Spring 2005

Design and Implementation of One and Two-Degree-of-Freedom Magnetic Suspension and Balance Systems

Mark W. Adams
Old Dominion University

DESIGN AND IMPLEMENTATION OF ONE AND TWO-DEGREE-OF-FREEDOM MAGNETIC SUSPENSION AND BALANCE SYSTEMS

by
Mark W. Adams
BS, December 1996, Old Dominion University

A Thesis submitted to the faculty of Old Dominion University in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

ELECTRICAL ENGINEERING

OLD DOMINION UNIVERSITY
May 2005

Approved by:

Oscar R. González (Director)

W. Steven Gray (Member)

Colin P. Britcher (Member)

ABSTRACT

DESIGN AND IMPLEMENTATION OF ONE AND TWO-DEGREE-OF-FREEDOM MAGNETIC SUSPENSION AND BALANCE SYSTEMS

Mark W. Adams
Old Dominion University, 2005
Director: Dr. Oscar R. González

The main objectives of this research were to design and implement one and two-degree-of-freedom (1 DOF and 2-DOF) magnetic levitation systems to levitate permanent magnet cores contained in PVC pipes, 8.4 cm and 76.2 cm in length, respectively. This project used the components of a Magnetic Suspension and Balance System (MSBS) that is being built to provide obstruction free positioning of test models in six degrees of freedom (6-DOF) inside the Princeton University/Office of Naval Research High Reynolds Number Test Facility (HRTF). The HRTF, a specialized wind tunnel designed to simulate undersea conditions by creating a low-speed, 3500 PSI air environment, imposes design challenges unique to this MSBS. Among these challenges are the need to control magnetic flux densities through the two-inch thick stainless steel walls of the suspension chamber and to suspend a heavy test object for long periods due to the limited access to the chamber's interior.

The main design specifications were regulation of the vertical displacements of the suspended magnet and PVC pipe assemblies and a set point for pitch angle in the 2-DOF system. Laser beam position sensors were used to derive control feedback. The defining electromagnetic equations were reformulated into three-dimensional equations of motion according to well-known assumptions for large gap magnetic levitation systems. The resulting kinetic equations were converted to a linearized state-space model through a truncated Taylor series expansion. Using this modeling approach, a series of 1-DOF and 2-DOF controllers were designed and implemented using actual MSBS components configured outside the stainless steel suspension chamber. Moreover, the 2-DOF levitation was successfully repeated inside the actual suspension chamber with only a simple adaptation to the controller design. Through the demonstration of a scaled-down system incorporating all the essential dynamics and hardware of the full system, and the employment of the engineering techniques to construct and operate an actual working system, the paper argues from a practical standpoint for the completion of full-scale MSBS.

This thesis is dedicated to my family, whose dismay over the challenge to family life posed by my mid-life pursuit of a masters graduate degree was exceeded only by their perseverance, faith in Christ and their encouragement to me.

This is to acknowledge my advisor, Dr. Oscar González, an outstanding professor, engineer and mentor to his students. I heartily commend and thank him for his genuine care, patience and expertise in guiding me through this long process. Secondly, I wish to recognize the professionalism and patience of my committee as a whole.

TABLE OF CONTENTS

Page

Chapter

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

Figure Page

Figure Page

CHAPTER I

INTRODUCTION

Introduction

Magnetic levitation is the stable positioning of objects counter to gravity and other external forces without mechanical supports. It has been the subject of serious speculation and study at least from the early twentieth century when Dr. Robert Goddard, "Father of American Rocketry", described an idea for a magnetically levitated train. Since then many practical applications of magnetic levitation, maglev for short, have been developed in the areas of science, industry and medicine. Prototypes of maglev trains have been built and a thirty-kilometer commercial maglev line is running in Shanghai, China [1]. Both Dr. Goddard's concept then and maglev train designs today involve lifting train cars and propelling them along mechanical guide rails using magnetic fields. A more widespread use of maglev technology is the use of contact-less magnetic bearings in high-speed, near zero-gravity or extreme temperature conditions in which mechanical bearings would be impractical. Also, medical applications are becoming more widespread, where, for example, maglev technology is used for precision steering of catheters and probes inside the human body.

The examples above are non-trivial because the phenomenon of magnetic levitation opposes a fundamental characteristic in nature set forth in what is known as Earnshaw's Theorem. As noted in [2], this theorem states "that a group of charged particles governed by inverse square law forces cannot be in a stable equilibrium." According to Moon, this idea applies to the very devices employed in maglev systems, namely "a set of magnets and fixed circuits with constant current sources [2]."

This thesis presents the design and implementation of one and two-degree-of-freedom (1 and 2-DOF) magnetic levitation systems. The 1-DOF design specifications are to regulate the vertical position of a permanent magnet core within a 1.5 cm range centered at a point 17.8 cm below a single electromagnet. The permanent magnet core is housed in an 8.4 cm long PVC pipe and the entire suspended assembly weighs 192 g. The 2-DOF design specifications are to regulate the vertical displacement and pitch of a permanent magnet core housed in a 76.2 cm long PVC pipe

---

The journal model used for this thesis is the IEEE TRANSACTIONS and JOURNALS (April 2002)

with a total weight of 4.38 kg. The vertical displacement is regulated within a 1.5 cm range centered at a point 17.8 cm below a pair of electromagnets. The pitch angle is regulated over a range plus or minus one degree from the horizontal.

The primary significance of these experiments is that they use for the first time in closed-loop control the components of a Magnetic Suspension and Balancing System (MSBS) under development by Old Dominion University. The MSBS will be a large-gap magnetic levitation system intended to position test objects inside a specialized wind tunnel known as the High Reynolds Number Test Facility (HRTF). The HRTF, a project managed by Princeton University under a grant from the U.S. Office of Naval Research, is intended to provide a high pressure, low velocity environment suitable for testing submarine models. The constraints imposed by HRTF pose a unique set of challenges for the MSBS.

Much of the MSBS groundwork had been laid prior to the research supporting this thesis. This earlier work included derivation of mathematical models, validation of the models using computer simulations with realistic physical quantities, proofs of concept with laboratory components, procurement, fabrication and assembly of parts of the MSBS subsystems, prototyping of a graphical user interface (GUI) and testing of isolated subsystems (see [3], [4], [5] and [6]). In short, the framework for the MSBS design and physical configuration had been set up and partially validated.

Beginning in the spring of 2003, efforts have focused on the primary goal of this thesis: to demonstrate the practicality of building a full-scale MSBS through the successful implementation of 1-DOF and 2-DOF systems comprising actual MSBS components to be used in the HRTF.

A secondary goal for the thesis is to support future MSBS development by pulling together the heretofore disparate background references in a compact, integrated format. To provide future engineers a strong footing to advance the project, it's necessary to know the context in which the most recent work was done. For this reason, a considerable portion of this thesis is spent explaining the theoretical underpinnings of the MSBS and developing a framework for understanding the functional and physical aspects of the system. To understand the design constraints imposed by the HRTF, the next section describes the MSBS.

MSBS Design Goals and Physical Layout

This section presents an overview of the planned functionality and physical layout of the MSBS. The immediate purpose for this is to give the reader the background necessary to understand and appreciate the theoretical and practical aspects of the MSBS. Moreover, to help guide follow-on

MSBS development, this section also introduces functional and physical configuration baselines for the MSBS. As noted in the National Consensus Standard for Configuration Management, ANSI/EIA-649, these baselines provide at least a three-fold benefit: 1) they facilitate a common and more thorough understanding of the system's operation among all the parties involved in system development and use; 2) they give direction and boundaries to engineering efforts; and 3) they enable the impacts of design changes to be assessed more quickly and accurately [7]. First, the high-level MSBS functional goals and early design choices are explained in order to orient the reader to the MSBS. Following that the functional and physical baselines themselves are presented.

Overview of MSBS Design Goals

The MSBS is intended to accurately and reliably position a one meter long test object in six degrees of freedom inside a cylindrical steel chamber (part of the HRTF) under pressures up to 3500 PSI. The six degrees of freedom are commonly referred to as roll, pitch, yaw, vertical, lateral (left to right) translation and heave (forward and backward) motion. Fig. 1 illustrates the Princeton HRTF containing the MSBS test chamber, labeled "Section A".

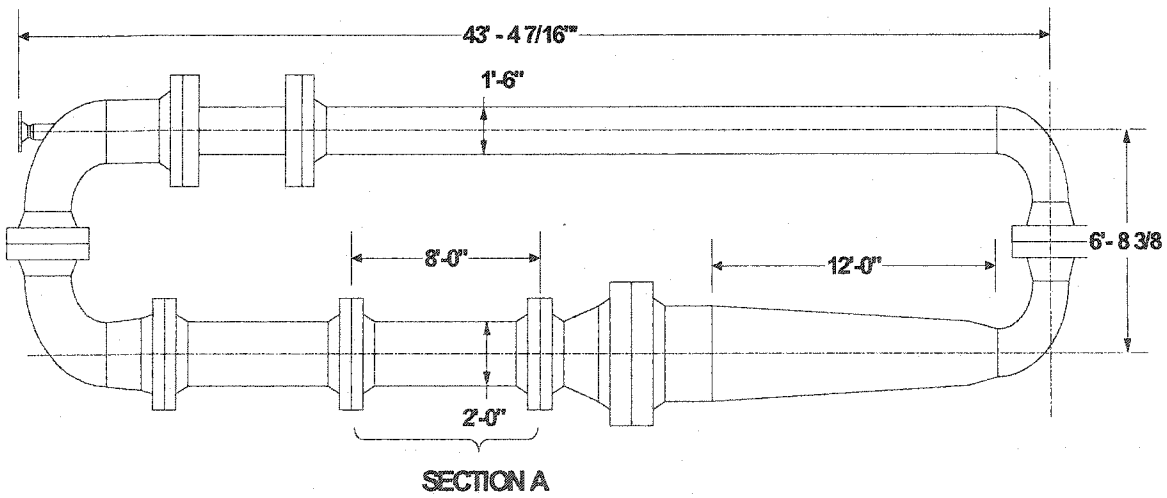


Fig. 1. Schematic of the Princeton University/ONR HRTF (see [3], p. 683, and [8]).

The high-pressure HRTF environment requires an MSBS test chamber constructed of highly sturdy, nonmagnetic material. This effectively constrains the MSBS diameter to about two feet due to the extraordinary fabrication costs for diameters much larger than that. Several designs were considered, but it was decided to construct the test chamber of two-inch thick stainless steel and place the electromagnets, or suspension coils, outside the chamber. Because the suspension coils were to be placed outside the chamber, the gap between the coils and the suspended test

object was larger than it would have been otherwise, requiring larger coils to allow effective control of magnetic fields. Large coils would in turn require large current sources and cooling systems. Finally, the high pressure environment would require the MSBS design to include pressure canisters for the delicate optical position sensors.

Fig. 2 is a schematic diagram showing the MSBS suspension chamber surrounded by a suite of ten electromagnets, or suspension coils. The four pairs of transversely mounted coils were custom-built to produce the lift force on the model, and the two coils wound coaxially around the ends of the chamber are intended to control the longitudinal movement of the model. The axially wound coils are not yet built.

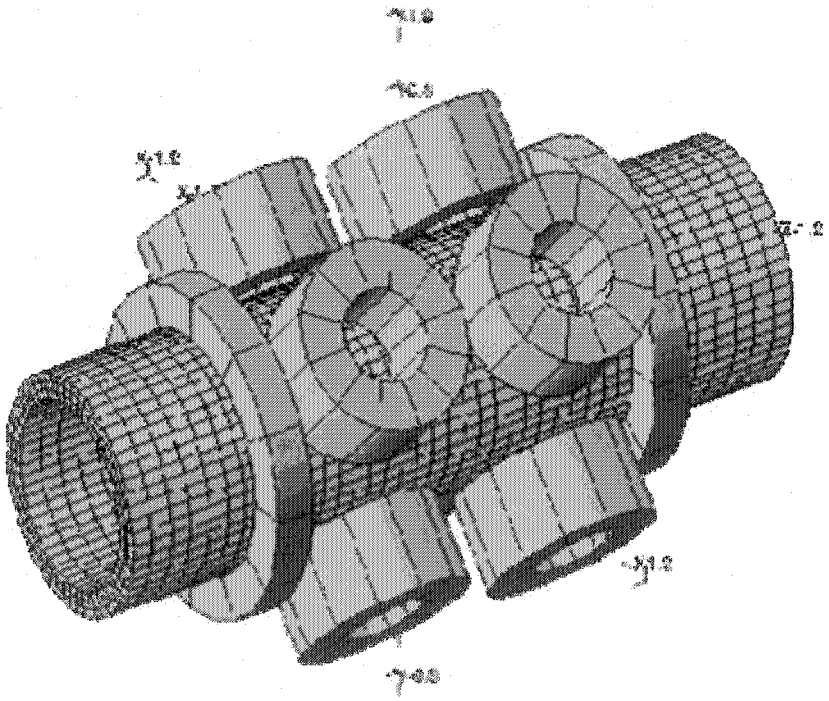


Fig. 2. Schematic of MSBS suspension chamber with electromagnets (see [9], p. 686).

Functional Baseline

A functional baseline, also called a requirements baseline in some references, formally defines the required high-level performance attributes for a system, normally in the form of a comprehensive set of authoritative documents that describe what the system must do [7]. The two essential elements of this baseline are a systematic enumeration of required functions and their authentication, and the baseline's chief purpose is to provide a "point of departure" for

future design changes [7]. The functional baseline in this thesis is a simple one in that it consists mainly of a tabulated set of concise descriptions of the major MSBS functions as determined from earlier MSBS theses and conference papers and reports. For the purpose of this thesis, the authentication of these functions is assumed and the source documents are not integrated into the baseline as described in the definition above.

Before proceeding to the actual functional baseline, it is noted that not all the lower-level MSBS performance specifications are uniquely or straightforwardly derived from the high-level design goals. For example, the minimum number of suspension coils needed to implement six degree-of-freedom control can be derived directly from the electromagnetic theory, but the settling time allowable in the controller represents the engineer's best judgment in the design. Second, some of the MSBS design choices were based on equipment choices made in response to budget or time constraints, rather than directly from specific high-level requirements. For example, the controller's range is somewhat constrained by the particular position sensors selected. Third, most of the MSBS requirements were identified and the design choices made well before the work on this paper was started, as reported in [3]. (Additional work is presented in [4], [5] and [9].) As a result, the MSBS baselines presented here should not be construed as exclusive of other designs. Nevertheless, the functional and physical baselines do perform their most essential function, which is to provide a useful point of departure for design improvements [7].

Table I is an abbreviated functional baseline for the current MSBS design. (Appendix A includes a more detailed version.) In addition to a hierarchical listing of requirements statements, the table also lists certain design decisions which correspond to these requirements. Design decisions are the particular choices of algorithms, methodologies or equipment types determined to be the best or most cost-effective way to implement a functional requirement. Design decisions are not "must do" statements although, once adopted, they can place constraints on requirements and other design decisions.

TABLE I
ABBREVIATED MSBS FUNCTIONAL CONFIGURATION BASELINE

Identifier	Functional Requirements	Related Design Decisions
FR1000	Operate within the environmental and operational constraints of the HRTF	----------
FR1100	Operate under 3500 PSI internal pressure.	Two-inch, nonmagnetic stainless steel chosen for construction of suspension chamber. Nonmagnetic material needed to avoid interference with MSBS operation. Materials costs constrain inner diameter to approx. 19 in.
FR1200	Maintain airtight seal with HRTF under 3500 PSI internal pressure.	----------
FR1300	Support Reynolds number in range of researcher's needs.	Cylindrical and uniform construction. Actuators mounted externally. Level of effort required to open suspension chamber limits accessibility.
FR1310	Employ test object supporting high Reynolds numbers.	Baseline design for test object calls for an ellipsoidal model with 12:1 length to diameter ratio.
FR1400	Operate unattended and continuously for periods of several days without interior access to HRTF.	Robust sensor scheme selected. (See FR2210) Linear design selected for controller. (See FR2210)
FR1410	Accommodate changes in pressure, airspeed and test object position commands without manual intervention.	Choose a conservative range of movement with respect to system's physical capabilities.
FR1420	Allow for calibration and testing of interior components without manual intervention.	Control program designed to compensate for changes in sensor input/output characteristics.
FR1430	Maintain safe internal temperature in electro-magnets for long periods of unattended operation.	Electromagnets wound with insulated copper tubing with square cross-section. Circulating, chilled water cooling system designed.
FR1440	Employ automated condition monitoring and emergency system shut-down.	----------

TABLE I, CONT'D

Identifier	Functional Requirements	Related Design Decisions
FR2000	Automatically control position of suspended test object in six-degrees-of-freedom without direct mechanical contact.	----------
FR2100	Actuate suspended test object in six degrees of freedom using electromagnets.	----------
FR2110	Employ minimum of five independently driven actuators (electromagnets).	Five separately controlled current amplifiers capable of supplying up to 120A DC at 150 V DC.
FR2120	Produce aggregate flux density of sufficient magnitude to create force adequate to counter both weight of test object and the aerodynamic forces acting on it.	Large, iron-core electromagnets custom designed and built for MSBS. Drive current anticipated to be up to 120 DC amps. So-called X configuration for electromagnets selected.
FR2200	Sense and provide test object position without direct mechanical contact.	Optical sensors selected for design.
FR2210	Measure deviation of test object from fixed point in an inertial coordinate system attached to the suspension chamber.	Selected sensors were SunX LA-511 optical sensors with 15-mm range of detectable motion.
FR2300	Execute real-time, multivariable control in a way that HRTF users to readily change parameters and data capture schemes.	----------
FR2310	Provide rapid prototyping.	dSPACE Controller Design and Implementation system selected to permit both design and operation on same hardware/software suite.
FR3000	Provide real-time control, monitoring and data logging to MSBS operators.	----------

Physical Baseline Description

The physical baseline description in this thesis combines elements of two common configuration management concepts: the product configuration baseline and the product structure. The product configuration baseline, or just product baseline, is defined as the set of all documents needed to completely describe a system's functional and physical attributes, usually just prior to the production phase. In other words, the product baseline contains all the drawings, specifications, etc. needed to build the system [7]. The product structure is a method for laying out, in

hierarchical form, the composition of a product in terms of structural components, equipment type and quantity, software versions, interfaces, etc. The product structure makes reference to but does not necessarily include the design documentation [7]. The physical baseline used in this thesis is much like the product structure except that it is not replete with references to design documentation. In addition, the physical baseline used here makes reference to all the source documents available for the MSBS, though does not integrate them as a formal product baseline would.

Table II is an abbreviated physical baseline showing the highest level hardware and software components with brief descriptions. Appendix A contains a more detailed physical baseline description with references to the source documents describing these items.

TABLE II
ABBREVIATED MSBS PHYSICAL BASELINE

Group	Identifier	Item Name	Qty	Item Description
MSBS Plant Group	1A1	Suspension chamber	1	Type 304L stainless steel cylindrical suspension chamber, nom. 24 in. O.D., nom. 19 in. I.D., 96 in. L.
	1A1A1	Radial coil mounting assembly	1	Structural fiberglass frame used to mount radial suspension coils in X-configuration around suspension chamber
	1A1A2	Radial suspension coil	8	Custom built, iron-core magnets
	1A1A3	Axial suspension coil	2	Water-cooled magnets wound around suspension chamber (axially).
	1A2	Position sensor assy	1	Position sensors and mounting structure.
	1A2A1	Sensor mounting frame	1	Non-magnetic frame holding position sensors in configuration to sense movement in 5 degrees-of-freedom.
MSBS Plant Group	1A2A2	Laser beam sensor set	5	SunX Trading Co., Model LA-511
	1A3	Magnet cooling system		Chilled water circulation system.
	1A4 [a]	Test model assy. no. 3	1	Cylindrical PVC test model,30 in. L, 3.5 in. dia
	1A4A1	Permanent magnet core	6	Neodymium-Iron-Boron disks, 3.0 in. dia., 0.5 in. thickness.

TABLE II , CONT'D

Group	Identifier	Item Name	Qty	Item Description
Amplifier Group	2A1	Electromagnet Drive Current Amplifier	6	Copley Controls Corp Model 232P PWM Power Amplifier
	2A2A1	High voltage/high current power supply	1	Clinton Model S600/150S
Control System Group	3A1	Host Computer	1	Dell Precision PWS330, Intel Pentium 4 CPU, 523 KB RAM
	3A2	Real Time Interface	1	DS-1103 Real Time Interface
	3A3	Signal interface panel	1	CP-1103 Interconnection Module
Software Group	4A1	Host Computer Operating System	1	MS Windows 2000, v 5.00.2195 Service Pack 1
	4A2	Matlab Software Suite	1	Control system design software.
	4A2A1	Matlab R12.1 (inc. Simulink)	1	Matlab R12.1 (inc. Simulink)
	4A2A2	Matlab Control System Tool Box, v5.1	1	Matlab Control System Tool Box, v5.1

[a] Test model presented here is the last one used in the set of experiments covered in this thesis and the one that most closely represents the model anticipated during actual operations.

Fig. 3 illustrates the interrelationships among the major MSBS components listed in TABLE II. The legend identifies the four major groupings within the physical baseline.

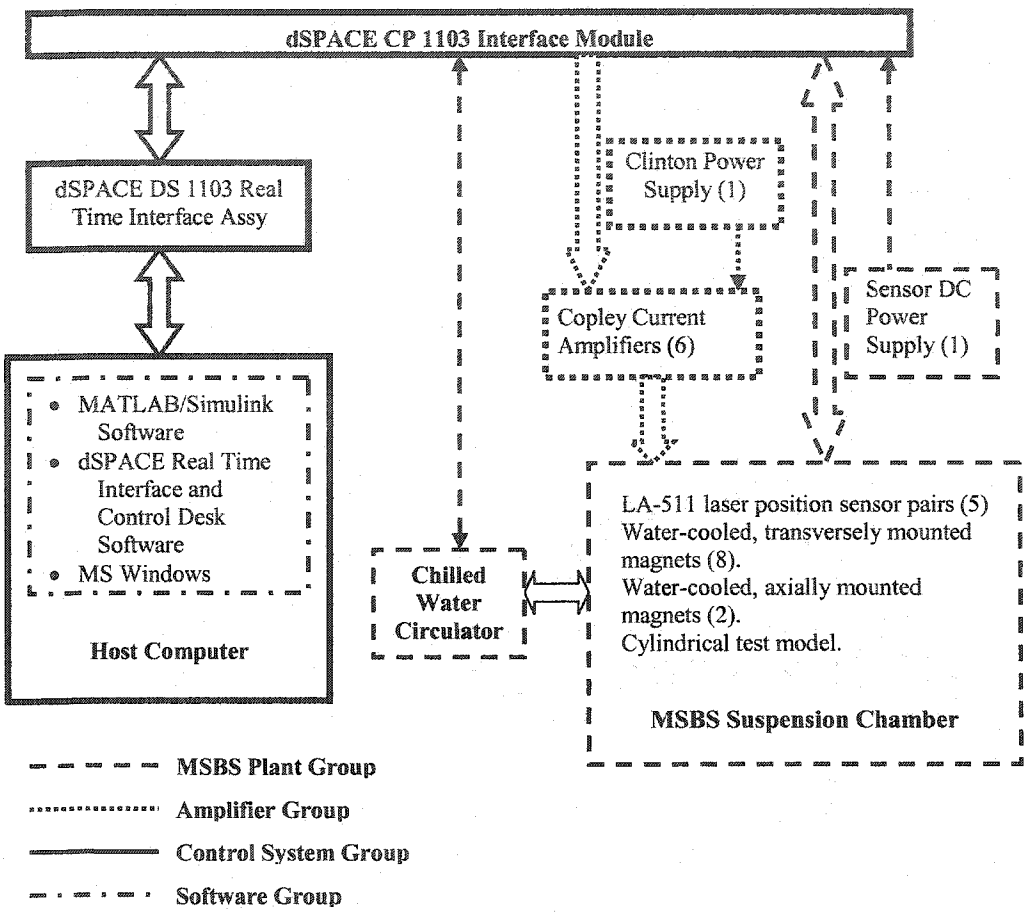


Fig. 3. High-level MSBS physical configuration diagram.

Document Layout

To meet its objectives, the document is laid out in the following chapters:

CHAPTER I INTRODUCTION. This chapter introduces the MSBS and its major design challenges and provides breakdowns of the system's functional and physical attributes.

CHAPTER II MSBS THEORETICAL BACKGROUND. This chapter explains the electromagnetic and control systems theory needed to design the MSB control system.

CHAPTER III GENERAL IMPLEMENTATION METHODS. This chapter provides a background for transforming the mathematical model into a physical controller by introducing the use of certain components in the MSBS software and hardware suite.

CHAPTER IV ONE-DEGREE-OF-FREEDOM TESTS. This chapter explains the role of the one-degree-of-freedom tests in developing the full MSBS and describes how the tests were conducted and the importance of their results.

CHAPTER V APPLICATION OF OPTIMAL CONTROL THEORY. This chapter shows how optimal control theory is applied to the design of a state regulator for the MSBS.

CHAPTER VI TWO-DEGREE-OF-FREEDOM EXPERIMENTS. This chapter continues the previous description by discussing how optimal control is applied to the multiple input, multiple output (MIMO) case, in particular the two-degree-of-freedom configuration of the MSBS.

CHAPTER VII CONCLUSION. This chapter provides a broad summary of the document and highlights key points in the remaining MSBS development work.

APPENDICES. These contain technical references such as equipment descriptions and drawings, operating procedures, experimental data and Matlab scripts.

CHAPTER II

MSBS THEORETICAL BACKGROUND

Introduction

As pointed out in Chapter I, the electromagnetic theory and mathematical equations needed to design the MSBS have long been identified and understood, if not fully implemented. The purpose of this chapter is to introduce the underlying physics and mathematics in enough detail to enable the reader to understand the development effort covered in this thesis and to pursue the MSBS design and construction to its completion.

The motivation for a separate treatment of the theoretical background lies in the fact that the most recent MSBS design documentation, [4], [5], [6] and [9], as well as the maglev work on which that paper was primarily based, [10] through [13], covered the MSBS theoretical background only in varying degrees, according to the documents' particular individual emphases. None of the references contained a broad-scoped development of the basic MSBS equations, beginning with a derivation of the electromagnetic force and torque equations and proceeding through to the derivation a linear model. However, it was just this kind of background that was needed to bring the MSBS development forward—for at least two reasons. First, despite the prior successful computer simulation of a 5-DOF linear quadratic regulator [4], to actually build the MSBS in full scale, several new interim controller designs were needed, each design requiring the 5-DOF linear model to be adapted to smaller scale versions of the final system. Second, understanding the design issues of robust stability and performance requires knowledge of the physical characteristics of the MSBS plant.

A final note is needed before proceeding to the development of the MSBS theoretical background. Whereas the goal of the MSBS project is to build the 6-DOF system described by the functional and physical configurations described in the previous chapter, the remainder of this paper primarily addresses modeling a 5-DOF system and the design and implementation of 1-DOF and 2-DOF systems. The reason is two-fold. First, controlling a 6-DOF system requires a slightly larger and more complicated mathematical model [12]. Second, detecting the rolling motion of cylinder around its long axis would be difficult with the optical shadow sensors selected for this design. Some kind of fin would be needed that changed cross section in the sensor beam, yet at this point in the project it is unknown what the HRTF design constraints will allow. Therefore, it was opted to forgo the more complicated design and prove the maglev concepts and techniques

using the same basic electromagnetic configuration of the full 6-DOF system but with a 5-DOF controller design.

This chapter briefly explains the electromagnetic principles underlying the MSBS operation, derives the equations of motion describing the basic MSBS dynamics, then develops a general five-degree-of-freedom (5-DOF) linear model for a five-coil MSBS plant and finally introduces other system dynamics that should be considered in the control system design.

Electromagnetic Theory

The basis for the MSBS is the interaction of one or more regulated magnetic fields produced by a fixed array of electromagnets with the constant magnetic field of a smaller, moveable permanent magnet core inside the suspended test object. This interaction produces forces and torques that move the core relative to the fixed magnet array. If the current in the electromagnets can be regulated so the resulting forces and torques suspend the core in stable equilibrium, then the basic MSBS design goals have been met. The start of the design process is then to obtain equations describing the electromagnetic forces and torques. This section develops these equations, which are based on Maxwell's equations for magnetic fields.

Reference Frames and Assumptions

Before the MSBS electromagnetic equations can be derived, two rectangular coordinate systems, or reference frames must be defined, as illustrated in Fig. 4. The first is the inertial reference frame, which contains the fixed array of electromagnets and the suspension equilibrium point, located at the reference frame's origin. The second reference frame is the one attached to the suspended object, referred to as the body coordinate system. The origin of this system is the centroid of the suspended core, which for convenience is modeled as a cylinder. The two coordinate systems are distinguished by using an overbar on the axis labels to denote the body coordinate system. When the suspended object is at equilibrium, the two coordinate systems are coincident, as pictured in Fig. 4.

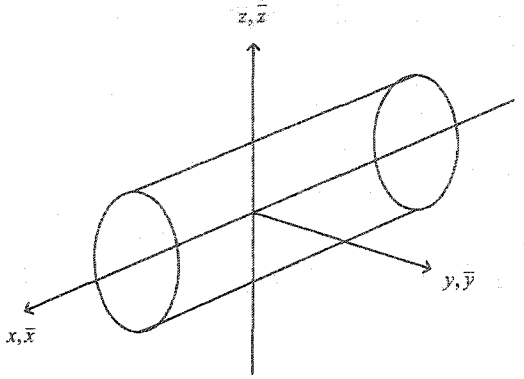


Fig. 4. Cylindrical permanent magnet in inertial and body coordinate systems.

Several assumptions concerning the electromagnetic environment of the MSBS are essential to constructing the MSBS linear model. The first assumption, as stated in [11], is that "The torques and forces on a magnetic dipole in a steady magnetic field are identical to those on an infinitesimal current loop with the same magnetic moment." This, along with the fact that a cylindrical permanent magnet approximates a closely wound solenoid whose magnetic dipole is simply the number of turns times the magnetic dipole of each turn—in a uniform magnetic field [15]—allows the fundamental torque and force equations for an infinitesimal loop to be extended directly to a permanent magnet in the MSBS.

The second assumption is that the MSBS is in the class of so-called "large gap" suspension systems, where the volume of the permanent magnet core is small relative to the volume of the electromagnets and small relative to the gap between the core and the electromagnets. For these systems the magnetic flux densities can be considered uniform over the volume of the suspended permanent magnet core, which simplifies the force and torque equations [10]. This assumption was successfully applied in several working large-gap systems, including an earlier proof of concept demonstration of the MSBS [1], [16]. It should be noted that prior to the research supporting this thesis, the use of this assumption had not been validated in a system approximating the dimensions of the MSBS.

The third assumption is that the magnetic flux densities produced by the electromagnets, or suspension coils, can be suitably approximated by second-order Taylor series expansions about the centroid of the core at its equilibrium point. As a result, the values of the magnetic flux densities and their gradients need only be calculated (or measured) at the suspension equilibrium point. This assumption has been validated in practical systems of different configurations [11].

The fourth primary assumption is that time invariant, or magnetostatic equations can be used to reasonably approximate the electromagnetic environment in the MSBS. This assumption is itself based on two others: 1) the magnetic field variations in the MSBS are not rapid enough to require a time varying model, and 2) the movements of the suspended core are slow enough to consider the object static. The importance of this last assumption is shown by first considering Maxwell's equation for a time varying magnetic field

$$\nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t},$$

where $\mathbf{E}$ is the electric field produced by the time-varying magnetic flux density $\mathbf{B}$. Taking the surface integral of both sides of this equation and applying Stokes theorem[1] to the left hand side gives

$$\oint_c \mathbf{E} \cdot dl = -\int_S \frac{\partial \mathbf{B}}{\partial t} \cdot d\mathbf{s},$$

which shows that a time rate of change in a magnetic field induces an electric field.

Next, consider a moving charge $\mathbf{q}$ with velocity vector $\mathbf{u}$ in a steady magnetic field. The moving charge induces a electric field in the conductor [17]. Thus, both a moving conductor in a static magnetic field and a time-varying magnetic field cause an electric field to be present in a conductor and Faraday's law of electromagnetic induction applies,

$$\oint_C \mathbf{E} \cdot dl = -\int_S \frac{\partial \mathbf{B}}{\partial t} \cdot d\mathbf{s} + \oint_C (\mathbf{u} \times \mathbf{B}) \cdot dl, \tag{1}$$

where $\mathbf{E}$ and $\mathbf{B}$ are electric field, magnetic flux density vectors, respectively, and $\mathbf{u}$ is the velocity vector of a moving conductor. The role of an induced electric field in the development of MSBS forces can be seen in Lorentz's force equation

$$\mathbf{F} = q(\mathbf{E} + \mathbf{u} \times \mathbf{B}), \tag{2}$$

where $\mathbf{F}$ is the force vector and q and $\mathbf{u}$ are the moving charge and its velocity vector. In light of (2), both electric and magnetic force components must be considered whenever conditions exist that produce non-zero terms on the right hand side of (1). However, when a magnetostatic model is assumed, i.e., $\mathbf{B}$ is not time varying and the permanent magnet is static, the force equation reduces to

[1] Stokes theorem states the surface integral of the curl of a vector field is equal to the line integral of that vector field around the closed contour bounding that surface.

$$\mathbf{F} = q(\mathbf{u} \times \mathbf{B}), \tag{3}$$

a much simpler starting point for developing the electromagnetic torque and force equations.

Electromagnetic Torque and Force Equations

Following is a development of the torque and force equations needed to model the MSBS operation. Consider a magnetostatic condition as described above where a constant magnetic field **B** acts on a plane circular loop with cross-sectional area A carrying current **I**. The simplified Lorentz' force equation (3) describes force on an electric charge moving in that loop. This equation can be adapted as in [17] to express just the force on an incremental portion of the loop as follows:

$$d\mathbf{F} = \mathbf{I}dl \times \mathbf{B}.$$

Because of the loop's symmetry, the net force on the entire loop is zero. However, a torque **T** is produced that tends to rotate the loop until the plane of the loop is normal to the **B** field. The torque can be calculated by

$$\mathbf{T} = \mathbf{I}\oint_{Loop} \mathbf{r} \times (dl \times \mathbf{B}),$$

where **r** is the position vector of the differential loop element, *dl*, with respect to the origin of the of the inertial reference frame. Using the vector identity $A \times (B \times C) = B(A \cdot C) - C(A \cdot B)$, the torque can then be expressed as

$$\mathbf{T} = I\oint_{Loop} \mathbf{dl}(\mathbf{r} \cdot \mathbf{B}) - I\oint_{Loop} \mathbf{B}(\mathbf{r} \cdot \mathbf{dl}). \tag{4}$$

Applying Stokes theorem and using a result from [11], the two integrals in (4) become

$$\mathbf{T} = I\int_S \mathbf{dA} \times \nabla(\mathbf{r} \cdot \mathbf{B}) - I\mathbf{B}\int_S (\nabla \times \mathbf{r}) \cdot \mathbf{dA},$$

where **dA** is a vector normal to the differential area with magnitude equal to the area. The right hand integral equals zero because the curl of the position vector is zero.[2] Also, if **B** is constant

[2] If taken from the coordinate system origin, the position vector **r** is really the gradient of a scalar field, i.e., the position of an incremental portion of the current loop; hence $\nabla \times \mathbf{r} = 0$, since the curl of the gradient of any scalar field is identically zero.

across the current loop, then $\nabla(\mathbf{r}\cdot\mathbf{B})=\mathbf{B}$ [3], and the basic equation for torque on an infinitesimal loop situated in a magnetic flux density field reduces to

$$\mathbf{T}=I\int_S d\mathbf{A}\times\mathbf{B}=I\mathbf{A}\times\mathbf{B}. \tag{5}$$

If **A** in the equation above is taken to be the area of an infinitesimal loop, the quantity IA is the loop's magnetic dipole moment, **m**, in units A-m^2. Then (4) becomes

$$\mathbf{T}=\mathbf{m}\times\mathbf{B}. \tag{6}$$

The equation for magnetic force is derived from the above torque equation. Consider when the magnetic dipole moment **m** of an infinitesimal loop lies at an angle θ to the flux density field **B.** (Note, for an infinitesimal loop there is no requirement for **B** to be uniform.) The magnitude of the torque on the loop is given by

$$T=mB\sin\theta.$$

An incremental change in θ involves a change in torque and thus a change in the loop's potential energy as described in

$$dU=dW=dT=mB\sin\theta d\theta, \tag{7}$$

where dU, dW and dT are the magnitude changes in potential energy, work and torque, respectively. The potential energy in the loop can be found by integrating (7) and is given by

$$U=-mB\cos\theta=-\mathbf{m}\bullet\mathbf{B}. \tag{8}$$

If the value of U is taken to be zero when the loop is has its maximum potential energy, i.e., when it is oriented such that **m** and **B** are perpendicular, then the negative sign properly describes the loop's potential energy as θ decreases.

The force producing the torque on the infinitesimal loop is derived from the above expression for potential energy. Consider a positive force displacing the infinitesimal current loop from its position of maximum potential energy toward its position of zero potential energy, or equilibrium position. The incremental work done in moving the loop causes a decrease in potential energy equal to

$$dW=-dU=\mathbf{F}\bullet d\mathbf{r}=-\nabla U\bullet d\mathbf{r},$$

[3] The gradient of the dot product of the position vector and B field is given by

$$\left[\mathbf{a}_x\frac{\partial}{dx}+\mathbf{a}_y\frac{\partial}{\partial y}+\mathbf{a}_z\frac{\partial}{\partial z}\right]\left(xB_x+yB_y+zB_z\right)=\mathbf{a}_xB_x+\mathbf{a}_yB_y+\mathbf{a}_zB_z=\mathbf{B}.$$

which when considering (8), implies

$$\mathbf{F} = -\nabla U = \nabla(\mathbf{m} \bullet \mathbf{B}). \tag{9}$$

Expanding the right hand side of equation (7) yields

$$\nabla(\mathbf{m} \bullet \mathbf{B}) = \mathbf{m} \times (\nabla \times \mathbf{B}) + (\mathbf{m} \bullet \nabla)\mathbf{B} + \mathbf{B} \times (\nabla \times \mathbf{m}) + (\mathbf{B} \bullet \nabla)\mathbf{m}.$$

All terms on the right hand side go to zero except $(\mathbf{m} \bullet \nabla)\mathbf{B}$ [11] [4], allowing the equation for magnetic force on an infinitesimal loop (9) to be expressed as

$$\mathbf{F} = (\mathbf{m} \bullet \nabla)\mathbf{B}. \tag{10}$$

Next the torque and force equations for a cylindrical permanent magnet are derived in the form they will be used in the MSBS model. Consider the vector quantity **M**, the infinitesimal current loop's magnetic moment density. For a magnetic moment produced in an incremental volume,

$$\mathbf{M} = \partial \mathbf{m} / \partial v, \text{ or}$$

$$\mathbf{m} = \int_V \mathbf{M} dv,$$

where the latter expression is noted in [11]. From equations (6) and (10), the differential torque and force per unit volume can then be expressed

$$\partial \mathbf{T} = (\mathbf{M} \times \mathbf{B}) \partial v \tag{11}$$

and

$$\partial \mathbf{F} = (\mathbf{M} \bullet \nabla)\mathbf{B} \partial v. \tag{12}$$

Consider next a set of N identical coaxial current loops, i.e., a solenoid. The torque, magnetic moment, and magnetization on such a solenoid in a uniform magnetic field are simply equal to N times those quantities for a single loop. Recall the assumption a permanent magnet with the same magnetization behaves like a solenoid in the same magnetic field [15]. Thus the formulae for torques and forces for a current loop can be extended to the cylindrical permanent magnet in the MSBS.

[4] From Maxwell's equations for a non-time varying field, the curl of the **B** field is proportional to the free current density, which is zero inside the permanent magnet core. Also from Maxwell's equations, the divergence of **B** is zero. Finally, the curl of the magnetic dipole moment is zero because **m** represents a curl-free vector.

In equation (2) of [11], the total torque on a cylindrical permanent magnet is given by

$$\bar{\mathbf{T}} = \int_V [\partial\bar{\mathbf{T}} + (\bar{\mathbf{r}} \times \partial\bar{\mathbf{F}})] dv \,, \tag{13}$$

where the two terms in the integral indicate two components of torque. The first term $\partial\bar{\mathbf{T}}$ is the incremental torque derived from the interaction of the **B**-field with the magnetic dipole moment **m** in an incremental volume, as in (11). The second term in (13) represents the torque produced by the Lorentz force on an incremental volume located at a position $\bar{\mathbf{r}}$ in the body coordinate system as shown in Fig. 5. Substituting equations (11) and (12) in (13), gives

$$\bar{\mathbf{T}} = \int_V \{(\bar{\mathbf{M}} \times \tilde{\bar{\mathbf{B}}}) + [\bar{\mathbf{r}} \times (\bar{\mathbf{M}} \bullet \nabla)\tilde{\bar{\mathbf{B}}})]\} dv \,. \tag{14}$$

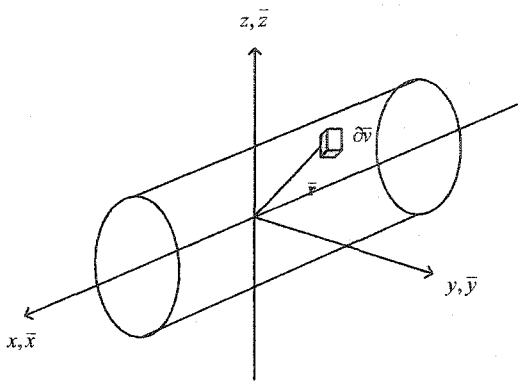


Fig. 5. Illustration of an incremental volume within the permanent magnet core.

A new notation is introduced in equation (14) above. The ~ symbol over the variable is a short hand means of denoting a truncated Taylor series expansion. For example, the magnetic flux density at a given point within the suspended magnet in body coordinates, $\tilde{\bar{\mathbf{B}}}$, is given by

$$\tilde{\bar{\mathbf{B}}} = \bar{\mathbf{B}} + \bar{\mathbf{r}}_x \frac{\partial B}{\partial x} + \bar{\mathbf{r}}_y \frac{\partial B}{\partial y} + \bar{\mathbf{r}}_z \frac{\partial B}{\partial z} + \frac{1}{2}\bar{\mathbf{r}}_x \frac{\partial^2 B}{\partial x^2} + \frac{1}{2}\bar{\mathbf{r}}_y \frac{\partial^2 B}{\partial y^2} + \frac{1}{2}\bar{\mathbf{r}}_z \frac{\partial^2 B}{\partial z^2} \,. \tag{15}$$

The equations for the total force on the cylindrical magnet can be found by integrating the term for force in the integrand in (14) as follows:

$$\bar{\mathbf{F}} = \int_V (\bar{\mathbf{M}} \bullet \nabla)\tilde{\mathbf{B}} dv \,. \tag{16}$$

The expression $(\bar{\mathbf{M}} \bullet \nabla)\tilde{\mathbf{B}}$ can also be written $[\partial\tilde{\mathbf{B}}]\bar{\mathbf{M}}$, where

$$\partial\tilde{\mathbf{B}} = \begin{bmatrix} \tilde{B}_{\bar{x}\bar{x}} & \tilde{B}_{\bar{x}\bar{y}} & \tilde{B}_{\bar{x}\bar{z}} \\ \tilde{B}_{\bar{y}\bar{x}} & \tilde{B}_{\bar{y}\bar{y}} & \tilde{B}_{\bar{y}\bar{z}} \\ \tilde{B}_{\bar{z}\bar{x}} & \tilde{B}_{\bar{z}\bar{y}} & \tilde{B}_{\bar{z}\bar{z}} \end{bmatrix} \tag{17}$$

and

$$\bar{\mathbf{M}} = \begin{bmatrix} M_{\bar{x}} \\ M_{\bar{y}} \\ M_{\bar{z}} \end{bmatrix}. \tag{18}$$

Equation (18) is noted in [11]. In the planned MSBS configuration, the magnetization vector is coincident with the long axis of the suspended permanent magnet and oriented in the positive $\bar{x}$ direction. Accordingly, $M_{\bar{x}}$ is the only non-zero component of **M**, which leads to the expression

$$\left[\partial\tilde{\mathbf{B}}\right]\bar{\mathbf{M}} = M_{\bar{x}} \begin{bmatrix} \tilde{B}_{\bar{x}\bar{x}} \\ \tilde{B}_{\bar{x}\bar{y}} \\ \tilde{B}_{\bar{x}\bar{z}} \end{bmatrix}. \tag{19}$$

Therefore, if the **B**-field is uniform throughout the volume of the core, the equations for force components are

$$F_{\bar{x}} = M_{\bar{x}} \int_v \tilde{B}_{\bar{x}\bar{x}} dv = M_{\bar{x}} \int_v \left(B_{\bar{x}\bar{x}} + B_{\bar{x}\bar{x}\bar{x}}\bar{x} + B_{\bar{x}\bar{x}\bar{y}}\bar{y} + B_{\bar{x}\bar{x}\bar{z}}\bar{z} \right) dv, \tag{20}$$

$$F_{\bar{y}} = M_{\bar{x}} \int_v \tilde{B}_{\bar{x}\bar{y}} dv = M_{\bar{x}} \int_v \left(B_{\bar{x}\bar{y}} + B_{\bar{x}\bar{y}\bar{x}}\bar{x} + B_{\bar{x}\bar{y}\bar{y}}\bar{y} + B_{\bar{x}\bar{y}\bar{z}}\bar{z} \right) dv, \tag{21}$$

$$F_{\bar{z}} = M_{\bar{x}} \int_v \tilde{B}_{\bar{x}\bar{z}} dv = M_{\bar{x}} \int_v \left(B_{\bar{x}\bar{z}} + B_{\bar{x}\bar{z}\bar{x}}\bar{x} + B_{\bar{x}\bar{z}\bar{y}}\bar{y} + B_{\bar{x}\bar{z}\bar{z}}\bar{z} \right) dv, \tag{22}$$

where the **B** terms with multiple subscripts are the first and second order gradients with respect to the axes of the body coordinate system and v is the volume of the core. Because of the symmetry of the cylindrical core, and the assumption of uniform fields and gradients within the core (assumption number two explained at the beginning of this section), the position terms ($\bar{x}$, $\bar{y}$ and $\bar{z}$) integrate to zero, and based on [11], (20) through (22) reduce to

$$F_{\bar{x}} = vM_{\bar{x}}B_{\bar{x}\bar{x}} \tag{23}$$

$$F_{\bar{y}} = vM_{\bar{x}}B_{\bar{x}\bar{y}} \tag{24}$$

and

$$F_{\bar{z}} = vM_{\bar{x}}B_{\bar{x}\bar{z}}. \tag{25}$$

Finally, the equation for torque on the cylindrical core is derived from (14). Employing the Taylor series expansion as in (15), from [11] the equations for the components of torque are

$$T_{\bar{x}} = 0 + M_{\bar{x}}\int_v (\tilde{B}_{\bar{x}\bar{z}}\bar{y} - \tilde{B}_{\bar{x}\bar{y}}\bar{z})dv \tag{26}$$

$$T_{\bar{y}} = -M_{\bar{x}}\int_v \tilde{B}_{\bar{z}} + M_{\bar{x}}\int_v (\tilde{B}_{\bar{x}\bar{x}}\bar{z} - \tilde{B}_{\bar{x}\bar{z}}\bar{x})dv \tag{27}$$

$$T_{\bar{z}} = M_{\bar{x}}\int_v \tilde{B}_{\bar{y}} + M_{\bar{x}}\int_v (\tilde{B}_{\bar{x}\bar{y}}\bar{x} - \tilde{B}_{\bar{x}\bar{x}}\bar{y})dv. \tag{28}$$

Note the overbars indicating the core's magnetization is defined in the body coordinate system, whereas the magnetic flux density is defined in the inertial reference frame. When expanded, the integrals in the above equation are fairly complex. For example, the first term under the integral in (26) becomes

$$T_{\bar{x}} = M_{\bar{x}}\int_v (B_{\bar{x}\bar{z}}\bar{y} + B_{\bar{x}\bar{z}x}\overline{yx} + B_{\bar{x}\bar{z}y}\bar{y}^2 + B_{\bar{x}\bar{z}z}\overline{yz})dv.$$

However, just as in the force equations (20) through (22), the position terms integrate to zero. Furthermore, subsequent developments of the torque and force equations in [12] and [14] show that due to symmetry, certain the second order field gradients terms are negligible. Consequently, for a uniform magnetization throughout the core, the torque equations used to create the MSBS model can be approximated by the much simpler equations:

$$T_x = 0, \tag{29}$$

$$T_{\bar{y}} = -vM_{\bar{x}}B_{\bar{z}} \text{ and} \tag{30}$$

$$T_z = vM_{\bar{x}}B_{\bar{y}}. \tag{31}$$

Equations of Motion

This section uses the torque and force equations just derived to develop equations of motion suitable for modeling the MSBS. The equations specifically of interest are the ones describing the acceleration and velocity of the suspended object in the five controllable degrees of freedom, as illustrated in Fig. 6. These equations are complicated by the fact the MSBS torques and forces arise from fields originating in two different reference frames, or coordinate systems: 1) the

inertial reference frame, where the controlling magnetic fields originate and the control feedback signals are generated; and 2) the body reference frame in which the magnetization vector is defined. For this reason, two coordinate system transformations are employed in this section to develop the kinematic equations for the MSBS, the Euler rate transformation for angular rates and the inertial-to-body coordinate system transformation.

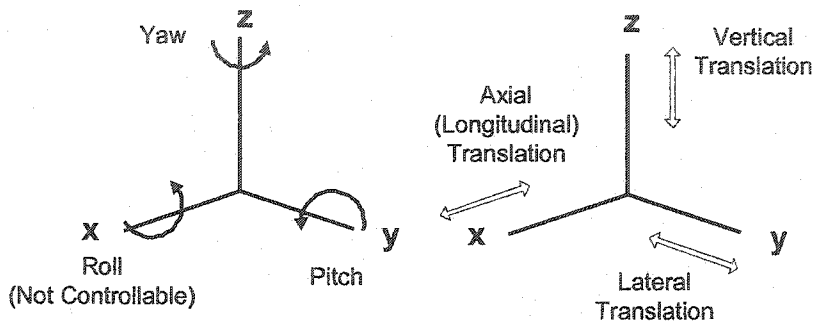


Fig. 6. Illustration of six-degrees-of-freedom in the MSBS coordinate system.

The angular acceleration of a rigid body in the body's own coordinate system can be expressed

$$\dot{\bar{\Omega}} = \mathbf{I}_c^{-1}\bar{\mathbf{T}}, \tag{32}$$

where $\dot{\bar{\Omega}}$ is a vector of angular acceleration components, $\mathbf{I}_c$ is a diagonal matrix comprised of the moments of inertia of the body's principal axes and $\bar{\mathbf{T}}$ is a vector containing the components of torque. Because the suspended object is cylindrical, its moments of inertia about the central diameter (i.e., the $\bar{y}$ and $\bar{z}$ axes in Fig. 4) are equal. The moment of inertia about the longitudinal axis is zero because, as noted in the previous section, the torque generated around the x-axis is zero.

To get the angular velocity vector, the angular acceleration in (32) is integrated and the body coordinate rate-to-Euler rate transformation applied as in [10], [12], yielding

$$\dot{\boldsymbol{\theta}} = \mathbf{T}_E\bar{\Omega}. \tag{33}$$

This enables the angular velocity defined in body coordinates to be expressed in inertial frame coordinates, which is where the suspended core's angular velocity will actually be measured. The Euler transformation matrix in (33), $\mathbf{T}_E$, is defined for a 3-2-1 (x-y-z) rotation sequence. From [10] $\mathbf{T}_E$ is defined as

$$\mathbf{T}_E = \begin{bmatrix} 1 & \tan\theta_y \sin\theta_x & \tan\theta_y \cos\theta_x \\ 0 & \cos\theta_x & -\sin\theta_x \\ 0 & \sec\theta_x \sin\theta_x & \sec\theta_y \cos\theta_x \end{bmatrix}.$$

It can be seen that by assuming small angles and low rates of change, $\mathbf{T}_E$ approximates an identity matrix and from [12] equation (33) becomes

$$\dot{\boldsymbol{\theta}} \cong \bar{\boldsymbol{\Omega}}. \tag{34}$$

The utility of the approximation will be seen later in this section when the system linear model is derived.

The translational, or linear, acceleration of the suspended magnetic core in body coordinates is defined as a function of magnetic force in the expression

$$\dot{\bar{\mathbf{V}}} = m_c^{-1}\bar{\mathbf{F}}, \tag{35}$$

where $\dot{\bar{\mathbf{V}}}$ and $\bar{\mathbf{F}}$ are the acceleration and force vectors and m_c is the core's mass.

To get the inertial translational velocity, an approach similar to the one taken with equation (34) yields

$$\mathbf{V} = \mathbf{T}_m^{-1}\bar{\mathbf{V}}, \tag{36}$$

where $\mathbf{T}_m$ is the inertial-to-body coordinate system transformation matrix, defined in [10] as

$$\mathbf{T}_m = \begin{bmatrix} c\theta_z c\theta_y & s\theta_z c\theta_y & -s\theta_y \\ (c\theta_z s\theta_y s\theta_x - s\theta_z c\theta_x) & (s\theta_z s\theta_y s\theta_x + c\theta_z c\theta_x) & c\theta_y s\theta_x \\ (c\theta_z s\theta_y c\theta_x + s\theta_z s\theta_x) & (s\theta_z s\theta_y c\theta_x + c\theta_z s\theta_x) & c\theta_y c\theta x \end{bmatrix},$$

where c and s are shorthand for cosine and sine. Assuming the angles of displacement are small between the two coordinate systems and the rates are small, then, as noted in [12], the matrix can be approximated as

$$\mathbf{T}_m = \begin{bmatrix} 1 & \theta_z & -\theta_y \\ -\theta_z & 1 & \theta_x \\ \theta_y & -\theta_x & 1 \end{bmatrix}. \tag{37}$$

This leads to the approximation used in [12]:

$$\mathbf{V} \cong \overline{\mathbf{V}}. \tag{38}$$

Thus far basic equations of motion for both linear and angular acceleration and velocity have been presented in equations (32), (33), (35) and (36). Next, the equations for electromagnetic torque and force are added to (32) and (35) to produce equations of motion as functions of controllable physical quantities, the magnetic flux density **B** and its gradients. To do this, the coordinate transformation matrix $\mathbf{T}_m$ must be applied to the field quantities as described below.

Because the magnetization vector $\overline{\mathbf{M}}$ is defined in the body coordinate system, and the magnetic field **B** is defined (produced) in the inertial coordinate system, $\mathbf{T}_m$ must be applied as follows to get the torque in body coordinates

$$\overline{\mathbf{T}} = \overline{\mathbf{M}}\mathbf{T}_m\mathbf{B}. \tag{39}$$

The expression for force is slightly more complicated because the magnetization vector in body coordinates must first converted to inertial coordinates before the product $\mathbf{M}\tilde{\mathbf{B}}$ can be calculated. Then, that product must be converted back from inertial to body coordinates before force is calculated. The basic force equation becomes

$$\overline{\mathbf{F}} = v\mathbf{T}_m\left[\partial\tilde{\mathbf{B}}\right]\mathbf{T}_m^{-1}\overline{\mathbf{M}}. \tag{40}$$

One more transformation is necessary, and that is to represent gravitational force in suspended body coordinates:

$$\overline{\mathbf{F}}_g = \mathbf{T}_m\mathbf{F}_g = \mathbf{T}_m\begin{bmatrix} 0 \\ 0 \\ -m_c g \end{bmatrix} = \begin{bmatrix} \theta_y m_c g \\ -\theta_x m_c g \\ -m_c g \end{bmatrix}$$

After performing the conversions, adding the gravitational force component and setting the rotation angles and rates around the x-axis to zero, the torque and force equations become

$$T_x = 0 \tag{41}$$

$$T_y = vM_x(-\theta_y B_x - B_z) \tag{42}$$

$$T_z = vM_x(-\theta_z B_x + B_y) \tag{43}$$

$$\mathbf{F}_g = \begin{bmatrix} \theta_y m_c g & 0 & -m_c g \end{bmatrix}^{\mathrm{T}}$$

$$F_x = vM_x(B_{xx} + 2\theta_z B_{xy} - 2\theta_y B_{xz} + \theta_z^2 B_{yy} - 2\theta_z\theta_y B_{yz} + \theta_y^2 B_{zz}) + \theta_y m_c g \tag{44}$$

$$F_y = vM_x(-\theta_z B_{xx} + B_{xy} - \theta_z^2 B_{xy} + \theta_z B_{yy} + \theta_y\theta_z B_{xz} - \theta_y B_{yz}) \tag{45}$$

$$F_z = vM_x(\theta_y B_{xx} - B_{xz} - \theta_z\theta_y B_{xy} - \theta_z B_{yz} + \theta_y^2 B_{xz} - \theta_y B_{zz}) - m_c g\,. \tag{46}$$

A further simplification is possible since for small angles their products are negligible. This reduces equations (44) to (46) to

$$F_x = vM_x(B_{xx} + 2\theta_z B_{xy} - 2\theta_y B_{xz}) + \theta_y m_c g \tag{47}$$

$$F_y = vM_x(-\theta_z B_{xx} + B_{xy} + \theta_z B_{yy} - \theta_y B_{yz}) \tag{48}$$

$$F_z = vM_x(\theta_y B_{xx} + B_{xz} - \theta_z B_{yz} - \theta_y B_{zz}) - m_c g\,. \tag{49}$$

One more expansion is done before the final equations of motion are formed. This time the terms in (41) to (43) and (47) to (49) consisting only of magnetic field quantities (e.g., B_z, B_{xy}, etc.) are expanded around the equilibrium point. Finally, the complete set of twelve detailed equations of motion can be produced by substituting the expanded equations for torque and force, into equations (32) and (35), giving

$$\dot{\overline{\Omega}}_x = 0 \tag{50}$$

$$\dot{\overline{\Omega}}_y = I_c^{-1} vM_x(-\theta_y B_x - B_z - B_{zx}x - B_{zy}y - B_{zz}z) \tag{51}$$

$$\dot{\overline{\Omega}}_z = I_c^{-1} vM_x(-\theta_z B_x + B_y + B_{yx}x + B_{yy}y + B_{yz}z) \tag{52}$$

$$\dot{\theta}_x = 0 \tag{53}$$

$$\dot{\theta}_y = \overline{\Omega}_y \tag{54}$$

$$\dot{\theta}_z = \overline{\Omega}_z \tag{55}$$

$$\dot{\overline{V}}_x = \frac{1}{m_c}[vM_x(B_{xx} + B_{xxx}x + B_{xxy}y + B_{xxz}z + 2\theta_z B_{xy} - 2\theta_y B_{xz}) + \theta_y m_c g] \tag{56}$$

$$\dot{\overline{V}}_y = \frac{1}{m_c}[vM_x(-\theta_z B_{xx} + B_{xy} + B_{xyx}x + B_{xyy}y + B_{xyz}z + \theta_z B_{yy} - \theta_y B_{yz})] \tag{57}$$

$$\dot{\bar{V}}_z = \frac{1}{m_c}[\nu M_x(\theta_y B_{xx} + B_{xz} + B_{xzx}x + B_{xzy}y + B_{xzz}z - \theta_z B_{yz} - \theta_y B_{zz}) - m_c g]. \quad (58)$$

$$V_x = \bar{V}_x \quad (59)$$

$$V_y = \bar{V}_y \quad (60)$$

$$V_z = \bar{V}_z \quad (61)$$

System Equations

In this section a linear mathematical model for the MSBS is derived based on its equations of motion. The premise for the linearization is that the magnetic fields are continuous and differentiable over the volume of the permanent magnet, allowing the MSBS plant to be treated as a linear system in a small region around some equilibrium point. The linear model developed in this section in turn forms the basis for designing the MSBS controller.

The Linear Model

The non-linear form of the MSBS plant is

$$\dot{\mathbf{X}} = f(\mathbf{X}, \mathbf{U}) = f\left(\begin{Bmatrix} ([\mathbf{I}_c]^{-1}\mathbf{T}) \\ (m_c^{-1}\mathbf{F} \end{Bmatrix}\right). \quad (62)$$

Based on the equations of motion (50) through (61), the variable vector $\mathbf{X}$ is defined as

$$\mathbf{X}^T = [\Omega_{\bar{x}} \quad \Omega_{\bar{y}} \quad \Omega_{\bar{z}} \quad \theta_x \quad \theta_y \quad \theta_z \quad V_{\bar{x}} \quad V_{\bar{y}} \quad V_{\bar{z}} \quad x \quad y \quad z]. \quad (63)$$

The plant is non-linear because the equations of motion are multivariable, nonlinear functions of states. The variable vector $\mathbf{U}$ in (62) represents the control input to the system, the current flowing in the five MSBS suspension coils. $\mathbf{U}$ is denoted by

$$\mathbf{U}^{\mathrm{T}} = [I_1 \quad I_2 \quad I_3 \quad I_4 \quad I_5]. \quad (64)$$

Coil currents are chosen as the system control inputs because the equations of motion are partly functions of magnetic flux density. At the distances of concern in a large gap suspension system like the MSBS, the operating values of the magnetic fields are well-approximated by the ratio found in [10]:

$$B = \frac{I}{I_{\max}} b_{\max}. \quad (65)$$

In this ratio, I is a variable representing the instantaneous control current in a given suspension coil, I_{max} is a constant denoting the maximum coil current allowed and b_{max} is a constant denoting the magnetic field value produced by I_{max}. Note that b_{max} is determined for a specific point in space, i.e., the equilibrium point. This ratio also holds for first and second order gradients of **B**.

The procedure for linearizing the non-linear MSBS plant is to perform a Taylor series expansion around some nominal equilibrium point $\overline{\mathbf{X}}_o$ within the composite magnetic field. There is also a set of equilibrium currents $\mathbf{I}_o$ associated with the spatial equilibrium point. A first order Taylor series expansion of (62) with the substitution of (63) and (64) results in

$$\dot{\overline{\mathbf{X}}}_o + \partial\dot{\mathbf{X}} = f(\mathbf{X}_o, \mathbf{I}_o) + \mathbf{A}\partial\mathbf{X} + \mathbf{B}\partial\mathbf{I},$$

where **A** and **B** are given by

$$\mathbf{A} = \left.\frac{\partial f}{\partial X}\right|_{X_o, I_o}, \text{ and} \tag{66}$$

$$\mathbf{B} = \left.\frac{\partial f}{\partial I}\right|_{X_o, I_o}. \tag{67}$$

Taking only the first order terms of the expansion gives a reasonable approximation as noted in [12] Consequently, the linear state equation form of (62) is

$$\begin{bmatrix} \dot{\overline{\mathbf{\Omega}}} \\ \dot{\boldsymbol{\theta}} \\ \dot{\overline{\mathbf{V}}} \\ \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \mathbf{A} \begin{bmatrix} \overline{\mathbf{\Omega}} \\ \boldsymbol{\theta} \\ \overline{\mathbf{V}} \\ x \\ y \\ z \end{bmatrix} + \mathbf{B} \begin{bmatrix} I_1 \\ I_2 \\ I_3 \\ I_4 \\ I_5 \end{bmatrix}.$$

Note four of the six state variable symbols do not have the over-bar symbol, indicating they are referenced to the inertial coordinate system. This is possible because of the approximations in (34) and (38), and it creates an advantage for control system design in that these state variables are directly measurable in the inertial reference frame.

Next the partial derivatives in (66) are taken with respect to the state variables in (63) and, if the components associated with the torque along the x-axis are excluded, the following 10-by-10 state matrix results:

$$\mathbf{A} = \mathbf{W}_1 \begin{bmatrix} 0 & 0 & -B_x & 0 & 0 & 0 & 0 & -B_{zx} & -B_{zy} & -B_{zz} \\ 0 & 0 & 0 & -B_x & 0 & 0 & 0 & B_{yx} & B_{yy} & B_{yz} \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -2B_{xz} + \dfrac{m_c g}{\nu M_x} & 2B_{xy} & 0 & 0 & 0 & B_{xxx} & B_{xxy} & B_{xxz} \\ 0 & 0 & -B_{yz} & B_{yy} - B_{xx} & 0 & 0 & 0 & B_{xyx} & B_{xyy} & B_{xyz} \\ 0 & 0 & B_{xx} - B_{zz} & B_{yz} & 0 & 0 & 0 & B_{xzx} & B_{xzy} & B_{xzz} \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}, \tag{68}$$

where

$$\mathbf{W}_1 = \begin{bmatrix} \nu\dfrac{M_x}{I_y} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \nu\dfrac{M_x}{I_z} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \nu\dfrac{M_x}{m_c} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \nu\dfrac{M_x}{m_c} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \nu\dfrac{M_x}{m_c} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{69}$$

Similarly, to form the input matrix **B**, the partial derivatives in (67) are taken with respect to the control inputs vector **I**. However, to do this, the magnetic fields and their gradients must be expressed in ratios, as in equation (65). For example, the x-component of magnetic flux density is approximated by

$$B_x = \frac{I}{I_{\max}} B_{x-\max}.$$

Because the present MSBS design calls for five electromagnet currents,

$$\left.\frac{\partial f(\mathbf{X},\mathbf{I})}{\partial \mathbf{I}}\right|_{X_o, I_o}$$

leads to an input matrix **B** with five columns. However, only five of the equations of motion contain **B**-field components and taking the partial derivative produces only five non-zero rows. It should also be noted that the partial derivative is evaluated for the model at its equilibrium point. Consequently, the derivatives containing angles or positions go to zero, since in the linear model these variables represent perturbations from equilibrium. The resulting input matrix is

$$\mathbf{B} = \mathbf{W}_2 \begin{bmatrix} -b_{z1} & -b_{z2} & -b_{z3} & -b_{z4} & -b_{z4} \\ b_{y1} & b_{y2} & b_{y3} & b_{y4} & b_{y5} \\ b_{xx1} & b_{xx2} & b_{xx3} & b_{xx4} & b_{xx5} \\ b_{xy1} & b_{xy2} & b_{xy3} & b_{xy4} & b_{xy5} \\ b_{xz1} & b_{xz2} & b_{xz3} & b_{xz4} & b_{xz5} \end{bmatrix}, \tag{70}$$

where

$$\mathbf{W}_2 = \begin{bmatrix} \frac{vM_x}{I_{cy} I_{\max}} & 0 & 0 & 0 & 0 \\ 0 & \frac{vM_x}{I_{cz} I_{\max}} & 0 & 0 & 0 \\ 0 & 0 & \frac{vM_x}{m_c I_{\max}} & 0 & 0 \\ 0 & 0 & 0 & \frac{vM_x}{m_c I_{\max}} & 0 \\ 0 & 0 & 0 & 0 & \frac{vM_x}{m_c I_{\max}} \end{bmatrix}. \tag{71}$$

The matrix above assumes $I_{\max}$ is the same for each coil.

Calculating Initial Conditions

The linear state space model $\{\mathbf{A},\mathbf{B},\mathbf{C}\}$ has been derived in terms of constants representing the magnetic flux densities and their gradients at the equilibrium point. However, the values of the constants are not yet determined. The common method for deriving numeric values for the state space model, employed in [3], [4], [12], and [14], consists of three steps: 1) calculate the value of the magnetic field component needed to hold the suspended core at the equilibrium point it turns

out only one is needed); 2) from the equilibrium value of the field calculate the values of the equilibrium current required to produce that field; and 3) using the equilibrium current values, calculate the remaining field components at equilibrium. This procedure is applied in Chapter VI where the values are determined for a two-degree-of-freedom configuration of the MSBS configuration.

Additional System Dynamics

To completely describe the MSBS for purpose of automatic control, consideration must be given to two additional system characteristics: 1) the dynamic response of the current amplifier/suspension coil combination; and 2) the effects of eddy current produced by the suspension coils in the 2.5–inch thick stainless steel wall of the MSBS. The role of these last two factors, amplifier dynamics and eddy current effects, in designing controllers for the MSBS is presented in more detail in later chapters.

Current Amplifier Dynamics

In Chapter II it was pointed out the MSBS suspension coils may be driven by currents ranging between -100 to 100 amperes, thus requiring a current amplifier with considerable dynamic range. Depending on its frequency response, this amplifier could pose a potentially significant performance limitation without compensation in the controller design. In [5] the dynamics for a current amplifier similar to the one slated for use in the MSBS were approximated experimentally by the third order transfer function

$$\frac{K(s+z_1)}{(s+p_1)(s+p_2)(s+p_3)},$$

where K = -51.585, z_1 = - 209.8, p_1 = - 499.8, $p_{2,3}$ = - 93 + 210i. Fortunately, in [4] the amplifier was found to have a relatively high roll-off frequency of about 400 Hz, indicating the amplifier dynamics may not have to be modeled.

Eddy Current Effects

An experiment described in [5] indicates eddy currents in the steel MSBS wall cause a 10-dB per decade roll-off in magnetic flux density magnitude, indicating a half-order transfer function. The 3 dB bandwidth at a point two inches from the inside of the test chamber wall was found to be approximately 27.5 Hz, but rapidly decreased to approximately 12.5 Hz at ten inches from chamber wall. The latter distance is more realistic for actual MSBS operation and the relatively

small bandwidth indicates the likelihood eddy currents will have to be modeled later in the controller design.

Conclusion

This chapter has explained how a mathematical linear model is developed for the MSBS. The equations above have been validated to some extent in models or other proof of concept demonstrations, although not in settings containing actual MSBS parameters or equipment. For example, researchers at NASA successfully demonstrated a six-degree-of-freedom, large-gap suspension system, but this system used different configuration of magnets configured in a planar array. References [4], [5], and [6] describe how controllers were designed and evaluated using Matlab's Simulink modeling software, but these controllers were based on hypothetical physical parameters quite different from the ones to be used in the actual MSBS. Finally, [5] and [6] describe actual levitation experiments, but these used only one or two components of the MSBS hardware and software suite. As will be seen in later chapters, knowledge of the theory behind the linear model is employed in building a working MSBS.

CHAPTER III

GENERAL IMPLEMENTATION METHODS

Introduction

Chapter I laid out the scope of the MSBS design and enumerated the high-level functional requirements supporting the design goals. Chapter II elaborated on the physics and control systems theory that underlie the MSBS operation. This chapter takes another step in developing the MSBS background by describing how the design goals and equations introduced in the first two chapters are actually addressed and implemented in software and hardware. In so doing, the chapter not only completes the background needed to begin the discussion on specific MSBS developments covered described in later chapters, but introduces several new techniques that are actually building blocks essential to the completion of the MSBS project.

Beyond the challenge of designing, fabricating and interconnecting the major MSBS components (e.g., the suspension chamber and electromagnets), the goal of instantiating the MSBS model and meeting the system's requirements in real hardware and software embodies at least three major sub-tasks: 1) performing in real time the calculations associated with signal conversions, system control logic and linear controller equations; 2) converting machine code into physical signals and vice-versa; and 3) providing real-time system monitoring and control for operators. It should be noted these three engineering issues have all been addressed in varying degrees using other terminology in [4], [5], [6] and [9]. However, no single reference has covered these issues as topics unto themselves, resulting in sketchy guidance for newcomers to MSBS control system design. This chapter uses the three tasks above as a framework for both familiarizing the reader with the system's development and operating environments and illustrating several practical techniques needed for building the full MSBS.

This chapter contains two main sections. The first introduces the structure of the MSBS development and operating environment and provides a general overview on how the designer uses its components. The second section illustrates the workings of the MSBS hardware and software in more detail by explaining the implementation of several specific MSBS functions, including performing sensor calibration, data capture and on-line parameter changes.

The MSBS Development and Operating Environment

This section describes the software and hardware environment in which the MSBS is both developed and intended to operate. Recall from Chapter I that one of the functional requirements

was for the MSBS to permit "rapid prototyping", which would allow MSBS developers and users the flexibility to quickly make design changes to the control system without resorting to a dedicated "development" platform. (See FR2310 in Table I.) It was partly for this reason the dSPACE system was selected, which employs nearly the same software and hardware components for both development and operation. Nevertheless, the sets of components used in development and operation do differ slightly and these differences are pointed out in the discussion of design procedures. To provide a convenient distinction between the developmental and operational use of the MSBS, the concept of the MSBS application is introduced in this section.

The Hardware and Software Environment

The MSBS software and hardware environment is where the three implementation issues presented above are tackled. This environment, illustrated in Fig. 7, contains multiple software applications running on two different hardware platforms, with inputs and outputs to and from the physical plant and its ancillary equipment and operator inputs (not shown).

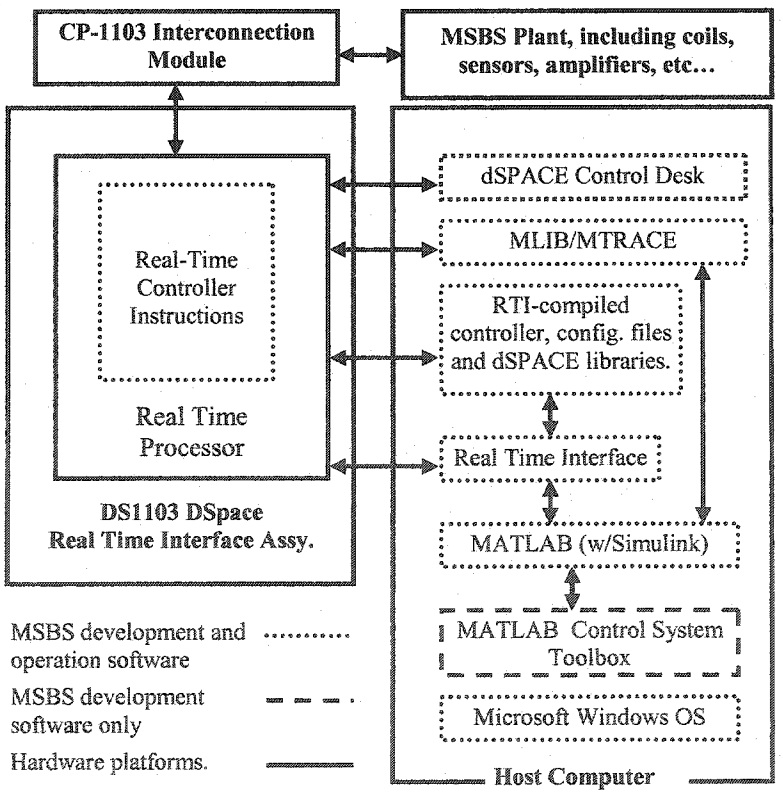


Fig. 7. Schematic of MSBS development and operating environments.

The first two major implementation issues, real-time execution of the controller, signal processing algorithms and control logic (the control system) and the digital interface between the control system and the physical plant, are handled primarily inside the DSpace DS1103 Real Time Interface (RTI) module. During normal system operation, the RTI contains the controller and signal conversion instructions running on the real-time processor (RTP), which is a Power PC 604e microprocessor in the current MSBS configuration. The RTI also holds a set of digital-to-analog (D/A) and analog-to-digital (A/D) converters, which serve as interfaces to the CP-1103 Interconnection Module. The RTI also simultaneously maintains a bidirectional interface with the DSPace Control Desk application on the host PC.

Position sensor signals are received via BNC connectors on the CP-1103 and passed to the A/D converters on the RTI. After conversion to digital form, the sensor signals are processed by the RTP to determine the displacements and rotations of the MSBS test object, which serve as feedback to the controller algorithm also running on the RTP. Similarly, the CP1103 receives TTL status signals from the Copley current amplifiers and the cooling subsystem, which are converted to digital form and used in the control logic on the RTP. In the opposite direction, control commands from the RTP are converted to TTL form and sent to the amplifiers and cooling system.

The third implementation challenge, providing operators real-time system monitoring and control, is met using the dSPACE Control Desk software running on the host PC. Control Desk communicates with the RTP either via the host PC's main communications bus or an external bus link if the DS-1103 RTI Card is installed externally. (The experiments covered here used an external DS-1103. See physical configuration baseline in Appendix A.) This interface allows the operator to load the control system instructions onto the RTP and start the controller operation. During normal operation, Control Desk lets the operator monitor and adjust selected parameters in the control system algorithm.

Basic MSBS Development Issues

Even though both development and operations use essentially the same hardware and software environment, only the operational use of the MSBS environment has been addressed thus far. The following paragraphs describe how the system designer creates the control system instructions in machine code and builds an operator interface to monitor and control the necessary system parameters.

The designer begins by modeling the operations of the entire control system in Matlab's Simulink modeling tool. When this is done in the MSBS development environment, and Matlab is configured appropriately, a link is automatically establish with the dSPACE Real-Time Interface (RTI) software. This link equips Simulink with RTI-specific interface blocks for use in the Real Time Processor (RTP) and commands that allow Simulink block diagrams to be compiled into machine code and placed into operation on the RTP. While designing the control system is challenging, the basic procedure for converting the design into machine code for the RTP is relatively simple:

1) When constructing the model of the control system in Simulink, the designer inserts the special RTI blocks to implement external interfaces such as control outputs and sensor inputs. These blocks embed a link to the RTI software.

2) To compile the control system model into executable coded and load the code on the RTP, the designer executes the "Build" command within Simulink. (This command is located on the main Simulink toolbar whenever the the model is open.)

To build the operator interface, the designer first uses the graphical user interface (GUI) design tool within Control Desk to create "virtual" instrument panels. These panels contain virtual displays and controls (e.g., line graphs, numeric readouts, thumbwheel switches and pushbuttons) through which operators interact with the control system. (See later chapters for specific MSBS screens.) Multiple operator screens with different sets of instruments can be created for a single set of machine code instructions, or application, on the RTP.

The crux of the Control Desk functionality, however, is the real-time linkage between the virtual instruments in the GUI and their corresponding parameters in the machine code on the RTP. This linkage is established through a system description file generated at the same time the Simulink model is compiled. This .sdf file presents the functions contained in the various Simulink modeling blocks as system "variables" for use in Control Desk. To assign the virtual instruments to specific parameters in the real-time control system, the designer opens the .sdf file within the Control Desk application and manually links the desired system variables to the correct virtual instruments. These associations are recorded in the .sdf file.

In addition to variable monitoring and control, Control Desk can also capture the values of selected dynamic variables over adjustable intervals using variable sample rates. This feature allows critical control system variables such as control inputs, feedback signals and system states

to be stored and analyzed in any number of ways using Matlab. Finally, Control Desk groups the several virtual instrument panels and the .sdf file pertaining to a particular control system into a single Control Desk "experiment".

Another component in the MSBS development and operating environment, the MLIB/MTRACE interface, gives Matlab direct read/write access to the control system on the RTP. This feature is potentially more capable than the Control Desk instrument panel because it brings to bear the whole range of calculating power of Matlab and its tool boxes, including the use of Matlab scripts, on the task of analyzing and updating control system variables in real time.

To summarize, during regular operation, the MSBS "application" runs in a software environment comprising the following major components:

1) The compiled controller, sensor signal conversion and control logic algorithms, i.e., control system, loaded on the dSPACE Real Time Processor (RTP).
2) Control Desk Software on the host computer.
3) Real Time Interface (RTI) software.
4) MLIB/MTRACE software for live parameter updating.
5) MATLAB software for live parameter updating.

The MSBS application is created in the MSBS development environment by first designing and coding the control system in Simulink and incorporating specific RTI blocks in the model. Then the Simulink model is compiled and loaded on the RTP on the RTI. This accomplishes the first two major implementation tasks: doing the control system calculations in real time and creating signal interfaces to the physical system. The system designer then creates a Control Desk experiment by designing the instrument panels and designating the variable links needed for the application. This step meets the third design challenge, providing an operator interface. Detailed descriptions of the installation and operation procedures for the dSPACE software can be found in [18], [19], [20], and [21].

Implementation of System Functions

This section describes some of the innovations made during the recent research. These innovations introduced the following functionality:

1) Simplified sensor signal processing
2) Sensor calibration using on-line parameter updates
3) Capturing data on system performance

4) Conversion of the controller output to control signals for the Copley amplifiers.

The descriptions of these innovations include technical background on the problems they are intended to solve as well as the specific algorithms and modeling features employed to solve them. As a result, the discussions in this section more illustrate in more detail the use of the MSBS software and hardware. Implementations of specific MSBS controllers are covered in later chapters.

Simplified Sensor Signal Processing

Recall from Chapter II that the MSBS is ultimately intended to control a test model in six degrees-of-freedom: vertical and horizontal lateral translation, longitudinal translation, roll, pitch and yaw. However, at this time the controller design only addresses five degrees-of-freedom, leaving roll control for future development. Thus, given the required ellipsoidal test model and the type of sensors selected for the MSBS, a minimum of five measurements are needed to sense the model's movement. These measurements are illustrated in Fig. 8, which shows the beams associated with the five emitter-receiver pairs.

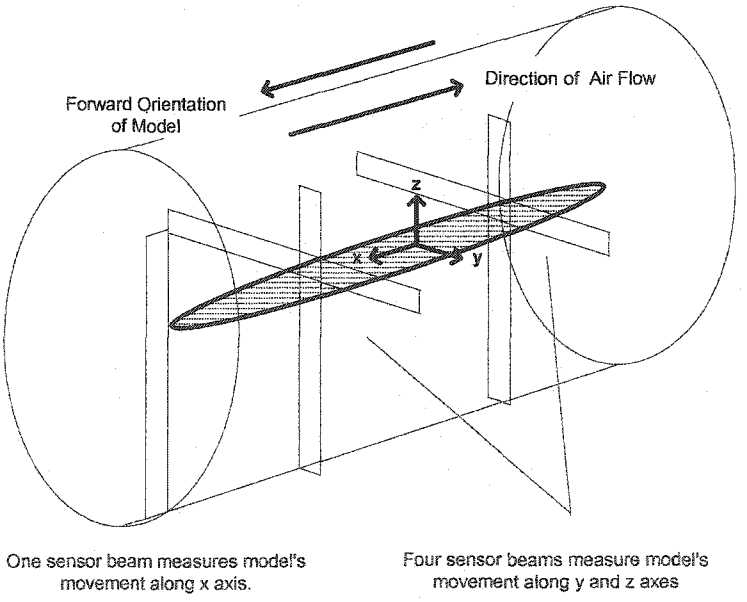


Fig. 8. Schematic diagram of sensor beam placement

The LA-511 sensors chosen for the MSBS are optical shadow sensors, i.e., they detect the percentage of a beam blocked by the sensed object. The sensors are aligned so that their beams

are partially blocked by a when the test object is in its equilibrium position. Other types of non-contact position sensing have been used for maglev systems, such as balanced induction sensors [22] or balanced optical reflection sensors [23], but these types of sensor systems were not chosen due to their relative complexity.

The sensor receivers produce an output voltage proportional to the amount of beam received. Knowing the sensor output values for the test object's equilibrium position allows the test object's deviation from equilibrium to be calculated. Fig. 9 illustrates the relationship of the test object to LA-511 sensor beams. For ease of calculation and to permit a symmetrical range of motion, the equilibrium point is chosen such that every beam is blocked by fifty percent. This ties the sensor/feedback subsystem somewhat intricately to the plant design, because the equilibrium point is determined by the electromagnetic fields produced by the actuators and the characteristics of the suspended object. The point is, while the positions of the sensors are not specified, they are critical to the performance of the system.

