printf("M");
_settextposition(16,2);
printf("N");
_setcolor(7);
for(ll=52;ll<=244;ll=ll+48)
{
  _moveto(19,ll);
  _lineto(24,ll);
  _moveto(476,ll);
  _lineto(484,ll);
}
/*******************************************************/
/* Gain control knobs Display  */

_setcolor(12);
_rectangle(_GBORDER,2,300,480,478);
_settextposition(20,2);
_settextcolor(14);
_outtext("Feedback gain control");
_settextposition(20,40);
_outtext("Position Control");
_setcolor(5);
jj=0;

for(ll=21;ll<=29;ll=ll+2) /* lableing the gain knobs */
 {
   _settextposition(ll,27);
   printf("%4.3f",gain[jj]);
   jj++;
   _settextposition(ll,4);
   printf("0");
 }
for(ll=319;ll<=447;ll=ll+32)  /* gain control bar display */
   _rectangle(_GBORDER,42,ll,196,ll+22);
```

```
jj=0;
for(ll=320;ll<=448;ll=ll+32)    /* Presetting gain display */
{   /* on the screen     */
    gainstart=(int)(gain[jj]*80)+40;
    _setcolor(6);
    _rectangle(_GFILLINTERIOR,40,ll,gainstart+3,ll+20);
    ++jj;
}
/* display of position control knobs  */

_setcolor(4);
 for(ll=320;ll<=448;ll=ll+32)      /* position control bar display */
    _rectangle(_GBORDER,310,ll,460,ll+20);
_setcolor(10);
jj=0;
for(ll=320;ll<=448;ll=ll+32)
  {posibar=385+(int)(position[jj]*75);
_rectangle(_GFILLINTERIOR,posibar,ll,posibar+1,ll+20);
++jj;
}
/*************************************/
/* labling of gain knob display    */

_settextposition(21,2);
_outtext("X");     /* x or axial */
_settextposition(23,2);
_outtext("Y");     /* y or side  */
_settextposition(25,2);
_outtext("Z");     /* z or vertical */
_settextposition(27,2);
_outtext("M");     /* M - pitch movement */
_settextposition(29,2);
_outtext("N");    /*N - yaw movement */
/************************************/

/* Display of options */
```

```
_settextposition(2,62);
printf("E- enable plot");
_settextposition(3,62);
printf("D- disable plot");
_settextposition(5,62);
printf("Select Channel");
_settextposition(6,62);
printf("  Press ");
_settextposition(7,62);
printf(" X Y Z M N");
_settextposition(9,62);
printf("S- single plot") ;
_settextposition(10,62);
printf("A - all plots");
_settextposition(11,62);
printf("T - triggered ");
_settextposition(12,62);
printf("C - continuous");
_settextposition(13,62);
printf("F1- unzoom plot");
_settextposition(14,62);
printf("F2- zoom plot");
_settextposition(15,62);
printf("F4- Save data");
_settextposition(16,62);
printf("F5- - Yaw ");
_settextposition(17,62);
printf("F6- + Yaw ");
_settextposition(18,62);
printf("F9-  AutoYaw off");
 _settextcolor(4);
 _settextposition(18,75);
 _outtext("off");
_settextposition(19,62);
printf("F10- AutoYaw on");
_settextposition(24,62);
```

```
printf("Yaw Error:");
_settextposition(22,62);
printf("Yaw Angle:");
}
```

Timer board setup

```
#include <graph.h>
#include <stdio.h>
#include <conio.h>
#include <math.h>
#include <float.h>
#include <dos.h>
#include <bios.h>
#include <signal.h>
#include <stdlib.h>
#include <malloc.h>
#include "suspend.h"

