NASA-CR-191890

Old Dominion University Research Foundation

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

APPROACHES TO CONTROL OF THE LARGE ANGLE MAGNETIC SUSPENSION TEST FIXTURE

By

Mehran Ghofrani, Graduate Research Assistant

Principal Investigator: Colin P. Britcher

Annual Progress Report
For the period November 1, 1991 to October 31, 1992

Prepared for
National Aeronautics and Space Administration
Langley Research Center
Hampton, Virginia 23681-2199

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

December 1992

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

APPROACHES TO CONTROL OF THE LARGE ANGLE MAGNETIC SUSPENSION TEST FIXTURE

By

Mehran Ghofrani, Graduate Research Assistant

Principal Investigator: Colin P. Britcher

Annual Progress Report
For the period November 1, 1991 to October 31, 1992

Prepared for
National Aeronautics and Space Administration
Langley Research Center
Hampton, Virginia 23681-2199

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

Submitted by the
Old Dominion University Research Foundation
P.O. Box 6369
Norfolk, Virginia 23508-0369

December 1992

ABSTRACT

Approaches to Control of The Large Angle Magnetic Suspension Test Fixture

Mehran Ghofrani

Old Dominion University, 1992

Advisor: Dr. Colin P. Britcher

The Large Angle Magnetic Suspension Test Fixture is a five degree-of-freedom system, developed and built at NASA Langley Research Center. It is intended for study of control techniques in magnetic suspension systems with large angular capabilities. In this study, steps have been taken to prove the system in practice, using the existing hardware. A classical control approach, using dual phase advance compensators, is applied in simulation and hardware. A single decoupled degree-of-freedom of the system is stabilized and controlled in simulation. The procedure is then employed for all five degrees-of-freedom. The design and implementation of an analog and a digital controller are described. Results from simulation and the actual system are compared and analyzed. The ability of the system to sustain suspension over a large angular range has been proven in hardware.

ACKNOWLEDGMENTS

This thesis is being submitted in lieu of a progress report for the research project entitled, "Large Angle Magnetic Suspension Test Fixture," supported by the National Aeronautics and Space Administration, research grant NAG-1-1056, Nelson J. Groom, Guidance and Control Division, Spacecraft Controls Branch, is technical monitor.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

a, b, c, d, e	constants of the digital filter
A, B	state-space A and B matrices of the plant
$\widetilde{A}$, $\widetilde{B}$	state space matrices of the power amplifiers and coils combination
$\mathcal{A}$, $\mathcal{B}$	A and B matrices without $\mathcal{W}_2$
$\mathcal{A}\mathcal{A}_c$, $\mathcal{B}\mathcal{B}_c$, $\mathcal{C}\mathcal{C}_c$, $\mathcal{D}\mathcal{D}_c$	state space system for five DPA compensators in parallel
$\mathcal{A}_c$, $\mathcal{B}_c$, $\mathcal{C}_c$, $\mathcal{D}_c$	state space system for a DPA compensator
$\vec{B}_0$	magnetic field generated by the coils at the centroid of the core, T
$\vec{B}$	magnetic field generated by the electromagnets, T
B_i	magnetic field component in i direction, T
B_{ij}	magnetic field gradient (∂B_i / ∂j), T/m
$[\partial B]$	field gradient matrix, T/m
$\partial\vec{B}$	field gradient vector, T/m
dt	sample rate of the digital controller
$\mathcal{E}$	open loop plant eigenvalues
$\vec{\bar{F}}$	force vector on the permanent magnetic core, N
$\vec{F}$	force due to the electromagnets on the magnetic core, N
G_z	plant yaw degree-of-freedom transfer function
G_{zp}	yaw transfer function with coil and amplifier
g	gravitational constant, m/s
g_{sensor}	row matrix containing the sensor gains, V/m
g_1, g_3	the gains of sensors one and three, V/m
g_{ave}	average of sensors 1 and 3, or sensors 4 and 5, V/m

g_{ol}	the decoupled gains for each degree-of-freedom, V/m, V/rad
H	phase advance compensator transfer function
H_{DPA}	dual phase advance compensator transfer function
$\vec{\bar{H}}$	angular momentum vector in core coordinates, kg-m^2/s
$\dot{\vec{\bar{H}}}$	torque vector in core coordinates, N-m
$\vec{I}$	current vector containing the five coil current, A
$\vec{I}_0$	zero position current vector, A
I_0	B_x due to $\vec{I}_0$
I_{max}	current at which fields and field gradients were calculated, A
I_i	current through coil i, A
I_y, I_z, I_c	moment of inertia about the $\bar{y}$ or $\bar{z}$ axis, kg-m^2
[I]	moment of inertia matrix, kg-m^2
$[I]^{-1}$	inverse of moment of inertia matrix
I_{yaw}	plant input current due to yaw, A
K, k	forward and feedback gain of the current amplifiers
$[K_i]$	row vector containing field component in i due to each coil
$[K_B]$	field matrix containing the field components due to each coil,
$[K_{\partial B}]$	gradient matrix containing gradient components due to each coil
L_i	self inductance of coil i, Hen
L_{ij}	mutual inductance between coil i and j
[L]	inductance matrix, Henries
$\vec{\bar{M}}$	core magnetization vector in core coordinates
$M_{\bar{x}}$	core magnetization x component in the core coordinate
$[\bar{M}]$	core's magnetization cross product matrix
m_c	mass of the suspended element, kg
$\tilde{m}_{ij}$	interpolated mixer matrix element (i,j)
m_{ijk}	mixer matrix element (i,j) at yaw angle step j

R_i	resistance of coil i, Ω
T	the period of the compensator's break frequency, s
$\vec{\bar{T}}$	torque vector in core coordinates, N-m
$\vec{T}$	torque due to the electromagnets on the magnetic core, N-m
$[T_m]$	coordinate transformation matrix
$[T_m]^{-1}$	inverse of the coordinate transformation matrix
U_k	compensator's discrete-time input at sample k
$\vec{\bar{V}}$	velocity vector of the suspended element, m/s
$\dot{\vec{\bar{V}}}$	time derivative of velocity vector of the suspended element, m/s
$\forall$	volume of the magnetic core, m^3
V_i	voltage across coil i terminal, V
V_{Di}	demand voltage to current amplifier i, V
$\mathcal{W}_2$	constant matrix containing the core's specifications
$\hat{X}$	state variable vector
$\hat{X}_0$	state variables evaluated at zero position
x, y, z	inertial coordinate system
$\bar{x}$, $\bar{y}$, $\bar{z}$	core coordinate system
x_b, y_b, z_b	electromagnet coordinate system
Y_k	compensator's discrete time output
α	compensator break frequency ratio constant
∇	gradient operator
θ_{actual}	actual yaw angle of the sensor frame, rads
θ_k	yaw angle at calculated intervals, rads
θ_i	angular position about i axis, rads
$\ddot{\theta}_i$	angular acceleration about i axis, $rads/s^2$
$\vec{\bar{\Omega}}$	angular velocity vector in core coordinates, rads/s
$\dot{\vec{\bar{\Omega}}}$	time derivative of velocity vector in core coordinates, $rads/s^2$

$[\bar{\Omega}]$	cross product matrix
Ω_i	angular velocity about i axis, rads/s

CHAPTER 1

INTRODUCTION

The concept of magnetic levitation and magnetic suspension is not new in origin. Many successful attempts have been made throughout this century. According to reference 1, Kemper suspended a mass of 210 kg in 1935. Advancements in control systems and electronics have greatly increased the interest and achievements in this area. The first practical actively controlled magnetic suspension, built at the University of Virginia in 1937 [2], is the basis of most modern magnetic suspension applications. Magnetic suspension can be applied to many fields, some of which are contact-less magnetic bearings [3], levitated trains [4], supports for wind tunnel models [5], and vibration isolation [6].

In some literature the term "suspension" is used in situations where an object is supported by magnetic forces from above, and the term "levitation" is used for cases where the object is repelled from forces below. There are cases where a combination of the two may be employed. However, in this study the two terms will be used interchangeably.

There are two major groups of applications of magnetic suspension, small gap and large gap. The term "gap" refers to the ratio of distance between the suspended element and the actuator to the actuator size. The two groups differ in analysis methods. In a small gap magnetic suspension system, the field at the surface of the actuator may be assumed uniform, and magnetic circuit theory may

be utilized [7]. A large gap magnetic suspension system may require a full three-dimensional analysis of the magnetic field at the point of suspension [8]. The gap can be as low as a fraction of a mm in small gap applications and as large as several meters in large gap systems. An example of a small gap application is magnetic bearings, and of a large gap application is the wind tunnel support systems. Here only large gap magnetic levitation is discussed.

A simple classical example of magnetic suspension is the case of a steel ball suspended by an electromagnetic coil above it. As shown in figure 1.1, the ball's weight is supported by the magnetic field generated by the current flowing through the coil. This system is inherently unstable, therefore an active feedback controller is necessary to stabilize the system. A simple optical sensor can be used to detect the position of the steel ball, and the signal is fed back to the controller. The first stage of the controller is usually a comparator, which compares the actual position to the desired position. The error from the comparator is passed through some form of a lead compensator, to accommodate the lag due to the coil and the mass of the ball. This produces a feedback signal based on the rate of the error signal. The compensated signal is finally fed to a current power amplifier which supplies the coil [9].

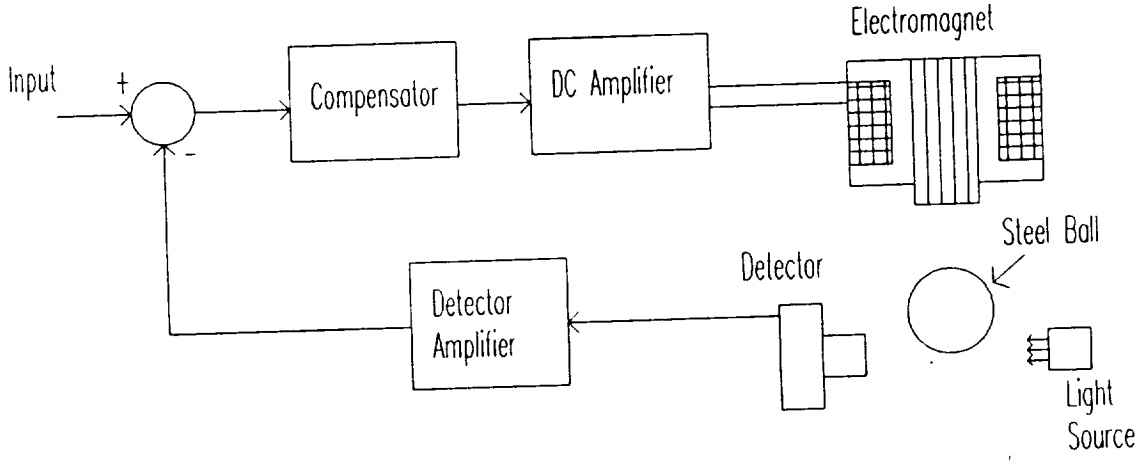


Figure 1.1- Single degree-of-freedom steel ball suspension

The example given above has only one degree-of-freedom but is representative of the many complex systems possible. One such system is the Large Angle Magnetic Suspension Test Fixture (LAMSTF). The LAMSTF is a research apparatus developed at NASA Langley Research Center. It is designed to control a suspended element in five degrees-of-freedom, with the capability of full 360° rotation about its vertical axis. The technology is adoptable to applications such as vibration isolation and accurate positioning systems [10].

In this phase of the research the primary goal is to make the presently available hardware operational. This is a proof-of-concept step leading to later studies of performance maximization [11]. After describing the available hardware and its modeled dynamics, the steps leading to implementation of an analog and a digital controller for LAMSTF are presented. In the final phase, the large angular capability of this system is demonstrated in practice.

CHAPTER 2

THE LARGE ANGLE MAGNETIC SUSPENSION TEST FIXTURE

2.1 General Description

The distinct design feature of LAMSTF is its "planar array" electromagnetic coil arrangement. Five coils are mounted on the perimeter of a circle of about 13.77 cm radius, at a spacing of 72° apart, on a 1/2" thick, square aluminum plate. All the coils are positioned with their axes parallel to the axis of the circle. The aim of the design is to levitate an element, filled with permanent magnet material, at a height of about 10 cm above the coils. A sketch of this arrangement is shown in figure 2.1. Since there are no electromagnets above, all the suspension forces would be repulsive, and directed from below a horizontal plane over the coils. The pure repulsive force suspension method is one of the unusual features of this system [12][13]. The system would be required to control the suspended element in five degrees-of-freedom, and this has required a minimum of five electromagnetic actuator coils [14].

In this chapter the details of the plant are described. The plant primarily consists of the suspended element, the sensors, the coils, and the power amplifiers.

2.2 The Suspended Element

The suspended element is an aluminum tube about 5.32 cm long and 0.9525 cm outside diameter. The tube is filled with 16 wafers of Neodymium-Iron-Boron (Nd-Fe-B) permanent magnet material. Each magnetic wafer is 0.7963 cm in

diameter and 0.3135 cm in thickness, having a magnetization of about 9.5493×10^{-5} A/m. The wafers are arranged in N-S-N-S sequence and are epoxied into the aluminum tube, resulting in a total mass of about 22.5 grams. Using the bifilar pendulum method, the moment of inertia of the model about its transverse axis was found to be approximately 5.508×10^{-6} kg.m^2.

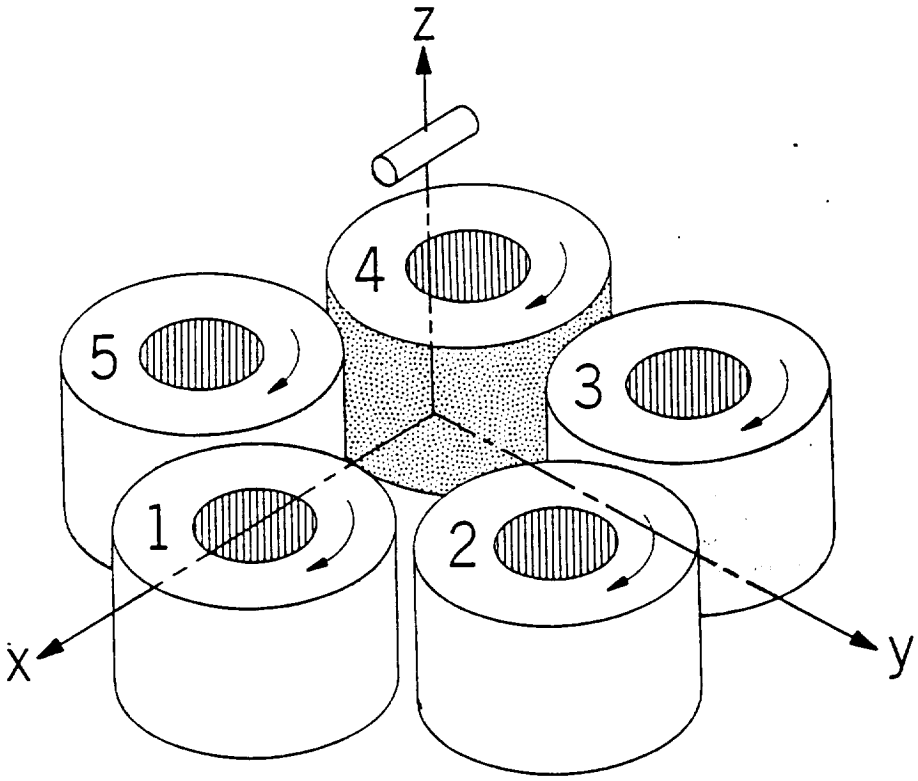


Figure 2.1-The planar array coil configuration

2.3 Position Sensors

The detection of the core's position is performed by five sets of infrared LEDs and sensors. These LED-sensor pairs are installed in two perpendicular planes (vertical and horizontal), which allow detection of five degrees-of-freedom of the core. The beams from the infrared LEDs, which are incident on the sensors,

would be partially blocked by the suspended element. The relative position of the core can then be determined from the amount of light received. This method is common in wind tunnel magnetic suspension applications and has been quite successful.

The LEDs and sensors presently used are intended for fiber-optics. Their advantages are that they are compact and inexpensive. However, the beams from the LEDs have a dispersion of about 20° (included cone angle), which significantly reduces the amount of light received by the sensors. To collimate the beams from the LED's, miniature plano-convex lenses were installed on the LEDs and sensors. The lenses are secured in aluminum tubes which also act as shields to stray light.

The sensors and LEDs are all mounted on an aluminum framework which is above the coil array. This framework has the ability to rotate about the vertical axis a full 360°. The schematic diagram of the sensor arrangement is shown in figure 2.2.

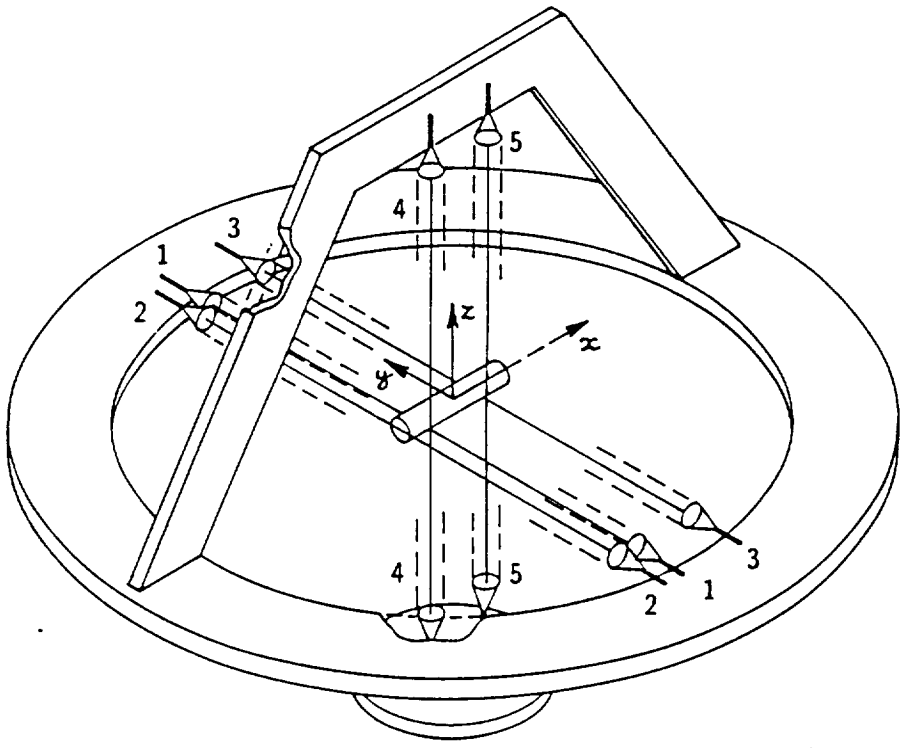


Figure 2.2 A sketch of the sensor frame

2.4 Coils and Power Amplifiers

The currents through the coils are controlled by five switching power amplifiers, capable of delivering a maximum of 30A continuous and 60A peak level. The amplifiers, which are commercially purchased, have a switching frequency of 22 KHz, and require a D.C. supply of 150 V. The amplifiers are equipped with individual enable switches and output current read-outs. The power amplifiers, their power supplies, and their supporting circuitry are housed in an enclosed chassis. These amplifiers function in a voltage-to-current convertor mode. Their response behavior and sensitivity are, to some degree, user adjustable. The amplifiers are set to give a flat response and to have a gain of 3 A/V. The outputs of the amplifiers are directly connected to the coils via cables.

The electromagnetic coils are made of 509 turns of AWG 10 enameled copper wire wound on bakelite spools, with soft iron cores. The windings on the coils are covered with epoxy resin to reduce deformity due to high current forces. The electrical characteristics of the coils has been measured as follows:

Resistance		0.74 Ω
Self Inductance		27.5 mH
Mutual Inductance:		
	(adjacent)	1.6 mH
	(non-adjacent)	0.37 mH

Table 2.1- Coil Specifications (with iron core)

The coils cannot adequately dissipate the thermal energy generated in them due to their internal resistances. A 30 minute run of one coil at 15A, starting

from room temperature, resulted in temperature of about 150°F. To protect against thermal runaway, each coil has been equipped with a temperature sensing device, which is monitored by a set of five digital temperature controllers. These controllers are of the OMEGA CN9000 series. The temperature controllers are connected such that they can activate an alarm and disable the power amplifiers at set temperatures. The temperature controllers are presently set to sound an alarm at 150° F and to disable the amplifiers at 160° F.

CHAPTER 3

SYSTEM MODELING

In order to study the behavior of the system, an accurate model of the plant is necessary. The plant includes the dynamics of the suspended element, the coils, and the power amplifiers. The mathematical model of the plant has been derived in detail in reference 15, however a brief summary will be given here for demonstration.

In general, magnetic systems are nonlinear [16]. After developing a nonlinear model, a linearizing procedure is performed to simplify analysis and implementation. This chapter first shows the basic steps in developing the nonlinear plant model, after which linearization steps are presented. Numerical results are provided later which are then used to study the actual plant's dynamics.

3.1 Nonlinear Plant Equations

A set of coordinate systems needs to be defined in order to model the motions of the core and analyze its dynamics. Figure 3.1 illustrates the coordinate system arrangement defined for this system. An orthogonal right-handed coordinate system is attached to the core and is referred to as the core coordinate system. Its components are denoted as $\bar{x}$, $\bar{y}$, and $\bar{z}$. The origin of the core coordinate system is placed at the centroid of the core, with its $\bar{x}$ axis along the axis of the core. The inertial coordinate system is denoted by x, y, and z; it is fixed with respect to

the coils. The x axis of the inertial system always intersects the axis of coil one. The core coordinates are initially coincident with the inertial coordinates.

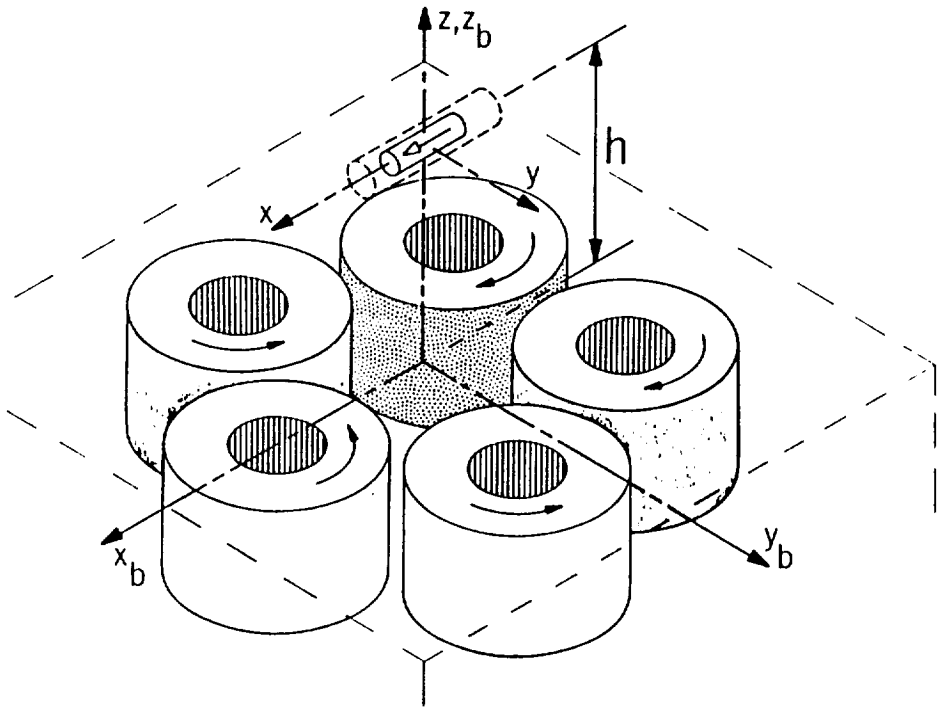


Figure 3.1- The coordinate systems of the LAMSTF

As the core moves from the zero position, the core coordinates deviate from the inertial coordinates. In order to locate the coils with respect to the inertial system, a third coordinate system is attached to the coil array at a distance h below the inertial coordinate system. The components of this system are denoted as x_b, y_b, and z_b. The x_b and y_b axes are parallel to x and y axes of the inertial system, respectively, and the z_b is coincident with the z [15].

Using traditional aircraft dynamics terminology, the motions of the core are defined as follows: Roll is rotation about the $\tilde{x}$ axis. Yaw is rotation about the $\tilde{z}$ axis; Pitch is rotation about the $\tilde{y}$ axis. The linear motions are axial, side, and vertical denoted by x, y, and z. The sign convention follows the right-hand rule definition.

One of the assumptions made in the design of this system is that the model would initially suspend at zero inertial coordinate position. However, the system should later be capable of tracking the core through a 360° rotation about the z axis. Since the core's magnetization is in the direction of its $\bar{x}$ axis, there would be no control of roll motion. The motions in all other degrees-of-freedom, axial, side, vertical, and pitch, are kept to very small amounts.