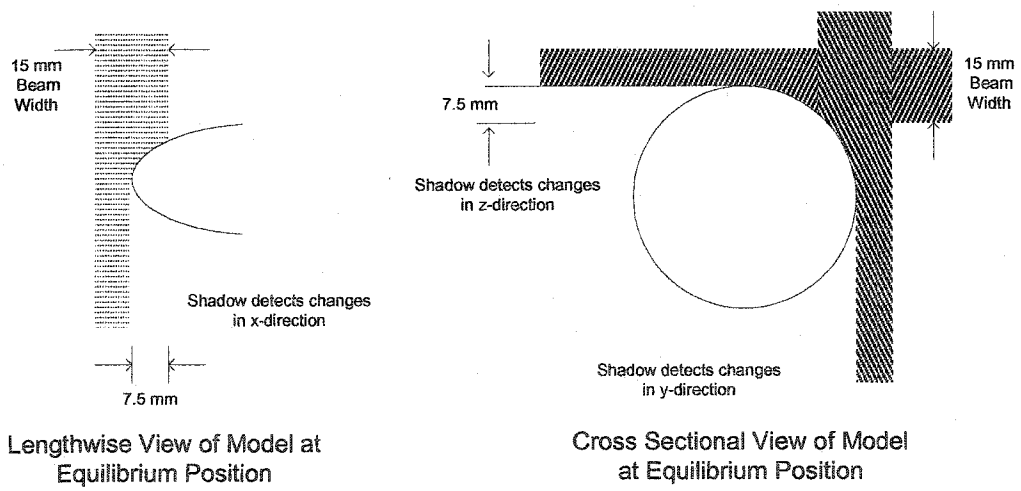


Fig. 9. Schematic diagram of position measurement.

Of course the purpose of the sensors is to provide the controller information about the states of the test object, specifically, the test object's deviations from equilibrium in the five-degrees-of-freedom. Doing this first requires converting the five raw sensor outputs into the linear displacements they represent. Following this, the five displacements are used to calculate the test object's translational and angular movements. The following paragraphs describe how the MSBS application performs these calculations.

Because the desired precision and response are not given for the sensor system/feedback path, from the system's intended use and the size of the test model, it assumed the sensor system must support sub-millimeter positioning.

In the work documented in [9], the sensor receiver's response was assumed to be nonlinear and the voltage-distance conversion was done by referencing the sensor outputs to calibrated look-up tables, implemented with standard modeling blocks in Simulink. These look-up tables were built by measuring the outputs of the sensors over the full range of their beam widths. This involved a time consuming process of positioning a calibration tool at eight or more settings within the beam and recording the receiver output voltages. Fortunately, later analysis of the input-output response of several LA-511 receivers showed the sensors were linear over most of their ranges. Accordingly, it was decided to replace the look-up tables with voltage-distance conversions based on the simple slope-intercept equation $y = mx + b$. The simplification afforded by this approach is described in the following paragraphs. More importantly, its validity was practically demonstrated in actual levitation experiments.

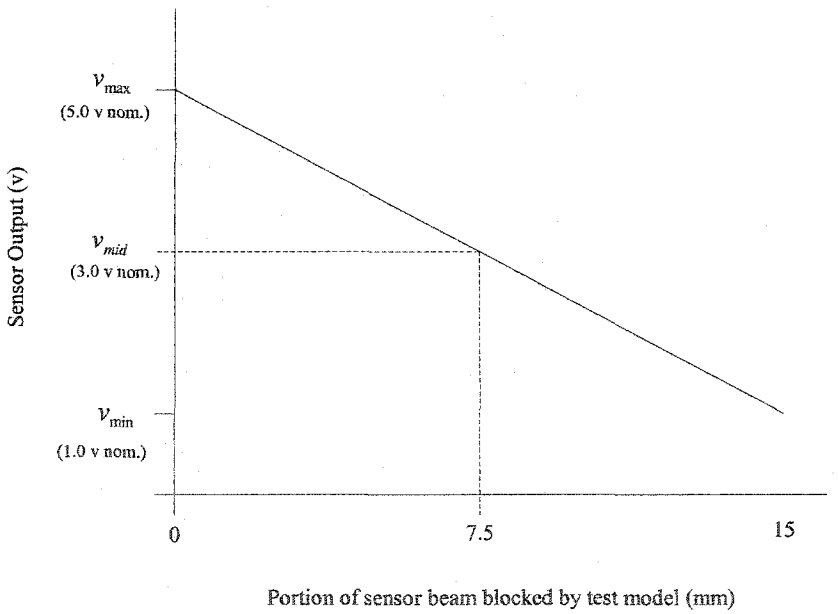


Fig. 10 Input-output relationship of an LA-511 sensor receiver [24].

The voltage-distance calculations are based on the typical manufacturer's input-output relationship for the LA-511, illustrated in Fig. 10. Given a nominal output range of v_{min} to v_{max} for a totally unobstructed to totally obstructed 15 mm beam, the slope is

$$|m| = (v_{max} - v_{min})/(\text{beamwidth}).$$

Using the slope-intercept equation, and noting that the slope in Fig. 10 is negative, a measure of displacement can be calculated by

$$v_{meas} = v_{\max} - |m| \times d, \text{ or}$$

$$d = (v_{\max} - v_{meas})/|m|, \tag{72}$$

where v_{meas} is the sensor output and d is the total beam obstruction in mm.

Equation (72) applies to a situation where d directly represents an object's movement with respect to the beam's edge. But since the MSBS control system requires both positive and negative displacements, the test object's equilibrium position is set at the point where the sensor beam is 50% obstructed. In this case, the displacement from equilibrium is given by

$$d = \frac{(v_{\max} + v_{\min})/2 - v_{meas}}{|m|}. \tag{73}$$

Assume that at equilibrium the upper edge of the test object lies at the midpoint of the beam as illustrated in the right hand side of Fig. 9. (Also assume the test object is wider that the beam.) At this point v_{meas} equals the midrange voltage $(v_{\max} + v_{\min})/2$ and the measured displacement equals zero. As the object moves down, allowing more of the beam to pass, the measured voltage increases, and equation (73) indicates a negative displacement. Correspondingly, upward movement of the test object produces a positive displacement. In the MSBS application, equation (73) is implemented for each of the five sensor pairs, giving the complete set of measurements needed to calculate the translational and rotational displacements of the test object.

The five displacement measurements are the minimum needed to implement the five-degree-of-freedom (5-DOF) controller. But processing these measurements is complicated due to the ambiguities introduced by the combinations of three simple types of movement: lateral (along the y and z axes), axial (along the x axis), and rotational (pitch or yaw). An analysis of this problem was done in [9], which is expanded on below. First, the calculations of the simple motions by themselves are explained and then the combinations of movements.

The simplest calculation is the one for strictly axial translation. Assuming the test object moves only along the length of the test chamber, i.e., the z-axis, axial translation is merely the distance measured by the z-axis sensor. (See the diagrams in Figs. 8 and 9.)

Simple lateral motion, either vertical or horizontal, requires a slightly more complicated calculation. Using a minimum number of sensors requires there be only two sensors to measure both lateral and rotational displacement in each of the x-y and x-z planes. This can be seen in the sensor placement schematic in Fig. 8. Taking the x-z plane as an example, and referring to Fig. 11, it can be seen that vertical displacement can be determined by taking the average of the two measured displacements, S1 and S2. In other words, lateral displacement is defined in terms of the movement of the test object at a point midway between the sensors. Comparing Figs. 11(a) and 11(b) shows this method holds when the test object is not level. The equation for this simple movement type is

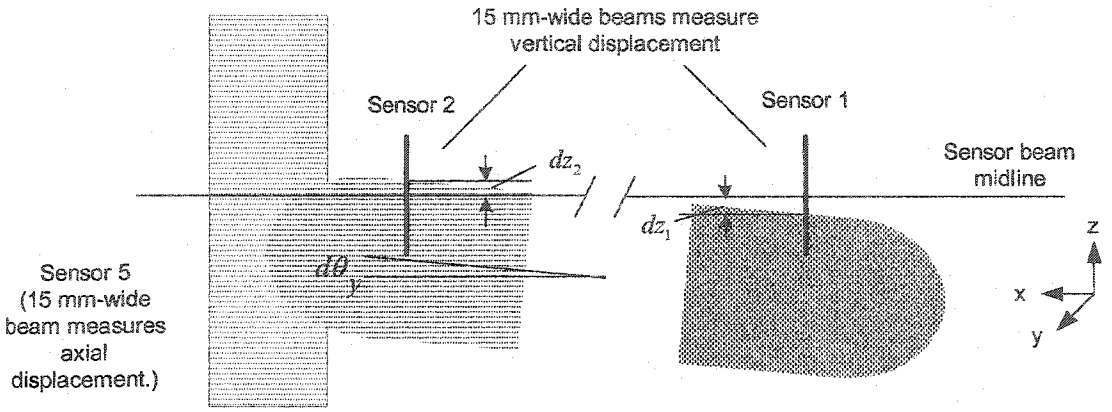
$$dZ = \left(dz_1 + dz_2\right)/2 . \tag{74}$$

Slightly more complicated are the two cases of simple rotational movement: rotation in the x-z plane (pitch) and rotation in the x-y plane (yaw). (As discussed earlier, roll is not controllable in the current system design.) Because the calculations have the same form for both pitch and yaw, only the calculation for displacement in pitch, denoted $d\theta_y$, is developed.

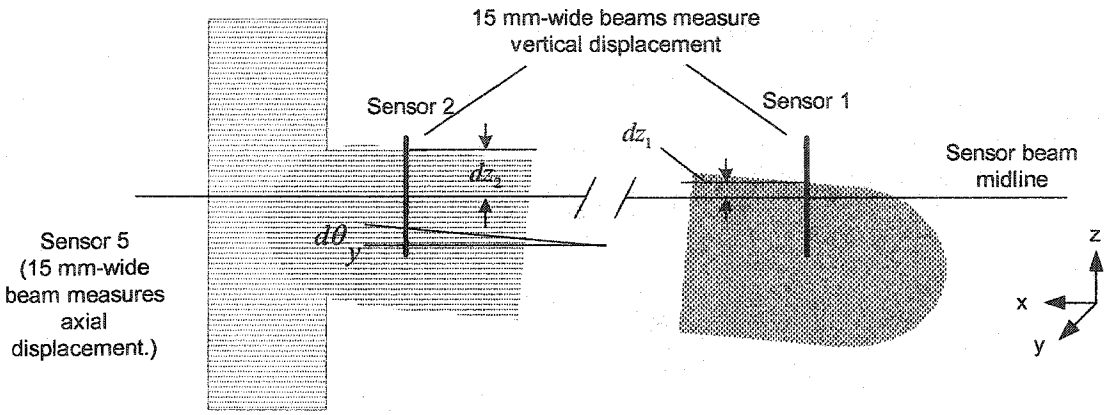
In Fig. 11(a) the test object is shown with a negative rotation, where sensors S1 and S2 measure negative and positive displacements, or dz_1 and dz_2, respectively. The rotational displacement $d\theta_y$ is given by the formula

$$d\theta_y = -\tan^{-1}\left(2dz_2/d\right) = \tan^{-1}\left(2dz_1/d\right) = \tan^{-1}\left(\left(dz_1 - dz_2\right)/d\right), \tag{75}$$

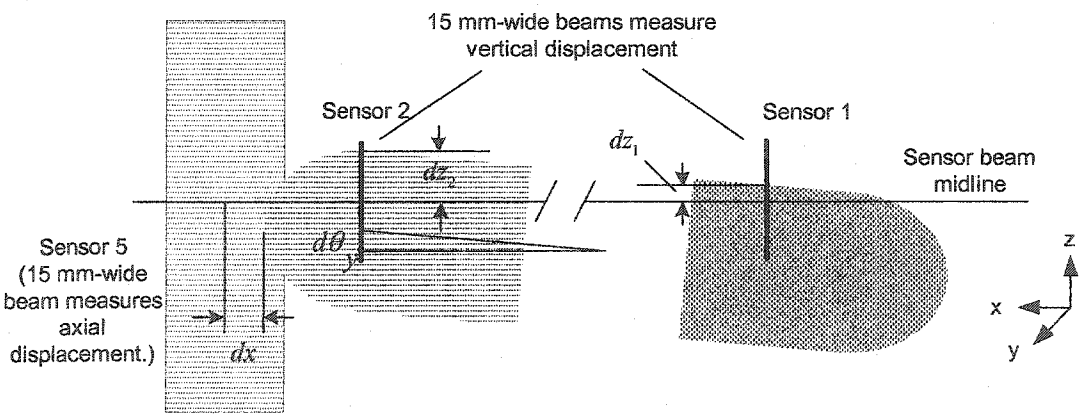
where d is the separation between the sensors along the x axis. This equation also holds when the object undergoes simple lateral translation, illustrated in Fig. 11 (b). Note the order of the displacements in (75) applies when positive rotation is in the clockwise direction. To maintain the correct sign for the angle, the positions of dz_1 and dz_2 must be reversed when positive is taken in the other direction.



(a) Rotational movement (pitch) with no vertical translation



(b) Rotational movement (pitch) with positive vertical translation



(c) Rotational movement (pitch) with vertical and axial translation

Notes: (1) Test object and sensor beams not drawn to scale.
(2) Only the ends of the test object are shown in the diagrams.
(3) Sensor 1 and Sensor 2 beams are shown in cross section.

Fig. 11. Illustration of rotational and translational movement in the x-z plane.

Normally the test object will typically undergo several types of movement at once, requiring the effects of one type of movement on another to be considered. One of these combinations of

movements is when lateral and rotational movement both occur in the same plane, illustrated in Fig. 11(b). Fortunately, as stated above, displacements are calculated just as they were when these movement types occurred by themselves, using equations (74) and (75).

Another combination is when the object axial displacement and pitch, shown in Fig. 11(c). In the planned configuration for the MSBS, calculating the axial displacement is can be treated the same with or without rotation, but only because the current design provides for an axisymetric shape at the front end of the test model and allows only small rotation angles. These constraints serve to minimize the dependence of dx on $d\theta_y$ and $d\theta_z$. These constraints will have to be considered if and when the MSBS requirements are changed to allow large rotation angles.

To elaborate on the range-of-motion constraints, note that a sensor beam width of 15 mm limits the detection range of each sensor to plus or minus 7.5 mm. This directly limits the controllable range of simple translational movement, but also indirectly constrains the controllable range of angular motion, as pointed out in [9] This stems from the geometry of the sensor system and is illustrated by the uppermost curve in Fig. 12. This curve represents the calculated rotation angle, based on equation (74), when both sensors for that plane of rotation measure the maximum linear displacement of 7.5 mm. This curve indicates the maximum measurable angle decreases from about seven degrees to one degree as the sensor separation increases from 5 to 34 inches. (The anticipated length of the actual MSBS test model is about 35 in. [3].) The sensor spacing in the 2-DOF experiments covered in Chapter VI was 30 inches, giving a maximum measurable rotation of 1.13 degrees.

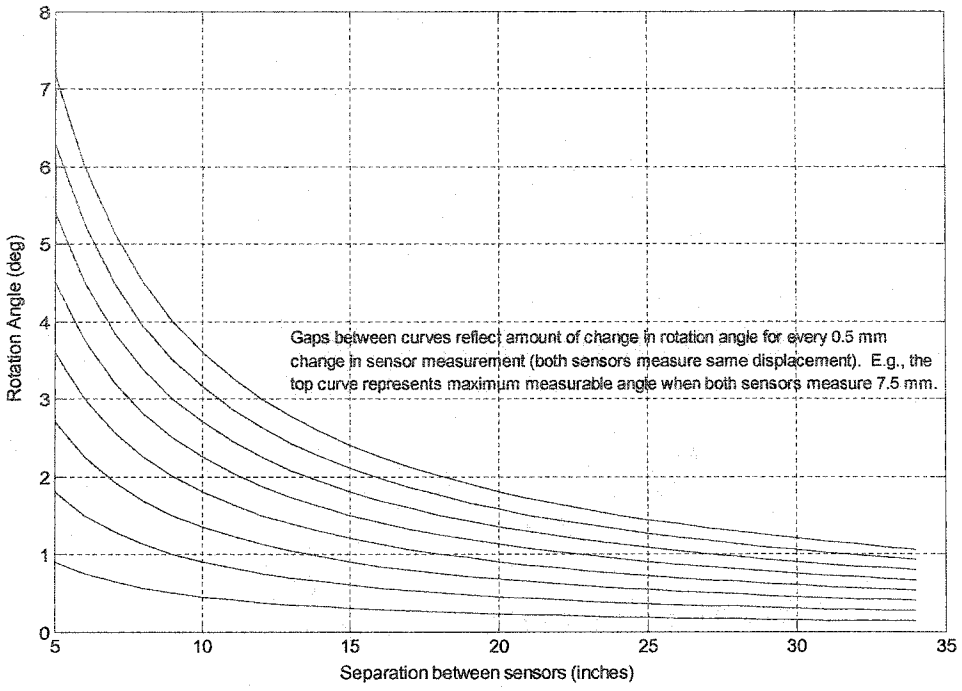


Fig. 12. Family of curves describing change in angular displacement measurement with increasing sensor separation.

However, a limitation on the maximum angle is not the only constraint imposed by the sensor system. Each curve in the family represents the calculated angle for a given displacement as measured at Sensor 1 and Sensor 2 in Fig. 11, i.e., dz_1 and dz_2. The gaps between the curves indicate the changes in calculated angle for 0.5 mm changes in measured displacement. The wider gaps at narrow sensor separations plainly indicate a decrease in measurement resolution with the increase in measurable angle. For example, a 0.5 mm shift at 5-inch sensor separation produces a nearly one-degree change in rotation angle. The same shift at a 30-inch sensor separation produces a 0.15-degree angle change. This poses a trade-off for the designer between limited maximum rotation angles and effective measurement resolution of the rotation angle.

Continuing the explanation of combined rotational and axial displacement, calculating the displacement angle in this case turns out to be the same as it is for rotation by itself. Consider Fig. 11(c), which shows the upper edge of a test model with negative pitch $d\theta_y$ and negative displacement along the x-axis, dx. The tangent of the displacement angle is expressed as

$$\tan\theta_y = dz_1 \Big/ \left(\frac{d}{2} + dx\right) = -dz_2 \Big/ \left(\frac{d}{2} - dx\right), \text{ or}$$

$$\tan\theta_y = \left(dz_1 - dz_2\right) \Big/ \left(\frac{d}{2} + dx + \frac{d}{2} - dx\right) = \left(dz_1 - dz_2\right)/(d), \tag{76}$$

leading to

$$d\theta_y = \tan^{-1}\left(\left(dz_1 - dz_2\right)/d\right). \tag{77}$$

Finally, simultaneous rotational, axial and lateral movement is considered. As state above, rotation angles and axial displacements are calculated the same before. However, the fixed sensor configuration produces ambiguity in the lateral displacement when the test model moves in an axial direction. Compare Figs. 11(a), 11(b) and 11(c), where the rotation angles are the same but the displacements measured by the respective sensors differ because of the axial shift. In Figs. 11(a) and 11(b) the displacement is calculated as before:

$$dZ = \left(dz_2 + dz_1\right)/2 .$$

But for Fig. 11(c) the same calculation yields a non-zero answer where there is no actual lateral movement

$$dZ = \left(dz_2 + dz_1\right)/2 \neq 0 .$$

This is an apparent lateral displacement. Thus it can be seen that lateral displacement can have both apparent and actual components, that is

$$dZ_M = dZ + d\bar{Z} ,$$

where $d\bar{Z}_M$, $d\bar{Z}$ and dZ are the measured, apparent and actual model displacements, respectively. To resolve the ambiguity, first let

$$dZ_M = dZ + d\bar{Z} = \left(dz_2 + dz_1\right)/2 - dx \tan d\theta_y .$$

Then, from (75) the actual displacement can be expressed in terms of the measured displacement and a correction factor:

$$dZ = dZ_M - d\bar{Z} = \left(dz_2 + dz_1\right)/2 - dx\left(dz_1 - dz_2\right)/d . \tag{78}$$

Obviously, when there is no axial displacement, the apparent lateral displacement disappears. To give the true displacement, the apparent displacement is removed. Note the signs of the terms

assume the MSBS coordinate system, where the negative direction on the x-axis is to the right in the figure.

Based on equations (74), (77) and (78), Table III summarizes all the equations needed to convert sensor signals to five DOF feedback:

TABLE III
SUMMARY OF ALGORITHMS FOR SENSOR SIGNAL CONVERSION [a]

Axial Displace-ment	Lateral Displace-ment	Rotational Displace-ment	Lateral Displacement Calculations	Rotational Displacement Calculations
O	O	X	------	$d\theta_y = \tan^{-1}\left((dz_1 - dz_2)/d\right)$ $d\theta_z = \tan^{-1}\left((dy_1 - dy_2)/d\right)$
O	X	O	$dZ = (dz_2 + dz_1)/2$ $dY = (dy_2 + dy_1)/2$	------
O	X	X	$dZ = (dz_2 + dz_1)/2$ $dY = (dy_2 + dy_1)/2$	$d\theta_y = \tan^{-1}\left((dz_1 - dz_2)/d\right)$ $d\theta_z = \tan^{-1}\left((dy_1 - dy_2)/d\right)$
X	O	X	------	$d\theta_y = \tan^{-1}\left((dz_1 - dz_2)/d\right)$ $d\theta_z = \tan^{-1}\left((dy_1 - dy_2)/d\right)$
X	X	O	$dZ = (dz_2 + dz_1)/2$ $dY = (dy_2 + dy_1)/2$	------
X	X	X	$dZ = (dz_2 + dz_1)/2$ $-dx(dz_1 - dz_2)/d$ $dY = (dy_2 + dy_1)/2$ $-dx(dy_1 - dy_2)/d$	$d\theta_y = \tan^{-1}\left((dz_1 - dz_2)/d\right)$ $d\theta_z = \tan^{-1}\left((dy_1 - dy_2)/d\right)$

[a] Solutions for axial displacement are the same under all conditions, i.e., axial displacement is direct measurement from axial displacement sensor.

To see how the MSBS application implements the sensor signal processing, consider the Simulink model in Fig. 13. The blocks on the left-hand side implement the equation to convert sensor output voltages to the distances they represent. Assuming the same sensor configuration as the one depicted in Fig. 13, the set of blocks on the right-hand side of the model implement the calculations for vertical translation and pitch. When the MSBS application is compiled, the Matlab code represented by these blocks will become part of the instruction set running on the Real Time Processor (RTP) and the blocks will become variables available to Control Desk.

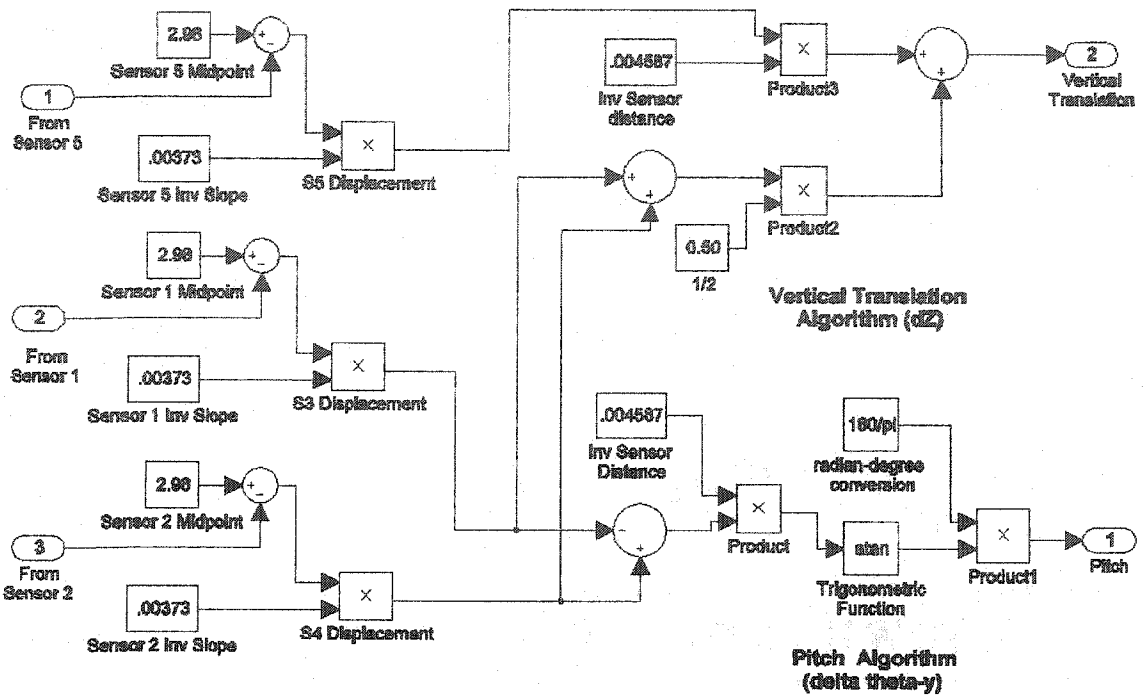


Fig. 13. Representative Simulink model showing sensor signal conversion to pitch and vertical translation.

Sensor Processing Calibration and Live Parameter Update

Voltage-distance conversion was described as the first step in sensor signal processing and is based on a simple linear equation, repeated here

$$d = \frac{\left[\left(v_{max} + v_{min}\right)/2\right] - v_{meas}}{|m|}.$$

However, because the output range of a given sensor receiver can change over time, and the range can vary between individual receivers, a method to compensate for these variations is needed to maintain the accuracy of conversions. If look-up tables were used to do the conversions, the compensation method would require rebuilding the tables, a time consuming procedure consisting of opening the test chamber, adjusting a test model through a calibrated range of positions and measuring the corresponding voltage outputs [9]. Fortunately, use of the above equation enables the voltage-distance algorithms to be updated based on two relatively easily obtained values, v_{min} and v_{max}.

It would be simple enough to measure v_{min} and v_{max} by reading the sensor output voltage when the beam is both fully on and fully obstructed, and then calculate the slope $|m|$ based on those values. However, it was found the typical sensor response is not linear over its entire range, specifically at the range ends. Thus using v_{min} and v_{max} directly would not provide a reliable voltage-distance conversion. Consequently, a means was devised to reliably derive the maximum linear output range for any given sensor using v_{min} and v_{max}.

First, an experiment was set up to analyze the input-output characteristics of five LA-511 sensor receivers. In this experiment, a calibrated positioning device was used to move a sample test object from one edge of the sensor beam to the other in one-millimeter increments while the voltage output was recorded for each position. This procedure was performed five times for each of the five sensors. Then for each sensor the average of the five outputs at each object position was calculated and a plot of the averages generated. A comparison of the five plots validated the assumption that the LA-511 outputs had a large linear region extending to just about a millimeter from their end points, implying the simple linear conversion formula could provide an acceptable range of operation for the MSBS.

To complete the analysis, the two extreme data points for each set of responses were discarded and a linear regression analysis was done on the remaining points to calculate the slope of the linear portion of the response. By comparison, the slopes derived from v_{min} and v_{max} were flatter than those of the linear region. This experiment showed that basing the voltage-distance conversion algorithm on a truncated sensor range would not only better match the typical MSBS operating range, but that there was enough similarity between the output plots that a standard method could probably be devised for all sensors.

To develop the standard method for truncating the output ranges, a means was needed to express the amount of truncation in terms of the two easily measured values, v_{min} and v_{max}. First, based on the analysis of the plots, it was determined a reasonable first step would be to truncate the input ranges of the sensors by one millimeter at each end. Then for each sensor the average outputs at beam widths of 1 and 14 mm were used to determine the average upper and lower limits of the linear output region for that sensor. For example, for Sensor 3 the sample mean for all five sets of measurements is 4.90 v at 14 mm and 1.29 v at 1 mm, which correspond to 97.1% and 6.43% of Sensor 3's total output range, respectively. Lastly, the averages of all five sensors' upper and lower percentages were determined, which were 95.37 % and 4.47 %, respectively.

(Unfortunately the sample size was limited due to the small number of sensors available.) Thus a simple calibration technique was found to compensate for changes in v_{min} and v_{max} and create viable working parameters for the voltage-distance conversion algorithm, namely using the equations

$$v_{\text{max-working}} = v_{\text{min-meas}} + 0.9537 \times (v_{\text{max-meas}} - v_{\text{min-meas}}), \text{ and}$$

$$v_{\text{min-working}} = 1.0447 \times v_{\text{min-meas}}.$$

To validate this technique, the percentages obtained above were applied to all five sensors and the endpoints of the truncated ranges were used to calculate an input-output slope for each sensor. Then, simulated input-output responses were generated using the conversion algorithm using the slopes obtained by the truncation method. It was reasoned that if the simulated responses were close to the responses obtained from the actual sensor measurements, this approach would be validated.

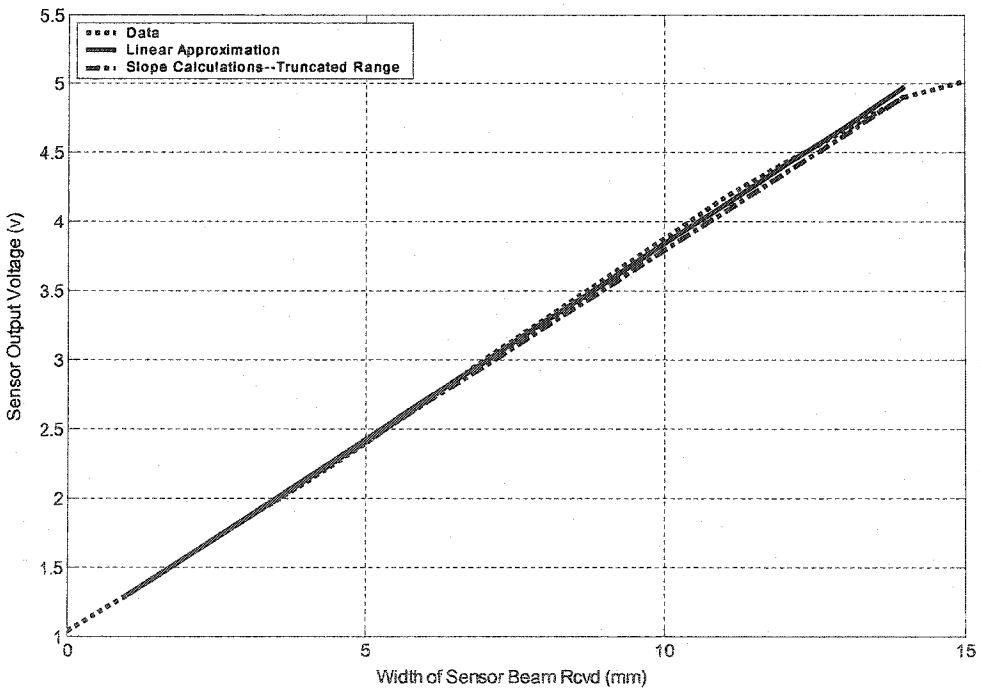


Fig. 14. Measured and calculated responses of an LA-511 sensor.

From a comparison of the plots, this method did seem acceptable, as indicated by Fig. 14. The dotted line is the average measured response of one of the sensors over its entire range; the solid

line is the linear approximation of the truncated response; and the dashed line is the simulated response using the truncated slope as described above. (Matlab scripts for this experiment are provided in Appendix D.) Most importantly, subsequent levitation experiments validated this method for deriving conversion algorithm parameters.

Having shown the viability of a relatively simple way to derive reliable sensor signal conversion parameters, the task remaining was to implement this method in the context of a real-time parameter update, i.e., while the MSBS is operating. Live parameter updating employs a separate dSPACE software program MLIB/MTRACE. This program provides an interface between the Matlab workspace on the host PC and the MSBS application on the real time processor (RTP). This interface enables variables in the real time application to be changed while the application is running.

To use MLIB/MTRACE, two preliminary steps are necessary. First, Matlab must be configured to operate with the MLIB/MTRACE software as described in [20]. Second, the MSBS GUI in Control Desk has to be set up to recognize parameter changes. This is done—while the MSBS application and its GUI are up and running—by identifying a parameter file and declaring what parameters are to be changed. After that a Matlab script containing MLIB/MTRACE commands can be executed to read the max/min sensor voltages, calculate the corresponding input-output slopes and midpoints and replace these values currently in use in the MSBS application. Finally, the new parameter values are saved as the new default values next time the MSBS application is run. Appendix B contains a detailed procedure for live parameter updating and Appendix D contains a sample Matlab script using the MLIB/MTRACE commands.

Data Capture

In the earlier discussion of the MSBS operating environment, it was pointed out the Control Desk software can be set up to capture the instantaneous values of various MSBS states and conditions, or variables, selected by the designer. One application of this feature would be to characterize the MSBS controller's performance by capturing test object's position as the control input is varied. The following is an overview explanation of the basic ideas involved in the operation. Much more detailed information is presented in [18] and [21].

Control Desk provides two types of instruments in its GUI design tool: 1) the single-shot, or virtual instrument and 2) the time-trace, or data acquisition instrument. These instrument types are based on two different dSPACE software segments called system services, which bring the data from the real-time application on the Real Time Interface (RTI) to the Control Desk on the

host PC. For each application such as the MSBS, there is a system description file (.sdf) that defines these services. (Recall that the .sdf file, built when the Simulink model is compiled, also contains the names of all the application's variables.) Capturing data associated with single-shot instruments such as numeric readouts and LED displays requires the designer to specifically configure the application's Simulink model so as to establish the appropriate system service in the .sdf file. However, data acquisition instruments use the default time-trace system service, meaning that whenever this type of instrument is linked to a variable in the .sdf file, data capture connections are automatically established between Control Desk and the RTI. This enables the designer to capture data associated with plotting instruments such as X-Y plotters simply by setting defining the data capture parameters within Control Desk.

Control Desk provides a window for the operator to select the data to be captured and specify the parameters for the capture, including the total sampling time, the sample rate and the method for triggering the capture. As noted above, only data associated with data acquisition instruments can be selected without prior configuration of the application program on the RTI. The captured data can be written to disk in the form of structured data files. Some of the basic capture parameter settings are listed in the table below.

TABLE IV
BASIC DATA CAPTURE PARAMETERS

Setting	Description
Capture Start/Stop	Choose whether or not to start capturing the data automatically as soon as the application starts.
Interval Length	The duration of the entire data capture interval.
Down Sampling	Ratio between samples read by the default sampling rate of the application and the sample rate of the data capture.
Trigger On Signal	Select a signal (variable) within the application to trigger the data capture.
Acquisition Mode	Simple: Capture the value of a variable over a single period.
	Autosave: Simple capture with automatic file save using same file name.
	Autoname: Simple capture with automatic file save using filename generated automatically based on root filename provided by the user. (E.g., MSBS_001.mat, MSBS_002.mat, etc…)
	Continuous: Capture that continues until explicitly stopped.
	Stream to Disk: Continuous capture that continuously write data to disk.

When Control Desk writes the captured data to disk, it places them in structured data files with the .mat extension. Each file is structured with two or three tiers of data fields which contain the raw data itself as well as information about the data such as numeric form of the data (single or dual precision, integer, etc...), the name and version of the application that captured the data, and the parameters of the data capture. The specific information stored in the .mat file depends on the application generating it, but typical meta-data for MSBS experiments includes the capture interval length, down-sampling rate and sample period.

The following example, adapted from one of the MSBS experiments, illustrates the basic data capture procedure. Consider the evaluation of a two-degree-of-freedom (2-DOF) controller for the MSBS. Four data sets are needed: the control inputs for elevation and pitch and the elevation and pitch states. This data is displayed on plotting instruments in the MSBS GUI so data capure capture can be set up in the Control Desk data capture window. Suitable capture parameters would include: those in Table V.

TABLE V
TYPICAL CONTROL DESK DATA CAPTURE SETTINGS

Parameter Name	Setting
Capture mode	Autoname
Capture variables	Elevation input, pitch input, elevation position, pitch angle
Data filename	MSBS_response_xxx
Triggering	Manual (i.e., operator initiated)
Interval length	5 sec.
Sample Period	.001 sec.
Downsample rate	100

Using these settings, the resulting .mat data file would contain four sets of 50 data points describing the conditions of the four variables listed above over a period of five seconds. Plots of this data or other analyses can be done using Matlab. The Matlab script in Fig. 15 is a simple example.

```
load MSBS_response_001;                          Load data file into Matlab workspace.

length= MSBS_response_001.Capture.Length         Retrieve capture interval length.
speriod= MSBS_response_001.Capture.SamplingPeriod;        Retrieve sample period.
dsrate= (MSBS_response_001.Capture.Downsampling);     Retrieve downsample rate and
dsrate=double(dsrate);                              Convert to double precision
                                                      variable for calculation.
n=((length/speriod)/dsrate)+1;                         Calculate number of data
                                                           points in .mat file.
t1=(1:n)*.01;                                   Create time axis for data plots.
Pitch= MSBS_response_001.Y(4).Data;                Load data into Matlab arrays.
Position= MSBS_response_001.Y(3).Data;
pos_cmd= MSBS_response_001.Y(2).Data;
pitch_cmd= MSBS_response_001.Y(1).Data;
plot(t1, pitch_cmd, t1, Pitch(i,:)); figure;     Plot pitch command and responses.
plot(t1, pos_cmd, t1, Position));                  Plot elevation position command
                                                              and responses.
```

Fig. 15. Sample Matlab script illustrating manipulation of data from structured data files.

Amplifier Input Signal Conversion

In Chapter I the need for large current amplifiers to drive the suspension coils was explained. In the MSBS architecture, these amplifiers are indirectly controlled by the system controller algorithm running on the RTP. The controller's output signals do not drive the coils directly but rather represent the values of the drive currents; specifically the voltage level of the controller output corresponds numerically to the desired current level in the suspension coil. The Copley current amplifiers convert the input signal voltage to some level of output current, but not in direct numeric correspondence as the controller algorithm expects it to be. For example, a 20 V signal into the amplifier does not produce a 20 A output. This gives rise to the important engineering consideration discussed below, namely the DC input-output characteristic of the Copley P232 power amplifiers.

To ensure the amplifiers produce the coil current demanded by the controller, it's necessary to first characterize the ratio of amplifier's output current to its DC control input voltage and then insert a compensator for the amplifier's DC response into the controller algorithm. Note that this DC characteristic differs from the amplifier's dynamic, or frequency response that was discussed in Chapter II.

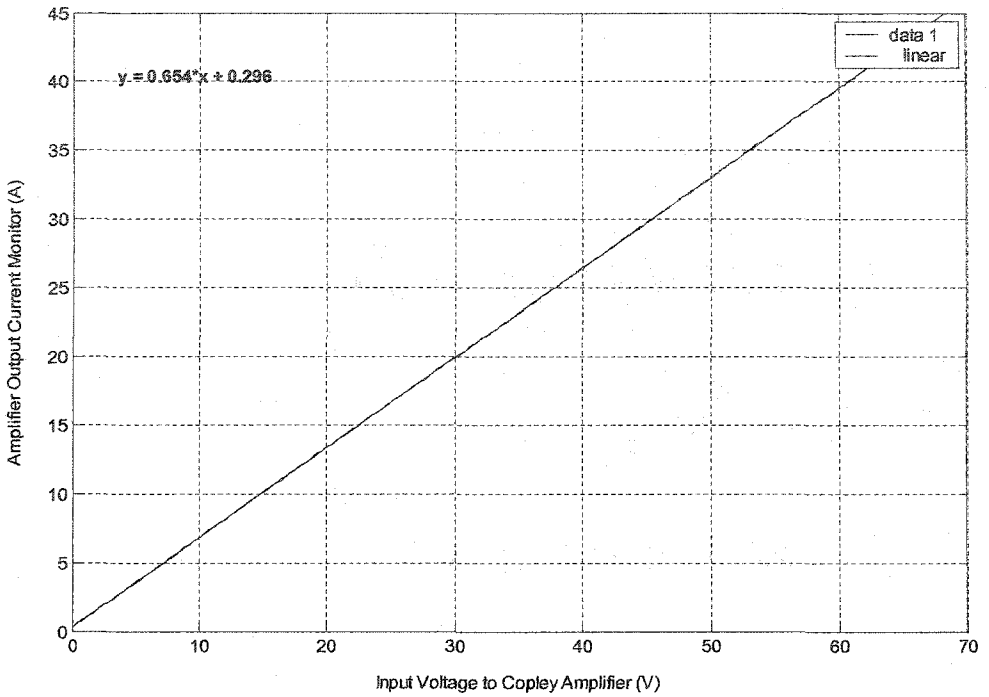


Fig. 16. Response of Copley 232P amplifier to range of inputs.

An experiment was set up using amplifier control logic, a Control Desk instrument panel, one of the Copley current amplifiers and one of the MSBS suspension coils. The amplifier input was adjusted in 5 V increments and the input voltage and the coil current were recorded and plotted. Fortunately, the input-output response was very linear, as illustrated by the virtual overlap of the plotted data and its corresponding fitted curve in Fig. 16.

The equation describing the amplifier input-output characteristic is

$$f_{amp}(v_{in}) = 0.654 v_{in} + 0.296 = i_{out},$$

where v_{in} is the voltage input and i_{out} is the amplifier output current. As stated above, the controller output forms the input to the amplifier v_{in}. Now if it is desired i_{out} be the same in sign and magnitude as the controller output, then it stands to reason that v_{in} must first be transformed by the inverse of the input-output characteristic before being applied to the amplifier. This transformation is given by

$$f_{amp}^{-1}(v_{in}) = (i_{des} - 0.296)/0.645, \tag{79}$$

where i_{des} is the desired output current. Because by definition i_{des} is also equal in sign and magnitude to v_{in}, then this leads to amplifier output given by

$$f_{amp}\left(f_{amp}^{-1}\left(v_{in}\right)\right) = v_{in} = i_{des}.$$

This transformation is implemented in the as the last process in the control system before the control system is converted to analog form. A Simulink block diagram of the transformation is shown in the following chapter along with some other specific implementations for the one-degree-of-freedom experiment.

Conclusion

The primary goal of this chapter was to complete the background work started Chapters I and II in order to provide the basis for understanding the more complicated MSBS experiments to follow. The secondary purpose was to facilitate future development. The chapter described the structure of the MSBS software and hardware environment and introduced the basic methods of using that environment to implement MSBS functions. Several new solutions to engineering design problems were presented as means to illustrate the implementation methods, including solutions relating to sensor signal processing, data capture and manipulation and control signal processing. The following chapters take up the more complicated explanations of specific control system issues. In the process, further elaboration is made on the practical engineering challenges of producing the completed MSBS.

CHAPTER IV

ONE-DEGREE-OF-FREEDOM TESTS

Introduction

The ultimate goal of the work supporting this thesis is to produce a full-scale MSBS prototype. Chapters I through III have laid out the design requirements, the system configuration, a large portion of the theoretical background and the basic implementation methods for achieving that goal. This chapter begins the explanation of how control systems theory was applied to actual MSBS hardware and software by describing a set of experiments centered on implementing a one-degree-of-freedom (1-DOF) maglev system.

Previous work included sensor testing, validating suspension coil parameters, control system modeling and simulation using the dSPACE software application and some prototyping with power amplifiers similar to the Copley P232 and small electromagnets. Most of this work was documented in [4], [5] and [6]. However, no test had been done of a closed-loop control system employing primarily MSBS components and levitating a test object at a distance near its intended gap. The 1-DOF experiments described in this chapter were the first successful attempt to do so and served to validate important assumptions and design decisions upon which MSBS controller design is based. These include the assumption of a "large-gap" system as described in Chapter II, the assumption of a large, linear range for the LA-511 sensors, use of the voltage-distance conversion algorithm for the sensors and use of the input signal conversion for the Copley 232P amplifiers.

Up to this point a linear model of the MSBS has been defined but the model's use in control system design has not been discussed. This chapter addresses that aspect of design by adapting the 5-DOF model to the 1-DOF case and applying classical single-input, single-output (SISO) control theory. In addition, this chapter also expands on the implementation methods introduced in the previous chapter by describing how the controller is implemented in Simulink and explains several practical engineering concerns.

The remainder of this chapter is divided into three main sections. The first section covers the preliminary work needed to perform the 1-DOF experiment, including the adaptation of the 5-DOF linear model to the 1-DOF case and the hardware and software setup. The next section discusses the first group of 1-DOF experiments including the controller design, experimental procedures and the results. The third main section gives a similar presentation of a second set of 1-DOF experiments.

Experimental Preliminaries

The purpose of the 1-DOF experiments was to begin to demonstrate the feasibility of meeting the MSBS design requirements using the linear model described in Chapter II and the actual components selected for the operational system. To that end an experiment was devised that would provide a meaningful evaluation of some of the assumptions and MSBS design choices. This section describes some of the work preparatory to the experiment in order to orient the reader to the specific physical setup and theory involved as well as to illustrate additional techniques that will also be needed to implement larger, multiple degree-of-freedom prototypes. The subsections deal with the physical setup, application of the electromagnetic theory, determination of the magnetic fields, controller design and several specific Simulink modeling features.

The 1-DOF System Setup

The 1-DOF experiment was set up to levitate a permanent magnet core at a distance below a suspension coil approximating the gap anticipated in the actual MSBS. The setup is described as follows. A pair of MSBS suspension coils, mounted on a backplane of structural fiberglass (as they will be in the full 6-DOF MSBS) were placed on a wooden frame such that their magnetic axes were oriented vertically, or parallel to the z-axis in the MSBS coordinate system. (See Fig. 17.)

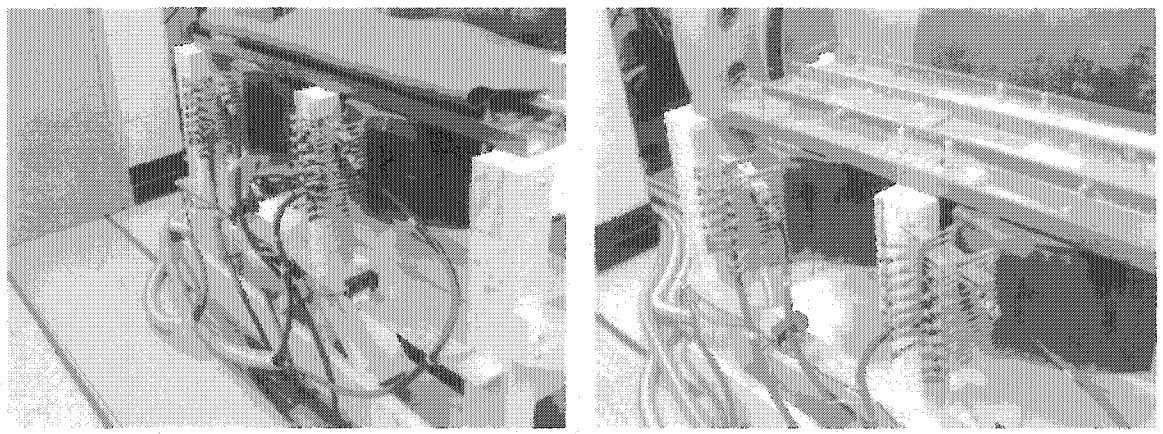


Fig. 17. MSBS suspension coils on test frame.

One of the coils was wired to one of the Copley 232P amplifiers so that its magnetic flux density was oriented downward, or in the negative z direction. An LA-511 sensor pair was mounted below the active suspension coil such that the 15 mm beam was oriented vertically and the beam crossed directly below the center of the coil. The height of the sensor pair was adjusted so that

the top of the permanent magnet core would be at the midline of the sensor beam when the suspended core was at its equilibrium point. This is illustrated in the schematic diagram of the 1-DOF system in Fig. 18. The composition of the permanent magnetic core is discussed in more detail in following paragraphs.

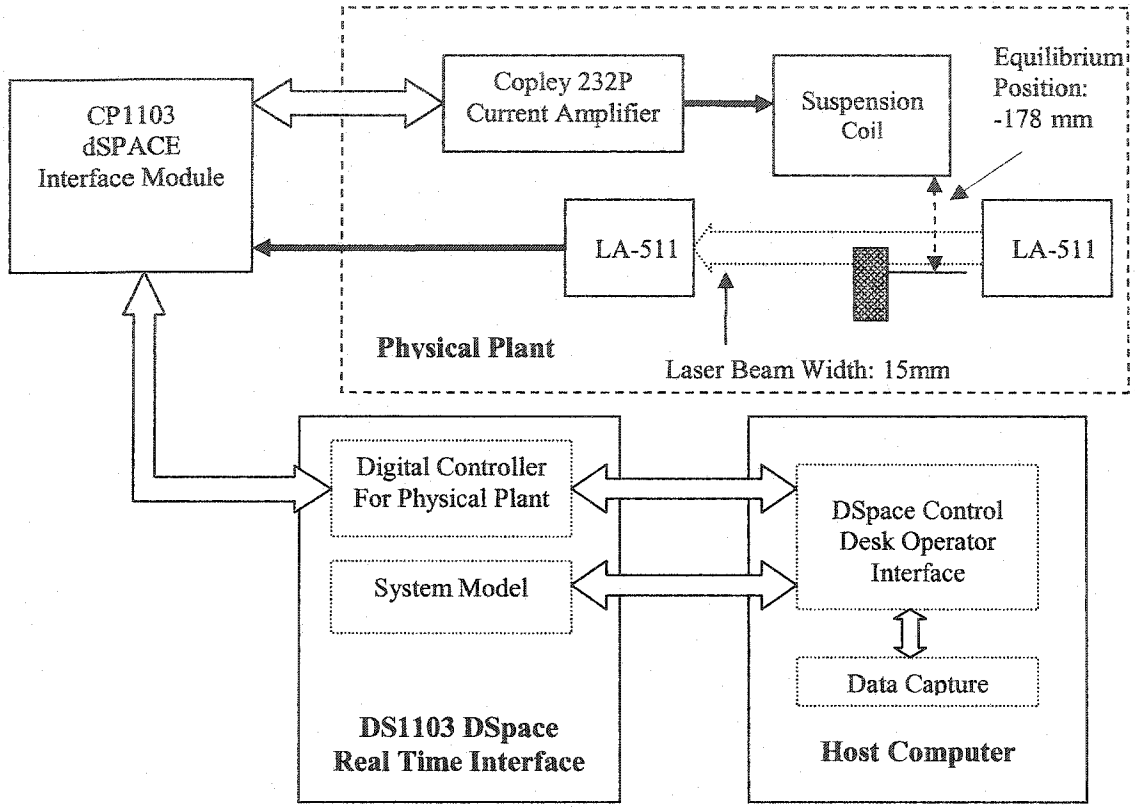


Fig. 18. Schematic diagram of 1-DOF Maglev system using MSBS components.

The 1-DOF setup also employed the CP1103 Interface Module, DS-1103 Real Time Interface and host computer containing the dSPACE and Matlab software. The basic operation involved detecting the suspended core's change in position with the laser sensor pair, changing the sensor output to position feedback to a controller and regulating the current in the suspension coil via the Copley amplifiers. Basically, except for the stainless steel suspension chamber and a full-sized test object, the 1-DOF experimental setup contained parts of the whole MSBS development and operating environment described in Chapter III. Table VI summarizes these components .

TABLE VI
MAJOR COMPONENTS COMPRISING 1-DOF EXPERIMENTAL SETUPS

System Component	1st Setup—Part of Actual MSBS?	2nd Setup--Part of Actual MSBS?
Suspension coils	Yes	Yes
Copley 232P current amplifier	Yes	Yes
Clinton DC power source (and assoc. control circuitry	No	Yes
Levitation environment (chamber)	No	No
LA-511 laser sensors	Yes	Yes
Permanent magnet core	No	No
Simulink-based controller instructions	Yes	Yes
DSPace RTI software	Yes	Yes
DS-1103 Real Time Interface	Yes	Yes
DSpace CP-1103 Interconnection Module	Yes	Yes

Special consideration is given here to the composition of the two different permanent magnetic cores levitated in the two sets of 1-DOF experiments. The first group of experiments employed the same core used in earlier tests, described in [6]. This core was comprised of four small permanent magnet wafers, approximately one centimeter square by 2 millimeters thick. The magnets' magnetization vectors, **M**, were normal to their square surfaces. The several magnets were joined together magnetically, i.e., placed together with their magnetizations aligned, to form essentially a single cubic magnet. The initial value used for this core's volume was 1.6088e-6 m^3, as stated in [6]. Later the core was re-measured the value changed to 1.9531e-6, as shown in the figure.

This permanent magnet core was housed in a small, plastic egg-shaped case to both protect it from damage and to simulate the concept of using the core inside a larger model more suited to wind tunnel testing. Fig. 19 shows the construction of the core and its relationship to the sensor beam and equilibrium position. Note the core is not in the center of the case and thus the centroid of the core is not the same as the centroid of the test object as a whole.

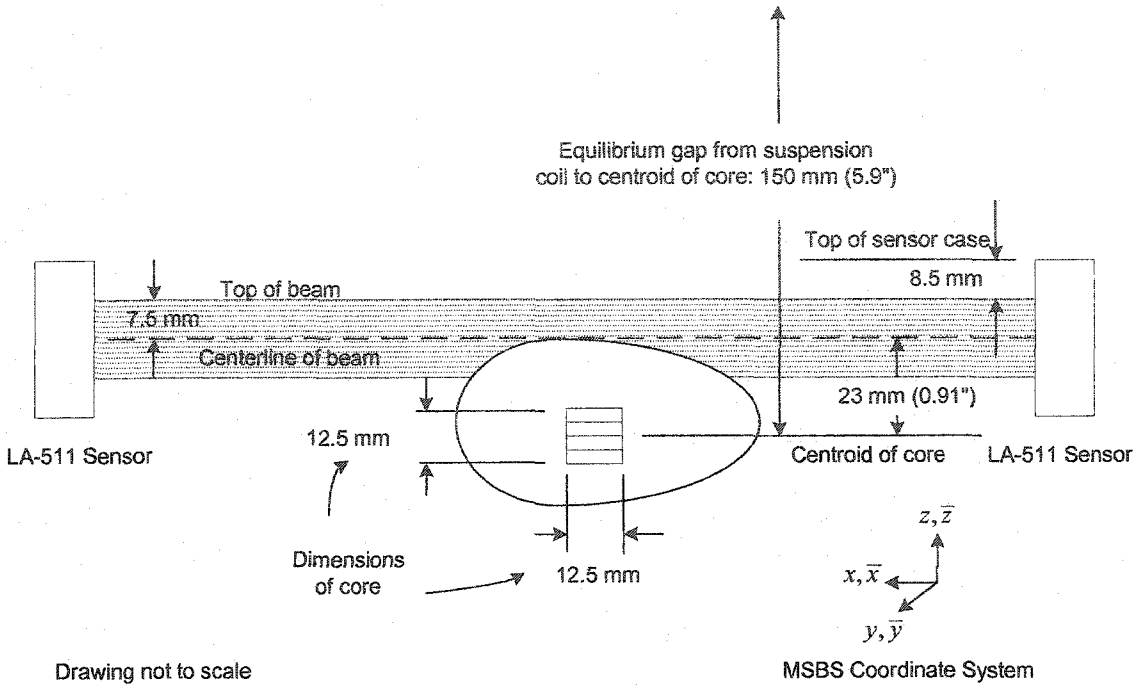


Fig. 19. Permanent magnet core used in first set of 1-DOF experiments.

The second test object and its permanent magnet core were constructed in a similar way, with multiple smaller magnets joined together in a larger plastic case. Fig. 20 illustrates this configuration in relation to the sensor beam and equilibrium position. The second core contains five Neodymnium-Iron-Boron (NdFeB) discs separated by nylon wafers (shaded in the figure). Just as with the first core, the magnetization vectors of the individual magnets are normal to the large surface area so that when joined together the discs form a cylindrical bar magnet with its magnetization vector **M** oriented along its long axis. The magnets are housed inside a PVC tube with a rounded PVC cap with the dimensions of the core and test object assembly shown in the figure. The top is rounded so that it presents a symmetric cross section to the sensor beam. In this way, the system is less sensitive the object's rotation around the z-axis, a motion that occurred frequently and was not controllable in the one-DOF system. This improvement was suggested by the tendency for the egg-shaped test object to induce spurious elevation changes due to its profile change during rotation. Also note that the equilibrium position was lowered for the second core, from 150 mm below the suspension coil to 178 mm.

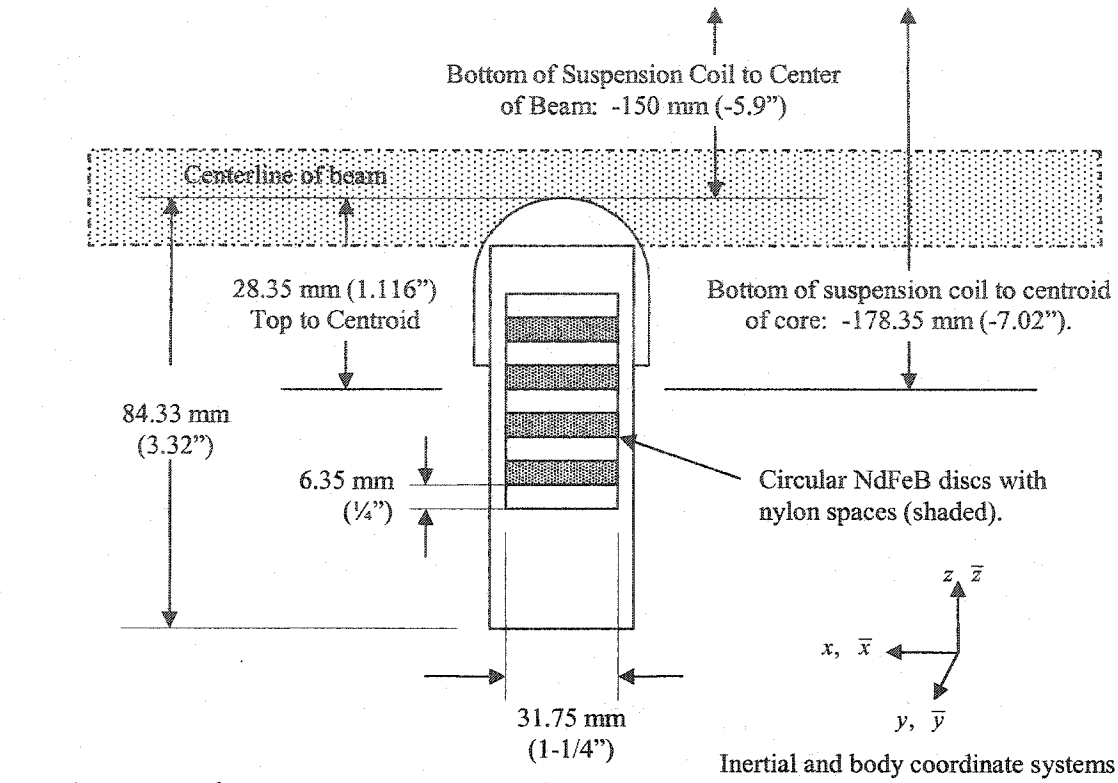


Fig. 20. Permanent magnet core used for second set of 1-DOF experiments.

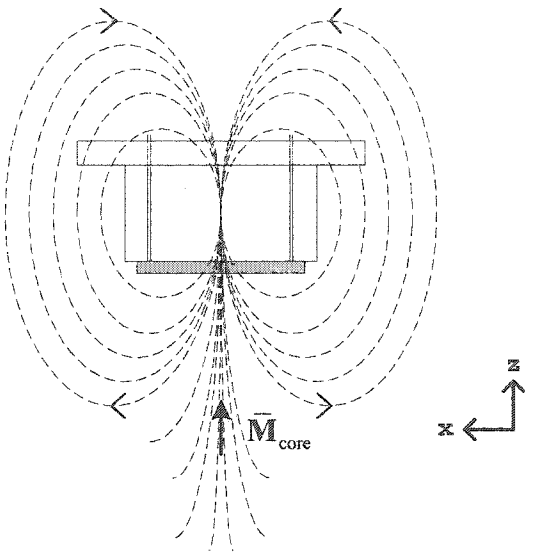
In Chapter II it was pointed out the magnetic torque and force calculations are based on the value of the magnetic flux density at the origin of the inertial coordinate system, which is also the centroid of the core when the core is at its equilibrium. Thus to correctly place the sensors, the position of the centroid relative to the centerline of the sensor beam must be determined for the suspended object's equilibrium position. These measurements are shown in both Figs. 19 and 20. Taking Fig. 20 as an example, note the equilibrium position of the centroid is -178.35 mm, but the point at which the suspended object's position is measured, the object's top in this case, is displaced some 28 mm above the centroid of the core.

Adaptation of the 5-DOF Linear Model

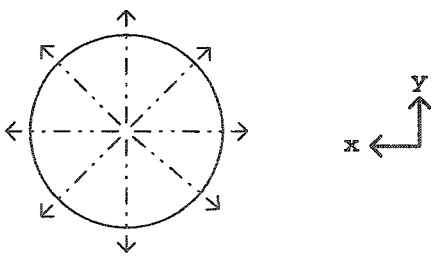
Chapter II explained the underlying electromagnetic theory and provided mathematical models for the 5-DOF MSBS. This section adapts that theory for use in the 1-DOF case, where only the vertical translation of the permanent magnet core is of interest.

In the 1-DOF configuration only one component of magnetic flux density is considered, i.e., B_z. Because the force and torque equations are based on the flux density vectors and their gradients at a single (equilibrium) point, and that point is assumed to lie on the axis of the suspension coil's

core, it is reasonable to assume B_z is effectively the only field component present. This can be seen in Fig. 21, which illustrates the magnetic flux lines and the suspended core's magnetization vector, $\mathbf{M}$, in a single coil configuration.



(a) Side view of coils



(b) Underside of coil

Fig. 21. Magnetic flux lines for 1-DOF configuration.

An additional consideration in adapting the 5-DOF model is that no conversions are needed between inertial and body coordinate systems and $z \equiv \bar{z}$. This is because only movement along the z-axis is involved and the core's magnetization vector remains aligned with the z-axis. As expected, the equations of motion are greatly simplified leaving only the equation for force along the z-axis.

From Chapter II the equation for magnetic force is

$$\bar{\mathbf{F}} = \int_V (\bar{\mathbf{M}} \bullet \nabla)\tilde{\mathbf{B}} dv \,. \tag{16}$$

The integrand can also be written $[\partial\tilde{\mathbf{B}}]\bar{\mathbf{M}}$, where

$$\partial\tilde{\bar{\mathbf{B}}} = \begin{bmatrix} \tilde{B}_{\bar{x}\bar{x}} & \tilde{B}_{\bar{x}\bar{y}} & \tilde{B}_{\bar{x}\bar{z}} \\ \tilde{B}_{\bar{y}\bar{x}} & \tilde{B}_{\bar{y}\bar{y}} & \tilde{B}_{\bar{y}\bar{z}} \\ \tilde{B}_{\bar{z}\bar{x}} & \tilde{B}_{\bar{z}\bar{y}} & \tilde{B}_{\bar{z}\bar{z}} \end{bmatrix} \tag{17}$$

and

$$\bar{\mathbf{M}} = \begin{bmatrix} M_{\bar{x}} \\ M_{\bar{y}} \\ M_{\bar{z}} \end{bmatrix} \tag{18}$$

By assuming the magnetization is uniform throughout the core and noting M_z is the only non-zero component of **M**, equation (14) can be rewritten

$$\bar{\mathbf{F}} = v\left[\partial\tilde{\mathbf{B}}\right]\bar{\mathbf{M}} = M_{\bar{z}}\begin{bmatrix} \tilde{B}_{\bar{x}\bar{z}} \\ \tilde{B}_{\bar{y}\bar{z}} \\ \tilde{B}_{\bar{z}\bar{z}} \end{bmatrix}, \text{ or}$$

$$F_x = vM_z\tilde{B}_{xz}, \tag{80}$$

$$F_y = vM_z\tilde{B}_{yz}, \tag{81}$$

$$F_z = vM_z\tilde{B}_{zz}, \tag{82}$$

where v and M_z are the volume of the core and the core's magnetization vector, respectively, and B_{xz} B_{yz} and B_{zz} are the z-direction gradients of the components of magnetic flux density. Although equations (80) and (81) indicate there may be components of force in the x and y directions, for the 1-DOF experiments B_x and B_y are assumed negligible and F_x and F_y are disregarded. This leaves only equation (82) to form the equation of motion. Also note the over bar notation was dropped to indicate the body and inertial coordinates are the same.

Though the five equations of motion in the 5-DOF case are reduced to only one, two critical modeling assumptions still apply in the 1-DOF case: 1) the force is linear over a small region around the equilibrium point; and 2) the magnetic flux density and its gradients can be expressed as functions of the instantaneous coil currents. This is one key motivation for the 1-DOF tests: validation of the assumptions.

Recall two of the basic assumptions described in Chapter II: 1) in a large-gap maglev system the magnetic flux densities and their gradients are uniform throughout the volume of the suspended core; and 2) the magnetic fields can be reasonably approximated by taking a Taylor series expansion and leaving out second order and higher terms. Also note the method of expressing the magnetic flux density as a function of current in

$$B = \frac{I}{I_{\max}} b_{\max}, \tag{65}$$

where the I is a variable representing the instantaneous control current in a given suspension coil, I_{max} is a constant denoting the maximum coil current allowed and b_{max} denotes the magnetic field produced by I_{max} [10]. Thus the magnetic flux gradient in (74) can be approximated by

$$\tilde{B}_{zz} = \frac{i}{i_{\max}} b_{zz} + \frac{i}{i_{\max}} b_{zzz} \partial z, \tag{83}$$

where b_{zz} and b_{zzz} represent the maximum values of the first and second flux density gradients at the equilibrium position (higher order terms are neglected) and ∂z represents the vertical distance from the equilibrium position. Lower case variables are used because the 1-DOF tests involve only a signal coil current and field, not a composite as in the 5-DOF case. In addition, the over-bar notation is dropped because there is no difference in coordinate systems for the single dimension case. This leads to the following nonlinear equation of motion for the core around its equilibrium point:

$$\ddot{z} = \frac{\nu M_z}{m} \frac{i}{i_{\max}} b_{zz} + \frac{\nu M_z}{m} \frac{i}{i_{\max}} b_{zzz} \partial z - g. \tag{84}$$

The nonlinearity of equation (76) is apparent when it is rewritten as

$$\ddot{z} = \frac{\nu M_z}{m i_{\max}} \left[b_{zz} + b_{zzz} \partial z \right] i - g,$$

where the factor $\left[b_{zz} + b_{zzz} \partial z \right]$ produces the nonlinearity.

A linear equation for the acceleration of the core in the vicinity of the its equilibrium point is derived by taking a truncated Taylor series expansion of (84),

$$\ddot{z} + \partial \ddot{z} = f(z, i) + \alpha \cdot \partial z + \beta \cdot \partial i,$$

where ∂z and $\partial \ddot{z}$ are the deviations in position and acceleration, respectively. Because only the perturbations from equilibrium are of interest in the control system, this equation is simplified to

$$\partial \ddot{z} = \alpha \cdot \partial z + \beta \cdot \partial i, \tag{85}$$

where

$$\alpha = \left.\frac{\delta \ddot{z}}{\delta z}\right|_{z = z_0, i = i_0} \quad \text{and} \tag{86}$$

$$\beta = \left.\frac{\delta \ddot{z}}{\delta i}\right|_{z = z_0, i = i_0}. \tag{87}$$

Equations (85) and (86) are evaluated at the equilibrium position ($z = z_o$) and the corresponding equilibrium current ($i = i_o$), giving

$$\alpha = \frac{vM_z i_o b_{zzz}}{i_{\max} m} \quad \text{and} \quad \beta = \frac{vM_z i_o}{i_{\max} m}\left(b_{zz} + b_{zzz}\partial z\right). \tag{88) (89}$$

The value of i_o .is obtained from equation (85) above by substituting for α and β, setting $\partial \ddot{z}$ and ∂z to zero and solving for i_0 :

$$i_o = \frac{mgi_{\max}}{vM_z b_{zz}}. \tag{90}$$

To simplify the notation, the notation for the perturbations ∂z , $\partial \ddot{z}$ ∂i are shortened to just z , $\ddot{z}$ and ∂i . Then the linearized 1-DOF system can represented in state space form as follows:

$$\dot{\mathbf{z}} = \mathbf{Az} + \mathbf{Bu}, \quad \mathbf{y} = \mathbf{Cz} + \mathbf{Du},$$

or

$$\begin{bmatrix} \dot{z} \\ \ddot{z} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ \alpha & 0 \end{bmatrix}\begin{bmatrix} z \\ \dot{z} \end{bmatrix} + \begin{bmatrix} 0 \\ \beta \end{bmatrix} i, \text{ and} \tag{91}$$

$$y = \begin{bmatrix} 1 & 0 \end{bmatrix}\begin{bmatrix} z \\ \dot{z} \end{bmatrix} + [0]i. \tag{92}$$

Once the values of the constants in (88) and (89) are obtained, a transfer function for the 1-DOF plant can be derived using the above state space relationship. In that this is a single-input, single-output (SISO) system, the transfer function will have the form

$$y = \mathbf{C}(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B} = \frac{\beta}{(s^2 - \alpha^2)}.$$

The following table summarizes the terms used in the development of the 1-DOF equation of motion.

TABLE VII
EXPLANATION OF TERMS IN 1-DOF EQUATION OF MOTION

Term	Explanation
∂z	Perturbation variable representing the change in height of the suspended object with respect to equilibrium.
$\partial \ddot{z}$	Perturbation variable representing the change in acceleration of the suspended object.
v	Volume of the suspended permanent magnet.
M_z	Magnitude of the magnetization vector for the permanent magnet core.
m	Mass of the suspended permanent magnet.
b_{zz}	First gradient of the magnetic flux density b_z of the suspension coil, measured with coil current equal to imax.
b_{zzz}	Second gradient of the magnetic flux density b_z of the suspension coil, measured with coil current equal to i_{max}.
i_{max}	Suspension coil current, set at some value greater than the largest value needed to suspend the permanent magnet anywhere in the desired range. This is the coil current at which the magnetic flux density b_z is determined.
i	instantaneous suspension coil current needed to produce a differential acceleration , i.e. the value that just balances the magnetic force with gravity.
g	Gravitational force constant

Magnetic Field Measurements

Because this was the first attempt to model the MSBS using one of its actual suspension coils and levitating an object at a realistic gap, there no values were available for the magnetic flux density in the region of interest to the 1-DOF experiment. (An earlier 1-DOF experiment described in [6] used the MSBS suspension coil but at a different operating point.) As a preliminary step to this experiment values for b_z, b_{zz} and b_{zzz} were determined using a two-stage process: first the z-component of the magnetic flux density b_z was measured and then the gradients b_{zz} and b_{zzz} were derived for the intended equilibrium points.

Two MSBS suspension coils were supported on a wooden frame as described earlier in the chapter. One of them was energized at $i_{max} = 39.8A$ with polarity such that the magnetic flux

density was oriented in the negative z direction. It should be noted that at the time of the first set of 1-DOF experiments, the 150 VDC Clinton power supply was not on line and a smaller power supply had to be used. This limited the current to well below the capacity of the coil.

Measurements of b_z were taken by attaching the test probe of an F.W. Bell Series 9900 Gaussmeter in a vertical position to a moveable jig calibrated in millimeters. The tip of the probe, held in an upward position in the jig, was placed directly below the center of the core of the suspension coil and moved in ten mm increments from a distance 11 mm below core to a point 291 mm below the core. The z-component of the B-field was recorded at each increment.

The procedure was repeated for the same suspension coil and then, as a cursory check of the uniformity of MSBS coils and amplifiers, the procedure was repeated for a second coil and amplifier. As expected, the measurements were consistent, both between the two sets taken for the first coil and those taken between the first and second coils. Fig. 22 contains plots of the lower portion of the range (-291 mm < z < -81 mm) for all three sets of measurements. In this region the plots were particularly close. Appendix C contains the complete measurement data. These results indicated the same B-field parameters could probably be used for all the coils in the 5-DOF system, and the same input-output characteristic could be used all the amplifiers.

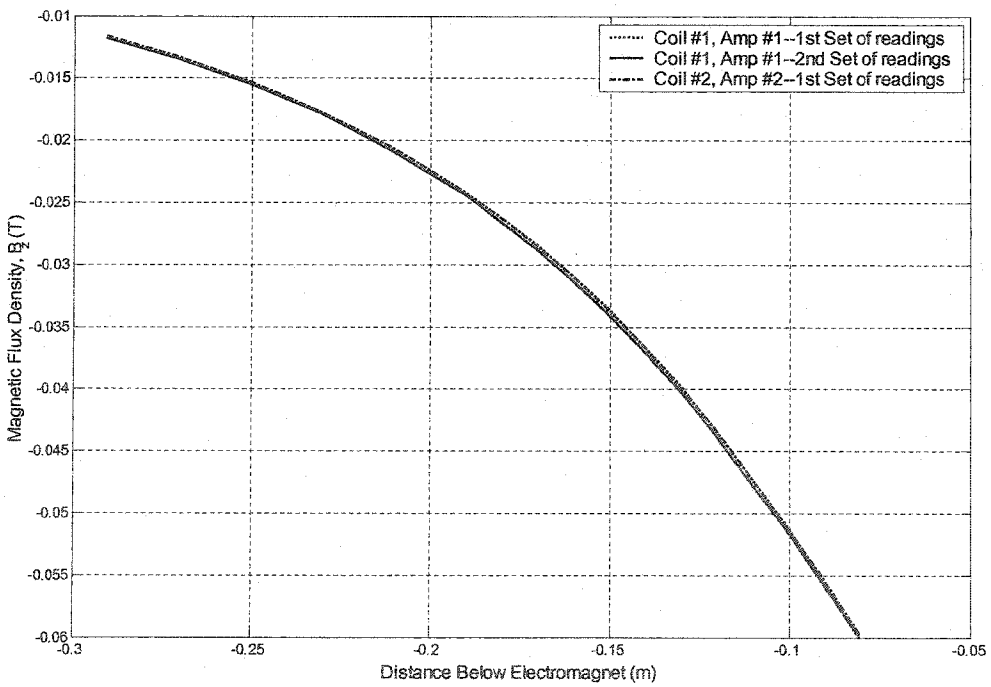


Fig. 22. Comparison of magnetic flux density measurements for MSBS suspension coils 1 and 2 over range -291 mm < z < -81 mm.

The data for coil number one were then plotted and the curve was fitted with a fourth degree polynomial using Matlab's basic curve fitting function. The intent was to obtain a closely fitted curve so that its first and second derivatives could be used to find values for b_{zz} and b_{zzz} at any arbitrary equilibrium point along the –z-axis. The resulting polynomial is shown in Fig. 23, where using the variables x and y provided by the Matlab algorithm represent z and b_z, respectively.

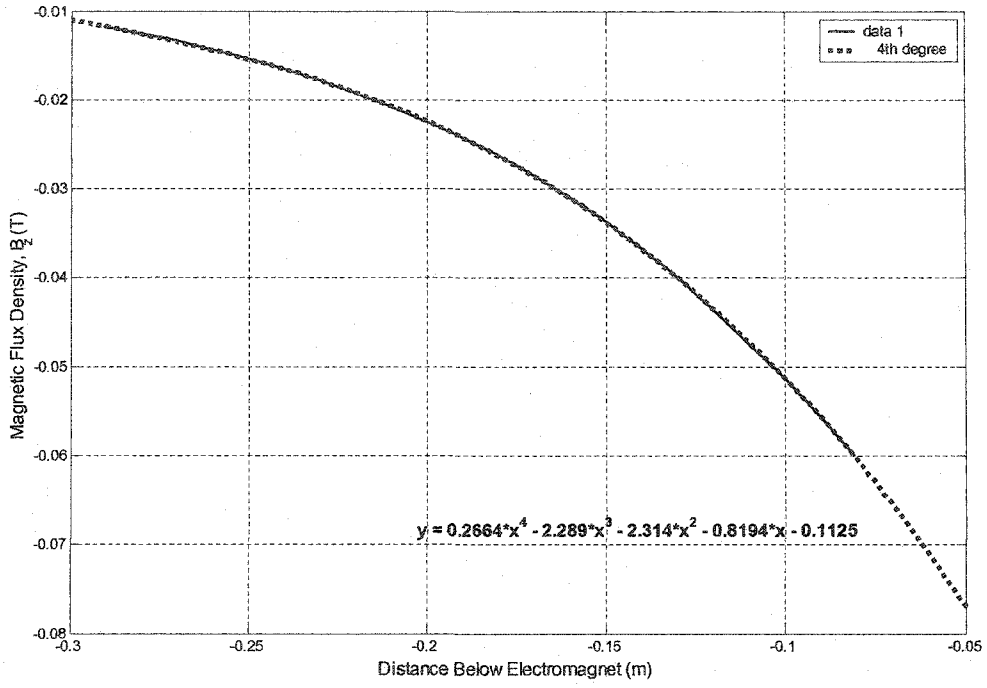


Fig. 23. Plot of measurements and and fourth degree polynomial curve fit for B_z over range -291 mm < z < -81 mm.

For all the 1-DOF experiments the "best fit" polynomial

$$b_z = 0.2664z^4 - 2.289z^3 - 2.314z^2 - 0.8194z - 0.1125 \tag{93}$$

was differentiated and used to calculate the parameters needed to complete the linear model. Fig. 24 illustrates the curves obtained for b_{zz} and b_{zzz}.

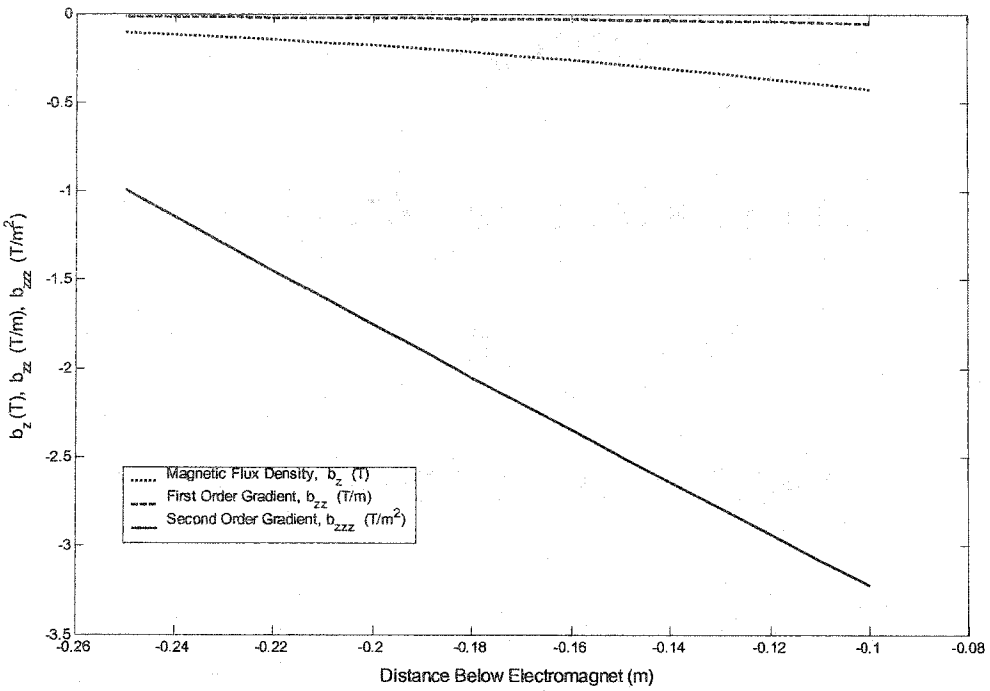


Fig. 24. Magnetic flux density together with first and second order gradients, B_z, B_{zz}, and B_{zzz}, over range -250 mm < z < -100 mm.

One other important finding is pointed out here. Recall the assumption that the instantaneous value of magnetic flux density, as well as its gradients, can be closely approximated by linear functions of the instantaneous coil currents in the vicinity of the equilibrium point. This approximation for the 1-DOF experiments is illustrated in the following expressions:

$$B_z = \frac{i}{i_{\max}} b_z,\ B_{zz} = \frac{i}{i_{\max}} b_{zz} \text{ and } B_{zzz} = \frac{i}{i_{\max}} b_{zzz},$$

where i and $i_{\max}$ are the instantaneous and maximum coil currents, respectively, and B_z and b_z are the instantaneous and maximum B-fields. During the first set of 1-DOF tests, an experiment was conducted in which the B-field and gradients that were calculated using $i = 15.74$ and $i_{\max} = 39.8$, were compared directly to the measured values at $i_{\max} = 15.74$. The results are summarized in Table VIII and, as expected, validate the use of the technique. Plots of the measurements are provided in Appendix C and the Matlab script used to do the analysis is provided in Appendix D.

TABLE VIII
DIFFERENCES BETWEEN MEASURED AND CALCULATED FLUX DENSITIES AND GRADIENTS [a]

Parameter [b]	Numeric Difference	Percent Difference
B_z	0.000152	1.11
B_{zz}	0.0023	1.9765
B_{zzz}	0.0178	1.6164

[a] At equilibrium point z = -150mm.
[b] "Measurements" here refer to the values derived from "best fit" curve obtained from the data plots.

The 1-DOF Controller Design Process

Development of the 1-DOF controller was basically a reiterative process in which a likely compensator was first designed and evaluated using a nonlinear model of the 1-DOF plant. Then the compensator's closed-loop performance with the physical plant was compared to its performance with the nonlinear model. The following paragraphs address several preparatory steps for controller design.

The first step in producing the SISO compensator was to obtain a specific linear transfer function for the plant at its expected operating point. To do this, actual numeric values were needed for the parameters listed in Table VII. Most of these values were obtained directly from component specifications or measurements. The magnetic flux density, b_z, and its gradients, b_{zz} and b_{zzz}, were derived from B-field measurements as explained above. To facilitate a reiterative development process, the known parameter values and the calculations for the derived ones were coded in a Matlab script described below.

The main function of the Matlab script is to use the coefficients of the 4^{th} degree "best fit" polynomial (93), the magnetic core volume, the suspended object's mass (core and housing), the maximum coil current i_{max}, and other given parameters to calculate the state-space form of the linearized plant model. The program uses the Matlab *ss2t.m* command to convert the state-space model to a plant transfer function. In addition, the script calculates the values of constants used in the Simulink non-linear model (discussed below). Finally, all the values are placed on the Matlab workspace for use in subsequent calculations. For example, the state-space model and transfer function for the linear 1-DOF plant with an equilibrium point at -150 mm are shown in the following equations:

$$\begin{bmatrix} \dot{z} \\ \ddot{z} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 86.43 & 0 \end{bmatrix} \begin{bmatrix} z \\ \dot{z} \end{bmatrix} + \begin{bmatrix} 0 \\ 0.4665 \end{bmatrix} i, \tag{94}$$

$$y = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} z \\ \dot{z} \end{bmatrix} + [0] i. \tag{95}$$

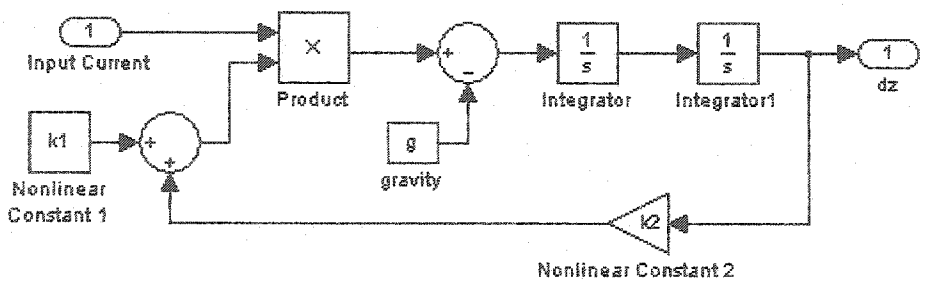
$$G_{\text{maglev}} = \frac{0.4665}{(s+9.297)(s-9.297)} \tag{96}$$

The design procedure also entails evaluating the compensator in a closed loop system with a nonlinear model of the 1-DOF plant. A nonlinear model was constructed from the 1-DOF nonlinear equation of motion, equation (84). Fig. 25(a) shows how this equation was implemented in Simulink. The two constants in the model, k_1 and k_2, represent the constants in the nonlinear equation, i.e.,

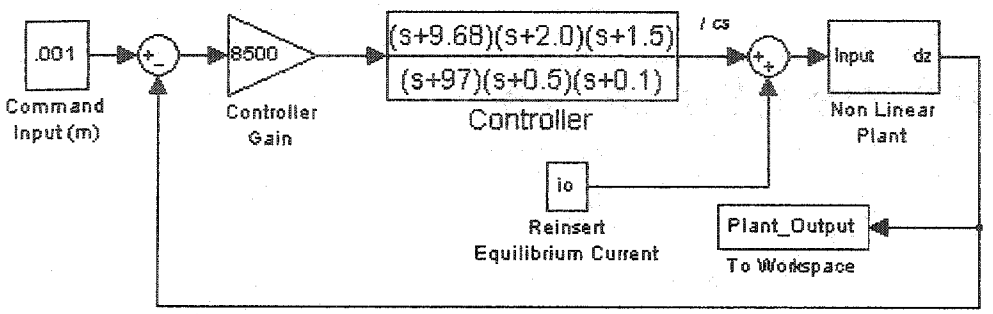
$$k_1 = \frac{vM_z}{mi_{\max}} b_{zz} \text{ and} \tag{97}$$

$$k_2 = \frac{vM_z}{mi_{\max}}. \tag{98}$$

These constants are generated and placed on the Matlab script along with the transfer function and other parameters. The nonlinear model contains two integrator blocks that derive the perturbation position, ∂z, based on equation (84).