/* timersetup: sets up the sample rate clock on the second ADC for interrupt */
void timersetup(void)
{
  int rate;
/*    Timer ic AM9513 setting  */
    outp(TIMER+1,0xdf); /* disarms counters */
    outp(TIMER+1,23); /* sets next outp to the master register*/
    outp(TIMER,0xc0); /* master rgister low byte */
    outp(TIMER,0x41); /* master register high byte */
    outp(TIMER+1,0x01); /* sets next outp to the counter 1 register */
    outp(TIMER,0x22); /* sets wave form and base frequency */
    outp(TIMER,0x0c);
    rate=(int)(31250/samplrate);
    outp(TIMER+1,0x09); /* sets next outp to the load register */
    outp(TIMER,rate); /* it will count down from rate and set off an interrupt
*/
    outp(TIMER+1,0x61); /* load and arm  */
}
```

```
#include <graph.h>
#include <stdio.h>
#include <conio.h>
#include <math.h>
#include <float.h>
#include <dos.h>
#include <bios.h>
#include <signal.h>
#include <stdlib.h>
#include <malloc.h>
#include "suspend.h"

/* keyboard: User interaction through keyboard*/
void keyboard(void)
{
  if(kbhit()!=0)
    {
     chin=(char)getch();
     switch(chin)
{
  case 'X':case 'x': ch2='x';
 _setcolor(8);
 _rectangle(_GFILLINTERIOR,35,320,39,478);
 _setcolor(12);
 _rectangle(_GFILLINTERIOR,35,320,39,340);
 if(plotmode=='s')
 {
    _setcolor(plotcolor);
    _rectangle(_GFILLINTERIOR,20,5,480,290);
 }
 break;
  case 'Y':case 'y': ch2='y';
 _setcolor(8);
 _rectangle(_GFILLINTERIOR,35,320,39,478);
```

```
_setcolor(12);
_rectangle(_GFILLINTERIOR,35,352,39,372);
if(plotmode=='s')
{
   _setcolor(plotcolor);
   _rectangle(_GFILLINTERIOR,20,5,480,290);
}
break;
 case 'Z':case 'z': ch2='z';
_setcolor(8);
_rectangle(_GFILLINTERIOR,35,320,39,478);
_setcolor(12);
_rectangle(_GFILLINTERIOR,35,384,39,404);
if(plotmode=='s')
{
   _setcolor(plotcolor);
   _rectangle(_GFILLINTERIOR,20,5,480,290);
}
break;
case 'M':case 'm': ch2='m';
_setcolor(8);
_rectangle(_GFILLINTERIOR,35,320,39,478);
_setcolor(12);
_rectangle(_GFILLINTERIOR,35,416,39,436);
if(plotmode=='s')
{
   _setcolor(plotcolor);
   _rectangle(_GFILLINTERIOR,20,5,480,290);
}
break;
 case 'N':case 'n': ch2='n';
_setcolor(8);
_rectangle(_GFILLINTERIOR,35,320,39,478);
_setcolor(12);
_rectangle(_GFILLINTERIOR,35,448,39,468);
if(plotmode=='s')
```

```
{
   _setcolor(plotcolor);
   _rectangle(_GFILLINTERIOR,20,5,480,290);
}
break;
 case 'S': case 's': plotmode='s';
_setcolor(plotcolor);
_rectangle(_GFILLINTERIOR,20,5,480,290);
break;
 case 'A': case 'a': plotmode='a';
break;
 case 'T': case 't':trig=1;
break;
 case 'C': case 'c': trig=0;
break;
 case 0:
    chin=(char)getch();
    switch(chin)
    {
case RIGHT: gainupdown=1;
 gainplot();
 break;
case LEFT: gainupdown=-1;
 gainplot();
 break;
case UP: positionsign=1;
 positionplot();
 break;
case DOWN: positionsign=-1;
 positionplot();
 break;
case F1: if(zoom>1) zoom=zoom-1;
 _settextposition(1,60);
 printf("%i ",zoom);
 break;
case F2: if(zoom<10) zoom=zoom+1;
```