3.1.1 Equations of Motion

The assumptions used to derive the equations of motion for the core are that the core is a rigid body, the core has negligible product of inertia, and that the core has no rolling motion. Therefore, the inertia matrix for the core would be a 3×3 diagonal matrix with $I_y = I_z = I_c$ [15].

$$[I] = \begin{bmatrix} I_x & 0 & 0 \\ 0 & I_c & 0 \\ 0 & 0 & I_c \end{bmatrix}$$

Using the general angular momentum equation

$$\vec{\bar{H}} = [I]\,\vec{\bar{\Omega}} \tag{3.1}$$

The torque on the suspended element can then be written as follows:

$$\vec{\bar{T}} = \dot{\vec{\bar{H}}} = [I]\,\dot{\vec{\bar{\Omega}}} + \vec{\bar{\Omega}} \times ([I]\,\vec{\bar{\Omega}}) \tag{3.2}$$

By performing some mathematical manipulations the above equation can be rewritten as below:

$$\vec{\bar{T}} = [I]\,\dot{\vec{\bar{\Omega}}} + [\bar{\Omega}]\,[I]\,\vec{\bar{\Omega}} \tag{3.3}$$

The term $[\bar{\Omega}]$ is a skew-symmetric cross product matrix shown in equation 3.4.

$$[\bar{\Omega}] = \begin{bmatrix} 0 & \Omega_{\bar{z}} & \Omega_{\bar{y}} \\ \Omega_{\bar{z}} & 0 & \Omega_{\bar{x}} \\ \Omega_{\bar{y}} & \Omega_{\bar{x}} & 0 \end{bmatrix} \tag{3.4}$$

Solving equation 3.3 for angular acceleration the following equation is obtained:

$$\dot{\vec{\bar{\Omega}}} = [I]^{-1}\,(\,\vec{\bar{T}} - [\bar{\Omega}]\,[I]\,\vec{\bar{\Omega}}\,) \tag{3.5}$$

Since there is no rolling motion, equation 3.5 simplifies to the following:

$$\dot{\vec{\bar{\Omega}}} = \frac{1}{I_c}\,\vec{\bar{T}} \tag{3.6}$$

The force on the core can likewise be determined. In the coordinates of the core the force can be written as below:

$$\vec{\bar{F}} = m_c\,(\,\dot{\vec{\bar{V}}} + [\bar{\Omega}]\,\vec{\bar{V}}\,) \tag{3.7}$$

Solving for the acceleration one can obtain the following:

$$\dot{\vec{\bar{V}}} = \frac{1}{m_c}\,\vec{\bar{F}} - [\bar{\Omega}]\,\vec{\bar{V}} \tag{3.8}$$

3.1.2 Torques and Forces on the Magnetic Core

The suspended element is subject to forces and torques generated from the interaction of the magnetic field from the coils and the field from the permanent magnet core. The governing equations for torque and force are shown in equations 3.9 and equation 3.10 respectively [15]:

$$\vec{T} = \int_{\forall} (\vec{\bar{M}} \times \vec{B}) \, d\forall \tag{3.9}$$

$$\vec{F} = \int_{\forall} (\vec{\bar{M}} \cdot \nabla) \vec{B} \, d\forall \tag{3.10}$$

Where the gradient operator is defined as:

$$\nabla^T = \left[\frac{\partial}{\partial x} \quad \frac{\partial}{\partial y} \quad \frac{\partial}{\partial z} \right] \tag{3.11}$$

The magnetization of the core can be assumed to be uniform over the volume of the core. The size of the core compared to its distance to the coils and the coils themselves is small. It is assumed that the fields in the region of the core are linear functions of position. Therefore, the gradients of the fields are constants over the volume of the core. The integrals in equations 3.9 and 3.10 can now be approximated as the core's volume multiplied by the relevant integrand, evaluated at the centroid of the core. The resulting equations are shown below:

$$\vec{T} \simeq \forall (\vec{\bar{M}} \times \vec{B}_0) \tag{3.12}$$

Similarly the force equation reduces to the following:

$$\vec{F} \simeq \forall (\vec{\bar{M}} \cdot \nabla) \vec{B}_0 \tag{3.13}$$

The dot product in equation 3.13 produces a scalar. Combining this result with the flux density vector $\vec{B}$, using tensor algebra, the above equation can be written as follows:

$$\vec{F} = \forall \, [\, \partial B \,] \, \vec{\bar{M}} \tag{3.14}$$

where

$$[\, \partial B \,] = \begin{bmatrix} B_{xx} & B_{xy} & B_{xz} \\ B_{yx} & B_{yy} & B_{yz} \\ B_{zx} & B_{zy} & B_{zz} \end{bmatrix} \tag{3.15}$$

and where

$$B_{ij} = \frac{\partial B_i}{\partial j}$$

From Maxwell's equation, $\nabla \times \vec{B} = 0$, the above matrix must be symmetrical in the region of the core. Further, applying the identity $\nabla \cdot \vec{B} = 0$, the diagonal terms of the matrix must be zeros.

In the above equations, $\vec{\bar{M}}$ is represented with respect to the core coordinates and $\vec{B}$ is represented with respect to the inertial coordinates. This mismatch in the coordinate systems can be resolved by transforming B to the core coordinate system, using a transformation matrix T_m.

$$\vec{\bar{T}} = \forall \, [\bar{M}] \, [T_m] \, \vec{B} \tag{3.16}$$

$$\vec{\bar{F}} = \forall \, [T_m] \, [\partial B] \, [T_m]^{-1} \, \vec{\bar{M}} \tag{3.17}$$

where

$$T_m = \begin{bmatrix} \cos\theta_z & \sin\theta_z & 0 \\ -\sin\theta_z & \cos\theta_z & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.18}$$

3.1.3 System Equations

The next step in developing the plant model is to combine the equations of motion with the torque and force equations. Substituting the torque equation into the angular acceleration equation, the following result is obtained:

$$\dot{\vec{\Omega}} = \frac{1}{I_c} \forall [\bar{M}] [T_m] \vec{B} \tag{3.19}$$

Similarly, substituting the force equation into the linear acceleration equation results in the following:

$$\dot{\vec{V}} = \frac{1}{m_c} \forall [T_m] [\partial B] [T_m]^{-1} \vec{M} \tag{3.20}$$

Equations 3.19 and 3.20 are the basic equations of motion for the core in core coordinates, neglecting torques and forces due to disturbances. The variable inputs in these equations are the field intensity vector $\vec{B}$ and the field intensity gradient matrix $[\partial B]$. Since the core is held fixed at its operating point with only small displacements, the assumption can be made that the field components and their gradients are only functions of coil currents. The field intensity vector can now be represented by a linear combination of the coil currents [15].

$$\vec{B} = \frac{1}{I_{max}} [K_B] \vec{I} \tag{3.21}$$

Each column in $[K_B]$ corresponds to a coil and contains field components produced by that coil at a maximum current I_{max}. The vector $\vec{I}$ contain the coil current.

The same arrangement can be made for the gradients, by arranging the field gradient's three columns into one column and representing it as vector.

In equation 3.22 $[K_{\partial B}]$ contains field gradients in its columns at maximum current, for each coil. The values in the two matrices $[K_B]$ and $[K_{\partial B}]$ are determined using a magnetic field analysis software which uses the finite element method for the given configuration [17].

$$\partial \vec{B} = \frac{1}{I_{max}} [K_{\partial B}] \vec{I} \tag{3.22}$$

The equations of motion can now be integrated to relate the input currents to the core position. However, the above model is nonlinear due to the coordinate transformations involved. A linearizing technique should be employed to simplify analysis and implementation.

3.2.1 Linearization of the system model

The system can now be linearized and put in a state space format for analysis. The details of the linearization of the plant model can be found in reference 18, however a summary of the steps involved is presented here. One can simplify the equations of motion by using small angle approximation. This is based on the assumption that displacement angles about the $\bar{z}$ and $\bar{y}$ axis are small; this results in $\sin\theta \simeq \theta$ and $\cos\theta \simeq 1$. With these approximations the torque and force equations can be simplified to the following equations.

$$T_{\bar{y}} = \forall\ M_{\bar{x}}\ (\ -\theta_y B_x - B_z) \tag{3.23}$$

$$T_{\bar{z}} = \forall\ M_{\bar{x}}\ (\ -\theta_z B_x - B_y) \tag{3.24}$$

$$F_{\bar{x}} = \forall\ M_{\bar{x}}\ (\ B_{xx} + 2\,\theta_z B_{xy} - 2\,\theta_y B_{xz}\ ) + m_c\,\theta_y\,g \tag{3.25}$$

$$F_{\overline{y}} = \forall\ M_{\overline{x}}\ (\ -\theta_z\, B_{xx} + B_{xy} + \theta_z\, B_{yy} - \theta_y B_{yz}) \tag{3.26}$$

$$F_{\overline{z}} = \forall\ M_{\overline{x}}\ (\ \theta_y\, B_{xx} + B_{xz} + \theta_z\, B_{yz} - \theta_y\, B_{zz}\ ) \tag{3.27}$$

The above torque and force equations can be expressed in terms of angular and linear accelerations. As developed earlier, the fields and field gradients can be represented as functions of coil currents only. This will result in a set of nonlinear differential equations. These equations represent a system with currents as input and core position rates as output. A set of relevant variables $\hat{X}$ can be defined, and $\dot{\hat{X}}$ can be expressed as a function of $\hat{X}$ and I, shown in equations 3.28 and 3.29, respectively.

$$\hat{X}^T = \lfloor\ \Omega_y\ \Omega_z\ \theta_y\ \theta_z\ V_{\overline{x}}\ \ V_{\overline{y}}\ \ V_{\overline{z}}\ \ x\ y\ z\ \rfloor \tag{3.28}$$

$$\dot{\hat{X}} = f\ (\ \ddot{\theta}_y, \ddot{\theta}_z, \ddot{x}, \ddot{y}, \ddot{z}) = f\left(\begin{bmatrix} \vec{T} \\ \vec{F} \end{bmatrix}\right) \tag{3.29}$$

The method used here to linearize the system is the Taylor series expansion about a fixed point. Using only the first order terms of the series and subtracting the initial operating point [18] one obtains the following:

$$\delta\dot{\hat{X}} = A\ \delta\hat{X} + B\ \delta I \tag{3.30}$$

where

$$A = \mathcal{W}_2 \frac{\partial\ f(\vec{T}, \vec{F}\ )}{\partial \hat{X}}\ \Bigg|_{\hat{X}_o, I_o} \tag{3.31}$$

$$B = \mathcal{W}_2 \frac{\partial\ f(\vec{T}, \vec{F})}{\partial I}\ \Bigg|_{\hat{X}_o, I_o} \tag{3.32}$$

$\mathcal{W}_2$ is a diagonal matrix with moment of inertia I_c and core mass m_c arranged in its diagonal.

$$\mathcal{W}_2 = \begin{bmatrix} \frac{\forall M_x}{I_c} & 0 & \cdots & & & & & \cdots & 0 \\ 0 & \frac{\forall M_x}{I_c} & \ddots & & & & & & \vdots \\ \vdots & \ddots & 1 & & & & & & \\ & & & 1 & & & & & \\ & & & & \frac{\forall M_x}{m_c} & & & & \\ & & & & & \frac{\forall M_x}{m_c} & & & \\ & & & & & & \frac{\forall M_x}{m_c} & & \\ & & & & & & & 1 \quad \ddots \quad \vdots & \\ & \vdots & & & & & & \ddots \quad 1 \quad 0 & \\ & 0 & \cdots & & & & & \cdots \quad 0 \quad 1 & \end{bmatrix} \tag{3.33}$$

Matrix A can be expanded and simplified to the following:

$$A = \mathcal{W}_2 \begin{bmatrix} 0\ 0 & -B_x & 0 & 0\ 0\ 0 & -B_{xz} & -B_{yz} & -B_{zz} \\ 0\ 0 & 0 & -B_x & 0\ 0\ 0 & -B_{yz} & B_{yy} & B_{yz} \\ 1\ 0 & 0 & 0 & 0\ 0\ 0 & 0 & 0 & 0 \\ 0\ 1 & 0 & 0 & 0\ 0\ 0 & 0 & 0 & 0 \\ 0\ 0 & \left(\frac{m_c g}{\forall M_{\bar{x}}} - 2B_{xx}\right) & 2B_{xy} & 0\ 0\ 0 & B_{(xx)x} & B_{(xx)y} & B_{(xx)z} \\ 0\ 0 & B_{yz} & (B_{yy}-B_{xx}) & 0\ 0\ 0 & B_{(xy)x} & B_{(xy)y} & B_{(xy)z} \\ 0\ 0 & (B_{xx} - B_{zz}) & B_{yz} & 0\ 0\ 0 & B_{(xz)x} & B_{(xz)y} & B_{(xz)z} \\ 0\ 0 & 0 & 0 & 1\ 0\ 0 & 0 & 0 & 0 \\ 0\ 0 & 0 & 0 & 0\ 1\ 0 & 0 & 0 & 0 \\ 0\ 0 & 0 & 0 & 0\ 0\ 1 & 0 & 0 & 0 \end{bmatrix} \tag{3.34}$$

Following some manipulations as in [18], assuming that the coil currents are not affected by the core movements and neglecting small terms, the B matrix can simplified to the following:

$$B = \frac{1}{I_{max}} \mathcal{W}_2 \begin{bmatrix} -K_{z_1} & -K_{z_2} & -K_{z_3} & -K_{z_4} & -K_{z_5} \\ K_{y_1} & K_{y_2} & K_{y_3} & K_{y_4} & K_{y_5} \\ \cdots & 0 & \cdots & & \\ \cdots & 0 & \cdots & & \\ K_{xx_1} & K_{xx_2} & K_{xx_3} & K_{xx_4} & K_{xx_5} \\ K_{xy_1} & K_{xy_2} & K_{xy_3} & K_{xy_4} & K_{xy_5} \\ K_{xz_1} & K_{xz_2} & K_{xz_3} & K_{xz_4} & K_{xz_5} \\ \cdots & 0 & \cdots & & \\ \cdots & 0 & \cdots & & \\ \cdots & 0 & \cdots & & \end{bmatrix} \tag{3.35}$$

The system model is now linearized. Using the state variables defined in equation 3.28 and the above A and B matrices, the state-space representation of the system can now be analyzed.

3.2.2 Initial Condition

In the absence of gravitation and disturbances the core would suspend at a defined zero position, in this case at a height of 10 cm above the top of the coils over the center of the array, with no current through the coils. However, in order to compensate for the weight of the model, a set of steady currents should be supplied to the coils. This results in a constant force $F_{\bar{z}}$ shown below, with all other forces and torques equal to zero.

$$F_{\bar{z}} = m_c\, g \tag{3.36}$$

The above conditions will results in the following:

$$B_y = B_z = B_{xx} = B_{xy} = 0 \tag{3.37}$$

$$B_{xz} = \frac{m_c\, g}{\forall M_{\bar{x}}} \tag{3.38}$$

From equation 3.35, the initial current can be calculated using the following:

$$I_0 = I_{max} \begin{bmatrix} [K_y] \\ [K_z] \\ [K_{xx}] \\ [K_{xy}] \\ [K_{xz}] \end{bmatrix}^{-1} \begin{bmatrix} B_y \\ B_z \\ B_{xx} \\ B_{xy} \\ B_{xz} \end{bmatrix} \tag{3.39}$$

Note that the the only non-zero term in the B matrix is B_{xz}. Since the K matrix is square with non-zero determinant, and is therefore invertible, I_0 can readily be determined.

3.3 Numerical Analysis

The numerical values for the magnetic field intensity and its gradients at the point of suspension can be calculated in two steps. The Biot-Savart law may be used to determine the fields and field gradients due to the currents in the coils with no iron core present [19]. A finite difference method can then be applied to the result to account for the effect of the iron cores, through an iterative process. To solve the equations involved in this method would be very tedious. A magnetic field analysis software, based on finite element and finite difference methods, has been used to determine the field intensity and its gradients at the point of interest [6]. Assuming a constant current of 10A through the coils, the

following table, containing field intensities, their first and second gradients, has been determined.

Coil	B_x, T	B_y, T	B_z, T	B_{xx}, T/m	B_{xy}, T/m	B_{xz}, T/m	B_{yy}, T/m	B_{yz}, T/m	B_{zz}, T/m
1	0.00231	0	-0.00094	0.02179	0	-0.02723	-0.0189	0	-0.00287
2	0.00071	0.00220	⋮	-0.01503	0.01196	-0.02203	0.01790	-0.02590	⋮
3	-0.00187	0.00136		0.00772	-0.01936	0.02203	-0.00485	-0.01600	
4	-0.00187	-0.00136		0.00772	0.01936	0.02203	-0.00485	0.01600	
5	0.00071	-0.00220		-0.01503	-0.01936	-0.00841	-0.0179	0.02590	

Table 3.1- Electromagnet fields and first-order gradients

Coil	$B_{(xx)x}$, T/m/m	$B_{(xy)x}$, T/m/m	$B_{(xz)x}$, T/m/m	$B_{(xy)y}$, T/m/m	$B_{(xy)z}$, T/m/m	$B_{(xz)z}$, T/m/m
1	0.03434	0	-0.53466	-0.21916	0	0.00254
2	-0.18276	-0.14560	0.16371	-0.11938	0.08735	0.00078
3	0.16559	0.13733	-0.26790	0.22896	-0.14134	-0.00205
4	0.16559	-0.13733	-0.26790	0.22896	0.14134	-0.00205
5	-0.18276	0.14560	0.16371	-0.11938	-0.08735	0.00078

Table 3.2 - Electromagnet second-order gradients

The matrices $\mathcal{A}$, $\mathcal{B}$ and $\mathcal{W}_2$ are shown below after numerical substitution.

$$\mathcal{A} = \begin{bmatrix}
0 & 0 & 7.846\text{e-}3 & 0 & 0 & 0 & 0 & -9.25\text{e-}2 & 0 & 0 \\
0 & 0 & 0 & 7.846\text{e-}3 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -9.25\text{e-}2 & 0 & 0 & 0 & 0 & 4.710\text{e-}1 & 0 & -2.366\text{e-}4 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 9.015\text{e-}1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -2.366\text{e-}4 & 0 & -8.614\text{e-}3 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0
\end{bmatrix} \tag{3.40}$$

$$\mathcal{B} = \begin{bmatrix}
4.0710\text{e}1 & 4.0710\text{e}1 & 4.0710\text{e}1 & 4.0710\text{e}1 & 4.0710\text{e}1 \\
0 & 9.5278\text{e}1 & 5.8899\text{e}1 & -5.8899\text{e}1 & -9.5278\text{e}1 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
2.3101\text{e-}1 & -1.5935\text{e-}1 & 8.1846\text{e-}2 & 8.1846\text{e-}2 & -1.593\text{e-}1 \\
0 & 1.268\text{e-}1 & -2.0525\text{e-}1 & 2.0525\text{e-}1 & -1.268\text{e-}1 \\
-2.8869\text{e-}1 & -8.9161\text{e-}2 & 2.3356\text{e-}1 & 2.3356\text{e-}1 & -8.9161\text{e-}2 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0
\end{bmatrix} \tag{3.41}$$

where $$A = \mathcal{W}_2 \, \mathcal{A}$$

and $$B = \frac{1}{I_{max}} \mathcal{W}_2 \, \mathcal{B}$$

Using the specifications of the core, $\mathcal{W}_2$ is as follows:

$$\mathcal{W}_2 = \begin{bmatrix} 4.3308e5 & 0 & \cdots & & & & & \cdots & 0 \\ 0 & 4.3308e5 & \ddots & & & & & & \vdots \\ \vdots & \ddots & 1 & & & & & & \\ & & & 1 & & & & & \\ & & & & 1.0602e2 & & & & \\ & & & & & 1.0602e2 & & & \\ & & & & & & 1.0602e2 & & \\ & & & & & & & 1 & \ddots & \vdots \\ \vdots & & & & & & & \ddots & 1 & 0 \\ 0 & \cdots & & & & & & \cdots & 0 & 1 \end{bmatrix} \tag{3.42}$$

The matrix A, which contains the dynamics of the open loop plant, has the following eigenvalues.

$$\mathfrak{E} = \begin{bmatrix} 59.2596 + 0j \\ -59.2596 + 0j \\ 0 + 7.9717j \\ 0 - 7.9717j \\ 0 + 0.9556j \\ 0 - 0.9556j \\ 58.2942 + 0j \\ -58.2942 + 0j \\ 9.7762 + 0j \\ -9.7762 + 0j \end{bmatrix} \tag{3.43}$$

There are five pairs of eigenvalues, each of which can be associated with one or two degrees of freedom. As can be seen, three of these pairs are unstable, and the other two are marginally stable. The modes of the system are listed in table 3.3.

Mode #	Eigenvalues	Stability	Degree of Freedom
1	$\pm$ 59.26 rad/s	Unstable	x, θ_y (Axial , Pitch)
2	$\pm$ 7.972 rad/s	Stable Oscillatory	x, θ_y (Axial, Pitch)
3	$\pm$ 58.29 rad/s	Unstable	θ_z (Yaw rotation)
4	$\pm$ 0.956 rad/s	Stable Oscillatory	z (Vertical motion)
5	$\pm$ 9.776 rad/s	Unstable	y (Lateral motion)

Table 3.3 Open loop modes of the suspended element

The two most unstable eigenvalue pairs, modes one and three, are referred to as the "compass needle" modes; this represents the cores tendency to turn and align its ends with the unlike polarities. A sketch of the modes, in relation to the core's motions is shown in figure 3.2.

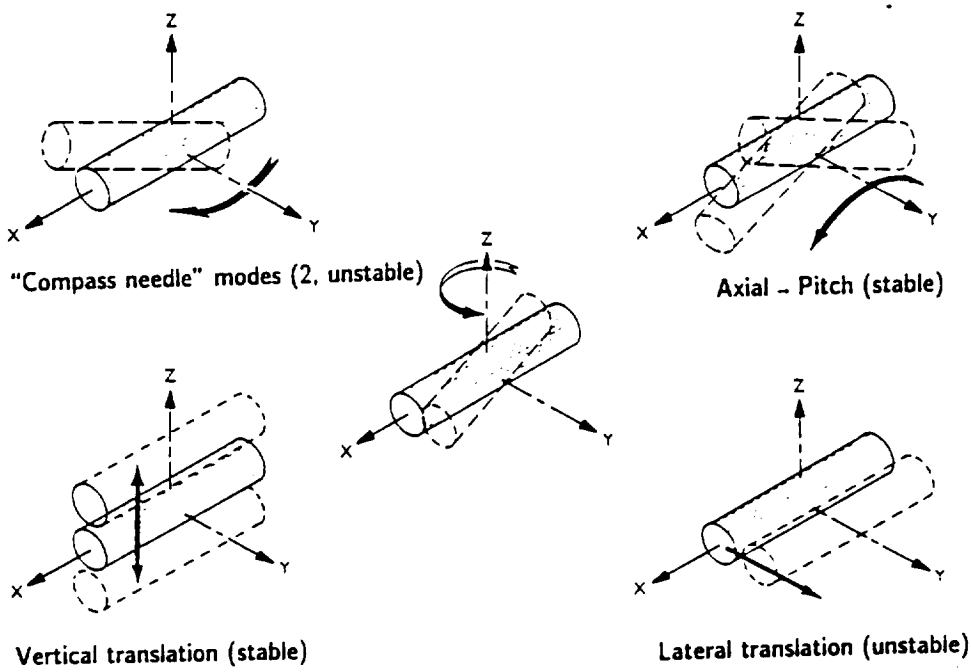


Figure 3.2- The modes of the model

3.4 Coils and Power Amplifiers Modeling

Up this point it was assumed that the coils were capable of producing the magnetic field instantaneously. The coils are inductors, and inductors produce lags in the system. The power amplifiers are constant current sources based on their command input voltages; they are voltage to current converters. However, they cannot produce instantaneous infinite power to compensate for the lag in the coils. For an accurate model of the system, it is required to model the the coil-amplifier block.

The basic circuit representation of an amplifier and a coil is shown in figure 3.3.