(a) Nonlinear 1-DOF plant model



(b) Nonlinear 1-DOF model in closed-loop system

Fig. 25. Simplified block diagram of nonlinear 1-DOF plant and controller.

To test the controller, the nonlinear model is placed as a subsystem in a larger Simulink model of the overall closed-system, illustrated in Fig. 25 (b). The system model shows the poles, zeros and gain for one of the compensators used in the 1-DOF experiments. Also note the block labeled "Rensert Equilibrium Current". Because the compensator is designed around a linearized plant model, i.e., one that describes the plant's operation only within a small region around its operating, or equilibrium, point, the compensator only commands the marginal amount of current needed to restore the plant to its equilibrium. In other words, the compensator doesn't supply the total current i needed in the plant (or nonlinear model) to produce B_{zz} and B_{zzz} according to

$$B_z = \frac{i}{i_{\max}} b_z,\ B_{zz} = \frac{i}{i_{\max}} b_{zz} \text{ and } B_{zzz} = \frac{i}{i_{\max}} b_{zzz}.$$

Therefore, a quiescent bias, or equilibrium current must be added to the compensator output to produce the full control signal necessary to drive the non-linear plant, real or modeled. That is, in the above equation

$$i = i_{eq} + i_{Cs},$$

where i_{eq} and i_{Cs} are the equilibrium and compensator currents, respectively.

Notice there is no corresponding subtraction of equilibrium position from the plant output analogous to the bias current reinsertion. This is because both the linear and nonlinear 1-DOF system equations are perturbation-based, where the deviation from equilibrium ∂z is the variable of interest. Finally, the system model in Fig. 25(b) contains a functional block, "Plant_Output", that places the output of the nonlinear plant, ∂z, on the Matlab workspace during simulation runs.

Initial compensator design was done using Matlab's graphical design tool for SISO systems, Sisotool. Sisotool provides a more-or-less real-time root locus design capability where the changes in the root loci are plotted as the compensator poles, zeros and gains are changed. Fig. 26 illustrates a typical set of views presented by Sisotool for the second order 1-DOF system.

The method for using Sisotool in these experiments was to first run the Matlab script that calculates the constants and plant transfer function and places them on the Matlab workspace. Then Sisotool was launched and the transfer function imported from the workspace and used within the tool to evaluate candidate compensators. When suitable a suitable compensator was identified using root locus and bode plot analysis, the transfer function was exported to the workspace for subsequent evaluation in the Simulink model.

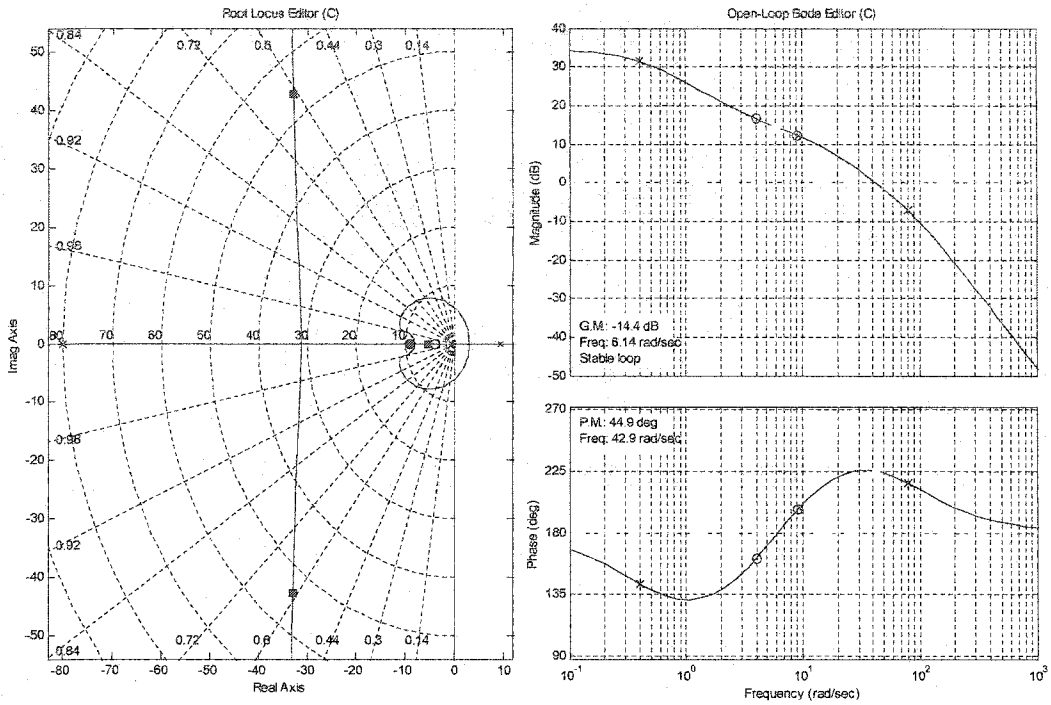


Fig. 26. Typical Simulink-generated root locus and Bode plots for first 1-DOF controller.

Additional Implementation Details

The last portion of this section covers additional implementation techniques needed to produce a working controller for 1-DOF and larger systems. Chapter II presented the basic modeling procedure. Once an adequate compensator is identified, it is placed in a Simulink model containing the blocks needed to provide sensor signal processing, amplifier control logic, coil drive current and operator functions. This larger model is then compiled and loaded on the on the DSpace Real Time Processor (RTP). The remainder of this section will address additional, detailed modeling constructs and instructions needed to set up an operating maglev system.

A simplified, high-level controller block diagram is shown in Fig. 27. The controller contains the following subsystems: 1) the sensor subsystem containing the algorithms to convert sensor outputs to controller feedback signals; 2) the amplifier subsystem containing the logic and voltage conversions to apply the controller output to the Copley current amplifier; 3) the controller turn-on subsystem containing logic needed to detect when the suspended object is within the sensor range; and 4) the compensator subsystem that contains the actual compensator, gain adjustments, and logic to cut the compensator out of the circuit.

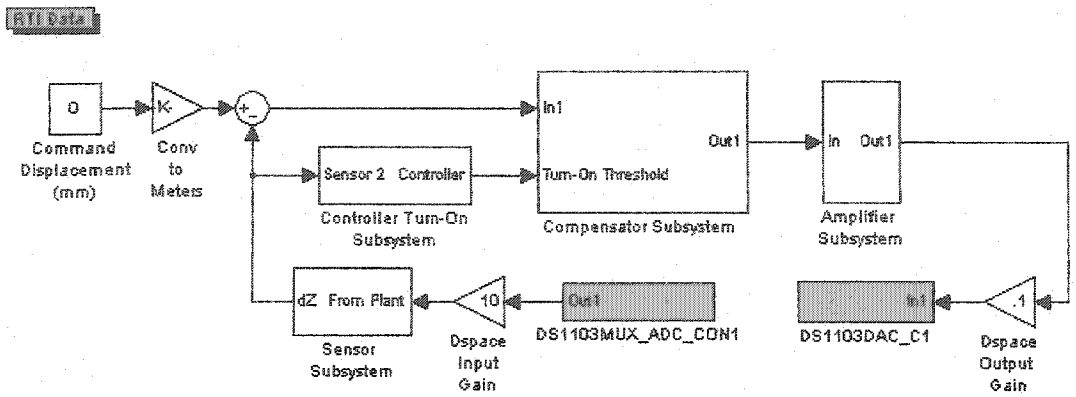
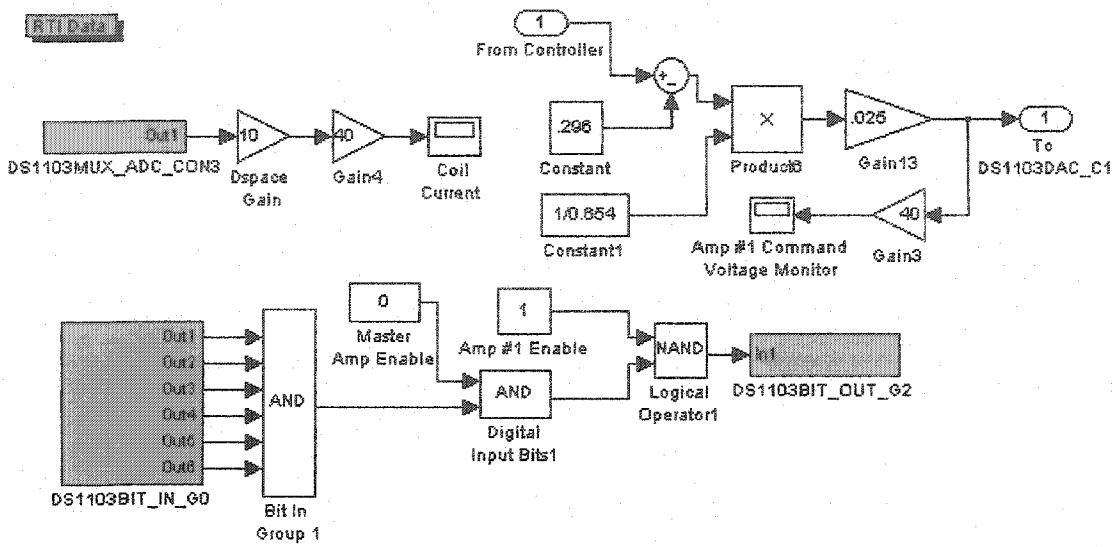


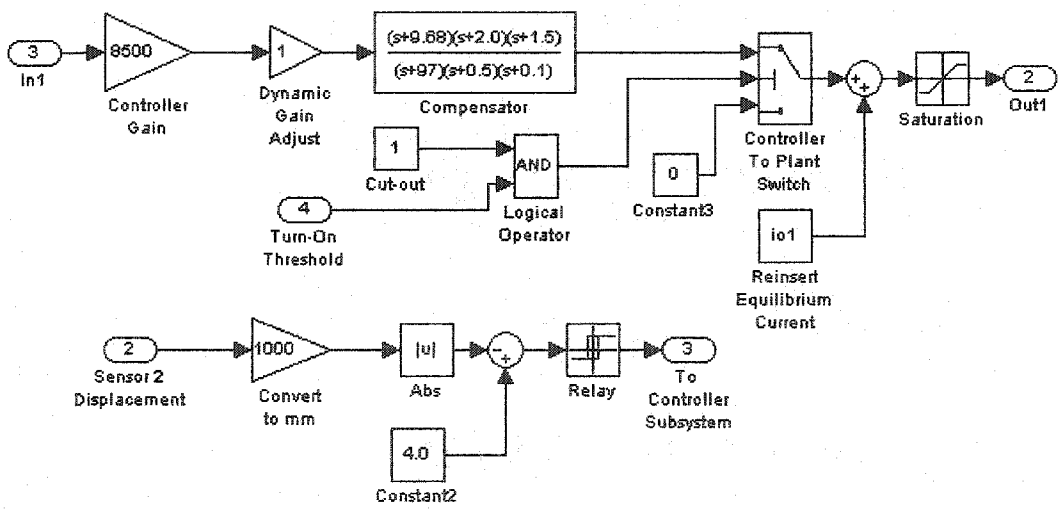
Fig. 27. Simplified Block Diagram of Simulink Controller Model.

Recall from Chapter III that Matlab can be configured to incorporate functions from the dSPACE Real Time Interface (RTI) library, which are incorporated into Simulink as specific modeling blocks. These blocks enable the compiled model running on the RTP to communicate with the real world through the RTI. The shaded blocks in Fig. 27 represent two such RTI functions. The block titled "DS1103MUX_ADC_CON1" denotes a four-part multiplexer on the DS1103 that handles signal from up to four separate analog inputs on the CP1103 Interface Module. The A/D conversion is done in the DS1103. When placing the block in the model, the designer configures it to reference the correct connector on the CP1103. The other shaded block "DS1103DAC_C1" represents D/A conversion on the DS1103 for a single output from the Simulink model. Likewise, this block is configured to reference the correct connector on the CP1103.

Though not dSPACE blocks themselves, two other blocks are needed to implement a real-world interface via the RTI. These are the triangular gain blocks adjacent to the RTI blocks in Fig. 27. These gains are conversion factors required to match the voltage levels actually present in the RTI with the signal levels they represent in the coded algorithms. From RTI to model, the conversion gain is 10, and in the opposite direction it is 0.1. Finally, note the small shaded block in the upper left-hand corner. This indicates the model contains RTI functions.



(a) Selected components of amplifier subsystem



(b) Components of Controller turn-on circuitry

Fig. 28. Selected Blocks of Typical 1-DOF controller model.

Fig. 28. contains portions of two of the subsystems depicted in Fig. 27, selected to illustrate several dSPACE modeling constructs. The first is the shaded RTI block in Fig. 28(a) labeled "DS1103BIT_IN_GO". This block represents the conversion of TTL logic levels on the CP1103 Interface Module to binary ones and zeros used in the RTP. The designer can configure the block to handle to up to eight inputs on the CP1103 by selecting the corresponding pin number on the TTL connector. The designer can also set the block's default logic levels at system turn-on. The other block, "DS1103BIT_OUT_GO", performs the same function in reverse. Because these

blocks handle only binary signals and the machine-to-TTL conversion is hard-wired into the CP1103, accompanying gain blocks are not needed. In Fig. 28(a) the amplifier subsystem, these blocks bring "amplifier ready" signals from the Copley 232P amplifiers and convey the amplifier enable signals to the amplifiers.

Fig. 28 also illustrates several practical engineering techniques, one of which is to provide operators the ability to monitor the MSBS coil currents during actual operation. Note the group of four blocks in the upper left hand corner of Fig. 28(a), including the RTI block "DS1103MUX_ADC_CON3". These convert the coil current monitor signal from the Copley 232P amplifier to digital form and rescale the signal to the actual current level it represents. The physical-to-machine code conversion was discussed in the paragraph on Fig. 27. The rescaling is needed because the Copley current monitor signal is scaled at 0.025 volt/amp, or 1/40 times the magnitude of the actual current. The last block in the conversion process is an artifice by which a Control Desk instrument can be linked to the coil current value. This set of four blocks is replicated for each amplifier in the system.

This same technique is employed in reverse to provide the coil current command signal to the Copley amplifiers, as illustrated in the upper right hand side of Fig. 28(a). Because the Copley amplifier assumes an input signal calibrated at a ratio of 40 Amps/volt, the controller output is converted by the inverse of that ratio, 0.025. (This is a separate and subsequent conversion to the one that compensates for the amplifiers input-output characteristic, discussed in Chapter III.) This functionality is replicated for each amplifier.

Fig. 28(b) addresses another engineering concern that has to do with preventing large excursions in coil current. It was found during the 1-DOF experiments that when the suspended test object moves out of the sensor beam, the controller can drive the coil current up to dangerously high levels. This condition could easily occur if the controller is operating while the test object is being launched. It can also occur whenever the plant does not respond to the controller, such as when the test object falls out of the beam and is no longer controllable. Further, this effect is compounded when the compensator contains poles very near the origin, which causes the compensator to act as an integrator. The application of a constant error signal (from an unresponsive plant) causes the compensator output to rise to high levels.

The remedy to this problem is to keep the compensator out of the loop until the sensors show the suspended object is within a few millimeters of its equilibrium point. This functionality is referred to in Fig. 28(b) as the controller turn-on circuitry and works as follows. First, shown in the lower part of the figure, the magnitude of the suspended object's position is subtracted from a

reasonable boundary value for the turn-on threshold, in this case 4 mm. When the position changes from outside the sensor range to within plus or minus 4 mm of the equilibrium point, this difference becomes greater than zero and triggers the relay output to a logic level one. This logic level is passed to the compensator subsystem in the upper portion of Fig. 28(b), where it is applied to a logic "and" block in combination with a manual controller cut-out switch. When both inputs are high, a logic one is applied to the "Controller-To-Plant Switch", allowing the compensator signal to pass to the amplifier subsystem. This prevents the compensator from driving the control current up before the plant is ready to be controlled.

However, this functionality only solves part of the problem. To prevent the threshold detector from turning off when the object moves outside the 4 mm boundary, thus limiting the 15 mm MSBS operating range, the relay is set to turn off at a threshold lower than its turn on boundary. That is, the relay does not turn off until the difference between the turn-on boundary and the absolute value of the position falls to negative 3.5 mm. This keeps the compensator out of the control loop until the test object moves to within 4 mm of equilibrium and then takes it back out of the loop only if the object moves out of the sensor range.

It may be noted there is a saturation block following the compensator in Fig. 27(b) that also limits coil current excursions. However, this alone cannot mitigate the over-current problem described above. First, the saturation range must be set wide enough to accommodate large control current transients. Second, limiting the excursion range does not in and of itself prevent current buildup when the test object is out of the beam, nor does it prevent transients produced when moving the object within the beam manually.

Finally, once a compensator is designed, and a Simulink model of the control system is compiled and loaded onto the RTP, a GUI is designed using Control Desk to allow operators to both control the experiment and capture the necessary data. The graphic in Fig. 29 is a screen shot of a basic control panel for the 1-DOF experiment. As described in Chapter II, the virtual instruments on the panel are linked to the appropriate variables in the RTP.

Fig. 29 contains several instruments used in all the MSBS experiments covered in this thesis. For example, the two numeric displays in the upper left of the figure show the raw voltage of the sensor output and the object's displacement from equilibrium. On the right side of the figure there are four data plotting instruments that display the input and output signals for both the nonlinear model and the physical plant, enabling a side-by-side comparison compensator's performance in both the modeled and real system. As noted in Chapter III, variables linked to these instruments are automatically available for data capture.

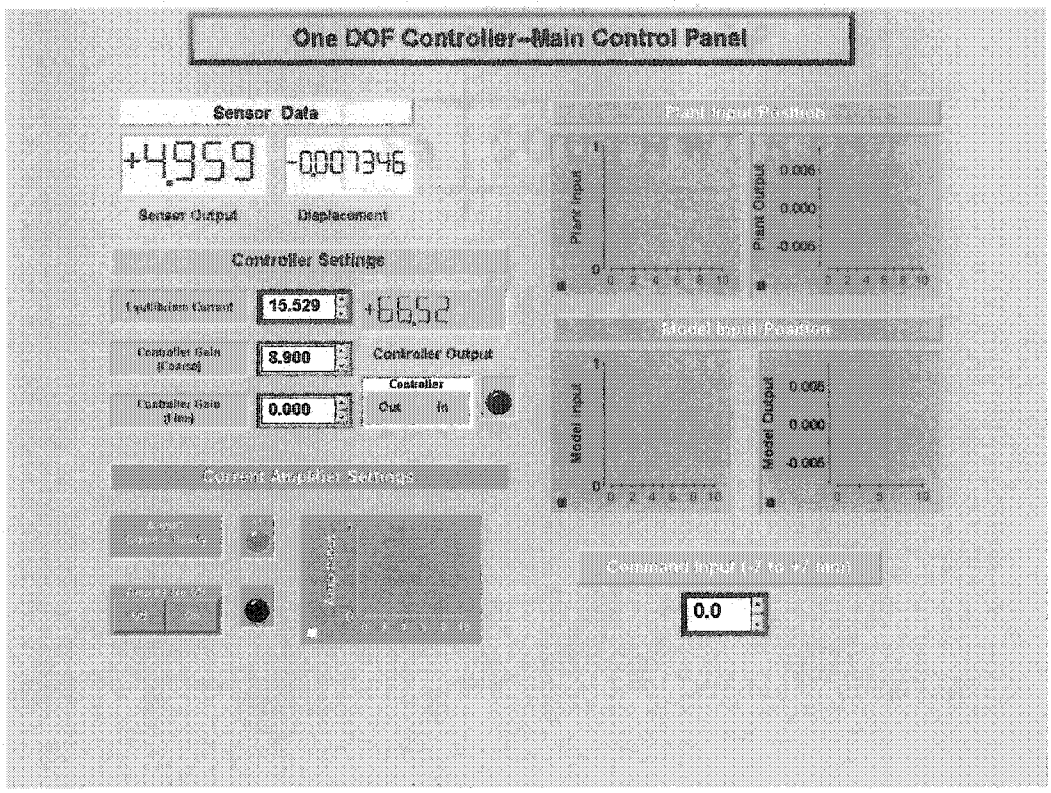


Fig. 29. Screen capture of main operator control panel for joint 1-DOF test.

The control screen also contains several simulated thumbwheel switches and push buttons. These provide the operator with compensator gain adjustments, commands to change the desired displacement of the test object and on-off switches for the amplifier and the controller. The amplifier cutoff switch was particularly important as a safety measure.

The modeling constructs and design solutions presented in this section are some of the basic ones needed to set up controller experiments, and the sample block diagrams are only simplified versions of the more complex ones actually used. Further techniques and more detailed Simulink models will be presented in the ensuing discussions of the actual models and specific experiments.

Initial 1-DOF Experiments

This section describes the first 1-DOF experiment: simultaneous testing of the MSBS model and physical plant. The section builds on the basic design methods presented thus far by providing more specific information on the specific 1-DOF controller and Simulink model used in these experiments.

The motivation for this experiment was two-fold: 1) to validate the MSBS modeling assumptions using major MSBS components, and 2) to identify new implementation issues and gain practical experience operating the hardware and software. The first goal would be addressed by comparing the performance of a compensator in both the 1-DOF nonlinear model in Simulink and the physical 1-DOF system. The second goal would be met in the course of setting up the software and hardware. As it was pointed out earlier in the chapter, lessons learned during this experiment and the next were the stimulus for much of the discussion earlier in the chapter.

This section first covers the experimental setup and compensator design, including the use of Sisotool and the composition of the Simulink model. Following that the experimental procedure is presented and then a discussion of the results, emphasizing their relevance to the full MSBS

Experimental Setup

The 1-DOF maglev for this experiment was designed to control the vertical position of an object within a 14 millimeter range centered at 150 millimeters below the core of a larger electromagnet, as illustrated in the schematic diagram in Fig. 30. (This experiment used the first suspended object, i.e., the egg-shaped case and permanent magnet core described in detail in Fig. 19.) The full 15 mm range was not attempted because it was uncertain at that point how well the sensors conversions would work at the range extremes. Otherwise, the experimental setup was the same as depicted in Fig. 18.

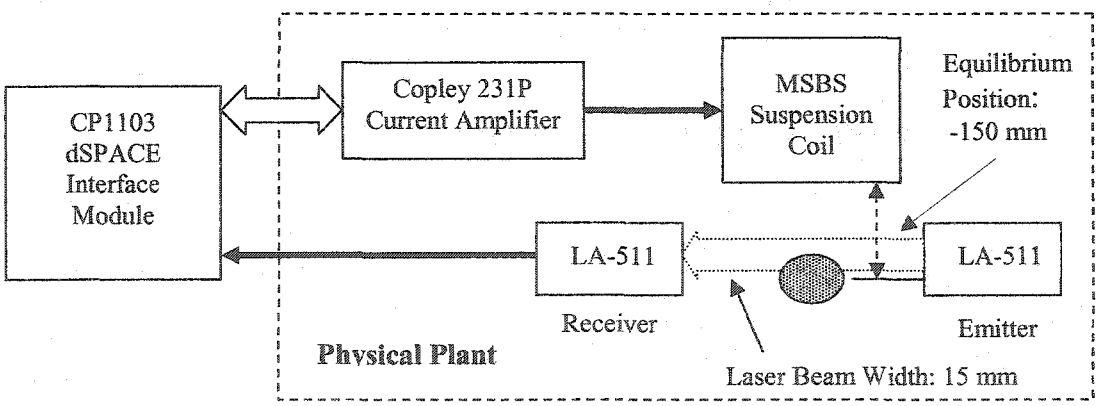


Fig. 30. Schematic diagram of first 1-DOF maglev plant.

To design the controller, it was first necessary to obtain values for the physical constants and develop a linear model of the plant. This was done following the approach of the previous section, i.e., using a Matlab script to calculate the values and place them on the Matlab workspace.

The calculations were based on an equilibrium position at 150 millimeters below the suspension coil, a distance approximating the suspension position planned for the actual MSBS. First the functions for the magnetic flux density gradients, b_{zz} and b_{zzz}, were obtained by taking the derivatives of the "best-fit" polynomial for b_z. The equations for the derivatives were then solved for z_o = -150mm. Then equilibrium current was calculated and the coefficients for the state space equation derived using b_{zz}, b_{zzz}, i_o, M_z, m and v. Also, the parameters needed for the nonlinear model, k_1 and k_2, were calculated and placed on the Matlab workspace. The resulting values are listed in Table IX. Note in the table that during the course of this experiment, several values were revised in an effort to understand how sensitive the system model was to parameter variations. Appendices C and D contain the measurement data, the curves and the Matlab script used to derive the quantities in the table.

TABLE IX
MODELING PARAMETERS FOR ONE DOF MAGLEV—EXPERIMENT NO. 1

Parameter	Values
Mass of suspended permanent magnet, m	0.0235 kg
Equilibrium Distance, z_o	-0.150 m
Maximum coil current, i_{max}	39.8 A
Coefficients (descending order) of "best fit" plot of Bz	0.2664, -2.289, -2.314, -0.8194, -0.1125
First gradient of magnetic flux density at z_o, b_{zz}	-.2833 T/m
Second gradient of magnetic flux density at z_o, b_{zzz}	-2.4960 T/m^2
Magnetization of permanent magnet (initial), M_z	-9.5731e 5 A/m
Volume of permanent magnet, v (initial)	1.6088e-6 m^3
Equilibrium current, i_o (initial)	21.028 A
First coefficient, α (initial)	86.4283
Second coefficient, β (initial)	0.4665
Nonlinear model constant, k_1	0.4665
Nonlinear model constant, k_2	4.110
Parameters Revised Due to Increased Permanent Magnet Volume [a]	
Adjusted volume of permanent magnet, v	1.9531e-6 m^3
Equilibrium current, i_o	17.3216 A
First coefficient, α	86.4283
Second coefficient, β	0.5663

[a] Note, during the experiment the equilibrium current was adjusted from the calculated value in an effort to get a better response from the physical plant.

TABLE IX, CONT'D

Parameter	Values
Parameters Revised Due to Both Increased Magnet Volume and Magnetization Vector [a]	
Magnetization of permanent magnet, M_z	-1.1392e 6 A/m
Volume of permanent magnet, v	1.9531e-6 m^3
Equilibrium current, i_o	14.556 A
First coefficient, α	86.4283
Second coefficient, β	0.6740

[a] Note, during the experiment the equilibrium current was adjusted from the calculated value in an effort to get a better response from the physical plant.

The Simulink model constructed for this set of experiments is shown in Fig. 31. Fig. 31 (a) is a block diagram of the overall joint 1-DOF model. Note the nonlinear plant model and controller are almost totally separate from the physical plant and its (identical) controller. This of course was done to enable the simultaneous, independent compensator testing. Fig. 31(b) is an expanded view of the "Nonlinear Plant and Controller" block in Fig. 31(a).

Several of the blocks in Figs. 31(a) and (b) have been discussed already. Additional blocks to note include those labeled "Gain Adjust", which were inserted to allow the compensator gain to be adjusted in both coarse (x 1000) and fine (x 10) increments while the controller was in operation. (Note "coarse" and "fine" are defined relative to the baseline compensator gain, K_c = 8900, calculated in Sisotool. This functionality was useful in tuning the controller to the plant once a suitable set of compensator poles and zeros were identified. Also note the "Command Input" block in Fig. 31(a). When linked to the GUI, this permitted a common step input to both the model and physical system. Finally, it can be seen Fig. 31 does not contain the controller turn-on threshold circuitry described in the previous section. This functionality was not implemented until the second set of 1-DOF experiments.

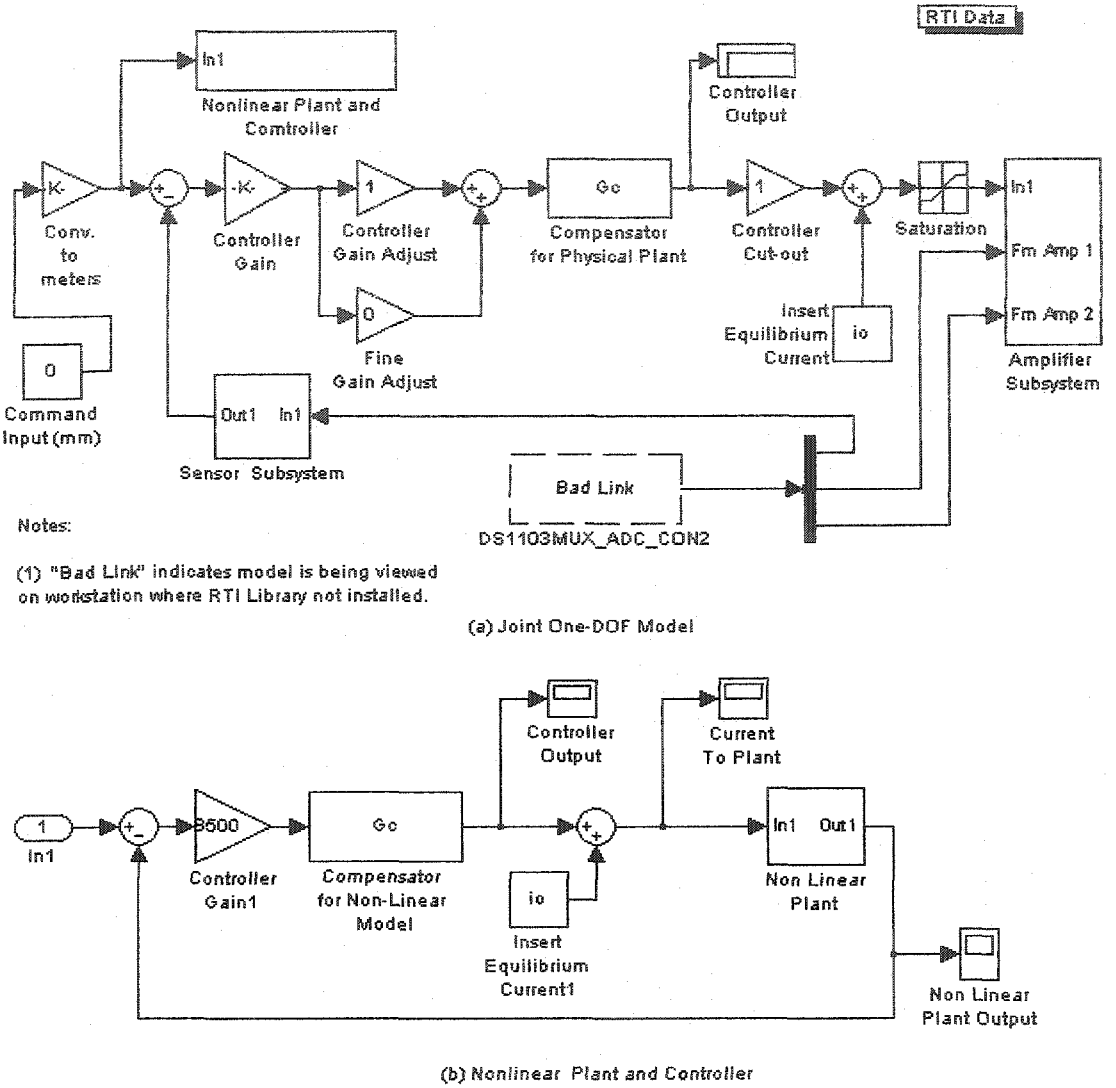


Fig. 31. Simulink model used for first set of 1-DOF experiments.

After deriving the linear plant model and its transfer function, Matlab's Sisotool was used to design a lead-lag controller for the 1-DOF plant. In view of the primary goals of this experiment, validating the modeling assumptions in the context of actual MSBS equipment and gaining expertise in building the full MSBS system, the objective of this controller design was simply to obtain a level of performance and robustness sufficient to conduct a rough side-by-side comparison of the plant and model. In this way at least the controllability of the real system could be gauged and any major discrepancies in the model could be identified.

As described in the 1-DOF controller design process above, the linearized plant transfer function was calculated and placed on the Matlab workspace:

$$G_{\text{maglev}} = \frac{0.4665}{(s+9.297)(s-9.297)}. \tag{96}$$

Equation (95) was imported into Sisotool to begin the controller design. The transfer function shows the plant was unstable with poles at s = ± 9.3, so to start the design the plant was stabilized with negative output feedback and a simple lead compensator, $K_c = 8900 \times \frac{(s+9.2)}{(s+92)}$. This controller stabilized the plant but did not provide an acceptable steady state response to a step input as indicated in Sisotool's closed-loop step response plot. Various combinations of lead-lag controllers were tried while observing then root locus, bode and step response plots as pictured in Figs. 32 and 33.

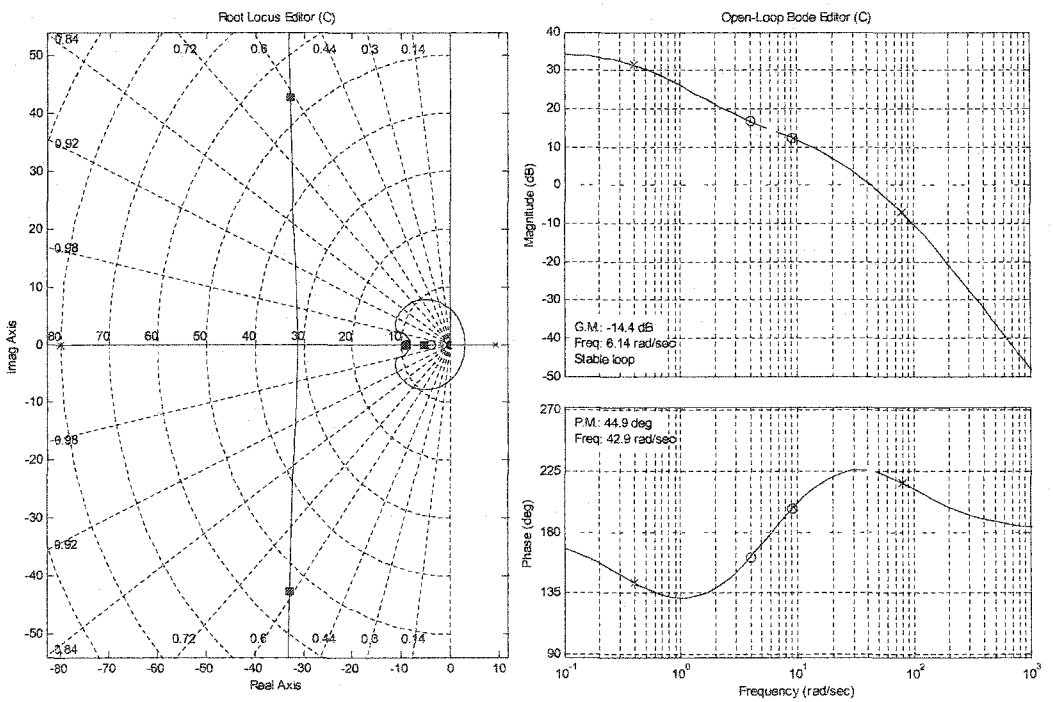


Fig. 32. Root locus and Bode plots for first 1-DOF controller.

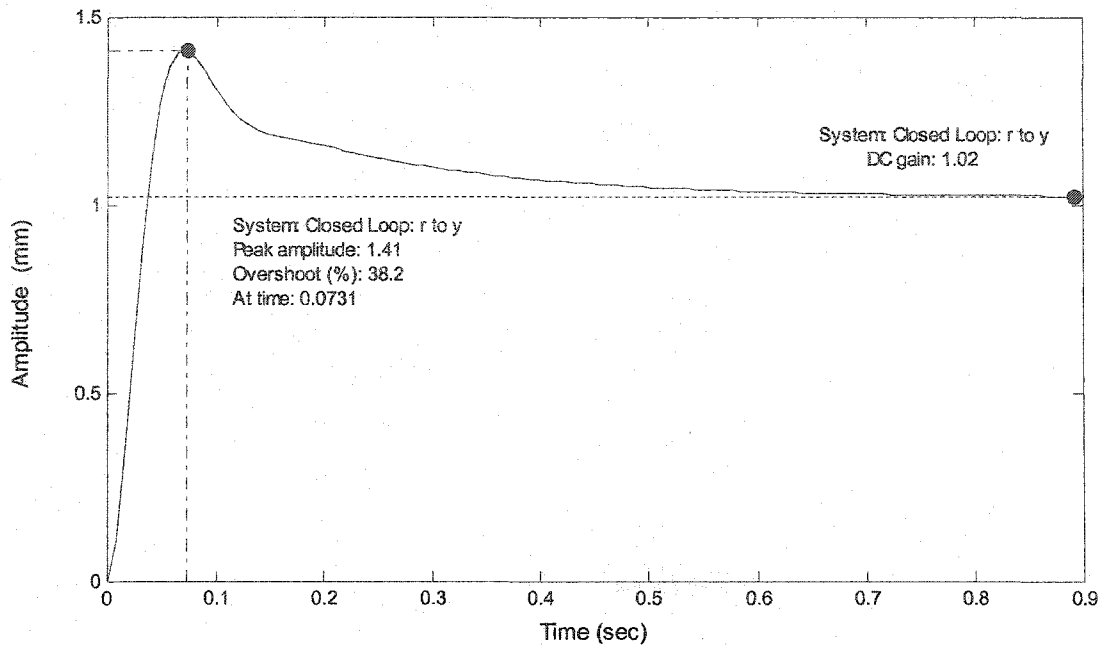


Fig. 33. Simulated step response for first 1-DOF controller.

After several iterations of pole-zero placement and gain settings, a compensator was selected to begin the experiment. Subsequently the plant model had to be changed slightly, but it turned out the same compensator was suitable for each transfer function. The several plant transfer functions and controllers are listed in Table X.

TABLE X
LINEAR PLANT TRANSFER FUNCTIONS AND CONTROLLERS FOR EXPERIMENT NO. 1

Parameter Set [a]	Linear Plant Transfer Function	Controller
Initial	$G_{maglev}=\frac{0.4665}{(s+9.297)(s-9.297)}$	$K_c=8900\times\frac{(s+9.2)(s+4.0)}{(s+80)(s+0.4)}$
First Revision	$G_{maglev}=\frac{0.5663}{(s+9.297)(s-9.297)}$	Same as above.
Second Revision	$G_{maglev}=\frac{0.6740}{(s+9.297)(s-9.297)}$	Same as above.

[a] Note, during the experiment the equilibrium current was adjusted from the calculated value in an effort to get a better response from the plant.

Experimental Procedure

The basic procedure for conducting the 1-DOF experiment consisted of loading 1-DOF application in dSPACE and launching the test object. Once the system was stable at equilibrium, a range of step inputs were commanded via the Control Desk GUI while the inputs and outputs of both the model and plant were sampled.

The experiment was place in operation as follows:

1) Load compiled Simulink model, called the "application" in DSpace, onto the RTP using the Control Desk software.

2) Verify links between the Control Desk screen and the variables.

3) Turn off amplifier enable and controller cut-out switches on the GUI.

4) Energize amplifier and sensor power supplies.

5) Align sensor emitter and receive pair and adjust gain to provide a full-beam output of 5.0 V.

6) Remove amplifier inhibit on the Copley 232P front panel and set controller gain according to the compensator transfer function. (See Table X.)

7) The Copley amplifier and compensator were brought on line and the test object was launched in the sensor beam.

Once the test object was stable in the sensor beam, a series of five-second data captures was undertaken over a range of command inputs from minus seven to plus seven millimeters. A one-millisecond sample period was used with a down sampling rate of 100, resulting in ten data points per second.

Both the nonlinear model and physical plant tracked the command input very closely in that each one millimeter command input produced a one millimeter increment in output. (See plots in discussion on results.) However, the physical plant produced a consistent positive 0.66 mm offset in its output over the entire range of command inputs. It was also noted the compensator did not adjust the coil current to the level of the calculated equilibrium current values, even with a zero command input. In fact, the compensator output remained between -5.5 and -7.2 A throughout the experiment, indicating the compensator was continually seeking to drive the plant input to an unattainable equilibrium point. Further, it was observed that as the bias current was lowered (using the virtual dial on the main operator's screen) the compensator output moved toward zero and position decreased. It was found a new equilibrium current setting of 15.74 A allowed the

compensator output to return to zero at equilibrium and caused the position offset to go to zero. These observations plainly indicated the calculated equilibrium current was too high for the actual plant.

Several secondary experiments were then performed to investigate the apparent miscalculation of equilibrium current. The current equation itself was not in question since it had been successfully employed in several one and five DOF experiments. (For example, see [4].) For this reason the investigation focused on verifying the values for the constants in the equation, which is repeated here for convenience:

$$i_o = \frac{mgi_{\max}}{vM_z b_{zz}}. \tag{90}$$

The first constant to be considered was the magnetic flux density gradient b_{zz}. To verify this value a secondary experiment was devised where another set of B-field readings were taken with the coil current set at what appeared to be the correct equilibrium current, 15.74 A, the values for b_z, b_{zz} and b_{zzz} were plotted. These values very closely approximated those obtained from the calculations

$$B_z = \frac{i}{i_{\max}} b_z,\ B_{zz} = \frac{i}{i_{\max}} b_{zz} \text{ and } B_{zzz} = \frac{i}{i_{\max}} b_{zzz}.$$

This sub-experiment confirmed the correct values for flux density had been used and that the field measurement technique was correct. These results were addressed in more detail under the discussion on B-field measurements in the section on Experimental Preliminaries.

Attention was next turned to the physical constants associated with the test object: total mass (m), core volume (v) and magnetization (M_z). While these values had appeared to be correct in an earlier experiment, the write-up did allow that they were manufacturer's specifications at best in [6]. A multi-step investigation, beginning with the easiest step, was performed to verify the values:

1) Verify mass of test object.

2) Calculate the minimum error in v and M_z needed to produce the apparent discrepancy in equilibrium current.

3) Disassemble test object and re-measure volume of permanent magnet.

First, the total mass of the test object was easily confirmed to be within 0.1 grams of the stated value. Then a set of calculations was performed using a Matlab script to determine if a small error in v or M_z could produce the six ampere discrepancy in equilibrium current. The reasoning was that if realistic errors in the nominal manufacturer's specifications could account for the discrepancy, then it would be reasonable to revise the given values and proceed with the tests. This would obviate the need to disassemble the test object and measure the core, or worse, try to measure the core's magnetization.

The Matlab script calculated equilibrium current using values for v and M_z incremented over a certain range in 2% gradations. There were 42 pairs of v and M_z that satisfied the equation

$$\left|i_{eq} - 14.59\right| \leq 0.2, \tag{99}$$

where i_{eq} is the calculated equilibrium current and 14.59 A is the new estimated plant equilibrium current. Unfortunately, the results showed the values for both v and M_z would have to be at least 10% and 30% larger, respectively, to account for the discrepancy in current. Therefore, it was decided to disassemble the sealed test object and measure the volume of the magnet.

The volume of the core was rechecked and found to be 1.9531e-6 m^3 versus 1.6088e-6 m^3 as originally thought. This 21.4 percent increase represented a surprisingly large difference in measurement per side. Nevertheless, the revised volume measurement was taken as the correct value, resulting in a new equilibrium current of 17.3216 A. The parameter values obtained using the new volume are shown in the second group in Table IX.

The Matlab test script was then modified to vary only the magnetization M_z. With the gradations for M_z at 2%, a magnetization value of -1.1392e 6 A/m was found to produce an equilibrium current of 14.56 A, within the window defined by (99). This showed that an error as small as 16% in the nominal magnetization value could explain the remainder of the current discrepancy. The parameter values obtained using the revised values for volume and magnetization are shown in table IX. Notably, the table shows the plant's poles as reflected in α did not change with the decreasing current, only the plant's DC gain.

A second run of the 1-DOF experiment was done to confirm whether the errors in volume and magnetization constituted a credible explanation for the discrepancy in the equilibrium current. Because the plant's poles did not change, a change in compensator was not anticipated, except for possibly the gain. For this reason, only the plant parameters, accessible on the Control Desk GUI, and not the nonlinear model were change for this test. When the test object was launched, it was

found an equilibrium current of 14.63 A and compensator gain of 8900 (same as for the previous test) produced the least offset between model and plant positions. (Note that 14.63 is within the 0.2 A selection window defined in (98).) With the equilibrium current to the plant set at 14.63 A, input and output data for both plant and model were captured for a range of inputs -7 mm to +7 mm. The data plots for this experiment are presented in Figs. 34 and 35.

Experimental Results

This section concludes with a presentation of the data and discussion of the findings from this experiment. To assist in analyzing the data, Matlab scripts were written to extract the sampled data points from the .mat files created by the data capture function in Control Desk. A total of 51 points per variable per command input were stored in Matlab vectors for convenient use in plotting and analysis.

The upper plot in Fig. 34 shows the sampled input currents for both the model and plant. One obvious result is the plots of the model input are fairly uniformly spaced. However, analysis of the values shows a slight widening in the spacing with increasing coil current. For example, between command inputs of 6 and 5 mm, the input differed by 0.1712 A. This difference grew to 0.2136 A, 24% increase, for command inputs -6 and -7 mm. (Higher current corresponds to increasingly positive command inputs.)

A second prominent feature is the noisiness of the plant inputs. To smooth out the noisy signals, Matlab script calculated the sample mean for the 51 data points for each data capture. The lower plot in Fig. 34. shows the sample mean for the input for each command input. An unexpected result was the large difference in model and plant input values. An analysis of the data points shows that for a command input of zero, the mean model input was the calculated equilibrium current, 21.028 A, as expected. But the corresponding mean plant input was 14.556 A.

Fig. 35. illustrates the outputs of the model and physical plant over the command input range. The plots depict a nearly linear response, also borne out by analysis of the data, which shows the increments in outputs to be uniform to within less than 2%. In fact, the tight tracking between command input and system output is within the theoretical steady state response to a .001 m step input obtained from Matlab. However, a serious problem with the plant output was its consistent offset of 0.658 mm. It is probably because of this offset the plant could not accommodate a command input of +7 mm, since the command input and offset combined would push the object out of range of the sensor.

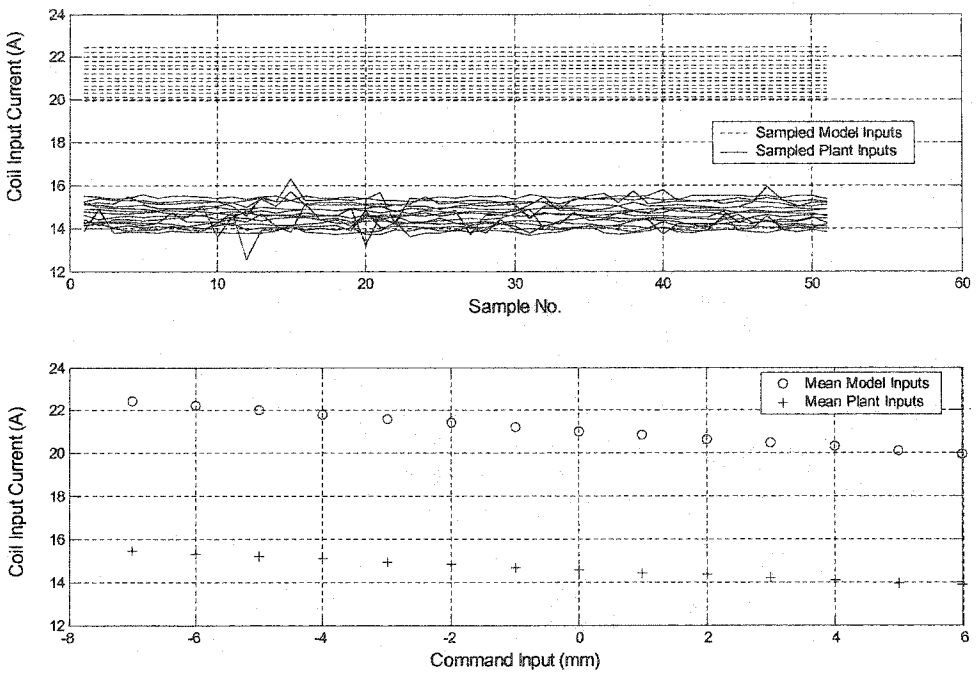


Fig. 34. Comparison of sampled data points and sample means for model and plant inputs for commands ranging from -7 mm to +6 mm.

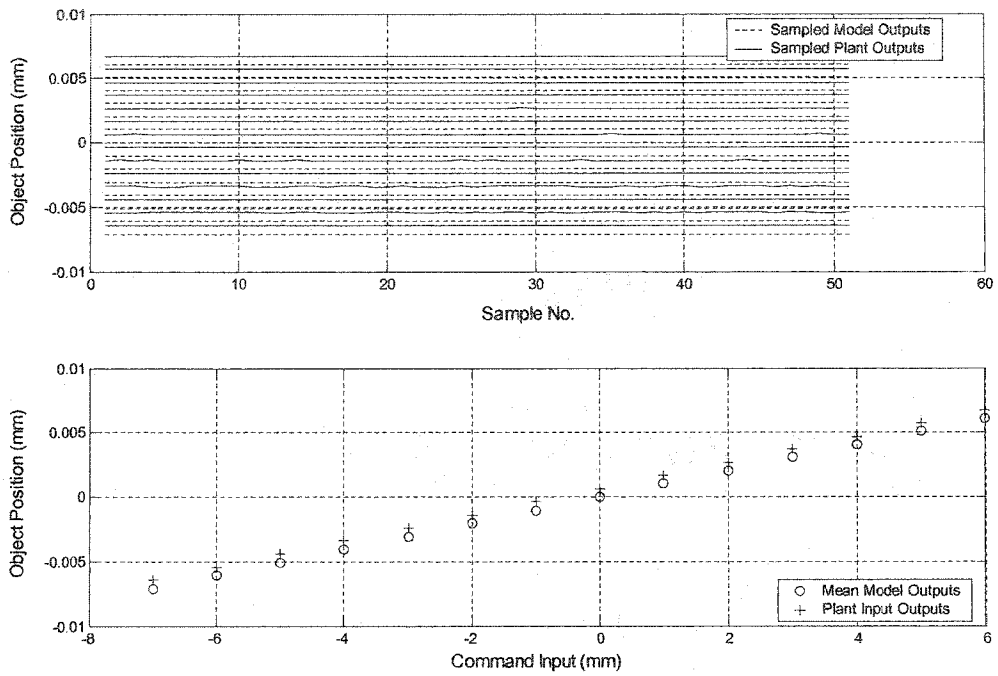


Fig. 35. Comparison of sampled data points and sample means for model and plant outputs for commands ranging from -7 mm to +6 mm.

The second round of tests using the revised equilibrium current level produced a marked improvement in positioning accuracy with a position offset less than 0.05 mm, as illustrated in Figs. 36 and 37. The plots of the model and plant and inputs were nearly the same as for the original equilibrium current, which was expected because only the plant parameters were changed. but the output plots were much improved, as shown in Fig. 37.

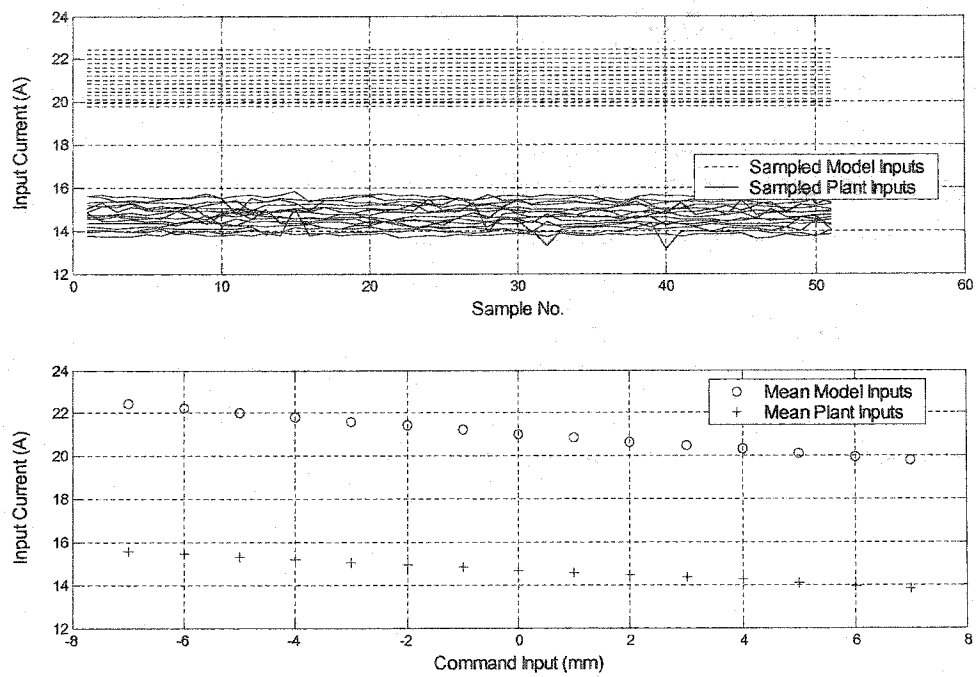


Fig. 36. 1-DOF model and plant inputs for revised equilibrium current.

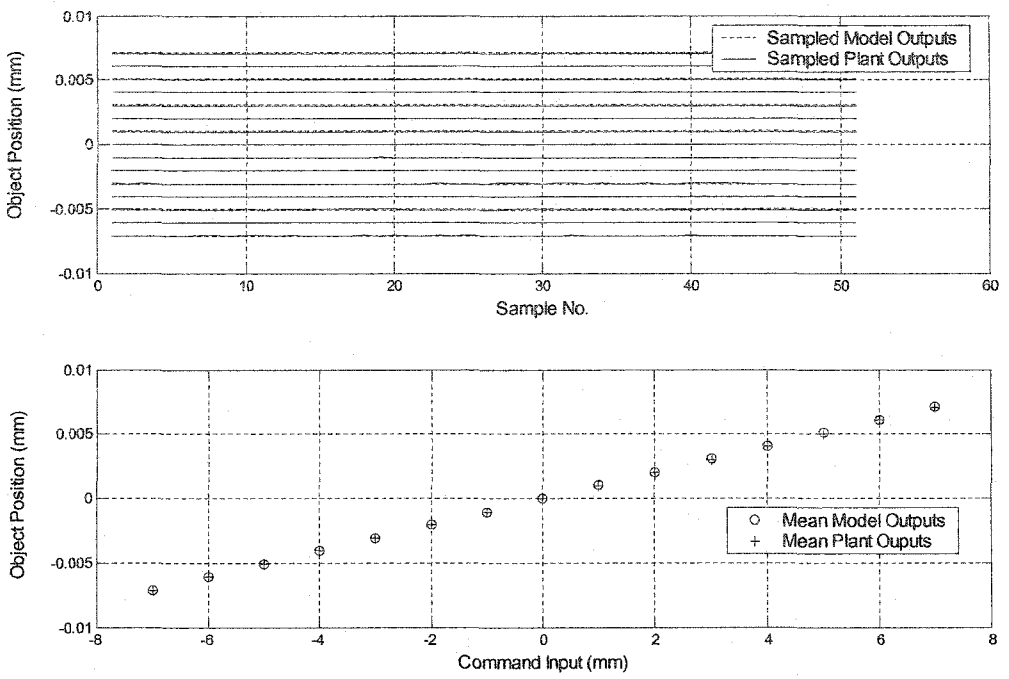


Fig. 37. Comparison of model and plant outputs for revised equilibrium current.

Second Set of 1-DOF Experiments

This section describes a second set of 1-DOF experiments undertaken to refine the 1-DOF compensator. Similar to the first set of experiments, this one involved a reiterative process of controller design and closed-loop evaluation, except that this time the evaluation used only the physical plant and not the nonlinear model. The rationale was that because the first experiments showed the linearized transfer function would readily provide a workable compensator—using the revised value of M_z—it was not necessary to evaluate candidate compensators against the model to get them into the ballpark, so to speak. Also, during this set of tests, the 1-DOF setup was altered slightly in anticipation of later 2-DOF tests.

Having demonstrated the viability of the basic MSBS modeling approach, the primary motivation for this group of experiments was to investigate the potential for improving the lead-lag controller already developed. If performance improvements could be realized using the same modeling assumptions and parameters as before, that would further demonstrate the validity of the range of linearity in the force equation, linearity of the sensors, accuracy of the physical parameters such as flux density, magnetization and equilibrium position. In addition, a favorable outcome would

also be a strong indicator the MSBS model incorporated all the significant system dynamics. (Of course, it us understood at this point in the development the set of dynamics does not include the effects of the eddy currents in the stainless steel suspension chamber.) A secondary motivation is to provide more experience with the hardware and software; some of the practical issues discussed in this section were presented in the experimental preliminaries section.

In this section, the experimental setup is explained first, then the experimental procedure, and finally the experimental results are presented with an emphasis on their relevance to the design of the full MSBS.

Experimental Setup

The setup for this set of experiments was similar to that used in the first set. The same overall system configuration as in Fig. 18 was used. The second, larger test object was employed, as described in the section on experimental preliminaries and illustrated in Fig. 20. Because of the new object's larger size, the equilibrium point was moved from -150 mm down to -178 mm, and as a result, the sensors and their mounting rail were lowered. As indicated above, additional changes in anticipation of the 2-DOF experiments included an additional sensor mounted below the second suspension coil (see side-by-side coil positions in Fig. 17.), and additional Simulink blocks to process the second sensor output signals. However, these additions were not used during these experiments

These experiments followed the same modeling techniques described earlier in the section on experimental preliminaries. A MATLAB script was used to calculate certain parameters and obtain transfer functions for the linearized plant. The modeling parameters used in this set of experiments are listed in Table XI. Note the equilibrium current was revised once during the procedure.

TABLE XI
MODELING PARAMETERS FOR ONE DOF MAGLEV—EXPERIMENT NO. 2

Parameter	Values
Mass of suspended permanent magnet and case, m	0.192 kg
Equilibrium Distance, z_o	-0.178 m
Maximum coil current, i_{max}	39.8 A
Coefficients (descending order) of "best fit" to plot of Bz	0.2664, -2.289, -2.314, -0.8194, -0.1125
First gradient of mag. flux density at z_o, b_{zz}	-.2185 T/m
Second gradient of mag. flux density at z_o, b_{zzz}	-2.0769 T/m^2
Magnetization of permanent magnet, M_z	-9.7085 e 5 A/m

TABLE XI, CONT'D

Parameter	Values
Volume of permanent magnet, v	16.09 e-6 m^3
Equilibrium current, i_o [a]	21.97 A (Initial) 21.55 A (Revised) [a]
First coefficient, α	93.2566
Second coefficient, β	0.4466

[a] During the experiment the equilibrium current was reduced slightly from the calculated value to get a better response from the plant.

As before, Matlab's Sisotool was used to design an output feedback compensator for the plant. However, a controller more accurate than the previous one was sought, in particular one which, after demonstrating stability and robust gain and phase margins, limited the percent steady-state value, percent overshoot and control input. An accurate steady-state value was important for two reasons. First, fairly precise positioning is required, given the scale of the models intended for use with the MSBS. Second, the ability to tune the system to produce small steady-state errors over its operating range tends to show the viability of the linearized plant model and other assumptions used to design the controller. This is, as stated above, one of the primary goals of this set of experiments. The percent overshoot is important because the test object's movement is tightly constrained by the width of the sensor beam. (At an elevation of .006 m from equilibrium, a percent overshoot of only twenty-five percent could cause a command input of plus .001 m to momentarily move out of the beam, which could irrecoverably disrupt the feedback loop.) The magnitude of the control input was constrained in this experiment by the capacity of the power supply feeding the Copley current amplifiers. The power supply's upper limit was 40 Amps, so given the quiescent current of 22 Amps, large deviations of input current could not be tolerated. Finally, rise time and settling time were of small concern because of the intended operational setting for the MSBS, i.e., tests running 48 hours or longer.

The design involved several iterations of parameter choice using Sisotool, followed by testing on the 1-DOF plant. This entailed finding the best trade-off between opposing system characteristics: steady-state value, percent overshoot and maximum deviation of control input current. The first step in the design was to bring down the steady-state error by inserting an additional lag element in the compensator close to zero. Then the compensator gain and pole-zero locations were varied

to find a "baseline" compensator giving acceptable gain margin (GM), phase margin (PM), percent overshoot, steady-state value and maximum input current. This compensator was tested on the actual 1-DOF system.

During the latter part of the design process, the gain and the two lag parts of the compensator were adjusted in small increments around the baseline to produce the best set of responses. A Matlab script was used to produce a series of 1-DOF systems based on 81 pole-zero permutations in double-lag compensator. The most dynamic trade-off was seen between percent overshoot and maximum input current while adjusting gain. Also, shifting the lag compensators' poles and zeros slightly toward the origin reduced the percent overshoot, but increased the steady-state error slightly. Finally, a compromise in pole-zero selection was reached and tested in the actual 1-DOF plant. This compromise is reflected in the transfer function

$$9000\times\frac{(s+9.68)(s+2.0)(s+1.5)}{(s+97)(s+0.5)(s+0.1)} \tag{100}$$

The values in Table XII illustrate the main trade-off in pole-zero location for the lag compensators. Note the first three rows contain pole-zero combinations producing the lowest percent overshoot. Unfortunately, the steady state values for these compensators had nearly 8.5% error. The last three rows contain pole-zero combinations resulting in the lowest steady state values. However, the percent overshoot increases by 60% for these combinations. The middle row contains the values used in the compromise, those in equation (100).

TABLE XII
DESIGN TRADE-OFFS USING DIFFERENT DOUBLE-LAG COMPENSATORS [a,b]

z1	p1	z2	p2	%OS	Steady State	Max Input Current	GM	PM
-1.5	-1.0	-1.0	-1.0	25.5280	1.0840	9.0000	-13.0869	52.1019
-1.5	-1.0	-1.5	-1.0	25.5280	1.0840	9.0000	-13.0869	52.1019
-1.5	-1.0	-2.0	-1.0	25.5280	1.0840	9.0000	-13.0869	52.1019
-2.0	-0.5	-1.5	-0.1	38.7670	1.0039	9.0000	-12.0396	49.9625
-2.5	-.01	-1.0	-0.1	40.4091	1.0001	9.0000	-11.8222	49.2159
-2.5	-.01	-1.5	-0.1	40.4091	1.0001	9.0000	-11.8222	49.2159
-2.5	-.01	-2.0	-0.1	40.4091	1.0001	9.0000	-11.8222	49.2159

[a] Lag compensators shown in columns 1-4 are combined with lead compensator $9000\times\frac{(s+9.68)}{(s+97)}$ to obtain open and closed-loop system responses in the other columns.

[b] Not all pole-zero combinations that were evaluated are shown.

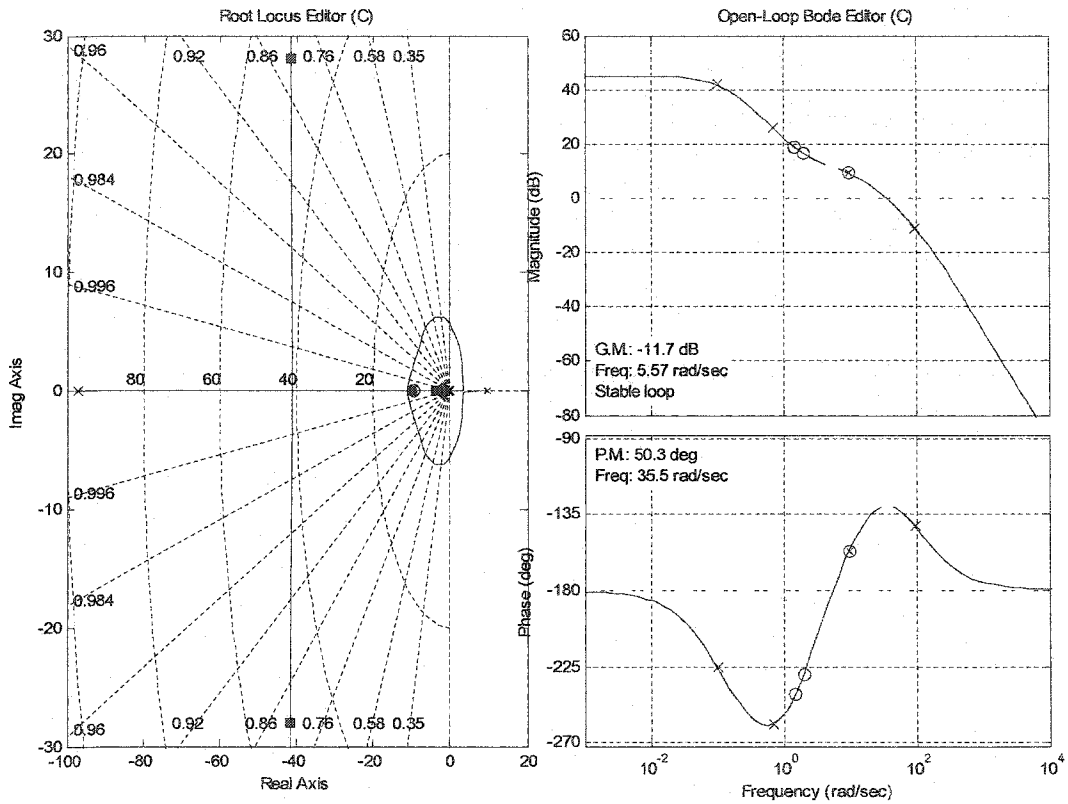


Fig. 38. Root locus plots and bode magnitude and phase plots for second one-DOF controller.

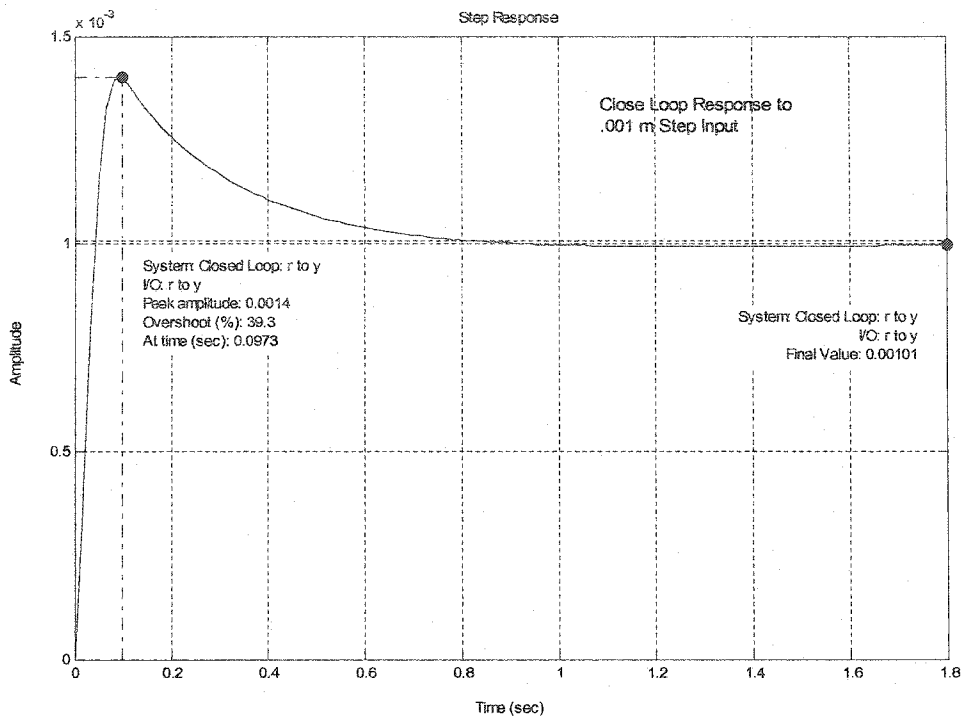


Fig. 39. Sisotool generated plot of step response for second one-DOF controller.

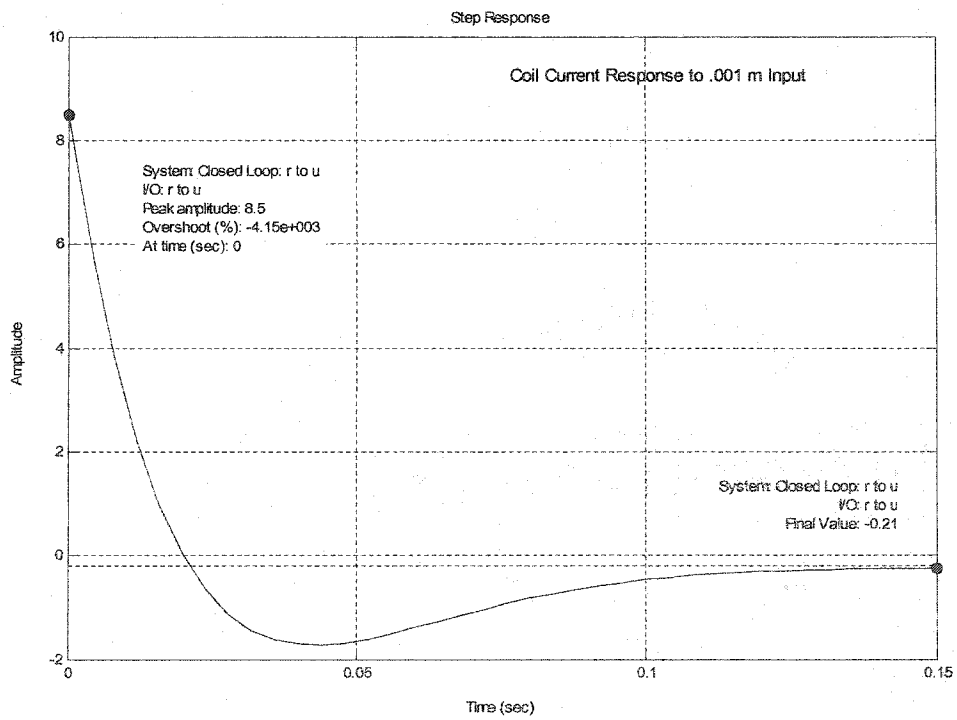


Fig. 40. Sisotool generated plot of control input for second one-DOF controller.

Once an acceptable controller was designed, the Simulink model depicted in Fig. 41. was completed and compiled to run on the dSPACE RTP. The model contains blocks to provide tandem control of a second 1-DOF levitation using another sensor pair, amplifier and suspension coil. As noted above these blocks were inserted in preparation for 2-DOF experiments as well as to allow a second coil to be tested. The figure also includes the controller turn-on threshold circuit, whose operation was discussed earlier. As the experiment was being set up, it was found this latest controller, with a pole at -0.1, behaved like an integrator, whose output voltage rose to significant levels if a large error voltage was applied for longer that fifteen or twenty seconds. As a result, the controller was found to build up dangerous voltage levels during the time it typically took for one person to launch the test object. Finally, while the controller turn-on threshold was being tested, it was found the calculated equilibrium current, 21.97 A did not allow the controller output to fall to zero when there was no error input. The equilibrium current was lowered to 21.55 A, which produced the smallest controller output at equilibrium.

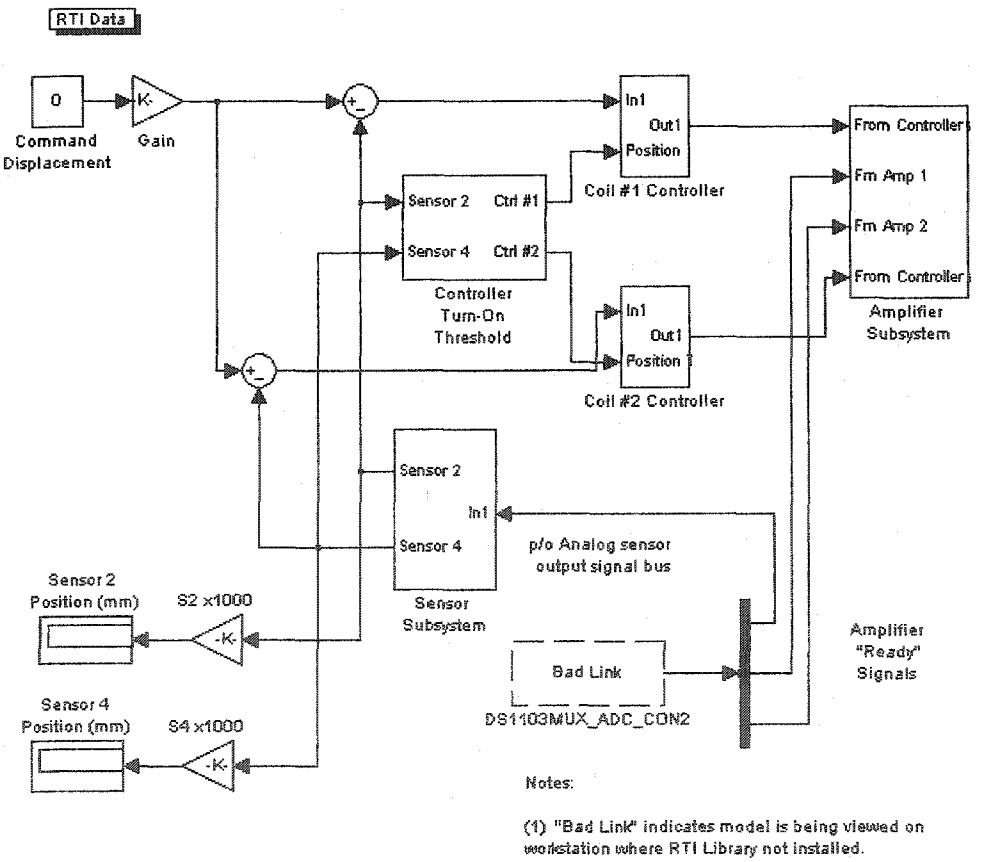


Fig. 41. Controller model for second set of 1-DOF experiments.

For this experiment, the Control Desk GUI was set up to capture the controller output (the input to the plant) and the plant output. (Sensor pair #2 and coil #1 were used.) Twelve ten-second data captures were taken while position change commands were issued to the plant via the GUI. The sample period was .001 s and a down sampling rate of 10 was used, resulting in 1001 data points per sample interval. The commands were given about two or three seconds into each capture interval to allow the plots to give a good before-and-after comparison of the controller and plant behavior. As before, the data were stored in .mat files for later extraction using a Matlab script.

Experimental Results

Table XIII summarizes the data capture sequence and results. There were ten one-mm and two five-mm step inputs. It was found the system could not tolerate step inputs larger than 5 mm, which did not come as a surprise given the large percent overshoot calculated during the design. (With a 38% overshoot, a step command of six mm results in a momentary position change of 8.28 mm, well out of the sensor range.) The mean plant inputs and outputs were based on the last five seconds (500 data points) in each capture, the idea being that the system would have settled from the step input before then.

TABLE XIII
SYSTEM PERFORMANCE MEASUREMENTS FOR DOUBLE-LAG COMPENSATOR

Capture No.	Command Input (mm)	Mean Plant Input Minus Equilibrium Current (A)	Mean Plant Output (mm)	Plant Output from Theoretical Steady State Value (mm)	Percent Difference
1	0 to 1	-0.6531	1.003862	1.0039	0.003737
2	1 to 2	-0.8446	2.007358	2.0078	0.021990
3	2 to 3	-1.0099	3.010614	3.0170	0.212088
4	3 to 4	-1.1871	4.013903	4.0156	0.042277
5	4 to 5	-1.4073	5.017771	5.0195	0.034448
6	0 to 5	-1.3630	4.995589	5.0195	0.478639
7	0 to -1	0.1275	-1.004479	-1.0039	0.057710
8	-1 to -2	0.3306	-2.008083	-2.0078	0.014099
9	-2 to-3	0.5184	-3.012125	-3.0170	0.161840
10	-3 to -4	0.7120	-4.015911	-4.0156	0.007752
11	-4 to -5	0.9385	-5.020060	-5.0195	0.011169
12	0 to -5	0.9206	-4.994255	-5.0195	0.505469

The following observations can be made from the table. First, the change in coil current is not symmetric around the equilibrium point, which could indicate one or more of a variety of miscalculations or mistaken measurements. Most likely this was related to the need to adjust the equilibrium current down from its calculated value. Second, there is a very small percentage difference between the observed steady state values and those calculated using the theoretical steady state value obtained in the design. More experimental results are presented in Figs. 42 through 45, which illustrate the plant inputs and outputs over the range of input commands.

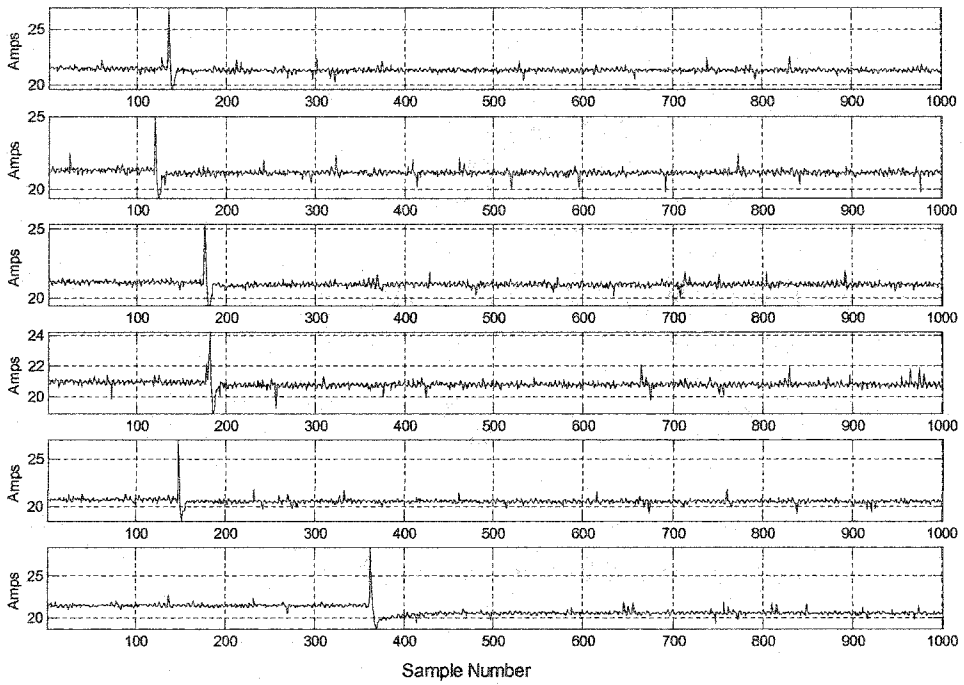


Fig. 42. Plots of plant inputs over a ten-second interval for step input commands ranging from +1 mm to +6 mm.

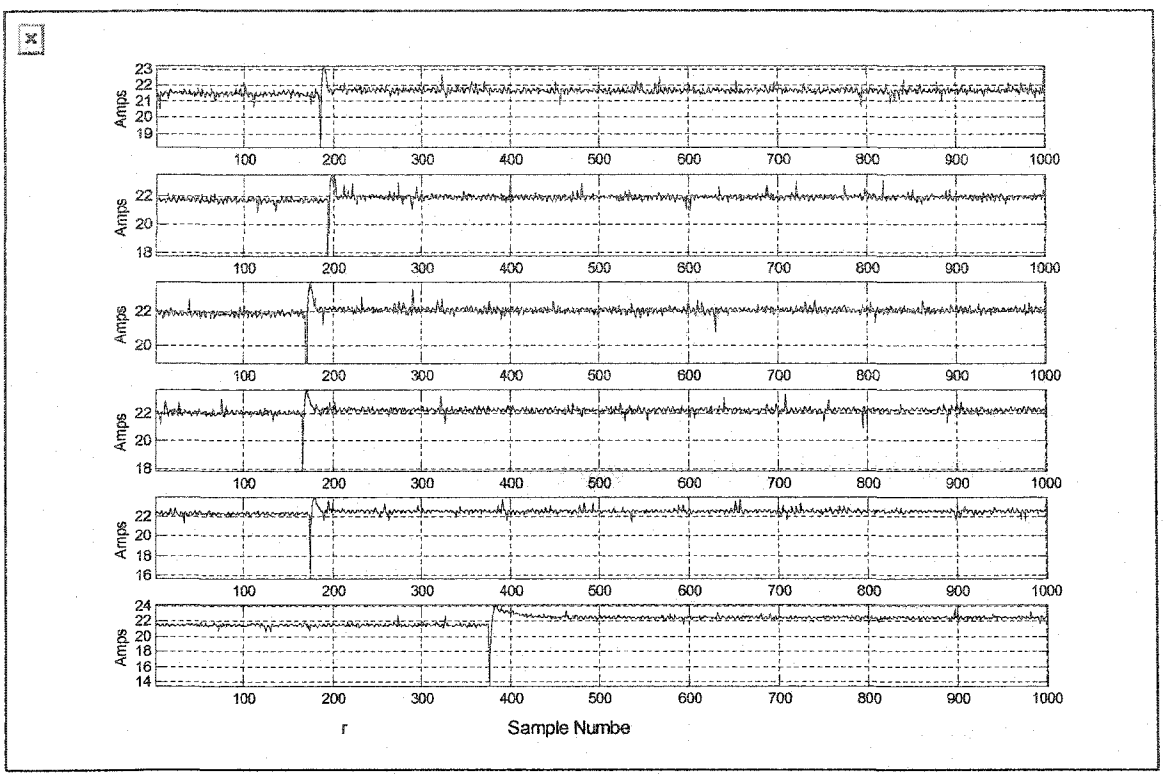


Fig. 43. Plots of plant input for negative command inputs.

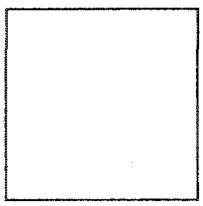


Fig. 44. Plots of plant outputs for positive command inputs.

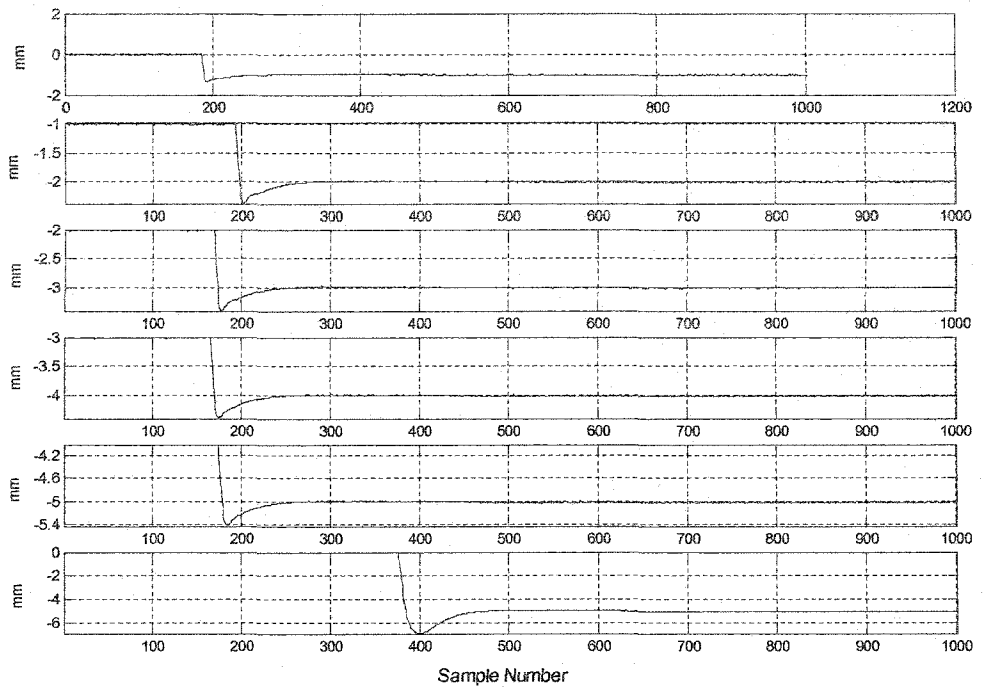


Fig. 45. Plots of plant output for negative command inputs.