```
 _settextposition(1,60);
 printf("%i ",zoom);
 break;
case F4:savedat();  /* Save step responses if needed */
 break;
case F5:yawangle=yawangle-yawanglestep;
 newmixer();
 break;
case F6:yawangle=yawangle+yawanglestep;
 newmixer();
 break;
case F9: rota=0;
 _settextcolor(8);
 _settextposition(19,75);
 _outtext("on");
 _settextcolor(4);
 _settextposition(18,75);
 _outtext("off");
 break;
case F10: rota=1;
 _settextcolor(8);
 _settextposition(18,75);
 _outtext("off");
 _settextcolor(4);
 _settextposition(19,75);
 _outtext("on");
 break;
  }
  break;
  case 'E':case 'e':  chd=1;
 break;
  case 'D':case 'd': chd=0;
 break;
  case ESC: ch=ESC; break;
 } }
     chin='.';}
```

Yaw Interpolation Program

```
#include <graph.h>
#include <stdio.h>
#include <conio.h>
#include <math.h>
#include <float.h>
#include <dos.h>
#include <bios.h>
#include <signal.h>
#include <stdlib.h>
#include <malloc.h>
#include "suspend.h"



void yawer(void)
{
   int loop1,loop2;

   y=0.0;
   xy=0.0;
   x2y=0.0;


   for(loop1=0;loop1<=10;loop1++)
{
function=0.0;

for(loop2=0;loop2<5;loop2++)
      {
      _settextposition(25,0);
      ag=(double)((theta[loop2]-testangle+(double)(loop1-5))
 *(double)3.14/(double)180.0);
      function=(current[loop2]*cos(ag))+function;
      }
```

```
y=y+function;
xy=xy+function*(loop1-5);
x2y=x2y+(double)((loop1-5)*(loop1-5))*function;
}

    correction=((42.9*xy)/(110.0*y-11.0*x2y));
    if(fabs(correction)>5) correction=0.0;

    ag=testangle - correction;

    testangle=.97*testangle + .03*ag;

    if((testangle >=360)||(testangle<=-360)) testangle=0;

    _settextposition(24,72);
    printf("%4.1lf ",testangle);
autoyaw=testangle-yawangle;
if(rota==1) /* Check for automatic yaw sensing */
    { /* option  */
    if(autoyaw>=.5)
{
  yawangle=yawangle+yawanglestep;
  newmixer();
}
     if(autoyaw<=-.5)
{
  yawangle=yawangle-yawanglestep;
  newmixer();
}
    }
}
```

Calculates the New Mixer Matrix

```
#include <graph.h>
#include <stdio.h>
#include <conio.h>
#include <math.h>
#include <float.h>
#include <dos.h>
#include <bios.h>
#include <signal.h>
#include <stdlib.h>
#include <malloc.h>
#include "suspend.h"



/* newmixer : calculates new mixing matrix and current distribution */
void newmixer(void)
{  int i,j,nextone,index1,index2,index3,index4,index5,seventy2;
   int angindex;
   double angstep=6, yawangletemp;
   double interpol;

   angindex=(int)(yawangle/yawanglestep);/* Index of the yaw angle  */
   seventy2=(int)(72.0/yawanglestep);/* Index of 72 deg based on*/
   nextone=1;/* the yaw step chosen   */
   if (angindex<0)
    {
     angindex=maxnum+angindex;
     nextone=-1;
    }
   index1=angindex;    /*determination of cosine matrix  */
   index2=abs(seventy2-angindex);    /*index for calculating the zero  */
```

```
index3=abs(2*seventy2-angindex);    /*position current. Every index*  */
index4=abs(3*seventy2-angindex);    /*is 72 degrees apart from its  */
index5=abs(4*seventy2-angindex);    /*adjasent neighbor.  */

if(index1>maxnum)    /* The five if statements rap around */
  index1=index1-maxnum;
if(index2>=maxnum)    /* the indceis to the begining after */
  index2=index2-maxnum;
if(index3>=maxnum)    /* 360 degrees rotation  */
  index3=index3-maxnum;
if(index4>=maxnum)
  index4=index4-maxnum;
if(index5>=maxnum)
  index5=index5-maxnum;