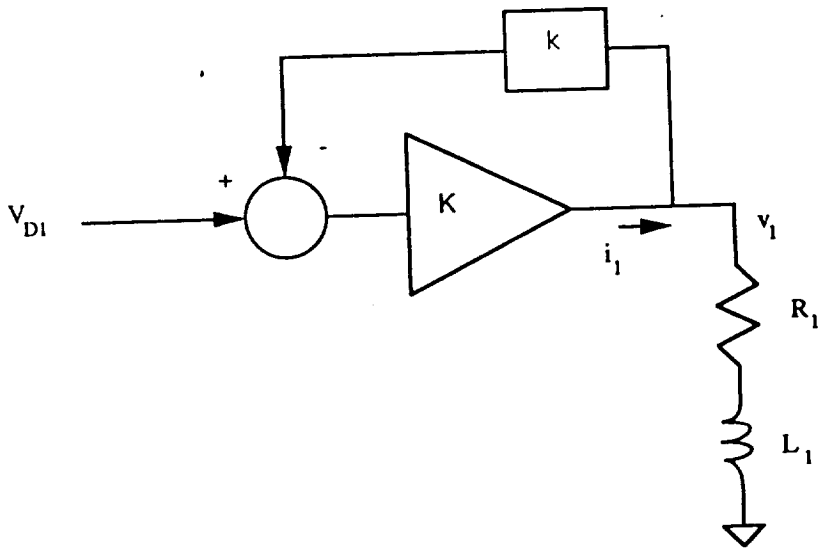


Figure 3.3- Simplified amplifier and coil circuit diagram

Simple circuit analysis will show that the coil terminal voltage can be represented by the following equation:

$$V_1 = I_1 (R_1 + L_1 s) + I_2 L_{12} s + I_3 L_{13} s + I_4 L_{14} s + I_5 L_{15} s \qquad (3.44)$$

The amplifiers are represented as having an output current feedback. Applying the Kirchhoff's voltage law, V_1 can also be written as follows:

$$V_1 = K\,(V_{D1} - k\,I) \tag{3.45}$$

Equating equations 3.44 and 3.45, the relation between the current demand input to the amplifier to the coil current for coil one can be represented as:

$$\begin{bmatrix} L_1 & L_{12} & L_{13} & L_{14} & L_{15} \end{bmatrix} \begin{bmatrix} \dot{I}_1 \\ I_2 \\ I_3 \\ I_4 \\ I_5 \end{bmatrix} = -\,(R_1 + Kk)\,I_1 + K\,V_{D1} \tag{3.46}$$

Applying the same procedure to all five coils and amplifiers, the overall coil-amplifier state-space model can be written as follows:

$$\dot{\vec{I}} = \widetilde{A}\,\vec{I} + \widetilde{B}\,\vec{V}_D \tag{3.47}$$

Assuming the coils have identical resistances, the $\widetilde{A}$ and $\widetilde{B}$ matrices are defined as follows:

$$\widetilde{A} = -(R + Kk)\,[\,L\,]^{-1} \tag{3.48}$$

$$\widetilde{B} = K\,[\,L\,]^{-1} \tag{3.48}$$

Since $L_{ij} = L_{ji}$, [L] is symmetric. Every coil has two adjacent and two non-adjacent neighboring coils. Due to the geometry of the array, the mutual inductances between all the adjacent coils are equal. Therefore, the mutual inductances between all the non-adjacent coils are also equal.

$$[L] = \begin{bmatrix} L_1 & L_{12} & L_{13} & L_{14} & L_{15} \\ L_{12} & L_2 & L_{23} & L_{24} & L_{25} \\ L_{13} & L_{23} & L_3 & L_{34} & L_{35} \\ L_{14} & L_{24} & L_{34} & L_4 & L_{45} \\ L_{15} & L_{25} & L_{35} & L_{45} & L_5 \end{bmatrix} \tag{3.49}$$

Numerical values for the coil parameters are given in table 2.1. To determine the values for K and k, the D.C. gain and the break frequency of the amplifier/coil combination is necessary. The D.C. gain is set to 3 A/V, and, from the frequency response measurements, the break frequency is estimated to be about 180Hz. This results in K=93.30 and k=0.325.

Substituting numerical values into equation 3.46, the state-space model becomes:

$$\begin{bmatrix} \dot{I}_1 \\ \dot{I}_2 \\ \dot{I}_3 \\ \dot{I}_4 \\ \dot{I}_5 \end{bmatrix} = \begin{bmatrix} -1138.2 & 62.34 & 10.10 & 10.10 & 62.34 \\ 62.34 & -1138.2 & 62.34 & 10.10 & 10.10 \\ 10.10 & 62.34 & -1138.2 & 62.34 & 10.10 \\ 10.10 & 10.10 & 62.34 & -1138.2 & 62.34 \\ 62.34 & 10.10 & 10.10 & 62.34 & -1137.7 \end{bmatrix} \begin{bmatrix} I_1 \\ I_2 \\ I_3 \\ I_4 \\ I_5 \end{bmatrix}$$

$$+ \begin{bmatrix} 3414.7 & -187.0 & -30.3 & -30.3 & -187.0 \\ -187.0 & 3414.7 & -187.0 & -30.3 & -30.3 \\ -30.3 & -187.0 & 3414.7 & -187.0 & -30.3 \\ -30.3 & -30.3 & -187.0 & 3414.7 & -187.0 \\ -187.0 & -30.3 & -30.3 & -187.0 & 3414.7 \end{bmatrix} \begin{bmatrix} V_{D1} \\ V_{D2} \\ V_{D3} \\ V_{D4} \\ V_{D5} \end{bmatrix} \tag{3.50}$$

The analysis of the system without some form of CAD software would be difficult, therefore the software MATLAB^{TM} will be utilized extensively. The state-space model for the coils and power supplies can be added to the system state space using "SERIES" command in MATLAB.

CHAPTER 4

Compensator Design

It was established in the previous chapter that the plant model is open-loop unstable. Some feedback control technique must be employed in order to achieve stable suspension. Several control schemes are available to stabilize the system. Aside from the classical techniques, there are modern state feedback schemes such as Linear Quadratic Gaussian and Linear Quadratic Regulator. However, in order to prove the planar array concept, a simple and well-tested method, classical dual phase-advance compensator (DPA), was chosen first [20]. The dual phase-advance compensator technique has been successfully used in many magnetic suspension applications and is relatively simple to implement [21][22]. Once the system is operational, other control schemes can be tried for their performance on this system. In this phase of the research no great effort has been made to maximize performance. The basic aim has been to make the system operational, and after implementation in hardware, to compare the actual system's behavior to the mathematical model. This chapter demonstrates some basic approach used to design the compensators, and shows some step responses of the closed loop system.

4.1 Decoupled System Behavior

The decoupled motions of the suspended element can easily be deduced from the sensor signals. This provides the required signals for five degrees-of-freedom, which are x, y, z, θ_y, and θ_Z (refer to figure 3.1). These signals can then be

processed through a set of five independent compensators and fed back to position command input comparators. The errors from these comparators are then used to drive the current amplifiers. A block diagram is illustrated in figure 4.1.

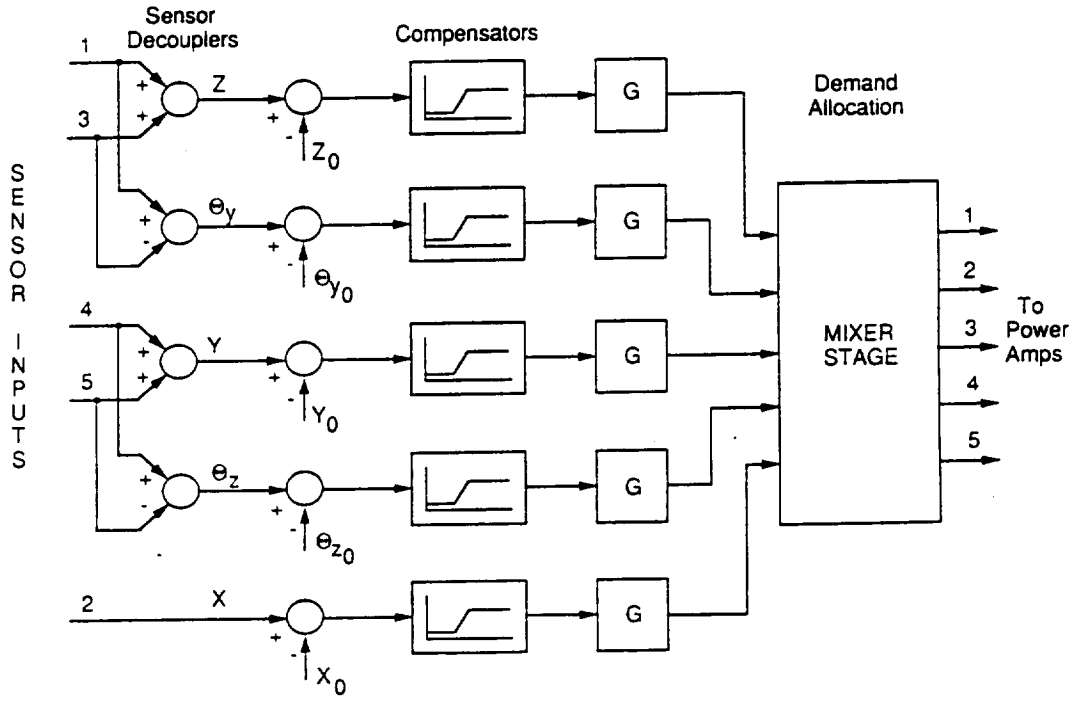


Figure 4.1 Controller block diagram

Most magnetic suspension systems follow the same pattern of feedback and control, however the "mixer" stage in figure 4.1 is fairly unusual. The coil currents are not decoupled with respect to the degrees-of-freedom; each coil affects more than one degree-of-freedom. The decoupled position signals must be converted to coil current demand signals to produce the desired magnetic fields. This is done by linear combination of the coil currents through a 5×5 mixing matrix, also known as current allocation matrix, located just prior to the current amplifier inputs. In chapter six, it will be shown that, by switching through a series of these mixing matrices, the core would be made capable of 360° rotation in yaw.

The application of the DPA compensator, like many other controller design approaches, relies, to some extent, on previous knowledge of the application and may involve some trial-and-error [23]. The DPA compensator is basically two lead stages in cascade, and it is essentially a high-pass filter. In general, systems using permanent magnets or electromagnets without current control are inherently unstable [1] and have lag characteristics. They require an active control with some method of lead compensation to increase their crossover frequency, and hence the overall system's bandwidth [23].

The plant model is a multi-input multi-output system, and in some cases it has coupled degrees-of-freedom. This makes analysis difficult using classical methods. However, a study of only one degree-of-freedom can give some useful information regarding the application of the dual phase-advance method to the full system.

As described in the previous chapter, the pitch and yaw modes are the most unstable of the five modes, and are referred to as the "compass-needle modes". Out of these two compass-needle modes, pitch is coupled with the axial motion, but yaw is fairly independent of other degrees-of-freedom. For simplicity yaw will be used as a demonstration of the application of DPA compensator to this system.

Equation 3.24 describes the dynamics of yaw motion. This equation can be written as following:

$$\ddot{\theta}_z = \frac{\forall M_x}{I_c} (- \theta_z B_x + B_y) \tag{4.1}$$

Since it was assumed that the fields can be expressed in terms of linear functions of the coil currents, equation 4.1 can be written as:

$$\ddot{\theta}_z = \frac{\forall M_x}{I_c I_{max}} (- \lfloor K_x \rfloor \vec{I} \theta_z + \lfloor K_y \rfloor \vec{I}) \tag{4.2}$$

The above equation can be linearized about the operating point, using a Taylor series expansion. Subtracting the linearization point, this results in the following equation:

$$\ddot{\theta}_z \simeq \frac{\forall M_x}{I_c I_{max}} \left(\lfloor K_x \rfloor \vec{I}_0 \theta_z + \theta_{z_0} \lfloor K_x \rfloor \vec{I} + \lfloor K_y \rfloor \vec{I} \right) \tag{4.3}$$

The assumption was that the core is at zero position, therefore $\theta_{z_0} = 0$. The row vector $\lfloor K_x \rfloor$ can be combined with the current vectors $\vec{I}_0/I_{max}$ to form the scalar I_0. In order to represent the five input currents to the system as a single command current, the row vector $\lfloor K_y \rfloor$ is multiplied by the second column of the mixer matrix shown in equation 4.18. The resulting transfer function of the above equation can be represented as shown below:

$$G_z = \frac{\theta_z}{I_{yaw}} = \frac{0.06}{\frac{I_c}{\forall M_x} s^2 - I_0} \tag{4.4}$$

Substituting numerical values into the above transfer function will produce the following characteristic equation and transfer function:

$$s^2 - 3.3982 \times 10^3 = 0$$

$$G_{zp} = \frac{2.3688 \times 10^4}{(s + 58.294)(s - 58.294)} \tag{4.5}$$

In order to properly represent the single degree-of-freedom, it is necessary to include the lag due to the power amplifier and the coil responses. Through experimental measurement, the break frequency of the amplifier-coil combination has been found to be about 1130.9 rads/sec. The power amplifiers have a D.C. gain of 3 A/V. Neglecting the mutual inductances, the overall open-loop transfer

function can be represented as in equation 4.6. From the root locus of G_{zp} in figure 4.2 , it can be seen that the closed-loop system is unstable for all gains. Therefore, a compensator is needed for stability.

$$G_{zp} = \frac{8.03662e7}{(s + 58.294)\,(s - 58.294)\,(s + 1130.9)} \tag{4.6}$$

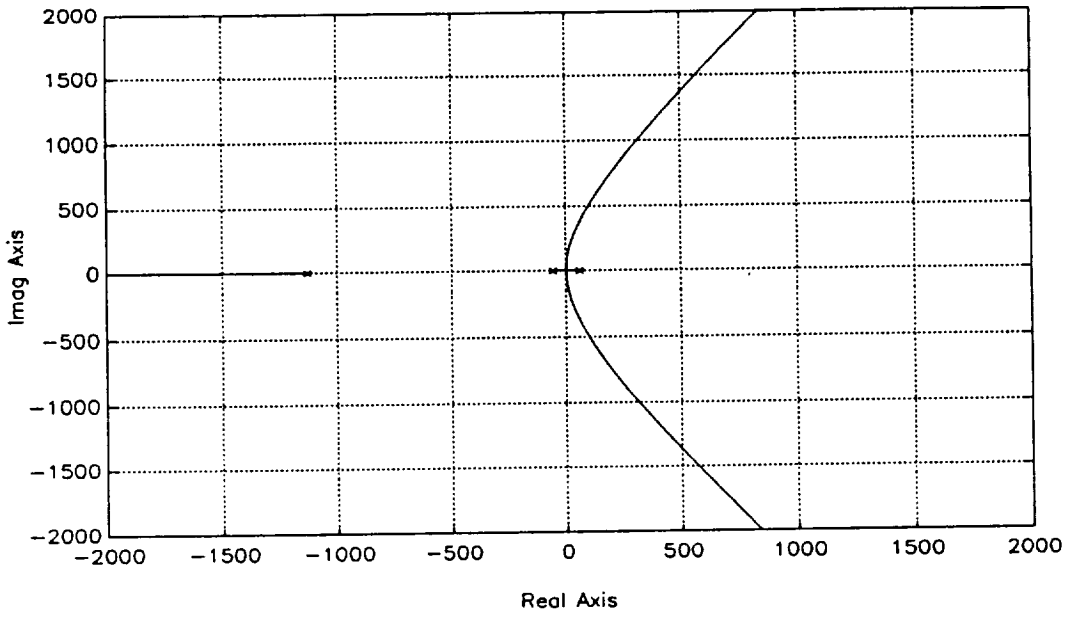


Figure 4.2- Root locus of yaw degree-of-freedom (no compensator).

4.2 Dual Phase Advance Compensator

The aim of the compensator is to place all the closed-loop poles of the system in the left half-plane. As it will be shown, the DPA will accomplish this task satisfactorily. The general transfer function of a lead compensator is:

$$H = \alpha \frac{1 + j\omega T}{1 + j\omega\alpha T} \qquad \alpha<1 \tag{4.7}$$

The above function places a zero at the point 1/T and a pole at $1/(\alpha T)$. It is a common practice to let $\alpha \simeq 0.1$, which puts the pole of the compensator far

enough to the left to minimize its effects on the system dynamics while at the same time it is close enough to limit high frequency noise. The only factor that remains to determine is the break frequency 1/T. One method of selecting T, for a type 0 system, is to place the zero of the compensator over the second highest pole of the system [23]. This location, from the eigenvalues, is at a frequency between 9 to 10Hz (58 to 59 rads). The bode plot of the lead compensator described above and the root locus of the system with the compensator are shown in figures 4.3 and 4.4, respectively.

In order to form a DPA compensator a second lead compensator is connected in series to the first. The equation for the dual phase advance compensator is basically the square of equation 4.7. Substituting $1/\alpha = 10$, the transfer function for the dual phase-advance is the following:

$$H_{DPA} = g \frac{(j\omega + 1/T)^2}{(j\omega + 10\ /T)^2} \tag{4.8}$$

The root locus of the system with the DPA compensator is shown in figure 4.5. Compared to the single phase advance, the DPA compensator may not seem much advantageous in the present case. This can be attributed to the current feedback in the amplifiers which increases the break frequency of the coils to 1130.9 rads/sec. If the break frequency of the coil alone was used (R/L = 26.9 rads/sec), the advantages of the DPA compensator would become more obvious. The DPA compensator is chosen here because of its popular application in magnetic suspension systems.

4.3 Application of The DPA to All Five Degrees-of-Freedom

The stabilizing technique given for the single degree-of-freedom system can be applied to all five degrees-of-freedom. However, to ease the process of analysis,

space-state method will be employed. The block diagram of this process is depicted in figure 4.6.

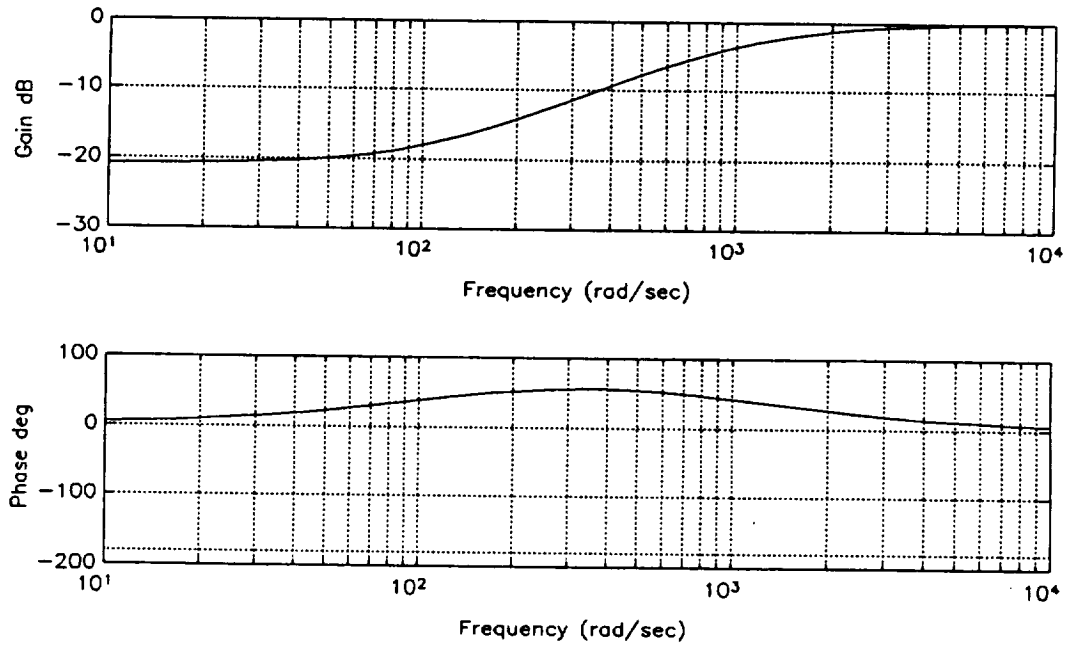


Figure 4.3- Bode plot of the lead compensator.

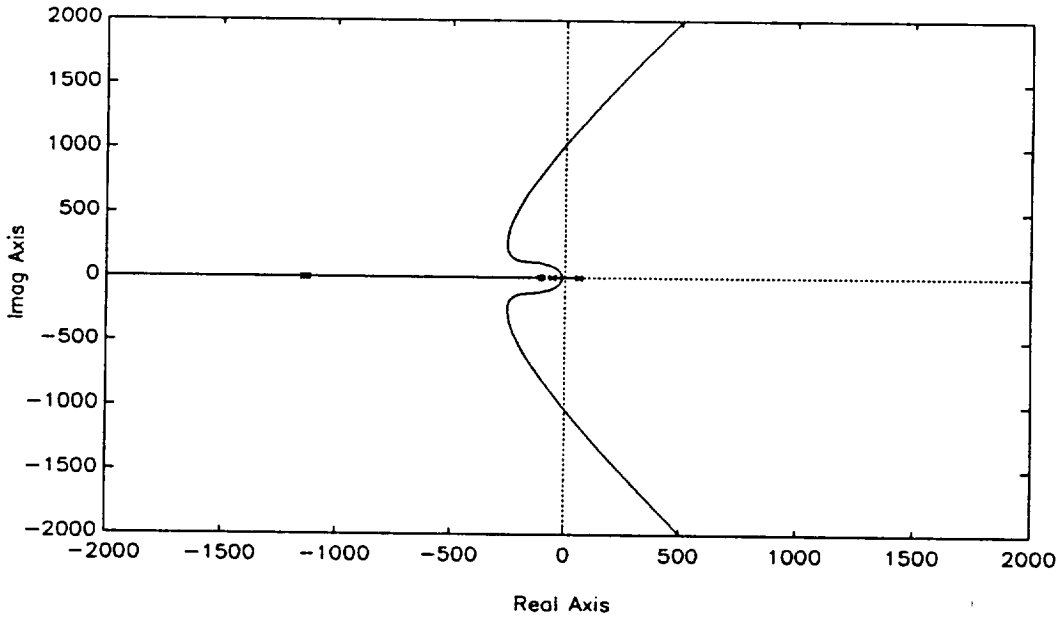


Figure 4.4- Root locus of yaw with one lead compensator.

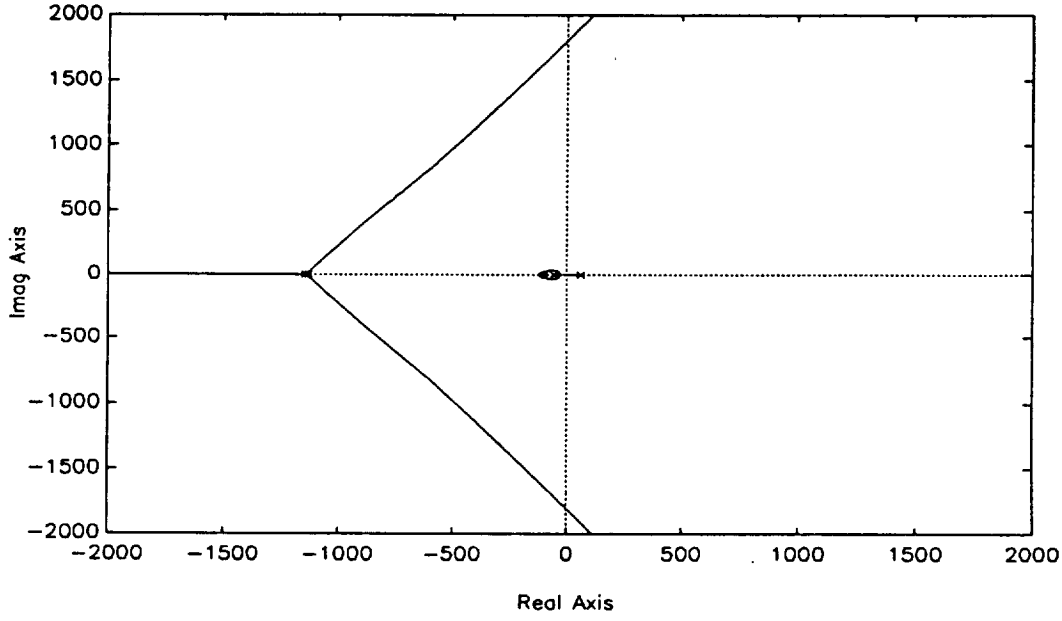


Figure 4.5- Root locus of yaw with DPA compensator.