A visual analysis of the plots shows the shapes of controller and physical plant responses closely resembled the theoretical responses obtained during the controller design. Although the physical system responses were much noisier, the controller output contains the same short, large-valued excursion followed by the smaller, longer deviation over the same 0.1 second period. Another similarity is the shape and magnitude of the plant output responses. The rise times to peak value and the settling times are similar for theoretical and actual responses and there are no oscillations or other anomalies appearing in the plant responses.

This second results from this set of experiments addressed the experiment's goals very directly. The close match in theoretical and measured steady state values is a strong indicator the MSBS plant is well-modeled by the non-linear equation. The accuracy of the force equations and linearization method are also indicated because the controller for the real plant can be tuned with predictable results in the setting of actual MSBS equipment. Also, it did not appear the Copley amplifiers introduced any significant phase shift, tending to rule out the need to model the amplifier dynamics.

On the other hand, the asymmetry of the changes in input current (Table XIII) tends to show the plant linearization was not thoroughly realistic. This was also borne out by the need to lower the

equilibrium current from its calculated value, even with a much more carefully measured test object. This finding was consistent with the first set of experiments.

Fortunately, the equilibrium current problem does not appear to have operational consequences other than to require the equilibrium current to be adjusted down. It is also possible the physical measurements of the sensor positions are in error, since the tools used to position the sensors were not very sophisticated. The 1-DOF maglev behavior over the desired command input range is acceptable.

Conclusion

By the conclusion of the previous chapter, the design goals, theoretical background and system configuration had been introduced. The goal of this chapter was to further elucidate how a full-scale prototype of the MSBS can be built by addressing two main areas: 1) the application of feedback control to a 1-DOF MSBS plant; and 2) additional implementation issues over and above those discussed in Chapter III. By successfully implementing a 1-DOF maglev system with the major MSBS components and a lead-lag controller with negative feedback, the premises for the MSBS modeling and design methods were decisively validated. These design assumptions include the so-called "large-gap" approximation, the linearity of the sensor responses, and the adequacy of using a first order Taylor series approximation in creating the linear model. In addition, this chapter dealt with practical engineering considerations pertinent to an implementation of the MSBS of any degree-of-freedom. These include the conversion algorithm for amplifier input and the "Turn-on Threshold" circuitry needed to limit compensator action when the plant is unresponsive. At the conclusion of this chapter, sufficient background has been laid to introduce one more design concept, before the design of a multiple-degree-of-freedom system is taken up.

CHAPTER V

APPLICATION OF OPTIMAL CONTROL THEORY TO THE MSBS

Introduction

This chapter covers the last of the 1-DOF experiments conducted in preparation for controlling multiple degrees of freedom. This particular set of experiments warrants its own chapter because the motivation for these experiments breaks from the earlier emphasis on model validation and rudimentary implementation, and focuses on gaining experience in optimal controller design in the context of the MSBS plant. This was a necessary step because the "classical" design techniques employed thus far would likely not suffice for the more complex MSBS, especially given its strong requirement for reliability and robustness.

The controllers up to this point had been "classical' in that their design was based on output feedback and employed the traditional root locus, bode and Nyquist analysis using the poles and zeros of transfer functions. The resulting compensator altered the plant's behavior by adding three pole-zero pairs to the system. For the 1-DOF maglev, a single-input, single-output (SISO) system, this did not significantly increase the system's order and complexity, and a controller was fairly easily obtained after a few iterations. However, the multiple-input, multiple-output (MIMO) system represented by the 5-DOF MSBS is considerably more complex. This can be appreciated by considering the twenty-five element transfer function matrix for the 5-DOF MSBS plant, developed by Jafri [4]. For example, the first column of that matrix is shown in Table XIV.

TABLE XIV
FIRST COLUMN OF 5-DOF TRANSFER FUNCTION MATRIX

Input-to-Output Mapping	Transfer Function
One-to-One	$\frac{0.023873}{s^2}$
One-to-Two	$\frac{0.023873(s+0.01293)(s-0.01293)}{s^4}$
One-to-Three	$\frac{0.79833}{(s+6.982)(s-6.982)(s^2+48.75)}$
One-to-Four	$\frac{0.081379s^2}{(s+6.982)(s-6.982)(s^2+48.75)}$
One-to-Five	$\frac{0.081379}{s^2}$

The transfer function of a compensator for the 5-DOF MSBS could quite easily add many poles and zeros to the system, increasing the likelihood of undesirable modes. Therefore, it was decided to control the MSBS plant using a non-zero set point regulator with linear state variable feedback, designed using optimization techniques [4]. In fact, this approach was shown to be feasible in a paper describing its application to a 5-DOF, large-gap maglev system developed at NASA [25]. The benefits of using state variable feedback include the fact that it does not add large numbers of poles and zeros to the system and that it lends itself well to computer assisted design, since it is done using state space form of the plant (see [26], p. 364). However, the rationale for using optimal control methods in the MSBS is best presented as follows:

1) The MIMO case provides a wide latitude for acceptable compensator design, driving the need for a methodical approach to determine the best compensator.

2) The "optimal" approach, using a cost function based on the state space formulation of the plant provides such a methodology.

3) The regulator control law lends itself to optimal control methods, with many implementation tools in Matlab and many proven examples in the literature. (For example, see [27], ch. 5, and [28], section 9.2.)

The practical result of using optimization techniques to design a regulator is that it presents the designer a convenient—and limited—set of "knobs" to tune, which will in the very least give stability to a physical plant whose critical areas of uncertainty are not well understood. Of course, candidate controllers always have to be evaluated, and in keeping with the desire to maintain a methodical practice, this created a need for an efficient tool to evaluate controllers.

The goals of this experiment then were as follows: 1) to apply optimal control theory to a 1-DOF maglev system; 2) to construct a working 1-DOF regulator based on optimal control principles; and 3) in the course of designing the controller, create controller validation tool useful for future MSBS development.

This chapter first presents a discussion of optimal control theory and its role in MSBS development pertains to linear state variable feedback and state estimation. Following the theory, the experimental set-up and procedure is explained, followed by a discussion of the results. The chapter concludes with an analysis of the progress made toward the full 5-DOF MSBS.

Optimal Control Theory Adapted to the MSBS

This section discusses two closely related but distinct concepts relative to the MSBS controller design. The first is the use of the regulator control law with linear state variable feedback; the second is the application of "optimal" control theory to the regulator control law, an application that is well described and widely employed in control systems work. Together these two ideas lead to a fairly straightforward methodology for developing an MSBS controller. In this section, an overview of regulator control theory is presented first, along with an explanation of state estimation. Then the use of optimal control in the design of regulators and state estimators is explained. Lastly in this section, the idea of optimal control is used in the formation of a methodology for MSBS controller design. This methodology will apply to the higher degree of freedom implementations.

The Regulator Control Law and State Estimation

As stated in the chapter introduction, the motivation for choosing a regulator design with linear state variable feedback over the "classical" lead-lag compensator includes this consideration: the reduction of complexity and associated risk of unforeseen adverse effects. The basis for what is called the regulator control law lies in the concept of linear system controllability. If a linear system is state controllable, then there always exists some input that will drive the system to an arbitrary state. According to [29], p. 187, in the early 1960's a nexus was described between state-controllability and the ability to shift a plant's eigenvalues using state variable feedback. Specifically, it was found if state controllability exists, state variable feedback can be used to generate any characteristic polynomial. That led to the idea of relocating a plant's poles so as to obtain the desired response characteristics. This concept of pole-shifting is clearly illustrated through the state space formulation of the plant, i.e.,

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t)$$

$$\mathbf{y}(t) = \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{u}(t),$$

to which the so-called regulator control law is applied:

$$\mathbf{u}(t) = -\mathbf{K}\mathbf{x}(t).$$

Note that K is an $m \times n$ matrix that produces a linear combination of the states at the plant input. Assuming all the states are controllable, substituting the control law in the plant equation gives

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) - \mathbf{B}\mathbf{K}\mathbf{x}(t) = (\mathbf{A} - \mathbf{B}\mathbf{K})\mathbf{x}(t),$$

which gives the plant a new characteristic equation derived from

$$a(s) = \det(s\mathbf{I} - \mathbf{A} + \mathbf{BK}).$$

Now, if a set of poles can be identified that would give the plant the desired response—the crux of the design challenge and the motivator for a workable design methodology—these poles can be used to calculate a desired characteristic equation. The task at that point is to match the characteristic equation derived from the control law with the one derived from the desired poles—by finding the right linear state variable feedback (LSVF) coefficient matrix, **K**.

It was stated the basis for pole placement is state controllability. However, even if the plant is not entirely controllable, linear state variable feedback can still be used to one's advantage if the uncontrollable modes are asymptotically stable, i.e., the uncontrollable eigenvalues are in the open left hand side of the s-plane. In this condition the plant is referred to as stabilizable. The regulator control law is applied only to the controllable states and the stable, uncontrolled states are left to dampen out naturally. There are two Matlab commands, *acker.m* and *place.m*, that take non-optimizing approaches to deriving the value of **K**, although these are not used in the MSBS controller design.

So far the discussion of the regulator control law has assumed all the plant states are accessible. Unfortunately, this is not the case for the MSBS or for most real-life plants. If for no other reason, the cost of measuring very many states in a physical system is prohibitive. Consequently, seeking to apply linear state variable feedback in this circumstance gives rise to the problem of state estimation. For this reason, the MSBS controller uses a dynamic asymptotic state estimator, sometimes called an observer.

The concepts of pole-placement and asymptotic state estimation share a type of duality derived from linear systems theory. With pole placement, if a system (**A**, **B**) is controllable, then its states can be used to form a feedback signal that will drive the plant (system) to some desired end state, effectively changing the modes. In a type of dual condition, a plant must be state-observable to solve the state estimation problem. The property of observability implies that a knowledge of the plant's states can be obtained from the plant's input and output vectors. This means that at any time $t \geq t_0$, the state vector $x(t)$ can be computed from a knowledge of the plant's input and output history from time t_0 forward (see [29], p. 294). The usefulness of this property in solving the problem of state estimation is this: because state-observability ensures information about the states is present in a system's output, un-measurable states can be duplicated using signals that can be measured, i.e., the system's input and output vectors.

Fig. 46. illustrates a regulator with state estimation for a continuous time system. As pointed out above, the LSVF matrix **K** operates on the plant states to realize the regulator control law. However, in the system illustrated in Fig. 46, the states themselves are not available and an asymptotic state estimator, or observer provides state estimates. The estimator takes the same input as the plant and applies it to a state space model of the plant, in theory replicating the plant output. But since a model can never perfectly capture all the dynamics of a real-world plant, there is always some error between the real plant and replicated output. The estimator feeds this output error, multiplied by the constant matrix **L**, back to the plant input to force the estimator output closer to the plant output.

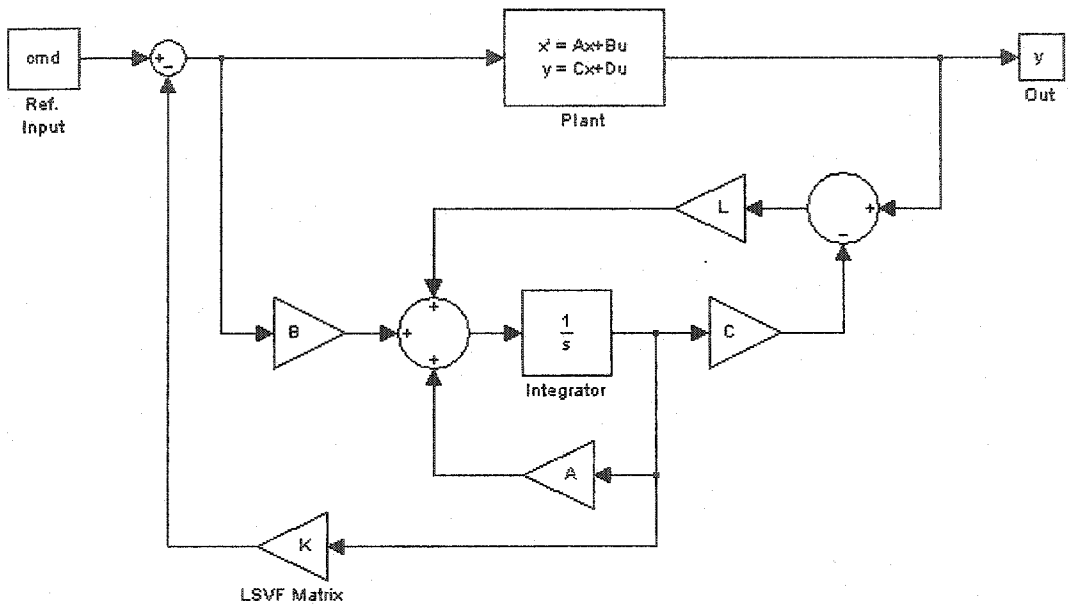


Fig. 46 Regulator with state estimation

The desired state estimation takes place as a result of the system's observability. As noted above, this property implies the plant's states are represented in, and derivable from, the plant's input and output. Thus, as the estimator's output is driven asymptotically toward that of the real plant, the estimator's states are also asymptotically driven toward the plant's, if indeed the plant is observable—and the state-space model of the plant is accurate. The rate at which the estimated state vector approaches the real state vector is determined by the values in **L**.

The estimator dynamics are given as follows

$$\dot{\hat{\mathbf{x}}} = \mathbf{A}\hat{\mathbf{x}} + \mathbf{B}\mathbf{u} + \mathbf{L}(\mathbf{y} - \mathbf{C}\hat{\mathbf{x}}) = \mathbf{A}\hat{\mathbf{x}} + \mathbf{B}\mathbf{u} + \mathbf{L}\mathbf{C}\mathbf{x} - \mathbf{L}\mathbf{C}\hat{\mathbf{x}},$$

$$\hat{\mathbf{y}} = \mathbf{C}\hat{\mathbf{x}} + \mathbf{D}\mathbf{u},$$

where $\{\hat{\mathbf{x}},\hat{\mathbf{y}}\}$, and $\{\mathbf{x},\mathbf{y}\}$ denote estimated and measured vectors, respectively.

Just as not all systems are fully state-controllable, not all are state-observable. However, in a dual property to the one that allows linear state variable feedback with uncontrollable, but stabilizable systems, state estimation can still be used with unobservable systems if the unobservable states are stable. Under this condition, the system is called detectable.

The central task in estimator design is the selection of the feedback matrix **L**, which determines where the estimator poles lie and thus how quickly the state estimates approach the actual states. The rule of thumb is that **L** should be chosen to make the estimator's response two to six times the bandwidth of the plant, although there are cases when this is not advisable (see [26], pp. 47, 304). While the selection of **L** does not influence the system response as directly as the value of **K**, this selection still represents another degree of freedom contributing to the complexity of the overall design process. As with pole placement for the regulator, the Matlab commands *Acker* and *Place* can be used to reposition the estimator poles to prescribed locations, although these commands are not used in the MSBS controller design.

Before proceeding, an explanation is offered for the terms "dual" and "duality" used in conjunction with controllability and observability. What is meant by dual in this context is that given a system realization $\{\mathbf{A},\mathbf{B},\mathbf{C}\}$, its dual is defined as $\{\mathbf{A}^{\mathrm{T}},\mathbf{C}^{\mathrm{T}},\mathbf{B}^{\mathrm{T}}\}$. The duality between controllability and observability implies the existence of one property in a system implies the existence of the other property in the dual system (see [29], p. 90). This is shown by considering two alternate definitions of controllability and observability, equivalent to earlier statements about the relationships between the system states and the inputs and outputs. One definition of state-controllability is that the so-called controllability matrix,

$$C = \begin{bmatrix} \mathbf{B} & \mathbf{AB} & \mathbf{A}^2\mathbf{B} & \dots & \mathbf{A}^{n-1}\mathbf{B} \end{bmatrix},$$

must be of rank equal to the number of states (see [30], pp. 29-32). Likewise, a definition of observability is that the observability matrix,

$$O = \begin{bmatrix} \mathbf{C} & \mathbf{CA} & \mathbf{CA}^2 & \dots & \mathbf{CA}^{n-1} \end{bmatrix}^{\mathrm{T}},$$

must be of rank equal to the number of states. Taking the transpose of the observability matrix above results in

$$O^{\mathrm{T}} = \begin{bmatrix} \mathbf{C}^{\mathrm{T}} & \mathbf{A}^{\mathrm{T}}\mathbf{C}^{\mathrm{T}} & (\mathbf{A}^{\mathrm{T}})^{2}\mathbf{C}^{\mathrm{T}} & \cdots & (\mathbf{A}^{\mathrm{T}})^{n-1}\mathbf{C}^{\mathrm{T}} \end{bmatrix}.$$

But by definition this is the controllability matrix of the dual of the original system.

Thus far the discussion has shown designing a controller with linear state variable feedback (LSVF) and state estimation requires the plant to be both controllable and observable, or at least stabilizable and detectable [30]. This observation leads to another important property of linear systems: if a system realization $\{\mathbf{A},\mathbf{B},\mathbf{C}\}$ is both controllable and observable, then it must also be minimal. This means there does not exist another triple $\{\mathbf{A}',\mathbf{B}',\mathbf{C}'\}$ with fewer states that will produce the same transfer function $G(s) = \mathbf{C}(s\mathbf{I}-\mathbf{A})^{-1}\mathbf{B}$. The implication for the controls designer is some degree of assurance that the plant is amenable to control using LSVF and state estimation if the linearized state-space model of the plant is jointly controllable and observable, a condition that is easy to verify with several commands in Matlab's Control System Toolbox. (Of course, this assurance is predicated on the fidelity of the model.) If a system is not minimal, but still stabilizable and detectable, then at least hidden modes should not pose a problem with system stability.

Optimal Control Design for the Regulator and State Estimator

It was stated the real challenge to the designer is finding locations for the plant poles that produce the desired closed-loop response with acceptable control inputs. Basic decisions such as which poles should be moved and by how much bring with them into the design process issues like how much control effort is needed (or desired), the uncertainty of the effects of new pole-zero combinations on the response, and whether some poles can even be controlled or observed at all. Increasing the complexity of the design decision further, in MIMO systems there can be multiple values of $\mathbf{K}$ and $\mathbf{L}$ in the regulator and state estimator that produce an acceptable response as shown in [26], p. 359. Also in MIMO systems, the poles and zeros have directions associated with them, meaning a given pole placement scheme could produce very different responses for different input vectors. One way to handle the complexity of the design challenge is to apply a systematic approach to design with as a few "tuning knobs" as possible. This is the motivation behind the "optimal" design employed in this project.

Generally speaking, optimal control refers to the practice of minimizing an objective function (a cost function) containing a dependence on one or more controller design parameters. The result

of the minimization is a compromise among design objectives, such as the allowable control effort, speed of response or measurement error. However, at least this approach guarantees stability of the system [26], [31]. The cost function can be biased, or weighted, toward one design consideration or another, and it is these weights that provide the tuning "knobs" for optimal control design.

There are several other optimal design approaches considered part of "modern" control theory. Two of these are H_2 and H_∞ optimal design, which seek to optimize the system with respect to certain error signals by minimizing of the H_2 and H_∞ norms of a specialized system formulation. The specialized formulation in this casen is the lower linear fractional transformation of the plant and controller when configured in what is called the general control configuration in [28], section 9.3. These design approaches use state space realizations.

The approach to optimal control used in the MSBS is referred to as the Linear Quadratic Gaussian (LQG) problem, so called because of its application to linear systems, use of a cost function in quadratic form and assumption of Gaussian stochastic disturbances within the system. LQG design applies optimization to both the regulator control law and state estimation processes and technically consists of two sub-problems: the Linear Quadratic Regulator (LQR) solution and the Kalman estimator design. Fortunately, because of what is called the Separation Principle, explained in succeeding paragraphs, the LQR and estimator problems can be addressed independently. An overview of each of these two design tasks will be presented as well as a brief explanation of the separation principle.

Briefly, the Linear Quadratic Regulator (LQR) solution involves minimizing a certain quadratic cost function containing the linear state variable feedback (LSVF) matrix, $\mathbf{K}$. The $\mathbf{K}$ thus derived produces an "optimal" regulator control law as defined by certain weighting matrices, $\mathbf{Q}$ and $\mathbf{R}$, which are also part of the cost function. Of course, the control law applies to a linear system, and the assumption is made all the system states are available. Following is a short explanation of the cost function and relationships of the matrices $\mathbf{K}$, $\mathbf{Q}$ and $\mathbf{R}$ to the design process.

In LQR design, "optimal" control refers to a balance between the quickness with which the regulated system will reach (or return to) its equilibrium point and the amount of control energy that has to be added to the system to achieve that rate of response. Optimality is, then, a condition determined by the designer according to many practical engineering factors such as system use, plant input and output capacities, actuator size and many others. As stated above, the

designer introduces his or her definition of optimality into the LQR solution via the weighting matrices **Q** and **R** in the cost function.

The cost function for continuous time (CT) systems is given by

$$J = \frac{1}{2}\int_0^\infty (\mathbf{x}(t)^T \mathbf{Q}\mathbf{x}(t) + \mathbf{u}(t)^T \mathbf{R}\mathbf{u}(t))dt\,, \tag{101}$$

where **Q** and **R** are the weighting matrices for the plant states and inputs, respectively. The corresponding cost function for a discrete time (DT) system is

$$J = \frac{1}{2}\sum_{k=0}^{N} \mathbf{x}(k)^T \mathbf{Q}\mathbf{x}(k) + \mathbf{u}(k)^T \mathbf{R}\mathbf{u}(k)\,. \tag{102}$$

In both CT and DT cost functions, **Q** and **R** represent the design choices in the LQR solution. The matrices Q and R are of dimensions (n x n) and (m x m), respectively, where n is the number of states and m the number of inputs to the plant. In addition, both **Q** and **R** must be non-negative definite, meaning they ensure their respective products are always non-zero for any value of **x** or **u**. (This can be ensured by requiring both matrices to be diagonal.) The rationale behind assigning values to the matrices is as follows. The numeric values in the **Q** diagonal represent the relative importance of forcing the corresponding state to its equilibrium point as quickly as possible. The numeric values of the **R** diagonal represent the relative cost of using the corresponding input. In other words, a low value in **R** allows a greater magnitude (voltage, current, force, etc.) on that control input in order to achieve the level of response signaled by the elements of **Q**.

The process of calculating the optimal LSVF matrix **K** for the LQR solution is a complex sequence of steps. (More detailed development of the calculations associated with the LQR solution can be found in [29], pp. 213-233, for CT systems, and [26], pp. 371-382, for DT systems.) First the objective, or cost functions (101) or (102) are augmented to reflect the constraints imposed by the plant dynamics, i.e., $\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t)$,

$\mathbf{x}(k+1) = \Phi\mathbf{x}(k) + \Gamma\mathbf{u}(k)$.

Then a mathematical method known as Lagrange Multipliers is applied to the composite cost function and the cost function is differentiated with respect to its constituent variables, the control input vector, the state vector and the Lagrange multiplier vector. The resulting set of three differential equations (or difference equations if using DT) represent a time-varying system, which, because of assumptions about the initial and final states of the system, poses a two-point boundary value problem. The solution to this problem leads to the creation of the Ricatti

equation, which is not easy to solve. The result is a time-varying value of **K**, which is optimal, but difficult to implement.

Fortunately, the value for **K** stays constant over most of the period during which the regulator is in operation, with the beneficial implication that, for practical purposes, the constant value for **K** is all that need be calculated. This leads to a simpler formulation of the above mentioned boundary value problem through the use of the Algebraic Ricatti equation. The steady state optimal solution, which is what is really meant when using the term LQR, produces a **K**-matrix related to the design parameters, **Q** and **R** through the following:

$$\mathbf{K} = \mathbf{R}^{-1}\mathbf{B}^{\mathrm{T}}\mathbf{P}, \tag{103}$$

where **P** is the non-negative definite solution to the Algebraic Ricatti Equation (ARE)

$$\mathbf{A}^{\mathrm{T}}\mathbf{P} + \mathbf{P}\mathbf{A} - \mathbf{P}\mathbf{B}\mathbf{R}^{-1}\mathbf{B}^{\mathrm{T}}\mathbf{P} + \mathbf{Q} = 0. \tag{104}$$

When controllability and observability are assumed, the solution **P** to (104) above leads to a unique, optimal stabilizing value of **K**. [29]. There is a similar relationship between the **K**-matrix for discrete systems and the discrete ARE. However, it should be noted that if an optimization is realized for the continuous cost function (101), and then a discrete time controller is sought for the continuous plant, the weighting matrices **Q** and **R** cannot be directly applied to the discrete cost function (102).

Part two of the LQG solution is the design of an optimal state estimator, in particular the Kalman estimator, or filter. The Kalman filter, named after one of the pioneers in modern control theory, is an optimal state estimator that accounts for Gaussian noise associated with the plant inputs and output sensors. In the Kalman filter, the optimization is done with respect to the mean square error of the estimated states, given certain noise conditions specified by the designer. The result is the feedback gain matrix **L** most statistically likely to minimize the state estimation error (see [26]. p. 394.) The structure of the Kalman estimator is the same as the estimator depicted in Fig. 46.

In a manner analogous to the use of **Q** and **R** in the LQR solution, certain matrices are used to change the parameters in the Kalman filter design and ultimately produce the optimal gain matrix **L**. The following equations illustrate the context from which these "tuning" matrices arise and their relevance to real life systems.

The Kalman filter is applied to the continuous time linear plant

$$\dot{x}(t) = \mathbf{A}x(t) + \mathbf{B}u(t) + \mathbf{G}w(t) \tag{105}$$

$$y(t) = \mathbf{C}x(t) + \mathbf{D}u(t) + v(t), \tag{106}$$

where w(t) and v(t) are process and measurement noise signals, respectively, and G is a process noise gain matrix (see [27], p. 8-100). The noise signals are assumed to be zero-mean, uncorrelated stochastic processes, whose covariances are given by

$$E\left\{w(t)w(t)^{\mathrm{T}}\right\} = \mathbf{Q}_n, \text{ and} \tag{107}$$

$$E\left\{v(t)v(t)^{\mathrm{T}}\right\} = \mathbf{R}_n. \tag{108}$$

Because the noise sources are assumed to be uncorrelated, or "white" noise, the $\mathbf{Q}_n$ and $\mathbf{R}_n$ are diagonal matrices representing the mean square energy level of the noise. And if the occurrence of noise is equally likely on all states or inputs, all the elements on the matrix diagonals are equal. In the Kalman filter calculations, as with the LQR solution, an optimal feedback gain L is calculated based on the unique solution to an Algebraic Ricatti Equation containing $\mathbf{Q}_n$, $\mathbf{R}_n$ and G. (A full development of the Kalman filter calculations for a DT system can be found in section 9.4 of [26].)

In general, measurement noise is understood well enough so that somewhat meaningful values can be used in $\mathbf{R}_n$ [26]. This is due to more accurate knowledge of the sensors, which are often separate, commercial off-the-shelf items with fairly reliable specifications. However, because the nature of process noise is not thoroughly known, and because true "white" noise does not occur in real systems, in practical controller synthesis $\mathbf{Q}_n$ and $\mathbf{G}_w$ represent tuning "knobs" used in an iterative, multi-step design process more than precise plant modeling parameters, as noted in several texts on control systems, including [26], [28], and [29].

Mention was made earlier of what is called the Separation Principle and its practical effect of allowing the estimator and regulator to be designed separately (see [28], section 8.3). This principle is shown briefly here using the state space equations for state feedback and state estimator configuration depicted in Fig. 45.

The equation for the state estimation error can be found by subtracting the estimator state equation from the plant equation $\dot{x} = Ax + Bu$, giving

$$(\dot{\mathbf{x}} - \dot{\hat{\mathbf{x}}}) = \mathbf{A}(\mathbf{x} - \hat{\mathbf{x}}) - \mathbf{LC}\mathbf{x} + \mathbf{LC}\hat{\mathbf{x}} = (\mathbf{A} - \mathbf{LC})\bar{\mathbf{x}},$$

where $\bar{x}$ denotes the state estimation error. Then a system equation in terms of the states and state estimation error can be written:

$$\begin{bmatrix} \dot{\bar{\mathbf{x}}}(t) \\ \dot{\mathbf{x}}(t) \end{bmatrix} = \begin{bmatrix} \mathbf{A}-\mathbf{LC} & 0 \\ \mathbf{BK} & \mathbf{A}-\mathbf{BK} \end{bmatrix} \begin{bmatrix} \bar{\mathbf{x}}(t) \\ \mathbf{x}(t) \end{bmatrix}. \tag{109}$$

The significance of (109) is apparent when the characteristic equation is formed as follows:

$$\begin{vmatrix} s\mathbf{I}-\mathbf{A}+\mathbf{LC} & 0 \\ -\mathbf{BK} & s\mathbf{I}-\mathbf{A}+\mathbf{BK} \end{vmatrix} = |s\mathbf{I}-\mathbf{A}+\mathbf{LC}||s\mathbf{I}-\mathbf{A}+\mathbf{BK}| = 0 \tag{110}$$

it is readily seen the poles of the estimator and those of the controlled plant do not influence one another. It should be noted, though, that while the pole placements in both the estimator and regulator are done independently, there is coupling between the two systems through the term $-\mathbf{BK}$.

For a stabilizable, detectable system, the above calculations are guaranteed to produce optimal gain matrices, **K** and **L**, resulting in an asymptotically stable LQG-regulated system [28]. Unfortunately, combining the Kalman estimator with an optimized regulator often results in much worse performance than that anticipated from the LQR solution by itself. This degradation is tied to the apparently arbitrary changes in the system gain and phase margins that occur when plant and estimator dynamics are placed in series to form the loop gain transfer functions [28].

Recall that gain and phase margin are indicators of how much the dynamics of the actual system may deviate from the modeled system before stability is lost. (System bandwidth and rejection of output disturbances and sensor noise are also tied to phase and gain margin.) These stability margins are determined from the open loop transfer function, using Bode or Nyquist plots, as a means of gauging the closed loop system performance.

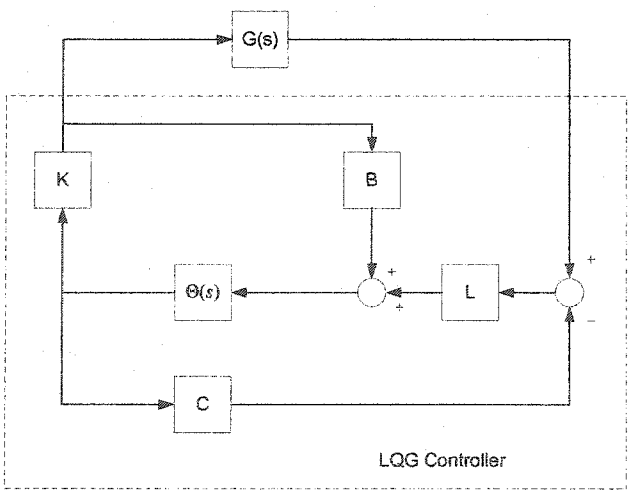


Fig. 47. Structure of combined LQR/Kalman Estimator controller.

Consider the LQG controller configuration in Fig. 47. The open loop transfer function taken at the output of the plant is given by

$$L_{OL} = -G(s)\mathbf{K}\left[\Phi(s)^{-1} + \mathbf{BK} + \mathbf{LC}\right]^{-1}\mathbf{L},$$

where

$$\Phi(s) = \left(s\mathbf{I} - \mathbf{A}\right)^{-1}.$$

It is through their complicating effect on the open loop transfer function that the combination of Kalman estimator and LQR can reduce stability margins and thus reduce system robustness. Therefore, even though the Separation principle allows independent LQR and estimator design, as a result of this phenomenon, the designer must take pains to validate the resulting LQG controller.

Method for 1-DOF MSBS Controller Design

Earlier in this chapter the point was made that a primary motivation for considering optimal control was to find a systematic approach to design using a manageable number of design parameters. The LQG solution described thus far realizes that goal in part by identifying three or four design parameters at least guaranteeing system stability if not desired robustness.

However, limiting the number of design parameters does not itself produce a workable design process. This is because the net effect of multiple LQG parameter changes is not intuitively apparent—even though the parameters' specific individual roles in the equations are known. For example, it is understood that in the LQR command the parameter **R** influences the size of the optimum control input and **Q** has designates the relative importance of individual states. But the best choices for the noise covariance and gain matrices in the Kalman estimator are not obvious a priori. Worse, the effects on performance of combining the LQR and Kalman estimator are virtually impossible to judge apart from testing the resulting system [28]. For this reason, the remainder of this section is given to presenting a methodology and workable procedure for applying the optimal control parameters.

This section first provides a small catalogue of parameters used in optimal LQG regulator design, which represent the "tuning knobs" available to the designer. Secondly, a general methodology for control system design is presented along with a procedure for implementing that methodology. Following that there is a discussion of the concept of controller robustness as pertains the design process. Lastly, the section gives an overview of the Loop Transfer Recovery (LTR) method used for optimal design.

Optimal Design Parameters

Before proceeding, it should be pointed out that even though this chapter lays the groundwork for optimal control of multiple degree of freedom (MIMO) systems, the chapter's focus is still on the 1-DOF (SISO) maglev. Because the SISO model was relatively simple, its controller design was something of a "toy" problem and the utility of optimal design was not fully realized in this experiment. The parameters presented in Table XV so far did not have to be selected with as much engineering judgment as they would for MIMO systems. However, the 1-DOF experiments did provide a valuable exercise for learning the Matlab commands and setting up procedures for evaluation controller performance. (The 2-DOF experiments, which make use of a full 5-DOF nonlinear model, will better illustrate the benefit of a systematic design process.) Table XV summarizes the "tuning knobs" available in the optimal control design method and their effects. The table also lists the Matlab commands that employ those parameters.

TABLE XV
SUMMARY OF DESIGN PARAMETERS USED IN OPTIMAL CONTROLLER DESIGN

Name	Symbol [a]	Description	Effect On Design	Associated Matlab Command
State Weighting Matrix	$\mathbf{Q}$	Positive semi-definite matrix (n x n) that assigns relative values to the states in the LQR cost function.	Higher values tend to cause regulator to drive states to equilibrium faster.	lqr.m, dlqr.m, lqrd.m
Input Weighting Matrix	$\mathbf{R}$	Positive semi-definite matrix (m x m) that assigns costs to control inputs in the LQR cost function.	Higher values tend to restrict control "effort" expended to drive states to equilibrium.	lqr.m, dlqr.m, lqrd.m
Process noise covariance matrix.	$\mathbf{Q_n}$	Constant (n x n) matrix representing the level of "white" noise present in the plant states. Serves as weighting matrix in ARE.		lqe.m, dlqe.m, kalman.m, kalmd.m
Sensor noise covariance matrix.	$\mathbf{R_n}$	Constant (m x m) matrix representing the level of "white" noise present in the measurements of the plant output. Serves as weighting matrix in ARE.		lqe.m, dlqe.m, kalman.m, kalmd.m
Process noise gain matrix.	$\mathbf{G_w}$	Gain matrix (n x m) describing dynamics associated with process noise vector in plant.		lqe.m, dlqe.m, kalman.m, kalmd.m

[a] As per the convention adopted for this thesis.

Design Method and Procedure

To make efficient use of the design parameters, an orderly sequence of steps is needed to ensure that all the parameters are adjusted through appropriate ranges and all reasonable controllers are evaluated. One such design process was developed in [4] for the 5-DOF MSBS simulations. In that process, an LQR solution was first derived for a 5-DOF MSBS model for a series of five successively more complex simulations, the first involving only the basic linear state space formulation and the last modeling all the known dynamics and disturbances. At each level of complexity the time and frequency domain responses of the system were evaluated and the matrices **Q** and **R** adjusted as necessary. Following the LQR iterations, a Kalman estimator was designed using the same five-stage approach. Finally, the total LQG solution was evaluated. The process in [4] is generalized in the following synopsis:

> Create ranges of design parameters based as much as possible on knowledge of system dynamics, system use, etc... Derive an LQR solution (an LSVF matrix **K**) and a Kalman estimator, and combine the two solutions into an LQG regulator. Simulate controller operation using a system model for and validate controller against time and frequency response criteria. Repeat for every parameter permutation, identifying each solution with its respective parameters.

The following procedure was used to execute the design method above for in the 1-DOF experiments covered in this chapter. The steps were implemented primarily using Matlab scripts and Simulink models.

TABLE XVI
CONTROLLER DESIGN PROCEDURE

Step No.	Action
1	Implement a nonlinear model of the 1-DOF system in Simulink and derive a state space formulation of the linearized plant. Analyze the plant for controllability and observability.
2	Produce ranges of values for design parameters **Q**, **R**, $\mathbf{Q}_n$, $\mathbf{R}_n$ and $\mathbf{G}_w$, based on knowledge of the MSBS. (See TABLE XV.)
3	Based on the design parameters **Q** and **R**, create a range of LSVF gain matrices **K** using Matlab's *lqr.m* command. Ensure each **K** is traceable to parameters that produced it.
4	Test the range of **K** matrices in a closed-loop LQR system containing the linearized plant model.
5	Based on one set of the parameters $\mathbf{Q}_n$, $\mathbf{R}_n$ and $\mathbf{G}_w$, design a single Kalman estimator using Matlab's *kalman.m* command. Ensure resulting estimator is traceable to its own parameters.
6	Combine one of the values for **K** with the Kalman estimator using *lqgreg.m*.

TABLE XVI, CONT'D

Step No.	Action
7	Test the resulting LQG regulator using a Simulink model of the nonlinear, closed-loop system. Record and evaluate the time responses of the model with respect to relevant criteria, such as percent overshoot, settling time, steady-state value and control input value. Plot the time responses as needed. If responses acceptable, go to next step. Otherwise, return to step 6 and select another K-matrix.
8	Using same LQG regulator produced in step 5, repeat simulation while dynamically applying output "disturbances" to the system model. Record and evaluate the time responses as before and compare to earlier results as a means to test the controller's stiffness. If responses acceptable, go to next step. Otherwise, return to step 6 and select another K-matrix.
9	Using same LQG regulator, repeat simulation while simulating perturbations in the plant's magnetic flux density. (Plant "perturbed" after controller is designed.) Record and evaluate the time responses and compare to earlier results as a means to test the controller's robustness.
10	Compare the most recent validation results with those from previous acceptable controllers and select best candidate. If necessary, repeat steps 5 through 9 for a different Kalman estimator.
11	Validate candidate controller in actual 1-DOF system.

Measures of Controller Quality

Steps 7 and 8 in Table XVI describe actions intended to measure the "quality" of the prospective controller in two distinct ways. First, step 7 seeks to more closely simulate the real-life operating conditions of the HRTF by "disturbing" the forces acting on the suspended test object. The system's behavior under this condition provides a measure of a quality referred to as controller "stiffness". Second, step 8 is an attempt to ascertain the controller's ability to function despite the inevitable discrepancies between the real plant and its linear model (upon which the controller is based). This aspect of controller quality is known as robustness with respect to model uncertainty. These two measures are explained in the following paragraphs.

The controller stiffness test in step 7 involves a measure of how rigidly the controller holds the plant (i.e., the magnetic core) in its equilibrium condition. For a wind tunnel application such as the MSBS, greater controller stiffness means greater resistance to fluctuations in the aerodynamic forces. To quantify this performance measure, and thus enable different controllers to be accurately compared, a stiffness ratio was devised in earlier MSBS work [4]. This ratio is the magnitude of a disturbance-producing force divided by the amount of displacement produced by that force. Displacement is measured with respect to the equilibrium position. In the 1-DOF

maglev system, stiffness is measured in terms of a known force applied in a vertical direction divided by the position change that force produces. (In the 2-DOF system, stiffness is also measured in terms of torque and its resulting angular displacement.) Step 7 then is essentially a refinement of the simulation model done to reduce the number of candidate controllers that will have to be tested under more time consuming conditions with the real MSBS plant.

As explained in Chapter II, the MSBS controller is designed around a linear model of the physical MSBS, and the linear model is in turn based on certain assumptions and simplifications. The discrepancies between the model and the real plant, always present to some extent, are collectively referred to as "model uncertainty", and the controller's ability to perform properly in the face of these model-to-plant mismatches is called robustness with respect to model uncertainty (e.g., [28], p. 253).

Robustness is sought in controller design by using various techniques to mathematically incorporate uncertainty in the formulation of the plant model—"representations of uncertainty"—thereby giving the controller based on that model an edge in handling modeling disparities. In general, this practice involves analyzing a set of plant models to see whether any of the candidates likely to be encountered by the prospective controller result in unacceptable performance. In other words, an attempt is made to discover what the worst-case result is likely to be. Chapters seven and eight in [28] and [30], respectively, contain detailed examinations of uncertainty modeling and analysis of robustness.

Step 8 in the design procedure seeks to evaluate controller robustness in a more practical way by dynamically varying the value of one of the physical characteristics modeled in the nonlinear plant model. This emulates the mathematical process described above by effectively creating a set of different plant models whose responses for each candidate controller can be evaluated.

The parameter varied in Step 8 is the one least precisely known, the magnetic flux density at the operating point. Of course, there is a high degree of confidence in this parameter for the 1-DOF system because the magnetic flux density was measured directly several times, as well as validated through earlier experiments. But in the 2-DOF and later configurations, the flux density will not be measured directly but calculated using a finite element analysis routine, which is itself based on a model of the MSBS electromagnetic environment. Furthermore, the force and torque equations for multiple magnet configurations are functions of several such calculated values, resulting in a multiplicative compounding of error. Thus it is reasoned magnetic flux density and its gradients are the parameters most responsible for modeling uncertainty.

The need for practical evaluation steps such as those in steps 7 and 8 of Table XVI is underscored by two factors: 1) the loose linkage between parameter selection and actual plant characteristics in optimal control design; and 2) the uncertainty of the stability margin produced by combining a given LSVF matrix with a Kalman estimator in an LQG solution.

Loop Transfer Recovery (LTR) Technique

Finally, this section introduces an alternate method for producing an LQG solution with greater likelihood of meeting the design goals is presented. This technique, called Loop Transfer Recovery (LTR), effectively incorporates indicators of closed-loop system response into each of the two major design steps: the Kalman estimator and LQR. First, a Kalman estimator (filter) is designed by iteratively adjusting the parameters discussed above to achieve a filter loop response that matches the corresponding response of the overall system. One or more different loop responses can be used as measures of suitability: the open loop, output sensitivity or complementary sensitivity function responses. When an acceptable estimator is obtained, the LQR weighting matrices **Q** and **R** are adjusted to produce an overall system response matching the one obtained for the Kalman estimator. In theory a match is possible—and the desired system response realizable—as the value of **R** approaches zero. The technique gets its name because using the loop transfer function, it "recovers" the robustness of the LQR solution lost by the introduction of the Kalman estimator.

Experimental Setup

This section explains the setup for the 1-DOF experiments using optimal control design. For the most part the setup duplicates that done for the earlier 1-DOF experiments, simply because the significant changes have more to do with the design process and digital controller implementation than with the physical system configuration. The same plant equations, flux densities, test object, equilibrium distance and hardware and software configuration were used as in the experiments covered in Chapter IV. (See Table IX and Figs. 18 and 20 in Chapter IV.) However, the system model in Simulink was significantly revised as was the operator GUI in Control Desk.

This section first describes the model for the LQG regulator and then shows how the nonlinear plant model was modified to enable the controller performance testing described in the previous section.

dSPACE Controller Model

The Simulink implementation of the LQG regulator in this set of experiments differs from the structure as illustrated shown in Fig. 45. In that case both inputs to the Kalman estimator, i.e.,

the control input *u(t)* and the plant output *y(t)*, along with the individual state space components of the estimator were explicitly shown. However, this set of experiments employed a single LQG regulator block whose transfer function was produced using the Matlab command *lqgreg.m*. This modeling structure combines the Kalman estimator with the LSVF matrix **K** as shown in Fig. 48.

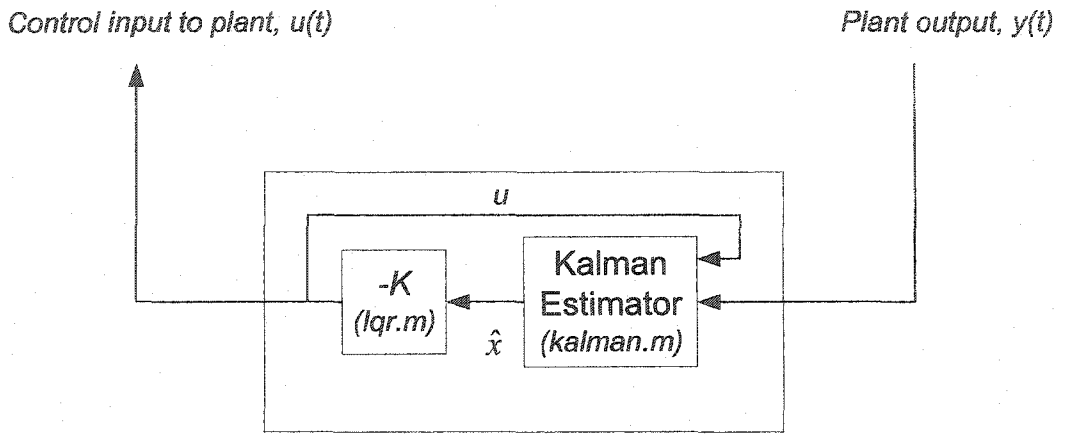


Fig. 48 Schematic diagram of LQG regulator structure showing Matlab commands [27].

Note there is only one external input to the regulator (the plant input is fed back internally). Also, positive feedback is used to connect the regulator block to the plant. The Matlab commands used to form the two subsystems are shown in italics.

Controller Validation Models for 1-DOF Optimal Control

Steps five through seven in Table XVI entail simulating the operation of a closed-loop 1-DOF system under different conditions. To conduct these simulations, three distinct models were created, the third being the most complex and the one with which the most of the experiments were conducted. It should be understood the creation of the controller validation models was an iterative process, cycling through three basic activities: parameter calculations in Matlab, model construction and trial runs of the controller design. Essentially, the validation models, the Matlab scripts used to run the simulations and a set of candidate controllers were all developed in parallel. The remainder of this section describes the structures of the validation models and the some of their associated design problems. The other aspects of 1-DOF controller design are covered in the next section, Experimental Procedure.

The first validation model, used in design step 4, was an LQR configuration consisting of a linear model of the plant in state space form and optimally derived state variable feedback matrix. (States were fed back directly rather than estimated.) This model was created and operated

entirely through Matlab commands without the use Simulink. Specifically, a set of optimal LSVF **K**-matrices was derived and a Matlab Linear Time Invariant (LTI) model of the closed-loop system was formed for each **K**. Then the closed-loop system response was obtained for a range of step inputs. (Stiffness and B-field perturbation tests were not done with the linear model.) The Matlab commands pertinent to this phase of testing are illustrated in Fig. 49.

```
K(i)=lqr(A_plant, B_plant, Q(i) , R(i));        derive optimal LSVF matrix
Ac=A_plant-B_plant*K(i);                  create pole-shifted state matrix
Gcl=ss(Ac, B_plant, C_plant, D);
ustep=ones(lent,1);                             create step input sequence
ustep=cmd*ustep;
[ystep]=lsim(Gcl,ustep,t);                            obtain step response
```

Fig. 49. Matlab commands to implement closed-loop LQR simulation

Fig. 48 illustrates Matlab implementation of equations (102) and (103). The command sequence shown assumes the existence of the state space variables A_Plant, etc., on the Matlab workspace. This is accomplished using the parameter derivation and linear modeling script described in the previous chapter. Note also the ranges of **Q**, **R** and **K** values are stored in arrays and the **K**-matrix carries the same index as the state and input weighting matrices that formed it.

The second model was also an LQR system but with a nonlinear model of the plant. This model was created in the process of developing the modeling structures and Matlab scripts needed to perform the stiffness and robustness tests described in Table XVI. The pertinent modeling blocks, all contained within the nonlinear plant model, are illustrated in Fig. 50.

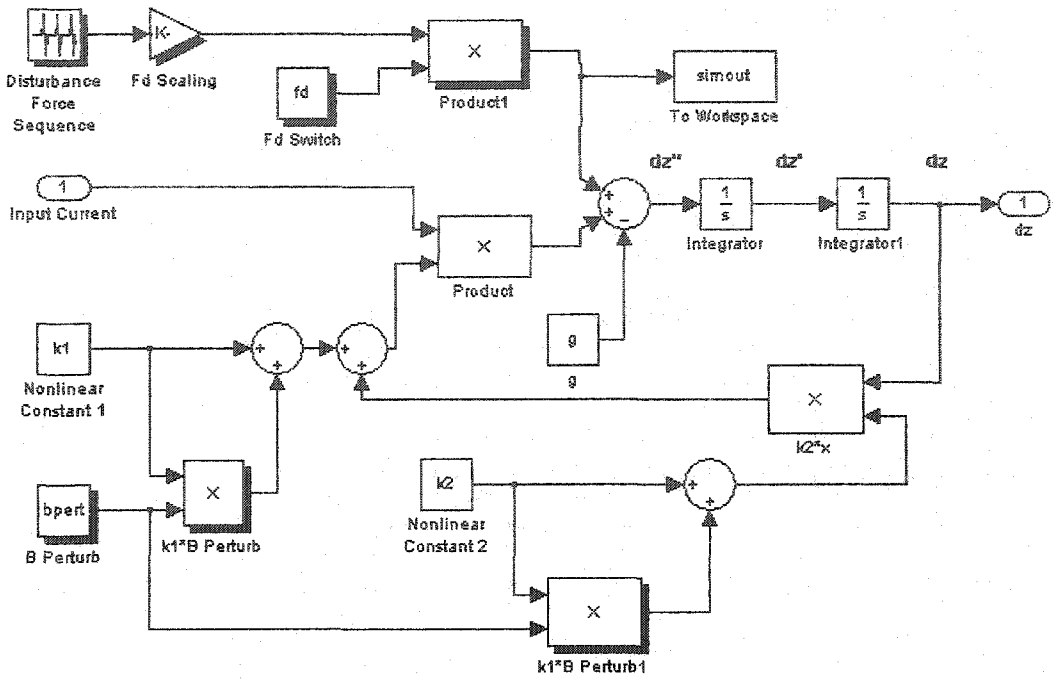


Fig. 50. Block diagram of the 1-DOF nonlinear model including vertical force and magnetic field disturbances.

Recall the dynamics of the nonlinear model, where the change in acceleration of the suspended object is related to the object's position through the equation

$$\partial\ddot{z} = \frac{vM_z}{m}\frac{i}{i_{\max}}b_{zz} + \frac{vM_z}{m}\frac{i}{i_{\max}}b_{zzz}\partial z - g\,.$$

The first alteration to the model was the introduction of a mechanism for determining controller stiffness. (Refer to the set of blocks along the top of the figure.) Since stiffness is a measure of how susceptible the controlled object is to external disturbances, it was reasoned a disturbing force could be simulated by introducing another vertical component of acceleration to the model. For example, to simulate a disturbance force of 1 Newton pressing upward on the suspended object, a constant equal to one over the object mass could be introduced at the point in the model where gravitational acceleration is added. The dynamics of the model would simulate the effect of the disturbance and in turn place a demand on the controller to adjust the system to equilibrium.

Then, by measuring the maximum deviation in the output, the controller stiffness could be calculated using the ratio

$$\frac{\text{Maximum Deviation (m)}}{\text{Disturbance Force (N)}}.$$

To make the stiffness tests more realistic with respect to the controller's eventual operational setting, the HRTF, a simple waveform was to devised to simulate the type of aerodynamic disturbance that might be encountered in a wind tunnel. Illustrated in Fig. 51., the disturbance "force" has a relatively sharp rise time and slower decay time. A Simulink *Repeating Sequence* block with parameters describing the waveform was used to generated the disturbance waveform.

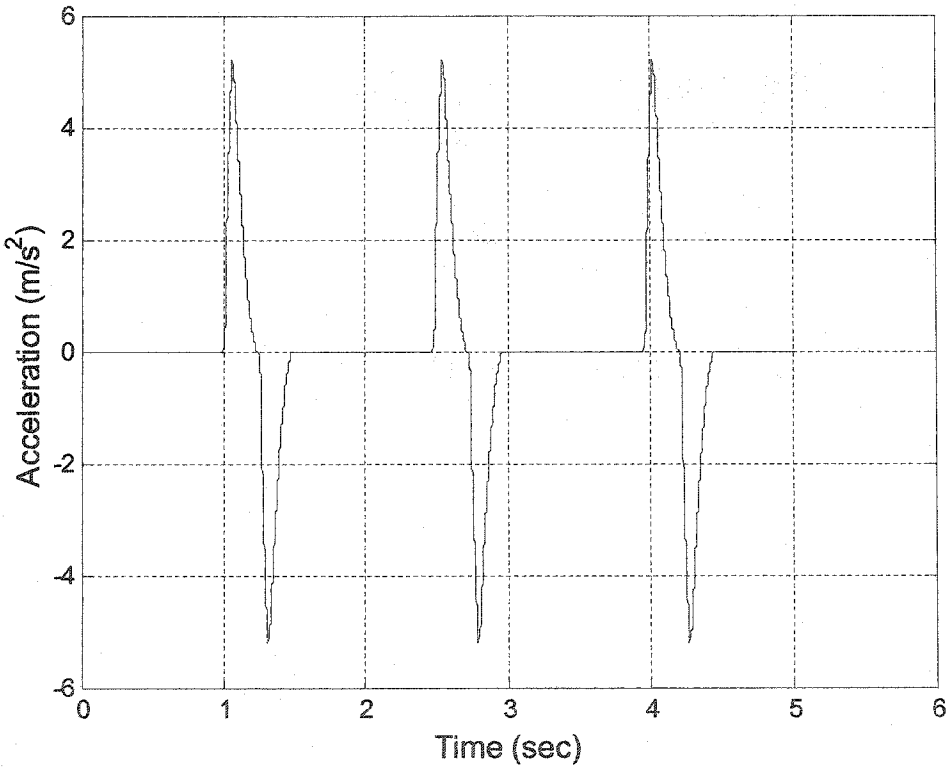


Fig. 51. Disturbance force waveform.

Actually, what is depicted in Fig. 51 is the disturbance "acceleration" applied in the 1-DOF experiments. This was obtained by dividing the output of "Disturbance Force" block by the reciprocal of the object mass, contained in block labeled "Fd Scaling". Because the mass of the object in this experiment is much less than a kilogram, the peak amplitudes are on the order of plus or minus 5.2 m/s^2. In the actual model construction, the waveform was built on 10-unit scale

for ease in creating the desired shape. Thus the scaling factor also contains a 0.1 reduction in order to simulate a 1-N force.

Note in Fig. 50 the disturbance waveform is multiplied by a factor labeled "Fd Switch". This provided a means to turn the disturbance on or off simply by setting a Matlab variable to zero or one from within a Matlab script. In this way, the same model could be used to run simulations both with and without the disturbance.

The second addition to the nonlinear model was a mechanism to test the controller's robustness in the face of modeling uncertainty. This was done by introducing perturbations to the nonlinear constants, k_1 and k_2, effectively changing the magnetic flux density within the model. As described in the previous section, the rationale for this test is that the linear controller can be considered robust if it performs well with an uncertain, nonlinear plant.

To simulate the most likely source of uncertainty in the maglev plant, the magnetic flux density, the nonlinear constants, k_1 and k_2 were randomly perturbed. These constants were convenient spots to introduce the changes since they're the only variables that carry magnetic flux information and they directly apply the magnetic flux density gradients to the model as seen in

$$k_1 = \left(\frac{vM_z}{mi_{\max}}\right) b_{zz} \text{ and}$$

$$k_2 = \left(\frac{vM_z}{mi_{\max}}\right) b_{zzz}.$$

The perturbations were implemented in the model by generating random percentages of k_1 and k_2 and adding them back to the constants themselves. This is illustrated in the lower left hand side of Fig. 50 by the blocks labeled "Nonlinear Constant 1", "B Field Perturb" and "K1 x B-Field Perturb". This additive method enables to the amount of perturbation to be randomly varied with a uniform probability distribution over a defined range centered on the constant's nominal value.

$$k_i' = k_i + Bpert \times k_i = (1 + bpert) k_i,$$

The variable *Bpert* is a random variable with range defined by a Matlab command

$$B_{pert} = (0.02*(2*\text{rand}-1)),$$

where the embedded command *rand.m* generates a uniformly distributed number on the interval (0.0, 1.0). In the inner parentheses, the random number is shifted to the interval (-1.0, 1.0), and the factor 0.02 scales that random number to the interval (-.02, .02). The result in the system

model is that the flux density gradients are randomly perturbed within a range of plus of minus two percent. For modeling simplicity, only one random variable is used, which is reasonable since the gradients are derived from linear operations on the flux density, and percentage changes in the density are carried through directly to the gradients. Also, just as with the disturbance forces, there is a switch that allows the same model to be used for perturbed and non-perturbed simulations.

The third validation model was the last produced during the experimental set up. This model contained an LQG regulator and the nonlinear plant model . Both were similar that only the LQG system is shown below in Fig. 52.

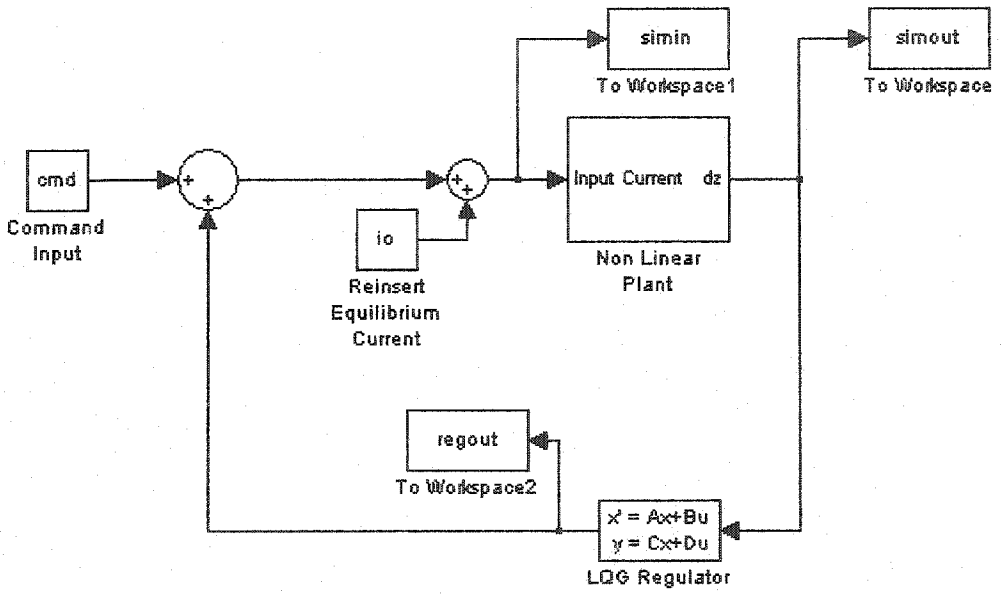


Fig. 52. Nonlinear system model for LQG regulator validation.

This model contains a linear controller and nonlinear plant just as the earlier 1-DOF models. Consequently, the bias, or equilibrium current is supplied to the model. However, the feedback path contains the regulator in state space form and feedback is positive, due to the internal structure of the LQG regulator block. There are three blocks, *simin*, *simout* and *regout*, that Simulink uses to place data from the model into arrays with the same names on the Matlab workspace. These arrays contain the time responses of the simulation at intervals equal to the simulation step size, 0.001 seconds for this set of experiments.

The variable in the "Command Input" block in Fig. 52 is not simply the desired output position, as in the earlier 1-DOF experiments. This has to do with the difference between the output feedback structure used before and the regulator control law used here. Recall that the basic regulator control law, $u(t) = -Kx(t)$, assumes there is no external reference input, i.e., the set point is zero. (The set point is the equilibrium reference for the system.) If different equilibrium points are desired, which is what is needed for the MSBS, then the control law and system equations have to be rewritten

$$u(t) = -Kx(t) + v_{des}(t)$$

$$\dot{\mathbf{x}}(t) = (\mathbf{A} - \mathbf{BK})\mathbf{x}(t) + \mathbf{B}v_{des}(t).$$

Switching to transfer function notation, the input-output relationship is given by

$$y_{des}(t) = G_{CL}(t)v_{des}(t)$$

The equilibrium condition is equated to the steady state condition of a stable system as $t \to 0$. In Laplace transform notation, the steady state output of the plant with a step input of magnitude v_{des} is expressed as

$$y_{des}(s) = \lim_{s \to 0} sG_{CL}(s) \cdot \frac{v_{des}}{s}.$$

This corresponds to the following

$$y_{des} = dcgain(G_{CL}(s)) \cdot v_{des},$$

which leads to

$$v_{des} = inv\left[dcgain\left(G_{CL}(s)\right)\right] \cdot y_{des}.$$

This is the simplest means to determine non-zero set points for the closed-loop regulator. For controller validation tests, the variable *cmd* was defined within a Matlab script using a command sequence similar to the following:

```
Gp = ss(A_plant,B_plant,C_plant,D_plant);
Gcl = feedback(Gp,Klqg,+1);
Icl = inv(dcgain(Gcl));
cmd = Icl*out_des.
```

In this way the set points for the regulator could be easily changed to determine the range over which the linear regulator and state estimator could effectively operate.

To implement a real-time controller for the MSBS hardware, the dSPACE model used for the last set of 1-DOF experiments was modified to incorporate the LQG regulator block shown in Fig. 52. In addition, the whole LQG nonlinear system was added as a subsystem in order to allow the controller to be tested with the plant and nonlinear model simultaneously.

Experimental Procedure

The purpose of this set of experiments was to gain familiarity with optimal control design in the relatively simple 1-DOF case so that multivariable controller design could be carried out more efficiently. Most of the experimental procedure for these experiments consisted of developing the Simulink models and Matlab scripts needed to automate the design process described in Table XVI. By the time these design "tools" were refined, several possible controller designs had been identified, leaving only a few relatively simple tests using the dSPACE real time controller and the actual plant. The previous section described the structure of the simulation models and this section explains the functions of the Matlab scripts.

To execute either the 11-step design process in Table XVI or the LTR method outlined in the previous section, both of which potentially involve many iterations, it was necessary to automate as many steps as possible. This involved developing a set of Matlab scripts to manage the design parameters, invoke the necessary commands to create the LQG regulator, simulate the controller operation and organize and evaluate the simulation results. This section first describes the functionality of the different Matlab scripts developed for the experiment and highlights the salient design issues associated with each. Secondly, the section briefly describes the controller test with the actual MSBS hardware. Finally there is a comparison between an optimally designed controller using the tools created in this chapter and the lead-lag controller described in the previous chapter.

Functionality of the Matlab Scripts

The first Matlab script was the relatively simple one used to evaluate an LQR design using the Linear Plant model—as described in Fig. 48 in the previous section. This experiment, corresponding to design steps two through four, provided a simple, preliminary evaluation of a range of values for the state and input weighting matrices, $\mathbf{Q}$ and $\mathbf{R}$. This part of the experiment also provided an opportunity automate the creation of a series of LSVF coefficients for a regulator without the encumbrance of programming other functions.

The next stage in the design process was to implement a series of LQR simulations using the nonlinear plant model and feed back the state variables directly. (This involved the second model described in the last section.) This programming in this stage addressed a significant amount of the functionality for controller validation without having to handle the additional parameters associated with state estimation. Following were the important programming functions implemented:

TABLE XVII
FUNCTIONALITY IMPLEMENTED IN SECOND STAGE OF DESIGN AUTOMATION

Sequence No.	Description of Functionality
1	Declare the variables needed in the Matlab workspace. This provided two benefits: 1) it reduced the number of variables that had to be passed between routines; and 2) it allowed the parameters in the Simulink model to be easily changed.
2	Derive a state space model of the plant and calculate constants k_1 and k_2 needed for the nonlinear plant model.
3	Determine whether to use a default set of values for the weighting matrices **Q** and **R** or take user input for the values. This experiment was designed to allow a "shotgun" approach to the selection of **Q** and **R**, where a wide range of values could be tried first and then narrowed down in successive tests.
4	Construct an array of values for **Q** and **R**, based on the facts that 1) **R** is a scalar for this system and 2) **Q** is a 2 x 2 diagonal matrix containing two different values. For ease of programming, both elements of Q were given the same range. Thus the number of permutations of design parameters was equal to the square of the number of values in the Q-range times the number of elements in the R-range. The column order of the parameter array was $[r_1, q_{11}, q_{22}]$.
5	For every permutation of these values find the optimal LSVF matrix, **K**, and the repositioned eigenvalues, E, by using the Matlab command *lqr.m*. Store resulting **K**'s and E's in an array and maintain association between them and their corresponding Q and R pair.
6	Implement a range of regulators based on the stored feedback matrices and obtain step input responses for each system using the both linear model and nonlinear Simulink model. Allow default range of set-points or user defined range. Store step responses in vectors.
7	Evaluate the step responses to determine whether they meet design specifications for time response. (E.g., percent overshoot, settling time and final value.) This was done by evaluating each step response vector against a set of threshold criteria adjustable by the user. Store responses meeting criteria along with the corresponding **Q** and **R** values.

TABLE XVII CONT'D

Sequence No.	Description of Functionality
9	Provide user the option to perform controller stiffness tests. Stiffness test done as follows:
	Determine range of set points, either default or user specified. This range is used to establish a process loop.
	Create closed-loop transfer function for the pole-shifted plant, invert the transfer function and determine its DC gain. Use the DC gain to calculate reference input needed to produce the desired set point.
	For the current reference input, obtain the step response of non-linear system without disturbance force. Determine final value of response as baseline for calculating stiffness ratio.
	For the same reference input, "turn on" the force disturbance in the model and obtain step response. Evaluate the response for its max/min values and calculate the positive and negative stiffness ratios. Use value of the amplitude peaks of the disturbance force divided by the maximum positive or negative deviations of the step response from its final value, e.g., $\text{Stiffness Quotient} = \frac{\text{Disturbance Force (N)}}{\text{Max deviation perturbed sys. -Steady state unpertured sys. (m)}}$
	Store stiffness ratios in an array. Repeat for the next set point in the range. .
10	Provide user the option to test the controller with **B**-field perturbations.
	Determine range of set points, either default or user specified.
	Create closed-loop fransfer function for pole-shifted plant, invert the transfer function and determine its DC gain. Use the DC gain to calculate the reference input needed to produce desired set point.
	Copy non-perturbed step responses into new matrix
	Conduct programmed number of simulations with current reference input. For each simulation, generate new random perturbation factor and obtain step response. Evaluate each step response for percent overshoot, settling time and final value.
	Take average of all response values for each reference input. Store response values with corresponding Q and R values.
11	If necessary, repeat entire sequence for different range of **Q** and **R** values.

The third stage of automating the 1-DOF design process involved incorporating the Kalman estimator into the controller design using the Matlab commands *kalman.m* and *lqgreg.m,* described in the earlier section on Experimental Setup. The first of these commands creates a Kalman estimator using the state space model of the plant and noise covariance matrices, $\mathbf{Q}_n$ and $\mathbf{R}_n$, provided by the designer. The second command creates a single regulator block by joining the LSVF matrix **K** with the Kalman estimator (see Fig. 48). Joining the programming and modeling functions produced in this stage with those done earlier resulted in a complete set of

Matlab scripts for implementing a semi-automated design process along the lines of Table XV. The following describes the functions added under this stage.

TABLE XVIII
FUNCTIONALITY IMPLEMENTED IN THIRD STAGE OF DESIGN AUTOMATION

Sequence No.	Description of Functionality
1	Give user (designer) the option to re-create the range of LSVF matrices before proceeding.
2	Create a Kalman state estimator using single set of values for the design parameters $\mathbf{Q}_n$, $\mathbf{R}_n$ and $\mathbf{G}_w$. This is the start of the outermost processing loop. The command syntax is [Kest]=kalman(sys_k, Q_n, R_n), where sys_k represents the plant augmented by process and measurement noise vectors, w and v. [Mat1] The state space equations are $\dot{\mathbf{x}} = \mathbf{Ax} + \mathbf{Bu} + \mathbf{G}_w\mathbf{w}$ $\mathbf{y}=\mathbf{Cx}+\mathbf{Du}+\mathbf{Hw}+\mathbf{v}$, where G_w and H are process noise gain matrices. Thus, sys_k is formed by the command sys_k=ss(A_plant, [B_plant G_w], C_plant, [D_plant H]). The resulting Kalman estimator takes an input vector $\begin{bmatrix} u & y \end{bmatrix}^T$ and outputs a vector $\begin{bmatrix} \hat{y} & \hat{x} \end{bmatrix}^T$. Because the 1-DOF maglev is a SISO system, $\mathbf{Q}_n$, and $\mathbf{R}_n$ are scalars and $\mathbf{G}_w$ is a vector.
3	Determine range of set points, either default or user-specified. Selection of the first value in this range is the first step in the first inner process loop.
4	For each LSVF matrix K created earlier in the process, form the LQG regulator using the command lqgreg.m. The command syntax is Lqgr = lqgreg(K, Kest). Selection of the first value in the range of K-matrices is the fist step in the second inner process loop.
5	With the current regulator and plant model, create the closed-loop transfer function and then invert the transfer function and determine its DC gain. Then use the DC gain to calculate the reference input needed to produce desired set point.
6	For each regulator, obtain the step response of non-linear system and evaluate for percent overshoot, etc. as before. Store time response values in an array along with the sequence number of the K-matrix used to form the regulator. The sequence number can be used to trace acceptably performing regulators to their design parameters.
7	Conduct controller stiffness and B-field perturbation tests as described in Table XVI.
8	Repeat regulator formation and simulation for the next value for K, i.e., the next value in the inner loop.
9	When all values for K tested, proceed to next set point value, or next value in the outer loop.
10	All time response values are stored in a three-dimensional array (K, Resp, SP), where K is the no. of LSVF matrices, Resp is the size of the vector storing the time response and SP equals the no. of set points tested.
11	If desired, return to the outermost processing loop and create a new Kalman estimator.

After to automating the procedure described in Table XV, the Loop Transfer Recovery (LTR) method was likewise implemented. Using this method, evaluation of the frequency response could by added to the validation process. As mentioned earlier, the basic two-part design is reversed in the LTR method. The Kalman estimator is designed first, based on the frequency response of its open loop transfer function. Then the LSVF matrix is tuned to provide an overall closed-loop system response that matches the response obtained for the Kalman estimator alone. The last stage of programming for this experiment was to implement an LTR design tool containing the functionality described below.

TABLE XIX
FUNCTIONALITY IMPLEMENTED IN LTR DESIGN AUTOMATION

Sequence No.	Description of Functionality
1	Declare variables needed in the Matlab workspace and derive a state space model of the plant and calculate constants k_1 and k_2.
2	Synthesize and evaluate a range of Kalman estimators using fixed values for the process and sensor noise covariance matrices, Qn and Rn, and a range of values for the process noise gain matrix, Gw. Select estimator giving best response. The frequency response of the loop transfer function for each resulting filter is plotted on screen for user evaluation. Frequency response is determined in terms of the singular values of the transfer function. Filter selection is done in two steps: first the user chooses up to four candidate filters based on the responses; then all the candidate response are displayed on a single graph and the user selects the best from that group.
3	Synthesize and evaluate one or more LQG regulators using state space models of the plant and Kalman estimator. Choose LQG regulator that provides open loop frequency response for the overall system (plant and regulator) that most closely matches response of Kalman estimator. Derive the linear state variable feedback matrix, **K**, using the command *lqr.m*. In this implementation state and input weighting matrices **Q** and R are fixed: **Q** = (**C**_plant' x **C**_plant) and R = 1. (R is a scalar for SISO systems.) The only parameter to be varied in the LQR design is the input weight R, which the user does by selecting either one of a range of values for an R-matrix multiplier, rho. The command syntax looks like $[\mathbf{K},\mathbf{S},\mathbf{E}]=LQR(\mathbf{A},\mathbf{B},\mathbf{Q},\mathbf{R}_{base}\times\rho)$, where R_{base}=1 and ρ is the user-selected multiplier.
4	Using the command *lqgreg.m*, form the LQG regulator from **K** and the estimator model provided. Produce the open loop transfer function for the overall system by placing the plant model and regulator in series.
5	In the case of multiple regulators, the user selects up to four candidate responses and then down-selects the best from that group. Frequency response is determined in terms of singular values of the transfer function.
6	Obtain step responses and perform stiffness and B-field perturbation tests as described in previous stages.

The Matlab script files created to perform these functions are provided in electronic form as a separate attachment to this thesis. However, Appendix E contains an index of the files and table listing the Matlab scripts created to achieve the functionality just described.

Upon completion of the design tool, several design iterations were conducted and candidate controller designs produced. Table XX summarizes the final design parameters selected as well as the transfer function settled on in the classic 1-DOF design covered in Chapter IV.

TABLE XX
CONTROLLER DESIGN PARAMETERS

Controller	Q_n	R_n	G_w	Q	R	Transfer Function
LQR	---	---	---	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$	1e-6	---
LQG	1	1	$\begin{bmatrix} 500 \\ 5000 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$	1e-6	---
LTR	1	1	$\begin{bmatrix} 400 \\ 8000 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$	1e-9	---
Lead-Lag	---	---	---	---	---	$9000\times\frac{(s+9.68)(s+2.0)(s+1.5)}{(s+97)(s+0.5)(s+0.1)}$

MSBS Hardware Test

Certainly the true validation of a controller lies in the demonstration of its performance as part of a real-world system. Therefore, once a likely candidate was obtained through the software validation steps, it was incorporated into an RTI-enabled Simulink model of the entire controller system, compiled and loaded onto the RTP as described in the previous chapter. In this experiment with the actual MSBS plant the LTR-designed controller described in Table XX was used. The dSPACE GUI was set up to capture the following variables: 1) the control input to the nonlinear model; 2) the control input to the plant (suspension coil current); 3) The output of the nonlinear model; and 4) the object displacement from equilibrium in the plant. Fourteen five-second data captures were conducted during which the input commands ranging from +7 mm to --7 mm. The system was unable to track the command -7 mm.

Comparison of LTR-Designed and Classic Controllers

Lastly, the stiffness and robustness against B-field perturbations of the lead lag controller from Chapter IV were measured. The Matlab script for the LTR design was modified to accomplish this.

Presentation and Discussion of Results

As noted above, the activities associated with this experiment consisted primarily of designing Simulink models of different controller configurations and creating the Matlab scripts needed to automate the use of these models. In addition, a lesser amount of time was required to validate a controller design with the actual MSBS equipment in a sample 1-DOF plant. This section presents some typical system responses obtained while developing the models and scripts and while validating a controller on the actual MSBS 1-DOF plant. The design parameters used are those listed in Table XX.

The experimental results are arranged to provide three types of comparisons. First a comparison is made between three different design approaches: 1) an LQR controller with full state feedback; 2) the same LQR only using a Kalman state estimator; and 3) a regulator with Kalman estimator synthesized using the LTR design method. Second, there is a comparison between the performance of a controller with a nonlinear model and the actual 1-DOF plant. Lastly, the stiffness and robustness of the classical controller described Chapter IV are compared to the same performance measures of an optimally designed regulator.

Comparison of Three Approaches to Optimal Design

The following plots and tables contain selected results from tests of the three different design approaches: LTR alone, LQG regulator from the iterative design procedure in Table XVIII and LTR-designed LQG regulator from the procedure in Table XIX. The results are arranged so to compare the three design approaches under similar operating conditions. First several groups of step input response plots are presented, followed by numeric data describing the different controllers' time response characteristics, their stiffness quotients and their robustness against B-Field perturbations. Figs. 53 to 55 illustrate the system responses to step inputs for the three different design approaches.

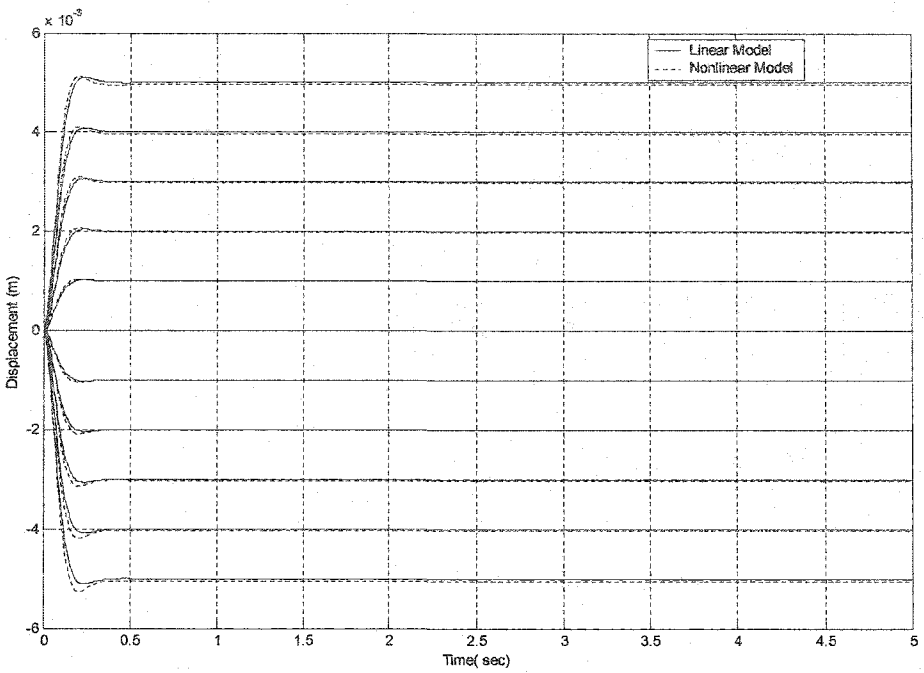


Fig. 53. Step responses of both linear and nonlinear models from simulation with LQR with full state feedback for range of set points: -5 to 5 mm.

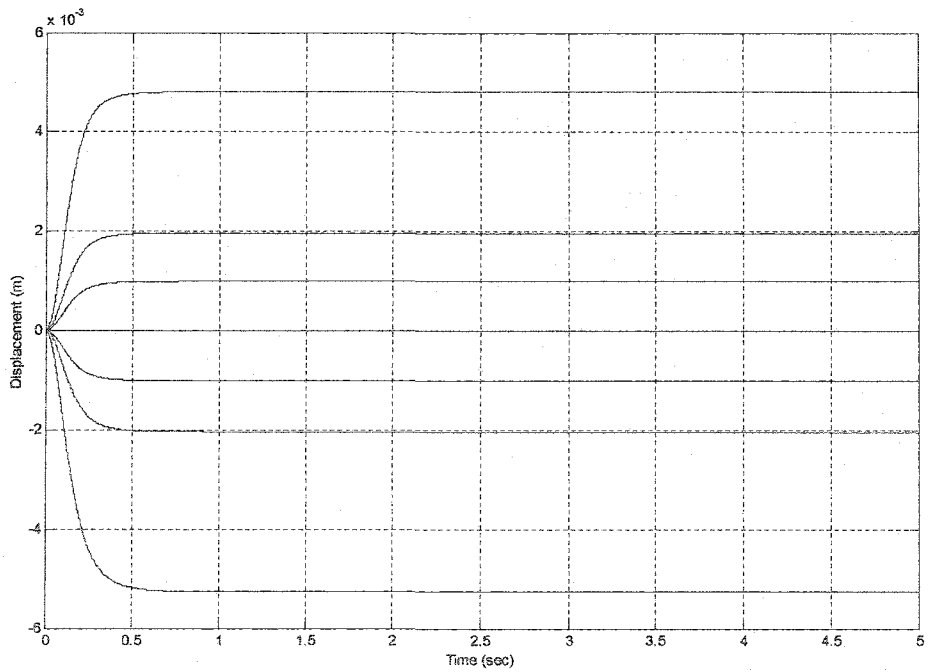


Fig. 54. Step responses of nonlinear model during siumulation with LQG regulator with state estimation for range of set points: -5, -2, -1, 0, 1, 2, 5 mm.

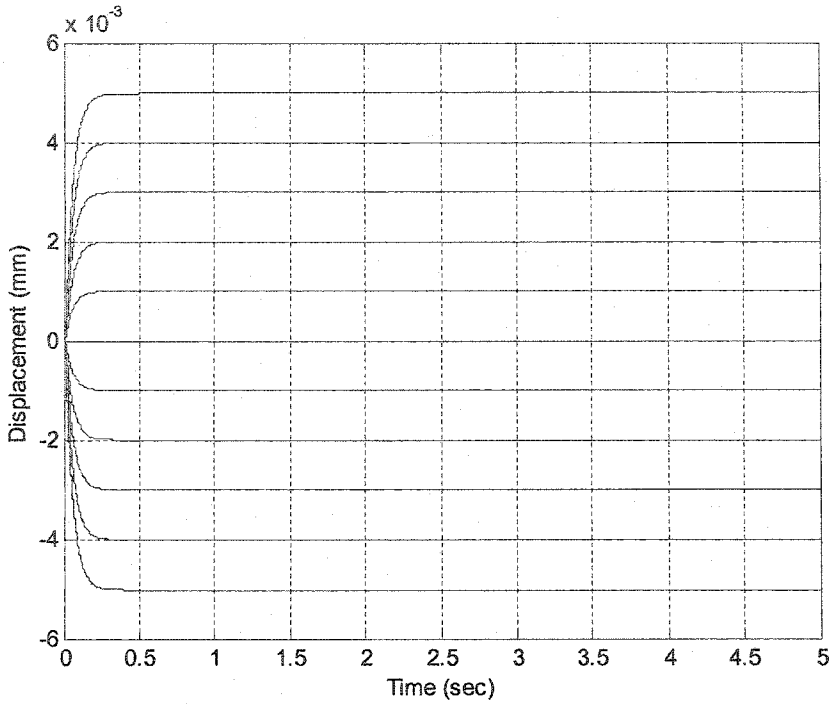


Fig. 55. Step responses of nonlinear model during simulation with LTR-designed LQQ regulator for range of set points: -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5 mm.

The next set of plots, Figs. 56 to 58, illustrate the system responses at set points -5 mm, 0 mm, and +5 mm, when a disturbance force in the output is simulated. These plots provide a good visual indication of the controllers' relative stiffness.

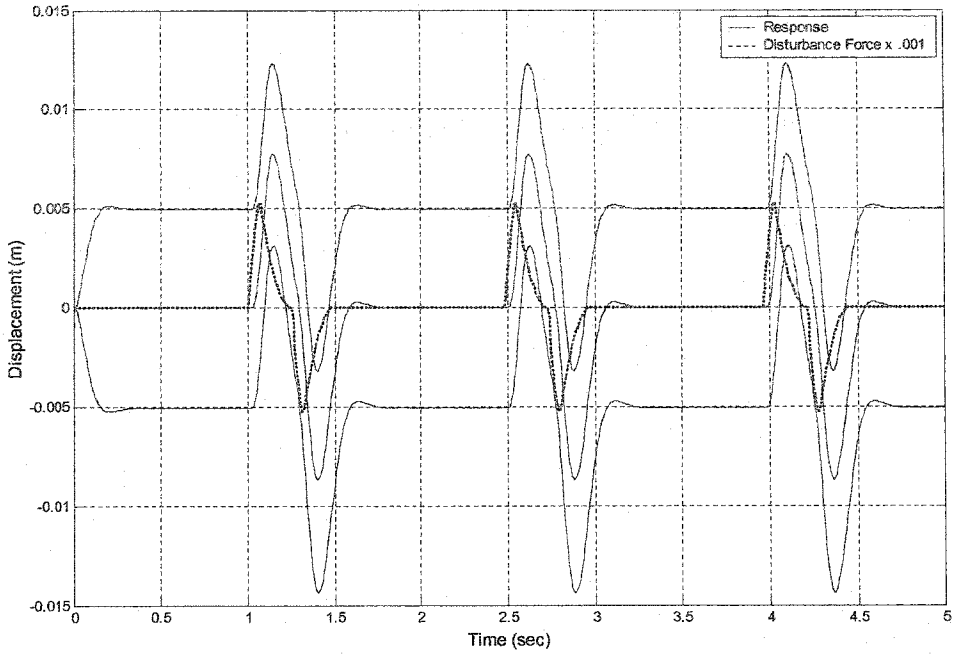


Fig. 56. Stiffness test responses for nonlinear model during simulation with LQR and full state feedback.

