CONTROL AND OPTIMIZATION OF A COMPACT 6-DEGREE-OF-FREEDOM PRECISION POSITIONER USING COMBINED DIGITAL FILTERING TECHNIQUES

A Thesis

by

JOSE CHRISTIAN SILVA RIVAS

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

December 2011

Major Subject: Mechanical Engineering

CONTROL AND OPTIMIZATION OF A COMPACT 6-DEGREE-OF-FREEDOM PRECISION POSITIONER USING COMBINED DIGITAL FILTERING TECHNIQUES

A Thesis

by

JOSE CHRISTIAN SILVA RIVAS

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Approved by:

Chair of Committee,	Won-jong Kim
Committee Members,	Alexander Parlos
	Hamid Toliyat
Head of Department,	Jerald Caton

December 2011

Major Subject: Mechanical Engineering

ABSTRACT

Control and Optimization of a Compact 6-Degree-of-Freedom Precision Positioner Using
Combined Digital Filtering Techniques.
(December 2011)

Jose Christian Silva Rivas, B.S., The University of Texas at Austin
Chair of Advisory Committee: Dr. Won-jong Kim

This thesis presents the multivariable controller design and implementation for a high-precision 6-degree-of-freedom (6-DOF) magnetically levitated (maglev) positioner. The positioner is a triangular single-moving part that carries three 3-phase permanent-magnet linear-levitation-motor armatures. The three planar levitation motors not only generate the vertical force to levitate the triangular platen but control the platen's position in the horizontal plane. All 6-DOF motions are controlled by magnetic forces only.

The positioner moves over a Halbach magnet matrix using three sets of two-axis Hall-effect sensors to measure the planar motion and three Nanogage laser distance sensors for the vertical motion. However, the Hall-effect sensors and the Nanogage laser distance sensors can only provide measurements of the displacement of all 6-axis. Since we do not have full-state feedback, I designed two Linear Quadratic Gaussian (LQG) multivariable controllers using a recursive discrete-time observer. A discrete hybrid $\mathcal{H}_2/\mathcal{H}_\infty$ filter is implemented to obtain optimal estimates of position and orientation, as well as additional estimates of velocity and angular velocity for all 6 axes. In addition, an analysis was done on the signals measured by the Hall-effect sensors, and from there several digital filters were tested to optimize the readings of the sensors and obtain the best estimates possible. One of the multivariable controllers was designed to close the control loop for the three-planar-DOF motion, and the other to close the loop for the vertical motion, all at a sampling frequency of 800 Hz. Experimental results show a position resolution of 1.5 μm with position noise of 0.545 μm rms in the *x*-and *y*-directions and a resolution of less than 110 nm with position noise of 49.3 nm rms in *z*.

ACKNOWLEDGMENTS

First of all, I thank my thesis advisor, Professor Won-jong Kim. Dr. Kim's patience and encouragement bolstered my confidence and fueled my excitement in my work. I truly admire his knowledge and experience in the field of electromechanical systems, control systems, and precision positioning devices. His vast knowledge in the theoretical and practical applications of these fields encouraged me to learn all I could to have a more complete understanding and systems approach to problem solving.

I would like to thank Professor Alexander Parlos for serving on my thesis committee. I was privileged to take his Multivariable Controls class, which was very enjoyable, enlightening and helped me a lot for my research. I also served as a teaching assistant for Dr. Parlos for the undergraduate controls class, which helped me communicate engineering and controls principles effectively to an audience. I would also like to thank Professor Hamid Toliyat for serving on my thesis committee. His suggestions and comments on my research and goals were very helpful and insightful.

I am very grateful to the students in Dr. Kim's research labs. I am very thankful for the unique friendship I have with each one, and for the technical assistance provided as well. I am especially grateful to Dr. Ho Yu, a previous doctoral student for Dr. Kim. He was the one in charge of building the current magnetically levitated positioner for my research. I would also like to express my gratitude to Vu Nguyen for his technical insight into my problems, for his suggestions in my research and for helping me get started on my research.

To my parents, Jose and Mayo, I give the utmost in appreciation. It seems that since my entrance to this world, they have continuously nurtured my mind, body, and spirit. I was always encouraged in many academic and sports-related pursuits, and owe my work ethic to the model they set. I am thankful to my father for the example he set as a man and my mother for her unconditional love.

TABLE OF CONTENTS

CHAPTER Page

LIST OF TABLES

Page

LIST OF FIGURES

Page

Page

Page

CHAPTER I

INTRODUCTION

1.1 High Precision Motion Control

In modern nanoscale or microscale engineering, wafer steppers, surface profilometers, and scanned probe microscopes require high-precision motion control. Especially the wafer stepper stage in semiconductor manufacturing is the main application of the work presented in this thesis. The wafer stepper stage is very important equipment for photolithography such as generating step-and-repeat motions. An optical source sheds a deep-ultraviolet (DUV) beam through the mask onto each die site on the wafer. The wafer stage is required to move the wafer in all 6 directions with minimum errors, it should have high resolution and accuracy. In this thesis, multivariable LQG controllers are presented for a compact 6-DOF maglev positioner with high precision.