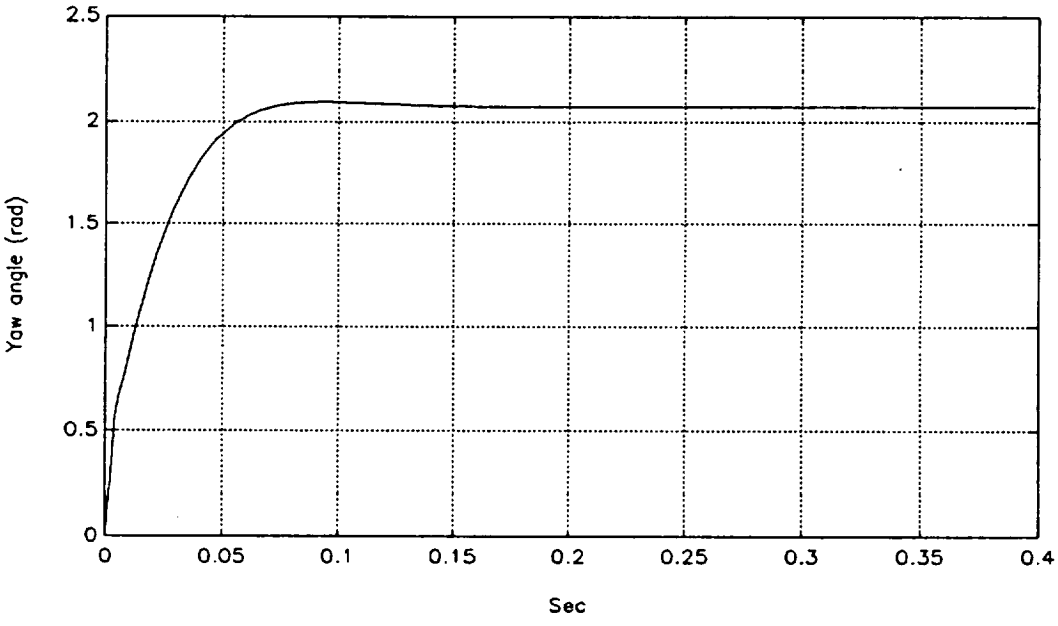


Figure 4.6- Step response in yaw for the selected gain

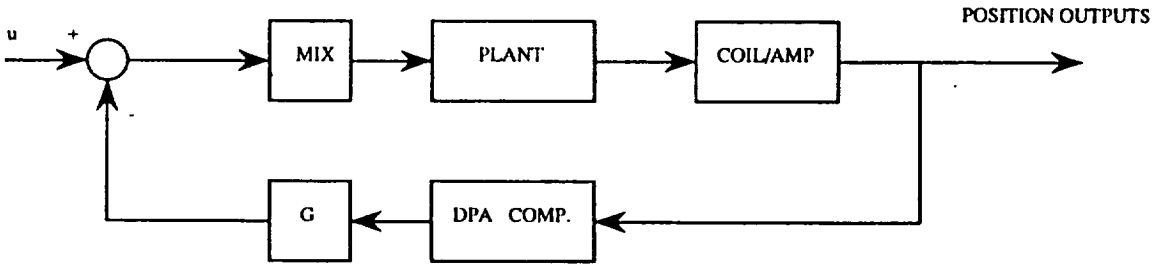


Figure 4.7- Plant and controller block diagram.

The state-space models describing the plant and power amplifiers and coils have already been determined in previous chapter. The necessary step now is to determine the state-space model for five independent compensators to adequately stabilize the overall system.

In order to select the compensator break frequencies for other degrees of freedom, the modes of the system should be examined. Again referring to table 3.3, the dynamics of the pitch degree-of-freedom (open loop poles of ± 59.26 rads/s) is very similar to that of yaw. It seems reasonable to use the same compensator values for pitch. From the three remaining modes, axial and vertical motions are marginally stable, and the unstable pole due to the lateral motion is very close to the imaginary axis. This allows for some flexibility in selecting the compensator parameters, since they are very close to being stable. An initial tendency would be to use similar compensator settings for all five degrees-of-freedom for simplicity, and this will be shown to be workable.

From the transfer function, one can construct the state-space model for a single compensator as follows:

$$\mathcal{A}c = \begin{bmatrix} -\frac{2}{T} & -\frac{1}{T^2} \\ 1 & 0 \end{bmatrix} \quad \mathcal{B}c = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad \mathcal{C}c = \begin{bmatrix} \frac{2n-2n^2}{T} & \frac{1-n^2}{T^2} \end{bmatrix} \quad \mathcal{D}c = [\, n^2 \,] \quad (4.9)$$

Substituting the values selected for the yaw degree-of-freedom into the above state-space equations results in the following:

$$\mathcal{A}c = \begin{bmatrix} -1.5385e3 & -5.9172e5 \\ 1 & 0 \end{bmatrix} \quad \mathcal{B}c = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad \mathcal{C}c = \begin{bmatrix} -1.7906e5 & -7.4964e7 \end{bmatrix} \quad \mathcal{D}c = [\, 127.69]$$

(4.10)

By arranging five of the above models in parallel, arranged in a single state-space system, the basic compensator structure is complete and can be attached to the plant model. This can be easily accomplished by using the "APPEND" command in MATLAB™. The resulting compensator state-space model is as follows:

$$\mathcal{A}\mathcal{A}c = \begin{bmatrix}
-1.53e3 & -5.92e5 & 0 & & & & & & & & \\
1 & 0 & 0 & 0 & & & & & & & \\
 & & 0 & -1.53e3 & -5.92e5 & 0 & & & & & \\
 & & & 1 & 0 & 0 & 0 & & & & \\
 & & & & 0 & -1.53e3 & -5.92e5 & 0 & & & \\
 & & & & & 1 & 0 & 0 & 0 & & \\
 & & & & & & 0 & -1.53e3 & -5.92e5 & 0 & \\
 & & \mathbf{0} & & & & & 1 & 0 & 0 & 0 \\
 & & & & & & & & 0 & -1.53e3 & -5.92e5 \\
 & & & & & & & & & 1 & 0
\end{bmatrix}$$

(4.11)

$$\mathcal{BB}c = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ & & \cdots 0 \cdots & & \\ 0 & 1 & 0 & 0 & 0 \\ & & \cdots 0 \cdots & & \\ 0 & 0 & 1 & 0 & 0 \\ & & \cdots 0 \cdots & & \\ 0 & 0 & 0 & 1 & 0 \\ & & \cdots 0 \cdots & & \\ 0 & 0 & 0 & 0 & 1 \\ & & \cdots 0 \cdots & & \end{bmatrix} \tag{4.11}$$

$$\mathcal{CC}c = \begin{bmatrix} \text{1.79e5} & \text{-7.49e7} & & & & & & & \\ & & \text{-1.79e5} & \text{-7.49e7} & & & \cdots 0 \cdots & & \\ & & & & \text{-1.79e5} & \text{-7.49e7} & & & \\ & & \cdots 0 \cdots & & & & \text{-1.79e5} & \text{-7.49e7} & \\ & & & & & & & & \text{-1.79e5} \ \text{-7.49e7} \end{bmatrix} \tag{4.12}$$

$$\mathcal{DD}c^{\mathrm{T}} = [\ 127.69 \quad 127.69 \quad 127.69 \quad 127.69 \quad 127.69] \tag{4.13}$$

The compensator's state-space model is connected in the feedback path of the closed loop system. Again, this is accomplished by using the command "FEEDBACK" in MATLAB™. The feedback gain is a variable which can be selected based on the desired response. The gains on all degrees-of-freedom can be lumped in a diagonal "gain matrix" and added to the compensator. In simulation these gains are combined with the sensor gains. The sensor gains are determined

through some experimental measurements. These gains vary somewhat depending on the optical alignments and the adjustment of the electronics. Typical measured sensors gains in V/m are shown below:

$$g_{sensor} = [\ 5077 \quad 5495 \quad 6024 \quad 6965 \quad 6560] \tag{4.14}$$

These gains are converted to gains for each degree of freedom using some approximations. The coupled sensor signals (sensors 1 and 3 and sensors 4 and 5) can be averaged and assumed to have the same gains. These averages can directly be used for linear motions (vertical and side motions). In the case of pitch and yaw, the distances between the two coupled sensors (sensors one and three for pitch and four and five for yaw) are used to derive angular displacements. Using simple geometry the following equation can be deduced:

$$\text{angular displacement} = \frac{V_1\, g_1 - V_3\, g_3}{d} = \left(\frac{g_{ave}}{d}\right) \Delta V \tag{4.15}$$

The value in parentheses is the angular gain for either pitch or yaw. A typical set of decoupled sensor gains is shown below:

$$g_{ol} = [\ 162.8\text{V/rad} \quad 183.9\text{V/rad} \quad 4882.1\text{V/m} \quad 5269.7\text{V/m} \quad 5137.8\text{V/m}] \tag{4.16}$$

These gains in conjunction with the compensator's gains are arranged in the diagonal of the 5×5 gain matrix. The gain matrix is added to the model by simple multiplication to the output of the compensator state space equation.

The forces and torques necessary to control the core are generated by a combination of currents in the coils. These currents are determined through a mixing matrix which transforms the position errors to currents necessary to

correct them. A general mixing matrix can be derived, based on 1 unit of field and field gradient in the position of the core. This produces a 5×5 matrix shown below:

$$\begin{bmatrix} I_{1D} \\ I_{2D} \\ I_{3D} \\ I_{4D} \\ I_{5D} \end{bmatrix} = \begin{bmatrix} 1.8282e3 & 0 & 1.9652e2 & 0 & -1.4687e2 \\ 2.3703e3 & 1.6446e3 & -1.5901e2 & 1.1553e2 & -4.5418e1 \\ 2.0347e3 & 1.0160e3 & 6.0751e1 & -1.8689e2 & 1.1885e2 \\ 2.0347e3 & -1.0160e3 & 6.0751e1 & 1.8689e2 & 1.1885e2 \\ 2.3703e3 & -1.6446e3 & -1.5901e2 & -1.1553e2 & -4.5418e1 \end{bmatrix} \begin{bmatrix} -B_z \\ B_y \\ B_{xx} \\ B_{xy} \\ B_{xz} \end{bmatrix} \quad (4.17)$$

This matrix is normalized column-by-column for simplicity. The resulting matrix is as follows:

$$\begin{bmatrix} 0.7713 & 0 & 1 & 0 & -1 \\ 1 & 1 & -0.809 & 0.618 & -0.309 \\ 0.768 & 0.618 & 0.309 & -1 & 0.809 \\ 0.768 & -0.618 & 0.309 & 1 & 0.809 \\ 1 & -1 & -0.809 & -0.618 & -0.309 \end{bmatrix} \quad (4.18)$$

4.4 Closed-Loop System Dynamics

The closed-loop poles determine the stability of the system for a given set of gains. Since a root locus plot of a multi-input multi-output system is not easy to interpret, a study of gain variation of one degree-of-freedom at a time can be helpful. Through variation of gains for one degree-of-freedom at a time, a reasonable set of gains for all degrees-of-freedom was determined. These gains are shown below, and the resulting closed-loop poles are depicted in figure 4.8.

$$\text{gains} = [\ 0.1 \quad 0.1 \quad 0.5 \quad 0.5 \quad 0.5\ ] \quad (4.19)$$

The stable region of each degree-of-freedom was determined by varying the gains, one degree-of-freedom at a time, on either side of the operating point gains. These are listed in the table below:

Degree of Freedom	Low Gain Limit	High Gain Limit
Pitch	0.01	.15
Yaw	0.08	.2
Axial	0.11	.50
Side	0.21	0.48
Vertical	0.12	.51

Table 4.1 Stability limits of the LAMSTF using the DPA

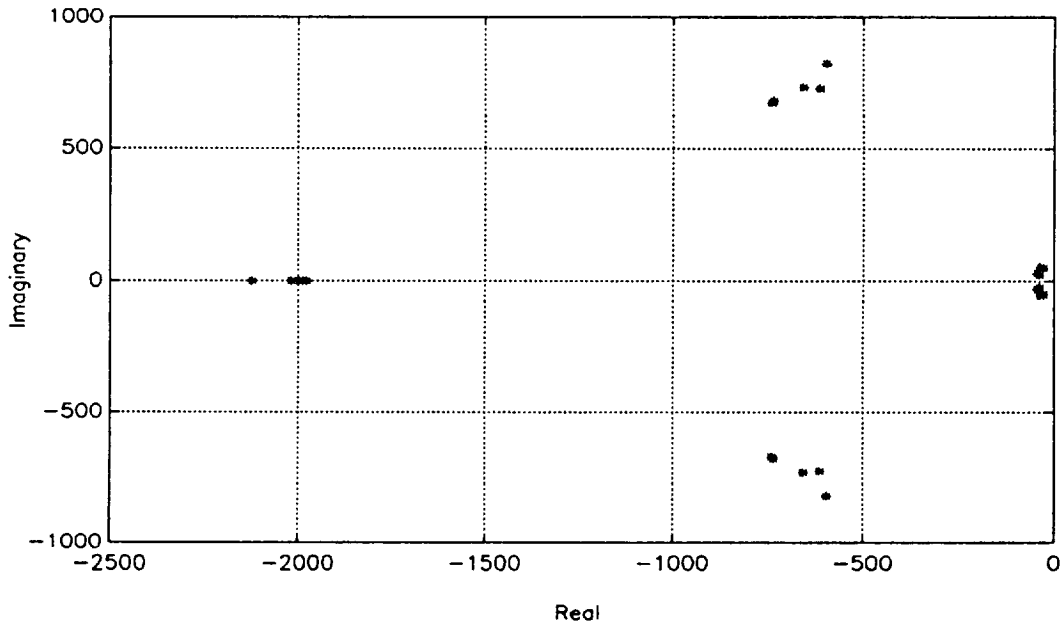
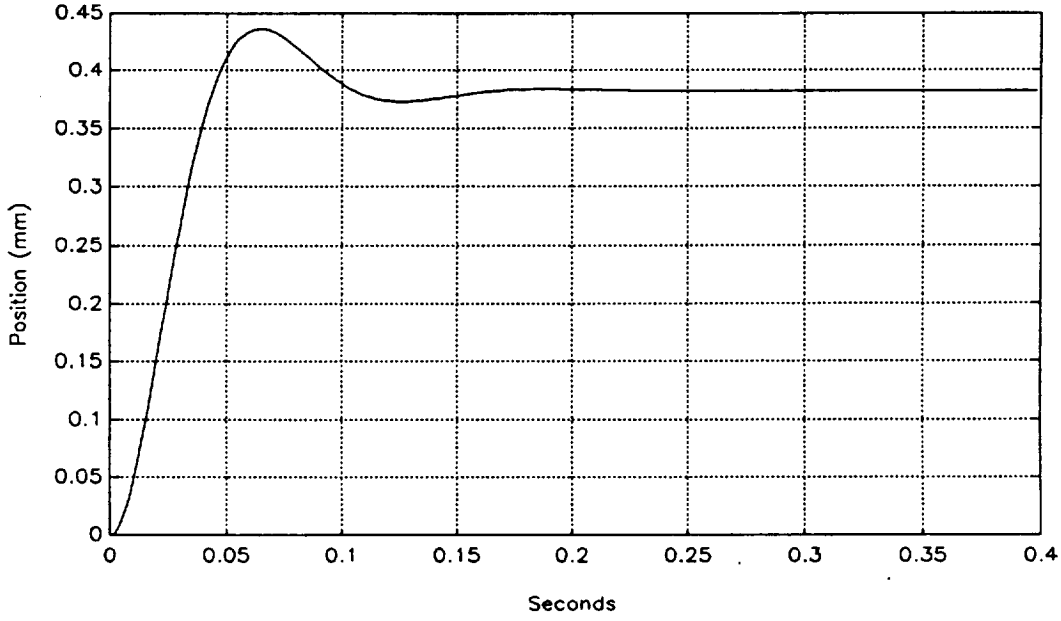
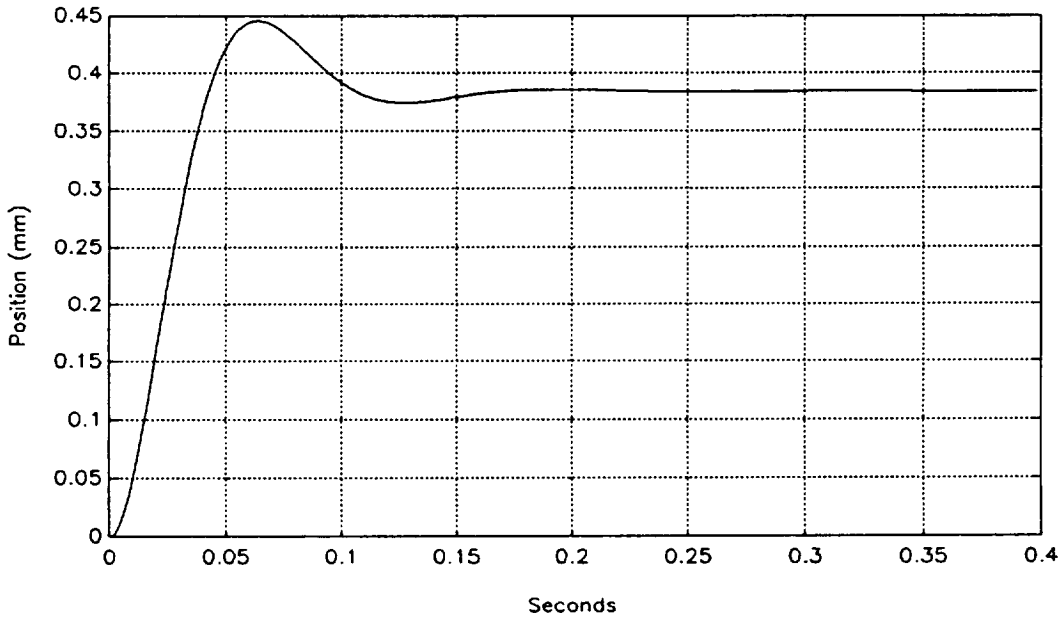


Figure 4.8- Closed-loop poles of the plant.

The resulting step responses for every controlled degree-of-freedom are shown in figure 4.9. As it can be seen, they all have a reasonable damping and rise time.

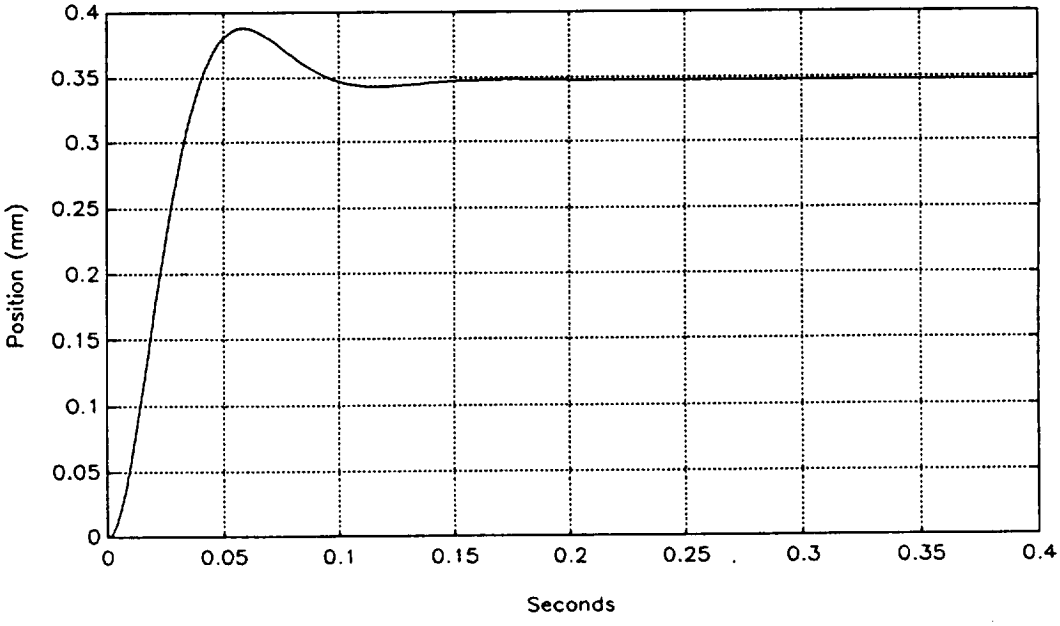


(a) - Step response in x (axial).

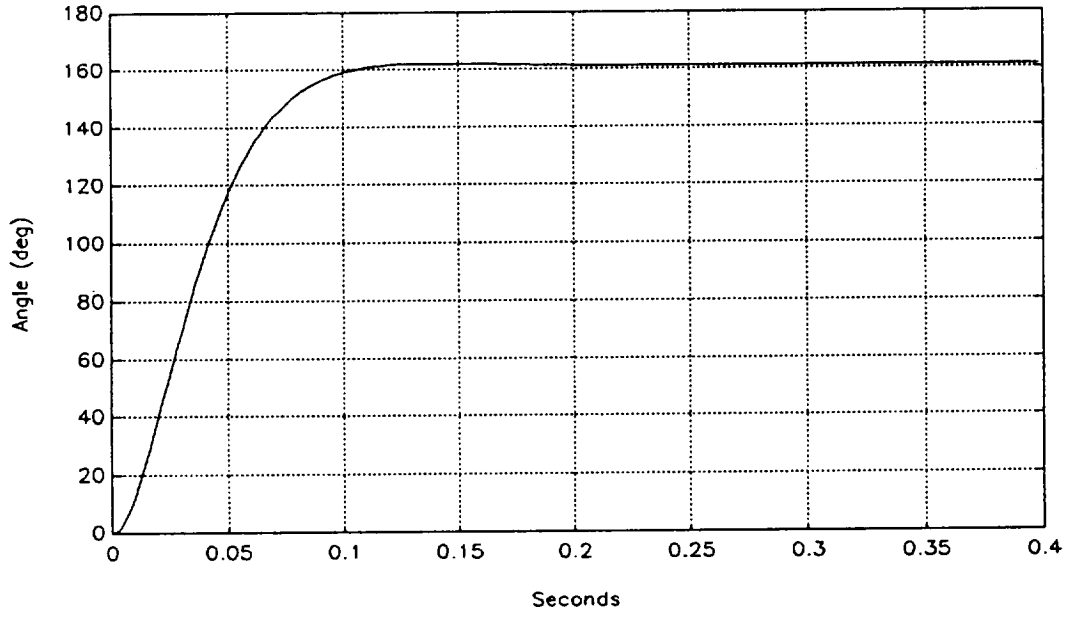


(b) - Step response in y (side).

Figure 4.9- Simulation step responses in all five degrees-of-freedom.

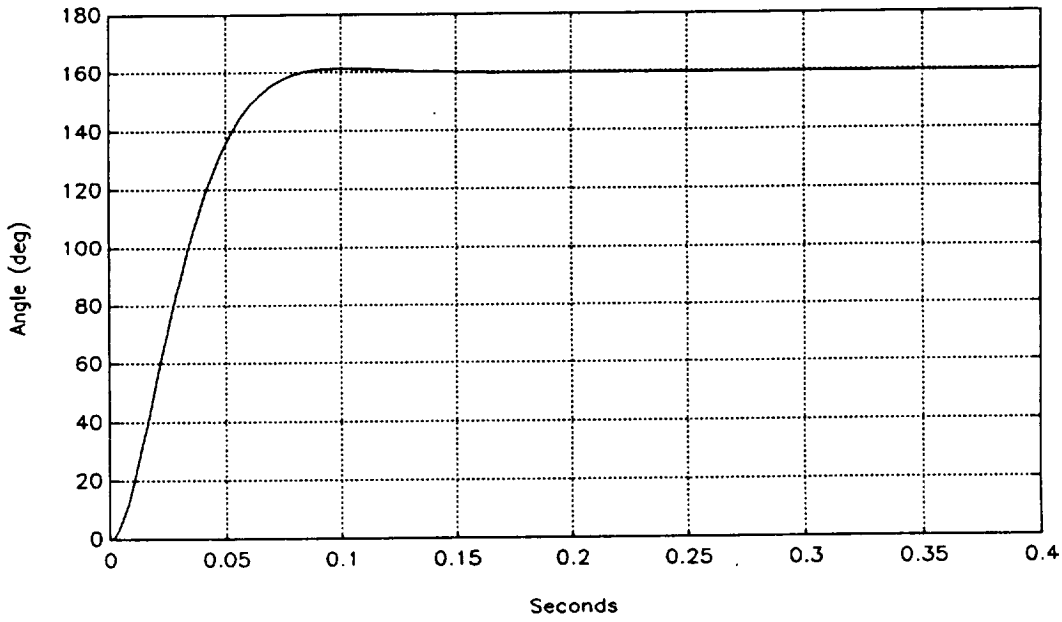


(c) - Step response in z (vertical).



(d) Step response in θ_y (pitch)

Figure 4.9- Continued.



(e) - Step response in θ_z (yaw).

Figure 4.9- Continued.

CHAPTER 5

Analog Controller

The major aim of this phase of the research has been to validate the concept of the planar array magnetic suspension system. Provisions had to be made to implement the previous chapter's controller design into hardware. Even though a digital controller would be more appropriate for this application, due to the initial inaccessibility to a suitable digital computer, an analog version of the controller was constructed. The analog controller would be a bridge between the theory and the digital controller. The analog controller consists of three major stages, sensor decouplers, DPA compensators, and current allocation mixer. This chapter will describe the basic principles of the analog controller designed for this system.

5.1 Sensor Decoupler

As explained previously, four out of five degrees-of-freedom detected by the sensors are coupled. By simple summing and differencing appropriate sensor signals, the decoupling can be achieved. Differential amplifiers and summers are the two basic analog circuits used to perform this task. The generic circuit diagrams of these two stages are shown in figure 5.1.

The sensor output voltages should range from 0.5V at full beam to 5.0V at no beam. However, these voltages deviate from ideal due to the lenses and sensor frame mis-alignments and circuit component tolerances. The sensor signals are conditioned through a set of amplifiers to produce outputs ranging form -0.5V at

no beam to 10.5V at full beam. These voltages are the inputs to the analog controller.