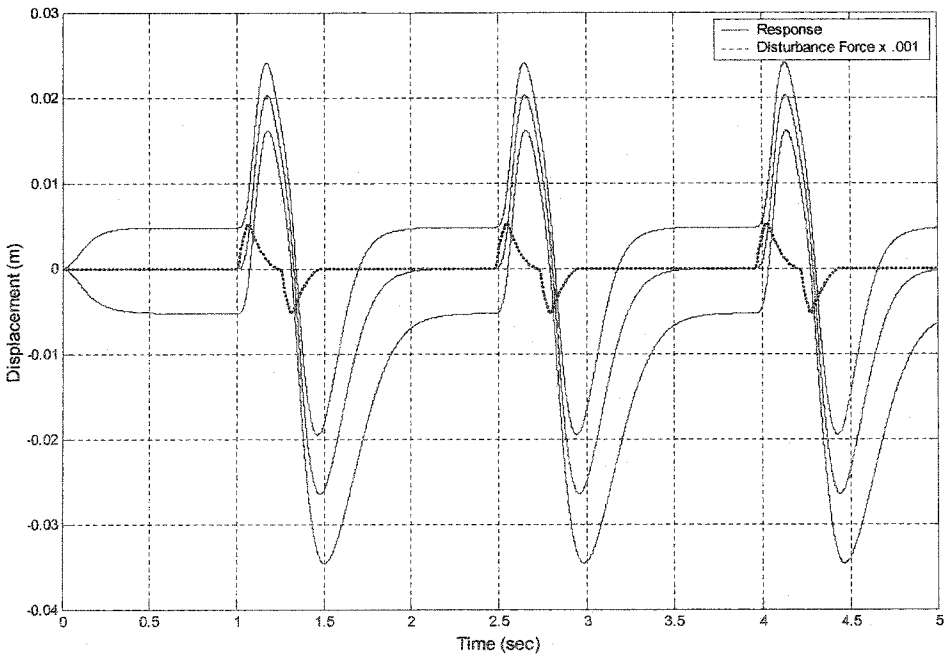


Fig. 57. Stiffness test responses during simulation for nonlinear model with iteratively-designed LQG regulator.

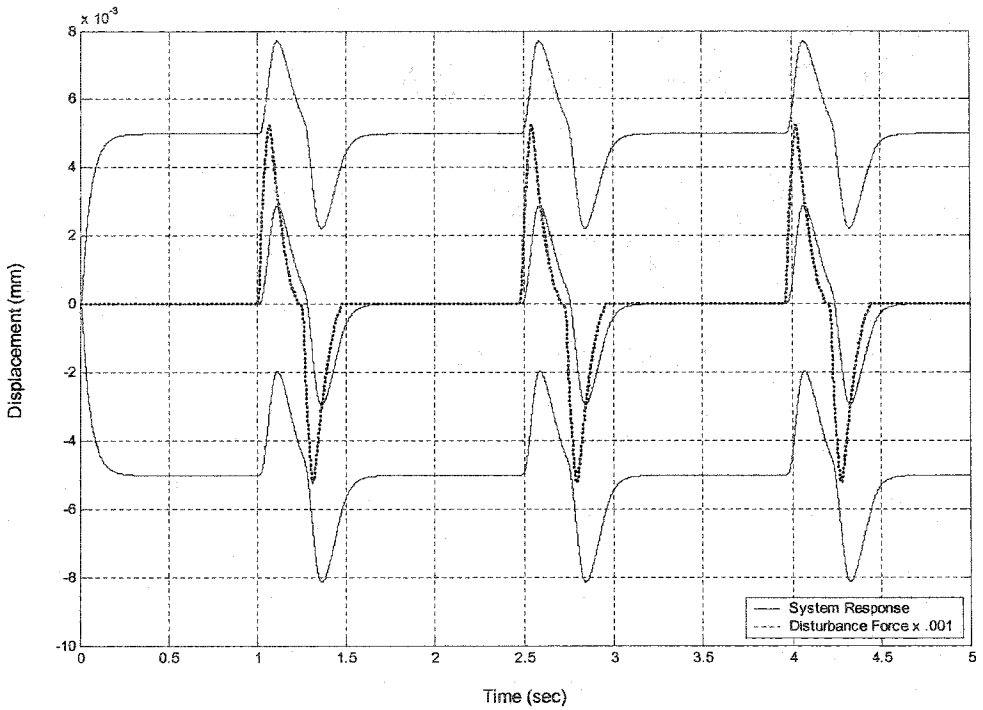


Fig. 58. Stiffness test responses for nonlinear model during simulation with LTR-designed LQG regulator.

Figs. 59 to 61 show the range of responses for different controllers when the B-fields in the nonlinear plant models are "perturbed".

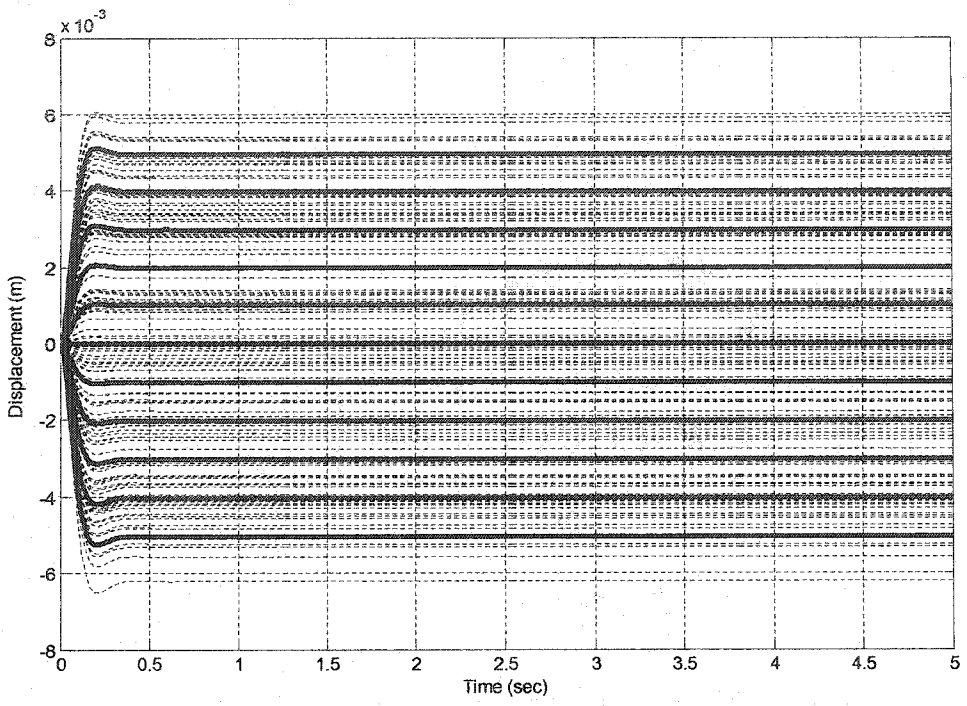


Fig. 59. Responses of nonlinear model during simulation with LQR and full state feedback when perturbations applied to magnetic flux density over range of set points -5mm to +5mm.

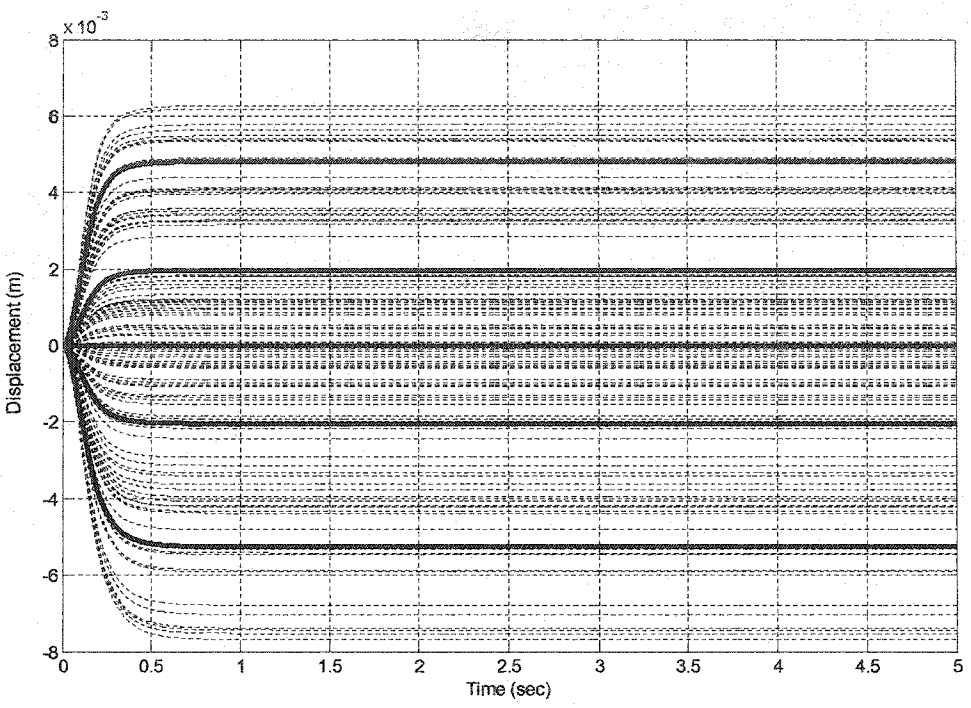


Fig. 60. Responses of nonlinear model during simulation with iteratively-designed LQG Regulator when perturbations applied to magnetic flux density at set points: -5 mm, -2 mm, 0 mm, +2 mm and +5 mm.

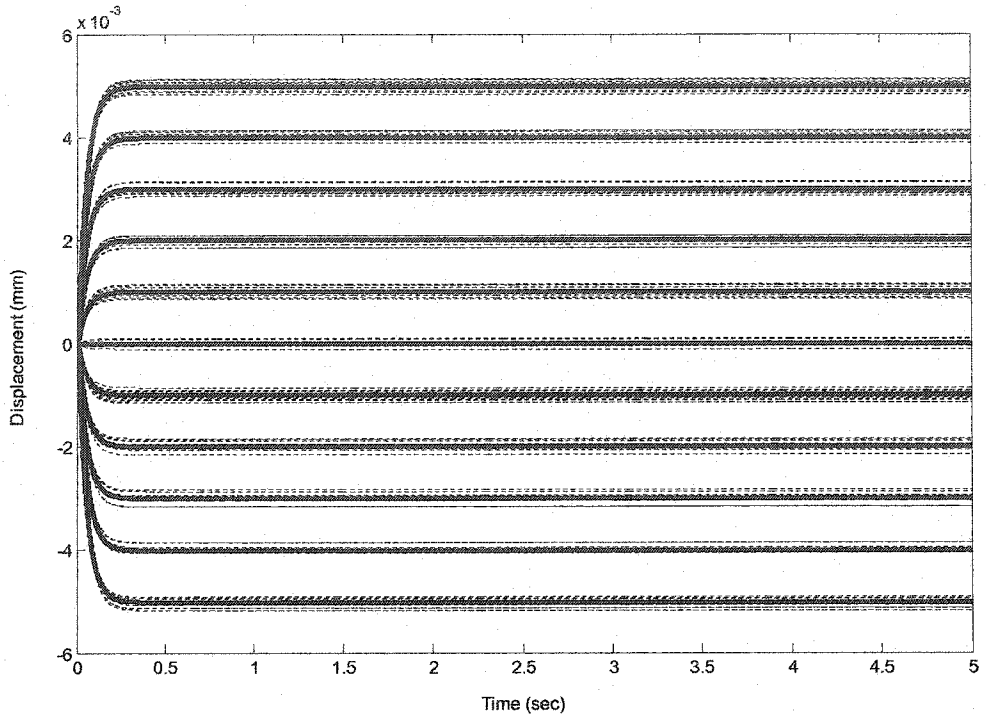


Fig. 61. Responses of nonlinear model during simulation with LTR-designed LQG Regulator when perturbations in magnetic flux density applied over range of set points -5 mm to +5 mm.

Fig. 62. illustrates the system open loop frequency responses obtained during the LQR, LQG and LTR designs. Note the slight decrease in bandwidth from the LQR to the LQG design and the large bandwidth increase from LQG to LTR. Fig. 63 illustrates the Kalman filter and system responses used for the LTR design.

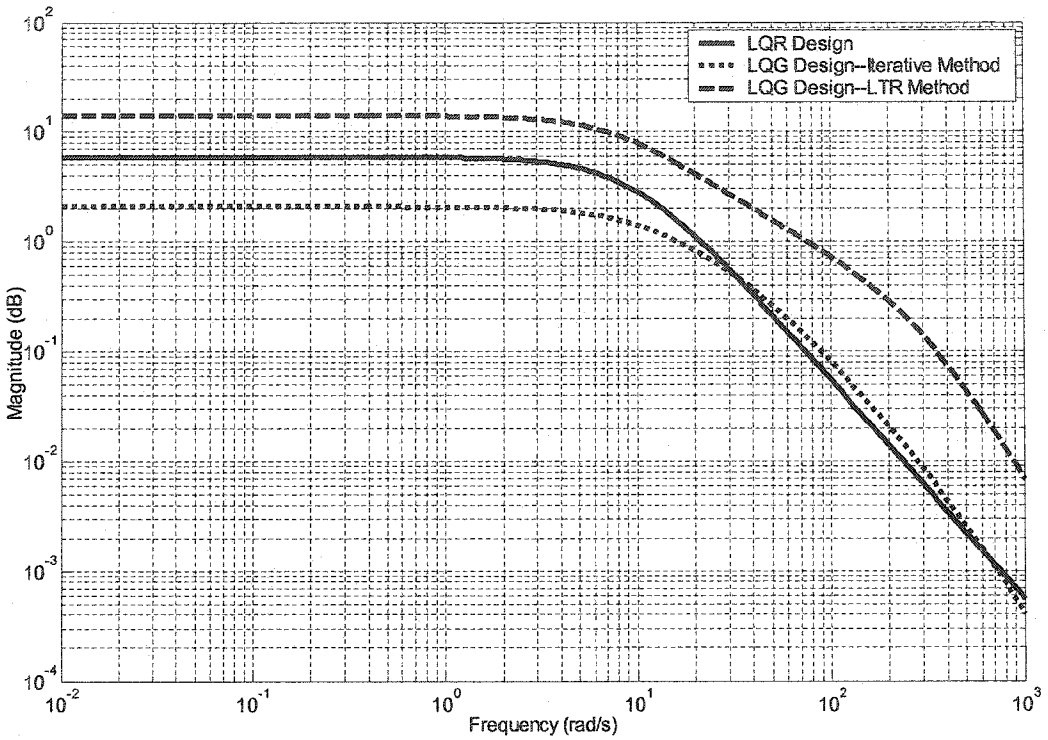


Fig. 62. System open loop responses for LTR, iteratively-designed LQG and LTR-designed LQG controllers.

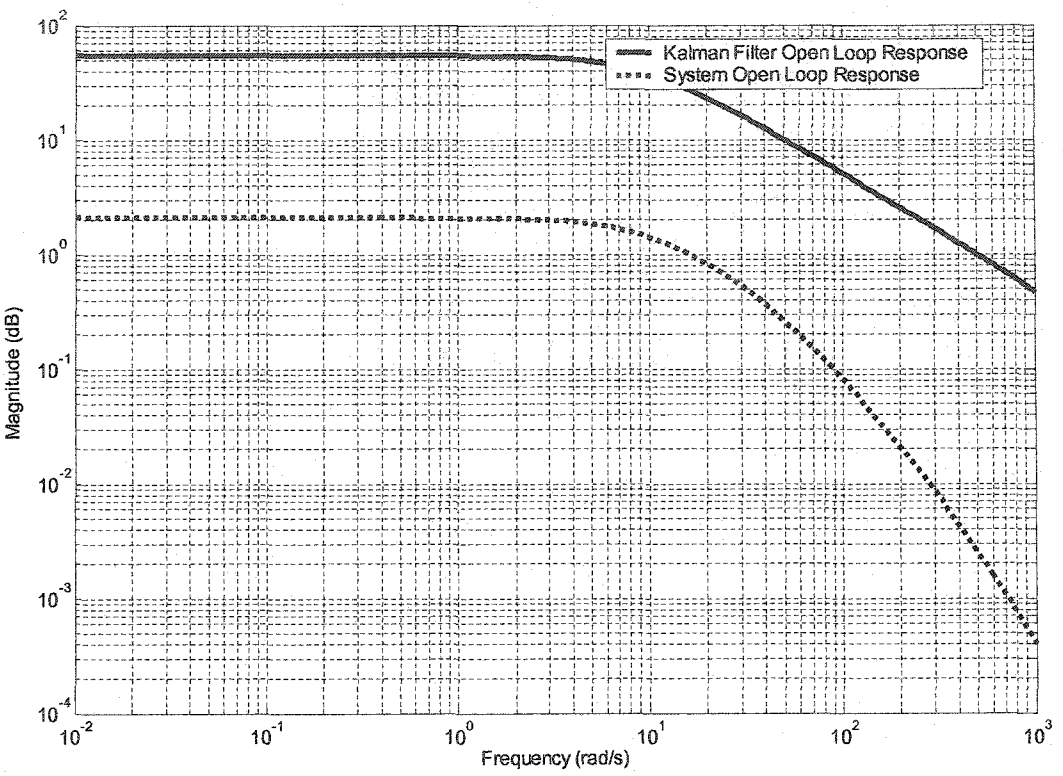


Fig. 63. Frequency response plots of open loop transfer functions used for LTR-designed controller.

Next, selected numeric data are presented from the different controller simulations. These data were obtained automatically by capturing the responses and performing the calculations using Matlab scripts.

TABLE XXI
STEP RESPONSE DATA FOR LQR, LQG AND LTR REGULATOR DESIGNS

Set Point (mm)/Design Type	% OS	2 % Settling Time	Steady State Value
-5 LQR (lm) [a]	2.0905	0.1900	-.00050
-5 LQR	5.1562	0.1700	-0.0051
-5 LQG	5.1624	4.7300	-0.0053
-5 LTR	0.3861	0.2030	-0.0050
-2 LQR (lm)	2.0905	0.1900	-0.0020
-2 LQR	4.2630	0.1700	-0.0020
-2 LQG	1.8775	4.4800	-0.0020
-2 LTR	0.1492	0.2030	-0.0020
-1 LQR (lm)	2.0905	0.1900	-0.0010
-1 LQR	3.9779	0.1700	-0.0010
-1 LQG	0.9119	4.4100	-0.0010
-1 LTR	0.0738	0.2040	-0.0010
1 LQR (lm)	2.0905	0.1900	0.0010
1 LQR	3.4259	0.1700	0.0010
1 LQG	-0.8635	4.2900	0.0010
1 LTR	-0.0722	0.2040	0.0010
2 LQR (lm)	2.0905	0.1900	0.0020
2 LQR	3.1585	0.1700	0.0020
2 LQG	-1.6832	4.2400	0.0020
2 LTR	-0.1428	0.2040	0.0020
5 LQR (lm)	2.0905	0.1900	0.0050
5 LQR	2.3894	0.1700	0.0050
5 LQG	-3.9154	4.1200	0.0048
5 LTR	-0.3459	0.2050	0.0050

[a] (lm) denotes that design approach was tested with linear model

TABLE XXII
STIFFNESS TEST DATA FOR LQR, LQG AND LTR REGULATOR DESIGNS

Set Point (mm)	Steady State Value (m)	Positive Stiffness Ratio	Negative Stiffness Ratio	Average Stiffness Ratio
-5	-0.0051	122.586	-107.631	115.108
	-0.0053	46.5691	-34.0359	40.3025
	-0.0050	328.816	-318.846	323.831
0	0	129.647	-115.136	122.392
		49.3041	-37.8255	43.5648
	0	347.713	-339.160	343.437
5	0.0050	136.182	-122.600	129.391
	0.0048	51.7796	-41.0580	46.4188
	0.0050	366.536	-359.330	362.933

TABLE XXIII
B-FIELD PERTURBATION TEST DATA FOR LQR, LQG AND LTR REGULATOR DESIGNS

	Percent Overshoot		2% Settling Time (s)		Steady State (m)		Max. Input Deviation	
Set Point (mm)	Mean	Standard Deviation	Mean	Standard Deviation	Mean	Standard Deviation	Mean	Standard Deviation
-5-LQR	7.5829	13.2781	0.1700	0.0000	-0.0052	0.0006	2.2257	0.8870
-5-LQG	4.5048	28.1321	0.4743	0.0172	-0.0052	0.0014	1.1523	0.5982
-5-LTR	1.1647	1.9622	0.2025	0.0005	-0.0051	0.0001	4.0940	0.3035
-2-LQR	23.9497	35.1267	0.1700	0.0000	-0.0024	0.0007	1.3434	0.9531
-2-LQG	0.1664	59.6546	0.4481	0.0110	-0.0020	0.0012	0.4295	0.5183
-2-LTR	-0.6779	4.8133	0.2033	0.0005	-0.0020	0.0001	1.5315	0.3215
0-LQR	0.0006	0.0003	0.1700	0.0000	0.0001	0.0007	0.3466	0.5087
0-LQG	0.0009	0.0006	0.4361	0.0086	-0.0001	0.0011	0.2256	0.3024
0-LTR	0.0001	0.0000	0.2036	0.0005	-0.0000	0.0001	0.1144	0.1352
2-LQR	-5.0287	38.2850	0.1700	0.0000	0.0018	0.0007	2.0431	0
2-LQG	0.9118	60.0971	0.4245	0.0073	0.0020	0.0012	0.4476	0
2-LTR	-0.2668	4.6384	0.2043	0.0005	0.0020	0.0001	5.4413	0
5-LQR	-0.7378	8.2290	0.1700	0.0000	0.0048	0.0004	5.1078	0
5-LQG	-9.8757	21.3230	0.4136	0.0049	0.0045	0.0011	1.1191	0
5-LTR	-0.3069	2.0279	0.2048	0.0006	0.0050	0.0001	13.6032	0

Comparison of Results from Nonlinear Model and Plant

The next set of results compares the performance of the nonlinear model and the actual 1-DOF plant using the same LTR-based controller design. (This was the same LTR design described in Table XX.) Both model and plant were operated simultaneously and shared the same reference input. Fig. 64 shows the responses of both modeled system and actual plant/controller combination systems to a series of step commands. Then, Fig. 65 highlights the difference in response of the two systems with respect to two performance measures, the control input signal and the object position. The top plot shows the difference between the two maximum control input deviations over the range of command inputs (MSBS current minus model current). For all inputs the actual MSBS plant experienced smaller excursions in control current. The lower plot in Fig. 66 illustrates the differences in steady state output (MSBS object position minus model object position). For every step command, the magnitude of the object's displacement was less with the MSBS.

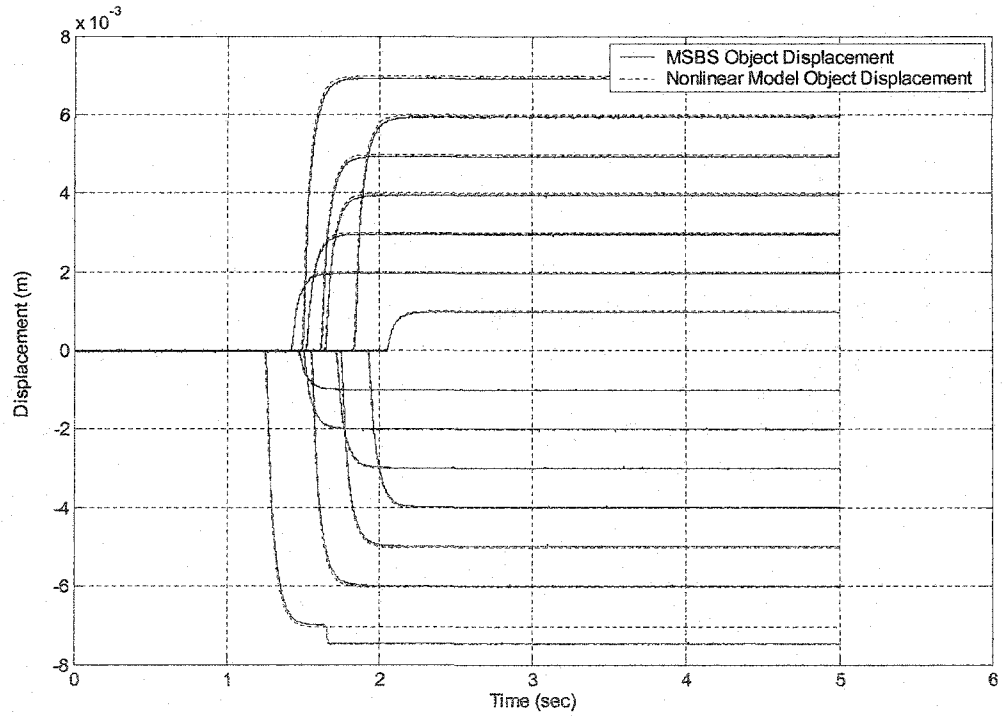


Fig. 64. Comparison of step responses of 1-DOF nonlinear model and MSBS plant over a sequence of commands ranging from -7 to +7 mm when using LTR-designed regulator.

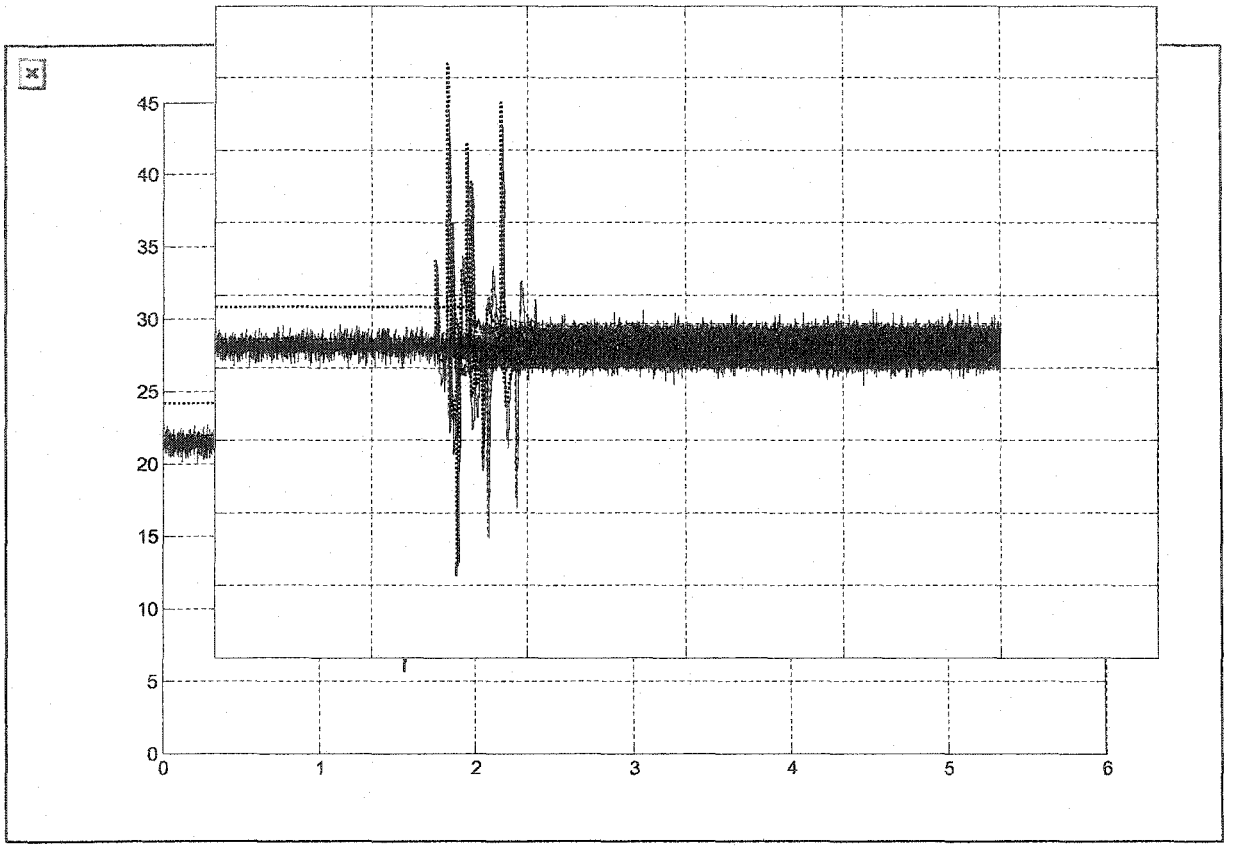


Fig. 65. Comparison of control input currents for 1-DOF nonlinear model and MSBS plant over a sequence of commands ranging from -7 to +7 mm.

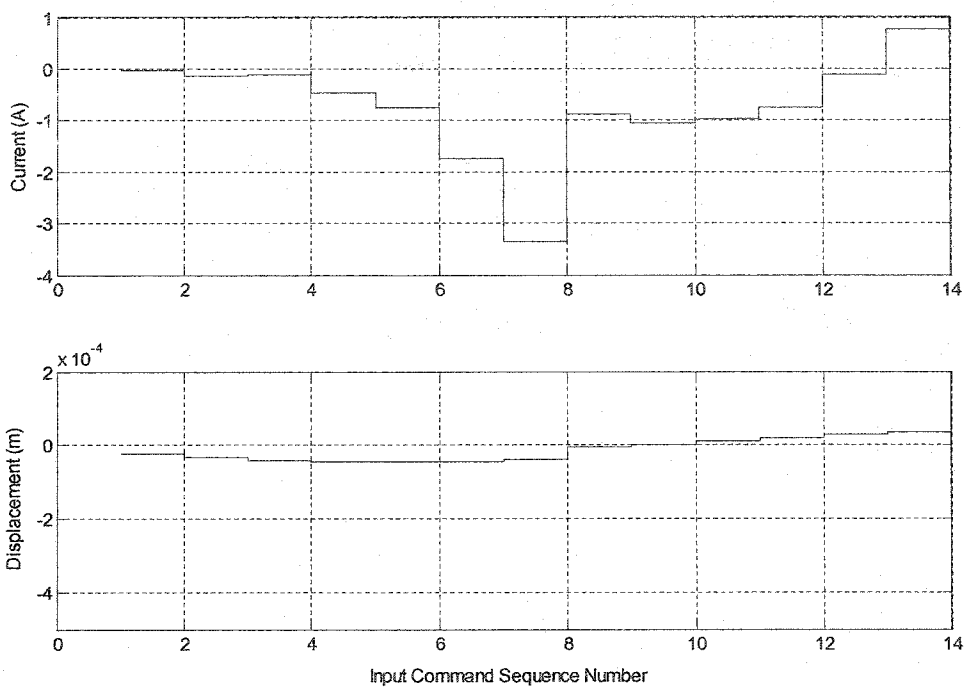


Fig. 66. Differences between MSBS plant and nonlinear model maximum control currents (upper plot) and outputs (lower plot) over a sequence of input commands: ranging from -7 to +7 mm. Calculation based on MSBS plant parameter minus the corresponding model parameter.

Comparison of Optimal Controller Design with Classical Design

The final set of results compares the stiffness and robustness against modeling uncertainty in a classical lead-lag controller and two regulators the LQR and the LQG regulator designed using the LTR method.

TABLE XXIV
STIFFNESS TEST DATA FOR LEAD-LAG CONTROLLER AND OPTIMAL REGULATORS

Set Point (mm)	Steady State Value (m)	Positive Stiffness Ratio	Negative Stiffness Ratio	Average Stiffness Ratio
-5	-0.0051	122.586	-107.631	115.108
	-0.0050	128.1954	-84.7437	106.4696
	-0.0050	328.816	-318.846	323.831
0	0	129.647	-115.136	122.392
	0	135.6316	-91.7179	113.6748
	0	347.713	-339.160	343.437
5	0.0050	136.182	-122.600	129.391
	0.0050	143.0634	-98.5562	120.8098
	0.0050	366.536	-359.330	362.933

TABLE XXV
B-FIELD PERTURBATION TEST DATA FOR LEAD-LAG CONTROLLER AND OPTIMAL REGULATORS

	Percent Overshoot		2% Settling TIme		Steady State (m)		Max. Input Deviation	
Set Point (mm)	Mean	Standard Devia-tion	Mean	Standard Devia-tion	Mean	Standard Devia-tion	Mean	Standard Devia-tion
-5-LQR	7.5829	13.2781	0.1700	0.0000	-0.0052	0.0006	2.2257	0.8870
-5-LeLa	45.2722	4.4984	0.1830	0.2506	-0.0050	0.0000	9.1569	0.2159
-5-LTR	1.1647	1.9622	0.2025	0.0005	-0.0051	0.0001	4.0940	0.3035
-2-LQR	23.9497	35.1267	0.1700	0.0000	-0.0024	0.0007	1.3434	0.9531
-2-LeLa	40.2779	11.1623	0.1951	0.2708	-0.0020	0.0000	3.5774	0.2854
-2-LTR	-0.6779	4.8133	0.2033	0.0005	-0.0020	0.0001	1.5315	0.3215
0-LQR	0.0006	0.0003	0.1700	0.0000	0.0001	0.0007	0.3466	0.5087
0-LeLa	0.0002	0.0001	3.1875	2.0122	0.0000	0.0000	0.0850	0.1792
0-LTR	0.0001	0.0000	0.2036	0.0005	-0.0000	0.0001	0.1144	0.1352
2-LQR	-5.0287	38.2850	0.1700	0.0000	0.0018	0.0007	2.0431	0
2-LeLa	43.5784	5.3802	0.2273	0.2920	0.0020	0.0000	17.0000	0
2-LTR	-0.2668	4.6384	0.2043	0.0005	0.0020	0.0001	5.4413	0
5-LQR	-0.7378	8.2290	0.1700	0.0000	0.0048	0.0004	5.1078	0
5-LeLa	36.3208	2.4604	0.0988	0.1741	0.0050	0.0000	42.5000	0
5-LTR	-0.3069	2.0279	0.2048	0.0006	0.0050	0.0001	13.6032	0

[a] LeLa denotes Lead-Lag Controller

Discussion of Results

There are several primary observations of the data to be made pertinent to the development of the MSBS controller. The first is that there was a significant degradation in performance when going from the LQR controller to the first LQG controller. This was predicted in the discussion of optimal control design earlier in the chapter, and is apparent in all three performance measurements: time response, stiffness ratio and robustness against B-field perturbations. There was greater steady state error with the LQG controller, evidenced by the fact the 2% settling times were consistently nearly as long as the test period itself. Also, the stiffness ratios were about 35% less for the LQG controller, a phenomenon also seen in the larger displacement amplitudes in the plots. Finally, the LQG controller's response to B-field perturbations was

poorer in this respect: the standard deviations in steady state values were twice as large for the LQG design as they were for the LQR controller. This reflects a definite lack of robustness to modeling uncertainty.

In contrast to the first observation, the second prominent feature of the data was the dramatic performance improvement seen in the LTR-designed over the straight iteratively-designed LQG. The LTR controller provided low steady state error and an even lower percent overshoot than the LQR design. The LTR design also produced stiffness ratios nearly three times those of the LQR design. Finally, the B-field robustness test data showed the LTR design had nearly zero mean steady-state error and the standard deviation in steady state values was 15% to 25% of that seen with the LQR design.

The contrast between the LTR-designed controller and the others was also seen in the plot of open loop responses. Note the loop response for the LTR-based system had greater gain and bandwidth than the others. That this controller demonstrated greater output disturbance rejection and damping is in keeping with classical loop shaping design techniques where large open loop gain at lower frequencies tends to allow greater disturbance rejection and larger open loop bandwidth provides faster response [28].

It should be pointed out the first LQG controller was designed strictly through iterations of synthesis and time response evaluation. The results were not very good after just a few iterations. Undoubtedly better results would have been by testing more permutations of design parameters. (However, only 10 values for each of three parameters would require 1000 evaluations to test every combination.) On the other hand, the LTR design involved no more iterations than LQG design and produced a much better controller. This points out the importance of incorporating frequency response into the design process.

A third major observation is that the optimal controller produced excellent command tracking in the actual MSBS plant. The controller was able to track positive step commands up to + 7 mm and negative commands up to – 6 mm. (This is better than the classical design tested in Chapter Five, which allowed only a plus or minus 5 mm command range.) The largest mean value of steady state error over the command range was 5%.

The fourth observation concerns the comparison the LTR-designed controller to the classical lead-lag design. The LTR design provided much lower percent overshoot than the classical lead-lag design for the perturbed plant. (This probably explains the wider command range.) The LTR controller was also stiffer by a factor of three.

Finally, the comparison of the nonlinear model and plant shows the importance of measuring the control input when validating the controller. Under the LTR method, the goal is to match the open loop response of the system as nearly as possible to the Kalman filter response by reducing the control input weight **R**. (This assumes the Kalman filter response is tuned to reflect design specifications.) But the real-world plant poses a practical limit on how "good" the LTR design can actually get, since lower values of **R** result in higher control inputs to the plant. That care should be taken to monitor the values during design is illustrated by the plot of simultaneous control inputs to the model and plant in Fig. 65. The open loop responses in the LTR design process were not very close (see Fig. 63), yet the resulting design, with **R** = 1e-9, allowed very large transients in the control input. The deviations were large enough so that the suspension coils in the plant could not meet the demands placed on them by the controller, and for input command –7 mm, the controller dropped the test object.

Conclusion

In the chapter introduction it was stated that the goals of this work were to apply optimal control principles to the MSBS, to demonstrate at least one optimal controller using MSBS hardware and software and to produce some software tools to aid in the design process. To reach these goals, optimal control theory was explored and implemented in a relatively simple controller for the 1-DOF maglev system. In the course of designing the 1-DOF controller, a design procedure was identified and various Matlab command sequences, or scripts were produced to automate some of the steps. Also, Simulink models were developed to validate the interim designs produced along the way. These scripts and models were loosely integrated into a type of design and validation software "tool" set.

Analysis of the controller tests performed during this set of experiments resulted in the following critical observations respecting MSBS controller design:

1) The optimally designed regulator and state estimator resulted in improved controller performance over the classical designs tried earlier.

2) Optimal design techniques argue for a systematic, automated methodology for MSBS controller design.

3) Including frequency response analysis in the validation can greatly expedite the design process.

4) Controller validation must include some threshold criteria for control input current.

Upon completion of the work underlying this chapter, significant progress toward meeting the overall project goal has been reached, namely understanding and applying the operational requirements, maglev dynamics, software and hardware characteristics and control theory. Two major tasks remain before the full 5-DOF controller can be developed: 1) adapting optimal control theory to the MIMO case; and 2) validating a plant model including the effects of the stainless steel suspension chamber.

CHAPTER VI

TWO-DEGREE-OF-FREEDOM EXPERIMENTS

Introduction

Before the work covered in this thesis was begun, the 5-DOF controller designs and simulations in [4] and [25] indicated the 5-DOF MSBS was possible. The experiments covered in Chapters IV and V have partially validated the overall approach to the MSBS design, including the system modeling equations, the use of the dSPACE real time controller suite with Matlab-based controller algorithms and the optimal control design method. However, another task remained to show the practicality of the 5-DOF MSBS; and that was the demonstration of a working multiple degree-of-freedom controller using actual MSBS hardware and a realistically sized model. This chapter covers the development and testing of a 2-DOF controller for the MSBS and the experiments performed to demonstrate the feasibility of controlling multiple degrees of freedom.

The primary motivation for the 2-DOF experiments stems from the main difference between SISO and MIMO systems, the existence of directivity in the MIMO plant. The chief effect of plant directivity is to produce different gains for different combinations of plant inputs, where inputs also encompass noise signals present throughout the plant. In turn this creates a more complicated set of conditions for system stability and robust performance, greatly increasing the controller's susceptibility to modeling errors. Validation of the modeling assumptions in the presence of plant directivity was seen as a prerequisite to proceeding to the full 5-DOF controller.

The 2-DOF configuration was seen as the logical choice for validation because it required only two suspension coils yet created a three dimensional electromagnetic environment in which to test the controller. In addition, this set of would use a large volume permanent magnet in the suspended object and maintained an equilibrium point close to the actual distances intended for the final MSBS, factors important to validating the modeling assumptions described above.

Another motivation for the 2-DOF experiments was to develop the controller design and validation programs to handle the more complex MIMO case. This would entail new Simulink modeling structures to handle more sensor signals and amplifiers. In addition, the MIMO case would require additional functionality in the operator GUI pertaining to system monitoring, control and safety.

The experiments covered in this chapter were designed to further the MSBS development in three ways: 1) by validating the modeling equations for the multiple degree of freedom MSBS; 2) by

further developing the controller synthesis and validation tools for the MIMO case; and 3) by identifying practical engineering considerations pertaining to an implementation with multiple degrees of freedom.

The chapter is organized as follows. First, the rationale for the 2-DOF experiments is developed and several concerns particular to the 2-DOF MSBS configuration are explained. Next, there is a discussion of the experimental setup, including the equipment configuration, the structure of the Simulink models and a special modification made to the DC power supply. Then the experimental procedure is described, including an explanation of the semi-automated process that was developed to expedite the controller design. Following that the results of the experiments are presented, and finally the chapter concludes with a summary statement concerning the success of the experiments.

Design Considerations for 2-DOF Experiment

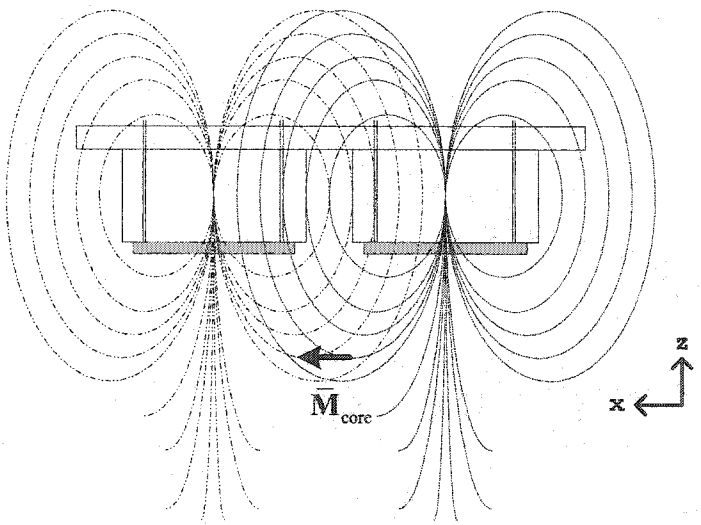
This section provides background for the 2-DOF experiment's setup and procedure. In particular, this section elaborates on the rationale and benefit of conducting a 2-DOF experiment as well as explains how the full 5-DOF linear model is scaled for the 2-DOF case. In addition, this section discusses several changes in the way optimal control theory was applied to the 2-DOF (MIMO) case over the way it was applied in the 1-DOF case. The section concludes with a discussion of a modification that was made to the Clinton DC power supply.

Purpose and Design of the 2-DOF Experiment

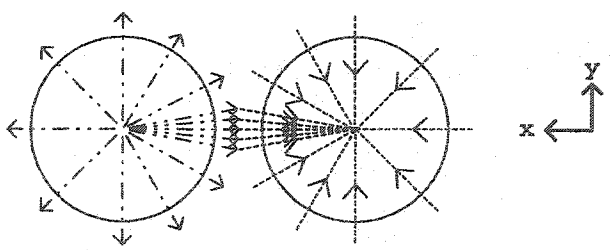
Two modeling assumptions in particular are of concern in this set of experiments. Set forth in Chapter II, these assumptions concern the magnetic fields in the MSBS and are major sources of uncertainty in the controller design. The first is the assumption the values of magnetic flux density and flux density gradients at the suspension equilibrium point, i.e., the origin of the inertial reference system, are known. For the MSBS, these values are obtained from a PC-based magnetic field modeling program, OPERA ^{TM} , which in turn bases its calculations on certain approximations of the physical parameters on the system being modeled. The second assumption is that the magnetic fields and gradients are uniform throughout the volume of the suspended permanent magnet. This assumption, predicated on the small dimensions of the suspended magnet relative to the suspension coils and the levitation distances, leads to an approximation of torques and forces in the modeling equations [10]. However, there was no specification as to how small or how far away from the suspension coils the suspended magnet has to be for this

assumption to hold. Consequently, the robust performance of a linear controller based on these assumptions needed to be assessed before proceeding to the full 5-DOF MSBS.

To best validate the MSBS modeling approach, an experiment was needed to implement the dynamics of 5-DOF MSBS as completely as possible using actual MSBS components. But an experiment based on the full 5-DOF system would have been impossible to set up and build within an acceptable time frame, given that the suspension coils had not been mounted around the suspension chamber. Fortunately, when only two suspension coils are operated in the vicinity of the test object, torques and forces are produced in three dimensions. This can be seen by considering the field lines illustrated in Fig. 67 in the context of equations (23) through (31). Thus it is possible to use a relatively easily built 2-coil configuration and still create much of the same electromagnetic interaction between the suspension coils and test object that would be seen in a larger system.



(a) Side view of coils



(b) Underside of coils

Fig. 67. Magnetic flux lines for the 2-DOF configuration.

This can be seen by examining the magnetic flux density lines for the two-coil configuration in Fig. 67. The two views show that in the region around the equilibrium point, denoted by the magnetization vector **M**, there are components of flux along all axes of the inertial reference system. Thus, in contrast to the 1-DOF configuration, where B_z is the only significant component of magnetic flux and F_z is the only significant force component, the set of forces and torques better represent the full 5-DOF environment.

However, it should be noted the 2-coil configuration does not completely model the full MSBS. One reason is the cancellation of the y-component of flux density B_y. Looking at the configuration from the underside, as illustrated in Fig. 67(b), the flux lines contain components in both the x and y directions. To see how the cancellation occurs, consider a perpendicular bisector constructed on a line segment connecting centers of the two coils. Because of the symmetric flux line pattern along that bisector, and the opposite flux orientation produced by each magnet, it can be seen the y-components would cancel one another along the perpendicular bisector, which represents the transverse axis of the suspended magnetic core. Furthermore, over any region symmetric around the line segment connecting the two coil centers, i.e., the longitudinal axis of the permanent magnet core, the net B_y is also zero. The effect of this is to remove B_y and B_{yy} from the torque and force equations. Nevertheless, the system will still produce some force and torque in five degrees of freedom.

To create the tri-axial magnetic field, the 2-DOF experiment was designed using a pair of MSBS suspension coils mounted in tandem on a structural fiberglass backplane. (Four of these two-coil assemblies had been built in anticipation of building the full 5-DOF configuration, pictured in schematic form in Fig. 2.) The two-coil assembly was placed on a wooden frame so that the coil axes were oriented vertically and the test object containing the permanent magnet core could be levitated below the coils. In this way, two actual MSBS coils could be used to levitate and control the height and pitch of a test object approximating its eventual operational size.

Obviously, one important element lacking in this simulation of MSBS operations was the stainless steel suspension chamber. Chapter II pointed out the presence of eddy currents in the chamber walls would probably affect the dynamics of the control system and these effects would have to be modeled. However, to characterize the effect of the chamber on the control system would have involved substantial effort to mount the coils and sensor system. It was intended to first develop a working 2-DOF controller and a usable controller design method, and afterward refine the controller as needed to compensate for the staineless steel pipe.

The 2-DOF Mathematical Models

Chapter II states that the matrix element values in the linear state space model are obtained by taking the partial derivatives of the torque and force equations with respect to each of the state variables and the suspension coil currents. The terms of the partial derivatives are evaluated for the system equilibrium condition and contain constants associated with the equilibrium coil currents, the structure of the suspended object and the permanent magnet it contains, and the magnetic flux densities and their gradients produced by the suspension coils at the specific equilibrium point. The following paragraphs show how the equations in Chapter II are adapted to produce the 2-DOF linear model.

Recall the method for determining the matrix element values in the linear model requires the equilibrium current to be determined first. This is done by first evaluating the input matrix **B** for the system at equilibrium.

In the 2-DOF system, the input matrix **B** is given by

$$\mathbf{B} = \frac{vM_x}{I_{\max}} \begin{bmatrix} -b_{1z}/I_c & -b_{2z}/I_c \\ b_{1y}/I_c & b_{2y}/I_c \\ 0 & 0 \\ 0 & 0 \\ b_{1xx}/m & b_{2xx}/m \\ b_{1xy}/m & b_{2xy}/m \\ b_{1xz}/m & b_{2xz}/m \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix},$$

where the values of magnetic flux densities and gradients, b_{1y}, b_{2y}, b_{1zx} etc., are those produced when I_{max} flows in suspension coils one and two. At equilibrium, the angular and translational displacements and velocities of the test object are zero, and the only forces acting on the object are gravity and the net vertical magnetic force produced by both coils. Of course these two forces must be equal and opposite. In light of these conditions, the input matrix **B** leads to a set of five linear equations

$$\frac{vM_x}{I_{\max}}\begin{bmatrix} -b_{1z}/I_c & -b_{2z}/I_c \\ b_{1y}/I_c & b_{2y}/I_c \\ b_{1xx}/m & b_{2xx}/m \\ b_{1xy}/m & b_{2xy}/m \\ b_{1xz}/m & b_{2xz}/m \end{bmatrix}\begin{bmatrix} I_{1eq} \\ I_{2eq} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ g \end{bmatrix},$$

where the numeric subscripts indicate the suspension coil numbers. This system of equations is solved for I_{1eq} and I_{2eq}, the equilibrium currents in coils one and two.

With the equilibrium coil current values, the values of the elements in the state space matrix A are then calculated. From equation (65) in Chapter II, the instantaneous magnetic flux density is estimated by the ratio of instantaneous coil current to the current at which the maximum flux density is measured. (This also applies to the flux density gradients.) Further, the net flux density (or gradient) at the equilibrium point is equal to the algebraic sum of the components produced by each magnet. The B-field components at equilibrium are calculated according to the following example:

$$-B_x = -\left[\frac{\left(I_{1eq}\cdot b_{1x} + I_{2eq}\cdot b_{2x}\right)}{I_{\max}}\right]$$

As a result, the numeric values for the 2-DOF **A** matrix given in symbolic form in equations (68) and (69) are determined using equilibrium B-field values. Using the physical parameters for this experimental setup, discussed in the following section, the following **A** matrix results:

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & -73.515 & 0 & 0 & 0 & 0 & -486.58 & 0 & 0 \\ 0 & 0 & 0 & -73.515 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -9.81 & 0 & 0 & 0 & 0 & -25.210 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -20.863 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 67.201 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

Recall that the input matrix **B** is determined by taking the partial derivatives of the force and equations with respect to the current. Consequently, the symbolic terms in **B**, used in (70) and (71) in Chapter II, contain the maximum B-field values. The remainder of the 2-DOF state space model is shown below.

$$\mathbf{B} = \begin{bmatrix} 0.4489 & 0.4489 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ -0.0616 & -0.0616 \\ 0 & 0 \\ 0.1142 & 0.1142 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

$$\mathbf{C} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{x}^{\mathrm{T}} = \begin{bmatrix} \dot{\theta}_y & \dot{\theta}_z & \theta_y & \theta_z & \dot{x} & \dot{y} & \dot{z} & x & y & z \end{bmatrix}$$

In [4], the 5-DOF version of this system was shown to be both state controllable and observable. However, with only two inputs and two outputs, and the magnetic field pattern being what it was, this was no longer the case for the 2-DOF configuration. A check of the rank of the controllability and observability matrices—using the Matlab commands *ctrb.m* and *obsv.m*—revealed the system was neither controllable nor observable.

To identify the uncontrollable and unobservable states, the state space model was transformed to diagonal form using a transformation matrix equal to the inverse of the eigenvector matrix. The diagonal of the resulting **A** matrix shown below contains the system eigenvalues (modes) and the elements in the resulting **B** and **C** matrices correspond to those modes by row in the case of **B** and by column in the case of **C**. All the zeros in **B** and **C** indicate the corresponding mode is uncontrollable (unobservable) from that input (output).

The diagonal form also reveals whether the system is stabilizable or detectable. Zero entries in **B** or **C** whose corresponding eigenvalues, or modes, in **A** are unstable show that the system is not stabilizable or detectable. The matrices below show the 2-DOF system was borderline stabilizable and detectable. This is because the uncontrollable eigenvalues lie on the imaginary

axis, making system stability highly susceptible to modeling errors or system perturbations. In fact this condition caused Matlab to return the error "undetectable system" when the commands *lqe.m* and *dlqe.m* were used on the full ten-state system with two inputs and outputs.

$$\mathbf{A}=\begin{bmatrix} +j11.07 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -j11.07 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -4.88 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4.88 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & +j8.57 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -j8.57 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 8.2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -8.2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & +j4.57 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -j4.57 \end{bmatrix}$$

$$\mathbf{B}=\begin{bmatrix} -.0474 & -.0474 \\ -.0474 & -.0474 \\ -.185 & -.185 \\ -.185 & -.185 \\ 0 & 0 \\ 0 & 0 \\ -.0575 & .0575 \\ -.0575 & .0575 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

$$\mathbf{C}=\begin{bmatrix} +j.0895 & -j.0895 & 1.97 & -1.97 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & .121 & -.121 & 0 & 0 \end{bmatrix}$$

Table XXVI presents a modal analysis of the 2-DOF system, which shows the eigenvalues and corresponding eigenvectors and notes which are controllable and observable. The top row contains the eigenvalues, or modes. The column underneath each contains the eigenvectors.

TABLE XXVI
MODAL ANALYSIS OF 2-DOF PLANT

Mode 1		Mode 2		Mode 3		Mode 4		Mode 5	
j 11.07	$-j$ 11.07	-4.8813	4.8813	j 8.574	$-j$ 8.574	8.1976	-8.1976	j 4.568	$-j$ 4.568
Controllable:u_1, u_2 Observable: y_1		Controllable: u_1, u_2 Observable: y_1		Uncontrollable Unobservable		Controllable: u_1, u_2 Observable: y_2		Uncontrollable Unobservable	
-.9909	-.9909	-.9606	-.9606	0	0	0	0	0	0
0	0	0	0	.9933	.9933	0	0	0	0
j.0895	$-j$ .0895	.1968	-.1968	0	0	0	0	0	0
0	0	0	0	$-j$ .1158	j .1158	0	0	0	0
-.0999	-.0999	.1922	.1922	0	0	0	0	0	0
0	0	0	0	0	0	0		.9769	.9769
0	0	0	0	0	0	.9926	.9926	0	0
j .0090	$-j$ .0090	-.0394	.0394	0	0	0	0	0	0
0	0	0	0	0	0	0	0	$-j$ .2139	j .2139
0	0	0	0	0	0	.1211	-.1211	0	0

To work around the fact that for all practical purposes the 2-DOF system was neither stabilizable nor detectable, it was decided to extract from the main model those states known to be controllable for the 2-DOF case, pitch angle and vertical displacement. The states were extracted using the Matlab commands:

```
[Am,Bm,Cm,Dm]=ssselect(A,B,C,D,[1 2],[1 2],[1 3 7 10]);
minsys=ss(Am,Bm,Cm,Dm);
```

The resulting controllable/observable system, which served as the basis for the controller design, is given by

$$\mathbf{A} = \begin{bmatrix} 0 & -73.515 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 67.201 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{111}$$

$$\mathbf{B} = \begin{bmatrix} 0.4489 & 0.4489 \\ 0 & 0 \\ -0.1142 & 0.1142 \\ 0 & 0 \end{bmatrix} \tag{112}$$

$$\mathbf{C} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{113}$$

$$\mathbf{x}^{\mathrm{T}} = \begin{bmatrix} \dot{\theta}_y & \theta_y & \dot{z} & z \end{bmatrix}. \tag{114}$$

Besides the linear model used to design the controller, a nonlinear model, capturing the dynamics of the 5-DOF system, was used to test the designs. The model used in these experiments was produced to conduct 5-DOF controller simulations in [4]. The nonlinear model is essentially a term-by-term implementation in Simulink of the torque and force equations developed in Chapter II.

Application of Optimal Control to the 2-DOF System

This remainder of this section explains the how the additional complexity of the multiple-degree-of-freedom configuration changes the use of optimal control theory as seen in the previous chapter. Of course, the underlying concepts of the regulator control law, state estimation, state observability and controllability discussed in Chapter VI still applied, as did the techniques of cost minimization and the roles of the design parameters, **Q**, **R**, etc. (It was MIMO controller design that motivated the application of optimal control in the first place.) However, some of the mechanics of the automated design process had to be reworked to accommodate MIMO design.

One adaptation to earlier Matlab scripts was to employ both maximum and minimum singular values as measures of the frequency responses of the Kalman estimator and plant. As mentioned at the beginning of the chapter, the directivity of a MIMO system is manifest in what are called the system's principal gains, or singular values. The principal gains in the various system transfer functions (e.g., the open loop, **L**, sensitivity, **S**, and complementary sensitivity, **T**) give rise to corresponding sets of principal input and output directions. For example, for an equal number of inputs and outputs, the input direction set $\{u_1, u_2, \ldots u_n\}$ corresponds to the principle gains $\{\sigma_1, \sigma_2, \ldots \sigma_n\}$, resulting in the output direction set $\{y_1, y_2, \ldots y_n\}$. Because the direction of an input vector is determined by its individual element values, the gains of **L**, **S** and **T** are dependent on the combinations of elements in their respective input vectors, including the stochastic noise signals throughout the system.

The result of system directivity is to require stability and performance measures to be made in terms of both maximum and minimum system gains. For example, to determine the minimum bandwidth of an open loop transfer function, $\mathbf{L} = \mathbf{GK}$, for all possible inputs, the performance measure is the zero-crossover of the smallest principal gain. On the other hand, to judge the minimum amount of output disturbance rejection provided by the system, the –3 dB point of the largest principal gain of $\mathbf{S} = (\mathbf{I} + \mathbf{L})^{-1}$ is considered. Consequently, the Matlab scripts had to be revised to calculate two values over a range of frequencies for at least two system transfer functions, which required quite a few more programming steps than the single open loop gain analysis used in the SISO analysis. Additionally, the Matlab scripts had to be adapted to use vectors instead of simple variables to handle the two-dimensional signals versus scalars.

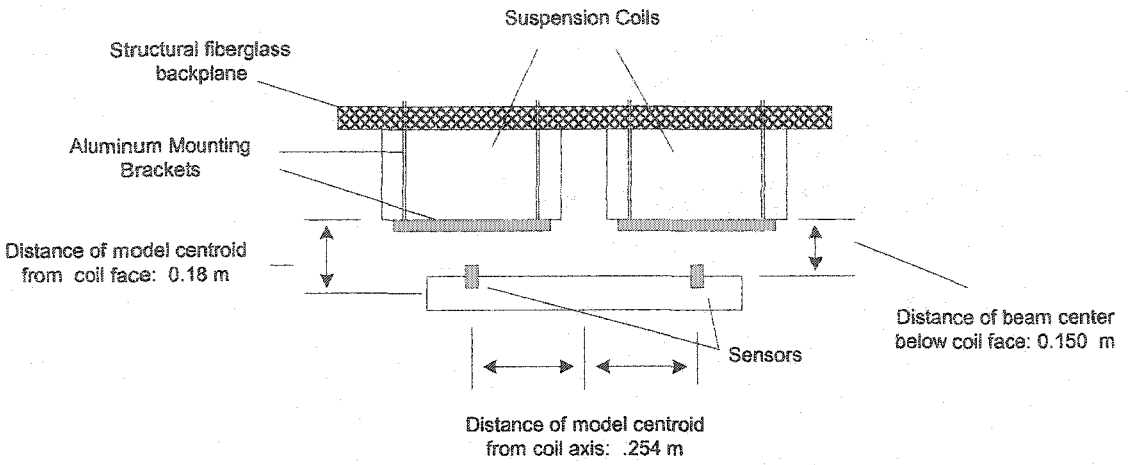
Experimental Setup

This section describes the physical configuration of the 2-DOF experiment, the Simulink models that were used and a special modification to the high-power DC power supply. The basic system architecture did not change from earlier experiments, although the hardware suite expanded and the control system modeling became more complex. Most of the experiments were done outsided the stainless steel suspension chamber, but just at the completion of this paper's writing, a simple 2-DOF levitation was conducted insided the suspension chamber.

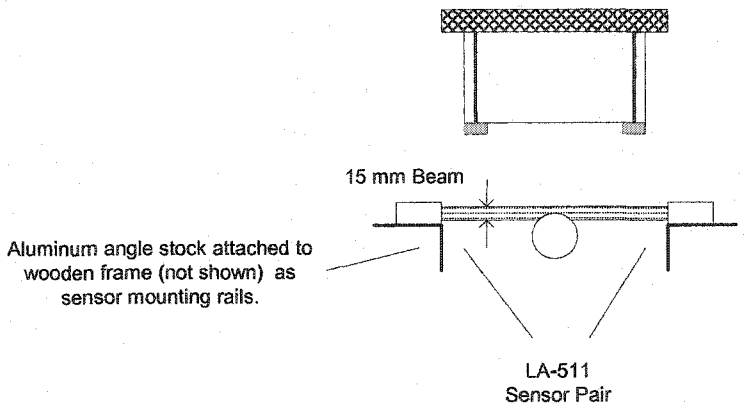
Hardware and Software Configuration

The 2-DOF experiments used the general MSBS configuration depicted in Fig. 18 in Chapter IV but involved more components than the 1-DOF experiments. As indicated in the previous section, the setup contained two vertically mounted coils, two sensor pairs and two amplifiers. Also, the test object was much larger than for earlier experiments. Fig. 68 illustrates the arrangement of the 2-DOF plant (minus the amplifiers) used for most of these experiments. For simplicity the wooden framework built to support the coils and the aluminum mounting rails for the sensors are not shown in this figure.

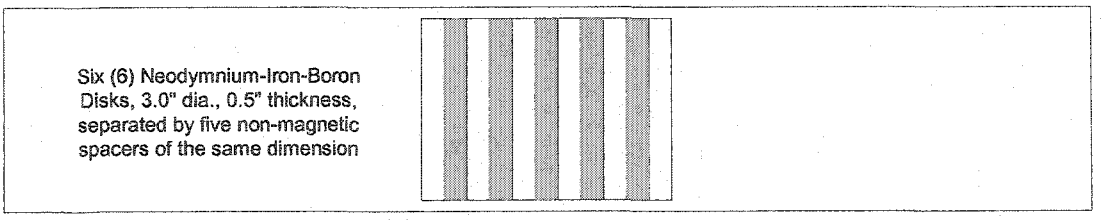
The side view of the 2-DOF setup in Fig. 68(a) illustrates the relationships between the major components, including the critical distances between them for the system at equilibrium. Note the equilibrium suspension distance is about 180 mm (7 inches), which corresponds to a likely position in the stainless steel suspension chamber when the MSBS will be in actual operation. Also note the actual equilibrium point is midway between the suspension coil axes. The distances shown in the figure are those used with OPERA ™ to calculate the B-field values. Fig. 68(b) contains an end view of the setup with the test object at its equilibrium point.



(a) Side view of 2-DOF test set up. Wooden support frame and sensor mounting rails not shown.



(b) End view of 2-DOF test set up. Wooden supprt frame not shown.



Cylindrical test model: 76.2 cm (30 in.) section of PVC conduit , 8.89 cm (3.5 in.) O.D.

(c) Test model showing permanent magnet (not to scale).

Fig. 68. Schematic drawings illustrating set up of 2-DOF plant.

Fig. 68(c) provides a more detailed view of the test object. An outer shell, whose size and shape were intended to roughly simulate an actual operational model in the MSBS, was constructed from a section of PVC pipe approximately 762 mm (30 inches) long and 89 mm (3.5 inches) in

outer diameter. The permanent magnet core consisted of a set of six neodymnium-iron-boron disks, each nominally 12 mm (0.5 inches) thick by 122 mm (3.0 inches) in diameter, assembled with identically sized, non magnetic spacers between the disks. The magnet assembly was mounted in the center of the pipe so that the assembly and shell were coaxial and the magnetization vectors were oriented to the left as shown in Fig. 68(a). The fact that the core assembly and the shell were both cylindrical and shared a common longitudinal axis greatly simplified the calculation of the moments of inertia, I_x, I_y and I_z. Specifically, the moments of inertia for the whole object could be determined piecewise by individually calculating the moments for the core and shell, each with different lengths and densities, and then adding the components. For these calculations, the magnetic disks and the spacers were treated as a single unit with respect to the mass and size of the core.

Table XXVII summarizes the values of the physical parameters in the 2-DOF plant, minus the values of the magnetic flux densities and gradients at the equilibrium point. These were used to derive the state space model. For comparison, the corresponding values used for the 5-DOF simulation in [4] are also provided.

TABLE XXVII
PARAMETERS USED IN 2-DOF SYSTEM MODELING

Parameter	Values for 2-DOF Experiment	Values for 5-DOF Simulation
Dimensions of Suspended Object (length x dia.)	760 mm x 89 mm	Not available
Mass of PVC Shell	1.64 kg	Not available
Dimensions of permanent magnet core [a]	140 mm x 76.2 mm	Not available
Mass of permanent magnet core [a]	2.723 kg	Not available
Total Mass of Suspended Object, m	4.383 kg	20.67 kg
Volume of Permanent Magnet, v	3.475e-4 m^3	2.65e-3 m^3
Magnetization of Permanent Magnet, M_x	9.7085e 5 A/m	9.5493e 5 A/m
Moment of Inertia (about transverse axes), I_y, I_z [b]	0.0879 kg-m^2	0.837 kg-m^2
Moment of Inertia (about longitudinal axis), I_x [b]	0.0085 kg-m^2	0.0116 kg-m^2
Equilibrium Current (all coils), I_1, I_2, I_3, etc…	-42.9590 A 42.9590 A	102 A 102 A -102 A -102 A 0 A
Equilibrium Distance below coils, z	-0.178 m	Not available
Equilibrium offset from coil axes	0.254 m	Not available
Flux Density at Equilibrium B_{ij} [b]	B_{xz} = 0.1268 T/m	B_{yz} = 0.0106 T/m

[a] For purpose of calculating moments of inertia only, core comprised both magnetic disks and spacers.

[b] Note coordinate axis labeling in this thesis, x-y-z, corresponds to z-x-y in [4].

Table XXVIII contains the maximum values of the magnetic flux densities and and their gradients at the origin of the inertial reference system, or equilibrium point. These values were calculated using the finite element analysis program OPERA ™ assuming a coil current of 100 amps.

TABLE XXVIII
MAXIMUM MAGNETIC FLUX AND GRADIENT VALUES AT EQUILIBRIUM POINT

Field or Gradient	Coil No. 1 [a] (T, T/m, T/m^2)	Coil No. 2 (T, T/m, T/m^2)	Field or Gradient	Coil No. 1 (T, T/m, T/m^2)	Coil No. 2 (T, T/m, T/m^2)
B_x	-0.0223	0.0223	B_{xxx}	0.3793	-0.3793
B_y	0.0	0.0	B_{xyx}	0.0	0.0
B_z	-0.017	-0.017	B_{xzx}	-1.0782	-1.0782
B_{xx}	-0.796	-0.796	B_{xyy}	0.3139	-0.3139
B_{xy}	0.0	0.0	B_{xyz}	0.0	0.0
B_{xz}	-0.1476	0.1476	B_{xzz}	-1.0111	1.0111
B_{yy}	0.0837	0.0837	B_{yyy}	0.0	0.0
B_{yz}	0.0	0.0	B_{yyz}	0.0	0.0
B_{zz}	-0.1304	-0.1304	B_{yzz}	0.0	0.0
			B_{zzz}	1.0782	1.0782

[a] Coil No. 1 is the one in the positive x direction relative to the other, i.e. the one in the direction toward which the magnetization vector of the suspended magnet is pointed.

As stated above, the stainless steel suspension chamber was not used for the 2-DOF experiments except for one brief experiment at the close of the research period. For this, a second pair of suspension coils, mounted on a fiberlass backplane just as in the setup in Fig. 68, were placed on top of the suspension chamber. The sensors were arranged in the same basic configuration as in Fig. 68 on a temporary wooden frame inside the chamber and the same test object was used. The same controller setup used but the levitation gap was greater than for exterior 2-DOF experiments.

Power Supply Modifications

Earlier experiments, employing only a single MSBS suspension coil with an equilibrium current of only 22 amps, required only a small, 40-ampere DC power supply to operate the Copley power amplifier. But for the 2-DOF experiments, using two coils with equilibrium currents on the order of plus or minus 42 amps, the large 150-volt, 500-ampere Clinton DC power supply was required.

Unfortunately, when the 2-DOF experiments were begun, the Clinton power supply posed a significant risk of overload due to two features of the MSBS hardware installation. First, the input stage of the Copley amplifier contains a 26 mF filter capacitor, whose positive side is connected to the DC input line, thus presenting a large capacitive load. Secondly, the MSBS configuration had six Copley amplifiers with inputs ganged together on large buss bars, effectively placing all six capacitors in parallel. Consequently, at system turn on, the six amplifiers could produce a very large current demand on the power supply. The lower output voltage of the 40 amp power supply limited the current draw and the smaller power supply did not experience excessive surge currents when connected to the amplifier buss bar. But at 150 volts the Clinton power supply did not limit the current and in some earlier experiments, surge currents had burned out one of the power supply rectifiers.

To reduce the risk of overload, a phased turn-on circuit was added to the Clinton's control panel. This so-called "soft-start" circuit contained a separate transformer and rectifier that "pre-charged" the capacitors in the amplifier bank while holding the main supply out of the circuit. At the end of a user-specified period, the Clinton supply was switched in to power the amplifiers and the auxiliary circuit dropped out. The only adaptation to the original turn-on procedure was that the user would have to press and hold the Clinton start button until the soft-start circuit timed out, a period of about ten seconds. Appendix E contains a schematic of the soft-start circuit.

Models for Simulation and Real Time Control

This set of experiments used two slightly different controller configurations. Both employed a regulator with state estimation using a Kalman estimator, but the second one also incorporated integral control with a separate output feedback loop. In addition, both 2-DOF controllers were designed in discrete time based on a zero-order hold data sampling. As in the 1-DOF case, this set of experiments employed both validation models for the controller design process and real time controller models for use with dSPACE. The following paragraphs highlight the new functionality added to these models.

The controller validation models contained a nonlinear model of the 6-DOF plant, based on the one developed in [4]. However, for these experiments, that original nonlinear model had to be modified in three respects. First, variable naming was altered to recognize the change in coordinate axis labeling from the convention used in [4]. Secondly, the input and output vectors in the model were changed to reflect two inputs and outputs. Lastly, the mechanisms for introducing output disturbance forces and B-field perturbations (for robustness testing) were redesigned due to a change in the Matlab commands available in the current software suite.

These modifications are denoted by the use of drop down shadows in the overall block diagram shown in Fig. 69.

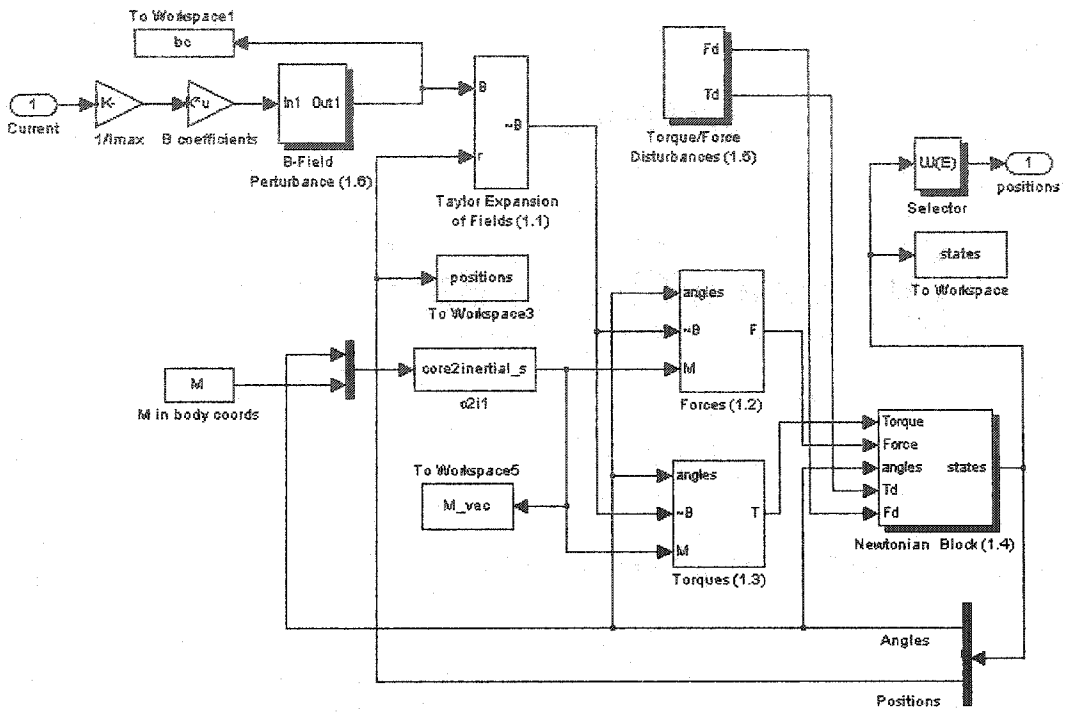


Fig. 69. Top level view of the 6-DOF nonlinear model modified for 2-DOF.

As noted earlier in the chapter, the nonlinear model was built as an explicit implementation of the 6-DOF equations of motion. For example, note the block labeled "Taylor Expansion of Fields" in Fig. 69. This subsystem contains Taylor series expansions that dynamically update the values of the magnetic fields and gradients according to the instantaneous displacement of the core from equilibrium, just as shown in

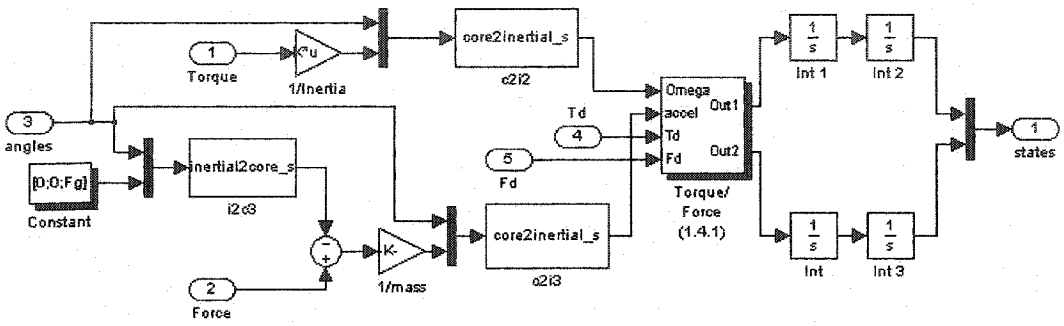
$$\tilde{\overline{\mathbf{B}}} = \overline{\mathbf{B}} + \overline{\mathbf{r}}_x \frac{\partial B}{\partial x} + \overline{\mathbf{r}}_y \frac{\partial B}{\partial y} + \overline{\mathbf{r}}_z \frac{\partial B}{\partial z} + \frac{1}{2}\overline{\mathbf{r}}_x \frac{\partial^2 B}{\partial x^2} + \frac{1}{2}\overline{\mathbf{r}}_y \frac{\partial^2 B}{\partial y^2} + \frac{1}{2}\overline{\mathbf{r}}_z \frac{\partial^2 B}{\partial z^2}. \tag{15}$$

The block "c2i1" in Fig. 69 invokes the Matlab s-function, "core2inertial_s", which converts the magnetization vector, **M**, from body to inertial coordinates for use in the force and torque calculations. Similarly to the Taylor expansion block, this function uses the instantaneous angular displacement of the core to compute the instantaneous value of the transformation matrix.

One other view of the nonlinear model is presented in Fig. 70 to indicate where the 2-DOF adaptations were made. This is the subsystem that implements the equations of motion, equations (50) through (61). It should be emphasized that although only four states are used to design the 2-DOF controller, as shown in (111) through (114), the nonlinear model simulates the full 6-DOF system, i.e., it contains all twelve states shown in the vector

$$\mathbf{X}^T = [\Omega_{\bar{x}} \quad \Omega_{\bar{y}} \quad \Omega_{\bar{z}} \quad \theta_x \quad \theta_y \quad \theta_z \quad V_{\bar{x}} \quad V_{\bar{y}} \quad V_{\bar{z}} \quad x \quad y \quad z]. \tag{63}$$

Thus the controller validations presented a rigorous test for the controller design in that the uncontrollable states were free to run in an open loop.



Drop shadow boxes indicate modifications or additions to orginal model.

Fig. 70. Subsystem of modified nonlinear model containing torque and force calculations.

Because of the overall similarity between the two controller designs in these experiments, only the models containing integral control are described here. Fig. 71 contains a block diagram of the system containing integral control. Note the zero-order hold block, which performs a sample and hold operation on the continuous time plant outputs at a specified sample rate, in this case 1 kHz. Also note a major difference from the 1-DOF systems in that this controller does not combine the Kalman estimator and the LSVF into a single LQG regulator block.

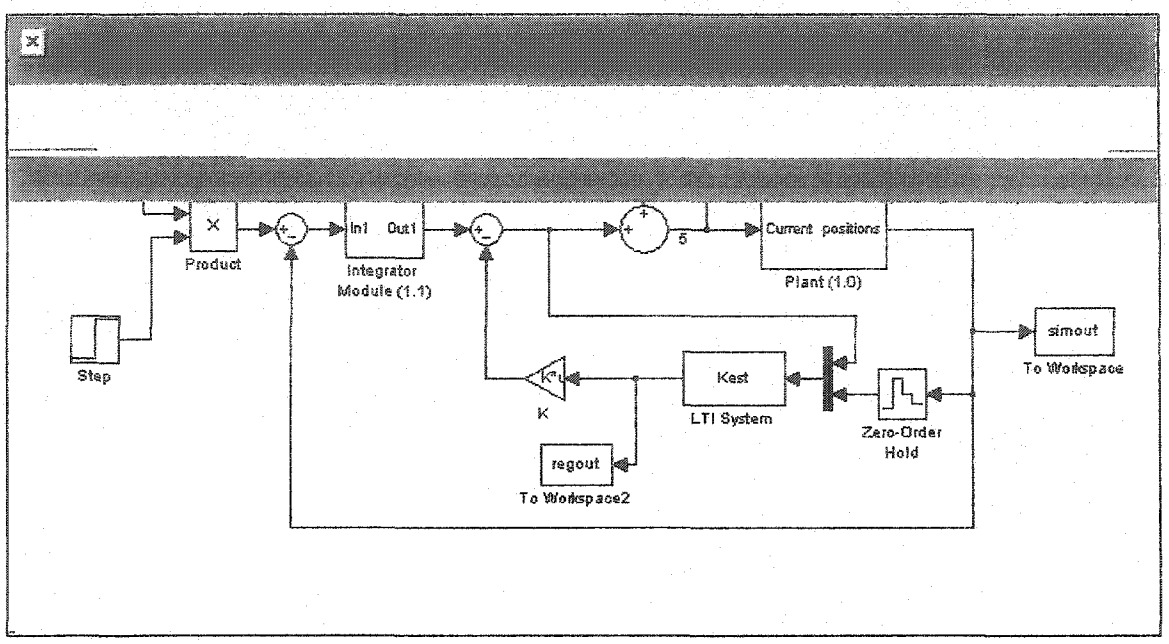


Fig. 71 Controller validation model with LQR, Kalman estimator and integral control.

The motivation for integral control was touched on briefly in Chapter IV where it was seen that adding compensator with poles near zero effectively changed the system to Type I and thus force the steady-state error to zero. This same idea was applied to these experiments by creating an output feedback loop, taking the difference between the reference input and output to produce an error signal and then integrating error signal. The integrated signal is summed with the regular states of the plant to form a new input vector. This is referred to as state augmentation because the integrals of the errors are effectively additional states added to the original state vector. The integrator module is depicted in Fig. 72.

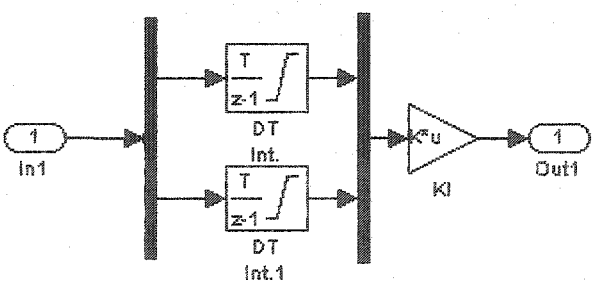


Fig. 72. Integrator subsystem in 2-DOF controller validation model.

Note the blocks K and KI in Figs 71 and 72, respectively. These are the normal (non-integral) LSVF gain matrix and the gains associated with the additional states, respectively. To derive these matrices, the following augmented state-space model is formulated and a modified control law stated. To the normal system equation

$$\mathbf{x}(k+1) = \mathbf{\Phi x}(k) + \mathbf{\Gamma u}(k)$$

is added the equation describing the integral of the error vector

$$\mathbf{x}_I(k+1) = \mathbf{x}_I(k) + e(k) = \mathbf{x}_I(k) + r - \mathbf{H x}(k).$$

These two equations together form the augmented state-space model

$$\begin{bmatrix} \mathbf{x}_I(k+1) \\ \mathbf{x}(k+1) \end{bmatrix} = \begin{bmatrix} \mathbf{I} & -\mathbf{H} \\ 0 & \mathbf{\Phi} \end{bmatrix} \begin{bmatrix} \mathbf{x}_I(k) \\ \mathbf{x}(k) \end{bmatrix} + \begin{bmatrix} 0 \\ \mathbf{\Gamma} \end{bmatrix} u(k) - \begin{bmatrix} \mathbf{I} \\ 0 \end{bmatrix} r(k),$$

which, when LSVF is applied, leads to the control law

$$u(k) = -\begin{bmatrix} \mathbf{K}_I & \mathbf{K} \end{bmatrix} \begin{bmatrix} \mathbf{x}_I(k) \\ \mathbf{x}(k) \end{bmatrix}.$$

To obtain the optimal **K** matrix, the basic LQR design procedure is followed except that 1) the state weighting matrix, **Q**, is augmented to include weights for the integrators; and 2) the augmented plant model is used rather than the original. A sample set of commands is presented to illustrate.

```
% Set LQR tuning parameters
R=[.01;.01];
Qint=[10 10];                  % portion of matrix diagonal for integrators
Qplant=[50 50 50 50];                                % basic matrix diagonal

% Create augmented plant to obtain integral control
phi_aug=[eye(2) H;zeros(4,2) Am];                    % Augmented state matrix
gam_aug=[zeros(2);gam];                              % Augmented input matrix
newdiag=[Qint Qplant];                    % Create augmented matrix diagonal
Q=diag(newdiag);                               % Produce augmented Q-matrix
[Klqr]=dlqr(phip, gamp,Q, R);                          % Obtain combined KI/K
KI=Klqr(:,1:2);                  % Assign first two columns to integrators
K=Klqr(:,3:6);                          % Assign remaining columns to LSVF
```

Fig. 73. Command set to augment plant with integrators and determine LSVF matrices.

Note that a single matrix Klqr is derived in the command set above and then subdivided to obtain the variables for the Simulink models. The dimensions of $\mathbf{K}_I$ are determined by the number of plant inputs and the number of outputs fed back.

Simulink models were created to produce the real time controllers in dSPACE. Just as with the 1-DOF experiments, these models contained the controller itself, the amplifier control logic and signal processing needed to run the plant. The model used for integral control is shown in Fig. 74. Several new features are explained in the following paragraphs.