1.2 Prior Art of the Single-platen Maglev Multi-axis Positioners

1.2.1 The First 6-DOF Planar Levitation Stage

The world's first one-moving-part 6-DOF precise planar magnetic-levitation stage was developed by Kim [1]. Figure 1-1 below describes the structure of the positioning system. In this design, the winding is stationary and the permanent magnet arrays are attached to the moving platen. There are totally 4 motors; each has 3 phases with 11 windings per phase. Each motor is to generate two force components, one in the *z*-direction and the other one in the *x*- or the *y*-direction.

This thesis follows the style of Journal of Dynamic Systems, Measurement and Control.

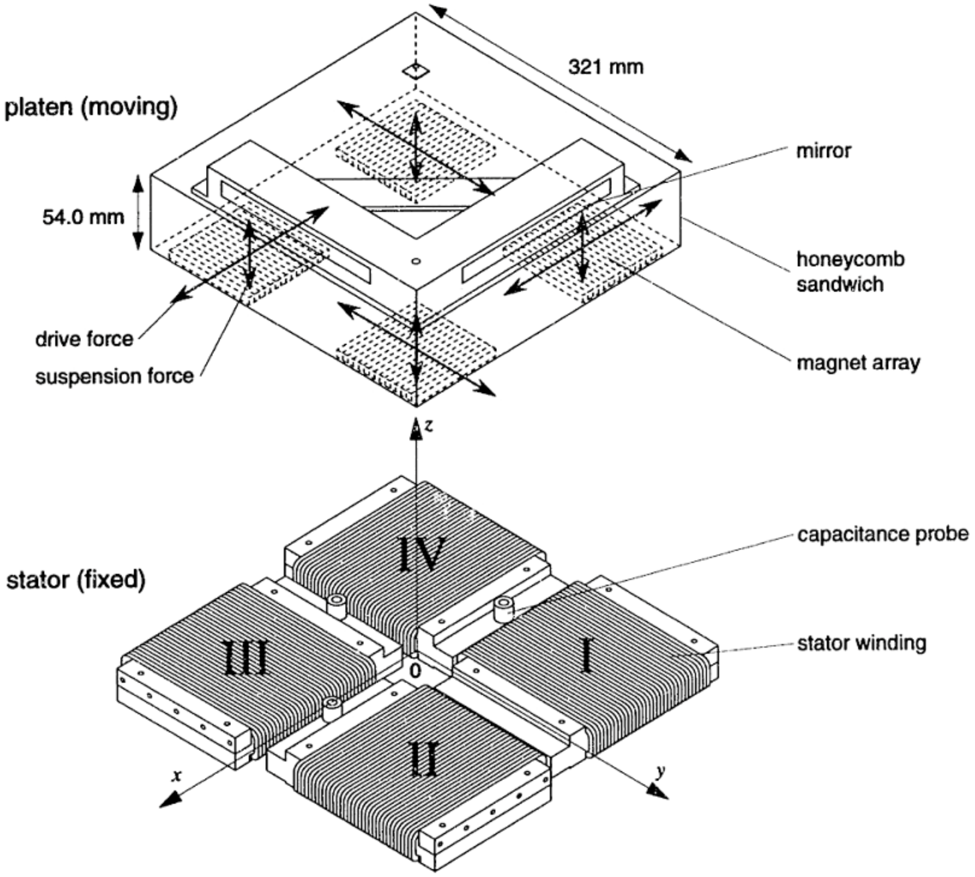


Figure 1-1: Structure of the design with four motors [1]

DQ decomposition was performed to obtain the electric current-force relation and to eliminate the nonlinearity due to trigonometric functions. Two orthogonal force components are decoupled to control the two degrees of freedom independently. As in Figure 1-2, the *D*-axis is defined as the *z′*-axis in the platen frame; the *Q*-axis leads the *D*-axis by $\frac{l}{4}$ [1]. Here, *l* is a spatial pitch of the Halbach magnet array.

This system has many advantages, such as, the linearity in force-current relation, no mechanical contact during moving, being symmetric in the *x*- and *y*-directions. There is no wire or cable connected with the moving platen except a thin ground wire for the capacitance gauges. Finally there is no heat stored and dissipated through the moving platen. Therefore, the thermal expansion error is minimized [1].

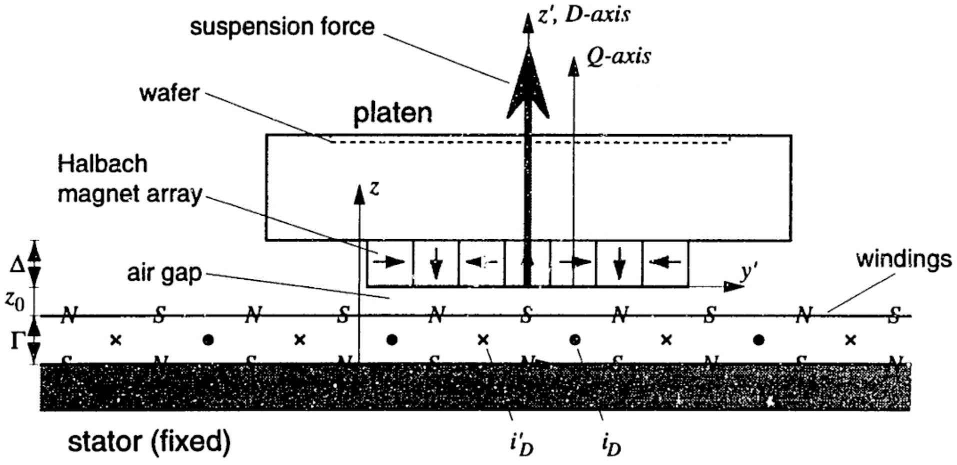


Figure 1-2: Suspension of the platen in dynamic equilibrium [1]