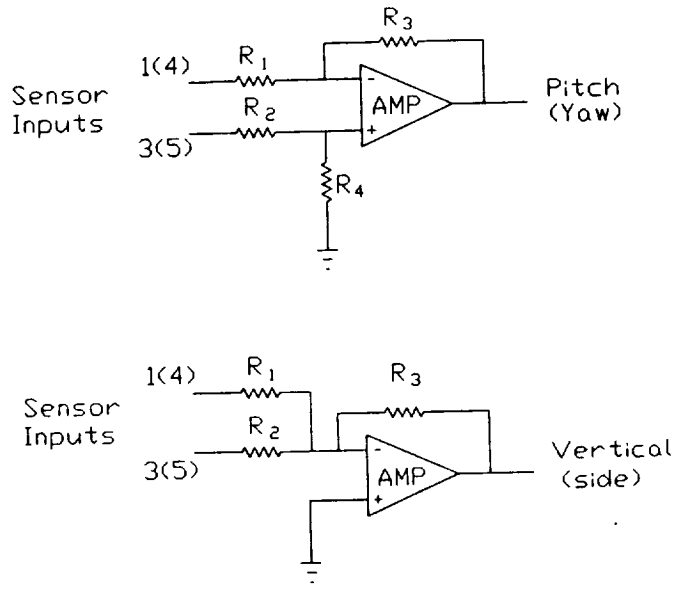


Fig 5.1 Differential amplifier and summer.

The summing of sensors one and three would produce the vertical motion signal, and summing sensors four and five produces the side motion signal. In order to produce a 0V to 10V signal for every degree of freedom, a gain of 0.5 is selected for the summers to average the summed inputs.

The differencing of the same combination of the above sensors can give the angular motion of the core. Differencing sensors one and three produces the pitch signal, and differencing four and five results in the yaw signal. The gain in these stages are kept to unity. The sensitivity of each sensor channel is determined through some calibration procedure.

The signal from sensor two represents the axial movement, and is not coupled with other degrees of freedom. Therefore, the axial signal passes through the decoupler stage unaffected. Vertical and side movement signals pass through inverters to correct for signal inversion in the adders.

5.2 Dual Phase Advance Compensator

In direct correspondence with the simulation, the next stage would be to compensate the decoupled signals. The schematic diagram of a typical analog dual phase advance is illustrated in figure 5.2.

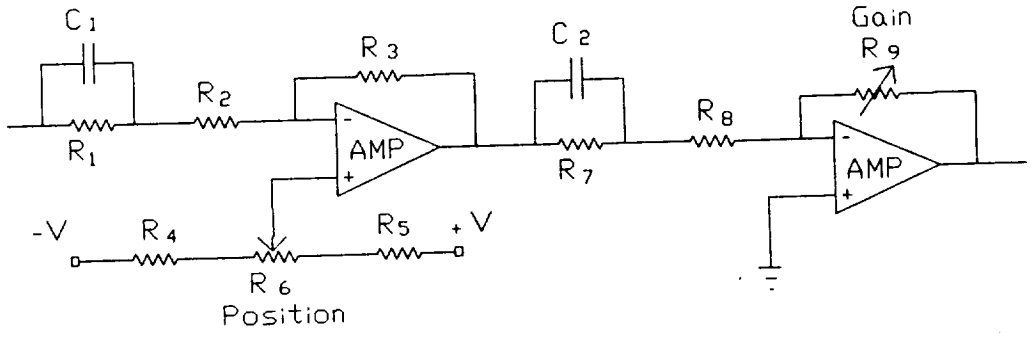


Figure 5.2- A typical analog DPA schematic diagram.

The corresponding equation of the phase advance, using resistor-capacitor network in the first stage, is as follows:

$$H = \frac{1 + R_1 C_1 s}{1 + R_1 C_1 n s} \tag{5.1}$$

The value of n can be shown to be:

$$n = \frac{R_2}{R_1 + R_2} \tag{5.2}$$

The break frequencies of the compensator can be set by selecting suitable values for R_1 and C_1. Since the two stages are identical, the same principals applies to components of the second stage C_2, R_7, and R_8. Taking into consideration the

electronic aspects of the circuit, the following component values have been selected to place a zero at about 16Hz: $R_1 = R_7 = 100$ kΩ, $R_2 = R_8 = 10$ kΩ, and $C_1 = C_2 = 0.1$ μf. With these components, the value of n=0.09. In the circuit the feedback resistor R_3 and R_9 mainly determine the circuit gain, and R_9 is made variable for this reason. In addition to the gain control, a D.C. offset to the first op-amp act as the position command input.

5.3 Mixer Stage

The mixer stage is composed of a set of five summers, which combine the five compensated signals to form the coil current demand signals. A simplified circuit diagram forming the current demand signals is shown in figure 5.3.

The calculated values of the resistors are mostly nonstandard. Therefore, variable resistors, adjusted for the desired values, were employed. Since the inputs to this stage are a combination of positive and negative polarity of the same signal, analog inverters were used to supply all the signals required. The outputs of the op-amps of this stage are directly fed to the power amplifiers.

5.4 System Response

The suspension of the model was successfully achieved. Figures 5.4 and 5.5 are actual photos of the model during suspension. The performance of the analog controller was quite satisfactory. Step responses of this controller were recorded. These step responses along with the mathematical simulation results are shown in figures 5.6 through 5.10. For all degrees-of-freedom except pitch, the actual step responses closely match the simulation results. The pitch signal seems to be less damped that the simulated result.

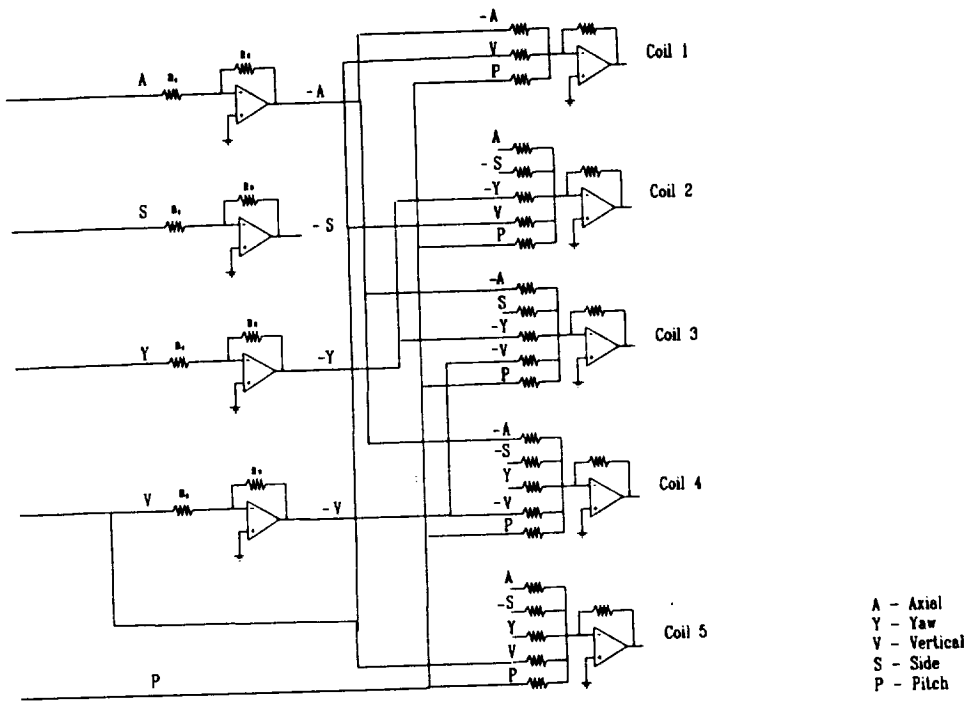


Figure 5.3 Current allocation (Mixer) network schematic.

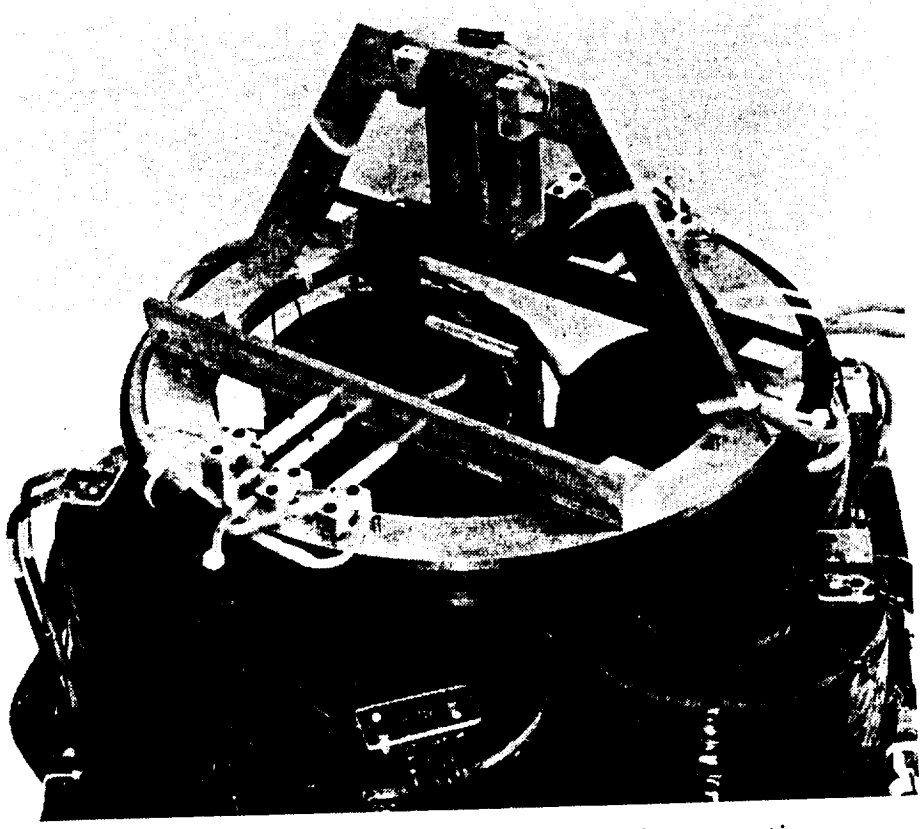


Figure 5.4- Photo of the LAMSTF during operation.

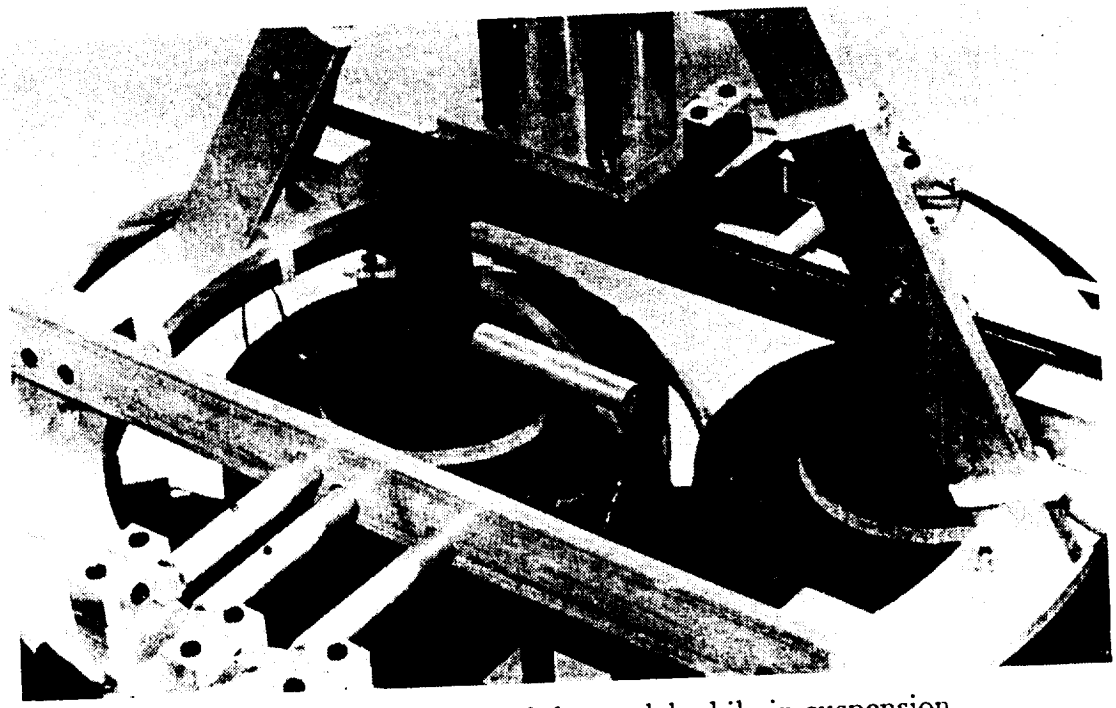
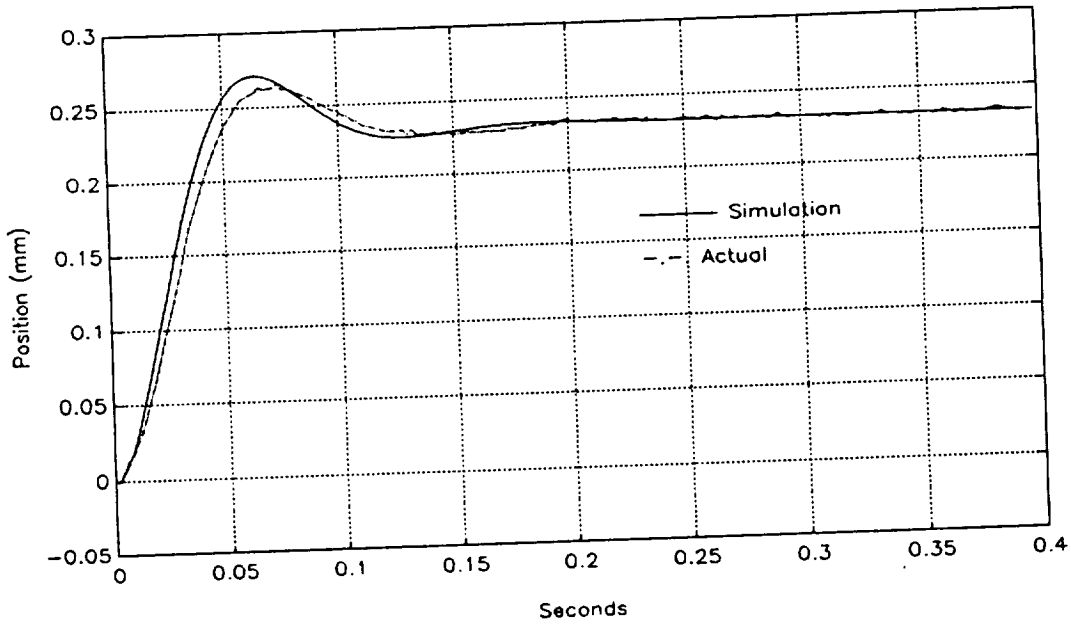
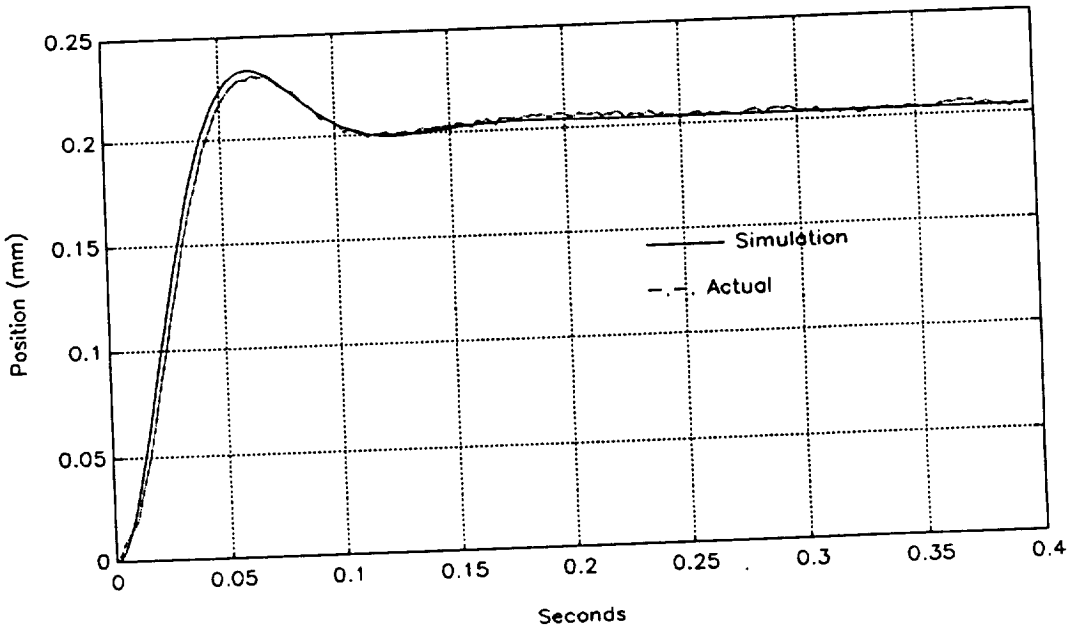


Figure 5.5- Closeup photo of the model while in suspension.

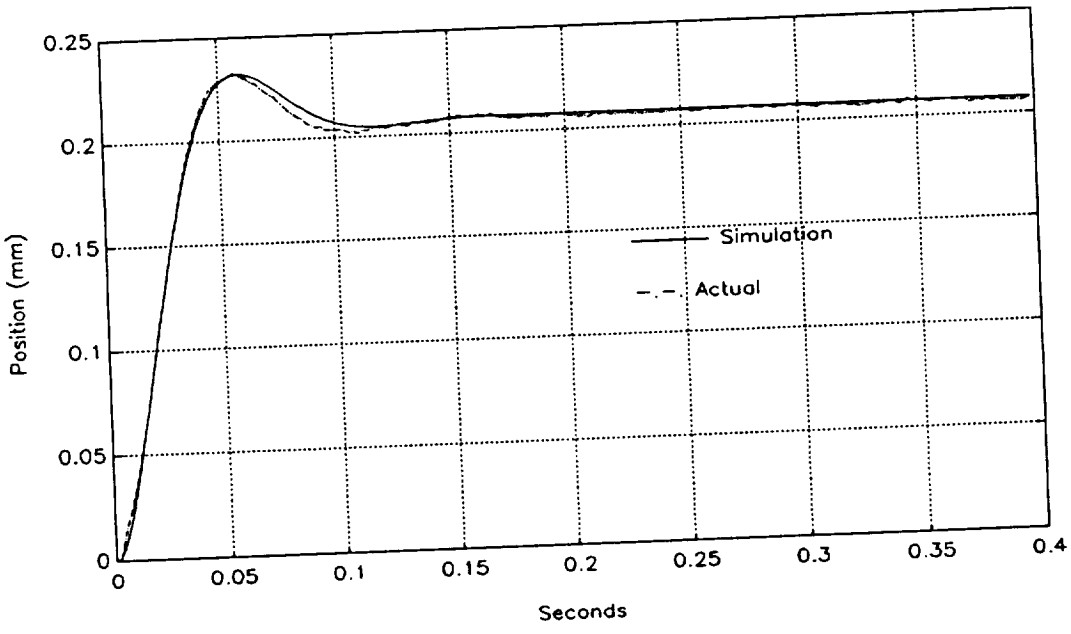


(a) - Step response in x (axial).

Figure 5.6 - Analog controller step responses.

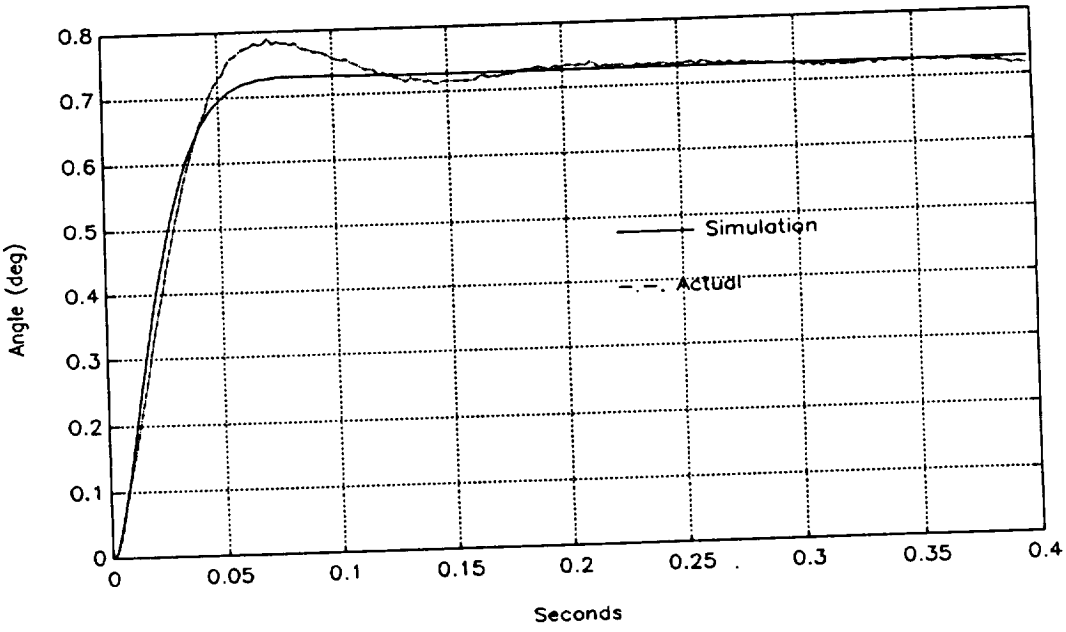


(b) Step response in y (side).

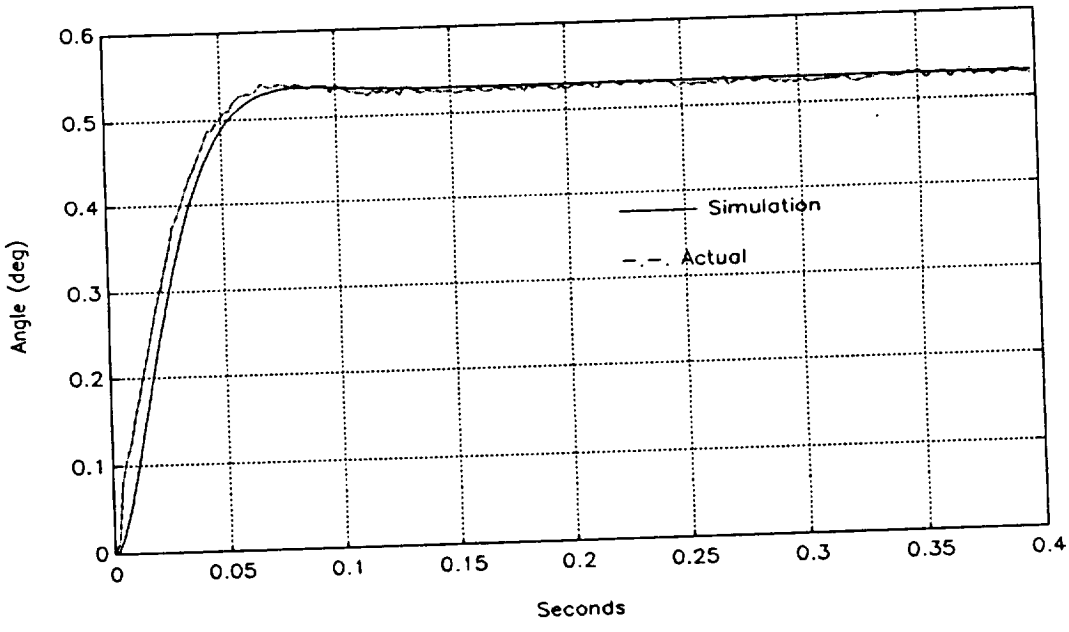


(c) - Step response in z (vertical).

Figure 5.6- Continued.



(d) - Step response in θ_y (pitch).

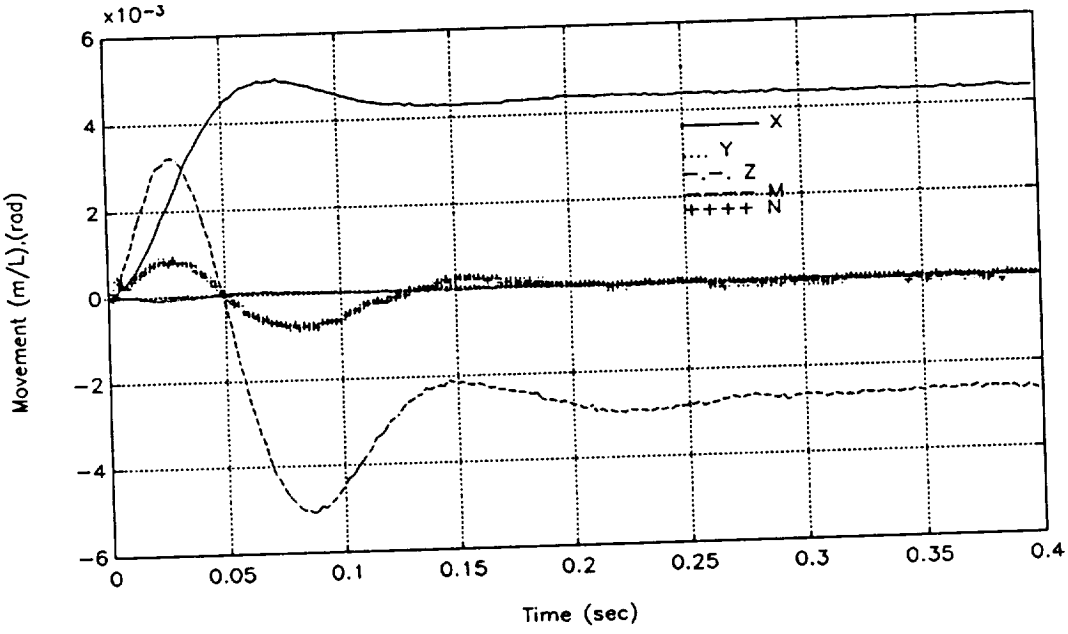


(e) - Step response in θ_z (yaw).