During one of the tests, it was observed the addition of separate integrators produced the same effects seen in the 1-DOF experiment where the compensator contained poles close to the origin. The error voltage would build up to high levels as the system was being placed in operation. Thus a second output from the turn-on threshold had to be added for the integrator module.

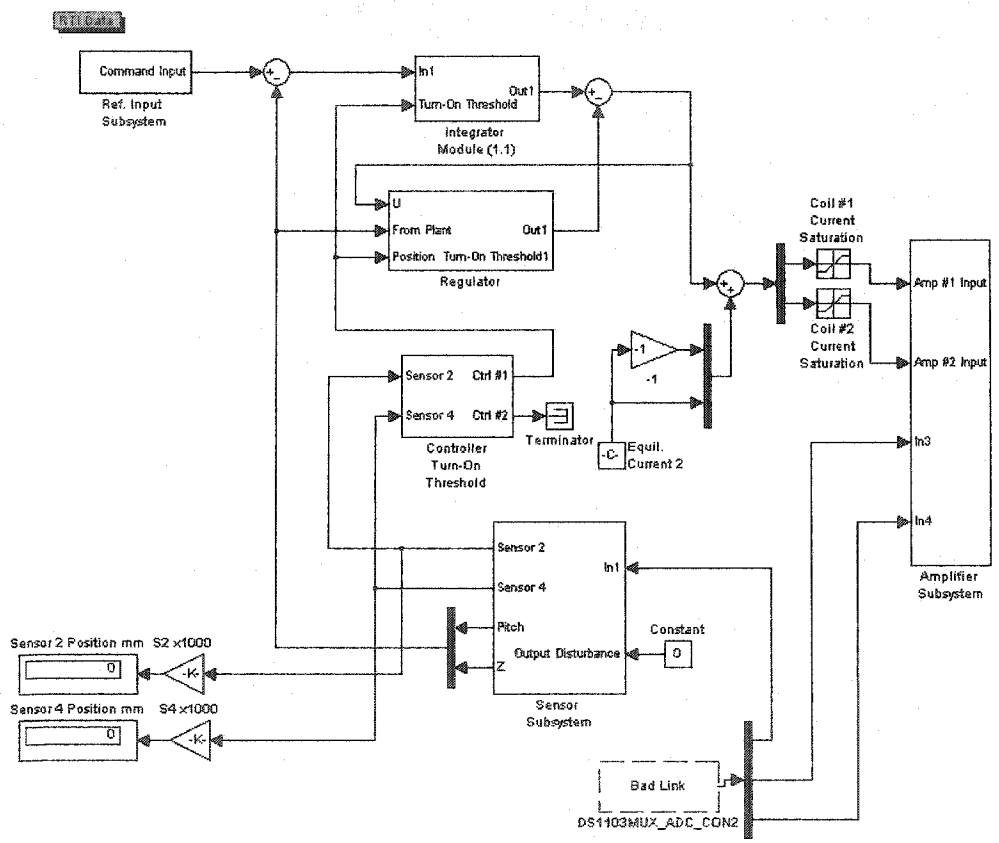


Fig. 74. Top level of Simulink model of real time controller for 2-DOF experiments.

Additional changes to the controller are contained in the block "Ref. Input Subsystem". One of these functions is the conversion of input commands from degrees and millimeters to radians and meters. This was needed because the angles in degrees are more natural for the operator, but the Matlab equations on which the controller was base were in radians and meters. Another function is the addition of a repeating command sequence that can be turned off and on at the Control Desk GUI.

Finally, additional operator screens were developed. The main operator's screen is presented in Fig. 75, which contains the essential control, monitor and data capture functions for the system. The additional complexity of the MIMO case is reflected in the use of two command inputs, one in "mm" and one in "deg", and two outputs, the "Z-Axis Displacement" and the "Pitch". Note that to simplify the control screen there is a single equilibrium current adjustment for both coils. Because the equilibrium currents are equal in magnitude but opposite in sign, it was a simple matter in the control logic to make a copy of the one setting, reverse its sign and apply it to the second amplifier/coil. The screen also contains an "Amplifier Master Enable" switch. This provides a single "kill" switch for both amplifiers, needed not only for emergencies, but also because it's safer to de-energize both coils simultaneously during system shutdown. When the coils are de-energized one at a time, the configuration momentarily simulates a 1-DOF system and forces the suspended core to rotate to align its magnetization vector vertically. This would cause the PVC housing to forcefully impact the magnets and frame.

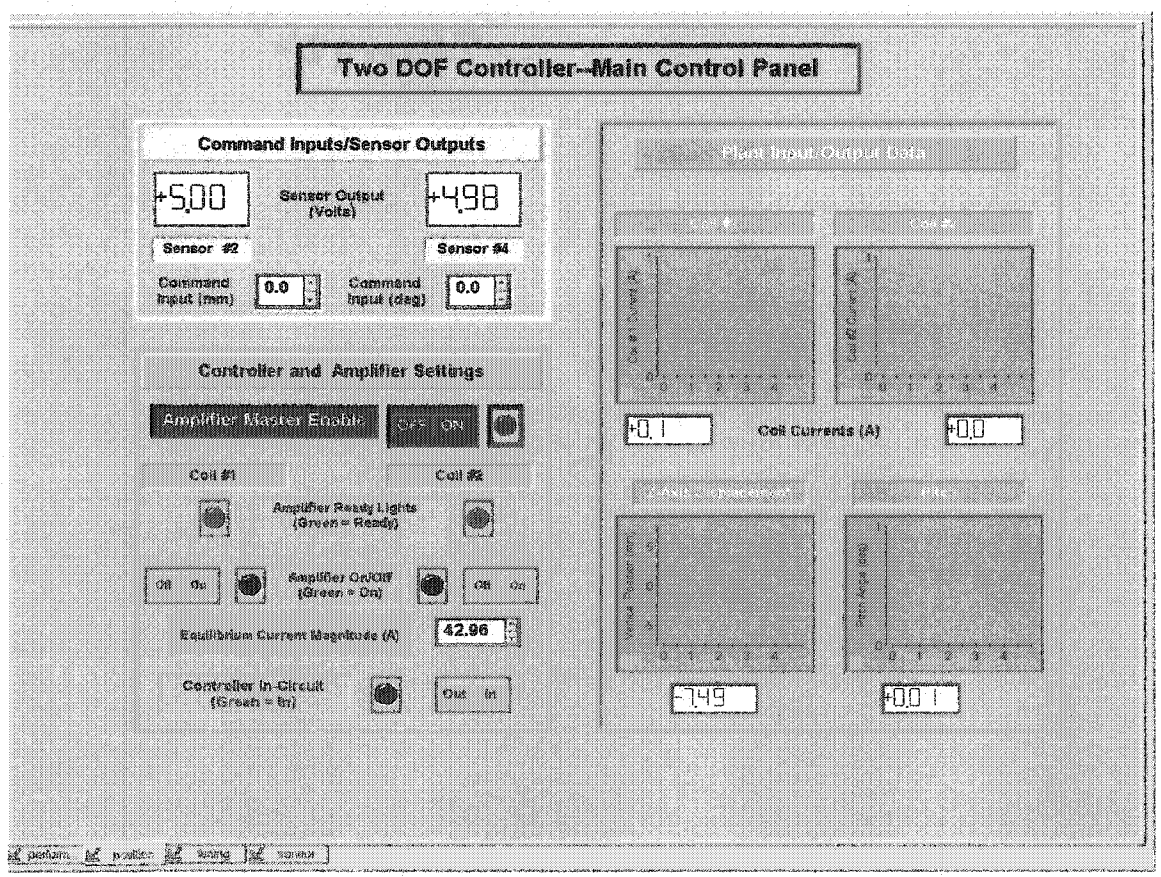


Fig. 75. Screen capture of main operator control panel for the 2-DOF experiments.

Experimental Procedure

This section covers the implementation of semi-automated controller synthesis procedures for the 2-DOF set up described in the last section. First, issues concerning optimal control design in the MIMO versus SISO case are addressed. Secondly, some of the considerations involving the design of a discrete time controller are covered. Then the two different design approaches used in this set of experiments are described and some of the features of their corresponding Matlab scripts are explained. Finally, the tests conducted with the actual MSBS plant are described.

Implementation of Discrete Time Controller

The controller for the 2-DOF maglev was designed in discrete time (DT) form, rather than in continuous time as were the earlier controllers. The reason for this was that designing directly in the DT form provides greater predictability of performance with the actual plant. This is because the controller is eventually implemented in sampled-data form anyway on the dSPACE real time processor (RTP). When the continuous time system is converted to sampled data form there are unknown and potentially drastic changes in its characteristics, due in part to the phase lag introduced by the zero-order hold function and the effects of using different sample rates. When the system is designed directly in discrete time, it is implemented as-is and there are fewer uncertainties regarding its performance with the real plant.

The most evident change in switching to discrete time design was using the zero-order hold (ZOH) form of the plant and state estimator models. This required inserting a ZOH block in the models between the plant and the Kalman estimator as well as the consistent use of the same sample period throughout all the Matlab functions and models.

Another important concern of using the DT form in optimal control design stems from the cross-coupling between the states and inputs that occurs when converting the CT cost function,

$$J = \frac{1}{2}\int_0^\infty (\mathbf{x}(t)^T \mathbf{Q}_C \mathbf{x}(t) + \mathbf{u}(t)^T \mathbf{R}_C \mathbf{u}(t))dt\,,$$

to an equivalent DT cost function. There is not a direct correspondence, so that the DT form is just

$$J = \frac{1}{2}\sum_{k=0}^{N} \mathbf{x}(k)^T \mathbf{Q}_C \mathbf{x}(k) + \mathbf{u}(k)^T \mathbf{R}_C \mathbf{u}(k)\,.$$

with the same matrices, $\mathbf{Q}$ and $\mathbf{R}$. Rather, the equivalent cost function for a DT realization $\{\Phi, \Gamma, \mathbf{H}\}$ is a complex expression that weights the product of $\mathbf{x}$ and $\mathbf{u}$ [26]. The relevance of

this is the state and input weighting scheme designed for the CT system does not apply directly to the DT system. Thus the rationale discussed in Chapter V for selecting the weights is undermined in the DT case—unless the equivalent DT forms are derived for **Q** and **R**. The Matlab command *lqrd.m* first finds the equivalent forms and then computes the LQR solution. However, this set of experiments was designed using the same controller synthesis commands shown to work in [4], which did not find the DT equivalent forms for **Q** and **R** (i.e., did not use *lqrd.m*). Consequently, for these experiments the relationship between the LQR design parameters and their effects was more ambiguous than it otherwise might have been.

Another adaptation had to be made to the earlier design scripts in order to correctly evaluate the frequency responses of the loop transfer functions. This was because of the aliasing that occurs in the sampled data environment when the frequency of the sampled signal exceeds a value equal to one half the sampling frequency, the so-called Nyquist frequency. Stated another way in the Sampling Theorem, this phenomenon implies a signal can be uniquely recoverd from discrete samples only if the sampling frequency is at least twice that of the original signal. This means, for a sample time of one millisecond and corresponding Nyquist Frequency of 500 Hz, plots of the loop gain responses of a discrete time implementation would not extend past 500 Hz since signals above that frequency could not be accurately represented. For this reason a function was added to the Matlab design scripts that converted the open loop transfer functions from a sampled to a continuous form at those points in the scripts that called for plotting loop responses. The method for conversion is termed "bilinear transformation".

Finally, a specific discrete time technique known as the Pincer method was employed in the LQR portion of the controller design. The Pincer method causes the resulting **K**-matrix to place all the system's closed loop poles within a specified radius from the origin of the z-plane. This allows designers to ensure all system modes settle to within some specified percentage of the final value in less than or equal to some specified time, t_s.

For a 1 % settling time equal to t_s, the Pincer method calculates a factor alpha such that at sample number k, the relationship of a state x(k) to its initial value of one is given by

$$\alpha^k x(k) = x(0), \text{ or}$$

$$x(kT)=(1/\alpha^k)x(0)=.01x(0).$$

This implies

$$1/\alpha^k \leq .01, \text{ or}$$

$$\alpha > 100^{\frac{1}{k}} = 100^{\frac{T}{t_s}},$$

where the target sample number, k, is related to the desired settling time through the relationship settling time divided by the sample period, or $k = \frac{t_s}{T_s}$.

The desired settling time is placed into the LQR calculation in Matlab by multiplying the state and control matrices phi and gamma by the factor alpha before using the matrices in Matlab's *dlqr.m* command. (Note this is a distinct command from *lqrd.m* discussed above.) A Matlab function LQSIM.m, created by Dr. Oscar González at Old Dominion University, was used to perform this process.

Automated Design Approach

Two design approaches were taken to automate the synthesis of a 2-DOF controller: the first using the LTR design technique and the second consisting strictly of a reiterative series of synthesis and evaluation of time responses based on a defined range of "tuning" parameters.

The LTR method was tried first, given the positive results obtained using this approach for the 1-DOF system. The process was basically the same as the last chapter, but the scripts were more complicated due to the directivity of the plant and multidimensional signals in the 2-DOF system. Specifically, a set of Matlab scripts were written to execute the steps listed in Table XXIX. A table listing the individual scripts themselves and denoting their interrelationships is provided in Appendix E.

TABLE XXIX
FUNCTIONALITY IMPLEMENTED IN SECOND STAGE OF DESIGN AUTOMATION

Sequence No.	Description of Functionality
1	Create the 2-DOF linear model and extract the four states of interest: $\dot{\theta}_y, \theta_y, \dot{z}, z$.
2	Create a range of process noise gain matrices based on user input, derive a four-state Kalman estimator for each matrix and obtain the open loop frequency response for each. (Assume fixed values for the process and sensor noise covariance matrices $\mathbf{Q}_n$ and $\mathbf{R}_n$.) Automatically prescreen the responses and display on screen only those responses meeting user-provided criteria (e.g., the zero-crossover bandwidth). Allow user selection of best response.
3	Based on keyboard input from the user, determine whether to use integral control.

TABLE XXIX, CONT'D

Sequence No.	Description of Functionality
4	Obtain the desired values for the weighting matrices, **Q** and **R**. Create range of matrices and obtain the corresponding LSVF matrices, **K**. Instead of the Matlab command *dlqr.m*, this experiment used a small routine to derive the **K**-matrices using the Pincer method, described above. [1]
5	For each LQR solution, combine the resulting **K**-matrix, the Kalman estimator selected previously and the plant model to obtain the system open loop frequency response. Plot the responses and allow user selection of the best match.
6	Obtain step response for a combination of reference inputs using the selected controller design and the appropriate nonlinear model. (I.e., with integral or non-integral control.) Determine pertinent time response values for each reference input: percent overshoot, settling time, steady state value, and maximum deviation of control input. Store responses in an array.
7	Obtain stiffness quotients for a combination of reference inputs using the selected controller design and the appropriate nonlinear model. Store responses in an array.
8	Perform B-field perturbation tests for a combination of reference inputs. Store responses in an array.

The results obtained from the LTR approach were disappointing, and a second approach was tried in which the design parameters used in [4] were applied as closely as possible. This approach was highly iterative and involved several thousand simulations. A set of Matlab scripts were written to perform the following functions. A table listing the individual scripts and denoting their interrelationships is provided in Appendix E.

TABLE XXX
FUNCTIONALITY IMPLEMENTED IN SECOND STAGE OF DESIGN AUTOMATION

Sequence No.	Description of Functionality
1	Create and reduce the linear plant model as in the LTR approach in Table XIX.
2	Based on keyboard input from the user, determine whether to use integral control.
3	Create ranges of design parameters: $\mathbf{G}_w$, $\mathbf{Q}$, $\mathbf{R}$ according to settings with in program. Hold other parameters fixed: $\mathbf{Q}_n$, $\mathbf{R}_n$. For each permutation of parameters, create LQR with Kalman estimator.
4	Obtain step responses, stiffness quotients and B-Field perturbation responses as above. (Note for this process step responses were determine for only one reference input combination: 1 degree pitch and 2 mm position change.) Save all data for each controller design in a separate file, including all parameter settings and responses.
5	When controller synthesis run is complete, retrieve each data file sequentially. Evaluate all responses according to user-provided criteria for percent overshoot, settling time and the maximum deviation in control current. Select values according to hierarchy: 1) maximum deviation of control input; 2) maximum percent overshoot; and 3) maximum settling time.

Using this approach, a controller was obtained with acceptable, though not impressive, response characteristics. (See Table XXXII in following section.) However, the combinations of parameters tried up to that point had been fairly exhaustive, requiring approximately 2500 separate simulations. After evaluating so many candidates, it seemed best at that point to validate the best controller obtained thus far against the real plant in order to set a baseline for further design.

The LTI model for the estimator (Kest) and the feedback matrices ($\mathbf{L}$ and $\mathbf{K}$) were loaded into a Simulink model similar to the one in Fig. 74 (but without the integrators) and the model was compiled and loaded on the dSPACE Real Time Processor (RTP). The variables in the controller were linked to the corresponding virtual instruments in a GUI similar to the one in Fig. 75 and the system was ready for turn-on.

The test object could not be levitated at first, and it seemed the torque response of the controller was much too vigorous. The controller model and GUI were modified to reduce the individual elements of the LSVF matrix and the gains corresponding to the pitch states were reduced, which allowed the test cylinder to be levitated. After adjustments were made to the angle feedback calculations, and the test cylinder was levitated with the gains restored to their initial values. The control input levels (suspension coil currents) and plant outputs were measured and recorded for a

series of input commands. The design parameters for that controller were stored in a Matlab script written to re-synthesize the controller when needed.

An improved controller was sought by incorporating integrators to reduce the steady state error. The second simulation model was designed and the design approach in Table XXX was revised as shown in Step 2 then repeated. The improvement in time response was so drastic that no more than 150 iterations were needed to derive an acceptable controller. The Simulink model for the real time controller had to be revised to incorporate the integrators. In addition, a feature was added to the reference input section where a pre-programmed repeating input sequence could be initiated from the operator control panel.

The test cylinder was launched without difficulty and the controller tested with a series command inputs. The system variables were captured as before and the controller design parameters saved in a Matlab script.

Table XXVII contains the design parameters used for the two controllers tested on the actual two-DOF plant.

TABLE XXXI
PARAMETERS FOR DESIGN OF CONTROLLERS TESTED ON THE 2-DOF PLANT

Parameter	LTR Design	Non-Integral Control	Integral Control
$\mathbf{Q_n}$	$\mathbf{Q_n} = \Gamma * \Gamma^T$	$\mathbf{Q_n} = \Gamma * \Gamma^T$	$\mathbf{Q_n} = \Gamma * \Gamma^T$
$\mathbf{R_n}$	$\mathbf{R_n} = \begin{bmatrix} 1e{-}5 & 0 \\ 0 & 1e{-}5 \end{bmatrix}$	$\mathbf{R_n} = \begin{bmatrix} 1e{-}5 & 0 \\ 0 & 1e{-}5 \end{bmatrix}$	$\mathbf{R_n} = \begin{bmatrix} 1e{-}5 & 0 \\ 0 & 1e{-}5 \end{bmatrix}$
$\mathbf{G_w}$ (diagonal)	$10000 * eye(4)$	$12500 * eye(4)$	$12500 * eye(4)$
$\mathbf{Q}$ (diagonal)	$[0 \quad 1 \quad 0 \quad 1]$ [a]	$[1000 \quad 3000 \quad 2000 \quad 3000]$	$[500 \quad 500 \quad 500 \quad 500 \quad 500 \quad 500]$
$\mathbf{R}$	$R = \begin{bmatrix} .1 & 0 \\ 0 & .1 \end{bmatrix}$	$R = \begin{bmatrix} .1 & 0 \\ 0 & .1 \end{bmatrix}$	$R = \begin{bmatrix} .1 & 0 \\ 0 & .1 \end{bmatrix}$

[a] Derived from $(C^T * C)$

Presentation and Discussion of Results

Data and Plots from Controller Synthesis and Validation

This section presents typical data and time response plots obtained from the automated, iterative synthesis and validation process for the 2-DOF LQR with Kalman estimator. Results are shown

for the regulator both with and without integral control. Stiffness quotients and B-Perturbation responses are shown for the designs that were selected for use with the actual MSBS plant.

Table XXXII shows selected results for an LQR/Kalman estimator controller (without integral control) using the parameters in Table XXVII. The reference input vector was the same for all simulation runs: [1 deg; 2 mm] and the validation routine contained the following threshold criteria in the rank order indicated:

1) % Overshoot 19%

2) Settling Time: 1.9 s

3) Max Input Deviation: 19 A

Note the steady state value was not included among the step responses because the maximum input deviation and percent overshoot were much higher priority in the initial validations.

TABLE XXXII
SELECTED RESULTS FROM ITERATIVE SYNTHESIS AND VALIDATION OF 2-DOF CONTROLLER

$Q_{1,1}$	$Q_{2,2}$	$Q_{3,3}$	$Q_{4,4}$	% OS (pitch/position)	Settling Time (s)	Max Input Deviation (A)
1000	3000	1000	1000	18.88 -0.40708	1.8695	13.035 17.884
1000	3000	1000	2000	18.88 -0.40212	1.8695	13.022 17.898
1000	3000	1000	3000	18.88 -0.39731	1.8695	13.008 17.911
1000	3000	2000	1000	18.88 -0.30756	1.8694	12.441 18.479
1000	3000	2000	2000	18.88 -0.30539	1.8694	12.431 18.489
1000	**3000**	**2000**	**3000**	**18.88** **-0.30326**	**1.8694**	**12.421** [a] **18.498**
1000	3000	3000	1000	18.88 -0.25929	1.8693	11.971 18.949
1000	3000	3000	2000	18.88 -0.258	1.8693	11.963
			3000	18.88 -0.2567	1.8693	11.955 18.964

[a] Boldface type indicates parameters selected for final controller.

Table XXXIII presents selected results from the synthesis and validation of a controller using integral control with the parameters from Table XXVII. The reference input vector in radians and meters was $\begin{bmatrix} .01745 & .002 \end{bmatrix}^T$.

TABLE XXXIII
SELECTED RESULTS FROM ITERATIVE SYNTHESIS OF 2-DOF CONTROLLER WITH INTEGRAL CONTROL

$Q_{3,3}$	$Q_{4,4}$	$Q_{5,5}$	$Q_{6,6}$	% OS (pitch/position)	Settling Time (s)	Max Input Deviation (A)
0	0	0	0	14.439 0.91536	1.9634	0.85549 1.507
0	500	0	0	9.4561 0.85521	1.9004	1.0758 1.812
0	500	0	500	9.4534 0.82339	1.9004	1.0437 1.8444
0	500	500	0	9.4505 0.36214	1.9003	0.92392 2.0736
0	500	500	500	9.4504 0.36082	1.9003	0.92315 2.0765
500	0	0	0	0.83009 0.61502	0.8573	2.081 2.5471
500	0	0	500	0.83017 0.60164	0.8573	2.0571 2.574
500	0	500	0	0.83025 0.25753	0.8574	1.7696 2.8902
500	0	500	500	0.83025 0.25663	0.8574	1.7665 2.8937
500	500	0	0	0.8281 0.61597	0.8564	2.095 2.5607
500	500	0	500	0.82818 0.60266	0.8564	2.0711 2.5874
500	500	500	0	0.82826 0.25796	0.8564	1.7836 2.9033
500	**500**	**500**	**500**	**0.82827** **0.25705**	**0.8564**	**1.7806** [a] **2.9068**

[a] Boldface type indicates parameters selected for final controller.

Figs. 76 and 77 are presented to compare the ranges of responses from the various non-integral and integral controller designs. Note the difference in amplitude scales.

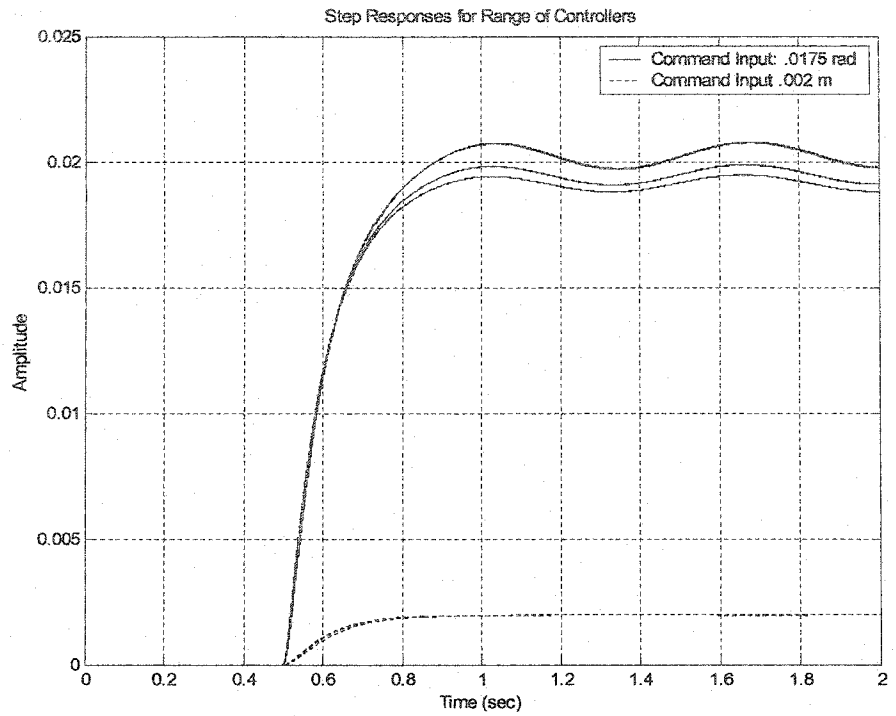


Fig. 76. Step responses for range of 2-DOF regulator designs without integral control.

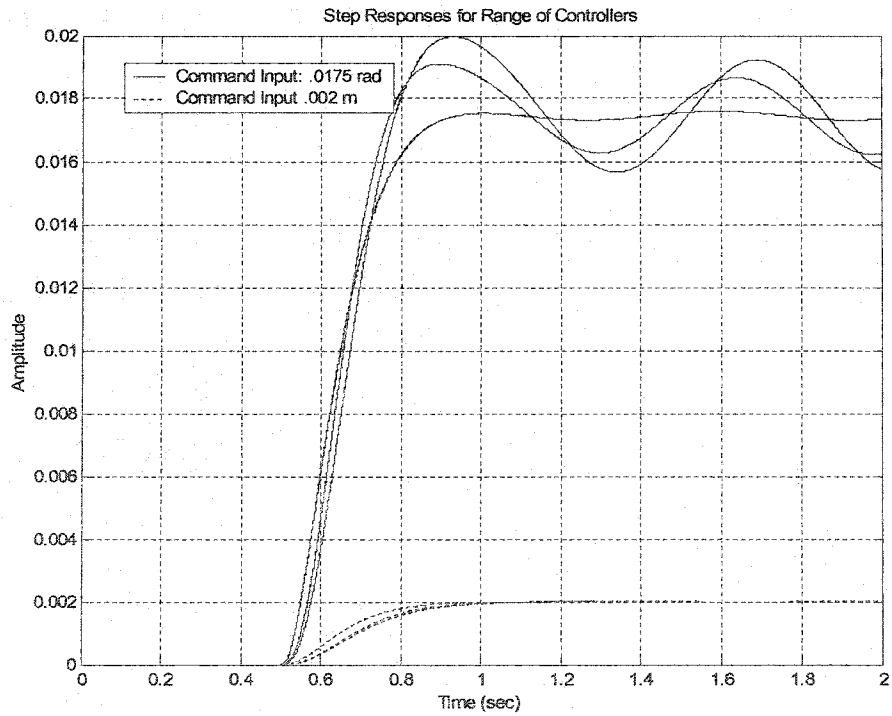


Fig. 77. Step responses for range of 2-DOF regulator designs with integral control.

Table XXXIV compares the stiffness quotients of the non-integral and integral controller designs selected for live testing on the MSBS plant. There are nine combinations of reference inputs.

TABLE XXXIV
COMPARISON OF STIFFNESS RATIOS BETWEEN NON-INTEGRAL AND INTEGRAL REGULATORS

Ref Inputs	Controller Design	Final Values	Positive Stiffness Ratio	Negative Stiffness Ratio	Average Stiffness Ratio
-0.017453 rad	Non-	-0.0197 rad	439.6592	-69.1497	254.4045
-0.002 m	Integral	-0.0020 m	2964.8858	-3235.3425	3100.1141
	Integral	-0.01705 rad	204.3579	-73.4800	138.9190
		-0.00199 m	4443.2390	-2934.7974	3689.0182
-0.017453 rad	Non-	0.0197 rad	433.7844	-70.1190	251.9517
0.000 m	Integral	0.0 m	3017.7652	-3288.1268	3152.9460
	Integral	-0.01706 rad	206.9195	-74.6756	140.7976
		0.00000 m	4516.5976	-2990.7229	3753.6603
-.017543 rad	Non-	-0.0197 rad	427.7805	-71.0467	249.4136
0.002 m	Integral	0.00199 m	3070.6036	-3340.6719	3205.6378
	Integral	-.01706 rad	209.2192	-75.8460	142.5326
		0.00200 m	4593.8493	-3046.6282	3820.2387
0.0 rad	Non-	0.000 rad	72.7555	-68.8804	70.8179
-0.002 m	Integral	-0.00200 m	2961.0097	-3298.3660	3129.6879
	Integral	0.000 rad	79.1545	-87.1619	83.1582
		-0.00200	453.4461	-2975.6258	3714.5360
0.0 rad	Non-	0.000 rad	73.9959	-70.2818	72.1388
0.000 m	Integral	0.0000 m	3013.7950	-3350.3100	3182.0525
	Integral	0.000 rad	80.8644	-88.7396	84.8020
		0.0000 m	4526.6361	-3031.0602	3778.8482
0.0 rad	Non-	0.000 rad	75.1858	-71.6459	73.4159
0.002 m	Integral	0.00199 m	3066.5378	-3402.0812	3234.3095
	Integral	0.000 rad	82.5196	-90.2608	86.3902
		0.00199 m	4603.7780	-3086.5027	3845.1403
0.017453 rad	Non-	0.0197 rad	38.9365	-66.4098	52.6731
-0.002 m	Integral	0.002 m	3037.7166	-3678.382	3202.7774
	Integral	0.01705 rad	47.7623	-94.2981	71.0302
		0.00198 m	4557.3153	-3011.3061	3784.3107
0.017453 rad	Non-	0.01974 rad	39.6894	-67.7843	53.7369
0.0 m	Integral	0.00000 m	3090.0218	-3418.8774	3254.4496
	Integral	0.01706 rad	48.8767	-96.2888	72.5828
		0.00001 m	4629.5991	-3066.4580	3848.0286
0.017453 rad	Non-	0.01974 rad	40.4216	-69.1269	54.7743
0.002 m	Integral	0.00199 m	3142.3337	-3469.7957	3306.0647
	Integral	0.01706 rad	49.9690	-98.1939	74.0815
		0.00201 m	4706.0326	-3121.6425	3913.8376

Table XXXV compares the results of B-field perturbation tests of the non-integral and integral controller designs selected for testing with the MSBS plant.

TABLE XXXV
B-FIELD PERTURBATION TEST DATA FOR NON-INTEGRAL AND INTEGRAL REGULATOR DESIGNS

	Percent Overshoot		2% Settling TIme		Steady State (m)		Max. Input Deviation	
Ref. Input	Mean	Standard Devia-tion	Mean	Standard Devia-tion	Mean	Standard Devia-tion	Mean	Standard Devia-tion
1.0 deg	18.8852	0.1560	1.8677	0.0021	0.0197	0.0000	12.3957	0.7714
2.0 mm	-1.0307	21.8540	1.0200	0.0783	0.0020	0.0004	18.5240	0.7714
1.0 deg[a]	2.7653	0.0171	1.8008	0.0010	0.0171	0.0000	0.9823	0.3438
2.0 mm[a]	0.5424	0.0046	0.9993	0.0008	0.0020	0.0000	2.4819	0.3671

[a] Integral Regulator Design

Fig. 78 contains the both the estimator and system open loop responses for the first design attempt using the LTR method. Each response is characterized by the maximum and minimum singular values of the corresponding transfer function. The parameter set used for this controller was based on the successful LTR design in Chapter V. Fig 79 contains the open loop responses for the non-integral controller designed by the iterative method. The LTR method was retried using the successful parameters.

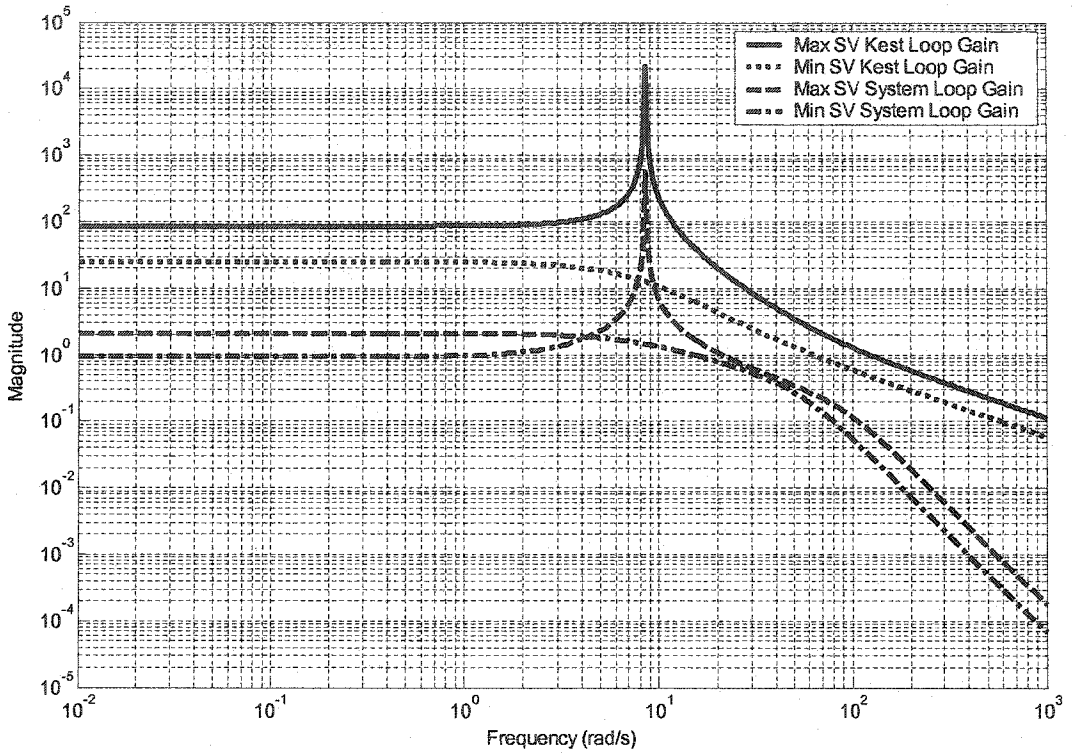


Fig. 78. Max/min open-loop singular values for initial LTR controller design. The upper two curves are the maximum and minimum singular values for the estimator open loop gain and the lower two curves are the singular values for the system open loop gain.

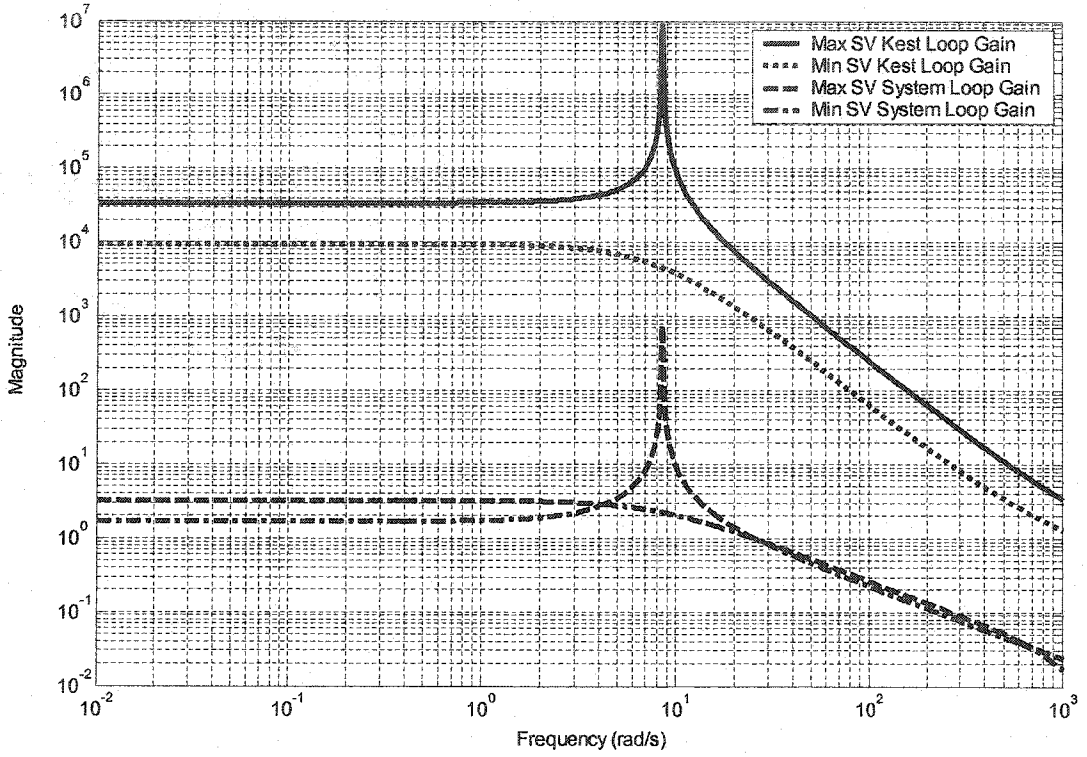


Fig. 79. Max/min open-loop singular values for acceptable controller design. The upper two curves are the maximum and minimum singular values for the estimator open loop gain and the lower two curves are the singular values for the system open loop gain.

The next group of figures presents the results of live testing on the actual MSBS plant using the non-integral controller. Only the coil currents for one command input, -0.5 to 0 degrees, are shown. A prominent spike is seen at the point in time the command was given.

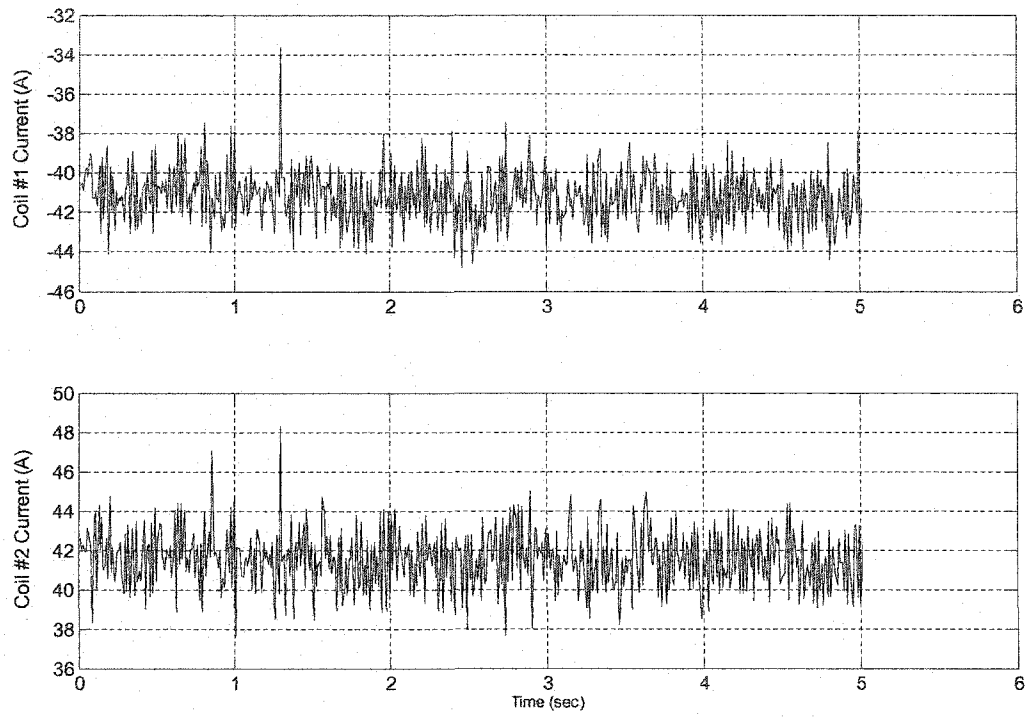


Fig. 80. Coil currents of plant with non-integral controller for change of pitch from neg. 0.5 to zero degrees.

Fig. 81 shows the 2-DOF plant's response to a series of one-half-degree pitch commands. Note the dotted lines showing the relative changes in position that occur simultaneously with the pitch commands. The plots show the stability in pitch is much better than that in vertical position, but they don't show interaction between the commands. Similarly, Fig. 82 displays the plant's response to a series of one-millimeter position commands with the corresponding changes in pitch denoted by dotted lines. These plots also show marginal response to position commands, but the output error is obscured by the instability.

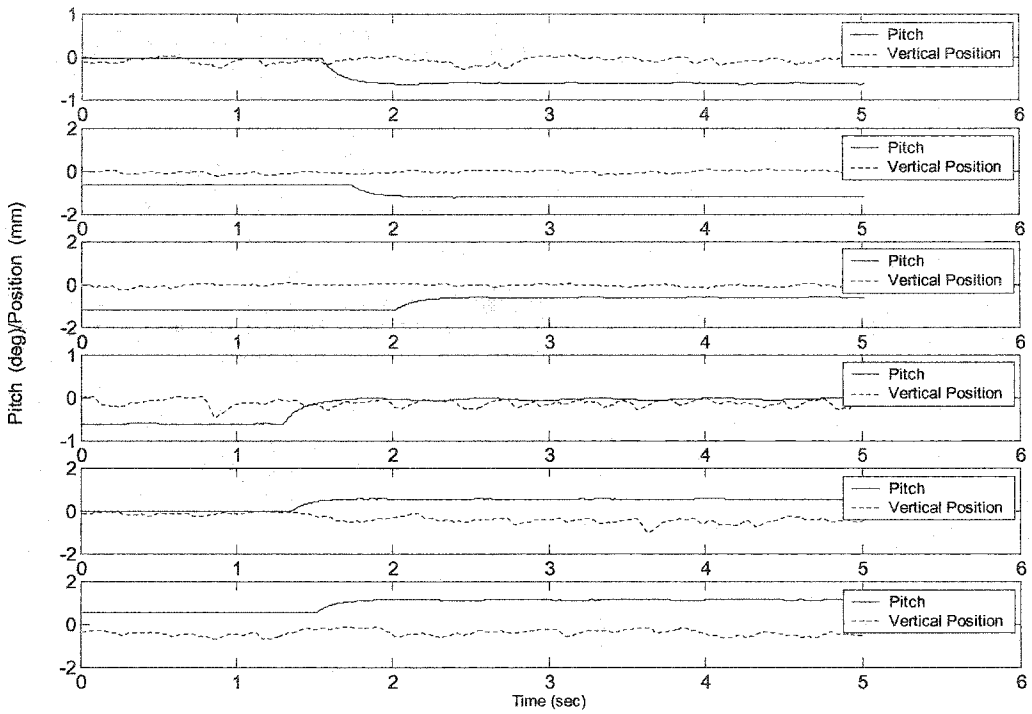


Fig. 81. Responses of plant with non-integral controller to pitch changes: 0 to -0.5, -0.5 to -1.0, -1.0 to -0.5, -0.5 to 0.0, 0.5 to 0.5, 0.5 to 1.0 degrees. Corresponding simultaneous position changes are also shown.

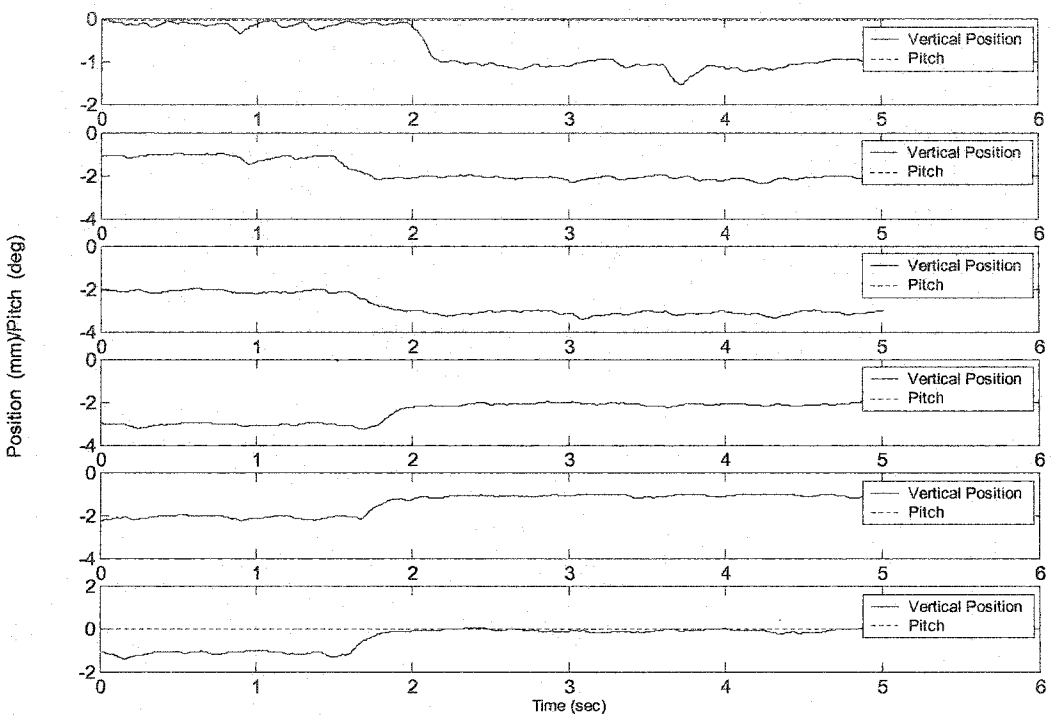


Fig. 82. Responses of plant with non-integral controller to position changes: 0.0 to -1, -1 to -2, -2 to -3, -3 to -2, -2 to -1, -1 to 0.0 mm. Corresponding simultaneous pitch changes also shown.

The last set of plots illustrates the behavior of the 2-DOF plant under the integral controller. Fig. 83. shows the coil inputs for the whole range of command inputs, issued over period of about 37 seconds. The time scale is compressed over what is shown in Fig. 80 so the current spikes occurring for the command changes are not as apparent as before. Also, the input current deviations are expected to be much smaller with integral control.

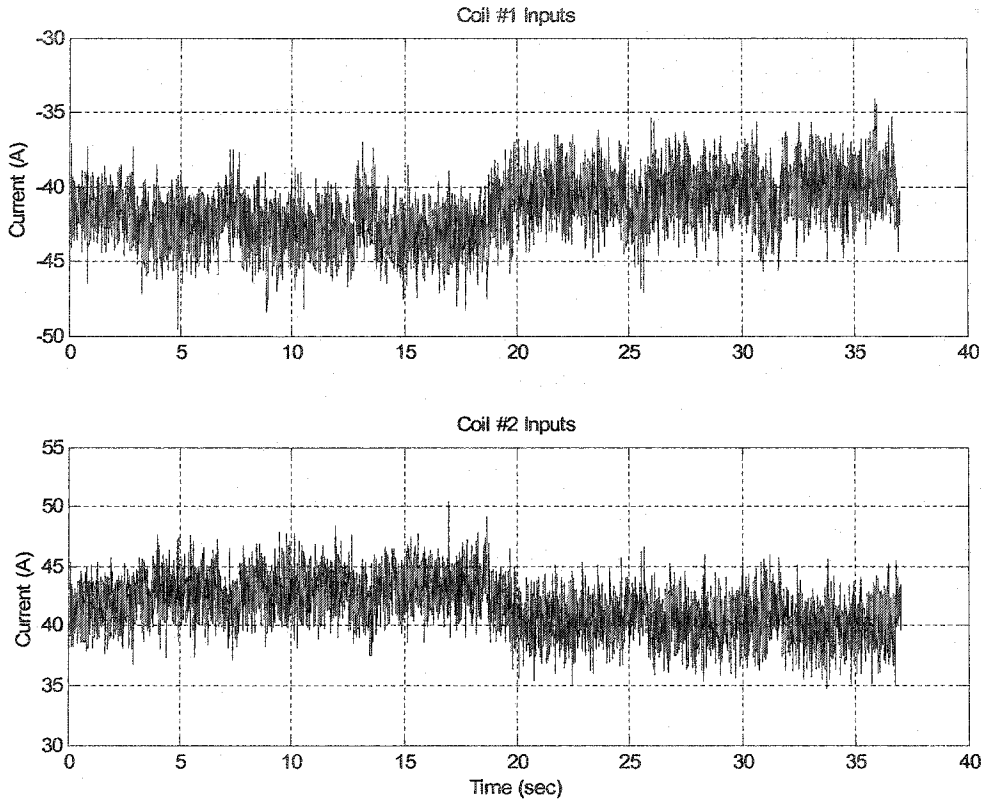


Fig. 83. Coil 1 current (upper plot) and coil 2 current (lower plot) for MSBS plant using integral control.

Figs. 84 and 85 compare the responses of the nonlinear model and actual 2-DOF plant responses for both pitch and position commands. The controller design included integral control using the parameters in Table XXVII. Fig. 86 illustrates how the x-axis position in the nonlinear model is unstable when the 2-DOF controller is used and that this instability is coupled to the pitch state by the model dynamics.

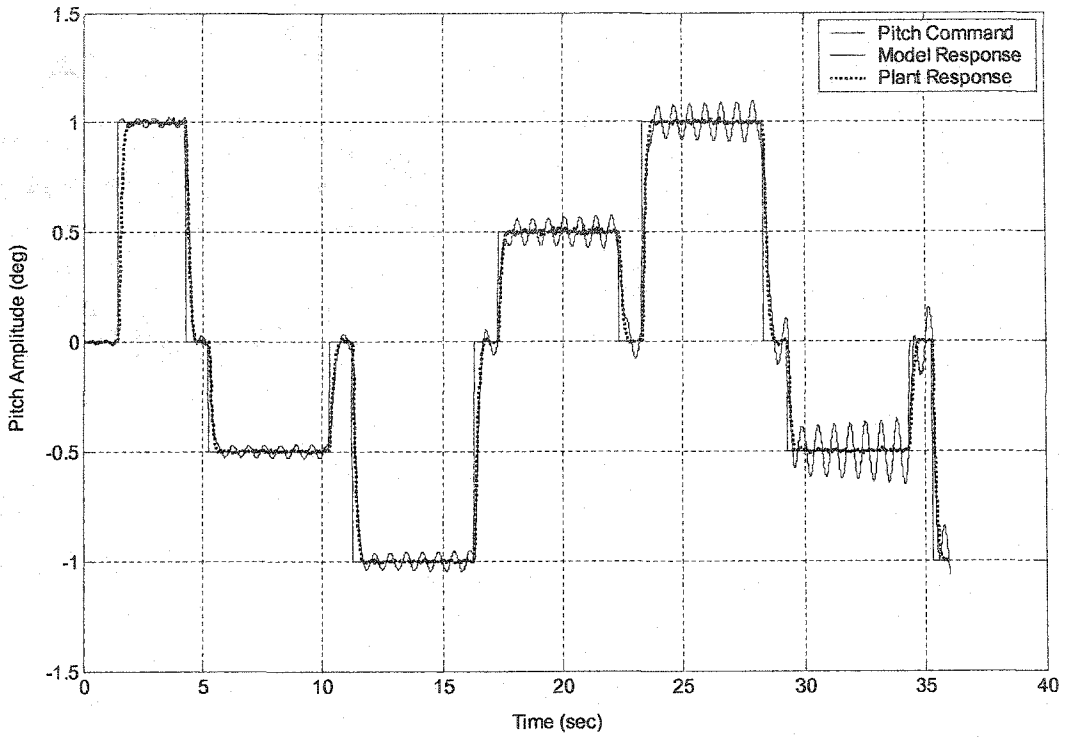


Fig. 84. Reference input for pitch and corresponding responses of nonlinear model and actual MSBS plant using LQG regulator with integral control.

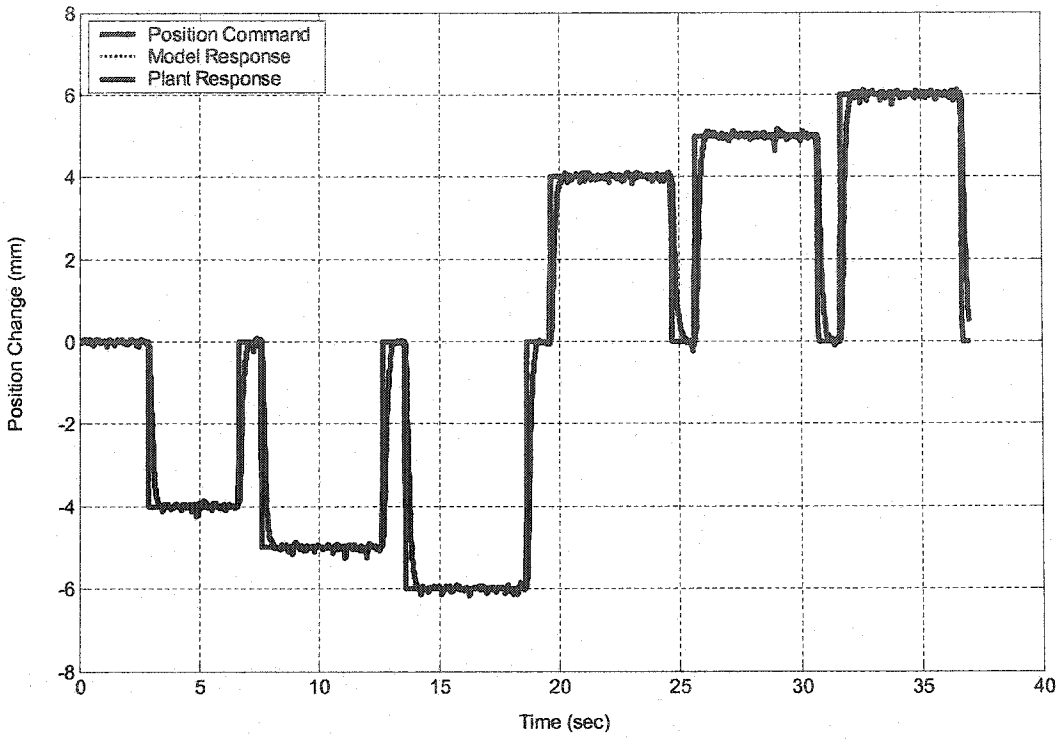


Fig. 85. Reference input for position and corresponding responses of nonlinear model and actual MSBS plant using LQG regulator with integral control.

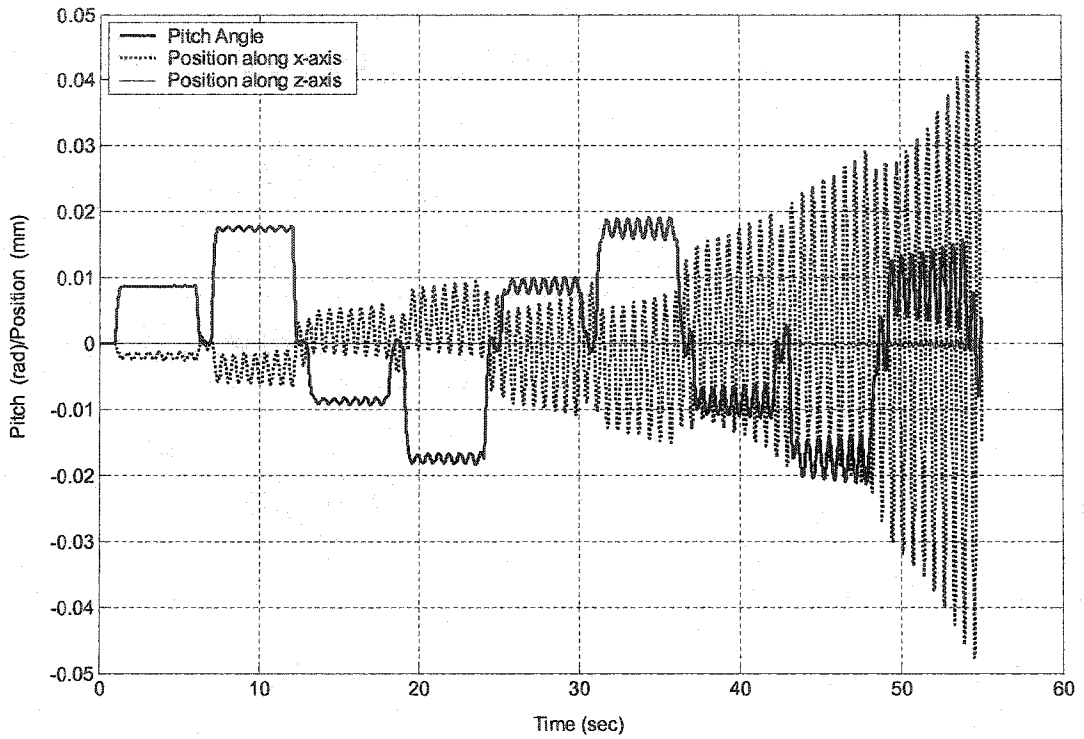


Fig. 86. Responses of nonlinear model states θ_y, x and z to changes in pitch reference input.

The following figures illustrate the 2-DOF experiments done on the wooden frame depicted in Fig. 17 in Chapter IV. Fig. 87 is a side view of the magnet assembly and suspended PVC tube. In this view the suspended tube is held at a slightly positive pitch angle of approximately one degree. The positive direction in pirch is taken as a positive angular displacement (i.e., anti-clockwise) around the x-axis. Note the suspended tube is situated such that its upper edge lies in the laser sensor beams. Fig. 88 is another side view illustrating the relative size of the PVC tube and the levitation gap. Note the major gradations on the ruler are in centimeters. Fig. 89 is an end view of the test frame and leviated object. This view more clearly shows the suspension gap, i.e., the distance between bottom of the coil and the centroid of the suspended object.

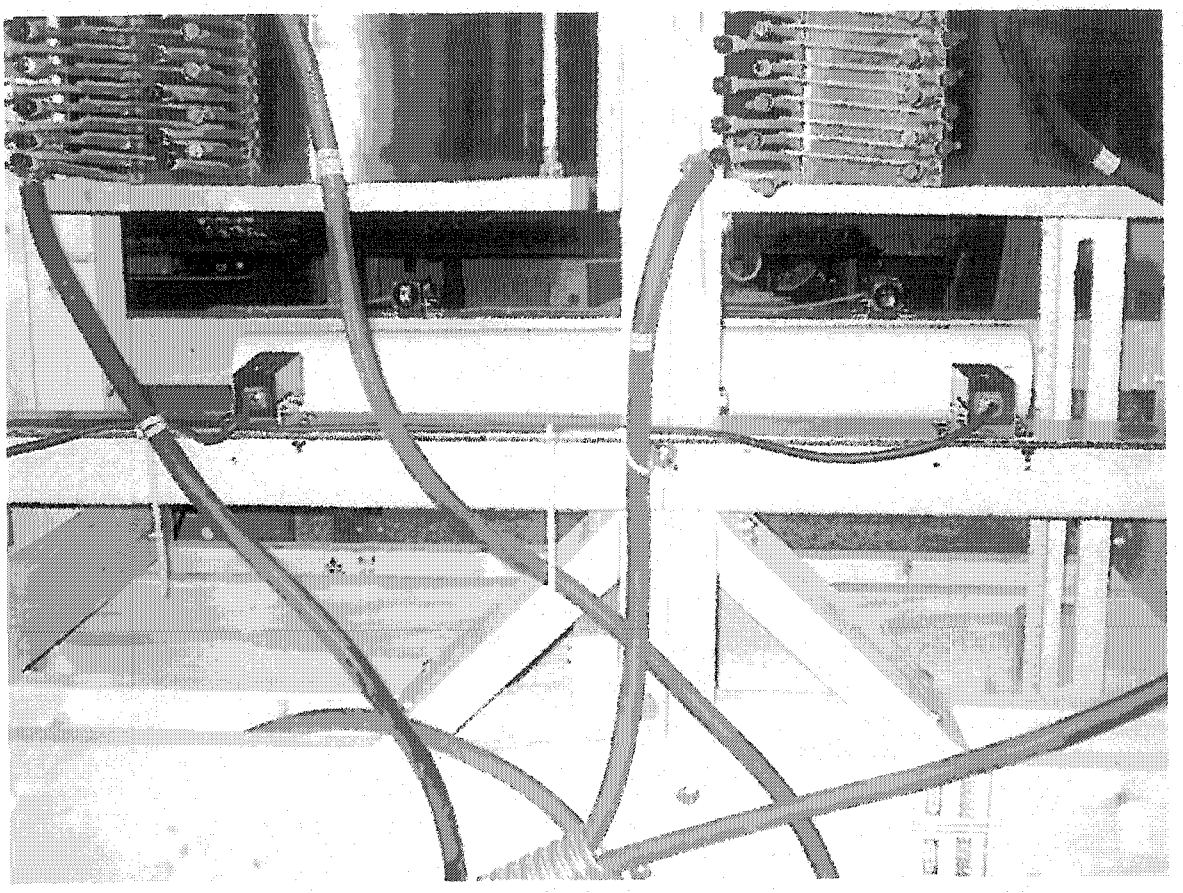


Fig. 87. Side view of 2-DOF levitation.

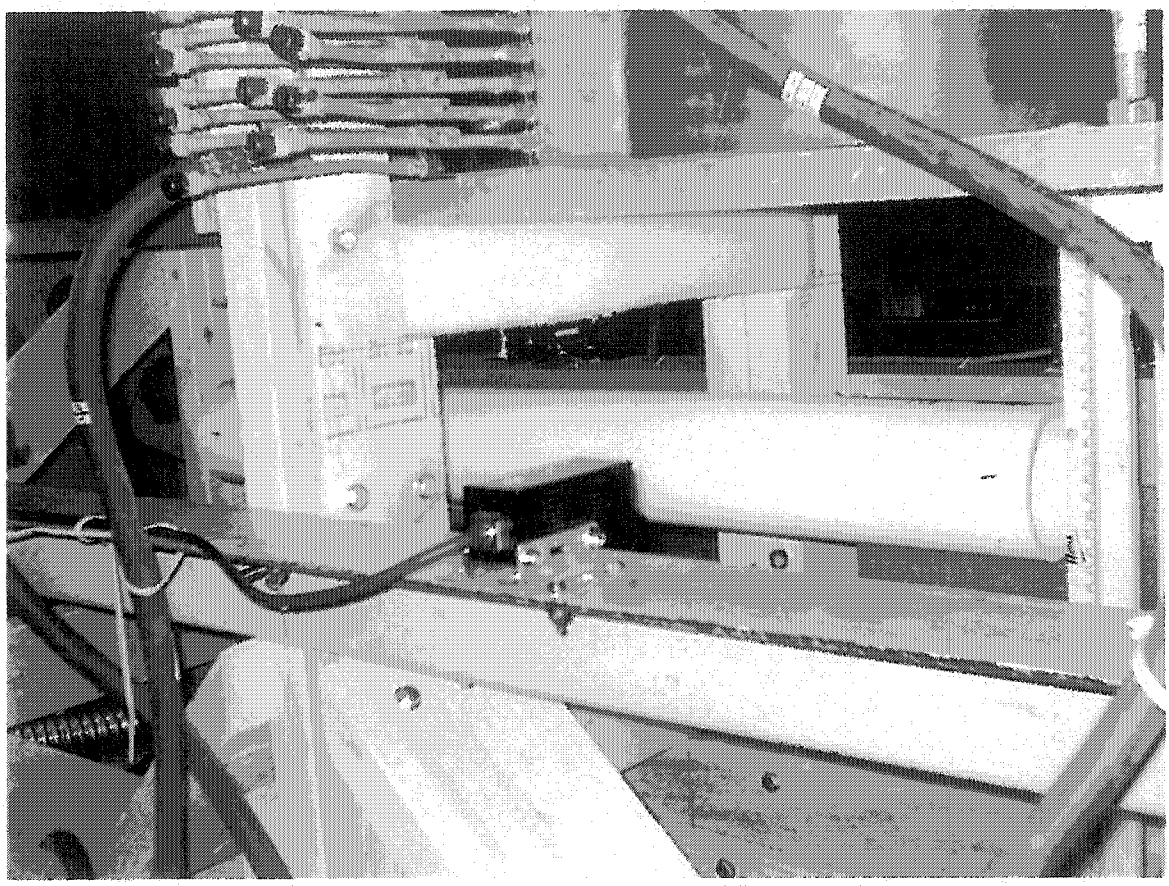


Fig. 88. Oblique view of 2-DOF leviataion. of suspended object and levitation gap.

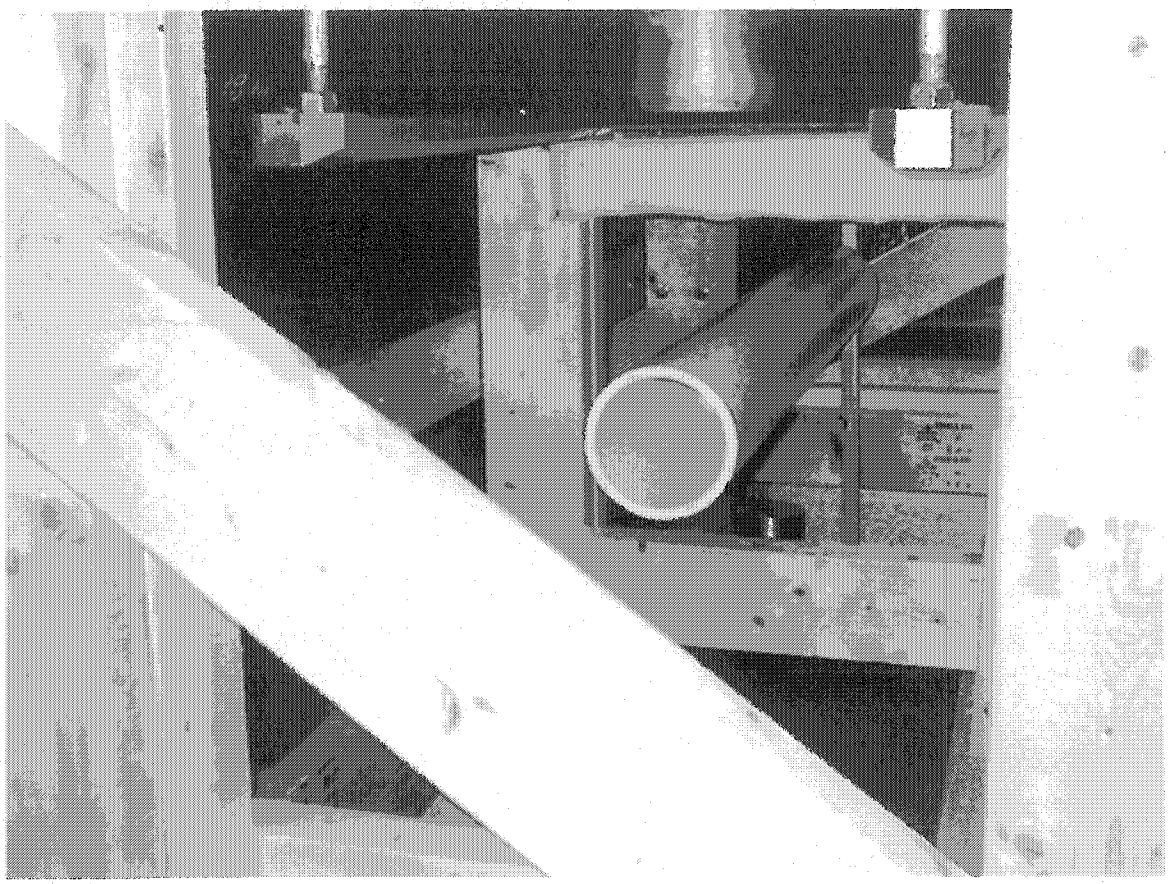


Fig. 89. End view of 2-DOF levitation.

At the close of the research leading up to this thesis, a 2-DOF levitation was performed inside the stainless steel suspension chamber. See Fig. 90, which shows the MSBS suspension chamber temporarily fitted with the a pair of suspension coils identical to those used in the wooden frame. (Note the axially wound coils are not yet completed.) First and LQG regulator without integral control was tried and then an LQG regulator with integral control (the one described in Table XXVII) was used. This experiment was intended to determine whether the basic 2-DOF controller design would work in this setting without significant tinkering. This was in fact shown to be true in that only the bias, or equiplibrium, current had to be adjusted to achieve the 2-DOF levitation depicted in Fig. 91. No measurements have been taken yet for this experiment.

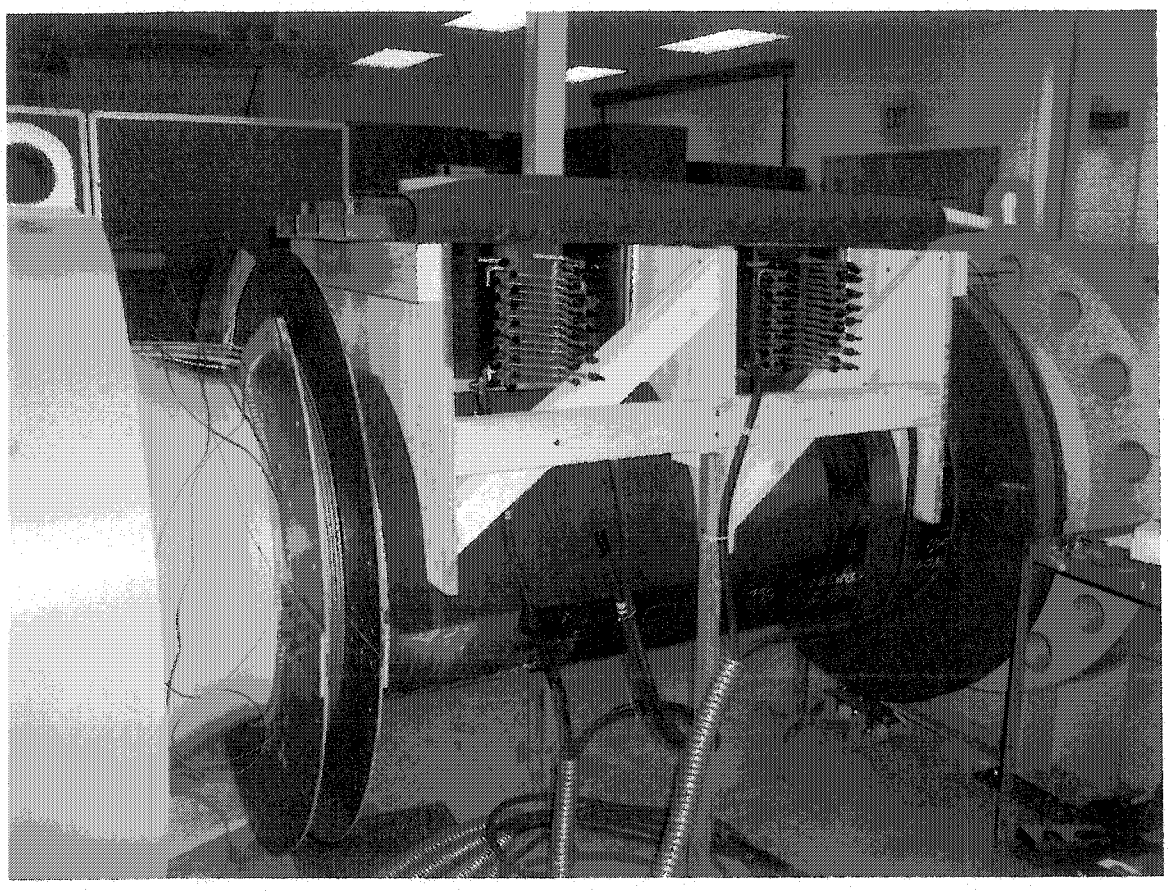


Fig. 90. MSBS suspension chamber with suspension coils temporarily mounted on top for 2-DOF levitation inside the chamber.

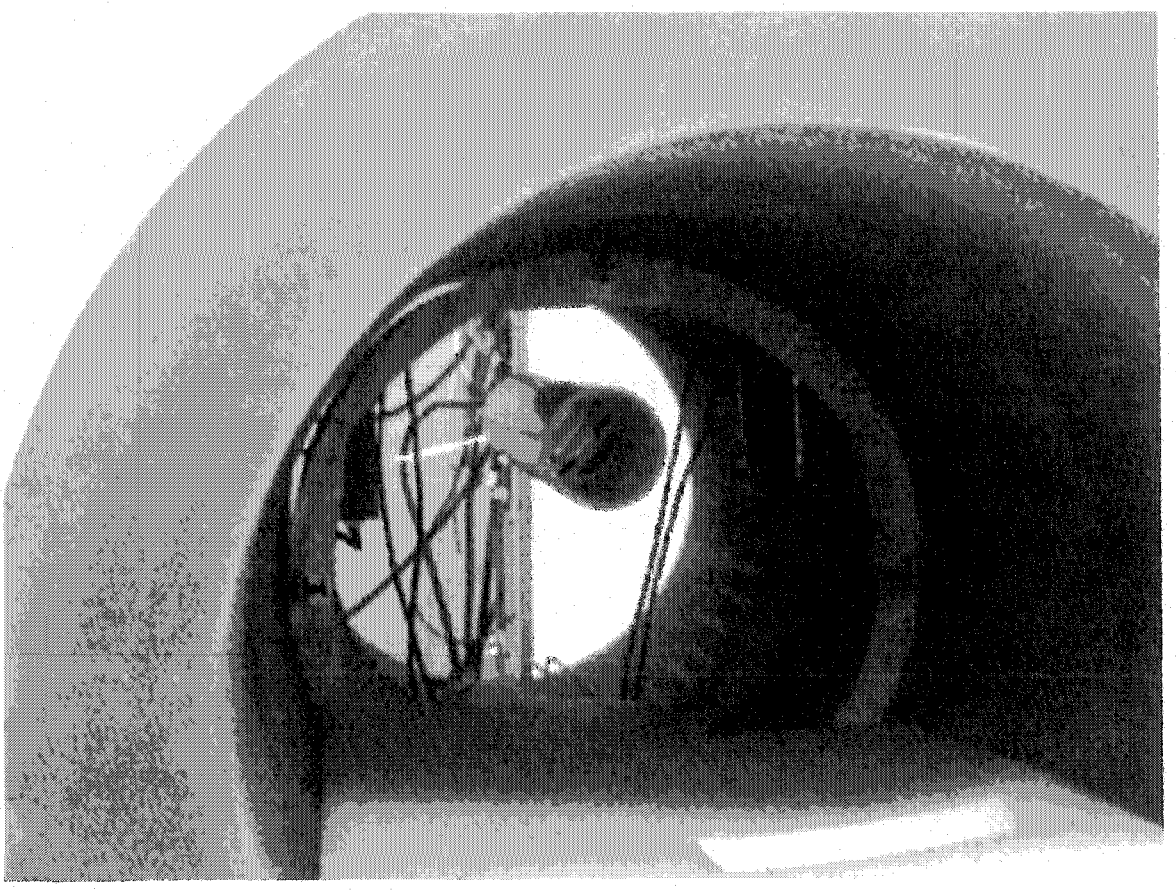


Fig. 91. 2-DOF levitation inside the MSBS suspension chamber.

Discussion of Results

The LTR design method was tried first with disappointing results. The controller was unable to maintain stability in the nonlinear model for even short period. The open loop responses in Fig. 78 indicate that should not have been a surprise, as the minimum system open loop gain was at or below 0 dB for most of the frequency spectrum. This illustrates the effects of directivity in MIMO plant response.

Later, after an acceptable design was found using the iterative approach and emulating the parameters in [4], the LTR method was revisited using the known good design parameters. The open loop responses from this design are shown in Fig. 79, which indicates a benefit to be gained

by incorporating loop shaping techniques with measurements of time domain responses into the controller design process. It was noted the sensor noise covariance values for the working LTR design were many orders of magnitude smaller than for the faulty design. To investigate the effect of this disparity, the pole locations of each design were checked. These values, compared in Table XXXVI, indicated the higher value of sensor noise caused the optimization algorithm to select an estimator that was too slow to stabilize the plant.

TABLE XXXVI
COMPARISON OF KALMAN FILTER EIGENVALUES FOR TWO DIFFERENT SENSOR NOISE COVARIANCE MATRICES

Lower Sensor Noise Matrix and Estimator Poles	Higher Sensor Noise Matrix and Estimator Poles
$R_n = \begin{bmatrix} 1e{-}5 & 0 \\ 0 & 1e{-}5 \end{bmatrix}$	$R_n = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$
$.5590 \pm j0.8170$, $.6639 \pm j0.5024$	$.9410 \pm j0.0631$, $.9688 \pm j0.0312$

The step response data in Tables XXXIII and XXXIV and the plots clearly show that the integral controller not only reduces the steady state error, but also the settling time and amount of control effort needed as well. The settling time for the first controller is essentially the whole period of the simulation run, meaning that state, θ_y, sustained oscillations. Another observation from that set of data is that the effect of weighting appears to be greater when applied to states corresponding to rates rather than positions. Also, the effects of different weights appear to be determined by their absolute value as well as their values relative to one another.

The stiffness response tables showed pitch stiffness decreased for the integral controller for negative pitch commands but increased for pitch commands 0.0 to 1.0 degrees. For the entire input range the z-axis stiffness increased significantly.

Notable in the responses of the selected controllers to B-field perturbations in Table XXXV was the fact the spread of some of the performance criteria was an order of magnitude lower for the second controller. And again in this same table, the reduction in control current transients is quite obvious. However, one performance value that did not improve in this test with integral control was the pitch state settling time. This will be addressed below in the discussion of the plant responses.

The most important—and telling—results were the plant responses. Several observations are made below regarding these responses. The plant outputs confirmed the observations about performance made from the simulation results, particularly the plots in Figs. 84 and 85.

Consistency between model and plant performance tends to validate the model and the design method. Neither of the controllers tested on the plant would allow the pitch to be adjusted beyond 1.5 degrees, but this was anticipated due to the sensors' ranges and positions, which limited the maximum measurable rotation to 1.13 degrees.

Figs. 84 and 86 illustrate a fundamental design challenge for these experiments, namely applying a 2-DOF controller to a system that is marginally unstabilizable with two inputs. This condition was illustrated earlier in the in the chapter where the controllability analysis showed that the uncontrollable modes had an eigenvalues on the imaginary axis.

Note in Fig. 86 the growing oscillations in the nonlinear plant's x-position output. This instability is fed back to other states via the Taylor series expansion block, which continuously recalculates the B-field components and their first order gradients based on the instantaneous position output (See Fig. 69.). Because the y-axis torque, $\mathbf{T}_y$, is a function of $\mathbf{B}_z$, which is partially dependent on the x-position output through the Taylor series expansion, the instability seen in Fig. xx is also reflected in the nonlinear model's response to pitch commands. To a lesser degree, oscillations are also coupled to the model's z-position output, which is dependent through the $\mathbf{F}_z$ equation and its dependence on $\mathbf{B}_{xz}$. It is suspected these undamped oscillations did not pose problems during the controller synthesis process runs because none of the simulations ran for long periods. However, these oscillations are probably responsible for the longer pitch state settling time, refered to above.

This same phenomenon occurs in the real plant, where oscillations in the x-position slightly vary the electromagnetic profile within the suspended magnet core, in turn slightly changing electromagnetic torque and force on the object. From the torque equation

$$T_y = vM_x(-\theta_y B_x - B_z) \tag{42}$$

it can be seen that if a non-zero flux density gradient B_{zx} exists, which it does in the field pattern for the 2-DOF experiment, changes in the x-position will effectively change the value of B_z. This in turn alters the torque on suspended core. The controller was able to manage those perturbations in torque—at least for short periods, as predicted by the B-field perturbation tests. Fortunately, this problem was easily overcome in the plant by manually damping the test cylinder's movement in the x-direction.

Finally, the 2-DOF levitation inside the suspension chamber cleary demonstrated that the basic controller design for the external 2-DOF system would work without much interference due to

eddy currents inside the chamber walls. This tends to support a claim that effects from the stainless steel do not have to modeled in the controller design, although these results were not conclusive since there were only two magnets operating simultaneous in this experiment. When all ten are in operation, there may be a more substantial eddy current effect.

Conclusion

The chapter covered the rationale, theoretical background and execution of several 2-DOF MSBS experiments and completes a bridge from classical to optimal control started in Chapter V. The experimental results demonstrated the effectiveness of an optimally designed regulator and state estimator for the MIMO case and so met the goal of validating the modeling approach in the context of an actual MSBS sestup. In the course of performing the experiments practical implementation techniques were introduced and a set of semi-automated controller synthesis "tools" were developed that can be easily adapted to implementations with higher degrees of freedom.

CHAPTER VIII

CONCLUSIONS

Introduction

The intent of this thesis was to design and implement one and two-degree-of-freedom levitation systems while making a practical case for building the full-scale MSBS for the HRTF. Ultimately, the success of the project was shown in the results of the 1 DOF and 2 DOF experiements, reported in Chapters IV - VI. But leading up to the successful experiments were the completion of many supporting tasks. These included not only model validation and controller design, but development of practical engineering techniques encompassing such areas as sensor calibration and position measurement algorithms, live parameter adjustments for the control system, development of an operator GUI with safety mechanisms, and surge current reduction in the high current DC power supply.

Discussion of the Findings

Prior to the research supporting this thesis, there had been no validation (i.e., demonstrations using actual hardware and software components) of the assumptions and approximations upon which preliminary designs were based. (See [4].) This paper discusses a succession of successful levitations using increasingly thorough instantiations of the intended MSBS environment, concluding with a demonstration of 2-DOF control inside the stainless steel suspension chamber. For each experiment, the specific correspondence between that configuration's attributes and those of the full system were pointed out. For example, in Chapter IV the tie between experimental and full systems lay in the simplifying approximations used to model the system, in the sensor signal algorithm and the use of an actual coil and amplifier combination. Then in Chapter VI the correspondence was expanded to include multiple suspension coils with a three-dimensional magnetic field, MIMO controller, larger suspension gap and pitch measurement algorithms. The experimental results made two strong points in validating the design assumptions:

1) Controllers based on an analytic model of the MSBS performed as expected and responded to parameter changes according to the basic control theory;

2) The responses of the system comprising actual MSBS hardware and software very closely matched results of computer simulations based solely on the mathematical assumptions and approximations.

A second support for the practicality of building the full-scale MSBS was the demonstration of a particular robust controller design—an optimal regulator using state estimation—that was highly suitable to the equipment suite and configuration of the intended system. The experimental results showed the controllers met critical performance specifications: percent overshoot and steady state error. In addition, the controllers exhibited what appeared to be adequate stiffness in both the simulations and actual experiments. Also, the simulations demonstrated the optimal regulator's robustness with respect to uncertainty in modeling the magnetic flux density and its gradients. (This finding also supports the validation of various modeling assumptions.) Finally, of key practical importance, controllers were designed that met performance criteria without driving the control input to the plant (i.e., coil currents) to excessive levels.

The experiments included all but one of the major MSBS components:

1) custom-built suspension coils;
2) Copley 232P power amplifiers
3) Clinton power supply;
4) dSPACE interfaces, real time processor and software;
5) LA-511 sensors; and
6) The stainless steel suspension chamber.

The chilled water cooling system was not tested. Several essential system capabilities were shown, including the interoperability of the components, the adequacy of their ranges and tolerances and their suitability to support for long periods of uninterrupted operation without access to the interior of the suspension chamber. Of note, a close similarity was shown between the performance of different amplifiers and coils, removing the need to individually characterize each of one. Also, related to the control system design, it was confirmed that the dynamics of the Copley amplifiers did not require special compensation in the controller design. Lastly, the stainless steel chamber did not appear to require special design considerations in the controller due to phase shifts induced by eddy currents. However, it should be noted the experiments thus far employed only two, not ten magnets around the chamber.