The maglev positioning system has a sampling rate of 5 kHz, a control bandwidth of 100 Hz and a maximum acceleration of 2 *g*. The overall mass of the moving platen is 5.6 kg, and its travel ranges are 50 mm in *x*, 50 mm in *y* and 400 μm in *z* [1].

1.2.2 Advanced Technology Program (ATP) Stage

The ATP stage was developed by Tiejun Hu, a former Ph.D. student of Dr. Won-jong Kim [2]. The moving platen works over a superimposed Halbach magnet matrix, which is the superimposition of two orthogonal single-axis Halbach magnet arrays. Figure 1-3 shows the moving platen with its components. The design uses the same linear force-current relation as in [1] by *DQ* decomposition. There are 3 motors with 3 phases per motor and 4 windings per phase. Totally 6 force components distributed as in Figure 1-4 are sufficient for the moving platen to move in 6 DOFs.

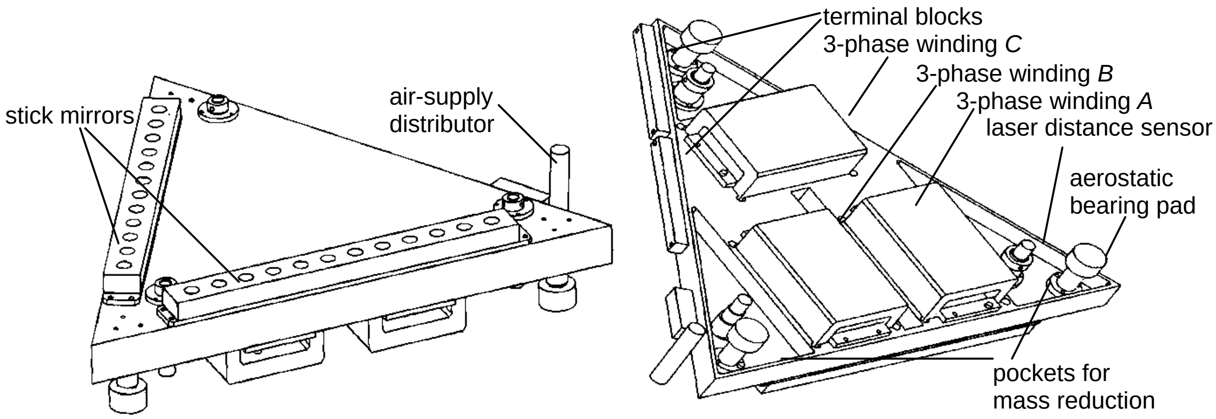


Figure 1-3: Top and bottom views of the moving platen [2]

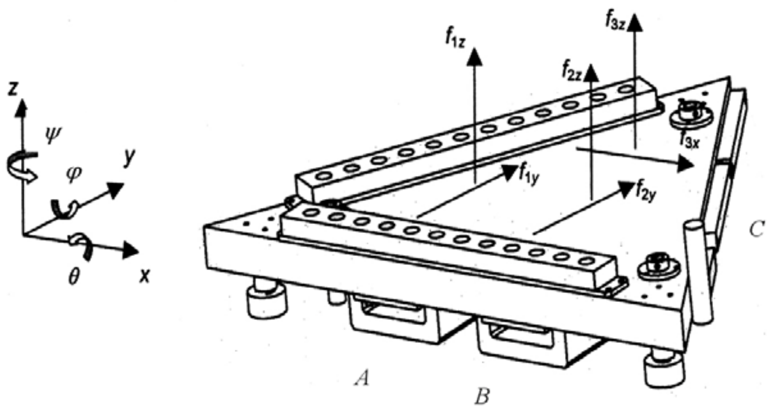


Figure 1-4: Forces generated by three sets of coils of the moving platen [2]

Three Agilent laser interferometers are used to sense the exact position of the moving platen in planar motions. There are two stick mirrors reflecting the laser beams. Three Nanogage 100 laser distance sensors are to measure the distances from the laser heads to the flat aluminum surface on top of the magnet matrix. With these measurements, the roll, pitch, and yaw angles of the moving platen can be determined.

The moving platen, which weighs 5.91 kg, can travel a range of 160 mm in *x* and *y*. The positioner demonstrated a position resolution of 20 nm and position noise of 10 nm rms in *x* and *y* and 15 nm rms in *z*. Maximum velocity achieved is 0.5 m/s at a 5 m/s^2 acceleration [2].