Figure 5.6 - Continued.

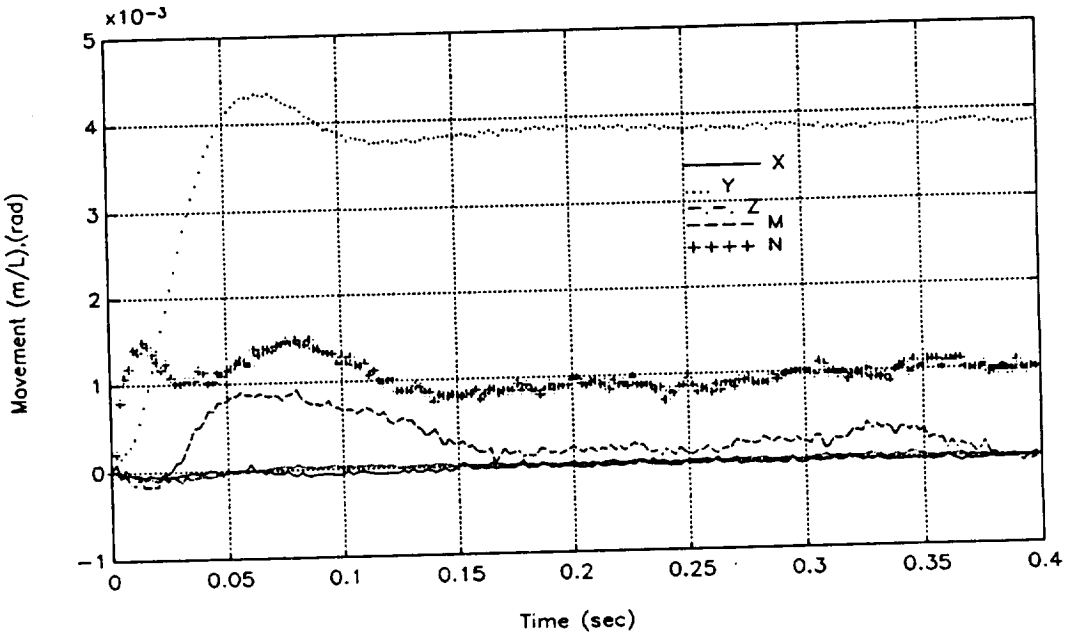
Figure 5.7 illustrates the step responses in one degree-of-freedom, with its effect on all the other degrees-of-freedom. The coupling between pitch and axial motions can clearly be seen.

It should be noted that the step size in these results are quite small. Larger steps can introduce other sources of error. In the first place, the sensors have a very small range of operation (± 3mm). Since the sensor response curves show nonlinearity at either extreme of its operating range, large steps can push the core to these limits, causing deviation from theory. Further, most of the modeling was based on the assumption that the model remains at zero position. Therefore, the steps should be reasonably small to keep the core as close to its operating point as possible, while it is large enough to minimize noise interference.

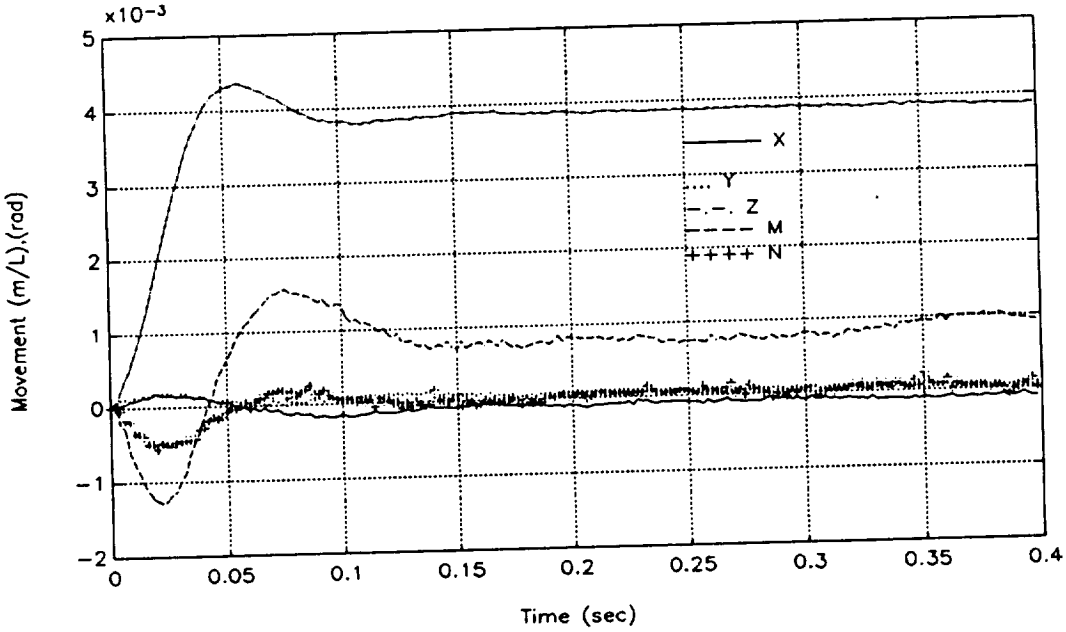


(a) - x (axial).

Figure 5.7 - Coupling among the degrees-of-freedom.

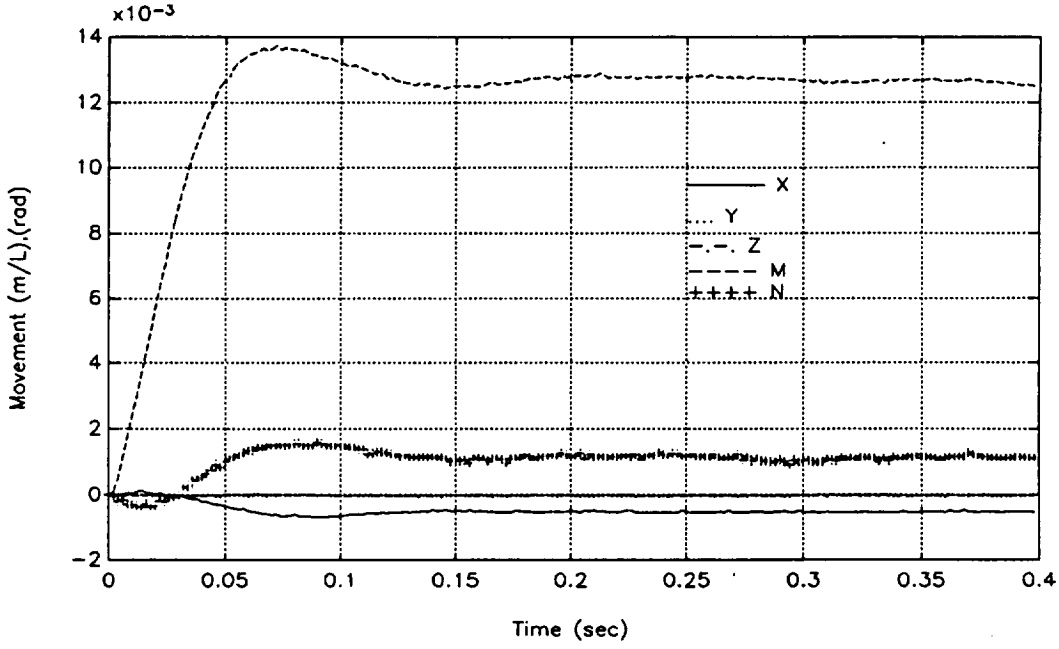


(b) - y (side).

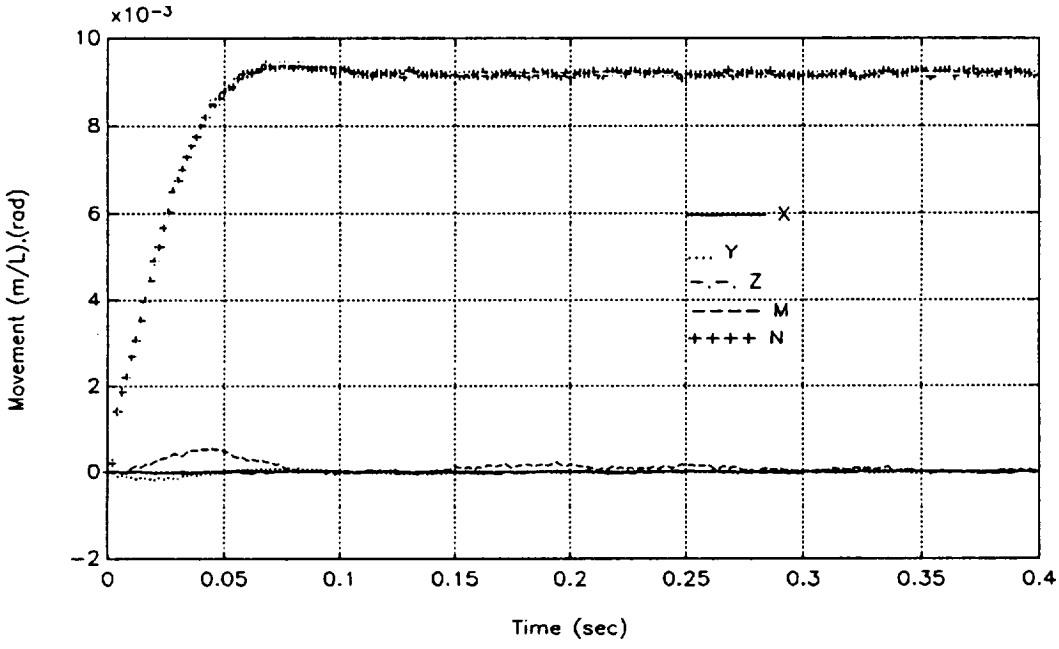


(c) - z (vertical).

Figure 5.7 - Continued.



(d) - θ_y (pitch).



(e) - θ_z (yaw).

Figure 5.7 - Continued.

CHAPTER 6

Digital Controller

The incorporation of a digital controller to the system was a necessary step in order to study the system's full capabilities. Since the system is a multi-input multi-output, coupled in most cases, it is not really suited for a fully analog controller. A digital controller would offer more flexibility and accuracy [24]. In the case of LAMSTF, one of its major design features is its 360° yaw rotation capability. As stated previously, yaw rotation requires an almost continuous change of the mixing matrix. This can easily be implemented in a control algorithm, where-as it would require a complex circuit in the analog controller.

An Intel 386 based personal computer was available, so a fully digital controller was implemented using this machine. C programming language is used to develop the controller software since it is a versatile language and is commonly used for many scientific applications. Added graphical displays and controls allow monitoring, and change of controller variables in real time. This chapter describes the present digital controller setup and the software algorithm.

6.1 Hardware Description

An A/D converter board, a D/A converter board, and a timer board are installed in a Dell 386 (33 MHz) computer. The timer board is used in the present setup as an interrupt source to the computer's processor to execute an interrupt service routine.

6.1.1 Analog to Digital Conversion

The A/D board is a Keithley Metrabyte DAS-40. It has a maximum speed of 250,000 samples per second in single channel mode. It processes the five sensor signals in single-ended multiplexed input format. It is programmed to perform direct I/O commands rather than direct memory access (DMA). The data processing sequence, acquiring samples from the five channels almost instantaneously and processing them before the next batch, was easier to implement using direct I/O technique. The board contains a pacer clock and can generate interrupts. However, no interrupts were used from this board because of insufficient on-board timer capabilities.

The A/D board is set to input voltage in the range 0-10V. For a 12-bit data conversion, a 10V range produces 2.44 mV/bit resolution.

6.1.2 Digital to Analog Conversion

The processed data is converted back to analog signals via a Keithley Metrabyte DDA-06 digital to analog converter board. This board contains six independent 12-bit D/A channels; only five of them are utilized presently. The programming of this board is straight forward using I/O instructions. Data is simply written to lower and higher bytes of each individual D/A, which are located consecutively from the base address of the board.

The DDA-06 has a selectable analog output range. Since the power amplifiers require an input range of ± 10V, the board is set to conform to this range. The 20V output range produces a resolution of 4.88 mV/bit.

6.1.3 Timer Board

One necessary requirement of the controller is to process data at a constant sample rate. If the program is simple and short, a crude method could be to add

a delay statement in the program to set the sampling rate. However, a more effective and accurate method is to use an interrupt service routine (ISR). In the early version of the digital controller used for LAMSTF, no interrupt was used. Successful suspensions were attempted, however implementation of an interrupt was still necessary for better performance. After the installation of the DAS-40, a timer board was used mainly as an interrupt generator.

The board used, a Tecmar Lab Master, is software programmable to produce a range of timer frequencies. The signal from the timer is setup to produce an interrupt request to the computer processor. The interrupt request table is programmed to service the DAS-40 board. With this configuration, the Lab Master generates interrupts at a predetermined frequency. The computer acknowledges the interrupt by calling the ISR, pointed by the interrupt level.

6.2 Discrete-Time Dual Phase Advance

The implementation of the DPA compensator requires conversion of the continuous-time design to discrete-time. Equation 4.8 represents the Laplace form of the DPA. To transform this equation to z domain, the bilinear transformation, often referred to as the Tustin's method, is used [24][25]. This transformation is shown in below:

$$s = \frac{2}{T}\frac{z-1}{z+1} \tag{6.1}$$

Replacing ω with s in equation 4.8, and using the above transformation results in the following discrete time DPA filter equation:

$$Y_k = aU_k + bU_{k-1} + cU_{k-2} - dY_{k-1} - eY_{k-2} \tag{6.2}$$

where the coefficients are as follows:

$$a = \frac{dt^2 + 4nT\,dt + 4n^2T^2}{dt^2 + 4Tdt + 4\,T^2} \tag{6.3}$$

$$b = \frac{2\,dt^2 - 8n^2T^2}{dt^2 + 4Tdt + 4\,T^2} \tag{6.4}$$

$$c = \frac{dt^2 - 4nT\,dt + 4n^2T^2}{dt^2 + 4Tdt + 4\,T^2} \tag{6.5}$$

$$d = \frac{2\,dt^2 - 8T^2}{dt^2 + 4Tdt + 4\,T^2} \tag{6.6}$$

$$e = \frac{dt^2 - 4T\,dt + 4T^2}{dt^2 + 4Tdt + 4\,T^2} \tag{6.7}$$

For a given sample rate and compensator break frequency all the above coefficients are constants.

6.3 Program Algorithm

The actual controller program is quite compact. However, many features have been added to facilitate real time observation and modification. Since the controller block is an interrupt service routine, it runs independently from the main program. The variables used in the ISR are declared as global in the main, therefore both the main and ISR can access them.

6.3.1 User Interface

The main runs continuously in a loop and monitors keyboard inputs and responds to the request made by the user. The ability to change parameters such as loop gain or the core's position, while in suspension, can be of great assistance in studying the behavior of the plant. The program allows the user to adjust the gains while the core is in suspension, and can display step responses in any degree of freedom. The graphical interface, in a form of a multi-channel oscilloscope, can plot all the core's movements in real time.

6.3.2 Dual Phase Advance

The discrete-time dual phase advance formula, shown in equation 6.2.2, is directly implemented in the program. All the coefficients are functions of the compensator's break frequency and the sample rate. The value for n has been set 10.0, as determined in the compensator's design. Based on a sample frequency of 400Hz, and the compensators break frequency of 17Hz, the DPA discrete equation becomes as follows:

$$Y_k = 23.56\ U_k - 36.02\ U_{k-1} + 13.77\ U_{k-2} - 0.287\ Y_{k-1} - 0.0206\ Y_{k-2} \quad (6.8)$$

Since there are five independent channels, the above equation is executed five times in each sample period.

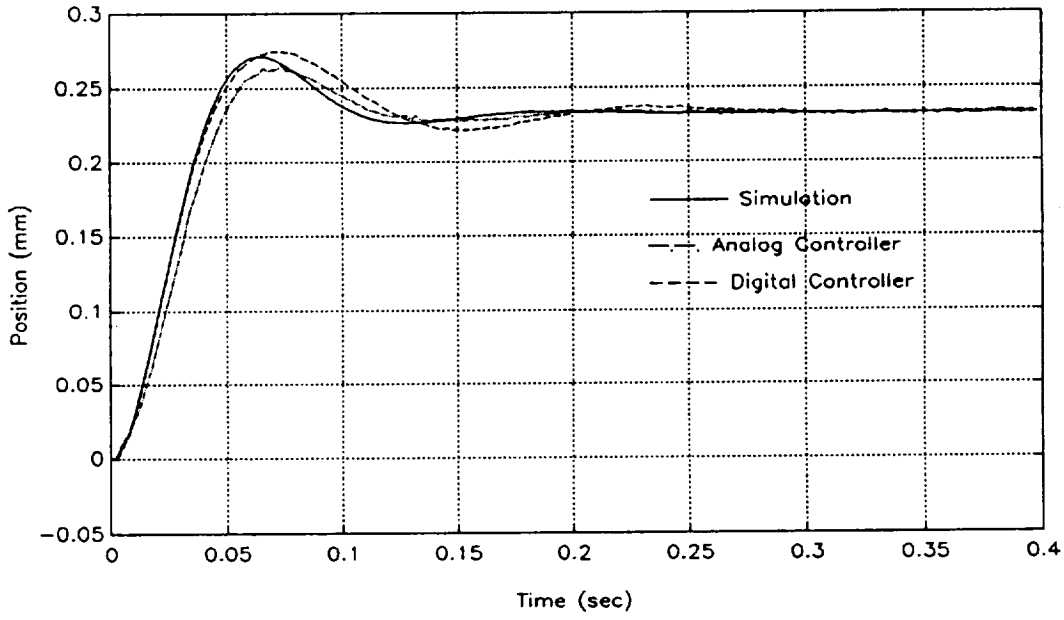
6.3.3 Interrupt Service Routine

The ISR, when initiated by the timers interrupt signal, performs five analog to digital conversion from the five sensor signals. It then performs the decoupling of the sensor signals by simple addition, subtraction, and scaling. The decoupled signals are compensated through the discrete DPA equation. The five

compensated signals are mixed through the current allocation matrix and output to the D/A board. The ISR supplies the appropriate acknowledge signals to the timer board and the computer interrupt handler.

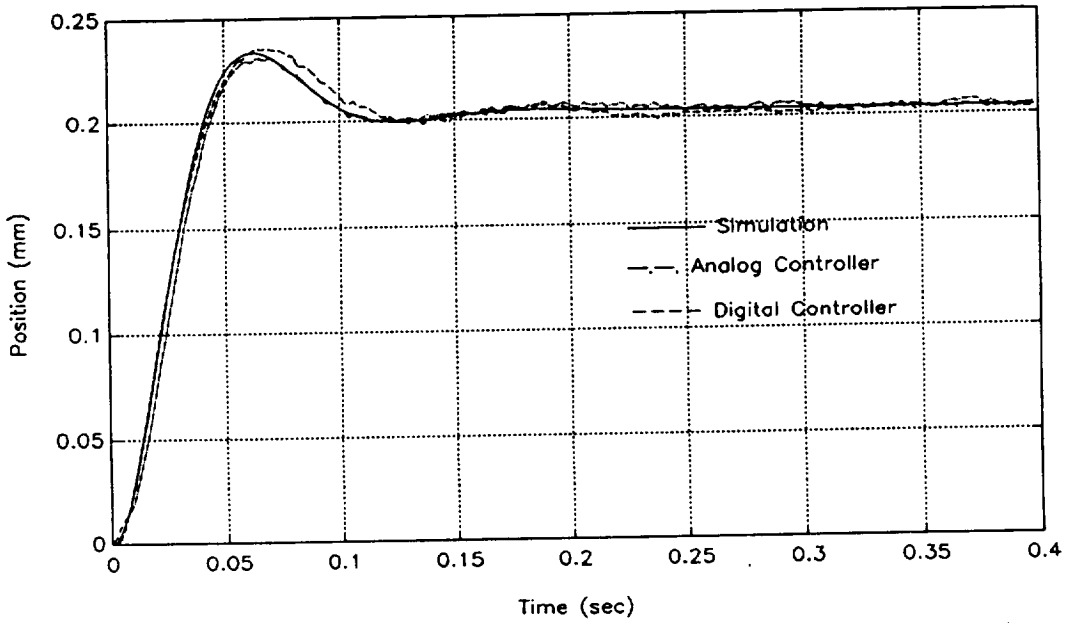
6.4 System Response

The step responses and their corresponding gains are recorded in MATLAB^{TM} readable format files. These responses are compared with the analog controller and the mathematical simulation responses. Figure 6.1 illustrates these responses for all five degrees of freedom. The simulation and the analog responses are superimposed on these plots for comparison. As with the analog controller, the digital controller's response closely resembles the simulation. Here again the pitch seems under damped compared to the simulated result.

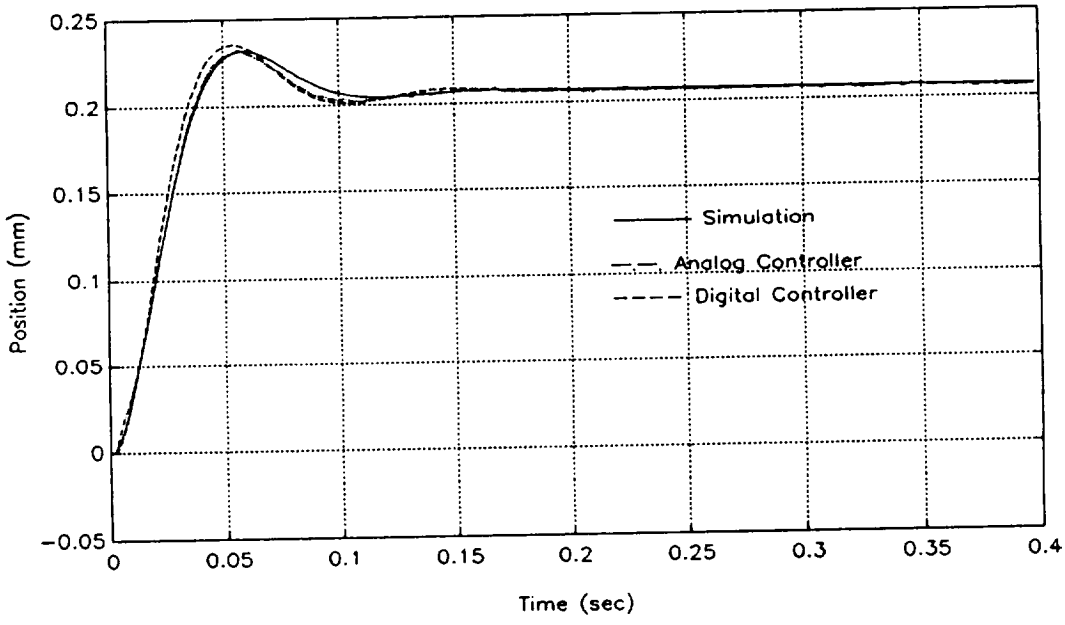


(a) - Step responses in x (axial).

Figure 6.1- Digital controller step responses.

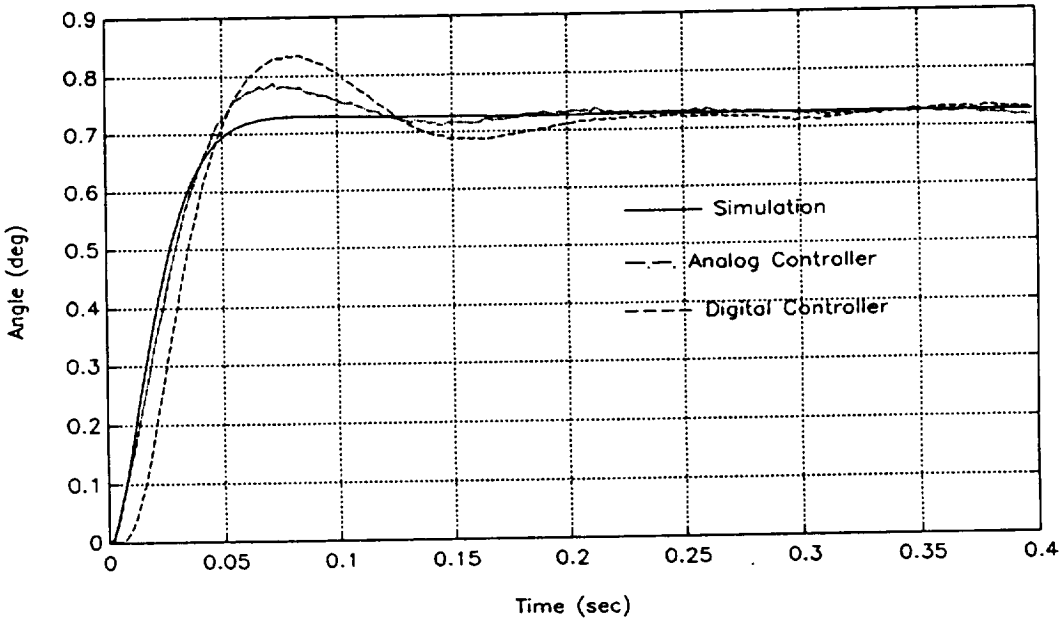


(b) - Step response in y (side).