The practical implementation techniques introduced in this work included a simplified voltage-to-distance conversion algorithm for the sensors, a workable means to compensate for sensor output variations, a method to adjust controller parameters during system operation, and a phased amplifier turn on procedure to guard against overloading the main power supply during system

startup. In addition, the estimation of magnetic field values using the program OPERA™ was shown to be adequate for fields from multiple coils. Lastly, a semi-automated process for designing MIMO controllers was developed using a set of Matlab scripts. This "tool set" will be essential in producing the higher degree-of-freedom controllers.

Follow-on Issues

Several major challenges remain in building the full 6-DOF MSBS. The first of these is reformulating the MSBS system equations and modeling the "X-configuration" of the suspension coils. This is an issue because of the need to operate two coils on opposite sides of the suspension chamber wired in series (see Fig. 2.). If the suspension point (origin of the inertial reference system) is not symmetric with respect to the coils, multiple calculations of magnetic flux density will be needed for the different coils. Also, the altered amplifier/coil dynamics presented by two suspension coils in series may require additional modeling.

Related to this, a second challenge will be revising the nonlinear system model for 6-DOF. For this two sub-tasks must be tackled. First, a means to produce the x-axis component of torque will have to be implemented. Fortunately, this is described mathematically in [12] and is based on the same assumptions and approximations validated in this paper. The second sub-task will be to devise a means to detect the rolling motion of the suspended object, preferably in a manner compatible with the eventual operational use of the system. A side note here is that the higher degree-of-freedom controller may require thousands more synthesis-validation iterations due to the larger number parameters, particularly the larger state and input weighting matrices.

A third remaining major design challenge is the investigation of controller operation through the stainless steel suspension chamber with the complete set of suspension coils. A concern first raised in Chapter II, this involves the expected roll-off in the magnitude of magnetic flux density caused by eddy currents produced in the suspension chamber walls induce an opposing counter magnetic field [5]. Although, the last experiment showed a 2-DOF controller could work without special modification through the stainless steel chamber walls, this only involved two coils, rather than the ten that will eventually be needed.

Concluding Remarks

Through the demonstration of a working, scaled-down system incorporating all the essential dynamics and hardware of the full system, the work supporting this paper removes the major uncertainties surrounding the MSBS design. Also, the engineering techniques and methods described herein contribute significant practical means to construct and operate an actual working system. Thus, in two respects, reducing the risk of further development as well as creating greater likelihood of success, this paper strongly contends from a practical standpoint for the completion of full-scale MSBS.

REFERENCES

[1] Philip Holmer, "Faster than a speeding bullet train," *IEEE Spectrum*, vol. 40, no. 8, pp. 30-34. Aug. 2003.

[2] Francis C. Moon, *Superconducting Levitation*. New York, NY: John Wiley and Sons, 1994, chapter 1.

[3] Colin P. Britcher, Oscar González, W. Steven Gray, Alexander Smits, Oscar Gomeiz, James E. Barkley, Adeel Jafri, "Design of a magnetic suspension and balance system for the Princeton/ONR High Reynolds Number Testing Facility," Proceedings from the Fifth International Symposium on Magnetic Suspension Technology, Jul. 2000, NASA/CP-2000-210291. pp. 675-686.

[4] Syeed Adeel Akhtar Jafri. "Studies related to the design and implementation of a magnetic suspension and balancing system." M.S. thesis, Dept. of Electrical and Computer Engineering, Old Dominion University, Norfolk, VA, 2000.

[5] James E. Barkley, Jr. "Eddy current analysis and modeling of five-degree-of-freedom magnetic suspension and balancing system," unpublished.

[6] V. Dale Bloodgood, Jr., "Implementation of a 1-DOF Magnetic Suspension controller Using dSPACE," unpublished.

[7] *National Consensus Standard for Configuration Management*, ANSI/EIA-649-1998, pp. v, 5-23.

[8] Beverly McKeon, James Allen, Alexander Smits, "The Princeton/ONR High Reynolds number Test Facility-HRTF," presented to APS Fluid Mechanics Division, New Orleans, Nov. 1999. Available: www.Princeton.edu/~gasdyn/Facilities/Facilities.html#HRTF.

[9] Peter Thomas Welch, Jr., "Design of a graphical user interface for a magnetic suspension and balance control system," unpublished.

[10] Nelson J. Groom, "Analytical model of a five degree of freedom magnetic suspension and position system," NASA Langley Research Center, TM-100671, Mar. 1989.

[11] Nelson J. Groom, "Expanded equations for torque and force on a cylindrical permanent magnet core in a large-gap magnetic suspension system," NASA Langley Research Center, TP-3638, Feb. 1997.

[12] Nelson J. Groom, "Simplified analytical model of a six-degree-of-freedom large-gap magnetic suspension system," NASA Langley Research Center, TM-112868, Jun. 1997.

[13] Nelson J. Groom, "A decoupled approach for magnetic suspension systems using electromagnets mounted in planar array," NASA Langley Research Center, TM-109011, Aug. 1993.

[14] Nelson J. Groom and Colin P. Britcher, "Open loop characteristics of magnetic suspension systems using electromagnets mounted in planar array," NASA Langley Research Center, TP-3229, Sep. 1992.

[15] Francis W. Sears, Mark W. Kamansky, Hugh D. Young, *University Physics*, Reading, MA: Addison-Wesley, 1982, p. 602.

[16] David E. Cox and Nelson J. Groom, Min-Hung Hsiao, Jen-Kuang Huang. "Modeling and identification of a large gap magnetic suspension system", Nasa Langley Research Center, Hampton, VA, Old Dominion University, Norfolk, VA, pp. 12-16.

[17] David K. Cheng, *Field and Wave Electromagnetics*, Reading, MA: Addison-Wesley, 1983.

[18] *dSPACE Real-Time Interface Implementation Guide for Release 3.4*, dSPACE GmbH, Paderborn, GE, May 2002.

[19] *dSPACE DS1103 PPC Controller Board Installation and Configuration Guide for Release 3.4*, dSPACE GmbH, Paderborn, GE, May 2002.

[20] *dSPACE MLIB/MTRACE Matlab-dSPACE Interface Libraries for Release 3.4*, dSPACE GmbH, Paderborn, GE, July 2001.

[21] *dSPACE Control Desk Experiment Guide for Release 3.4*, dSPACE GmbH, Paderborn, GE, May 2002.

[22] Timothy Schott, Thomas Jordan, Taumi Daniels, and Charles Alcorn, "Present status of the MIT/NASA Langley 6-Inch MSBS," presented at the International Symposium on Magnetic Suspension Technology, Hampton , VA, Aug. 1991.

[23] Gerhard Schweitzer, Hannes Bleuler, Alfons Traxler, *Active Magnetic Bearings*, Hochschulverlag AG an der ETH Zurich. 1994.

[24] *Technical Manual for LA-511*, SunX Trading Co.

[25] Nelson J. Groom and Philip R. Schaffner "An LQR controller design approach for a large gap magnetic suspension system (LGMSS)," NASA Langley Research Center, TM-101606, Jul. 1990.

[26] G. F. Franklin, J. David Powell, Michael L. Workman, *Digital Control of Dynamic Systems*, 3rd Ed., Menlo Park, CA: Addison Wesley Longman, 1998.

[27] *Matlab Control System Toolbox*, 2nd ed., The Mathworks, Inc., Natick, MA, 1996.

[28] Sigurd Skogestad, Ian Postlethwaite, *Multivariable Feedback Control*, Chichester, England: John Wiley and Sons, 1996.

[29] Thomas Kailith, *Linear Systems*, Upper Saddle River, NJ: Prentice-Hall, 1980.

[30] Kemin Zhou and John C. Doyle, *Essentials of Robust Control*, Upper Saddle River, NJ: Prentice-Hall, 1998.

[31] W. S. Levine, *The Controls Handbook*, 2nd Ed., CRC Press and IEEE Press, 1996, pp 759-778.

APPENDIX A

FUNCTIONAL AND PHYSICAL CONFIGURATION BASELINES

TABLE I-A
FUNCTIONAL CONFIGURATION BASELINE

Identifier	Functional Requirements	Related Design Decisions	Reference
FR1000	FR1000 Operate within the environmental and operational constraints of the HRTF		[3], [7]
FR1100	FR1100 Operate under 3500 PSI internal pressure.	Two-inch, nonmagnetic stainless steel chosen for construction of suspension chamber. Materials costs constrain inner diameter to approx. 19 in. Interiorly mounted components protected from pressure.	[3], [7]
FR1200	FR1200 Maintain airtight seal with HRTF under 3500 PSI internal pressure.		[3], [7]
FR1300	FR1300 Support Reynolds number in range of researcher's needs.	Cylindrical and uniform construction. Actuators mounted externally. Level of effort required to open suspension chamber limits accessibility. Interior mounted components must be small relative to chamber diameter and have aerodynamic shapes or covers.	[3], [7]
FR1310	FR1310 Employ test object supporting high Reynolds numbers.	Baseline design for test object calls for an ellipsoidal model with 12:1 length to diameter ratio. Baseline dimensions are 35.4 in. (0.9 m) L; 2.95 in. (0.075 m) dia. For simplicity and reduced power consumption, a permanent magnet chosen rather than magnetizing coils to create magnetic field in test object.	[3], [7]

TABLE I-A , CONT'D

Identifier	Functional Requirements	Related Design Decisions	Reference
FR1400	FR1400 Operate unattended and continuously for periods of several days without interior access to HRTF.	Robust sensor scheme selected. (See FR2210) Linear design selected for controller. (See FR2210)	[3], [7]
FR1410	FR1410 Accommodate changes in pressure, airspeed and test object position commands, including test object launch, without manual intervention.	Choose a conservative range of movement with respect to system's physical capabilities.	[3], [7]
FR1420	FR1420 Allow for calibration and testing of interior components without manual intervention.	Control program designed to compensate for changes in sensor input/output characteristics.	[3], [7]
FR1430	FR1430 Maintain safe internal temperature in electromagnets during long periods of unattended operation.	Electromagnets wound with insulated copper tubing with square cross-section. Circulating, chilled water cooling system designed	[3], [7]
FR1440	FR1430 Employ automated condition monitoring and emergency system shut-down.		[3], [7]
FR2000	FR2000 Automatically control position of suspended test object in six-degrees-of-freedom without direct mechanical contact.		[3], [7]
FR2100	FR2100 Actuate suspended test object in six degrees of freedom without direct mechanical contact, i.e., using electromagnets.		[3], [7]
FR2110	FR2110 Employ minimum of five independently controlled actuators (electromagnets).	Five separately controlled current amplifiers capable of supplying up to 120A DC at 150 V DC.	[3], [7], [12], [13]
FR2120	FR2120 Produce aggregate flux density of sufficient magnitude to create force adequate to counter both weight of test object and the aerodynamic forces acting on it.	Large, iron-core electromagnets custom designed and built for MSBS. So-called X configuration for electromagnets selected, where the magnets are mounted transversely to the suspension chamber and paired such that magnets on opposite sides of the suspension chamber share the same current source.	[3], [7]

TABLE I-A , CONT'D

Identifier	Functional Requirements	Related Design Decisions	Reference
FR2200	FR2200 Sense and provide test object position without direct mechanical contact.	Optical sensors selected for design.	[3], [7]
FR2210	FR2210 Measure deviation of test object from fixed point in an inertial coordinate system attached to the suspension chamber.	Selected sensors were Sunx LA-511 optical sensors with 15-mm range of detectable motion. Sensors selected establish range of controllable movement at approximately plus-or-minus 7.5 mm	[3], [7]
FR2211	FR2211 Convert raw sensor voltages to corresponding distances in form.		[3], [7]
FR2212	FR2212 Convert linear movements to angular movements		[3], [7]
FR2300	FR2300 Execute real-time, multivariable control in a way that HRTF users to readily change parameters and data capture schemes.		[3], [7]
FR2310	FR2310 Employ system that permits rapid prototyping.	dSPACE Controller Design and Implementation system selected to permit both design and operation on same hardware/software suite.	[3], [7]
FR2320	FR2320 Employ digital multivariable controller permitting real-time operator input.		[3], [7]
FR2330	FR2330 Convert actual system control and feedback signals to digital form compatible with controller platform.		[3], [7]
FR3000	FR3000 Provide real-time control, monitoring and data logging to MSBS operators.		[3], [7]

TABLE II-A
PHYSICAL CONFIGURATION BASELINE

Group	Identifier	Item Name	Qty	Item Description	Reference
MSBS Plant Group	1A1	Suspension chamber	1	Type 304L cylindrical stainless steel suspension chamber, nom. 24 in. O.D., nom. 19 in. I.D., 96 in. L.	[3], [7]
	1A1A1	Radial coil mounting assembly	1	Structural fiberglass frame used to mount radial suspension coils in X-configuration around suspension chamber	
	1A1A2	Radial suspension coil	8	Custom built iron-core magnets	
	1A1A3	Axial suspension coil	2	Water-cooled magnets wound around suspension chamber (axially).	
	1A2	Position sensor assy	1	Position sensors and mounting structure.	
	1A2A1	Sensor mounting frame	1	Non-magnetic frame holding position sensors in configuration to sense movement in 5 degrees-of-freedom.	
	1A2A2	Laser beam sensor set	5	Sunx Trading Co., Model LA-511	[3], [7]
	1A2A2A1	Laser emitter	5	Sunx Trading Co., Model LA-511P	
	1A2A2A2	Laser receiver	5	Sunx Trading Co., Model LA-511D	
	1A2A2A3	Laser sensor power supply	1	Leader Electronics Corp., DC Tracking Power Supply, Model LPS 152	
	1A2A2A4	Sensor cable junction box	1	Polystyrene junction box containing terminal strips needed to independently route sensor power and signal cables.	
	1A3	Magnet cooling system	1		
	1A3A1	Water chiller unit	1	Affinity Model	
	1A3A2	Chilled water manifold	6		
	1A3A3	Tubing assembly			

TABLE II-A, CONT'D

Group	Identifier	Item Name	Qty	Item Description	Reference
MSBS Plant Group	1A4 [a]	Test model assy. no. 3	1	Cylindrical PVC test model,30 in. L, 3.5 in. dia	
	1A4A1	Permanent magnet core	6	Neodymium-Boron-Iron disks, 3.0 in. dia., 0.5 in. thickness.	
	1A4A2	Magnet spacers	5	Wooden spacers between magnetic disks, 3.0 in. dia., 0.5 in. thickness.	
Amplifier Group	2A1	Electromagnet Drive Current Amplifier	6	Copley Controls Corp Model 232P PWM Power Amplifier	
	2A1A1	Amplier control panel	6	Copley Controls Corp Model 265P Display Panel Assy	
	2A2	Current amplifier power supply	1		
	2A2A1	High voltage/high current power supply	1	Clinton, Model S600/150S	
	2A2A2	Power supply step-start assy		Step start switch for Clinton power suppl	
	2A2A2A1	Isolation transformer		Tripp Lite, Model IS-500 500 KVA isolation transformer.	
	2A2A2A2	Soft-start CCA		Timed charging circuit card assy.	
Control System Group	3A1	Host Computer	1	Dell Precision PWS330, Intel Pentium 4 CPU, 523 KB RAM	
	3A2	Real Time Interface	1	DS-1103 Real Time Interface	
	3A3	Signal interface panel	1	CP-1103 Interconnection Module	

TABLE II-A, CONT'D

Group	Identifier	Item Name	Qty	Item Description	Reference
Software Group	4A1	Host Computer Operating System	1	MS Windows 2000, v 5.00.2195 Service Pack 1	
	4A2	Matlab Software Suite	1	Control system design software.	[27]
	4A2A1	Matlab R12.1 (inc. Simulink)	1	Matlab R12.1 (inc. Simulink)	[27]
	4A2A2	Matlab Control System Tool Box, v5.1	1	Matlab Control System Tool Box, v5.1	[27]
	4A3	dSPACE control system software suite	1	dSPACE v3.4	[18]
	4A3A1	Real Time Interface	1	Interface btwn Real Time Interface (RTI) and Simulink, v4.3	[18], [19]
	4A3A2	Control Desk Interface	1	Interface btwn Control Desk and Simulink, v1.0	[21]
	4A3A3	MLIB/MTRACE Interface to Matlab, v4.4	1	MLIB/MTRACE to Matlab interface libraries, v4.4	[20]
	4A3A4	dSPACE Control Desk, v2.2	1	dSPACE Control Desk, v2.2	[21]

[a] Test model presented here is the last one used in the set of experiments covered in this thesis and the one that most closely represents the model anticipated during actual operations.

APPENDIX B

MSBS OPERATING PROCEDURES

Introduction

This Appendix provides the following for the operation of the two 2-DOF experiment:

1) A brief description the MSBS operating environment.

2) A step-by-step system turn-on procedure.

3) A procedure for changing the controller parameters while the MSBS is operation.

Operating Environment

This section presents the primary software and hardware components of the MSBS and briefly describes their interaction. This background is intended to help MSBS operators prepare the system for use and to better understand its operation.

Software Architecture

Table I-B, extracted from the complete physical baseline in Appendix A, lists the software installed on the host computer used to operate the MSBS. Note the version numbers apply to the system configuration at the time the research supporting this thesis was completed.

TABLE I-B
MSBS OPERATIONAL SOFTWARE COMPONENTS AND VERSION NOS.

Item Name	Item Description
Host Computer Operating System	MS Windows 2000, v 5.00.2195 Service Pack 1
Matlab Software Suite	Scientific and engineering mathematics software. Includes the following: Matlab R12.1 (w/Simulink) Control System Toolbox, a specialized add-on functionality for Control System design
dSPACE Software suite	dSPACE v3.4 Control System Developer's Software. Includes the following: Real Time Interface (RTI) v4.3, interface between dSPACE and Simulink dSPACE Control Desk, Developer Version, v2.2 Control Desk Interface, v1.0, interface btwn Control Desk and Simulink MLIB/MTRACE Interface, v4.4, dSPACE-to-Matlab interface libraries

In addition to the software applications, the operating environment includes certain data and configuration files created during the development of the controller. MSBS operation requires

the files listed in Table II-B. (Only the main ones are listed.) Because the MSBS "application" is operated via Control Desk, the MSBS operational environment—at least in terms of the host computer's file structure—is organized around the Control Desk experiment. This is the basis for the folder structure illustrated in Table II-B. Reference [21] contains detailed information on the dSPACE file types and their roles in designing and operating an experiment.

TABLE II-B
PRIMARY DATA AND CONFIGURATION FILES USED DURING MSBS OPEARATION

Folder/File Name	Description
Primary Folder Name	This is the Working Root directory, set up by the user, where the main Control Desk (CD) experiment file resides, e.g., "2-DOF Integral Ctrl"
ExperimentName.cdx	Main experiment file, which contains links to all other files associated with experiment. User assigns name when creating experiment; name may or may not be same as the name of the Simulink model file.
InstrumentPane1_1.lay	File containing Control Desk layout window, comprising designer-selected set of instruments linked to the corresponding variables in the Real Time Processor (RTP). An experiment may include several layout windows.
ExperimentName.cdc	Control Desk connection file, which contains information on the data connections between the instruments and the real time hardware.
SimulinkModelName.mdl	Simulink model of the controller. This is compiled by the Real Time Interface (RTI) software to create the real time application.
SimulinkModelName.ppc	File containing the executable instructions to run the real time application (MSBS controller) on the Real Time Processor, Power PC 604e processor. (Applies to dSPACE systems using the DS-1103 RTI.)
SimulinkModelName.map	File generated by the linker; maps symbolic variable names to physical addresses.
SimulinkModelName.trc	ASCII file that contains descriptions of the variables in the real time application (MSBS controller).
SimulinkModelName.sdf	System description file that describes files to be loaded to individual components of the Real Time Processor. This file is generated when the along with the .trc file when the RTI software compiles the Simulink model. This file is also used when establishing links between instruments and variables.
UserName.par	Contains variable names, descriptions and values; used when changing values of variables while application is running.
P\param_change.m	Matlab script that invokes MLIB/MTRACE functions for live update of parameters used for sensor signal processing.
quick_con.m	Matlab script containing parameters and instructions to derive LQG regulator. Assumes values for state and input weighting matrices and noise covariance matrices are known.

TABLE II-B, CONT'D

Folder/File Name	Description
Two_DOF_Plant.m	Matlab script containing parameters and calculations necessary to derive linear model of 2-DOF plant and thus parameters needed to synthesize state estimator and state variable feedback gain constants. Script called by quick_con.m.
lqsim.m	Matlab script that implements Pincer's method when calculating the linear state variable feedback matrix K. Script called by quick_con.m.
SimulinkModelName_rti1103	This is a sub-directory to the working root directory; produced when the Simulink model is compiled; its files contain segments of code to be loaded on the real time processor.

Besides the Control Desk files, Table II-B also includes the Simulink model of the controller and the Matlab scripts needed to derive the variables needed by the Simulink model and place them on the Matlab workspace. Though not strictly needed to operate the MSBS once the Simulink model is compiled, these files are left here for convenience since the system is still under development. Having the files handily grouped along with the Control Desk experiment makes it easier to refine and recompile the controller model or add variables to the operator's screens.

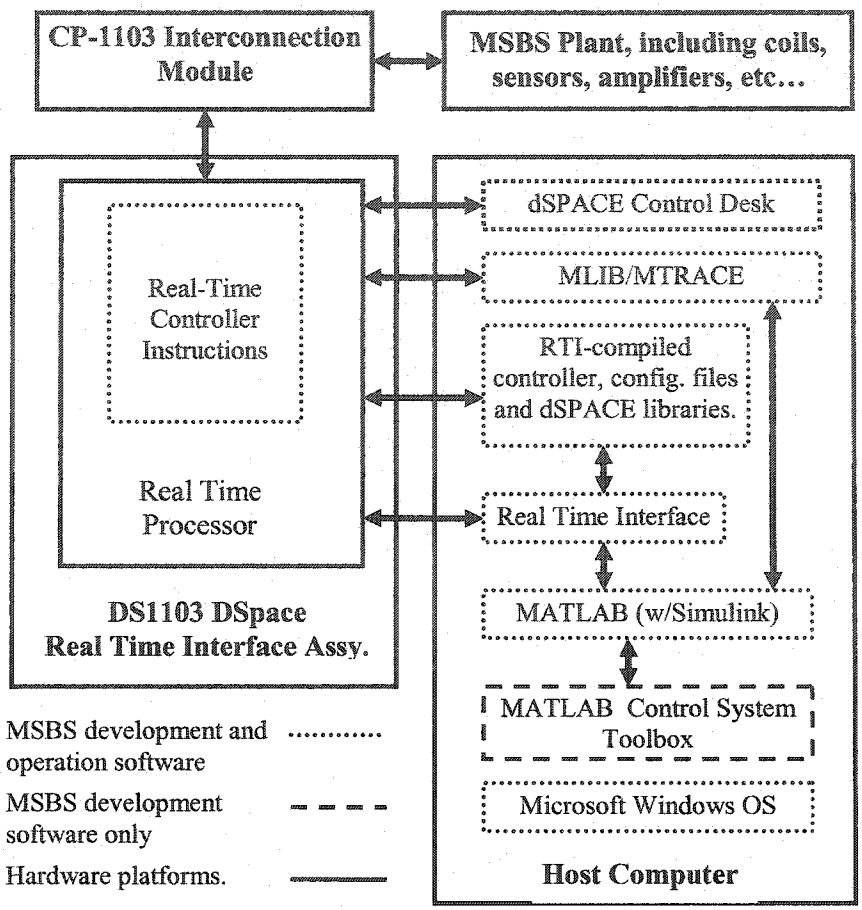


Fig. 1-B. MSBS Software

Fig. 1-B depicts the set of operational software components and their relationship to the hardware platforms they run on. Operator control is executed primarily using the dSPACE Control Desk program on the host computer. This program interacts with the real time application (compiled controller instructions) running on the real time processor, located on the DS-1103 Real Time Interface assembly. Note this piece of software code is not listed in Table IF because it is created when the MSBS is initialized and continues to exist only while the system remains in operation.

Hardware Configuration

Fig. 2-B illustrates the hardware configuration employed during MSBS operation. Although this figure describes the complete 6-DOF system, except for the quantities of certain components, it also applies to the 2-DOF configuration. The legend corresponds to the equipment groupings in the physical baseline in Appendix A.

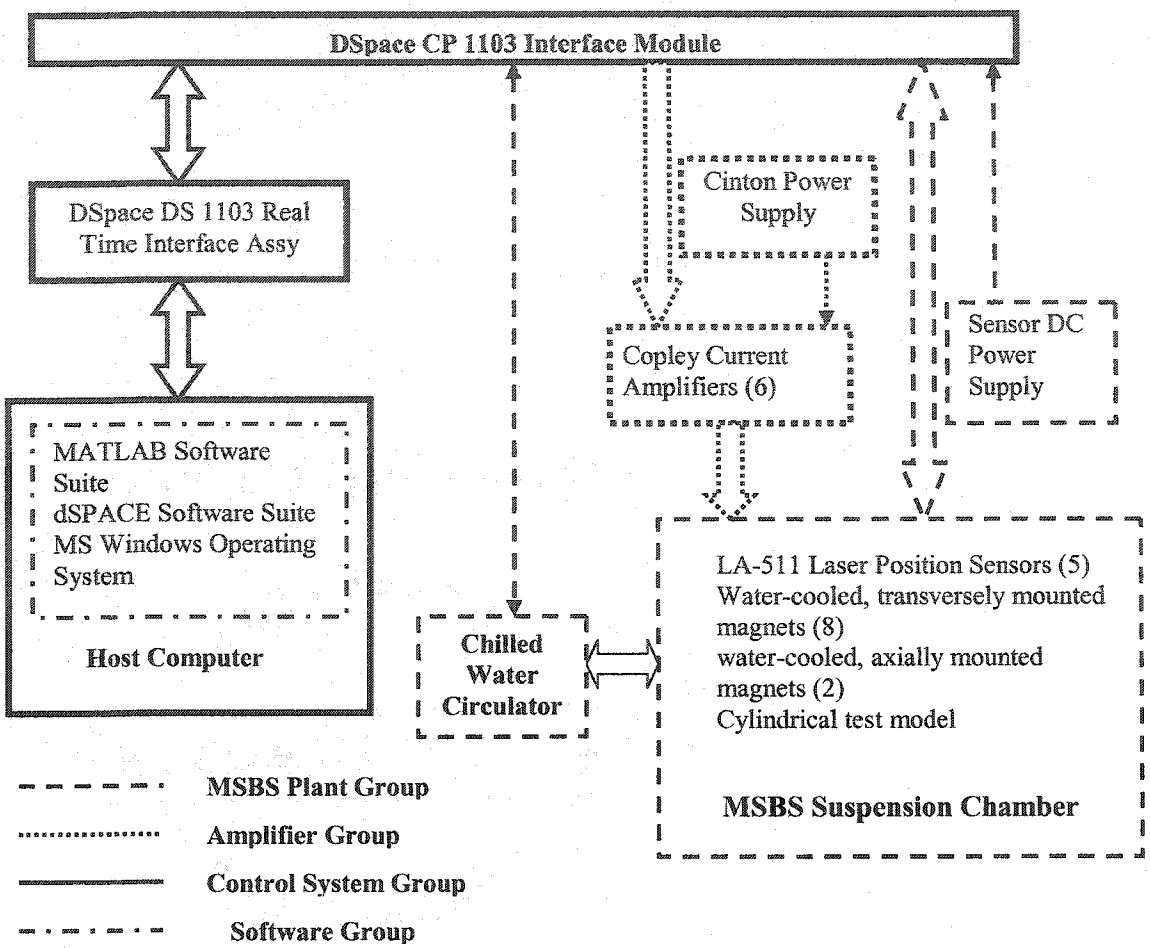


Fig. 2-B. MSBS Basic Hardware Block Diagram

Basic System Turn On Procedure

This section describes how to prepare and place the MSBS into normal operation. The procedure assumes that the control system has been modeled in Simulink, the model has been compiled and the operator control screens have been designed and properly linked to the correct variables and that the amplifier and sensor control systems have been tested. Note also the procedure refers to the system configuration in use at the completion of the work supporting this thesis.

Procedure

Table III-F provides the step-by-step procedure to place the MSBS into normal operation.

TABLE III-B
BASIC SYSTEM TURN-ON PROCEDURE

Step No.	Action
1	Remove watches, wallets, magnetically encoded cards and other items to safe place away from the suspension coils and permanent magnet core.
2	Place test object containing Neodymnium-Iron-Boron magnetic core at the bottom of the test frame. Ensure core is oriented so that the its magnetization vector is pointing in the positive x direction of the MSBS inertial reference frame.
3	Energize external dSPACE DS-1103 Interface Module.
4	Energize DC power supply for sensors and set output for approximately 15 volts.
5	Boot host computer and log on as MSBS user.
6	Launch Matlab and ensure Real Time Interface banner is displayed indicating the dSPACE software components are installed. Set current directory to the primary MSBS operating folder (see Table IIF).
7	Launch the DSpace Control Desk application. On the menu bar, select File, then Open Experiment. In the Open Experiment window that appears, click on the desired experiment name from within the MSBS operating folder. Then click open to launch Control Desk experiment. Control Desk will then open the experiment, gather all the associated files and present the operator control screen in use when the experiment was last saved. The experiment will come up in the edit mode.
8	From the menu bar select Platform, Application and then Load Application. One or more .sdf files will be listed; select the one with the same name as the Simulink model used to create the controller system (not necessarily the same name as the experiment). Click Open to start the real time controller application running in the background. Note: if error message appears indicating platform connection does not exist, perform step 7. Otherwise, proceed to step 8.
9	From the menu bar, select Platform, then Initialization, then Register. In the Register board window that appears, select DS1103 PPC Controller Board from the drop down menu in the Type field. Then click on Register. This will establish the Control Desk connection with the DS-1103 RTI module, i.e., the platform.

TABLE III-B, CONT'D

Step No.	Description
10	From the top tool bar (right below the menu bar), select Animation mode to activate the links between the instruments and the application already running in the background. At this point instruments will respond to the external signals received via the CP-1103 module. For example, there will be voltage readings for the laser sensors.
11	Visually check that the sensors' fields of view are completely unobstructed and that the LEDs on the receiver units also indicate good alignment with beam emitters. If the LEDs show poor alignment, the receiver units should be physically aligned to remedy the condition.
12	Note along the bottom of the operator screen several tabs with the names of the different operator screens. Select the Sensor menu, which contains large displays of the sensor output voltages. Adjust "span" pot on sensor receivers so that the sensor outputs are 5.0 volts for the unobstructed beams.
13	Ensure the Master Amplifier Enable switches on all control screens are in the off position. Return to the main operator control screen and place Amplifier On/Off switches in the off position and the Controller Cut-out switch in the out position. (Indicator "LEDs" on the screen will be dark.)
14	Ensure the Amplifier Inhibit switches on the front panel of the Copley amplifiers are in the out position (not depressed).
15	Turn on main breaker for Clinton DC power supply.
16	Energize the isolation transformer assembly inside the Copley amplifier rack.
17	Press and hold the green "soft-start" switch at the top of the amplifier rack until the Clinton power supply comes on and its cooling fan starts running. At this point the Copley power amplifiers should be energized.
18	Release the Inhibit buttons on Copley amplifiers and note on the operator control screen that the Amplifier Ready indicators are green. At this point the coil currents should be zero.
19	On all control screens, click on the Amplifier Master Enable switches and note the indicators change color to green.
20	Return to the main operator screen, click on the amplifier enable switches and observe that the coil current graphs display the equilibrium current value.
21	Click on the Controller Cut-out button to switch in the controller. Note the indicator changes color. At this point the controller is ready to begin driving the suspension coils once the suspended model is placed within several millimeters of its equilibrium position.
22	Launch the model ensuring the end marked positive is in the positive x direction of the reference coordinate system. CAUTION: suspension coil leads are at approximately 150 Volts DC—do not touch!

Live Parameter Update

This section describes a procedure by which MSBS operators can change the values of parameters within the MSBS control system while the MSBS is in operation. Further, the procedure also saves the modified parameters and loads them next time the system is initialized.

This ability is provided through the regular the MSBS operating software suite in conjunction with the MLIB/MTRACE software and Matlab. The particular implementation described here concerns parameters used by sensor system to convert the amount of light received by the sensors to the corresponding displacement from equilibrium. However, the same technique can be employed for other values.

Background

The MSBS sensor system contains several laser emitter and receiver pairs aligned so that when the suspended object is in its desired equilibrium position, the laser beams between each emitter and receiver are partially blocked. Because the receivers produce an output voltage proportional to the amount of beam they receive, the test object's deviation from its equilibrium position can be determined by how much the various receiver outputs vary. These outputs are converted to control system feedback by the MSBS application running on the dSPACE Real Time Processor.

The algorithm to convert receiver output voltages to the suspended object's displacement relies on the linearity of the laser receiver's input-output response. The algorithm uses an estimate of the slope of that response to relate the output voltage to the width in millimeters of the beam received. For example, the nominal sensor output range is from 1.0 to 5.0 volts as the amount of beam received varies from 0 mm to 15 mm (100%). Assuming a linear response, this gives a slope of 4.0 volts per 15 mm, or .26667 volts/mm. When an object blocks the beam and the sensor pair produces an output of 3.0 V, the object's position—relative to the beam edge—can be estimated as follows: 3.0 V – 1.0 V = 2.0 V equals the output due to beam reception. Thus the object's position is 2.0 V/.26667 V = 7.5 mm from the edge of the beam, or half way through the beam. .

Given the need for accurate values for the sensors' outputs, the motivation to implement some kind of parameter correction—while the system is operating—arises from two factors. First, sensor output voltages can change over time as well as vary between sensor pairs. Secondly, because the sensors will essentially be inaccessible in the completed system, the only practical way actually check the sensor outputs is to use the MSBS application itself. Thus, it is important to be able to update the value of the slopes and midpoints while the MSBS application is running.

Live Update Procedure

The procedure for updating and saving parameters consists of two separate activities. One is the set of Matlab commands used in conjunction with MLIB/MTRACE software to read the values of certain variables in the real time application, perform calculations using those values and then

modify the values of other values in the real time application based on the calculations. However, to ensure the revised values are employed the next time the MSBS application is started, Control Desk must be specifically configured. Table IV-B contains the step-by-step procedure, adapted from [20] and [21], to adjust the sensor voltage conversion algorithms in the MSBS sensor system. It will be seen the procedure essentially fall into two stages, set up and operation. Note the procedure refers to the system configuration in use at the completion of the work supporting this thesis.

TABLE IV-B
LIVE PARAMETER UPDATE PROCEDURE

Step No.	Action
1	Generate a parameter file as follows: While experiment is running on real time processor (RTP), select *Edit* mode from the upper tool bar. On menu bar, select Parameter Editor, then Generate Parameter File and then type in the name of parameter file, e.g., *sensor_params.par*. Click *Save.*
2	Designate selected parameter file for automatic, repeated use as follows: Select tab labeled SimulinkModelName.sdf at bottom of tool window (below instrument panel). Right click on the parameter file (e.g., *sensor_params.par*) in the left side of the tool window. Select **Declare Status Set**.
4	Right click again on same parameter file and select **Add to Experiment**. The parameter file should then be listed in the Experiment Navigator window on the left side of the screen.
5	In the Experiment Navigator window, right click on parameter file (e.g., *sensor_ params.par*) and select **Open on Experiment Load**.
7	Right click again on parameter file and select Use on Start Animation.
8	From menu bar select Parameter Editor—Automatic Update Parameters. Save experiment.
9	Live Parameter Update: While the experiment running, MATLAB window and set default folder to the root directory of the experiment.
12	Execute MATLAB script file *param_change.m*.
10	Save Updated Parameters as Defaults: In Control Desk, select Edit mode. In left-hand sided of Experiment Navigator window, right click on parameter file (e.g., *sensor_params.par*) and select close.
12	On menu bar, select **Parameter Editor—Generate Parameter File** and select same file name designated for this experiment. Click **OK** to overwrite existing version of file.
13	To ensure updated parameters are loaded as defaults for next experiment, upon conclusion of current experiment session, terminate the experiment on the RTP.

APPENDIX C

SELECTED DATA AND DATA ANALYSIS

This appendix contains selected measurement data and results of data analysis obtained during the experiments covered in this thesis.

Table I-C presents the magnetic flux density measurements taken in preparation for the 1-DOF experiments. The setup included two separate MSBS amplifier and suspension coil combinations, and measurements were taken using an F.W. Bell Series 9900 Gaussmeter in a vertical position to a moveable jig calibrated in millimeters. The tip of the probe, held in an upward position in the jig, was placed directly below the center of the core of the suspension coil and moved in ten mm increments from a distance 11 mm below core to a point 291 mm below the core. The z-component of the B-field was recorded at each increment. Three sets of measurements were taken, two for the first coil and one for the second.

TABLE I-C
Z-AXIS MAGNETIC FLUX DENSITY MEASUREMENTS FOR TWO COIL/AMPLIFIER PAIRS

Distance below coil (m)	Field measurements for Coil #1/Amp #1 (mT) [a]	Field measurements for Coil #1/Amp #1 (mT) [b]	Field measurements for Coil #2/Amp #2 (mT) [c]
-.011	-85.13	-83.21	-83.03
-.021	-83.71	-81.97	-81.74
-.031	-81.12	-79.85	-79.38
-.041	-77.79	-76.69	-76.38
-.051	-73.71	-73.07	-72.57
-.061	-69.29	-68.81	-68.66
-.071	-64.67	-64.49	-64.00
-.081	-60.00	-59.90	-59.71
-.091	-55.31	-55.59	-55.14
-.101	-50.95	-51.33	-50.98
-.111	-46.94	-47.37	-46.89
-.121	-43.13	-43.52	-43.09
-.131	-39.56	-39.98	-39.67
-.141	-36.38	-36.79	-36.43
-.151	-33.43	-33.82	-33.47
-.161	-30.76	-31.14	-30.74

TABLE I-C, CONT'D

Distance below coil (m)	Field measurements for Coil #1/Amp #1 (mT) [a]	Field measurements for Coil #1/Amp #1 (mT) [b]	Field measurements for Coil #2/Amp #2 (mT) [c]
-.171	-28.33	-28.64	-28.43
-.181	-26.08	-26.44	-26.08
-.191	-24.10	-24.32	-24.18
-.201	-22.28	-22.50	-22.23
-.211	-20.61	-20.80	-20.54
-.221	-19.09	-19.28	-19.08
-.231	-17.72	-17.82	-17.67
-.241	-16.50	-16.60	-16.47
-.251	-15.34	-15.41	-15.28
-.261	-14.27	-14.38	-14.20
-.271	-13.32	-13.45	-13.27
-.281	-12.47	-12.56	-12.39
-.291	-11.75	-11.75	-11.63

[a] For coil current in range 39.7-40.0 A. Magnetic Flux Density, Bz, oriented downward, or negative z direction. Same for all measurements.
[b] For coil current in range 39.7-40.0 A. Second of two sets of readings for this coil/amp combination.
[c] For coil current in range 39.6-39.9 A.

Chapter IV of the thesis describes an experiment performed to validate the assumption that magnetic flux density and its gradients can be approximated using the relationship

$$B = \frac{I}{I_{\max}} b_{\max}, \tag{65}$$

where the I is a variable representing the instantaneous control current in a given suspension coil, I_{max} is a constant denoting the maximum coil current allowed and b_{max} denotes the magnetic field produced by I_{max}. In this experiment the values of the B-field and its gradients calculate for I = 15.74 A were compared with those quantities when measured directly with coil current 15.74 A. Figs. 1-C and 2-C. illustrate the fact the approximation is quite close.

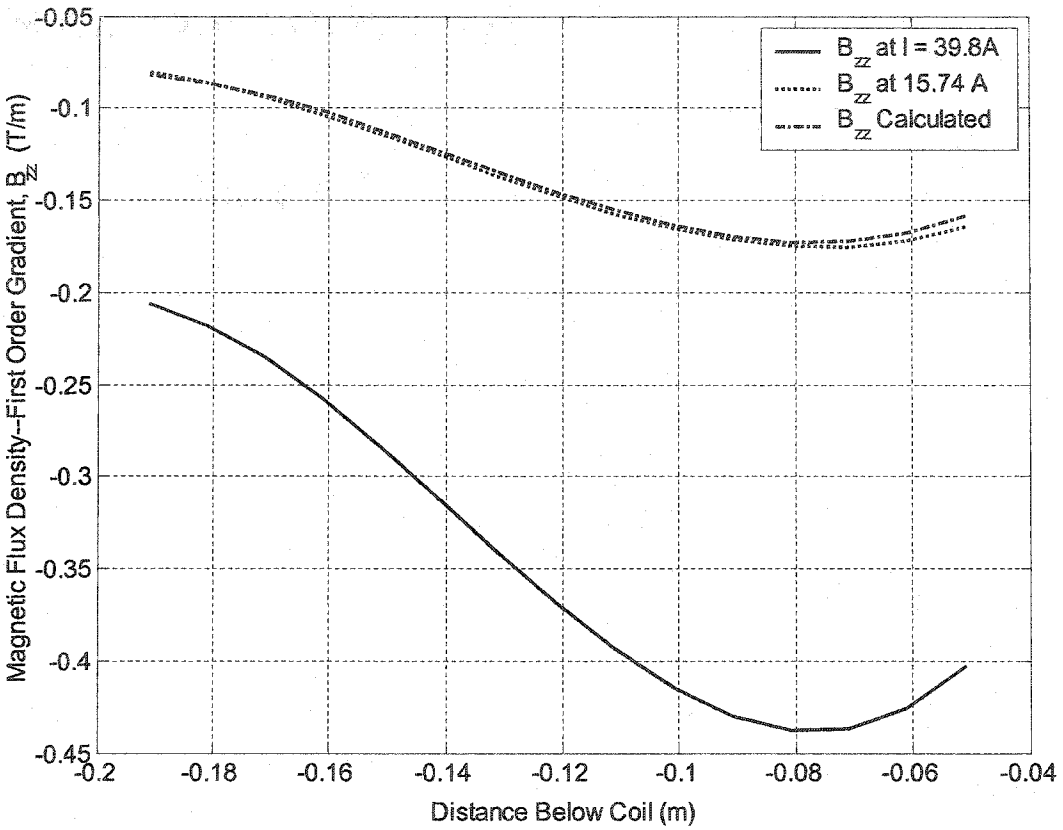


Fig. 1-C. Comparison of first order magnetic flux density gradients obtained by measurement and calculation for I = 15.74 A.. Also shown is gradient for I = 39.8 A.

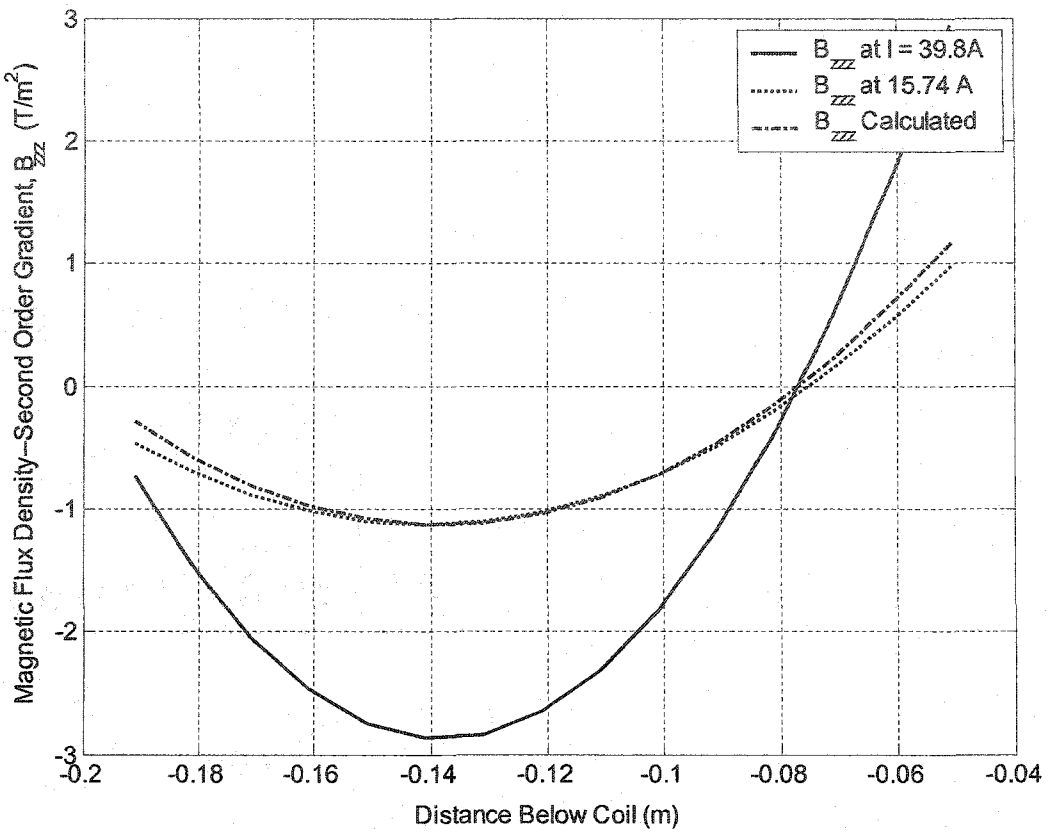


Fig. 1-C. Comparison of second order magnetic flux density gradients obtained by measurement and calculation for I = 15.74 A.. Also shown is gradient for I = 39.8 A.

The last part of this appendix is a table containing the results of the calculations of the magnetic flux densities and their gradients performed by the finite element analysis program Opera.

TABLE II-C
MAGNETIC FLUX DENSITY CALCULATIONS FOR 2-DOF EXPERIMENTS

Field or Gradient	Values calculated in Gauss using inches		Values calculated in Tesla using meters	
	Coil A [a] (Gauss)	Coil B (Gauss)	Coil A (Tesla)	Coil B (Tesla)
Bx	-223.25	223.25	-0.0223	0.0223
By	0	0	0	0
Bz	-117.07	-117.07	-0.0117	-0.0117
Bxx	-20.23	-20.23	-0.0796	-0.0796
Bxy	0	0	0	0

TABLE II-C, CONT'D

Field or Gradient	Values calculated in Gauss using inches		Values calculated in Tesla using meters	
	Coil A [a] (Gauss)	Coil B (Gauss)	Coil A (Tesla)	Coil B (Tesla)
Bxz	-37.5	37.5	-0.1476	0.1476
Byy	21.26	21.26	0.0837	0.0837
Byz	0	0	0	0
Bzz	-33.13	-33.13	-0.1304	-0.1304
Bxxx	2.447	-2.447	0.3793	-0.3793
Bxxy	0	0	0	0
Bxxz	-6.956	-6.956	-1.0782	-1.0782
Bxyy	2.025	-2.025	0.3139	-0.3139
Bxyz	0	0	0	0
Bxzz	-6.523	6.523	-1.0111	1.0111

[a] In 2-DOF experiments, coils lie along x-axis. Coil A is lies in the positive direction with respect to Coil B.

APPENDIX D

SELECTED MATLAB SCRIPTS

Introduction

This appendix contains various Matlab scripts used throughout the experiments conducted in support of this thesis. The scripts were selected based on their analysis or functional implementation of noteworthy contributions to the development of the MSBS.

Miscellaneous Matlab Scripts Used in Preparation for 1-DOF Maglev Experiments

The scripts in this section are those used to analyze the MSBS experimental setup with respect to its physical parameters such as the input-output relationship for the laser sensors and the analysis of the magnetic flux densities for the MSBS coils.

The first script is the one used to analyze LA-511 laser sensors to determine a reliable algorithm for converting sensor voltage to distance that could be easily updated according changes in sensor characteristics.

Sensor_measurements.m

```
%Sensor voltages vs. Width of Beam Received

format short;
s=5;  %number of sensors
n=5;  %number of sample sets
slopes=zeros(n,5);
x=(0:1:15);
Sensor_outputs=zeros(n,16,5);

%Sensor 1 Readings
Sensor_outputs(:,:,1)=...
[1.00 1.15 1.39 1.66 1.96 2.22 2.56 2.83 3.12 3.40 3.70 4.00 4.30 4.56 4.78 5.01;...
1.00 1.16 1.41 1.65 1.94 2.24 2.55 2.82 3.12 3.43 3.74 4.02 4.31 4.57 4.83 5.01;...
1.00 1.16 1.41 1.68 1.98 2.25 2.53 2.83 3.14 3.42 3.76 4.03 4.31 4.55 4.82 5.01;...
1.00 1.16 1.41 1.67 1.96 2.24 2.54 2.84 3.14 3.43 3.75 4.04 4.32 4.58 4.83 5.01;...
1.00 1.17 1.41 1.70 1.96 2.25 2.56 2.83 3.13 3.45 3.77 4.05 4.31 4.57 4.82 5.01];
%Sensor 2 Readings
Sensor_outputs(:,:,2)=...
[0.99 1.09 1.31 1.55 1.85 2.14 2.38 2.59 2.88 3.19 3.39 3.66 3.92 4.14 4.37 4.60;...
0.99 1.09 1.32 1.55 1.87 2.09 2.33 2.55 2.84 3.14 3.38 3.66 3.90 4.14 4.37 4.59;...
0.99 1.08 1.34 1.57 1.84 2.07 2.35 2.59 2.84 3.11 3.35 3.63 3.90 4.12 4.35 4.60;...
0.99 1.10 1.34 1.55 1.81 2.08 2.37 2.61 2.85 3.13 3.37 3.65 3.89 4.14 4.36 4.59;...
0.99 1.09 1.30 1.53 1.75 2.07 2.32 2.55 2.83 3.11 3.38 3.63 3.86 4.12 4.35 4.58];
%Sensor 3 Readings
Sensor_outputs(:,:,3)=...
[1.05 1.28 1.56 1.84 2.10 2.39 2.69 3.00 3.28 3.57 3.87 4.17 4.42 4.67 4.89 5.01;...
1.03 1.28 1.57 1.84 2.13 2.41 2.69 2.99 3.30 3.61 3.90 4.19 4.43 4.66 4.91 5.02;...
1.03 1.31 1.59 1.87 2.11 2.40 2.69 3.01 3.29 3.57 3.89 4.16 4.43 4.67 4.90 5.02;...
1.02 1.29 1.56 1.85 2.11 2.41 2.70 3.01 3.31 3.60 3.85 4.20 4.43 4.67 4.90 5.02;...
1.05 1.30 1.57 1.85 2.12 2.40 2.71 3.01 3.30 3.59 3.87 4.19 4.44 4.68 4.91 5.02];
%Sensor 4 Readings
Sensor_outputs(:,:,4)=...
[1.00 1.18 1.37 1.61 1.91 2.20 2.45 2.70 3.05 3.35 3.64 3.93 4.23 4.48 4.77 4.99;...
1.00 1.19 1.39 1.62 1.89 2.18 2.49 2.75 3.05 3.36 3.66 3.97 4.24 4.52 4.75 4.99;...
1.00 1.17 1.43 1.62 1.94 2.21 2.47 2.76 3.06 3.38 3.66 3.97 4.24 4.50 4.75 4.99;...
1.00 1.16 1.41 1.64 1.93 2.18 2.48 2.80 3.05 3.36 3.68 3.93 4.24 4.52 4.76 5.02;...
```

```
1.00 1.16 1.38 1.64 1.88 2.22 2.49 2.77 3.06 3.38 3.64 3.96 4.24 4.51 4.76 5.01];
 %Sensor 5 Readings
Sensor_outputs(:,:,5)=...
[1.00 1.18 1.36 1.61 1.88 2.12 2.33 2.60 2.86 3.14 3.43 3.65 3.90 4.12 4.31 4.45;...
1.00 1.14 1.36 1.65 1.87 2.11 2.39 2.62 2.89 3.12 3.42 3.63 3.89 4.10 4.34 4.45;...
1.00 1.17 1.37 1.64 1.89 2.14 2.39 2.64 2.93 3.16 3.43 3.67 3.92 4.14 4.36 4.45;...
1.00 1.17 1.40 1.64 1.91 2.14 2.40 2.63 2.93 3.17 3.42 3.68 3.90 4.14 4.37 4.45;...
1.00 1.18 1.39 1.65 1.89 2.14 2.42 2.65 2.91 3.16 3.42 3.66 3.87 4.15 4.37 4.45];

%Calculate sample mean for each sensor at each position (i.e., 0 mm, 1 mm, ... 15 mm)
for i= (1:1:n)
    for j= (1:1:16)
        sample_mean(i,j) = mean(Sensor_outputs(:,j,i));
    end
end

%Calculate slopes of sensor responses based on 2nd and 15th points in sensor ranges
for i= (1:1:n)
    slope(i)=(sample_mean(i,15)-sample_mean(i,2))/13;
end

%shorten x-axis by eliminating end-points (i.e., 0 and 16)
for i=(2:1:15)
    xlinear(i-1)=x(i);
end

%Calculate linear regression parameters for shortened ranges

% for each sensor based on sample means

a1=zeros(5,1); a2=zeros(5,1); a3=zeros(5,1); a4=zeros(5,1);
A=zeros(2,2,5);
C=zeros(2,5);
X=zeros(2,5);
for i=(1:1:s)       %create sample mean array for shortened range
    for j=(2:1:15)
        sm_linear(i, j-1)=sample_mean(i,j);
    end
end
for i=(1:1:s)
    a1(i) = sum(xlinear.^2);
    a2(i) = sum(xlinear);
    a3(i) = sum(xlinear);
    a4(i) = 14;
    C(1,i) = xlinear*sm_linear(i,:)';
    C(2,i) = sum(sm_linear(i,:));
    A(:,:,i)=[a1(i) a2(i);a3(i) a4(i)];
    X(:,i)=inv(A(:,:,i))*C(:,i);
end

%Calculate best fit using linear regression parameters
for i=(1:1:s)
    linear_approximation(i,:)=(X(1,i)*xlinear)+X(2,i);
end

%Using the sensor outputs corresponding to the shortened x-axis as described above
%(i.e., minus points 0 and 16), calculate the percent of the total sensor output range
%obtained by this shortened x-axis.
%E.g., say for sensor 1 at point 15 (14 mm), the sample mean for all sets of output
readings
%is 4.812 and at point 2 the sample mean is 1.16.  This corresponds to percentages
0.9537
%and 0.0447 respectively of sensor 1's total output range.

%To simplify calculations, the average high and low percentages of all sensors are
calculated.
%Then these two values are used below to calculate sensor outputs for the truncated
ranges. If
%the fit of these calculated outputs to the linear regression plot is acceptable, then
this
```

```
%technique may be useful for updating the sensor input-output slopes used in the
actual
%dSpace application

for i=(1:1:s)
    high_percent(i)=(sample_mean(i,15)-sample_mean(i,1))/(sample_mean(i,16)-
sample_mean(i,1));
    low_percent(i)=(sample_mean(i,2)-sample_mean(i,1))/(sample_mean(i,16)-
sample_mean(i,1));
end
high_percent=mean(high_percent);
low_percent=mean(low_percent);

%Calculate outputs using slopes derived from shortened ranges
for i=(1:1:s)
    new_high(i)=high_percent*(sample_mean(i,16)-sample_mean(i,1))+sample_mean(i,1);
    new_low(i)=low_percent*(sample_mean(i,16)-sample_mean(i,1))+sample_mean(i,1);
    new_slope(i)=(new_high(i)-new_low(i))/13;
    calc_outputs(i,:)=(new_slope(i)*(xlinear-1))+sample_mean(i,2);
end

%Plot sample means of (1) actual sensor outputs, (2) best fit derived from linear
reqression parameters
%and (3) the ouputs calculated using new slopes obtained from percentages.
for i= (1:1:s)
    figure; plot(x, sample_mean(i,:), 'r', xlinear, linear_approximation(i,:), 'g',...
        xlinear, calc_outputs(i,:), 'b');
    xlabel('Width of Sensor Beam Rcvd (mm)');
    ylabel('Sensor Output Voltage (v)');
    legend('Data', 'Linear Approximation', 'Slope Calculations--Truncated Range',2);
    grid minor
end
```

The second script in this section illustrates the commands used to read variables from the dSPACE real time processor, perform calculations based on those variables and then update the variables in the real time processor all while the dSPACE application is running.

Param_change.m

```
% This Matlab script uses the dSPACE MLIB/MTRACE Functions to perform a live update of
MSBS sensor voltage conversion parameters in the MSBS application.  It does this while
the application is running on the DSpace Real Time Processor (RTP), on the DS1103 Real
Time Interface (RTI). (Parameters are called "variables" in the dSPACE environment.)
The script first obtains the correct locations in the RTP for the variables
representing the maximum output of the Sensors and reads in the values.  Then the
sensor output midpoints and the inverse slopes of the input-output response are
calculated. Finally, the variables on the RTP corresponding to these parameters are
located and the newly calc values substituted for the existing ones.

% MLIB/MTRACE functions require the file dSPACErc.m to be present in directory
% \Matlab\Toolbox\Local and a call to this file be appended to
\MATLAB\Toolbox\Local\Matlabrc.m
%
% Created by Mark Adams, June 2003
% Last modified 03/11/04

%Select DS1103 for use with MLIB/MTRACE
mlibini;

%Get variable addresses for sensor voltages.
sens_voltages=...
    {'Model Root/Sensor \nSubsystem/Positioning Subsystem/Sensor S1/Out1';...
    'Model Root/Sensor \nSubsystem/Positioning Subsystem/Sensor S2/Out1';...
    'Model Root/Sensor \nSubsystem/Positioning Subsystem/Sensor S3/Out1';...
    'Model Root/Sensor \nSubsystem/Positioning Subsystem/Sensor S4/Out1';...
    'Model Root/Sensor \nSubsystem/Positioning Subsystem/Sensor S5/Out1'};
```

```
% Capture 10 samples of each sensor output voltage in unobstructed (full beam)
condition
voltages_vec1=mlib('GetTrcVar',sens_voltages);
mlib('Set', 'TraceVars', voltages_vec1, 'NumSamples', 10);
mlib('StartCapture');
while mlib('CaptureState')~=0, end;

% Save sensor output voltages in array
display('Sensor Voltages');
voltages=mlib('FetchData');

SensorOff=0.995;      % average min value for all sensors--much more stable than max
value

% Take sample mean of each Sensor output voltage and calculate midpoints and
% inverse slopes for each
SensorOn=mean(voltages');
SensorOff=.995;
midpoint_calc=((SensorOff+SensorOn)/2)';
inv_slope_calc=(0.015./(SensorOn-SensorOff))';


%Convert numeric arrays to cell arrays
midpoints=num2cell(midpoint_calc);
inv_slopes=num2cell(inv_slope_calc);

%Get variable addresses for sensor midpoints and reset values
sensor_midpoints=...
    {'Model Root/Sensor \nSubsystem/Positioning Subsystem/Voltage-
Displacement\nConversions/S1 Midpoint/Value';...
    'Model Root/Sensor \nSubsystem/Positioning Subsystem/Voltage-
Displacement\nConversions/S2 Midpoint/Value';...
    'Model Root/Sensor \nSubsystem/Positioning Subsystem/Voltage-
Displacement\nConversions/S3 Midpoint/Value';...
    'Model Root/Sensor \nSubsystem/Positioning Subsystem/Voltage-
Displacement\nConversions/S4 Midpoint/Value';...
    'Model Root/Sensor \nSubsystem/Positioning Subsystem/Voltage-
Displacement\nConversions/S5 Midpoint/Value'};
midpoints_vec=mlib('GetTrcVar',sensor_midpoints);
mlib('Write', midpoints_vec, 'Data', midpoints);

%Get variable names for sensor inverse slopes and reset values in Real Time Processor
sensor_inv_slopes=...
    {'Model Root/Sensor \nSubsystem/Positioning Subsystem/Voltage-
Displacement\nConversions/S1 Inv Slope/Value';...
    'Model Root/Sensor \nSubsystem/Positioning Subsystem/Voltage-
Displacement\nConversions/S2 Inv Slope/Value';...
    'Model Root/Sensor \nSubsystem/Positioning Subsystem/Voltage-
Displacement\nConversions/S3 Inv Slope/Value';...
    'Model Root/Sensor \nSubsystem/Positioning Subsystem/Voltage-
Displacement\nConversions/S4 Inv Slope/Value';...
    'Model Root/Sensor \nSubsystem/Positioning Subsystem/Voltage-
Displacement\nConversions/S5 Inv Slope/Value'};
inv_slope_vec=mlib('GetTrcVar',sensor_inv_slopes);
mlib('Write', inv_slope_vec, 'Data', inv_slopes);
```

The third script in this section is the one used to compare the values of the magnetic flux density and its gradients obtained both by measurement and by calculation using the relationship

$$B = \frac{I}{I_{\max}} b_{\max}.$$

This experiment was covered in Chapter IV.

Bzz_Bzzz_compare.m

```
% This program uses polynomials describing the z components of magnetic flux densitiy
measured at two different coil currents: (1) 39.8 A, the value Imax used to obtain the
Bz measurements used in the one-DOF MAGLEV test; and (2) 15.74 A, a value of Imax just
above the maximum current needed to levitate the permanent magnet at the extreme lower
end of the one-DOF command input range.  The program plots the values of Bzz and Bzzz
for each coil current together with plots of the calculated values of Bzz and Bzzz
based on the one-DOF modeling assumption: Bzz=(I/Imax)*bzz, where bzz=Bzz at Imax.
Finally, the program calculates the percentage difference between the calculated
values and the measured values of bzz and bzzz.
%
% The purpose of the program is to validate the modeling assumption by comparing
values obtained from assumption to values obtained by direct measurement.

format;
clear;
close all;
% Best fit polynomial coefficients for Bz for Imax = 39.8A:
a1=[63.92 35.06 5.773 -.05612 -.0865];

% Best fit polynomial coefficients for Bz for Imax = 15.74A:
a2=[21.95 12.25 1.998 -.0456 -.03519]

x=-.10;
zo=x;
b1=polyder(a1);
c1=polyder(b1);
b2=polyder(a2);
c2=polyder(b2);
clc
imax=39.9;
io=15.74;
for x=1:1:15
    bzz_imax(x)=polyval(b1,(-.05-((x-1)*.01)));
    bzzz_imax(x)=polyval(c1,(-.05-((x-1)*.01)));
    bzz_i(x)=polyval(b2,(-.05-((x-1)*.01)));
    bzzz_i(x)=polyval(c2,(-.05-((x-1)*.01)));
    bzz_calc(x)=bzz_imax(x)*io/imax;
    bzzz_calc(x)=bzzz_imax(x)*io/imax;
end
z=(-.051:-.01:-.191);
plot(z, polyval(a1,z),'-', z, polyval(a2,z),':');
grid; legend('B_z at I = 39.8A', 'B_z at 15.74 A');
xlabel('Distance Below Coil (m)');
ylabel('Magnetix Flux Density, B_z  (T)');

figure;
plot(z, bzz_imax,'-', z, bzz_i,':', z, bzz_calc,'-.');
grid; legend('B_z_z at I = 39.8A', 'B_z_z at 15.74 A', 'B_z_z Calculated')
xlabel('Distance Below Coil (m)');
ylabel('Magnetic Flux Density--First Order Gradient, B_z_z  (T/m)');

figure
plot(z, bzzz_imax,'-', z, bzzz_i,':', z, bzzz_calc,'-.');
grid; legend('B_z_z_z at I = 39.8A', 'B_z_z_z at 15.74 A', 'B_z_z_z Calculated')
xlabel('Distance Below Coil (m)');
ylabel('Magnetic Flux Density--Second Order Gradient, B_z_z_z  (T/m^2)');

clc;
disp('Absolute differences between calculated and measured values of bzz and bzzz')
Abs_bzz_diff=abs(bzz_i(11)-bzz_calc(11))
Abs_bzzz_diff=abs(bzzz_i(11)-bzzz_calc(11))

disp('Percent differences between calculated and measured values of bzz and bzzz')
Percent_bzz_diff=100*(bzz_i(11)-bzz_calc(11))/bzz_i(11)
Percent_bzzz_diff=100*(bzzz_i(11)-bzzz_calc(11))/bzzz_i(11)
```

Matlab Scripts Used for 1-DOF Controller Synthesis

This section contains Matlab scripts written to produce the 1-DOF linear quadratic gaussian (LQG) regulators described in Chapter V. Only the scripts for the basic iterative design method and not the loop transfer recovery (LTR) method are provide. The basic scripts are of value here to illustrate early experimentation with implementing optimal control concepts for the MSBS, including a design methodology. On the other hand, the LTR scripts, though also illustrative, were substantially revised for the multiple-input, multiple-output (MIMO) case and it was felt inclusion of the 1-DOF LTR scripts would be excessive.

ONE_DOF_LQR.m

```
% This program is an LQR design tool.
% Created by Mark Adams.
% Last Modified:  01/27/04

format; clc; clear; close all;
% Variables needed on Matlab Workspace
global cmd fb1 fb2 simout d_out io t1 default m bpert fd a_lqg b_lqg c_lqg d_lqg;
t1=clock;

% ****** Derive State Space Model and Nonlinear Model Coefficients for Plant *******
One_DOF_Plant;

% ****** Determine whether to use default values or user input ********
msg='Use default ranges for Q, R and desired regulator set points? (Y/N):  ';
default=prompt(msg);

% ********************************************************************************
% Call function QEVAL to obtain ranges of feedback coefficients and eigenvalues
% obtained from Matlab command [K,S,E]=LQR(A,B,Q,R), along with the values of
% the elements in the corresponding Q and R weighting matrices.

[K_range, E_range, input_range]=QEVAL(A,B);

% % ********************************************************************************
% Implement Range of LTI Systems and determine time responses for each system.
[step_resp]=CL_RESPONSES(A,B,C,K_range);

% *** Find Values Meeting Threshold Criteria and Corresponding Q,R for time responses
***
msg='Perform Criteria Threshold tests?  (Y/N):  ';
threshtest=prompt(msg);
if threshtest=='y'
[step_thresh]=THRESHOLDS(step_resp, input_range);
end

% ******** Determine whether to do controller stiffness tests ******************
msg='Perform controller stiffness tests?  (Y/N):  ';
stifftest=prompt(msg);
if stifftest=='y'
    [cstiff]=STIFFNESS_TEST(A,B,C,K_range)
end
% break
% ******** Determine whether to do B Field perturbation tests ******************

msg='Perform B Field perturbation tests?  (Y/N):  ';
btest=prompt(msg);
if btest=='y'
    [bptest]=B_PERTURB(A,B,C,step_resp,K_range);
end
```

```
msg='Calculate new lsvf coefficients based on different Q and R values? (Y/N): ';
recalc=prompt(msg);
if recalc=='y'
    default_old=default;
    default='n';
    [K_range, E_range, input_range]=qeval(A,B);
    [step_resp]=cl_responses(A,B,C,K_range);

    msg='Repeat Criteria Threshold tests?  (Y/N):  ';
    threshtest=prompt(msg);
    if threshtest=='y'
        [step_thresh]=thresholds(step_resp, input_range);
    end
    msg='Repeat controller stiffness tests?  (Y/N):  ';
    stifftest=prompt(msg);
    if stifftest=='y'
        [cstiff]=stiffness_test(A,B,C,K_range);
    end
    msg='Repeat B Field perturbation tests?  (Y/N):  ';
    btest=prompt(msg);
    if btest=='y'
        [bptest]=B_PERTURB(A,B,C,step_resp,K_range);
    end

    default=default_old;
end     % if recalc='y'

% ********** Determine whether to perform LQG regulator test ********
msg='Create Kalman state estimator and perform system response tests?  (Y/N):  ';
    estimator=prompt(msg);
    if estimator=='y'
        [lqg_resp]=LQG_TEST(A,B,C,K_range);
    end
```

ONE_DOF_PLANT.m

```
% This program calculates values for state space matrices A,B,C,D for a one-degree-of-
freedom maglev system. The program also calculates linearized plant transfer functions
and parameters for use in the Similink non-linear model.

% Program uses 4th degrree polynomials to describe z components of the magnetic flux
densities from each magnet.  The polynomials are fitted to the plots of actual
measurements.

%For coil and permanent magnet #1 (Coefficients from set B1.)
a=[.2664 -2.289 -2.314 -.8194 -.1125];
b=polyder(a);
c=polyder(b);
zo=-.17835;
bzz=polyval(b,zo);          %polynomial describing first order gradient
bzzz=polyval(c,zo);         %polynomial describing second order gradient
imax=39.8;
m=.192;                     %190.2 grams without adhesive filler
mu=pi*4e-7;
v=16.09e-6;
Mz=(-1.22/mu);
g=9.81;
io=(m*g*imax)/(v*Mz*bzz);           %Equilibrium current
A=[0 1;(v*Mz*io*bzzz)/(imax*m) 0];  %State Space form of linearized plant
B=[0 v*Mz*bzz/(imax*m)]';
C=[1 0];
D=[0];
% ******************************** Hold for use with non linear model ***************

% pause;

k1=(v*Mz*bzz)/(imax*m);     %Constants for use in non-linear model
k2=(v*Mz*bzzz)/(imax*m);
```

QEVAL.m

```
% This function takes the specified range limits for the values of Q and R, along with
the state space matrix A and input matrix for the plant, and calculates the
corresponding range of state variable feedback values and repositioned pole values
using the LQR function.

% Created by Mark Adams
% Last Modified:  01/27/04
%
% The function uses the input variables:
%
% A_plant:     The state variable matrix for the plant
% B_plant:     The input matrix for the plant
% Q_bound:     The limit for the range of values in the diagonal weighting matrix Q
%              (all elements have the same range.)
% R_bound:   The limit for the range of values in the weighting matrix, R.
% incr:        Size of the increments between values in Q and R.

% The function returns the variables:
%
% feedback_range:    An array containing the elements of the diagonal feedback matrix K
% pole_range:        An array containing the new eigenvalues obtained with s.v.
feedback
% QR_range:          An array containing the elements of the two weighting matrices for
the
%                    ranges specified by the input variables

function[feedback_range, pole_range, QR_range]=Qeval(A_plant, B_plant);
global default;

% *************** Use Test Range or Take User Input **********************

if default=='y'
    % Use Test Range
    Q=[1 0;0 1];
    R=1;
    [K,S,E]=lqr(A_plant,B_plant,Q,R);
    QR_range=[1 1 1];
    feedback_range=K;
    pole_range=E';
else
    % User Input range of Weighting Matrices, Feedback Constants, Eigenvalues
    range_u=input('Max value in range of elements of Q & R matrices:  ');  %High end
of range
    range_l=input('Min value (but not 0) in range of elements of Q & R matrices:  ');
%Low end of range
    no_values=input('No. of equally spaced values in range, including both range
endpoints:  ');    %Input no. of values in range.


    % (E.g., for desired range = 10 and no_values = 10, Q varies from [.1 0; 0 .1] to
    % [1.0 0; 0 1.0] in increments of 0.1 in the diagonal elements. Each element
    % is varied separately for a total of 10*10=100 permutations.  Also, R varies
    % from 0.1 to 1.  The total no. of permutations for = 10*10*10=1000.)
    % Note:  current configuration assumes both diagonal elements of Q and the single
    % element of R all have same range and are incremented by the same value.

    % ******** Set Up Parameters for Calculating Permutations of Q and R ************
    step_size=(range_u-range_l)/(no_values-1);
    range_len=(no_values^3);     %Determine length of arrays.
    pole_range=zeros(range_len, 2);     %pre-allocate arrays
    feedback_range=zeros(range_len,2);
    QR_range=zeros(range_len,3);
    step_size=(range_u-range_l)/(no_values-1);
    r=(range_l:step_size:range_u);
    q1=r;
    q2=q1;
```

```
    % ******************* Create Permutations of R and Q ***************************
    count=0;
    for i=1:no_values
        for j=1:no_values
            for k=1:no_values
                count=count+1;
                QR_range(count,:)=[r(i) q1(j) q2(k)];
                Q=[q1(j) 0; 0 q2(k)];
                [K,S,E]=lqr(A_plant, B_plant, Q, r(i));
                feedback_range(count,:)=K;
                pole_range(count,:)=E';
            end     % innermost for loop
        end         % next ror Loop
    end;            % outer for loop
end;            % If then else statement
```

CL_RESPONSES.m

```
% This function takes a range of state variable feedback constants and derives new
transfer functions for the plant.  The function then uses the LTI transfer functions
to implement closed loop unity feedback systems, which are tested with step and
impulse functions.  The function performs the same procedure for the nonlinear model
by invoking a Simulink model.  The tested system responses are returned in vectors.
%
% Created by Mark Adams
% Last Modified 01/27/04

% The function uses input variables:
%
% A_plant:  State variable matrix for the plant
% B_plant:  Input matrix for plant
% C_plant:  Output matrix for plant
% fcs:      An array containing linear state variable feedback (lsvf) coefficients

% The function returns variables:
%
% trstep    An array containing percent overshoot, settling time
%           and steady state response for both the linear and nonlinear models
%           as well as magnitude of the maximum deviation of the control input from
%           its equilibrium,
% tr_imp    An array containg maximum value, final value and settling
%           time of impulse response.  ****not implemented at this time****

function[trstep]=CL_RESPONSES(A_plant, B_plant, C_plant, fcs);
global cmd fb1 fb2 simout d_out default t1 io bpert fd;  %variables needed on Matlab
workspace
fd=0;           % switch for additive disturbance force in Simulink model
bpert=0;        % switch for additive b-field perturbations in Simulink model

if default=='y'
    d_out=[-.005 -.004 -.003 -.002 -.001 0 .001 .002 .003 .004 .005];
    d_out_len=11;
else
    disp('');
    d_out_l=.001*input('Enter lower limit of desired regulator set point range (in mm)
:');
    d_out_u=.001*input('Enter upper limit of desired regulator set point range (in
mm):  ');
    d_out_incr=.001*input('Enter increment between desired set points in range (in
mm):  ');
    d_out=[d_out_l:d_out_incr:d_out_u];
    d_out_len=length(d_out);
end     % if default

len=length(fcs(:,1));           %length of fcs equals no. of permutations to calculate
trstep=zeros(len,8,d_out_len);  %preallocate arrays to hold time responses
% trimp=zeros(len,7,d_out_len); % not implemented at this time
elapsed_time=zeros(d_out_len,1);
t=(0:.01:5.0);                      % Create time scale
lent=length(t);
D=[0];
```

```
for h=1:d_out_len                % sequence through all set points
    elapsed_time(h)=etime(clock,t1);
    for i=1:len                  % sequence through all lsvf coefficients once for
                                 % each set point

        %Create closed-loop fransfer function, Gcl, using lsvf coefficients
        Ac=A_plant-B_plant*fcs(i,:);
        Gcl=ss(Ac,B_plant,C_plant,D);
        % Invert closed-loop transfer function and evaluate at final value.
        % Then use to find input corresponding to final value = yd.  (Kailath p.206)
        %         Icl=dcgain(inv(Gcl));
        Icl=inv(dcgain(Gcl));
        cmd=Icl*d_out(h);
        trstep(:,8,h)=cmd;

        % Derive amplitude step function need to for set point
        ustep=ones(lent,1);                % Create step input sequence
        ustep=cmd*ustep;

        % Determine step response of closed-loop, pole-shifted, linear system
        [ystep]=lsim(Gcl,ustep,t);

        % Determine step response of non-linear system
        fb1=fcs(i,1);
        fb2=fcs(i,2);
        sim('nonlinear_lqr');

        % Determine final value of step response of linear system
        trstep(i,3,h)=ystep(lent);
        if ystep(lent)==0    %Avoid division by zero in following calculations
            ystep(lent)=1e-12;
        end

        % Determine final value of step response of non-linear system
        trstep(i,6,h)=simout(lent);
        if simout(lent)==0   %Avoid division by zero in following calculations
            simout(lent)=1e-12;
        end

        % Determine percent overshoot of linear system response to step input
        if d_out(h)<0.00
            trstep(i,1,h)=((min(ystep)-ystep(lent))/ystep(lent))*100;
        else
            trstep(i,1,h)=((max(ystep)-ystep(lent))/ystep(lent))*100;
        end

        % Determine percent overshoot of non-linear system response to step input
        if d_out(h)<0.00
            trstep(i,4,h)=((min(simout)-simout(lent))/simout(lent))*100;
        else
            trstep(i,4,h)=((max(simout)-simout(lent))/simout(lent))*100;
        end

        % Determine settling time (within 2% of final value) of linear system
        cross=0;
        for tstep=1:lent
            if abs(ystep(tstep)-trstep(i,3,h))<=.02*abs(trstep(i,3,h))
                cross=cross+1;
                if cross==2
                    trstep(i,2,h)=.01*tstep;
                    break
                end      % end if cross
            end          % end if abs
        end              % end for tstep
        if cross<=1
            trstep(i,2,h)=.01*lent;
        end      % end if cross

        % Determine settling time (within 2% of final value) of non-linear system
        cross=0;
```

```
        for tstep=1:lent
            if abs(simout(tstep)-trstep(i,6,h))<=.02*abs(trstep(i,6,h))
                cross=cross+1;
                if cross==2
                    trstep(i,5,h)=.01*tstep;
                    break
                end     % end if cross
            end         % end if abs
        end             % end for tstep
        if cross<=1
            trstep(i,5,h)=.01*lent;
        end     % end if cross

        % Determine magnitude of maximum deviation of control input to non-linear
model
        trstep(i,7,h)=max((max(simin)-io),(min(simin)-io));

    end                 % end for i=1:len
end                     % end for h=1:d_out_len
```

THRESHOLDS.m

```
% This function examines ranges of step and impulse response data (percent overshoot,
settling time, steady state value, max impulse response, impulse settling time, and
steady state value) for values that meet certain thresholds.  The function then
returns those values along with the corresponding values for weighting matrices that
produced them.
%
% Created by Mark Adams
% Last Modified:  01/27/04

% This function uses the following inputs:
% trstep:         A 3-D array containing step responses of linear and non-linear
systems
%                 created using a range of linear state variable feedback (lsvf)
%                 coefficients.
% QR_range:       A 3-D array containing LQR weighting matices used to produce a range
of
                  lsvf coefficients, which in turn were used to create the responses
in
                  trstep.

% The function returns the variables:
% sr_thresh       An array whose columns represent the time response values contained
%                 in the input array 'trstep' that meet designated threshold criteria.
                  The time response values represented are Percent OS, Settling Time,
                  Final Value. Along with the values selected are the corresponding
                  values of Q and R that produced them.  sr_thresh is three
dimensional,
                  e.g., sr_thresh(x,y,z), where x is no. of values meeting the
criteria
                  for any of the three responses of interest, y is the no. of tr
values
                  and their corresponding R&Q, z is the number of desired regulator
                  outputs for which time responses were measured.

function[sr_thresh]=THRESHHOLDS(trstep, QR_range);
global io d_out default;
disp(' ');
disp(' ');

if default=='y'
    % Use Test criteria
    perc_os=1;
    st=.65;
    perc_ss=.05;
    perc_io=.10;
else
    disp('Enter the threshold criteria for cl step responsed of candidate systems.
');
```

```
    perc_os=.01*input('Enter maximum percent overshoot (E.g., use 2 to indicate 2%.)
');
    st=input('Enter maximum desired settling time in seconds.  ');
    perc_ss=.01*input('Enter maximum allowable percent error in steady state value.
(E.g. 2 indicates 2%) ');
    perc_io=.01*input('Enter maximum percent increase in equilibriun current (E.g. 10
indicates 10%) ');
end

ioval=abs(perc_io*io);  % magnitude of the linear system's equilibrium current
s=size(trstep);         % size of array containing step responses
sr_thresh=zeros(s(1),4*s(2),s(3));  %preallocate arrays to hold values meeting
criteria

% ************ Select step responses meeting threshold criteria ****************

for h=1:s(3)        %Loop through range of desired regulator set points
    perc_ss;
    ssval=abs(perc_ss*d_out(h));

    b=1;            % Percent overshoot of linear model
    ind=0;
    for j=1:s(1)
        count=j;
        QR=QR_range(count,:);
        if trstep(j,1,h)<=perc_os
            ind=ind+1;
            sr_thresh(ind,b:b+2,h)=QR;
            sr_thresh(ind,b+3,h)=trstep(j,1,h);
        end
    end

    b=5;            % Settling Time of linear model
    ind=0;
    for j=1:s(1)
        count=j;
        QR=QR_range(count,:);
        if trstep(j,2,h)<=st
            ind=ind+1;
            sr_thresh(ind,b:b+2,h)=QR;
            sr_thresh(ind,b+3,h)=trstep(j,2,h);
        end
    end

    b=9;            % Percent error of steady state value of linear model
    ind=0;
    for j=1:s(1)
        count=j;
        QR=QR_range(count,:);
        if abs(trstep(j,3,h)-d_out(h))<=ssval
            ind=ind+1;
            sr_thresh(ind,b:b+2,h)=QR;
            sr_thresh(ind,b+3,h)=trstep(j,3,h);
        end
    end

    b=13;           % Percent overshoot of non-linear model
    ind=0;
    for j=1:s(1)
        count=j;
        QR=QR_range(count,:);
        if trstep(j,4,h)<=perc_os
            ind=ind+1;
            sr_thresh(ind,b:b+2,h)=QR;
            sr_thresh(ind,b+3,h)=trstep(j,4,h);
        end
    end

    b=17;           % Settling Time of non-linear model
    ind=0;
    for j=1:s(1)
```

```
        count=j;
        QR=QR_range(count,:);
        if trstep(j,5,h)<=st
            ind=ind+1;
            sr_thresh(ind,b:b+2,h)=QR;
            sr_thresh(ind,b+3,h)=trstep(j,5,h);
        end
    end

    b=21;           % Percent error of steady state value of non-linear model
    ind=0;
    for j=1:s(1)
        count=j;
        QR=QR_range(count,:);
%         abs(trstep(j,6,h)-d_out(h));
        if abs(trstep(j,6,h)-d_out(h))<=ssval
            ind=ind+1;
            sr_thresh(ind,b:b+2,h)=QR;
            sr_thresh(ind,b+3,h)=trstep(j,6,h);
        end
    end

    b=25;           % Percent change in equilibrium current
    ind=0;
    for j=1:s(1)
        count=j;
        QR=QR_range(count,:);
        if abs(trstep(j,7,h)-io)<=ioval
            ind=ind+1;
            sr_thresh(ind,b:b+2,h)=QR;
            sr_thresh(ind,b+3,h)=trstep(j,7,h);
        end
    end

end         %for h=1:s(3)
    disp('Following are the non-linear system responses meeting the threshhold
criteria  ')
    disp('for percent overshoot, settling time, final value and maximum control input:
')
    sr_thresh(:,13:28,:)
msg='Display both linear and non-linear responses?  (Y/N):  ';
all_vals=prompt(msg);
if all_vals=='y'
    sr_thresh
end % if all_vals
```

STIFFNESS_TEST.m

```
% This function takes plant state equations & a range of state variable feedback
constants and derives new transfer functions for the linearize plant.  The function
then uses the LTI transfer functions to determine appropriate cmd inputs for the
Simulink nonlinear lsvf model. The nonlinear model is then run using the cmd settings,
the corresponding state variable feedback constants and a known disturbance force that
simulates an aerodynamic disturbance.  The stiffness quotient, i.e. the ratio of the
force to the displacement it produces, for each cmd setting is returned.
%
% Created by Mark Adams
% Last Modified 01/27/04

% The function uses input variables:
%
% A_plant:  State variable matrix for the plant
% B_plant:  Input matrix for plant
% C_plant:  Output matrix for plant
% fcs:      An array containing the state variable feedback constants

% The function returns variables:
%
% fdresp    An array containing the stiffness quotients for the nonlinear system
%           for a range of lsvf constants and cmd inputs.
```

```
function[fdresp]=STIFFNESS_TEST(A_plant, B_plant, C_plant, fcs);
global cmd fb1 fb2 simout default t1 m bpert fd;  %variables needed on Matlab
workspace

len=length(fcs(:,1));           % length of fcs equals no. of permutations to
calculate
t=(0:.01:5.0);                  % Create time scale for simulation
lent=length(t);
hflent=floor(lent/2);
D=[0];
fd_max=1/m;     % represents magnitude of max disturbance force in Simulink model
in_range=[-.005 0 .005];
in_len=length(in_range);
fdresp=zeros(len,4*in_len);  %preallocate arrays to hold stiffness responses

fd=0;           % switch for additive disturbance force in Simulink model
bpert=0;        % switch for additive b-field perturbations in Simulink model

for h=1:in_len
    elapsed_time(h)=etime(clock,t1);
    gp=h+3*(h-1);

    for i=1:len

        %Create closed-loop fransfer function, Gcl, for pole-shifted plant
        Ac=A_plant-B_plant*fcs(i,:);
        Gcl=ss(Ac,B_plant,C_plant,D);

        % Invert closed-loop transfer function and evaluate at final value.
        % Then use to find input corresponding to final value = yd.  (Kailath p.206)
        Icl=inv(dcgain(Gcl));
        cmd=Icl*in_range(h);

        % Obtain response of non-linear system
        fd=0;
        fb1=fcs(i,1);
        fb2=fcs(i,2);
        sim('nonlinear_lqr');
        fv=simout(lent);    %obtain final value without Fd
        fd=1;
        sim('nonlinear_lqr');   % obtain output with Fd

        % Test response for max/min and calculate stiffness quotient
        a=1/max(simout(hflent:lent)-fv);
        b=1/min(simout(hflent:lent)-fv);
%         if fv~=0        % normalized displacement from nominal position
%             a=a/fv;
%             b=b/fv;
%         end
        c=(abs(a)+abs(b))/(2);
        fdresp(i,gp:gp+3)=[fv a b c];
        if len<=2
            plot(t,simout1);
        end
    end     %inner for loop
end         % outer for loop
```

B_PERTURB.m

```
% This function takes plant state equations & a range of state variable feedback
constants and derives new transfer functions for the linearize plant.  The function
then uses the LTI transfer functions to determine appropriate cmd inputs for the
Simulink nonlinear lsvf model. The nonlinear model is then run using the same cmd
settings used for the unperturbed time responses only with the B field coefficients
perturbed.

% Created by Mark Adams
% Last Modified 01/27/04
```

```
% The function uses input variables:
%
% A_plant:  State variable matrix for the plant
% B_plant:  Input matrix for plant
% C_plant:  Output matrix for plant
% fcs:      An array containing the state variable feedback constants

% The function returns variables:
%
% fdresp    An array containing the stif+fness quotients for the nonlinear system
%           for a range of lsvf constants and cmd inputs.

function[bpresp]=b_perturb(A_plant, B_plant, C_plant, trstep, fcs);
global cmd fb1 fb2 simout default d_out t1 m bpert io fd;  %variables needed on Matlab
workspace
len=length(fcs(:,1));           % length of fcs equals no. of permutations to
calculate
t=(0:.01:5.0);                  % Create time scale for simulation
lent=length(t);
D=[0];
d_out_len=length(d_out);
bpresp=zeros(len,12,d_out_len);  % preallocate array to hold B Field Perturbation
responses
fd=0;           % switch for additive disturbance force in Simulink model
bpert=0;        % switch for additive b-field perturbations in Simulink model

for h=1:d_out_len   % sequence through range of set points
    yd=d_out(h);
    cmd=trstep(1,8,h);
    elapsed_time(h)=etime(clock,t1);

    for i=1:len
        bpresp(i,1,h)=trstep(i,4,h);    % Copy non-perturbed time responses into new
matrix
        bpresp(i,4,h)=trstep(i,5,h);
        bpresp(i,7,h)=trstep(i,6,h);
        bpresp(i,10,h)=trstep(i,7,h);

        for j=1:5
            % Determine step response of non-linear system
            fb1=fcs(i,1);
            fb2=fcs(i,2);
            bpert=(0.05*(2*rand-1));
            sim('nonlinear_lqr');
            if abs(d_out(h))==.005
%             elseif abs(d_out(h))==.002
                plot(t, simout); hold on;
            end

            % Determine final value of step response of non-linear system
            fv(j)=simout(lent);
            if simout(lent)==0   %Avoid division by zero in following calculations
                simout(lent)=1e-12;
            end %if simout

            % Determine percent overshoot of non-linear system response to step input
            if d_out(h)<0.00
                perc_os(j)=((min(simout)-simout(lent))/simout(lent))*100;
            else
                perc_os(j)=((max(simout)-simout(lent))/simout(lent))*100;
            end     %end if d_out

            % Determine settling time (within 2% of final value) of non-linear system
            cross=0;
            for tstep=1:lent
                if abs(simout(tstep)-fv(j))<=.02*abs(fv(j))
                    cross=cross+1;
                    if cross==2
                        st(j)=.01*tstep;
                        break
                    end % end if cross
```

```
            end         %end if abs
        end             %end for tstep
        if cross<=1
            st(j)=.01*lent;
        end     % end if cross

        %Determine max input signal to non-linear model
        in(j)=max((max(simin)-io),(min(simin)-io));

    end     % end for j=1:5

    bpresp(i,2,h)=mean(perc_os);
    bpresp(i,3,h)=std(perc_os);

    bpresp(i,5,h)=mean(st);
    bpresp(i,6,h)=std(st);

    bpresp(i,8,h)=mean(fv);
    bpresp(i,9,h)=std(fv);

    bpresp(i,11,h)=mean(in);
    bpresp(i,12,h)=std(in);


  end               % end for i=1:len
end                 % end for h=1:d_out_len
```

LQG_TEST.m

```
% This function takes a range of state variable feedback constants and derives new
transfer functions for the plant.  The function then uses the LTI transfer functions
to implement closed loop unity feedback systems, which are tested with step and
impulse functions.  The function performs the same procedure for the nonlinear model
by invoking a Simulink model.  The tested system responses are returned in vectors.
%
% Created by Mark Adams
% Last Modified 01/28/04

% The function uses input variables:
%
% A_plant:  State variable matrix for the plant
% B_plant:  Input matrix for plant
% C_plant:  Output matrix for plant
% fcs:      An array containing linear state variable feedback (lsvf) coefficients

% The function returns variables:
%
% resp:     An array containing percent overshoot, settling time
%           and steady state response for both the nonlinear model
%           as well as magnitude of the maximum deviation of the control input from
%           its equilibrium,

function[resp]=LQG_TEST(A_plant, B_plant, C_plant, fcs);

%variables needed on Matlab workspace
global cmd fb1 fb2 simout d_out default t1 io bpert fd a_lqg b_lqg c_lqg d_lqg ;
fd=0;           % switch for additive disturbance force in Simulink model
bpert=0;        % switch for additive b-field perturbations in Simulink model
if default=='y'
    d_out=[-.005 -.004 -.003 -.002 -.001 0 .001 .002 .003 .004 .005];
    d_out_len=11;
else
    disp('');
    d_out_l=.001*input('Enter lower limit of desired regulator set point range (in
mm):  ');
    d_out_u=.001*input('Enter upper limit of desired regulator set point range (in
mm):  ');
    d_out_incr=.001*input('Enter increment between desired set points in range (in
mm):  ');
```

```
    d_out=[d_out_1:d_out_incr:d_out_u];
    d_out_len=length(d_out);
end     % if default

len=length(fcs(:,1));           %length of fcs equals no. of permutations to calculate
resp=zeros(len,6,d_out_len);    %preallocate arrays to hold time responses
elapsed_time=zeros(d_out_len,1);
t=(0:.01:5.0);                      % Create time scale
lent=length(t);
D=[0];

G=[1;1];        % Process noise gain in state equation x' = Ax + Bu + Gw
H=[1];        % Process noise gain in output equation y = Cx + Du + Hw + v

sys=ss(A_plant,B_plant,C_plant, D);
sys_k=ss(A_plant,[B_plant G],C_plant,[D H]);    % Separate control and disturbance
inputs
[kest]=kalman(sys_k,1,.1);                      % Create Kalman estimator

for h=1:d_out_len               % sequence through all set points
    elapsed_time(h)=etime(clock,t1);
    for i=1:len                 % sequence through all lsvf coefficients once for
                                % each set point

        %Create LQG Regulator using lsvf coefficients
        F=lqgreg(kest,fcs(i,:));                % Combine LSVF coefficients and Kalman
estimator
        [a_lqg, b_lqg, c_lqg, d_lqg]=ssdata(F);  % Obtain ss representation of LQG
Regulator

        %Create closed-loop fransfer function, Gcl
        Gcl=feedback(sys,F,+1);
        % Invert closed-loop transfer function and evaluate at final value.
        % Then use to find input corresponding to final value = yd.  (Kailath p.206)
        input_gain=inv(dcgain(Gcl));
        cmd=input_gain*d_out(h);
        resp(:,5,h)=d_out(h);

        % Determine step response of non-linear system
        sim('nonlinear_lqg');
%                   sim('nonlinear_lqr');

        % Determine final value of step response of non-linear system
        resp(i,3,h)=simout(lent);
        if simout(lent)==0   %Avoid division by zero in following calculations
            simout(lent)=1e-12;
        end

        % Determine percent overshoot of non-linear system response to step input
        if d_out(h)<0.00
            resp(i,1,h)=((min(simout)-simout(lent))/simout(lent))*100;
        else
            resp(i,1,h)=((max(simout)-simout(lent))/simout(lent))*100;
        end

        % Determine settling time (within 2% of final value) of non-linear system
        cross=0;
        for tstep=1:lent
            if abs(simout(tstep)-resp(i,3,h))<=.02*abs(resp(i,3,h))
                cross=cross+1;
                if cross==2
                    resp(i,2,h)=.01*tstep;
                    break
                end     % end if cross
            end         % end if abs
        end             % end for tstep
        if cross<=1
            resp(i,2,h)=.01*lent;
        end     % end if cross
```

```
        % Determine magnitude of maximum deviation of control input to non-linear
model
        resp(i,4,h)=max((max(simin)-io),(min(simin)-io));

        resp(i,6,h)=i;

    end                 % end for i=1:len
end                     % end for h=1:d_out_len
for h=1:d_out_len
    disp(' ')
    disp('       %  OS  | ST   |  SS Val |  Iin   | Set Point | Row Ind ');
    resp(:,:,h)
end     % for h=1:d_out_len
```

Matlab Scripts Used for 2-DOF Controller Synthesis

This section contains Matlab scripts written to produce the 2-DOF linear quadratic gaussian (LQG) regulators described in Chapter VI. The first group of scripts are those used to implement the loop transfer recovery (LTR) method. The second group are those used in conducting the iterative design procedure.