1.2.3 Long-stroke Magnetically Levitated Linear Actuator

A moving-coil electrodynamic planar motor was developed by Compter [3]. The stationary part is a two-dimensional Halbach magnet array. The moving part has four forcers with three phases per forcer. Figure 1-5 shows the structure of the planar motor with the stationary magnet plate and four forcers fixed to the moving part.

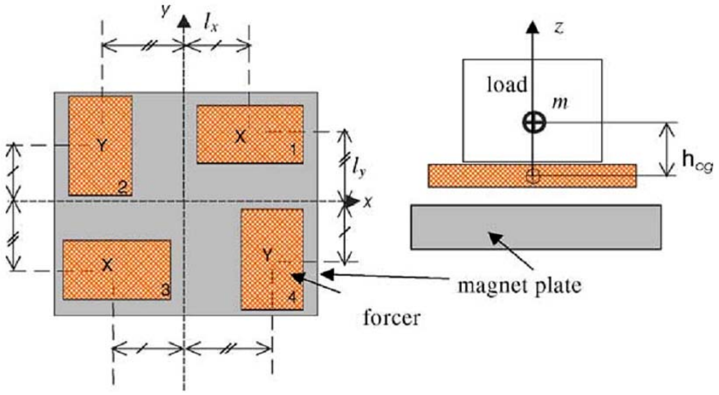


Figure 1-5: The electrodynamic planar motor with four forcers in the moving part [3]

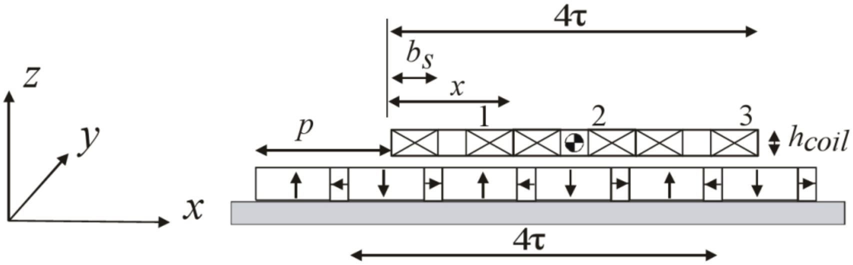


Figure 1-6: Cross-section of a forcer with the Halbach magnet array [3]

Figure 1-6 shows a cross-sectional view of a forcer with the Halbach magnet array. The horizontal distance between two adjacent coils is $4\tau/3$ where τ is one spatial pitch of the magnet array. Each forcer is a symmetrical three-phase motor. In Figure 1-7, the magnet orientation is illustrated. The angle between the longer sides of the coils and the edges of the magnet blocks is 45°.

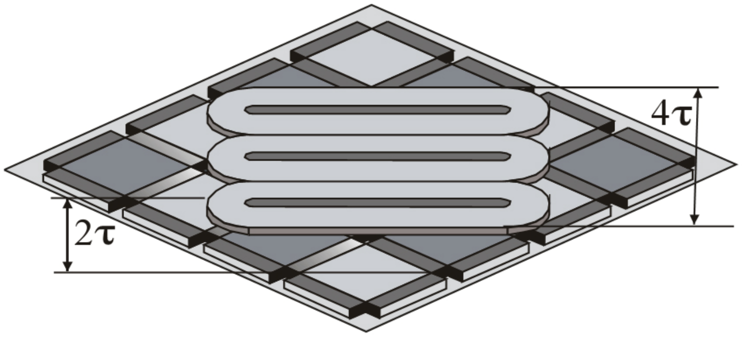


Figure 1-7: Magnet orientation [3]

Each forcer can generate two force components in the vertical direction and in a horizontal direction. The force-current relation for a forcer was derived with the introduction of the current amplitude and the current phase. In which, the current amplitude determines the force amplitude and the force angle is controlled by the current phase angle [3]. The positioning resolution, the maximum velocity, acceleration and travel ranges of this electrodynamic planar motor are not shown in [3]. In this system, the sensing accuracy of 1% of a spatial pitch of the magnet array is low if the spatial pitch is on the order of 10 mm.

1.2.4 Compact Positioner Moving over a Superimposed Halbach Magnet Matrix

The multi-axis compact positioner shown in Figure 1-8 was designed and implemented by Nguyen [4]. The single-moving-part positioner is designed to move in the magnetic field generated by a superimposed concentrated-field permanent magnet matrix. The travel ranges in two orthogonal directions are on the order of 100 mm. The moving platen, which has the size of 185.4 mm × 157.9 mm and weighs 0.64 kg, mainly consists of a plastic frame and six copper coils. It is actuated in the horizontal plane by flowing six independent electric currents into the coils. The platen is supported against gravity by three air bearings.