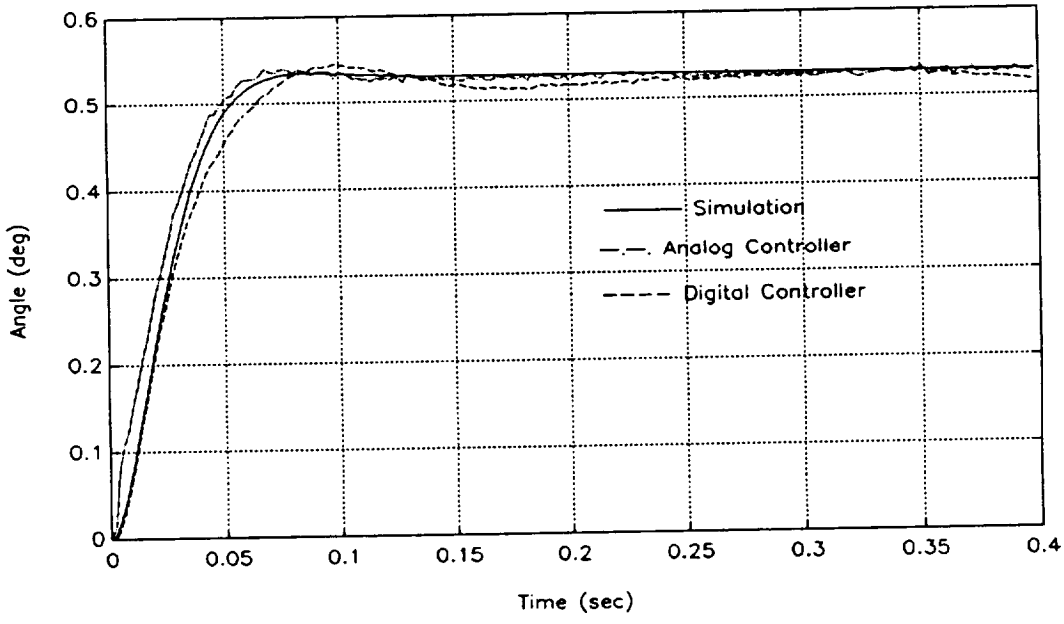


(c) - Step response in z (vertical)

Figure 6.1- Continued.



(d) - Step response in θ_y (pitch).



(e) - Step response in θ_z (yaw).

Figure 6.1- Continued.

6.5 Large Angle Yaw Rotation

One of the major point of interest in this design has been the potential to rotate the core over a range of 360° in yaw, while in suspension. This should be accomplished by rotating the sensor frame about the z axis. The five electromagnets, can then produce the fields and gradients necessary for suspension in any orientation in the x-y plane. This requires proper distribution of currents to the coils through the mixing matrix. Through some simple programming modifications this feature of LAMSTF was successfully demonstrated.

6.5.1 The Current Distribution Matrix

Considering the symmetric geometry of the planar array, after 36° of rotation, the negative $\overline{x}$ axis intersects the axis of coil four. At this configuration coil four effectively becomes a negative of coil one, coil five a negative of coil two, and so on. Therefore the mixing matrix at 36° is identical to that at 0°, with its rows rearranged to point to their new coil sequence. This pattern repeats every 36°; at 72° coil two becomes as coil one. Using this property, determination of the mixing matrices can be limited to only six. An array of 5×5 mixing matrices were calculated at 6° intervals, from 0° to 36°. Again the finite element method software, VF/GFUN, was used to generate the elements of these matrices, and every column was normalized individually. The resulting six matrices were copied and modified to form 60 matrices in a full 360° rotation. The $5 \times 5 \times 60$ matrix is entered into the program through a data file during the initialization of the controller program.

The basic idea is that, by interpolating between every adjacent matrix, finer matrix transition could be achieved. A simple linear interpolation routine is implemented. The interpolation equation used is as follows:

$$\tilde{m}_{ij} = m_{ij[k-1]} + d(m_{ijk} - m_{ij[k-1]}) \tag{6.9}$$

and

$$d = \frac{\theta_{actual} - \theta_{[k-1]}}{6^{\circ}} \tag{6.10}$$

$$\theta_{[k-1]} = \text{integer}\left(\frac{\theta_{actual}}{6^{\circ}}\right) \tag{6.11}$$

The interpolated mixer elements $\tilde{m}_{ij}$ are sent to the ISR to replace the previous values.

Provisions were made for negative angles and angles beyond $\pm 360^{\circ}$. This was simply a matrix index manipulation to reverse the indices for negative angles, and to point to zero index after a 360° rotation.

6.5.2 Initial Current Setting

Besides the mixing matrix, the initial current vector has to be modified for each new yaw position. The expectation was that the initial currents should follow a sinusoidal trend, with every coil current shifted by 72° in phase (see figure 6.2). To accomplish this, the maximum amplitude was defined to be the current value for coil one at zero angle yaw. All coil current values are calculated by taking the cosine of the angle plus their phase shift. Coil one has 0° phase shift, coil two 72° phase shift, and so on. This would produce a 0° yaw suspension current set as follow:

$$I_0 = [\ \text{-14.5A}\ \ \text{-4.5A}\ \ 11.8\text{A}\ \ 11.8\text{A}\ \ \text{-4.5A}\ ] \tag{6.12}$$

In the program it was found that using the cosine function continuously would encounter program errors at times; perhaps due to interrupt conflicts. For this

reason and to improve calculation speed, cosine values for a range of values are calculated and stored in an array. As the new position is sensed, the index to point to the corresponding cosine array element is determined.

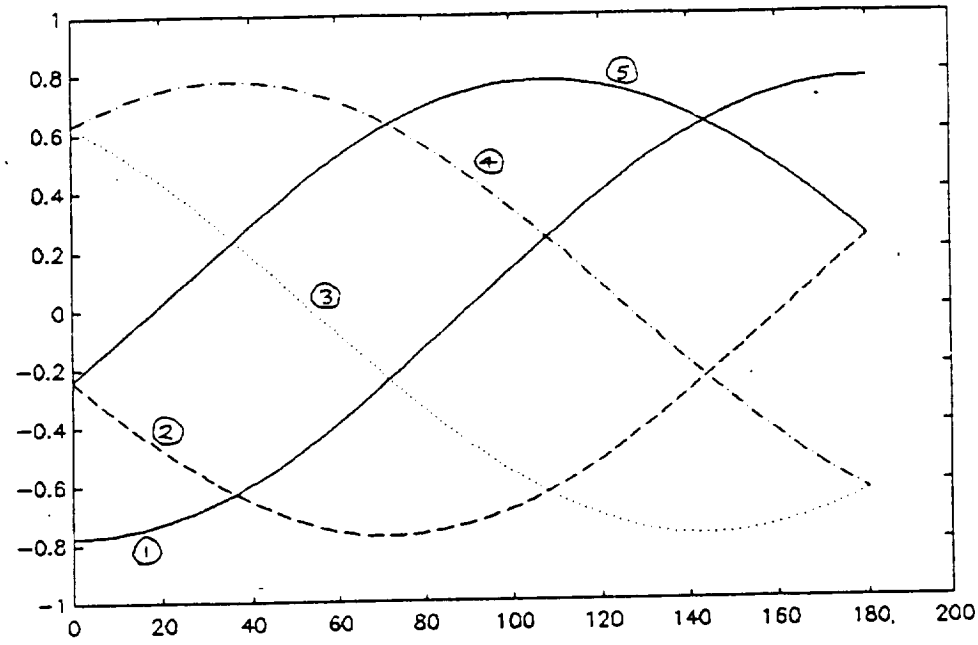


Figure 6.2- Current trend in the five coils for a 360° yaw rotation.

6.5.3 Mixer Switching Technique

Now that the provisions have been made to select a mixing matrix and an initial current set, a yaw angle input to the program is necessary. In the initial testing, the sensor frame was rotated and the yaw angle was manually increased to track the core. A precise method would be to attach an encoder to the sensor frame to monitor the sensor frame's position. However, since the controller is of type zero, and there is a steady state error in all degrees of freedom, this error may be used as a crude tracking signal. The yaw error signal is a low-amplitude random signal superimposed on a D.C. value. With the core suspended, the steady state error (the D.C. value) tends to increase or decrease as the sensor

frame is rotated. The core initially is suspended at a known yaw angle. As the error magnitude crosses a critical level with the rotation of the sensor frame, the yaw angle is stepped by one unit; presently a unit is set to a half a degree. This allows the program to keep track of the yaw angle without an actual feedback from the hardware. In the program the error in yaw is continuously monitored, once per loop. If the critical error level in detected a subroutine is called to determine the new mixing matrix and zero position currents. The ISR has immediate access to the calculated values.

One important factor in using the above technique is the random noise interference. Care must be taken not to allow instantaneous large magnitudes in the noise to be a cause of yaw angle change. A simple low-pass filter is employed to eliminate some of effects of the noise. For this purpose, a moving average filter was used. The equation for the filter used is as follows:

$$Y_k = \frac{1}{3}(Y_k + Y_{k-1} + Y_{k-2}) \tag{6.13}$$

The performance of the above filter seemed fairly satisfactory.

A series of photos of the suspended element at 30 degrees steps in yaw angle, from 0° to 120°, are shown in figure 6.3.

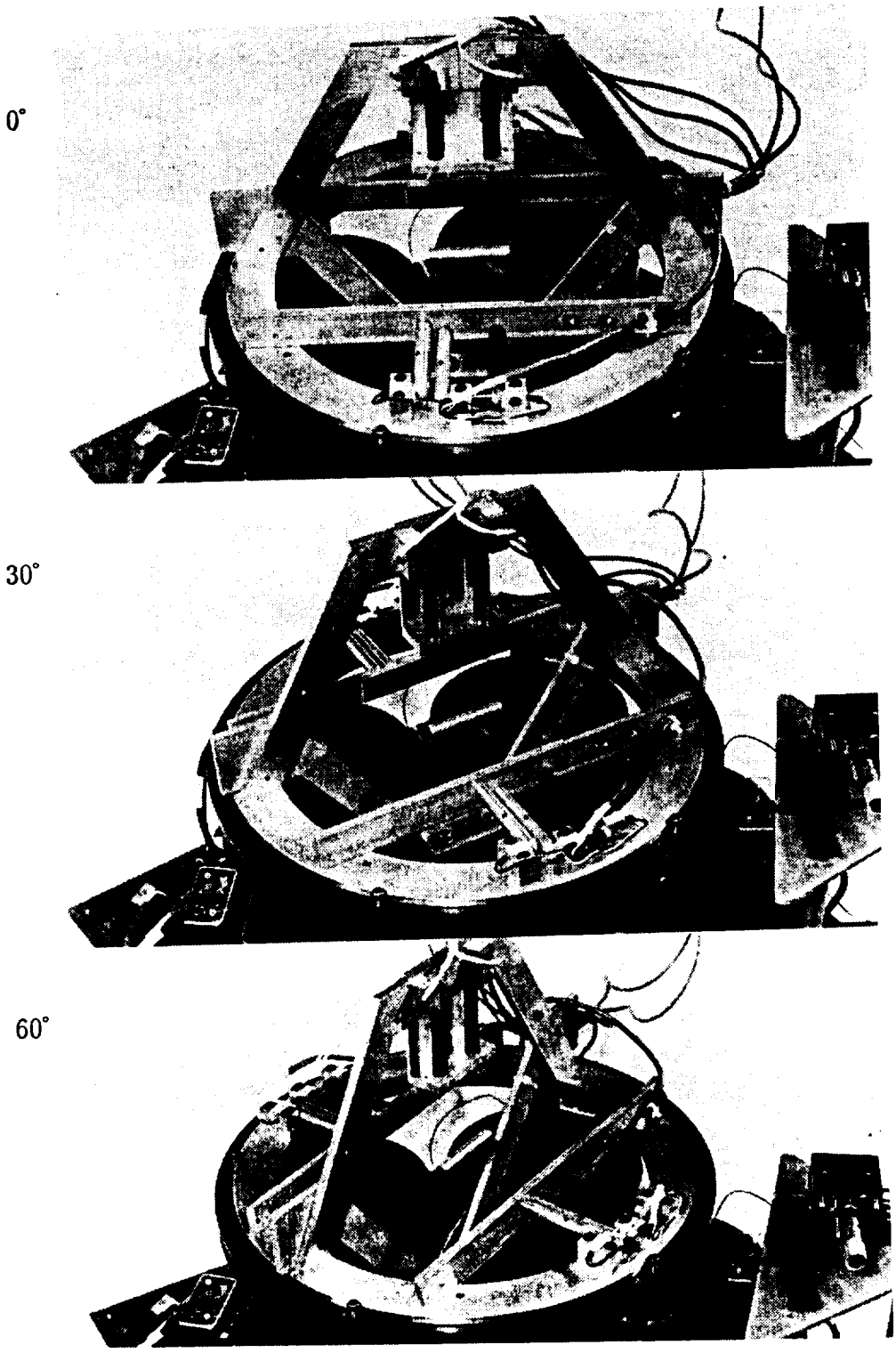
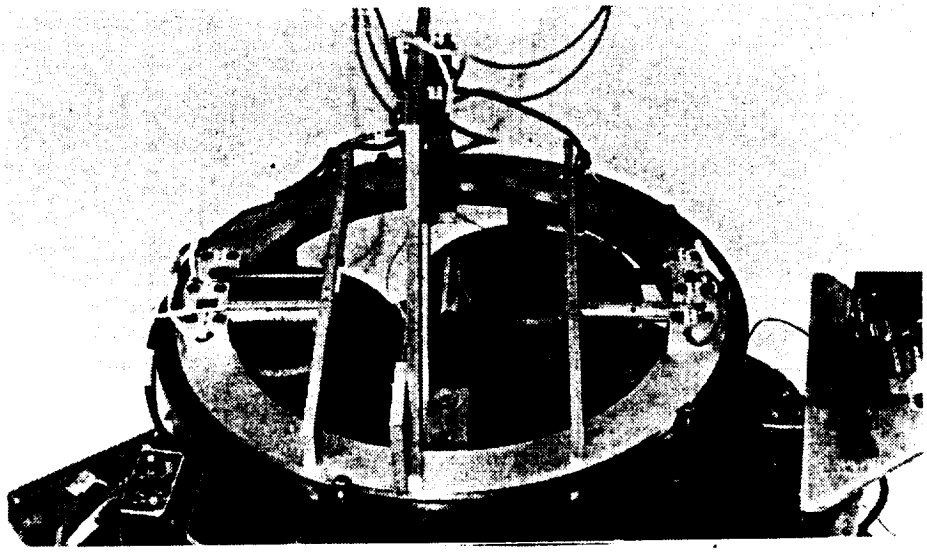


Figure 6.3 - Demonstration of large angular rotation.

90°



120°

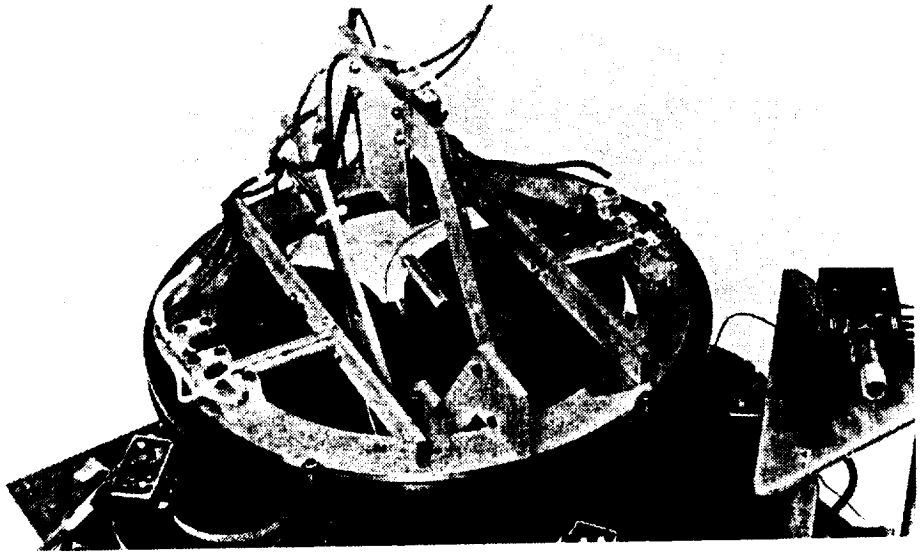


Figure 6.3 - Continued.

CHAPTER 7

Discussion

The overall performance of LAMSTF proved to be quite satisfactory. Even though its unique design had never been tested in hardware, successful operation was achieved without major difficulties. The system proved to be fairly robust. It would work well with mis-alignments in the sensor frame or small displacements from its designed operating point. The system response seems to match the simulated results very closely, except in the case of pitch which is a little under damped. The actual cause for this discrepancy has not yet been confirmed. However, some inaccuracies were discovered in the hardware which may contribute to this problem. It was found that the core suspends slightly above its designed operating point due to the position of the sensor frame. The fields and the field gradients, used in the simulation, have been calculated based on the designed operating point. This may cause noticeable error in more unstable modes such as the pitch. Another hardware inaccuracy was found in the position of the coils in the array. The radius of the coil array is about 1/8" larger than the actual design. This would result in lower field intensity at the operating point. New hardware is under construction to eliminate these errors. The eddy currents in the base plate are another potential source of error in the results. Some study of the effect of the eddy currents, induced in the base plate, directly under each coil, suggested negligible effect on the system. However, more

investigation should be done on the eddy currents generated by combination of coils, and their effects on the system response.

The digital controller proved to be very useful to the project, since it enhances the ability to make accurate measurements of the system's response, and makes the yaw rotation possible. The use of the ISR greatly increases the capabilities of the program. Since the main program itself does not have to run the controller, it can perform many added features such as the graphics display and the yaw rotation calculations, without affecting the sample rate.

The method utilized in switching between yaw angles can be refined to produce smoother transitions, without the direct use of the error in yaw. One proposed method is to use the current distribution in the coils as a mean of yaw angle detection. However, more studies need to be performed to establish its feasibility.

During the operation of the LAMSTF, structural vibration has been observed at higher controller gains. The frame along with the sensor electronics mounted on them are subject to the magnetic fields from the coils. The changing magnetic fields can induce current in the conductors. The structural vibration may be attributed to induction of the changing magnetic fields to the sensor and LED wiring. These unwanted signals are carried to the controller as a false position signal which produces correction signals from the controller. The correction signals are fed back to the current amplifiers, which produce more changing magnetic fields, causing oscillations. Some steps need to be taken to shield the cables and sensors from the magnetic fields, or to position the sensors further away from the coils. The sensor frame is mounted on bakelite studs which are directly bolted to the iron cores in the coils. As the current changes in the coils, the iron core, which behaves like a mass and a spring fastened to the base plate, moves and transmits vibrations to the sensor frame. These vibrations are again

carried to the controller as a position change from the suspended element, causing an oscillation. Again a new sensor frame mount, decoupled from the coils, is being constructed.

CHAPTER 8

Conclusion

The principle of the planar array has been proven in practice. The existing hardware was made operational using an analog and a digital controller. A simple and well tested controller approach, using the dual phase advance controller, was employed. The ability of the system to sustain suspension in a full 360° yaw rotation was demonstrated.

One of the decoupled degrees-of-freedom, yaw, was analyzed using the DPA compensator. The design was successfully implemented in all five degrees-of-freedom.

An analog controller, using op-amps, was designed and built. Actual suspension of the core was achieved without any major difficulties. Step responses from the analog controller were recorded and compared to the simulated results. All degrees-of-freedom were found to match closely to the simulation except the pitch degree-of-freedom which seemed slightly under damped.

A 386 based personal computer, equipped with an A/D, a D/A, and a timer, was used to develop a digital controller for the LAMSTF. Using the C programing language, a control software was developed for the system. The software utilizes an ISR to process the control algorithm. This increases the capabilities of the program to perform other tasks such as useful user interfacing. The software allows viewing and changing of the suspended element positions

graphically in real time. The user is able to change the controller gain and view and record step responses.

Step responses of the system, using the digital controller, were compared to the analog and simulated results. Like the analog controller, the digital controller showed a close match to the simulation, except in the pitch degree-of-freedom.

One of the major aims of the LAMSTF, to demonstrate the 360° yaw rotation, was accomplished. A series of current distribution matrices at 6° intervals were calculated and used in the program for the yaw rotation capability. The program uses a linear interpolating scheme to calculate finer mixing matrix components between the 6° calculated steps. A sinusoidal zero position suspension current trend for the yaw rotation was proven to work in practice.

REFERENCES

[1] Jayawant, B. V.: "Electromagnetic Levitation and Suspension Techniques." Edward Arnold, 1981.

[2] Holmes, F. T.: "Axial Magnetic Suspensions." Review of Scientific Instruments, Vol. 8, Nov. 1937, pp. 444-447.

[3] Proceedings of The Third International Symposium on Magnetic Bearings, Alexandria, Virginia, July 29-31, 1992.

[4] US Department of Transportation: "Assessment of the Potential for Magnetic Levitation Transportation Systems in the United States." Congressional Report, 1990.

[5] Britcher, C. P.; Parker, H. D.: "Progress Towards Extreme Attitude Testing with Magnetic Suspension and Balance Systems." AIAA Paper 88-2012, 1988.

[6] International Workshop on Vibration Isolation Technology for Microgravity Science Applications. Cleveland, Ohio, NASA Lewis, April 23-24, 1991.

[7] Johnson, R. G.; Pang, D.; Kirk, J. A.; and Anand, D.K.: "Physical Modeling of High Speed Magnetic Bearing Systems." Proceedings of the Third International Symposium on Magnetic Bearings, pp 474-482, July 29-31, 1992.

[8] Britcher, C. P.: "Large-Gap Magnetic Suspension Systems." International Symposium on Magnetic Suspension Technology, NASA CP 3152, pp 33-42, August 19-23, 1991.

[9] Carmichael, A. T.; Hinchliffe, S.; Murgatroyd, P. N.; and Williams, I. D.: "Magnetic Suspension Systems with Digital Controllers." Review of Scientific Instruments, Vol. 57, No. 8, August 1986.

[10] Groom, N. J.; Britcher, C. P.: "A Description of a Laboratory Model Magnetic Suspension Test Fixture with Large Angular Capability." Proceedings of The First IEEE Conference on Control Applications, Vol 1, pp 454, 1992.

[11] Groom, Nelson J.; Schaffner, R. P.: "An LQR Controller Design Approach for a Large Gap Magnetic Suspension System (LGMSS)." NASA TM- 101606, July 1990.

[12] Britcher, C. P.; Ghofrani, M., Britton, T., Groom, N. J.: "The Large-Angle Magnetic Suspension Test Fixture." International Symposium on Magnetic Suspension Technology. NASA CP 3152, pp. 971-985, 1992.

[13] Boom, R. W.; Abdelsalam, M. K.; Eyssa, M. Y. ; McIntosh, G. E.: "Repulsive Force Support System Feasibility Study." NASA CR-178400, 1987.

[14] Groom, N. J.; Britcher, C. P.: "Stability Considerations for Magnetic Suspension Systems using Electromagnets Mounted in a Planar Array." NASA CP-10066, Vol. 1, pp. 355-376, March 1991.

[15] Groom, N. J.: "Analytical Model of a Five Degree of Freedom Magnetic Suspension and Positioning System." NASA TM- 100671, March 1989.

[16] Groom, N. J.: "Overview of Magnetic Suspension Research at Langley Research Center." International Symposium on Magnetic Suspension Technology, NASA CP-3152, Part 1, pp 3-13, 1992.

[17] The VF/GFUN Reference Manual. VF068894, Vector Fields Limited, June 20, 1988.

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

[19] Kraus, J. D.: "Electromagnetics." McGraw-Hill Book Co. Third Edition 1984.

[20] Goodyer, M. J., "The Magnetic Suspension of Wind Tunnel Models for Dynamics Testing." Ph.D. Thesis, April 1968, Published as N78-78589, Ch. 1-8, and N78-78218, Ch. 9-14.

[21] Britcher, C. P.; Goodyer, M. J.; Eskins, J.; Parker, D.H.; and Halford, R.J.: "Digital Control of Wind Tunnel Magnetic Suspension and Balance Systems." IEEE, Inc., 1987, 9,

[22] Britcher, C. P.; Kilgore, W. A.; and Haj, A.: "Comparison of Digital Controllers Used in Magnetic Suspension Systems."