TWO_DOF_LTR.m

```
% This program is an LTR design tool.
% Created by Mark Adams.
% Last Modified:  04/14/04

format; clc; clear; close all;
% Variables needed on Matlab Workspace
global cmd Kin Ieq bpert fd Ic m Ts;
Ts=.001;
w=logspace(-2,3,400);   % Create log frequency range for responses

% ****** Derive Analytic State Space Model of Plant from Eqns *******
TWO_DOF_PLANT;
Izinv=sum([0 0 1]*Icinv)

% ** Extract States Relating to Pitch and Vertical Movement/Create DT Model **
[Am,Bm,Cm,Dm]=ssselect(A,B,C,D,[1 2],[1 2],[1 3 7 10]);
minsys=ss(Am,Bm,Cm,Dm);
sysd=c2d(minsys,Ts);
[ad,bd,cd,dd]=ssdata(sysd);
[at,bt,ct,dt]=bilin(sysd.a,sysd.b,sysd.c,sysd.d,-1,'Tustin',Ts);


LTR='y';            % Loop containing whole process
while LTR=='y'

    % *** Synthesize and evaluate range of Kalman Estimators ********
    disp(' ');
    disp('*** Synthesize and evaluate range of Kalman Estimators ********');
    [Kf,Gw,Rest,Kest,L_kf]=KALMAN_SYNTH(minsys,sysd);
    Kagain='n';
    msg='Repeat Kalman Estimator synthesis using different process noise gains? (Y/N):
';
    Kagain=prompt(msg);
    while Kagain=='y'
        [Kf, Gw, Rest, Kest, L_kf]=KALMAN_SYNTH(minsys,sysd);
        Kagain=prompt(msg);
    end

    % ***************** Perform Loop Transfer Recovery Process ******************
    [Qlqr,rho,Klqr,L_sys]=LTR_PROCESS(Gw, Rest, L_kf, minsys, sysd);
    LTRagain='n';
    msg='Repeat LTR Process using different multiplier value(s)? (Y/N): ';
```

```
    LTRagain=prompt(msg);
    while LTRagain=='y'
        [Qlqr, rho,Klqr,L_sys]=ltr_process(Gw,Rest,L_kf,minsys,sysd);
        LTRagain=prompt(msg);
    end

    %******  Implement LQG system model and determine time responses **********
    SR=prompt('Determine step response of nonlinear model? (Y/N) ');
    if SR=='y'
        [step_values, step_resp,tout]=STEP_RESPONSE(Klqr, minsys, sysd, 'n');
    end
    figure;
    plot(tout,step_resp);

    % ******** Determine whether to do controller stiffness tests **************
    disp(' ');
    msg='Perform controller stiffness tests?  (Y/N):  ';
    stifftest=prompt(msg);
    if stifftest=='y'
        [stf_values, stf_resp, tout, d_force]=STIFFNESS_TEST(Klqr, minsys,sysd);
    end

    %     ******** Determine whether to do B Field perturbation tests
********************
    disp(' ');
    msg='Perform B Field perturbation tests?  (Y/N):  ';
    btest=prompt(msg);
    if btest=='y'
        [bp_values, bp_resp, tout]=B_PERTURB(Klqr,minsys,sysd);
    end

    LTR=prompt('Repeat Entire LTR Implementation Process?  ');
End  %While
```

TWO_DOF_PLANT

```
% This program provides the state space form of the linearized
% two-DOF MSBS system.  It also places certain MSBS parameters
% on the MATLAB workspace for modeling.

% Created by Mark Adams
% Last Modified 03/03/04

% Third set of Values calculated by Dr. Britcher:
%       1Bx 2By 3Bz 4Bxx 5Bxy 6Bxz 7Byy  8Byz 9Bzz 10Bxxx 11Bxyx 12Bxzx 13Bxyy 14Bxyz
15Bxzz 16Byyy
%       17Byyz 18Byzz 19Bzzz
%
Bmax=[-.0223 0  -.0117 -.0796  0 -.1476 .0837 0 -.1304  .3793 0 -1.0782  .3139 0 -
1.0111 0 0 0 1.0782;...
       .0223 0  -.0117 -.0796  0  .1476 .0837 0 -.1304 -.3793 0 -1.0782 -.3139 0
1.0111 0 0 0 1.0782];

m=4.383; %mass of permanent magnet core and shell in kg determmined 1/14/04 Dr.
Britcher/M Adams

v=6*(0.5*.0254)*pi*(1.5*.0254)^2;   %volume of magnetic core composed of six 3.0" x
0.5" discs
mu=pi*4e-7;
Mx=(1.22/mu);
g=9.81;
Imax=100;

% Moment of Inertia for Test Model
mp=1.6383;          % mass of pipe alone
mph=1.6601;         % mass pf pipe and temp magnet holder
mphs=4.383;         % mass of pipe, holder and magnets
m=mphs-(mph-mp);    % mass of test object (magnets and pipe)
mm=mphs-mph;        % mass of magnets
```

```
lm=(11/2)*.0254;     % length of magnet subassy
Rm=1.50*.0254;        % radius of magnet subassy
lp=30*.0254;         % length of pipe
Rp=3.5*.0254;        % radius of pipe
Im=(mm/12)*lm^2+(mm/4)*Rm^2;     % moment of inertia of magnets around transverse axis
Ip=(mp/12)*lp^2+(mp/4)*Rp^2;     % moment of inertia for pipe
Ic=Im+Ip                        % total moment of inertia around transverse axes
Imx=0.5*mm*Rm^2;                 % moment of inertia of magnets around axis of symetry
Ipx=0.5*mp*Rp^2;
Ix=Imx+Ipx
a=-Bmax(1,3)/Imax;
b=-Bmax(2,3)/Imax;
c=Bmax(1,6)/Imax;
d=Bmax(2,6)/Imax;
e=(m*g)/(v*Mx);
A=[a b;c d]; C=[0 e]';
Ieq=inv(A)*C    %Determine equilibrium current vector for both coils
Beq=(Bmax(1,:)*Ieq(1)+Bmax(2,:)*Ieq(2))/Imax;   %Calculate aggregate fields and
gradients
W1=((v*Mx)/m)*eye(10);
W1(3,3)=1;W1(4,4)=1;W1(8,8)=1;W1(9,9)=1;W1(10,10)=1;
% Create A matrix whose terms are defined based on I=Ieq
A1=[0  0 -(m/Ic)*Beq(1)     0                0 0 0  -(m/Ic)*Beq(6) -(m/Ic)*Beq(8) -
(m/Ic)*Beq(9);
    0  0 0                 -(m/Ic)*Beq(1)   0 0 0   (m/Ic)*Beq(5)   (m/Ic)*Beq(7)
(m/Ic)*Beq(8);
    1  0 0                  0                0 0 0  0                0                0;
    0  1 0                  0                0 0 0  0                0                0;
    0  0 -Beq(6)            Beq(5)           0 0 0  Beq(10)          Beq(11)
Beq(12);
    0  0 -Beq(8)            (Beq(7)-Beq(4))  0 0 0  Beq(11)          Beq(13)
Beq(14);
    0  0 (Beq(4)-Beq(9))    Beq(8)           0 0 0  Beq(12)          Beq(14)
Beq(15);
    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];
A=W1*A1;
% Create B Matrix whose terms are defined with (Bmax/Imax)*I

B=((v*Mx)/Imax)*...
    [-Bmax(1,3)/Ic     -Bmax(2,3)/Ic;
      Bmax(1,2)/Ic      Bmax(2,2)/Ic;
      0                 0;
      0                 0;
      Bmax(1,4)/m       Bmax(2,4)/m;
      Bmax(1,5)/m       Bmax(2,5)/m;
      Bmax(1,6)/m       Bmax(2,6)/m;
      0                 0;
      0                 0;
      0                 0];
% Output matrix C
C=[0 0 1 0 0 0 0 0 0 0;
   0 0 0 0 0 0 0 0 0 1];
D=zeros(2,2);
[V,L]=eig(A);        % Deteremine eigenvectors and eigenvalues
Gpss=ss(A, B, C, D); % Create State Space model
Gp=tf(Gpss);          % Create transfer function: multiple-inpout, single-output

% Controllability and observability analysis
co=ctrb(A,B);
crank=rank(co);
if crank==size(A,1)
    controllable='Yes'
elseif crank<size(A,1)
    controllable='No'
end
ob=obsv(A,C);
orank=rank(ob);
if orank==size(A,1)
    observable='yes'
```

```
elseif orank<size(A,1)
    observable='No'
end

% Similarity transformation
sysbar=ss2ss(Gpss,inv(V));

% ***** Create additional constants for use in nonlinear Simulink model ****
kT=(v*Mx)/Ic;
kF=(v*Mx)/m;
Bmax=Bmax';
Icinv=[1/Ix 0 0; 0 1/Ic 0; 0 0 1/Ic];
M=[Mx;0;0];
cmd=[0;0];
Io=[0;0];
Fg=m*g;
num=[1];
den=[1];
t=(0:.001:5);

% disp('Maximum Fields and Gradients per Coil in following order:');
% disp('Bx  By  Bz  Bxx  Bxy  Bxz  Byy  Byz  Bzz  Bxxx  Bxyx  Bxzx  Bxyy  Bxyz  Bxzz')
% Bmax
% disp('Composite Fields and Gradients at equilibrium:');
% Beq
% disp('A matrix'); A
% disp('B Matrix'); B
% disp('C matrix'); C
% disp('Eigenvectors'); V
% disp('Eigenvalues'); diag(L)
%
% disp(' ');
% disp('Controllable?');
% controllable
% disp('Rank of controllability matrix:  ')
% crank
% disp(' ');
% disp('Observable?');
% observable
% disp('Rank of observability matrix:  ');
% orank
% disp(' ');
% disp('Similarity transformation of State Space Model:  ');
% sysbar
% disp('System Transfer Function'); Gp
```

KALMAN_SYNTH.m

```
% This function uses a range of tuning matrices to derive Kalman Estimators for the
plant specified in state space form.  The frequency responses of the open loop
transfer function for each Kalman Estimator are derived and plotted for evaluation.
The function returns the Kalman filter gain (Kf), the process noise gain mulitplier
Gwgain and the Kalman Estimator (KF) associated with the particular derivation
selected by the user via keyboard input.

% Created by Mark Adams
% Last Modified 06/01/04

% The function uses input variables:

% plant:    State Space representation of minimized continuous plant
% plantd:   State Space representation of minimized discrete plant

% The function returns variables:

% Kfgain:   The Kalman filter gain matrix for the Kalman Estimator selected by user
% Gwgain:   The gain matrix for the process noise used to derive the Kalman Estimator
%           selected by user.
```

```
% Kest:      A SS LTI object representing the Kalman Estimator selected by user
% Lresp:     Array containing Max/Min Singular Values of the Loop Gain of selected
%            Kalman Estimator

function[Kfgain, Gwgain, Rn, Kest, Lresp]=KALMAN_SYNTH(plant,plantd);
global Ts;
clc; close all;
w=logspace(-2,3,400);
Rn=eye(2)*1e-5;              % Sensor noise covariance matrix (scalar)
Qn=plantd.b*plantd.b';  % Process noise covariance matrix (scalar)
[at,bt,ct,dt]=bilin(plantd.a,plantd.b,plantd.c,plantd.d,-1,'Tustin',Ts);

    % *****  Obtain range of values for Process Noise (PN) Gain Matrix ********
PNagain='y';
while PNagain=='y'
    Gw_range1=PN_GAIN(plantd.b);
    if Gw_range1==0
        return
    end
    [range_len, rows]=size(Gw_range1);
    if range_len==1      % if only one element in range of PN Gain matrices, do not
evaluate
        Gw_range2=Gw_range1;
    else            % if more than 1 element in Gw_range
        Gw_range2=EVAL_LOOP_RESP(plantd,Gw_range1,Qn,Rn);
    end                 % if s(1)==1
    PNagain=prompt('Repeat Gain Matrix selection with different values?  ');
end     % while PNagain = 'y'

% **** Store selected PN Gain Matrices and Kalman Filter Loop Gain responses ****
[range_len,rows]=size(Gw_range2);
Lo_range=zeros(2,400,range_len);        % Preallocated array to hold selected Loop
Gain Responses
count=0;
for i=1:range_len       % Store Loop gain response in bilinear form for display
    L=lqe(at,diag(Gw_range2(i,:)),ct,Qn,Rn);
    LGe=ss(at,L,ct,dt);
    Lo=sigma(LGe,w);
    Lo_range(:,:,i)=Lo;
end;            % for i=1:range_len

% ***** Plot Loop Gain Responses and select candidate Kalman Estimators ******
holder=PLOT_RESPONSES(Lo_range, Gw_range2);

% ***** If 1 or More Candidate Loop Gain Responses Selected, Plot on Same Graph ******
if holder==0
    Kfgain=0;
    Gwgain=0;
    Kest=0;
    Lresp=0;
    disp(' ');
    disp('No Kalman Estimator Selected');
else    %then holder > 0
    symb=char('-',':','-.','--');
    symbtext=char('1: solid','2: dotted','3: dash-dot','4: dashed');
    close all;
    disp(' ');
    disp('Indices of selected responses and corresponding line styles:');
    symbtext(1:length(holder),:)

% ***** Plot Candidate Responses ******
    for i=1:length(holder)
        k=holder(i);
        loglog(w,Lo_range(:,:,k),symb(i,:));hold on;
        title('Max/Min Singular Values for selected Kalman Loop Gains');
    end     % for i=1:length(holder)
    ylabel('Magnitude');
    xlabel('Frequency (rad/s)');hold off;

% ***** Select Desired Kalman Estimator From Candidate Responses ******
    beep;
```

```
    msg='Select from candidate response(s)?  (Y/N)  ';
    select=prompt(msg);
    if select=='y'
        beep;
        disp(' ');
        if length(holder)==1
            ind=1;
        else
            ind=input('Enter index number above corresponding to desired response:
');
        end     % if range_len==1
        p=holder(ind);

% ** Except for Response curve, assign function output variables in DT form vs.
Bilinear
        Gwgain=diag(Gw_range2(p,:));
        Kfgain=dlqe(plantd.a,Gwgain,plantd.c,Qn,Rn);
        Ae=plantd.a-Kfgain*plantd.c;
        Be=[plantd.b Kfgain];
        Ce=eye(4);
        De=zeros(4,4);
        Kest=ss(Ae,Be,Ce,De,Ts);
%         L=lqe(at,diag(Gw_range2(p,:)),ct,Qn,Rn);
%         LGe=ss(at,L,ct,dt);
        Lresp=Lo_range(:,:,p);
        loglog(w,Lo_range(:,:,p));
        ylabel('Magnitude');
        xlabel('Frequency (rad/s)');grid;hold off;
        title('Max/Min Singular Values for Selected Kalman Loop Gain');
    else    % select = n
        Kfgain=0;
        Gwgain=0;
        Kest=0;
        Lresp=0;
        disp(' ');
        disp('No Kalman Estimator Selected');
    end     %if select = 'y'
end     % if holder > 0
```

PN_GAIN.m

```
% This function creates a process noise gain matrix for the Kalman Estimator

% Created by: Mark Adams
% Last Modified:  03/17/04

function[Gw]=PN_GAIN(B);
clc;
[m,p]=size(B);

% *****  Determine values for Process Noise Gain Matrix ********
disp('  ');
msg='Choose method to select element values for Process Noise Gain Matrix';
choice=menu(msg,'Default gain matrix--diag [100 100 100 100]','Identity Matrix Scaled
Up or Down',...
    'Identity Matrix Scaled over Range', 'Scale individual rows of Identity
Matrix',...
    'Exit PN_GAIN');
if choice==1
    Gw=100*ones(1,m);
    return
elseif choice==2       % User input values of elements in process gain matrix
    k=input('Enter desired scaling factor for B_Plant:  ');
    Gw=k*ones(1,m);
elseif choice==3    % User Input range of process gain matrices
    disp(' ');
    disp('Default range of scaling factors');
    r=[.1 1 10 100]
```

```
    default = prompt('use default range?  (Y/N):  ');
    if default=='y'
        Gw=PN_DIAG(r,B);
    else
        disp(' ');
        disp('Enter range of up to four (4) scaling factors in brackets.');
        r=input('E.g., [1 10 100 1000]:   ');
        Gw=PN_DIAG(r,B);
    end
elseif choice==4
    disp(' ');
    B
    disp('Enter range of values in brackets corresponding to rows in B_plant');
    Gw=input('E.g., [20 0 100 0]:   ');
elseif choice==5
    Gw=0;
    return
end     %if choice==
```

EVAL_LOOP_RESP.m

```
% This function evaluates loop gain responses of a range of Kalman estimators based on
a range of diagonal process noise gain matrices.  These matrices are derived from a
range of vectors provided as one of the inputs to the function. The function first
requests evaluation criteria from the user, then forms the Kalman estimators and
evaluates their Open Loop Gain responses against the user's criteria.  This function
calls the Matlab function SET_CRITERIA.m.  Note the loop gain responses are derived
using a bilnear transformation of the plant in order that the responses are properly
represented in the w-plane.  The function returns an array of vectors representing the
diagonals of the process noise gain matrices that produce acceptable Kalman
estimators.

% Mark Adams
% Last Modified 03/17/04

function[Gw_out]=EVAL_LOOP_RESP(plantd,Gw,Q,R);
global Ts
w=logspace(-2,3,400);
s=size(Gw);

% *** Set Loop-Shaping criteria to evaluate responses obtained from PN Gain matrices
***
[dir,T,ind]=SET_CRITERIA(w);
[at,bt,ct,dt]=bilin(plantd.a,plantd.b,plantd.c,plantd.d,-1,'Tustin',Ts);
count=0;
holder=0;
for i=1:s(1)
    L=lqe(at,diag(Gw(i,:)),ct,Q,R);
    LGe=ss(at,L,ct,dt);
    Lo=sigma(LGe,w);
    loglog(w,Lo);
    if dir==1       % threshold is a minimum (i.e, if Lo > T, then accept)
        if min(Lo(:,ind))-T>0
            count=count+1;
            indices(count)=i; % Determines which rows in Gw to accept
        else
            count=count;
        end
    elseif dir==2   % threshold is a maximum  (i.e., if Lo < T, then accept)
        if max(Lo(:,ind))-T<0
            count=count+1;
            indices(count)=i;
        else
            count=count;
        end
    end         % if dir==1
    if count==125
        break
    end
```

```
end                 % for i 1:s(1)
if count>0
    disp(' ');
    disp('Responses meeting criteria exceed 125');

    %  **** Resize Gw_range ****
    len=length(indices);  % Indices contains indices of rows accepted from original
Gw_range
    Gw_range_hold=zeros(len,s(2));
    for i=1:len
        Gw_range_hold(i,:)=Gw(indices(i),:);
    end
    Gw_out=Gw_range_hold;
    s=size(Gw);
    range_len=s(1);
else
    disp(' ');
    disp('No responses meet criteria');
    range_len=0;
    break
end     %if count > 0
```

SET_CRITERIA.m

```
% *** Set Loop-Shaping criteria to evaluate responses obtained from PN Gain matrices
****
%
% Mark Adams
% Last Modified 04/14/04

function[threshtype,threshval,freqindex]=SET_CRITERIA(wrange)
    disp('  ');
    disp('Select responses meeting loop shaping criteria.');
    default=prompt('Use default thresholds (20 rad/sec, 0 dB, Min sigma, min
Threshold)?  (Y/N)');
    if default=='y'
        wb=20; mag=0; dir=1;
    else
        disp(' ');
        wb=input('Enter approximate frequency of interest (in rad/sec):  ');
        disp(' ');
        mag=input('Enter response magnitude threshold (in dB):  ');
        disp(' ');
        dir=input('Enter type of threshold: minimum (1) or maximum (2):  ');
    end         %if default=='y'
    I=find(wrange>wb)    % Variables used to locate points in response curves
    freqindex=I(1)-1;
    threshval=10^(.1*mag);
    threshtype=dir;
```

PLOT_RESPONSES.m

```
% This function plots Max/Min Singular Values of Loop Gain Responses and select
candidate Kalman Estimators.
% Created by Mark Adams
% last modified:  04/14/04

function[indices]=PLOT_RESPONSES(resp_range, Gwrange);
[rows,cols,pages]=size(resp_range);
ind=0;
indices=0;
w=logspace(-2,3,400);
if pages==1
    indices=1;
    loglog(w,resp_range(:,:,1));
```

```
    title('Max/Min Singular Values for Kalman Loop Gain');
    ylabel('Magnitude');
    xlabel('Frequency (rad/s)');
else
    for i=1:pages
        plot_no=int2str(i);     %plot no. of the response being displayed
        loglog(w,resp_range(:,:,i));
        legend(plot_no,2);
        beep;
        title('Max/Min Singular Values for Kalman Loop Gain (Pause for keyboard
input.)');
        ylabel('Magnitude');
        xlabel('Frequency (rad/s)');
        r=int2str(4-ind);
        t=int2str(pages-i+1);
        disp('  ');
        sc=strcat(r,' Selections available out of:',t, ' remaining responses.');
        disp(sc);
        Gwrange(i,:)
        msg='Plot acceptable?  (Max of 4 selections allowed.) (Y/N):  ';
        accept=prompt(msg);
        if accept =='y'
            ind=ind+1;
            indices(ind)=i;
            if ind >=4
                break
            end %if ind >=4
        end    %if accept == 'y'
    end     %for i=1:pages
end         % if pages
```

LTR_PROCESS.m

```
% This function uses a given plant and Kalman Estimator, along with a range of tuning
matrices supplied by the user to aid in a Loop Transfer Revovery (LTR) process for a
Linear Quadratic Gaussian (LQG) regulator.  Also provided is a frequency response plot
of the max/min singular values of the Kalman estimator's loop gain transfer function.
The plant is provided in state space form. The function returns the LQR gain matrix,
Krlqr, the input weighting matrix multiplier used to derive that matrix, Rgain, and
the derived LQG regulator, LQGr. LQGr represents the incorporation of the Kalman
estimator and LQR gain into a single regulator block, and is provided in state space
form.  The function takes values for the multiplier, Rgain, from the user via keyboard
input.

% Created by Mark Adams
% Last Modified 04/14/04

% The function uses input variables:
%
% Gwgain:   Process noise gain matrix for Kalman Estimator
% Rn:       Sensor noise covariance matrix for Kalman Estimator
% Lout:     Loop gain response for Kalman Estimator
% plant:    CT Plant model in state space form
% plantd:   DT Plant model in state space form

% The function returns variables:
%
% Rgain:    The multiplier for the input weighting matrix in the LQR calculation,
%           i.e., Rgain*R.
% Kr:       The LQR gain matrix selected for the LQG Regulator
% LQGr:     An LTI object representing the LQG Regulator selected by the user

function[Q,Rgain,Kr,L]=LTR_PROCESS(Gwgain, Rn, Lout, plant, plantd)
global Ts;
% clc;
close all;
w=logspace(-2,3,400);       % Create log frequency range for responses
```

```
[at,bt,ct,dt]=bilin(plantd.a,plantd.b,plantd.c,plantd.d,-1,'Tustin',Ts);
plantw=ss(at,bt,ct,dt,Ts);  % minimized plant in bilinear form
Qb=eye(4);
Rb=eye(2);                  % Base value for Input weighting matrix
Qn=plantd.b*plantd.b';                  % Process noise covariance matrix for Kalman
Estimator
ts=.8;
xo=[0 0 0 0]';

% %*******   Determine whether to use default values or user input ********
msg='Choose Values for Diagonal of State Weighting Matrix, Q:';
choice=menu(msg,'Default Q=C(transpose)*C','Enter Multiplier Qgain * [1 1 1 1]',...
    'Enter Diagonal, e.g., [10 10 100 100]', 'Exit LTR_PROCESS');
if choice==4
    return
elseif choice==1
Q=plant.c'*plant.c;
elseif choice==2
    Qgain=input('Value of multipier Qgain:  ');
    Q=Qb*Qgain;
elseif choice==3
    Qdiag=input('Enter diagonal in square brackets, e.g., [10 10 100 100]    ');
    Q=diag(Qdiag);
end

%*******   Determine whether to use default values or user input ********
msg='Choose Input Weighting Matrix Multiplier Rgain:';
choice=menu(msg,'Default  Rgain=1e-10','Enter single value', 'Exit LTR_PROCESS');
if choice==3
    return
elseif choice==1
    Rgain=1e-10;
    R=Rb*Rgain;
elseif choice==2
    Rgain=input('Value of multipier Rgain:  ');
    R=Rb*Rgain;
end

% ****** Convert system to w plane for plotting responses ******
[y,x,t,Kr]=lqsim(at,bt,Ts,xo,Q,R,ts,'heading');
L_d=lqe(at,Gwgain,ct,Qn,Rn);
Areg=at-bt*Kr-L_d*ct;
Breg=L_d;
Creg=-Kr;
Dreg=zeros(2);
[Alp,Blp,Clp,Dlp]=series(Areg,Breg,Creg,Dreg, at,bt,ct,dt);
Lresp=sigma(Alp,Blp,Clp,Dlp,w);
loglog(w,Lout(1,:),'-',w,Lout(2,:),':',w,Lresp(1,:),'--',w,Lresp(2,:),'-.');
title('Max/Min Singular Values of Open Loop Gain Responses');
legend('Max SV Kest Loop Gain','Min SV Kest Loop Gain', 'Max SV System Loop Gain',...
    'Min SV System Loop Gain');

% ******* Produce DT LSVF gain matrix for use in simulation  *********
[y,x,t,Kr]=lqsim(plant.a,plant.b,Ts,xo,Q,R,ts,'heading');

LQSIM.m
function  [u,x,t,K]=lqsim(F,G,T,ic,q1,q2,ts,heading,n)
%       LQ SIMULATOR
%
%       [u,x,t]=lqsim(F,G,T,ic,q1,q2,ts,heading,n)
%
%    CONTINUOUS-TIME PLANT STATE EQUATION : xdot = F*x + G*u
%    SAMPLING PERIOD = T
%    x(t=0) = initial condition = ic
%    q1=Q,q2=R  = LQ cost weight matrices
%    ts     = desired settling time (Pincer's method)
%    heading  = portion of title (eg. 'Q1 = ___')
%    n      = optional number of samples

% clf
[phi,gam]=c2d(F,G,T);
```

```
%Pincer's method
k=ts/T;
alpha=100^(1/k);
phip=alpha*phi;
gamp=alpha*gam;

% disp('The steady-state gain is:')
K=dlqr(phip,gamp,q1,q2);
% disp(' ')
% disp('The closed-loop poles are ')
clpoles=eig(phi-gam*K);
ddamp(clpoles,T);
H=-K;                                          %u = -K*x
phicl=phi-gam*K;
J=[0];

if (nargin==8),
   [u,x]=dimpulse(phicl,ic,H,J,1);
   [n,nn]=size(u);                             %n = number of rows
 else
   [u,x]=dimpulse(phicl,ic,H,J,1,n);           %n is given in function call
end

t=-T:T:(n-2)*T;
t=t(2:n);                                      %the first real output is 2nd one
x=x(2:n,:);
u=u(2:n,:);
% axis([0 n*T -1.6 1])
% plot(t,x(:,1),'ro',t,x(:,2),'bx'),grid
% hold on
% plot(t,x(:,1),'r-',t,x(:,2),'b-')
% zohplot(t',u(:,1),'g--')
% ylabel('STATES, CONTROL INPUT'),...
% xlabel('TIME   (SEC)'),...
% title(['ZERO-INPUT PLOTS FOR ' heading ' AND R = ' num2str(q2)] )
% text(6,0.9,'---o--- X1')
% text(6,.75,'---x--- X2')
% text(6,.55,'------- U')

% hold off

% format compact
% satex
% T=1;
% ts=inf;
% q1=[1 0;0 0];
% h='q1=[1 0;0 0]';
% [y,x,t]=lqsim(A,B,T,[1;0],q1,.05,ts,h);
```

STEP_RESPONSE.m

```
% This function takes a Linear Quadratic Guassian regulator (composed of a Kalman
estimator and LQR state variable feedback matrix) and evaluates the time responses of
a nonlinear plant model using that LQG regulator.
%
% Created by Mark Adams
% Last Modified 04/21/04
%
% The function uses input variables:
%
% Plant:    State space form of minmized continuous plant
% Plant:    State space form of minmized discrete plant
% Kr:       Optimal state feedback coefficient

%     The function returns variables:

% resp_values   array containing percent overshoot, settling time
```

```
%               and steady state response for the nonlinear model and magnitude
%               of the maximum deviation of the control input from equilibrium.

function[resp_values, responses, t]=STEP_RESPONSE(Kr, plant, plantd, int_cntrl);

%     ********** variables needed on Matlab workspace ***************
global cmd Kin Nbar K KI simout default d_out t1 m Izinv bpert fd Ieq;

fd=0;    %   switch for additive disturbance force in Simulink model
bpert=0 %  switch for additive b-field perturbations in Simulink model
Izinv;
[am,bm,cm,dm]=ssdata(plant);
[ad,bd,cd,dd]=ssdata(plantd);

if int_cntrl=='y';
    KI=Kr(:,1:2);
    K=Kr(:,3:6);
    [Nx, Nu, Nbar]=refi(ad,bd,cd,K);
    Kin=eye(2);
else
    cl_reg=ss(am-bm*Kr,bm,cm,dm);
    Kin=inv(dcgain(cl_reg));
end
deg=[1.0 -1.0]
pitch=deg*pi/180;
z_pos=[.002 -.002];
len=length(deg);
simlength=2.0;              % Length of simulation in seconds
t=(0:.0001:simlength);    %Create time scale
lent=length(t);
resp_values=zeros(2*len^2,5);   %preallocate arrays to hold values
responses=zeros(lent,2*len^2);  %preallocate arrays to hold entire responses
incr=0;
for h=1:len               %sequence through all set points
    for j=1:len
        incr=incr+1;
        row=incr+(incr-1);
        cmd=[pitch(h);z_pos(j)];
        resp_values(row:row+1,1)=cmd

        %  *********** Obtain step response of non-linear system *******
        if int_cntrl=='y'
            sim('DT_nonlin_int');
        else
            sim('DT_exp_nonlin');
        end
        responses(:,row:row+1)=simout;

        % ****** Determine final value of step response of system ****
        if simout(lent,:)==[0 0]   % Avoid division by zero in following calculations
            simout(:,lent)=[1e-12;1e-12];
        elseif simout(lent,1)==0
            simout(lent,1)=1e-12;
        elseif simout(lent,2)==0
            simout(lent,2)=1e-12;
        end
        resp_values(row:row+1,4)=simout(lent,:)';

        % **** Determine percent overshoot of system step response ******
        if cmd(1,1)==0
            resp_values(row,2)=max(abs(min(simout(:,1))));
        elseif cmd(1,1)<0.00
            resp_values(row,2)=((min(simout(:,1))-cmd(1,1))/cmd(1,1))*100;
        elseif cmd(1,1)>0.00
            resp_values(row,2)=((max(simout(:,1))-cmd(1,1))/cmd(1,1))*100;
        end     %end cmd(1,1))==0
        if cmd(2,1)==0
            resp_values(row+1,2)=max(abs(min(simout(:,2))));
        elseif cmd(2,1)<0.00
            resp_values(row+1,2)=((min(simout(:,2))-cmd(2,1))/cmd(2,1))*100;
        elseif cmd(2,1)>0.00
```

```
            resp_values(row+1,2)=((max(simout(:,2))-cmd(2,1))/cmd(2,1))*100;
        end  %end cmd(2,1))==0

        % **** Determine settling time (within 2% of final value) of system
        %  For pitch response
        count=1;
        enter=0;
        for tstep=1:lent
            if abs(simout(tstep,1)-resp_values(row,4))<=.02*abs(resp_values(row,4))
                if enter==0
                    cross(count)=.0001*tstep;
                end
                enter=1;
            end
            if enter==1
                if abs(simout(tstep,1)-resp_values(row,4))>.02*abs(resp_values(row,4))
                    count=count+1;
                    enter=0;
                end
            end         % if enter==1
        end     %for tstep =1:lent
        if count>length(cross)
            disp('Settling time exceeds test interval');
        else
            resp_values(h,3)=cross(count);
        end         %If count>length(cross)
        %  For Position response
        count=1;
        enter=0;
        for tstep=1:lent
            if abs(simout(tstep,2)-
resp_values(row+1,4))<=.02*abs(resp_values(row+1,4))
                if enter==0
                    cross(count)=.0001*tstep;
                end
                enter=1;
            end
            if enter==1
                if abs(simout(tstep,2)-
resp_values(row+1,4))>.02*abs(resp_values(row+1,4))
                    count=count+1;
                    enter=0;
                end
            end     % if enter==1
        end     %for tstep =1:lent
        if count>length(cross)
            disp('Settling time exceeds test interval');
        else
            resp_values(row+1,3)=cross(count);
        end     % If count>length(cross)

        % **** Determine magnitude of maximum deviation of control input to system
****
        resp_values(row,5)=max(max(simin(:,1))-Ieq(1,1)),min(simin(:,1)-Ieq(1,1));
        resp_values(row+1,5)=max(max(simin(:,2))-Ieq(2,1)),min(simin(:,2)-Ieq(2,1));
    end     % for j=1:len

    end     % for h=1:len
    step_heading='Command Input--Percent OS--Settling Time--Final Value--Max Input
Deviation';
```

STIFFNESS_TEST.m

```
% This function takes plant state equations & a range of state variable feedback
constants
% and derives new transfer functions for the linearize plant.  The function then
% uses the LTI transfer functions to determine appropriate cmd inputs for the Simulink
% nonlinear lsvf model. The nonlinear model is then run using the cmd settings, the
```

```
% corresponding state variable feedback constants and a known disturbance force
% that simulates an aerodynamic disturbance.  The stiffness quotient, i.e. the ratio
% of the force to the displacement it produces, for each cmd setting is returned.
% which are tested with step and impulse functions.
%
% Created by Mark Adams
% Last Modified 04/20/04

% The function uses input variables:
%
% plant:    state space form of minimized CT plant
% plantd:   state space form of minimized DT plant
% Kr:       LSVF Matrix

% The function returns variables:
%
% fdresp    An array containing the stiffness quotients for the nonlinear system
%           for a range of lsvf constants and cmd inputs.

function[resp_values, responses,t, fd_out, td_out]=STIFFNESS_TEST(Kr,plant, plantd,
int_cntrl);
global cmd Kin simout simin regout m io bpert fd Ieq Ic fd_time_seq fd_ampl_seq K KI;

close all;
t=(0:.0001:2.0);                    % Create time scale for simulation
lent=length(t);
hflent=floor(lent/2);
z_pos=[-.002 0 .002];
pitch=[-1.0 0 1.0]*pi/180;
len=length(z_pos);
resp_values=zeros(2*len^2,5);       % preallocate array to hold stiffness responses
responses=zeros(lent,4*len^2);

fd=0;           % switch for additive disturbance force in Simulink model
bpert=0;        % switch for additive b-field perturbations in Simulink model

fd_time_seq=[0.0 0.5000 0.5100 0.5200 0.5300 0.5500 0.5600 0.5700 0.6100...
          0.6500 0.6900 0.7300 0.7500 0.7600 0.7700 0.7800 0.8000 0.8100...
          0.8200 0.8600 0.9000 0.9400 0.9800];

fd_ampl_seq=[0 0  1 4 6 9 10 10 6 3 1 0 0 -1 -4 -6 -9 -10 -10 -6 -3 -1 0];
[am,bm,cm,dm]=ssdata(plant);
[ad,bd,cd,dd]=ssdata(plantd);

if int_cntrl=='y';
    KI=Kr(:,1:2);
    K=Kr(:,3:6);
    [Nx, Nu, Nbar]=refi(ad,bd,cd,K);
    Kin=eye(2);
else
    cl_reg=ss(am-bm*Kr,bm,cm,dm);
    Kin=inv(dcgain(cl_reg));
end

i=0;
for h=1:len         % sequence through range of pitch set points
    for j=1:len     % sequence through range of z_pos set points
        i=i+1;
        k=i+(i-1);
        p=i+3*(i-1);
        cmd=[pitch(h);z_pos(j)];
        % Obtain response of non-linear system
        fd=0;
        if int_cntrl=='y'
            sim('DT_nonlin_int');
            fv=simout(lent,:);        % obtain final value without Fd
            no_dist_resp=simout;
            fd=1;
            plot(tout,simout);hold on;
            sim('DT_nonlin_int');   % obtain output with Fd
            fd_out=fdout;
```

```
            td_out=tdout;
        else
            sim('DT_exp_nonlin');
            fv=simout(lent,:);          % obtain final value without Fd
            no_dist_resp=simout;
            fd=1;
            sim('DT_exp_nonlin');    % obtain output with Fd
            fd_out=fdout;
            td_out=tdout;
        end

        % Calculate pos neg and avg. stiffness quotients (SQ) for pitch
        a=Ic/(max(simout(hflent:lent,1))-fv(1));
        b=Ic/(min(simout(hflent:lent,1))-fv(1));
        c=(abs(a)+abs(b))/(2);
        resp_values(k,2:5)=[a b c fv(1)];

        %Same for z_pos
        a=m/(max(simout(hflent:lent,2))-fv(2));
        b=m/(min(simout(hflent:lent,2))-fv(2));
        c=(abs(a)+abs(b))/(2);
        resp_values(k+1,2:5)=[a b c fv(2)];

        resp_values(k:k+1,1)=cmd;
        responses(:,p:p+1)=no_dist_resp;
        responses(:,p+2:p+3)=simout;
    end
end         % outer for loop
```

B_PERTURB.m

```
% This function takes plant state equations & a state variable feedback matrix and
derives new transfer functions for the linearize plant.  The function then uses the
transfer functions to determine appropriate cmd inputs for the Simulink nonlinear
model. The nonlinear model is then run using the same cmd settings used for the
unperturbed time responses only with the B field coefficients perturbed.

% Created by Mark Adams
% Last Modified 04/20/04

% The function uses input variables:
%
% plant:    State space form of minimized CT plant
% plantd:   State space form of minimized DT plant
% Kr        State variable feedback matrix derived using lqsim.m

% The function returns variables:
%
% fdresp    An array containing the stif+fness quotients for the nonlinear system
%           for a range of lsvf constants and cmd inputs.

function[resp_values, responses, t]=B_PERTURB(Kr, plant, plantd, int_cntrl);

%variables needed on Matlab workspace:
global cmd Kin simout default d_out t1 m Icinv bpert fd Ieq K KI;
t=(0:.0001:2.0);                  % Create time scale for simulation
lent=length(t);
no_runs=5;                      % number of sample runs
z_pos=[.002];
pitch=[1.0]*pi/180;
len=length(z_pos);
resp_values=zeros(2*len,9);        % preallocate array to hold b-field perturbed time
response values
responses=zeros(4,lent,len^2);     % preallocate array to hold b-field perturbed
responses
fv=zeros(2,no_runs);
perc_os=zeros(2,no_runs);
fv=zeros(2,no_runs);
```

```
inp=zeros(2,no_runs);
hold_pitch=zeros(no_runs,lent);
hold_z_pos=zeros(no_runs,lent);
fd=0;            % switch for additive disturbance force in Simulink model
bpert=0;         % switch for additive b-field perturbations in Simulink model
[am,bm,cm,dm]=ssdata(plant);
[ad,bd,cd,dd]=ssdata(plantd);
if int_cntrl=='y';
    KI=Kr(:,1:2);
    K=Kr(:,3:6);
    [Nx, Nu, Nbar]=refi(ad,bd,cd,K);
    Kin=eye(2);
else
    cl_reg=ss(am-bm*Kr,bm,cm,dm);
    Kin=inv(dcgain(cl_reg));
end

i=0;
for h=1:len          % sequence through range of pitch set points
    for j=1:len      % sequence through range of z_pos set points
        i=i+1;
        k=i+(i-1);
        p=i+3*(i-1);
        cmd=[pitch(h);z_pos(j)];
           if int_cntrl=='y'
       sim('DT_nonlin_int');
   else
       sim('DT_exp_nonlin');
   end
   no_perturb_resp=simout;
        for n=1:no_runs
            % Determine step response of non-linear system w/B-field perturbations
            bpert=(0.02*(2*rand-1));    % Generate random factor between -2 and +2 %
               if int_cntrl=='y'
       sim('DT_nonlin_int');
   else
       sim('DT_exp_nonlin');
   end
            % Determine final value of step response of non-linear system
            if simout(lent,:)==[0 0]   %Avoid division by zero in following
calculations
                simout(lent,:)=[1e-12 1e-12];
            elseif simout(lent,1)==0   %Avoid division by zero in following
calculations
                simout(lent,1)=1e-12;
            elseif simout(lent,2)==0   %Avoid division by zero in following
calculations
                simout(lent,2)=1e-12;
            end %if simout
            fv(:,n)=simout(lent,:)';

            % **** Determine percent overshoot of system step response ******
            if cmd(1,1)==0
                perc_os(1,n)=max(abs(min(simout(:,1))));
            elseif cmd(1,1)<0.00
                perc_os(1,n)=((min(simout(:,1))-cmd(1,1))/cmd(1,1))*100;
            elseif cmd(1,1)>0.00
                perc_os(1,n)=((max(simout(:,1))-cmd(1,1))/cmd(1,1))*100;
            end     %end cmd(1,1))==0
            if cmd(2,1)==0
                perc_os(2,n)=max(abs(min(simout(:,2))));
            elseif cmd(2,1)<0.00
                perc_os(2,n)=((min(simout(:,2))-cmd(2,1))/cmd(2,1))*100;
            elseif cmd(2,1)>0.00
                perc_os(2,n)=((max(simout(:,2))-cmd(2,1))/cmd(2,1))*100;
            end  %end cmd(2,1))==0

            % **** Determine settling time (within 2% of final value) of system
            %  For pitch response
            count=1;
            enter=0;
```

```
            for tstep=1:lent
                if abs(simout(tstep,1)-fv(1,n))<=.02*abs(fv(1,n))
                    if enter==0
                        cross(count)=.0001*tstep;
                    end
                    enter=1;
                end
                if enter==1
                    if abs(simout(tstep,1)-fv(1,n))>.02*abs(fv(1,n))
                        count=count+1;
                        enter=0;
                    end
                end          % if enter==1
            end      %for tstep =1:lent
            if count>length(cross)
                disp('Settling time exceeds test interval');
            else
                st(1,n)=cross(count);
            end         %If count>length(cross)

            count=1;
            enter=0;
            for tstep=1:lent
                if abs(simout(tstep,2)-fv(2,n))<=.02*abs(fv(2,n))
                    if enter==0
                        cross(count)=.0001*tstep;
                    end
                    enter=1;
                end
                if enter==1
                    if abs(simout(tstep,2)-fv(2,n))>.02*abs(fv(2,n))
                        count=count+1;
                        enter=0;
                    end
                end      % if enter==1
            end      %for tstep =1:lent
            if count>length(cross)
                disp('Settling time exceeds test interval');
            else
                st(2,n)=cross(count);
            end      % If count>length(cross)

            % ** Determine magnitude of maximum deviation of control input to system
****
%             inp(1,n)=max(max(simin(:,1))-Ieq(1,1)),min(simin(:,1)-Ieq(1,1));
%             inp(2,n)=max(max(simin(:,2))-Ieq(2,1)),min(simin(:,2)-Ieq(2,1));

            hold_pitch(n,:)=simout(:,1)';
            hold_z_pos(n,:)=simout(:,2)';

        end      % end for n=1:5

        %  Copy command input to first column of values matrix
        resp_values(k:k+1,1)=cmd;

        %  Calculate mean and std deviation of values over n samples and copy
        %  to output matrix
        resp_values(k,2)=mean(perc_os(1,:));
        resp_values(k+1,2)=mean(perc_os(2,:));
        resp_values(k,3)=std(perc_os(1,:));
        resp_values(k+1,3)=std(perc_os(2,:));

        resp_values(k,4)=mean(st(1,:));
        resp_values(k+1,4)=mean(st(2,:));
        resp_values(k,5)=std(st(1,:));
        resp_values(k+1,5)=std(st(2,:));

        resp_values(k,6)=mean(fv(1,:));
        resp_values(k+1,6)=mean(fv(2,:));
        resp_values(k,7)=std(fv(1,:));
        resp_values(k+1,7)=std(fv(2,:));
```

```
        resp_values(k,8)=mean(inp(1,:));
        resp_values(k+1,8)=mean(inp(2,:));
        resp_values(k,9)=std(inp(1,:));
        resp_values(k+1,9)=std(inp(2,:));

        %  Calculate mean and std deviation of responses over all samples and copy
        %  to output matrix
        responses(1,:,i)=mean(hold_pitch);
        responses(2,:,i)=mean(hold_z_pos);
        responses(3,:,i)=std(hold_pitch);
        responses(4,:,i)=std(hold_z_pos);
        responses(5:6,:,i)=no_perturb_resp';

    end                 % end for j=1:len
end                     % end for h=1:len
```

PROMPT.m

```
% This function takes a prompt message and returns a 'y' for
% an affirmative answer and a 'n' for a negative one.

% Created by Mark Adams
% Date last modified: 03/10/04

function[answer]=prompt(question);
answer=input(question,'s');
% if answer==' '
%     answer='y';
% end
answer=lower(answer);
```

TWO_DOF_TEST.m

```
% This program is a regulator synthesis and validation tool
% for the 2-DOF version of the MSBS.
%
% Created by Mark Adams.
% Last Modified:  05/25/04

format; clc; clear; close all;
% Variables needed on Matlab Workspace
global cmd Kin Ieq bpert fd m Ic Ts K KI Nbar;
Ts=.001;

% ****** Derive Analytic State Space Model of Plant from Eqns *******
Two_DOF_Plant;
% Izinv=sum([0 0 1]*Icinv);

% ***** Extract States Relating to Pitch and Vertical Movement/Create DT
Model**********
[Am,Bm,Cm,Dm]=ssselect(A,B,C,D,[1 2],[1 2],[1 3 7 10]);
minsys=ss(Am,Bm,Cm,Dm);
sysd=c2d(minsys,Ts);
[ad,bd,cd,dd]=ssdata(sysd);

Rn=eye(2)*1e-5;          % Sensor noise covariance matrix (scalar)
Qn=bd*bd';               % Process noise covariance matrix (scalar)
Gwf=[12500];
G_len=length(Gwf);
R_weights=[.1];
R_len=length(R_weights);
Q_weights=[500 500 500 500];
w_len=length(Q_weights);
% Qrange=zeros(w_len-1,4);
```

```
count=0;
for h=1:w_len
     for i=1:w_len
         for j=1:w_len
             for k=1:w_len
               count=count+1;
               Qrange(count,:)=[Q_weights(1) Q_weights(2) Q_weights(3) Q_weights(4)];
             end
         end
     end
end
[rows,cols]=size(Qrange);
loc=input('Enter location/date of test: NASA_24_mar,  srl...,  or h...  ','s');
current_start=input('Enter starting no. of sequence  ');
ov_len=G_len*R_len* rows;
last=current_start+ov_len-1;
count=0;
disp(['Total number of iterations:  ', int2str(ov_len)'']);
disp(' ');

%*********  Use Integral control with added zero? ***********
message=('Use integral control?');
ans=prompt(message);

for p=1:G_len   % Derive LQG regulators for range of Kalman process noise, LQR input
and state weighting values
    tic;
    dvals=Gwf(p)*ones(1,4);
    Gw=diag(dvals);
    L=dlqe(ad,Gw,cd,Qn,Rn);
    Ae=ad-L*cd;
    Be=[bd L];
    Ce=eye(4);
    De=zeros(4,4);
    Kest=ss(Ae,Be,Ce,De,Ts);
    Rest=Rn;
    for q=1:R_len           % Input weighting matrix values
        R=R_weights(q)*eye(2);
        for i=1:rows        % State weighting matrix values
            count=count+1;
            Q=diag(Qrange(i,:));
            ts=.8;
            xo=[0 0 0 0]';
            a=Am;
            b=Bm;
            if ans=='y'
                a=[eye(2) Cm;zeros(4,2) Am];
                b=[zeros(2);minsys.b];
                xo=[0 0 0 0 0 0]';
                newdiag=[500 500 Qrange(i,:)];
                Q=diag(newdiag);
            end
            [y,x,t,Klqr]=lqsim(a,b,Ts,xo,Q,R,ts,'heading');

            [step_values, step_resp,tout]=STEP_RESPONSE(Klqr, minsys, sysd, ans);
            plot(tout,step_resp);grid;
            title('Response from Step Response');

            [stf_values, stf_resp, tout]=STIFFNESS_TEST(Klqr, minsys,sysd, ans);
            figure;
            plot(tout,stf_resp);grid;
            title('Response from Stiffness Tests');

            [bp_values, bp_resp, tout]=B_PERTURB(Klqr,minsys,sysd, ans);
            figure;
            plot(tout,bp_resp);grid;
            title('Response from B-Field Perturbation Test.');

            % ****** Save Design Results  ************
            no=int2str(current_start+(count-1));
            results=strcat('results_',loc,no);
```

```
            save (results,'Gw','Rest','Q','R','Kin','Kest',
'Klqr','step_values','step_resp','stf_values','stf_resp','bp_values','bp_resp','tout')
;
            save (results,'Gw','Rest','Q','R','step_values','step_resp','stf_values',
'bp_values','tout');
             save (results,'Gw','Rest','Q','R','step_values','step_resp','tout');

            elapsed_time=toc;
            if count==1
                first_pass=elapsed_time;
                total_est=ov_len*first_pass;
            end
            co=int2str(count);
            tot=int2str(ov_len);
            est_time_remaining=total_est-elapsed_time;
            elapsed_time
            est_time_remaining
            msg=strcat(co, '_iterations completed out of_', tot);
            disp(msg);
        end
    end
end
```

SORT_RESULTS.m

```
% Created by Mark Adams
% Last Modified 05/25/04

clear
% ************* User Input ******************
loc=input('Enter location as expressed in data file name: nasa, srl, etc... ', 's');
diafilename=strcat(loc, '_controller_test');

first=input('Enter exp start no.  ');
last=input('Enter no. of last experiment to be looked at  ');
OS=input('Eneter max allowable percent overshoot  ');
ST=input('Enter max allowable settling time (sec)  ');
Id=input('Enter maximum allowable deviation of input current (A)  ');
int_cntrl=prompt('Was Integral Control Model Used?');
len=last-first+1;
clc;
diary (diafilename)

% ********** Generate Header **************
append=int2str(first);
datafilename=strcat('results_',loc, append);
load (datafilename);
sim_run_length=length(tout)/1e4
disp(' ');

% ******* obtain response values from .mat files and place in table *****************
if int_cntrl=='y'
    resp_values=zeros(2*len, 13);
else
    resp_values=zeros(2*len, 11);
end
for i=1:len
    row=i+(i-1);
    append=int2str(first+(i-1));
    datafilename=strcat('results_',loc, append);
    load (datafilename);
    Gw=diag(Gw);
    R=diag(R);
    Qw=diag(Q)';
    resp_values(row,1:2)=[Gw(1) R(1)];
    resp_values(row+1,1:2)=[Gw(1) R(1)];
    if int_cntrl=='y'
        resp_values(row,3:8)=Qw;
    resp_values(row+1,3:8)=Qw;
```

```
    resp_values(row:row+1,9:13)=step_values(1:2,:);
else
    resp_values(row,3:6)=Qw;
    resp_values(row+1,3:6)=Qw;
    resp_values(row:row+1,7:11)=step_values(1:2,:);
end
    plot(tout,step_resp(:,1), tout,step_resp(:,2), 'k:');hold on;
end
legend('Command Input: .0175 rad', 'Command Input .002 m');
title('Step Responses for Range of Controllers');
ylabel('Amplitude');
xlabel('Time (sec)');
format short g;
resp_values
threshold=THRESHOLDS(resp_values,OS,ST,Id);
s=size(threshold);
disp(['Thresholds: % OS--' num2str(OS) 'Settling Time:' num2str(ST) 'Max Input
Deviation: ' num2str(Id) 'No. Responses Qualified:' int2str(s(1))]);

% group pitch and position commands for like parameters
count=0;
adjacent_rows=zeros(2,11);
for i=2:s(1)
    if threshold(i,7)==2.0
        if threshold(i-1,7)==1.0
            if (threshold(i,1:6)-threshold(i-1,1:6))==0
                count=count+1;
                row=count+(count-1);
                adjacent_rows(row,:)=threshold(i-1,:);
                adjacent_rows(row+1,:)=threshold(i,:);

            end
        end
    end
end

s=size(adjacent_rows);
% disp(['Total number of iterations: 'int2str(ov_len)']);

disp(['Number of Paired pitch/position responses with same paramters: '
int2str(s(1))]);
adjacent_rows_short=zeros(s(1),10);
adjacent_rows_short(:,1:9)=adjacent_rows(:,1:9);
adjacent_rows_short(:,10)=adjacent_rows(:,11);
disp('Time Domain Response Values');
disp('           Gw          Rlqr          Qlqr
Cmd       %OS  ST Max I dev');
adjacent_rows_short
diary off;
format short;

% hold_SQ=zeros(1,10);
% max_SQ=0;
% hold_BP=zeros(1,7);
% hold_BP(1,7)=OS;

%     max_OS=max(step_values(:,2));
%     max_Iin=max(step_values(:,5));
%     max_SQ=max(stf_values(:,4));
%     I=find(stf_values(:,4)==max_SQ);
%     if max_SQ>hold_SQ(1,8);
%         hold_SQ(1,1:2)=[Gw(1) R(1)];
%         hold_SQ(1,3:6)=Qw;
%         hold_SQ(1,7)=stf_values(I,1);
%         hold_SQ(1,8)=max_SQ;
%         hold_SQ(1,9)=max_OS;
%         hold_SQ(1,10)=max_Iin;
%     end

%     max_OS=max(bp_values(:,2));
%     if max_OS<=hold_BP(1,7)
```

```
%          hold_BP(1,1:2)=[Gw(1) R(1)];
%         hold_BP(1,3:6)=Qw;
%         hold_BP(1,7)=max_OS;
%     end

% disp('Maximum value stiffness coefficient');
% disp('           Gw         Rlqr         Qlqr
Cmd       SQ      %OS       In Dev')
% hold_SQ
% disp('Minimum percent OS (of the higher valued output) with B-Field Perturbation');
% disp('           Gw         Rlqr         Qlqr
%OS')
% hold_BP
```

THRESHOLDS.M

```
% This function examines ranges of step response data (percent overshoot, settling
time, steady state value, max impulse response, impulse settling time, and steady
state value) for values that meet certain thresholds.  The function then returns those
values along with the corresponding values for weighting matrices that produced them.
%
% Created by Mark Adams
% Last Modified:  04/20/04

% This function uses the following inputs:
% trstep:   A 3-D array containing step responses of linear and non-linear systems
%           created using a range of linear state variable feedback (lsvf)
coefficients.
% perc_os   The threshold for percent overshoot
% st        The threshold for settling time
% Iin_dev   The threshold for max input current deviation

% The function returns the variables:
% sr_thresh: An array whose columns represent the time response values contained
%            in the input array 'trstep' that meet designated threshold criteria. The
ime
            response values represented are Percent OS, Settling Time, Final Value
along
            with the values selected are the corresponding values of Q and R that
            produced them. sr_thresh is three dimensional, e.g., sr_thresh(x,y,z),
where
            x is no. of values meeting the criteria for any of the three responses of
            interest, y is the no. of tr values and their corresponding R&Q, z is the
            number of desired regulator outputs for which time responses were
measured.

function[sr_thresh]=THRESHHOLDS(trstep, perc_os, st, Iin_dev);

s=size(trstep);   % size of array containing step responses
for i=1:s(1)/2    % convert command input and final values to degrees and millimeters
    row=i+(i-1);
    trstep(row,7)=trstep(row,7)*180/pi;
    trstep(row,10)=trstep(row,10)*180/pi;
    trstep(row+1,7)=trstep(row+1,7)*1000;
    trstep(row+1,10)=trstep(row+1,10)*1000;
end

% *** Select step responses meeting threshold criterion for max deviation of input
current ******
first_count=1;
for h=1:s(1)        %Examine responses for max deviation of input current
    if abs(trstep(h,11))<=Iin_dev
        first_hold(first_count,:)=trstep(h,:);
        first_count=first_count+1;
    end
end
s=size(first_hold);
```

```
% ****** Select responses meeting threshold criterion for percent overshoot
************
second_count=1;
for i=1:s(1)
    if abs(first_hold(i,8))<=perc_os
        second_hold(second_count,:)=first_hold(i,:);
        second_count=second_count+1;
    end
end

% ********* Select responses meeting threshold criterion for settling time
************
s=size(second_hold);
third_count=1;
for j=1:s(1)
    if second_hold(j,9)<=st
        third_hold(third_count,:)=second_hold(j,:);
        third_count=third_count+1;
    end
end
sr_thresh=third_hold;
```

APPENDIX E

CATALOGUE OF SCRIPTS FOR OPTIMAL CONTROLLER SYNTHESIS AND VALIDATION

Introduction

This appendix provides a catalogue of the Matlab scripts written to synthesize the different optimal controllers produced in support of this thesis. The scripts are grouped in tables according to controller type and design approach. The tables also show the interrelationships between the scripts by listing the calling script(s) for each entry.

The appendix is divided into two parts, corresponding to the 1-DOF and 2-DOF systems described in Chapters V and VI, respectively. For each of the two systems, there are two or three tables corresponding to the different controller designs undertaken, e.g., LQR, LQG, LTR.

Controller Synthesis and Testing for 1-DOF System

Chapter V of the thesis introduced the concepts of optimal control theory and described how that theory was adapted to the problem of magnetic levitation in the MSBS. Several controller designs were tried to gain familiarity with optimal control concepts and begin developing a set of Matlab-based tools to automate the controller design process.

Table I-E lists the Matlab scripts used to develop an automated process for synthesizing a linear quadratic regulator (LQR) for a 1-DOF setup. The process is initiated by invoking the first Matlab script in the table, ONE_DOF_LQR.m.

TABLE I-E
MATLAB SCRIPTS USED FOR THE 1–DOF LQR DESIGN

Matlab Script	Description	Called by
ONE_DOF_LQR.m	Main program that coordinates the overall process.	-----
ONE_DOF_PLANT.m	Uses values for physical constants and creates linear state space model of 1-DOF plant. Also derives nonlinear constants for use in simulations. Places all values on Matlab workspace.	ONE_DOF_LQR.m
PROMPT.m	Utility script that facilitates user queries.	All
QEVAL.m	Takes the specified range limits for the values of Q and R, along with the state space matrix A and input matrix B for the plant, and calculates the corresponding range of state variable feedback values K and repositioned pole values using the LQR function.	ONE_DOF_LQR.m

TABLE I-E, CONT'D

Matlab Script	Description	Called by
CL_RESPONSES.m	Uses range of K-matrices to derive a set of linear closed-loop regulator models, for which step responses are obtained using Matlab command *lsim.m*. Step responses for the same range of K-matrices are also obtained from the nonlinear Simulink model of the regulator.	ONE_DOF_LQR.m
THRESHOLDS.m	Evaluates step responses against user-provided criteria for percent overshoot, settling time and steady state value. Selects those regulator designs meeting criteria.	ONE_DOF_LQR.m
STIFFNESS_TEST.m	Produces range of LQR controllers based on range of LSVF matrices provided as input parameters. Evaluates controllers for stiffness by simulating disturbances in the nonlinear model. Provides so-called stiffness ratios for each controller design.	ONE_DOF_LQR.m
B-PERTURB.m	Evaluates controllers for robustness with respect to plant uncertainty by simulating random variations in plant parameters. Provides comparison of perturbed, non-perturbed step responses.	ONE_DOF_LQR.m

Table II-E lists the Matlab scripts used to develop an automated process for synthesizing a Linear Quadratic Gaussian (LQG) regulator for a 1-DOF setup. The process is initiated by invoking the first Matlab script in the table, ONE_DOF_LQG.m. Note several scripts from the LQR process are reused.

TABLE II-E
MATLAB SCRIPTS USED FOR THE 1-DOF LQG DESIGN

Matlab Script	Description	Called by
ONE_DOF_LQG.m	Main program that coordinates the overall process.	-----
ONE_DOF_PLANT.m	Same as in Table I-E.	ONE_DOF_LQG.m
PROMPT.m	Same as in Table I-E.	All
QEVAL.m	Same as in Table I-E.	ONE_DOF_LQG.m
CL_RESPONSES.m	Same as in Table I-E.	ONE_DOF_LQG.m
THRESHOLDS.m	Same as in Table I-E.	ONE_DOF_LQG.m
STIFFNESS_TEST.m	Same as in Table I-E.	ONE_DOF_LQG.m
B-PERTURB.m	Same as in Table I-E.	ONE_DOF_LQG.m

TABLE II-E, CONT'D

Matlah Script	Description	Called By
LQG_TEST.m	Synthesizes a range of LQG regulators based on a range of LSVF matrices specified in its input parameters and a single Kalman estimator design. The time response of each regulator is obtained for user-specified range of set points using a nonlinear Simulink model. Returns a single array containing time response values and the maximum deviation of the control signal.	ONE_DOF_LQG.m
STIFFNESS_TEST.m	Same as in Table I-E.	LQG_TEST.m
B_PERTURB_LQG.m	Similar to B_PERTURB.m except that this script obtains the perturbation response for only one LQG regulator provided as input parameter.	LQG_TEST.m

Table III-E lists the Matlab scripts used to develop an automated process for synthesizing an LQG regulator for a 1-DOF setup using the loop transfer recovery (LTR) technique. The process is initiated by invoking the first Matlab script in the table, ONE_DOF_LTR.m. Note several scripts from the LQR process are reused.

TABLE III-E
MATLAB SCRIPTS USED FOR THE 1-DOF LTR DESIGN

Script	Description	Called By
ONE_DOF_LTR.m	Main program that coordinates the overall process.	-----
ONE_DOF_PLANT.m	Same as in Table I-E.	ONE_DOF_LTR.m
PROMPT.m	Same as in Table I-E.	All
KALMAN_SYNTH.m	Synthesizes a range of Kalman estimators based on the linear plant model. Allows user to select best candidate based on frequency responses of the estimators' open loop transfer functions.	ONE_DOF_LTR.m
LTR_PROCESS.m	Synthesizes one or more LQG regulators using the previously chosen Kalman estimator and user-specified range of LSVF matrices. Allows user selection of best design based on frequency response of system open loop transfer function.	ONE_DOF_LTR.m

TABLE III-E, CONT'D

Matlab Script	Description	Called By
FREQ_RESPONSE.m	Evaluates step response of closed-loop (CL) nonlinear system using selected LQG regulator and user-specified range of set points. Returns array of time domain responses for the CL system: %OS, 2% settling time, SS value and max. deviation of control input.	ONE_DOF_LTR.m
STIFFNESS_TEST_LQG.m	Similar to STIFFNESS_TEST.m except that stiffness ratio obtained for only one LQG regulator provided as input parameter.	ONE_DOF_LTR.m
B_PERTURB.m	Same as in Table II-E.	ONE_DOF_LTR.m

Controller Synthesis and Testing for 2-DOF System

Chapter VI of the thesis explained how the concepts of optimal control theory were applied to the 2-DOF case of magnetic levitation using MSBS components. Building on the Matlab programming from the previous chapter, more complex scripts were developed to implement controller synthesis.

In Chapter VI of the thesis the first technique employed to obtain a 2-DOF controller was the Loop Transfer Recovery (LTR) technique. The overall LTR process was similar to the one used for the 1-DOF setup. However, because of the multivariable structure of the controller, there were many more controller candidates to be evaluated. This prompted a decision to break the but the individual scripts had to be more intricate to handle the.

TABLE IV-E
MATLAB SCRIPTS USED FOR THE 2-DOF LTR DESIGN

Matlab Script	Description	Called By
TWO_DOF_LTR.m	Main program--coordinates the overall process. Saves results from each simulation in sequentially numbered .mat files	------
TWO_DOF_PLANT.m	Uses values for physical constants and creates linear state space model of 2-DOF plant. Also derives constants for use in Simulink models.	TWO_DOF_LTR.m

TABLE IV-E, CONT'D

Matlab Script	Description	Called By
KALMAN_SYNTH.m	Uses a range of tuning matrices to derive Kalman Estimators for the plant specified in state space form. The frequency responses of the open loop transfer function for each Kalman Estimator are derived and plotted for evaluation. The function returns the Kalman filter gain (Kf), the process noise gain mulitplier Gwgain and the Kalman Estimator (KF) associated with the particular derivation selected by the user via keyboard input.	TWO_DOF_LTR.m
PN_GAIN.m	Creates a process noise gain matrix for the Kalman Estimator.	KALMAN_SYNTH.m
EVAL_LOOP_RESP.m	Evaluates loop gain responses of a range of Kalman estimators based on a range of diagonal process noise gain matrices. These matrices are derived from a range of vectors provided as one of the inputs to the function. The function first requests evaluation criteria from the user, then forms the Kalman estimators and evaluates their Open Loop Gain responses against the user's criteria. The loop gain responses are derived using a bilnear transformation of the plant in order that the responses are properly represented in the w-plane. Function returns array of vectors representing the diagonals of the process noise gain matrices that produce acceptable Kalman estimators.	KALMAN_SYNTH.m
SET_CRITERIA.m	Uses default or takes user-defined or loop-shaping criteria to evaluate responses obtained from PN Gain matrices.	EVAL_LOOP_RESP.m
PLOT_RESPONSES.m	Plots Max/Min Singular Values of Loop Gain Responses and select candidate Kalman Estimators.	KALMAN_SYNTH.m
LTR_PROCESS.m	Uses a given plant and Kalman Estimator, along with a range of tuning matrices supplied by the user to aid in a Loop Transfer Recovery (LTR) process for a Linear Quadratic Gaussian (LQG) regulator. The function returns the LQR gain matrix, Krlqr, the input weighting matrix multiplier used to derive that matrix, Rgain, and the derived LQG regulator, LQGr. The function takes values for the multiplier, Rgain, from the user via keyboard input.	TWO_DOF_LTR.m
STEP_RESPONSE.m	Evaluates step response of closed loop nonlinear model using LQG regulator or LQG regulator with integral control depending on user input.	TWO_DOF_LTR.m

TABLE IV-E, CONT'D

Matlab Script	Description	Called By
STIFFNESS_TEST.m	Calculates stiffness ratio for given controller—LQQ or LQG with integral control.	TWO_DOF_LTR.m
B_PERTURB.m	Determines effects of B-field perturbations on step response for given controller—LQG or LQG with integral control	TWO_DOF_LTR.m

TABLE V-E
MATLAB SCRIPTS USED FOR THE ITERATIVE 2-DOF CONTROLLER DESIGN

Matlab Script	Description	Called By
TWO_DOF_TEST.m	Main program--coordinates the overall process. Saves results from each simulation in sequentially numbered .mat files	------
TWO_DOF_PLANT.m	Same as Table IV-E.	TWO_DOF_TEST.m
STEP_RESPONSE.m	Same as Table IV-E.	TWO_DOF_TEST.m
STIFFNESS_TEST.m	Same as in Table IV-E.	TWO_DOF_TEST.m
B_PERTURB.m	Same as in Table IV-E.	TWO_DOF_TEST.m
SORT_RESULTS.m	Separate program that loads .mat files saved by TWO_DOF_TEST.m and stores time response values from the step response for each controller tested. Evaluates all responses according to criteria for percent overshoot, settling time and the maximum deviation in control current. Creates table showing design parameters and responses meeting criteria. Only reports responses paired by like design parameters.	----
THRESHOLDS.m	From table of time response values for all controllers tested, selects values meeting hierarchy of criteria whose specific values are specified by the user: maximum deviation of control input, maximum percent overshoot, and maximum settling time.	SORT_RESULTS.m

APPENDIX F

SOFT-START CIRCUIT FOR COPLEY AMPLIFIER

Introduction

This appendix provides a description of the "soft-start" circuit produced to protect the Clinton DC power supply from current surges when the Magnetic Suspension and Balancing System (MSBS) is energized.

Problem Description

As discussed in the main body of the thesis, the MSBS configuration consists of six Copley 232P power amplifiers connected in parallel on the DC power bus. Each Copley amp contains a 26,000 μF filter capacitor wired across the amplifier's input. Thus a high capacitance load is continuously present on the Clinton power supply's output, regardless of the number of amplifiers in actual operation. Consequently, at the moment it's energized, there is a tremendous surge current in the Clinton power supply. This poses a great potential hazard to both equipment and people.

Soft-Start Circuit

A remedy to the surge problem was devised whereby the capacitors in the Copley amps are "pre-charged" using a separate, external power supply. When charged to a point where the surge current would be acceptable, the Clinton supply is energized. Referring to the circuit in Fig. 1-G, the following explains how the soft-start circuit operates.

The Clinton power supply is normally energized by pressing a momentary contact switch on a control panel in the Copley amplifier equipment rack. The power supply is designed to latch itself on once the contact is made. (The latch is broken and the power supply de-energized by pressing a different momentary contact switch on the same panel.) To implement the soft start, the original push-to-start switch was removed from the circuit and replaced with normally open contacts of a time-delay relay. The original switch was wired into the new soft-start circuit between an isolation transformer and bridge rectifier. The coil of a time delay relay is placed across the bridge rectifier outputs so that when the push button is depressed, the relay begins its time delay cycle. At the same time the rectifier output is applied through a high wattage resistor and diode to the DC power bus connecting all six Copley amplifiers. At the end of the time delay, currently about ten seconds, the amplifier capacitors are significantly charged, the relay contacts close and the Clinton power supply turns on as normal.

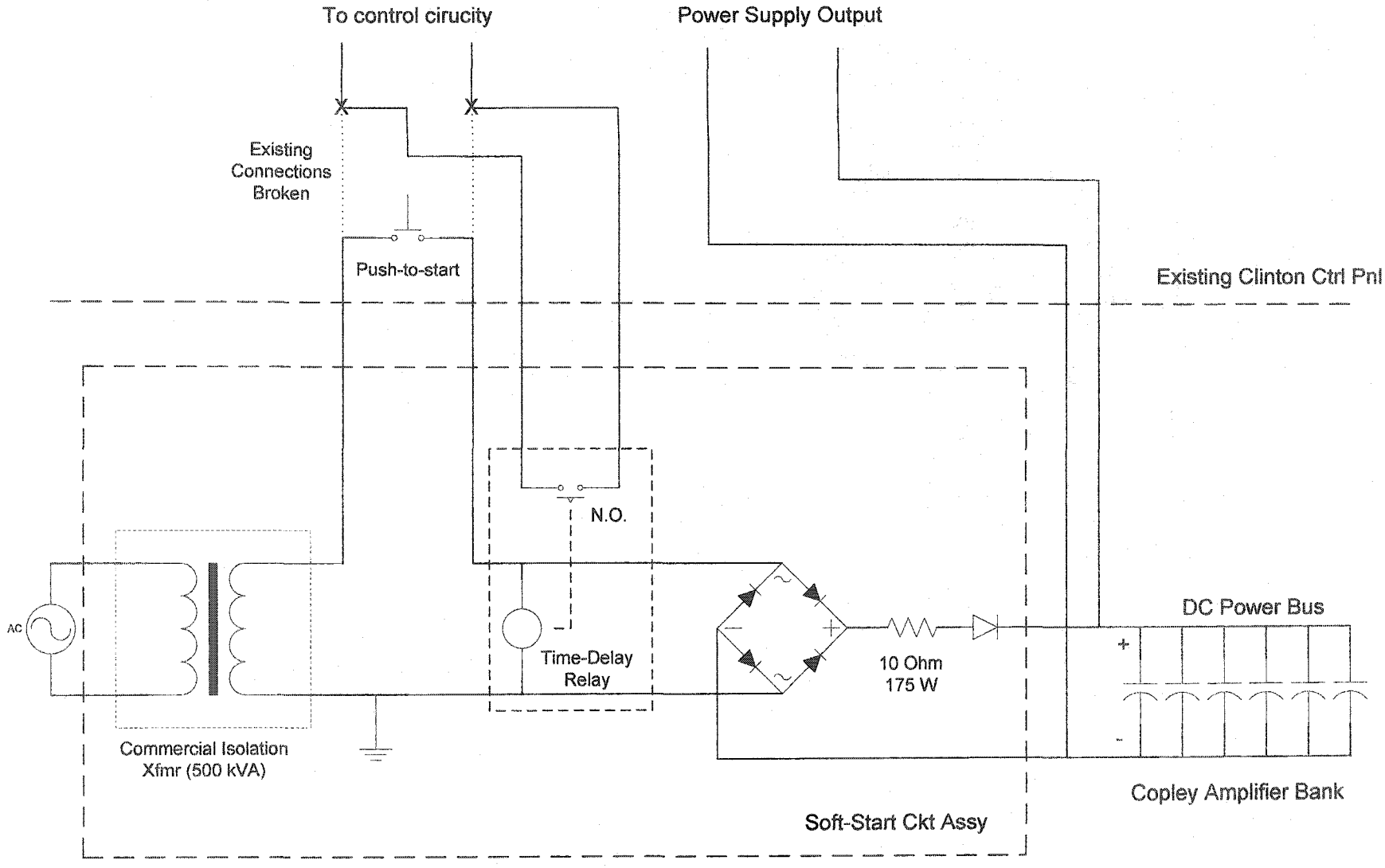


Fig. 1-G. Schematic diagram of MSBS "soft-start" circuit.

VITA

Mark W. Adams is currently a Lieutenant Commander on active duty in the United States Coast Guard. He attended Old Dominion University during the period May 2002 through May 2005 in pursuit of a Masters Degree in Electrical Engineering. He was awarded a Bachelor's degree in Electrical Engineering from Old Dominion University in December 1996.