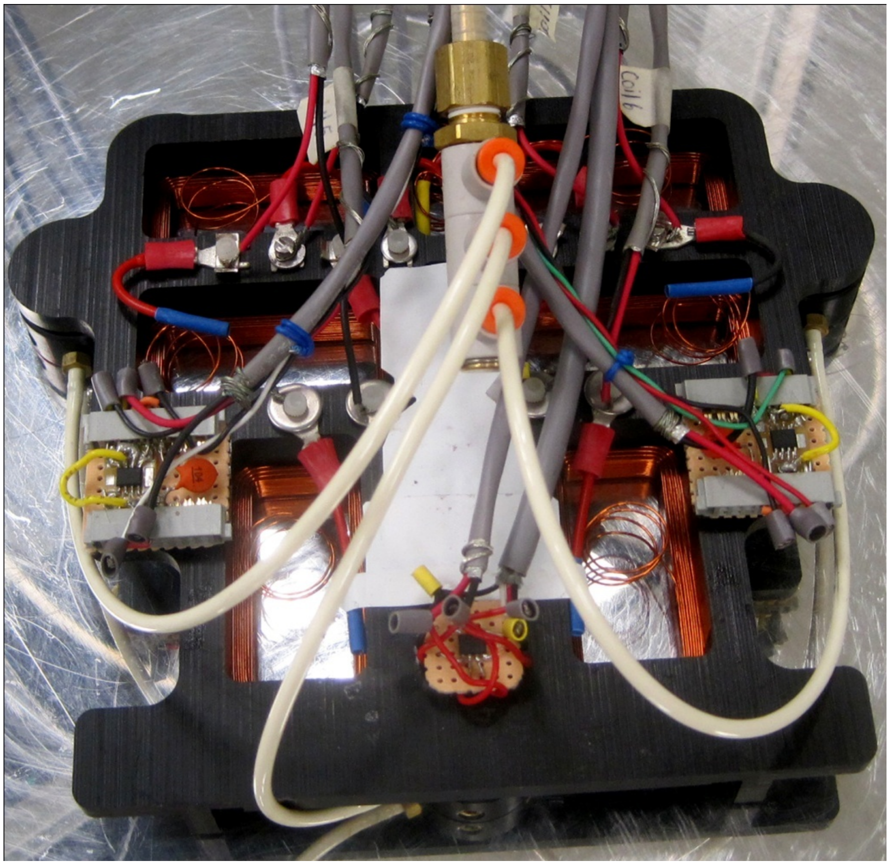


Figure 1-8: Top view of the platen with all components assembled [4]

The force calculation is based on the Lorentz force law, with a current-carrying rectangular coil placed in the magnetic field of the superimposed Halbach magnet matrix, the force acting on the coil is calculated by volume integration. The distances between the longer sides and between the shorter sides of the rectangular coil are designed to fit a half pitch and one pitch of the Halbach magnet array and the force-current relation for the entire platen with six coils is derived.

Three Hall-effect sensors are attached to the moving platen to measure the magnetic flux densities. The position of the moving platen is determined by the field solution of the magnet matrix and the magnetic flux densities sensed by the Hall-effect sensors. For the step responses with step sizes within 1-mm, the overshoots and the steady state errors are negligible. The achieved travel ranges are 15.24 cm in *x*, 20.32 cm in *y*, and 0.21 rad in the rotational motions about the vertical axis. The positioning resolution in *x* and *y* is 8 μm with the rms positioning error of 6 μm. The positioning resolution in rotation about the vertical axis is 130 μrad [4].

1.3 Overview of Compact 6-DOF Precision Positioner

The compact 6-DOF precision positioner was developed by Ho Yu, a former Ph.D. student of Dr. Won-jong Kim [5]. There are 3 motors as in [2] and the force allocations of the two designs are the same. In Figure 1-9, the motors are noted as 3-phase windings A, B, and C. In this design, the number of coils per motor is only 3, compared to 12 in [2]. Therefore, the total number of coils, the mass and volume of the moving platen, and the energy consumed are reduced considerably.

The most significant feature is that the only one levitated moving part, namely the platen, can generate all 6-DOF fine and coarse motions. Any 6-DOF motions can be generated by a combination of the 6 force components of the platen. In addition, there are several additional advantages such as:

1. A mechanically non-contact machine structure does not need lubricants, nor produce wear particles. Therefore, it is suited for clean-room environment.
2. Superimposing multiple linear motors as one actuator reduces the footprint.
3. Compared to traditional positioners, the single moving frame can have high natural frequencies.
4. The simple design eliminates complicated components and reduces manufacturing cost with high reliability [1–2].

The frame of the platen was made of Delrin with a mass density of 1.54 g/cm^2 in order to reduce its total mass. The triangular design was chosen for the design simplicity. The magnet matrix, a superimposed concentrated-field double-axis magnet matrix serves as a stator. The dimension of the magnet matrix is 304.8 mm × 304.8 mm × 12.7 mm.