offset[0]=MAXCUR*cosdat[index1];  /*  Levitation current calculation  */
offset[1]=MAXCUR*cosdat[index2];  /*  using the cosine array, which */
offset[2]=MAXCUR*cosdat[index3];  /*  selected depending on the angle */
offset[3]=MAXCUR*cosdat[index4];  /*  of the model.Initialoffset[0] */
offset[4]=MAXCUR*cosdat[index5];  /*  is the highest steady current.  */

yawangletemp=fabs(yawangle);

if(yawangle>=360)  /* Adjustment for angles>360*/
   yawangletemp=yawangle-360;

ang=(int)(yawangletemp/angstep);/* Determination of number of steps */
nextone=1;
/* Determination of the interpolating coefficent */
interpol=((yawangletemp)-(double)((ang)*angstep))/angstep;

 if(yawangle<0)
 {
  nextone=-1;
  ang=60-ang;
```

```
    }

/* Interpolation between the mixing matrices */
    for(i=0;i<=4;i++)
 {
    for(j=0;j<=4;j++)
 {
mix[i][j]=interpol*(mixall[i][j][ang+nextone]
-mixall[i][j][ang])
+mixall[i][j][ang];
 }
 }
_settextposition(22,72);
      printf("%5.2f",yawangle);
}
```

Graphical presentation of the degrees of freedom

```
#include <graph.h>
#include <stdio.h>
#include <conio.h>
#include <math.h>
#include <float.h>
#include <dos.h>
#include <bios.h>
#include <signal.h>
#include <stdlib.h>
#include <malloc.h>
#include "suspend.h"

/* gainplot: to display gain on each degree of freedom graphically */
void gainplot(void)
{
int gainknob,knobcolor;
switch(ch2)
{
  case 'x':
    if((gain[0]>=0) && (gain[0]<=2))
     {
knobcolor=3*(gainupdown+1);
_setcolor(knobcolor);
gainknob=40+(int)(gain[0]*80.0);
_rectangle(_GFILLINTERIOR,gainknob,320,gainknob+3,340);
gain[0]=gain[0]+gainupdown*gainstep;
_settextposition(21,27);
printf("%4.3f",gain[0]);

     }
    else
 if(gain[0]<-5) gain[0]=-5;
 if(gain[0]>5) gain[0]=5;
break;
```

```
   case 'y':
      if((gain[1]>=0) && (gain[1]<=2))
{
_setcolor(knobcolor);
gainknob=40+(int)(gain[1]*80.0);
_rectangle(_GFILLINTERIOR,gainknob,352,gainknob+3,372);
gain[1]=gain[1]+gainupdown*gainstep;
_settextposition(23,27);
printf("%4.3f",gain[1]);
      }
     else
  if(gain[1]<0) gain[1]=0;
  if(gain[1]>2) gain[1]=2;
    break;
   case 'z':
      if((gain[2]>=0) && (gain[2]<=2))
{
knobcolor=3*(gainupdown+1);
_setcolor(knobcolor);
 gainknob=40+(int)(gain[2]*80.0);
 _rectangle(_GFILLINTERIOR,gainknob,384,gainknob+3,404);
 gain[2]=gain[2]+gainupdown*gainstep;
 _settextposition(25,27);
 printf("%4.3f",gain[2]);
}
     else
  if(gain[2]<0) gain[2]=0;
  if(gain[2]>2) gain[2]=2;
    break;
   case 'm':
     if((gain[3]>=0) && (gain[3]<=2))
     {
knobcolor=3*(gainupdown+1);
_setcolor(knobcolor);
gainknob=40+(int)(gain[3]*80.0);
_rectangle(_GFILLINTERIOR,gainknob,416,gainknob+3,436);
```