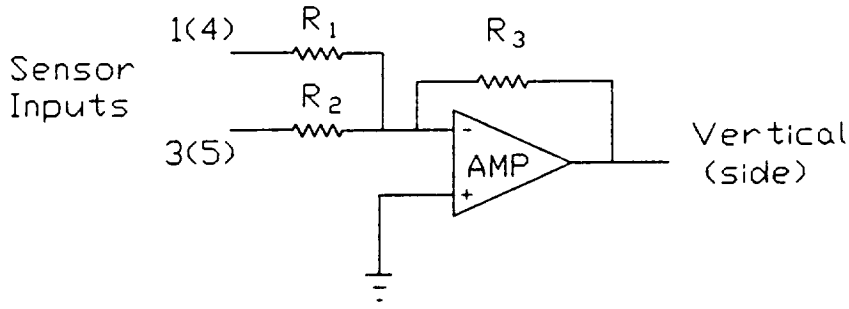
[23] D'Azzo, John J., C. H. Houpis, "Linear Control System Analysis & Design Conventional and Modern." Third Edition McGraw-Hill Book Company, New York, 1988.

[24] Kilgore, W. A.: "Comparison of Digital Controllers Used in Magnetic Suspension and Balance Systems." NASA CR-182087, December 1989.

[25] Franklin, G. F.; Powell, J. D.; Workman, M. L.: "Digital Control of Dynamic Systems." Addison-Wesley Publishing Co., Second Edition, 1990.

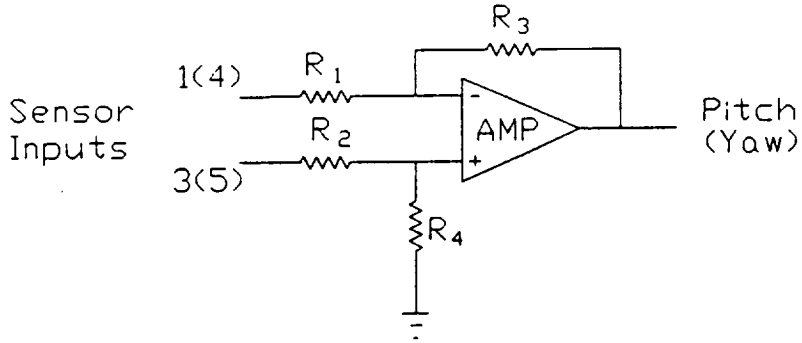
Appendix A

Circuit Diagrams



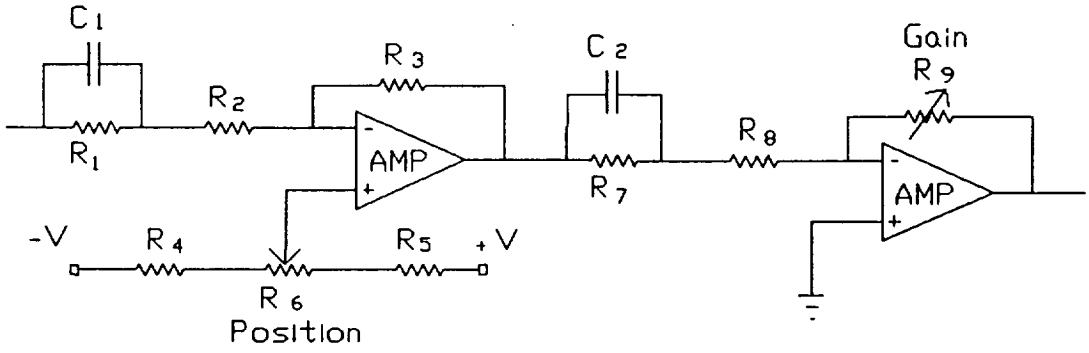
Summing amplifiers

$R_1 = R_2 = 10\ k\Omega,\ R_3 = 5\ k\Omega$



Differencing amplifiers

$R_1 = R_2 = R_3 = R_4 = 25\ k\Omega$



Dual phase advance circuits

$R_1 = R_7 = 100\ k\Omega$

$R_2 = R_8 = 10\ k\Omega$

$R_3 = 56\ k\Omega$

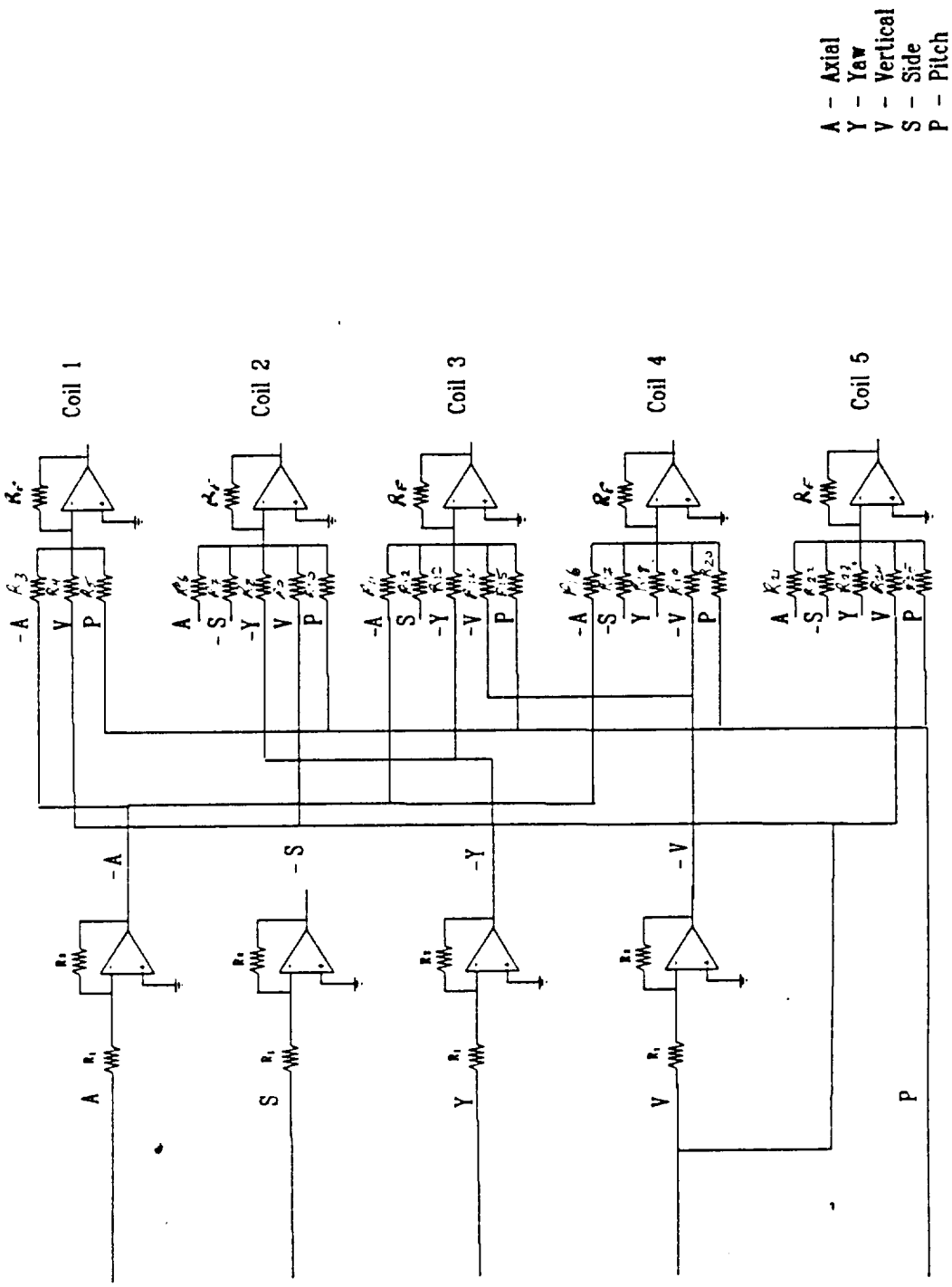
$R_4 = R_5 = 0\ \Omega$

$R_6 = 10\ k\Omega$

$C_1 = C_2 = 0.1\ \mu f$

Op-amps LF-353

$\pm V = \pm 5\ V$



Mixer circuit diagram

$R_1 = R_2 = 20\ k\Omega$

$R_f = 10\ k\Omega$

$R_3 = 10\ k\Omega$

$R_4 = 10\ k\Omega$

$R_5 = 12.96\ k\Omega$

$R_6 = 12.31\ k\Omega$

$R_7 = 16.18\ k\Omega$

$R_8 = 10\ k\Omega$

$R_9 = 32.36\ k\Omega$

$R_{10} = 10\ k\Omega$

$R_{11} = 32.36\ k\Omega$

$R_{12} = 10\ k\Omega$

$R_{13} = 16.18\ k\Omega$

$R_{14} = 12.31\ k\Omega$

$R_{15} = 13.02\ k\Omega$

$R_{16} = 32.36\ k\Omega$

$R_{17} = 10\ k\Omega$

$R_{18} = 16.18\ k\Omega$

$R_{19} = 12.32\ k\Omega$

$R_{20} = 13.02\ k\Omega$

$R_{21} = 12.31\ k\Omega$

$R_{22} = 16.18\ k\Omega$

$R_{23} = 10\ k\Omega$

$R_{24} = 32.36\ k\Omega$

$R_{25} = 10\ k\Omega$

Appendix B.

Program Listing

```
/*************************************************************************/
/*                        L.A.M.S.T.F.                                   */
/*          (Large Angle Magnetic Suspention Test Fixture)               */
/* Controller program for the five degrees of freedom magnetic           */
/* suspension system, at NASA Langley Hampton, Virginia.                 */
/*                 Program By: Mehran Ghofrani                           */
/*                  Old Dominion University                              */
/*************************************************************************/
#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>


#define Address 512   /* A/D base address  */
#define ADC_2 544   /* Timer board Address */
#define DDAbase 816   /* 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
/*********************************************************************/
#pragma intrinsic(outp,inp)

/* External Mode call (Function ) Prototype */
extern mscl_das40(int*,int*,int*);

#define Inport 512

/* das-40 variables */
int mode,flag,params[10];

/* Pointers to Allocated ram Buf's   */
unsigned int *DMA_buf,*bufoffset;

unsigned int chan=10;

/* Channel and Gain Array to use by Mode 2 */
int chans[5]={0,1,2,3,4};
int gains[5]={0,0,0,0,0};

int far *chans_ptr;
int far *gains_ptr;


/*********************************************************************/
  unsigned int adcsr;
  int dec,sampler;
  int y11,x1,samplrate,diviser;
```

```
  int intnum=0x0d;  /* interrupt vector number */
  int input,k,j,num,plotcolor=0,datcolor=15;
  int ii,i,l,x,y,chd,olddat[500][5];
  int zoom=1,ang=0;
  int positionsign,gainupdown,trig;
  int dec,tmp[5]={53,101,149,197,245},kv,ix=25;
  double a11,a,b,c,d,e,f,current[5],savdat[5][500];
  double v,v1[25],v2[25],vsensor[5],initialoffset[5]={4.86,1.631,-4.133,-4.133,1.631};
  double offset[5]={4.86,1.631,-4.133,-4.133,1.631};
  double gain[5]={.5,.5,.5,.1,.08},gainstep=.001,position[5],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 mixall[5][5][61],yawangle=0,yawanglestep=0.5,filt[5],autoyaw=0;
  double cosdat[800];
  int maxnum,yawerrorlimit;
  char ch,ch1,ch2,plotmode,loop;
  int rota=0;
  long backc;
/*********************************************************************/

  void (_interrupt _far *oldnum)(void);
  void _interrupt _far datin(void);
/*********************************************************************/

  void process_error(void);
  void initialize(void);
  void oscilloscope(void);
  void timerset(void);
  void adcinitialize(void);
  void plotdat(void);
  void keyboard(void);
  void shutdown(void);
```

```
    void outdat(void);
    void gainplot(void);
    void positionplot(void);
    void trigplot(void);
    void compensator(void);
    void timersetup(void);
    void acknowledge(void);
    void savedat(void);
    void getdat(void);
    void newmixer(void);
    void cosmak(void);
    void beactive(void);


main()
{
/*********************************************************************/
/* ISR vector setup */
/*********************************************************************/
    oldnum=_dos_getvect(intnum);   /* 0x0d IRQ-5 */
    _disable();
    _dos_setvect(intnum,datin);
    _enable();
   outp(0x21,0x00);
   outp(0x20,0x20);

    _setvideomode(_VRES16COLOR);  /*select video configuration */

/*********************************************************************/
    cosmak();
    getdat();
    adcinitialize();   /* Initialize the A/D converter board */
    printf(" Enter Sample rate Please :  ");
    scanf("%d",&samplrate);  /* Get the sampling rate */
    initialize();   /* Compute The compensators parameters */
    oscilloscope(); /* Display the Controller Screen */
```

```
    timersetup();   /* Set timer to interrupt at sample-rate */
/************************************************************************/
/* 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");
acknowledge();

  while(ch!=ESC)
{

keyboard();      /* Check for user input */

    filt[4]=filt[3];
    filt[3]=filt[2];
    filt[2]=filt[1];
    filt[1]=filt[0];
    filt[0]=v1[14];
    autoyaw=.5*(filt[0]+filt[1]);
    _settextposition(28,64);
    printf("%5.2f",autoyaw);

    if(rota==1)
     {

      if(autoyaw>=2)
        {
          yawangle=yawangle+yawanglestep;
          newmixer();
```

```
            yawangle=yawangle+yawanglestep;
             newmixer();
          }
        if(autoyaw<=-2)
          {
             yawangle=yawangle-yawanglestep;
            newmixer();
            yawangle=yawangle-yawanglestep;
            newmixer();
          }
        }
for(j=0;j<=4;++j)
  {
    v=v1[2+3*j];
    if(sampler<=2)
      { _setcolor(14);
        _rectangle(_GFILLINTERIOR,600,345,620,340);
      }
    else
       if(sampler>200)
          { _setcolor(1);
            _rectangle(_GFILLINTERIOR,600,345,620,340);
           }

if(chd!=0)
    {
       if(!trig || ix<474)
          {
             if(plotmode=='a') plotdat();
             if(plotmode=='s')
               {
                  switch(ch2)
                    {
                      case 'x': if(j==0) trigplot();
                      break;
                      case 'y': if(j==1) trigplot();
```

```
                        break;
                        case 'z': if(j==2) trigplot();
                        break;
                        case 'm': if(j==3) trigplot();
                        break;
                        case 'n': if(j==4) trigplot();
                        break;
                    }
                }
            }
        }
}

/*********************************************************************/
    if(trig==0)
    if(ix>=475) ix=25;/* reset screen position to begining when at end */
    if(ix<=475) ix++;
/* screen horizental advance chk plotdat */
}
    _setcolor(2);
    _rectangle(_GFILLINTERIOR,550,420,637,477);
    _settextposition(28,71);
    _outtext("R-Run");
    _settextposition(29,71);
    _outtext("ESC-exit");
    ch='.';
    ch1=getch();
 }
/*********************************************************************/
    _clearscreen(_GCLEARSCREEN);
     _settextposition(1,1);
     printf("The gains are:\n");
     for(j=0;j<=4;++j)
     printf("Gain[%d]=%8.5f\n",j,gain[j]);
    _settextposition(20,30);
/*********************************************************************/
```

```
    _setvideomode(_DEFAULTMODE);
    beactive();   /* keep the controller running  */
}
/***************---- end of main ---------- ******************************/
/*****************************************************************************/
/* oscilloscope: signal display screen setup       */
void oscilloscope(void)
{
  int ll,jj,gainstart,posibar;
/* 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 oscilloscope screen      */
 _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(8);
   backc=8;
  _setbkcolor(backc);
  _rectangle(_GFILLINTERIOR,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=320;ll<=448;ll=ll+32)      /* gain control bar display */
      _rectangle(_GFILLINTERIOR,42,ll,195,ll+20);
  jj=0;
  for(ll=320;ll<=448;ll=ll+32)    /* Presetting gain display */
  {   /* on the screen     */
      gainstart=(int)(gain[jj]*80)+40;
      _setcolor(7);
      _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(_GFILLINTERIOR,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 - pith movement */
_settextposition(29,2);
_outtext("N");   /*N - yaw movement */
_settextposition(2,62);
/***********************************/
/* Display of options */
printf("E- enable plot");
_settextposition(3,62);
printf("D- disable plot");
_settextposition(5,62);
printf("  Select Gain ");
_settextposition(6,62);
printf("X- gain of x");
_settextposition(7,62);
printf("Y- gain of y") ;
_settextposition(8,62);
```

```
printf("Z- gain of z") ;
_settextposition(9,62);
printf("M- gain of pitch") ;
_settextposition(10,62);
printf("N- gain of yaw") ;
_settextposition(11,62);
printf("S - single plot");
_settextposition(12,62);
printf("A - all plots");
_settextposition(13,62);
printf("T - triggered ");
_settextposition(14,62);
printf("C - continuous");
_settextposition(15,62);
printf("F1- unzoom plot");
_settextposition(16,62);
printf("F2- zoom plot");
_settextposition(17,62);
printf("F4- Save data");

}
/*************************************************************************/
/* keyboard: User interaction through keyboard                           */
void keyboard(void)
{
  char chin;
  if(kbhit()!=0)
    {
     chin=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=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;
       _settextposition(30,64);
       printf("Off",zoom);

       break;
       case F10: rota=1;
       _settextposition(30,64);
       printf("On ",zoom);
       break;
}
break;
case 'E':case 'e':  chd=1;
break;
case 'D':case 'd': chd=0;
break;
case ESC: ch=ESC; break;
```

```
 }
 }
    chin='.';
}
/*************************************************************************/
/* initialize: setting the parameters of the dual phase advance compensator */
  void initialize(void)
{
  double dt,T,n,Tnd;
  int breakfreq;
  printf("\n Enter Compensator's Break frequency (170hz): ");
  scanf("%d",&breakfreq);    /* Get the break frequency */
  dt=1/((double) samplrate);    /* period of the sampling */
  T=1/(2.0*pi*(double)(breakfreq));    /* 1/ break frequency*/
  printf("T= %9.6f\n",T);
  n=10.0;    /* 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("\n");
  printf("\n");
  printf("\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();
  }

 /*************************************************************************/
```

```
/* Plotdat: plot of signals */
void plotdat(void)
{
 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;
 }
/*************************************************************************/
/* trigplot: plot of signals */
void trigplot(void)
{    int plotcenter=148;

 kv=plotcenter-(int)(v*10*zoom);
if(kv>=290) kv=290;
_setcolor(plotcolor);
_setpixel(ix,olddat[ix][j+1]);
_setcolor(datcolor);
_setpixel(ix,kv);
olddat[ix][j+1]=kv;

 }
/*************************************************************************/
/* 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=6+gainupdown;
```

```
                _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]<0) gain[0]=0;
            if(gain[0]>2) gain[0]=2;
            break;
            case 'y':
            if((gain[1]>=0) && (gain[1]<=2))
            {
            knobcolor=6+gainupdown;
           _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=6+gainupdown;
              _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=6+gainupdown;
      _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=6+gainupdown;
      _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;
```

```
    }
}
/*************************************************************************/
/* 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(bkcolor);
     _rectangle(_GFILLINTERIOR,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(bkcolor);
     _rectangle(_GFILLINTERIOR,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(bkcolor);
```

```
        _rectangle(_GFILLINTERIOR,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(bkcolor);
        _rectangle(_GFILLINTERIOR,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(bkcolor);
        _rectangle(_GFILLINTERIOR,310,448,461,468);
        _setcolor(knobcolor);
        position[4]=position[4]+positionsign*positionstep;
      /*_settextposition(15,62);
        printf(" %f",position[4]);     */
        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;
        }
}
/************************************************************************/
/* shutdown: outputs zero to current drivers*/
void shutdown(void)
{
```

```
        int dtoa;
        for(dtoa=0;dtoa<=4;++dtoa)
                {
                        outp(DDAbase+dtoa*2,0);/* 00000000 for low byte at zero*/
                        outp(DDAbase+dtoa*2+1,8);/* 1000 for high byte at zero */
                }
}
/*************************************************************************/
/* timersetup: sets up the sample rate clock on the second ADC for interrupt */
void timersetup(void)
{
  int rate;
/* Timer ic AM9513 settings*/
   outp(ADC_2+4,196);  /* set no auto increment, interrupt on adc,
   external trriger */
   outp(ADC_2+9,23);/* Data pointer to master mode reg. Control Group*/
   outp(ADC_2+8,0); /* Set master mode register for scalar*/
   outp(ADC_2+8,144); /* BCD division, enable increment, 8 bit bus*/
/* FOUT off, compar disabled ; TOD disabled*/
   outp(ADC_2+9,5); /* 5 Set data pointer to counter 5    */
   outp(ADC_2+8,33);/* 49 No gating, Count on rising edge,*/
   outp(ADC_2+8,13); /* 13 F3 source (10000 hz),
/* disable special gating Reload from load, repeat count, Binary, */
/* count down, active high output*/
    rate=(int)(10000.0/samplrate);
    outp(ADC_2+8,rate); /* Binary number for the counter to repeat*/
    outp(ADC_2+8,0);
    outp(ADC_2+9,112);/* load and start    */
}
/*************************************************************************/
/* 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=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");
     }
     _settextposition(25,64);
      printf("                ");
     _settextposition(26,64);
     printf("                ");
 }
/*****************************************************************************/
/* cosmak : calculates cos values for angles 0..360  */
void cosmak(void)
```

```
{   int i,number;
    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;
}
/*****************************************************************************/
/* newmixer : calculates new mixing matrix and current distribution.*/
void newmixer(void)
{  int i,j,k,nextone,index1,index2,index3,index4,index5,seventy2;
  int angindex;
  double angstep=6, yawangletemp;
  double mixtemp[5][5],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;
_settextposition(10,10);
offset[0]=initialoffset[0]*cosdat[index1];  /*  Levitation current calculation  */
offset[1]=initialoffset[0]*cosdat[index2];  /*  using the cosine array, which   */
offset[2]=initialoffset[0]*cosdat[index3];  /*  selected depending on the angle */
offset[3]=initialoffset[0]*cosdat[index4];  /*  of the model.  Initialoffset[0] */
offset[4]=initialoffset[0]*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(25,64);
     printf("%6.3f",yawangle);
```

```
}
/*************************************************************************/
/* Intrrupt service routine */
/* datin() obtains data from ADC in the array      */
void _interrupt _far datin(void)
{
   unsigned int y11,Outport;
   int k,df;
   for(k=0;k<=4;k++)
     {
          mode = 7;
          flag = 0;
          params[0] = k;
          params[1] = 0;
          mscl_das40 (&mode, params, &flag);
          if (flag != 0) process_error();
          y=params[0];
          vsensor[k]=(float)(y/409.5);
      }
_enable();
/*************************************************************************/
/*   decoupler      */
   v1[2]=vsensor[1]-5;
   v1[5]=((vsensor[3]+vsensor[4]-10)*-0.5);
   v1[8]=((vsensor[0]+vsensor[2]-10)*0.5);
   v1[11]=(vsensor[0]-vsensor[2]);
   v1[14]=(vsensor[3]-vsensor[4]);

   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,ADC_2     ;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();
}
/*-------------------------------------------------------------------------*/
/*************************************************************************/
void adcinitialize(void)
{
   /* DAS-40 Initialization  */
   mode=0;
   flag=0;
   params[0]=0;
   mscl_das40(&mode,params,&flag);
   if(flag !=0) process_error();
   _clearscreen(_GCLEARSCREEN);

}
/*************************************************************************/
void process_error()
{
   putch(7); putch(7);
   printf ("**** Error %u detected in mode %u      ", flag & 0xff, ((flag & 0xff00) >> 8));
   exit(1);
```

```
}
/*************************************************************************/
void acknowledge(void)
{
  _asm
     {
        push ax
        push dx
        mov dx,ADC_2     ;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
     }
}

/*************************************************************************/
void beactive(void)
{
 /* stay in memory*/
    printf("Keep Controller Active? y/n\n");
    ch=getch();
    if((ch=='Y') || (ch=='y'))
     {
       _asm
        {
             push ax;
              push dx ;
              mov ah,31h ;
              mov al,0;
              mov dx,50h;
             int 21h;
```

```
                pop dx;
                pop ax;
            }
        }
       else
        {
          _dos_setvect(intnum,oldnum);
          shutdown();
        }
}
/*********************************************************************/
/* getdat()  loads mixing matrix data*/

void getdat(void)
{
    char fname[12];
    int i,j,k,dim;
    FILE *fp;
    printf("Enter Data File Name");
    _settextposition(15,30);
    scanf("%s",&fname);
    if((fp=fopen(fname,"r+"))!=NULL)
    {
      printf("Please wait for data to load\n");
      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, hit Enter");
        getch();
         fclose(fp);
        }
          else
         fprintf(stderr,"Oops file error");
}
```