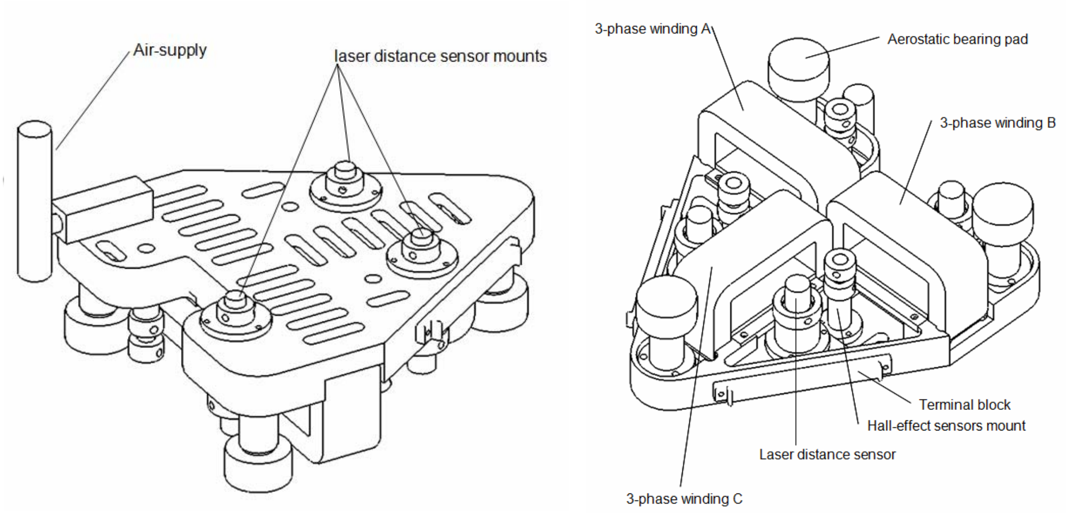


Figure 1-9: Structure and components of the moving platen [5]

There are 3 Hall-effect sensors attached to the moving platen to measure the magnetic flux densities at the center points of the sensors. The position of the moving platen is determined by the field solution of the magnet matrix and the magnetic flux densities sensed by the Hall-effect sensors, the collaboration of two Hall-effect sensors is needed to locate the position of the moving platen in a horizontal direction, and another Hall-effect sensor is needed to locate the position in the vertical direction. A Hall-effect sensor is only used for locating the position of the moving platen and checking if it is in a sensitive interval.

The total mass of the moving platen is 1.52 kg. The travel ranges are 220 mm in the *x*-direction and 200 mm in the *y*-direction. The rotation angle of 12° was achieved in rotation around the *z*-axis [5], which is perpendicular to the top flat surface of the magnet matrix. The positioning resolution in *x* and *y* is 15 μm with the rms positioning noise of 10 μm.

1.4 Thesis Overview

This thesis consists of seven chapters: introduction, electromagnetic structure, sensors, multivariable controller design, optimization of the Hall-effect sensor signals, 6-DOF closed-loop experimental results, and conclusions and future work.

Chapter I introduces the reviews of precision engineering and its applications. The introduction of levitation theory is given, and the overview of the proposed 6-DOF multivariable controller with high precision is provided.

Chapter II presents the electromagnetic structure and concepts for the 6-DOF precision positioner. It includes the magnet matrix theory, overview of the planar motors, the sensing system, and an overview of the instrumentation to create the interface between the control PC and the positioner.

Chapter III describes several parameters and specifications required to control the precision positioner. Dynamic models of the levitator based on the Newton's second law are derived. In addition, *DQ* decomposition to remove nonlinearity in dynamics is applied, and linearized state-space models in vertical and lateral modes are developed.

Chapter IV presents the design methodology, procedures and implementation of the multivariable LQG controllers for the maglev system, which includes the LQR control laws and the discrete hybrid filter acting as the state estimator.

Chapter V presents the analysis of the Hall-effect sensor signals, from which several digital filters are designed and tested in order to optimize the response of the signal for the planar motion.

Chapter VI provides the 6-DOF closed-loop experimental results including micro- and nano-scale stepping motions, long-range travel motions in precision motion control such as semiconductor manufacturing, as well as comparisons between my results and the results of the designer of the positioner.

Chapter VII gives the conclusions of this thesis and suggestions for future work. The reference section is followed and the Matlab and C-program codes are included in the Appendices.

1.5 Thesis Contributions

This thesis presents a multivariable controller design, implementation, and experimental results, as well as the combination of digital filtering techniques for the planar motion. Two LQG-controllers were designed and implemented to achieve the 6-DOF magnetic levitation, one controller for planar motion and another one for the levitation motion, using a discrete hybrid filter as the observer. The discrete hybrid filter is necessary as the sensors only provide displacement data and full state feedback is needed for the LQG controller to work. The hybrid filter provides optimal estimates of the displacement data as well as additional estimates of the velocities for all 6 axes. Also an analysis was conducted for the Hall-effect sensors. From this analysis several digital filters were designed and tested for the planar motion in order to filter out the static noise and obtain a better resolution. Several experimental results of nanometer- and sub-micrometer-level positioning in all 6 DOFs are presented in this thesis, as well as long range motions in the *x*-*y* directions.

CHAPTER II

ELECTROMECHANICAL DESIGN

This chapter introduces the electromagnetic structures and the electromechanical design for the compact positioner. The positioner is operating above the concentrated-field magnet matrix that is the superimposition of two Halbach magnet arrays. The magnetic field generated from a Halbach array is analyzed by Fourier series. Since the magnetic field is periodic, the positioner can generate driving and levitation forces.