```
gain[3]=gain[3]+gainupdown*gainstep;
_settextposition(27,27);
printf("%4.3f",gain[3]);
      }
      else
  if(gain[3]<0) gain[3]=0;
  if(gain[3]>2) gain[3]=2;
    break;
   case 'n':
      if((gain[4]>=0) && (gain[4]<=2))
      {
knobcolor=3*(gainupdown+1);
_setcolor(knobcolor);
gainknob=40+(int)(gain[4]*80.0);
_rectangle(_GFILLINTERIOR,gainknob,448,gainknob+3,468);
gain[4]=gain[4]+gainupdown*gainstep;
_settextposition(29,27);
printf("%4.3f",gain[4]);
      }
       else
  if(gain[4]<0) gain[4]=0;
  if(gain[4]>2) gain[4]=2;
    break;


}
}
```

```
#include <graph.h>
#include <stdio.h>
#include <conio.h>
#include <math.h>
#include <float.h>
#include <dos.h>
#include <bios.h>
#include <signal.h>
#include <stdlib.h>
#include <malloc.h>
#include "suspend.h"

/* Plotdat: plot of signals */
void plotdat(void)
{
   int plotcenter=148;
   if (plotmode=='s')
   kv=plotcenter-(int)(v*10*zoom);
   else
   kv=4+(j+1)*48-(int)(v*2*zoom);
   if(kv>=290) kv=290;
   _setcolor(plotcolor);
   _setpixel(ix,olddat[ix][j+1]);
   _setcolor(datcolor);
   _setpixel(ix,kv);
   olddat[ix][j+1]=kv;
}
```

Plots Position Knobs

```
#include <graph.h>
#include <stdio.h>
#include <conio.h>
#include <math.h>
#include <float.h>
#include <dos.h>
#include <bios.h>
#include <signal.h>
#include <stdlib.h>
#include <malloc.h>
#include "suspend.h"


/* positionplot: control and display of position knobs    */
void positionplot(void)
{
      int knobcolor,positionknob,bkcolor;
      knobcolor=10;
      bkcolor=4;
if(trig==1)
  ix=25;
  sampler=0;  /* counter for collecting data in ISR */

switch(ch2)
{

  case 'x':
_setcolor(0);
_rectangle(_GFILLINTERIOR,310,320,461,340);
_setcolor(bkcolor);
_rectangle(_GBORDER,310,320,461,340);
_setcolor(knobcolor);
position[0]=position[0]+positionsign*positionstep;
if(position[0]<-1) position[0]=-1;
```

```
if(position[0]>1) position[0]=1;
positionknob=385+(int)(position[0]*75.0);
_rectangle(_GFILLINTERIOR,positionknob,320,positionknob+1,340);
break;
  case 'y':
_setcolor(0);
_rectangle(_GFILLINTERIOR,310,352,461,372);
_setcolor(bkcolor);
_rectangle(_GBORDER,310,352,461,372);
_setcolor(knobcolor);
position[1]=position[1]+positionsign*positionstep;
if(position[1]<-1) position[1]=-1;
if(position[1]>1) position[1]=1;
positionknob=385+(int)(position[1]*75.0);
_rectangle(_GFILLINTERIOR,positionknob,352,positionknob+1,372);
break;
  case 'z':
_setcolor(0);
_rectangle(_GFILLINTERIOR,310,384,461,404);
_setcolor(bkcolor);
_rectangle(_GBORDER,310,384,461,404);
_setcolor(knobcolor);
position[2]=position[2]+positionsign*positionstep;
if(position[2]<-1) position[2]=-1;
if(position[2]>1) position[2]=1;
positionknob=385+(int)(position[2]*75.0);
      _rectangle(_GFILLINTERIOR,positionknob,384,positionknob+1,404);
   break;
  case 'm':
_setcolor(0);
_rectangle(_GFILLINTERIOR,310,416,461,436);
_setcolor(bkcolor);
_rectangle(_GBORDER,310,416,461,436);
_setcolor(knobcolor);
position[3]=position[3]+positionsign*positionstep;
if(position[3]<-1) position[3]=-1;
```