2.1 Halbach Magnet Matrix

2.1.1 Halbach Magnet Array

The magnet array in the base plate is employed as a stator. For the planar motor, the magnet array has a form of a plane. Magnetic arrays for the conventional permanent-magnet linear motors have both sides of the magnetic fields if there is no iron backing. Since the planar motor works only one side of the magnet array and the other side is wasted, an anti-symmetric magnetic field pattern was desired. The Halbach magnet array is the representative anti-symmetric single-side field magnetic array [6]. Figure 2-1 shows the single axis Halbach array. Four blocks of magnets with the magnetization rotated 90° in each other construct a single spatial pitch. Each adjacent magnet segment is rotated around an axis perpendicular to the direction, in which the array extends by a predetermined angle of 90° or 45°. The Halbach magnet array produces a $\sqrt{2}$ times stronger magnetic field than that of a conventional magnet array with the same volume.

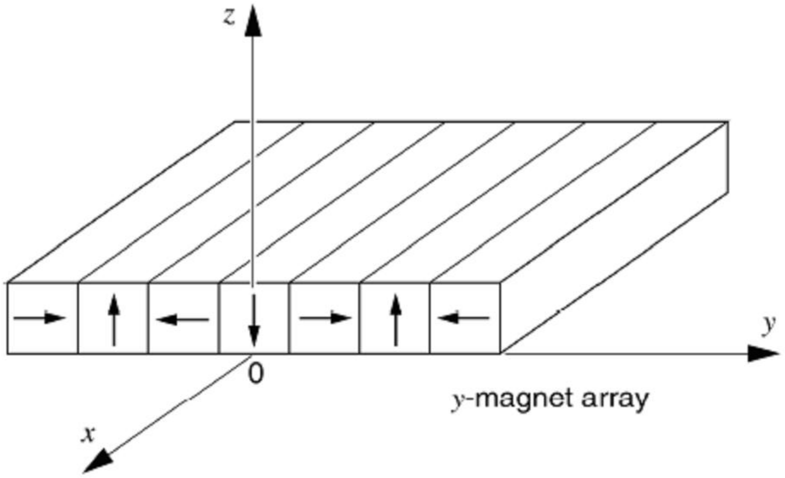


Figure 2-1: Halbach magnet array in *y* [1]

Halbach magnet array has a periodic geometry and the magnetic field so that Fourier series representation is suitable for analyzing the magnetic field. In our positioning stage, the strong fundamental field is generated on the side faced with windings on platen. The Halbach magnet array magnetization using Fourier series is first performed to model the magnetic flux on the magnet matrix.

The vertical and the lateral magnetization components of the Halbach magnet array are represented by the complex Fourier coefficients [1,6].

$$\tilde{M}_{xn} = \frac{1}{l}\int_0^l M_x e^{jk_n x'} dx', \tag{2.1}$$

$$\tilde{M}_{zn} = \frac{1}{l}\int_0^l M_z e^{jk_n x'} dx', \tag{2.2}$$

where k_n is the spatial wave number as follows,

$$k_n = \frac{2\pi n}{l}. \tag{2.3}$$

The spatial wavelength of the magnet array is represented by l which is identified with the pitch of 50.97 mm. The Halbach magnet array whose magnet has a square cross-section is presented as $\Delta = l/4$.

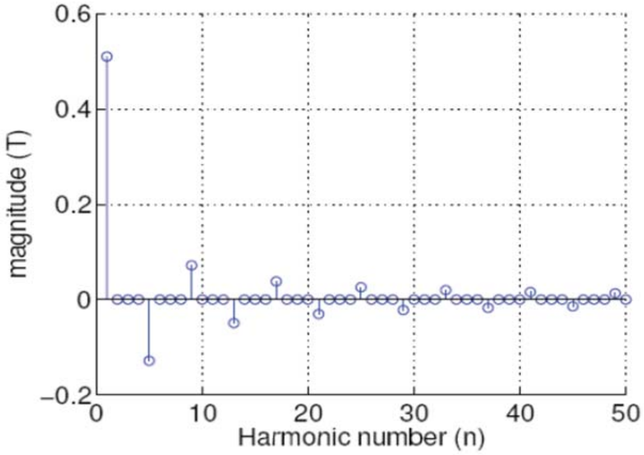


Figure 2-2: Fourier coefficients of the Halbach magnet array flux density [1]

The Fourier coefficients of the Halbach magnet array flux density on the strong side are shown in Figure 2-2. The total magnetization is represented in (2.4-2.5),

$$M_z = \sum_{n=-\infty}^{\infty} (\tilde{M}_{zn} e^{-jk_n y} + \tilde{M}_{zn} e^{-jk_n x}), \tag{2.4}$$

$$M_x = 2\sum_{n=0}^{\infty} (\tilde{M}_{xn} d^{-jk_n x}) = 2\sum_{n=0}^{\infty} \tilde{M}_{xn} (\cos k_n x - j \sin k_n x), \tag{2.5}$$

where M_{xn} , M_{yn} and M_{zn} are the Fourier coefficient with peak magnetization of M_0. After we define the magnetization components, the magnetic flux density with air gap Z_0 can be obtained. The following equations present the magnetic flux density using magnetoquasistatic (MQS) approximation,

$$\tilde{B}_{zn} = \frac{\mu_0}{2} (j\tilde{M}_{xn} + \tilde{M}_{zn}) e^{-\gamma_n Z_0} (1 - e^{-\gamma_n \Delta}) \tag{2.6}$$

$$\tilde{B}_{xn} = \frac{\mu_0}{2}(\tilde{M}_{xn} - j\tilde{M}_{zn})e^{-\gamma_n Z_0}(1 - e^{-\gamma_n \Delta}) \qquad (2.7)$$

where γ_n is the absolute value of the spatial wave number.