```
if(position[3]>1) position[3]=1;
positionknob=385+(int)(position[3]*75.0);
      _rectangle(_GFILLINTERIOR,positionknob,416,positionknob+1,436);
   break;
  case 'n':
_setcolor(0);
_rectangle(_GFILLINTERIOR,310,448,461,468);
_setcolor(bkcolor);
_rectangle(_GBORDER,310,448,461,468);
_setcolor(knobcolor);
position[4]=position[4]+positionsign*positionstep;
if(position[4]<-1) position[4]=-1;
if(position[4]>1) position[4]=1;
positionknob=385+(int)(position[4]*75.0);
 _rectangle(_GFILLINTERIOR,positionknob,448,positionknob+1,468);
 break;


 }
 }
```

Saves Step Responses

```
#include <graph.h>
#include <stdio.h>
#include <conio.h>
#include <math.h>
#include <float.h>
#include <dos.h>
#include <bios.h>
#include <signal.h>
#include <stdlib.h>
#include <malloc.h>
#include "suspend.h"


/*************************************************************************/
/* savedat() saves data to file    */
void savedat(void)
{
     char fname[12];
      int i,j;
      FILE *fp;
   _settextposition(25,64);
   printf("Save data? y/n\n");
   ch=(char)getch();
   if((ch=='Y') || (ch=='y'))
     {
_settextposition(25,64);
printf("Enter File Name");
_settextposition(26,64);
scanf("%s",&fname);
if((fp=fopen(fname,"w+"))!=NULL)
  {
   fprintf(fp,"stepdat=[\n");
   for(i=1;i<=200;i++)
  {
```

```
    for(j=0;j<=4;j++)
    fprintf(fp,"%le ",savdat[j][i]);
    fprintf(fp,"\n");
   }
    fprintf(fp,"];\n");
    fprintf(fp,"gains=[\n");
    for(i=0;i<=4;i++)
    fprintf(fp,"%le \n",gain[i]);  /* Save the gains */
    fprintf(fp,"];\n");
    fclose(fp);
   }
  else
  printf("Oops file error");
      }

}
```

APPENDIX B

Eddy Current Simulation

```
clear
format short e

% plant model starting from the analytical model (ls_modl)
load ls_modl; load varibles;

% Eddy current effects
num=[2.1749e-3 1];
den=[2.934e-3  1];
[ai,bi,ci,di]=tf2ss(num,den);
[aii,bii,cii,dii]=append(ai,bi,ci,di,ai,bi,ci,di);
[aiii,biii,ciii,diii]=append(aii,bii,cii,dii,aii,bii,cii,dii);
[ae,be,ce,de]=append(aiii,biii,ciii,diii,ai,bi,ci,di);

% adding the dynamics to the plant
[ap,bp,cp,dp]=series(ae,be,ce,de,A,B,C,D);

% loading the A B C D matrices of the controller
load lqg.mat
```

```
% begin putting the model together
[ad,bd,cd,dd]=c2dm(ap,bp,cp,dp,T,'zoh');
[atotal,btotal,ctotal,dtotal]=series(ac,bc,cc,dc,ad,bd,cd,dd);
[as,bs,cs,ds]=cloop(atotal,btotal,ctotal,dtotal,-1);

% Correcting inputs and output of system for position
bs=bs*p2s;
cs=s2p*cs;
ds=s2p*ds*p2s;

% Performing the simulated step response
[x,y]=dstep(as,bs,cs,ds,1,40);
plot(x(:,1))
```

APPENDIX C

Numeric Values For Eddy Current Loops

This list represents the initial startup values calculated from theory.

Main Coil

$Kz = 8.7736 \times 10^{-5}$

Ring

$Lm = 3.873 \times 10^{-7}$ $Le = 3.8573 \times 10^{-6}$ $Re = 2.243 \times 10^{-4}$ $Ke = 4.29 \times 10^{-6}$

Baseplate

$Lm = 4.9056 \times 10^{-5}$ $Le = 5.2523 \times 10^{-9}$ $Re = 1.4 \times 10^{-5}$ $Ke = 8.508 \times 10^{-10}$

Core

$Lm = 4.9056 \times 10^{-5}$ $Le = 4.4857 \times 10^{-7}$ $Re = 6 \times 10^{-3}$ $Ke = 4.601 \times 10^{-10}$

Appendix D

OPERA Field Calculating Files

The data files are generated using the file gradient.comi, which is listed below, in OPERA. After the data files are generated the MATLAB script file gradient.m is used to find the field gradients.

GRADIENT.COMI

```
grid -#1/2 0 -#1/10 #1/10 #1/10 #1/10 11 1 3 fieldx.dat no
grid 0 -#1/2 0  #1/10 #1/10 #1/10 1 11 1 fieldy.dat no
grid 0 0 -#1/2 #1/10 #1/10 #1/10 1 1 11 fieldz.dat no
```

GRADIENT.M

```
clear
format long e
load fieldx.dat
load fieldy.dat
load fieldz.dat
fieldx=fieldx(:,1:6);
fieldy=fieldy(:,1:6);
fieldz=fieldz(:,1:6);
bx=fieldy(6,4);
by=fieldy(6,5);
bz=fieldy(6,6);
fieldx1=fieldx([1 4 7 10 13 16 19 22 25 28 31],:);
fieldx2=fieldx([2 5 8 11 14 17 20  23 26 29 32],:);
fieldx3=fieldx([3 6 9 12 15 18 21 24 27 30 33],:);
fieldx1(:,1)=fieldx1(:,1)-fieldx1(6,1);
fieldx2(:,1)=fieldx2(:,1)-fieldx2(6,1);
fieldx3(:,1)=fieldx3(:,1)-fieldx3(6,1);
fieldy(:,2)=fieldy(:,2)-fieldy(6,2);
```

```
fieldz(:,3)=fieldz(:,3)-fieldz(6,3);
ans1=polyfit(fieldx2(:,1),fieldx2(:,4),2);
bxdx=ans1(1,2);
bxdxdx=2*ans1(1,1);
ans2=polyfit(fieldx2(:,1),fieldx2(:,5),2);
bydx=ans2(1,2);
bxdxdy=2*ans2(1,1);
ans3=polyfit(fieldx2(:,1),fieldx2(:,6),2);
bxdz=ans3(1,2);
bxdxdz=2*ans3(1,1);
ans4=polyfit(fieldy(:,2),fieldy(:,4),2);
bxdy=ans4(1,2);
bxdydy=2*ans4(1,1);
ans5=polyfit(fieldy(:,2),fieldy(:,5),2);
bydy=ans5(1,2);
bydydy=2*ans5(1,1);
ans6=polyfit(fieldy(:,2),fieldy(:,6),2);
bydz=ans6(1,2);
bydydz=2*ans6(1,1);
ans7=polyfit(fieldz(:,3),fieldz(:,4),2);
bzdx=ans7(1,2);
bxdzdz=2*ans7(1,1);
ans8=polyfit(fieldz(:,3),fieldz(:,5),2);
bzdy=ans8(1,2);
bydzdz=2*ans8(1,1);
ans9=polyfit(fieldz(:,3),fieldz(:,6),2);
bzdz=ans9(1,2);
bzdzdz=2*ans9(1,1);
ans10=polyfit(fieldx1(:,1),fieldx1(:,5),2);
bydx1=ans10(1,2);
ans11=polyfit(fieldx3(:,1),fieldx3(:,5),2);
bydx3=ans11(1,2);
l=fieldx2(1,3)-fieldx1(1,3);
g=[-l , bydx1
0 , bydx
l  , bydx3 ];
```

```
ans12=polyfit(g(:,1),g(:,2),2);
bxdydz=ans12(1,2);
save data.ma bx by bz bxdx bxdy bxdz bydy bydz bzdz bxdxdx bxdxdy bxdxdz
bxdydy bxdydz bxdzdz bydydy bydydz bydzdz bzdzdz -ascii
```