The field solutions for the magnet array at a plane Z_0 are shown in equations (2.6-2.7). From this one can derive the Fourier coefficients of the magnetic flux density from the magnet array illustrated in Figure 2-1. In the ideal case Halbach magnet array, the strong side of the magnet array generates the magnetic field of the fundamental, 5th, 9th ,…, orders. The magnetic field is a superimposition of those harmonics. The 3rd, 7th, 11th,…, order fields cancel out. A more purely sinusoidal field on the strong side of the Halbach magnet array can be generated from the explanations above. That means the planar motor may have little effect from the force ripple and obtain better power efficiency.

2.1.2 Concentrated-field Magnet Matrix

However to generate both planar and vertical motions, another Halbach magnet array is required. The superimposition of two orthogonal Halbach magnet array produces a concentrated-field magnet matrix [1]. The constant magnetic field is measured by Hall-effect sensors [6]. The result of the superimposition is shown in Figure 2-3 as a plane view, in addition, the magnet blocks are represented with an arrow, they have $1/\sqrt{2}$ remanence of the magnets noted with North (N) and South (S) poles, which are strong magnets. Shaded magnet blocks surrounding the North pole tip up 45° as shown in Figure 2-4(a) and shaded blocks surrounding the South pole tip down 45° as shown in Figure 2-4(b), blank spaces mean that the magnet field is canceled out. As mentioned in the previous section, the Halbach magnet array has a stronger fundamental field, by a factor of $\sqrt{2}$, than the other side. Since the concentrated-field magnet matrix follows the linear superposition between two orthogonal magnet arrays, the basic principle of the Halbach magnet array is applicable to the two-dimensional magnet matrices.

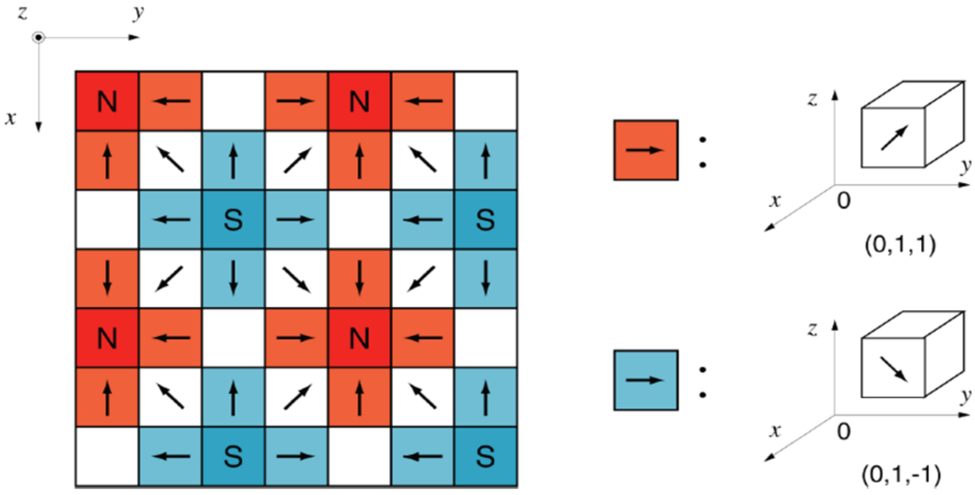


Figure 2-3: Concentrated-field magnet matrix [1]

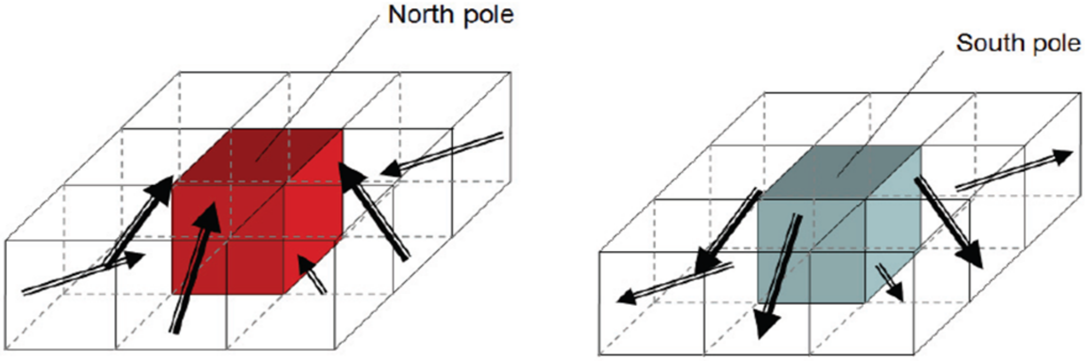


Figure 2-4: 3D views of the concentrated-field magnet matrix [2]

The magnet matrix designed and manufactured by Dr. Won-jong Kim's former students Nikhil Bhat and Dr. Tiejun Hu consists of a total of 6 pitches in the *x*- and *y*-directions, by adding the number of pitches in each direction, the travel range of the positioner can be increased. The magnet matrix is composed of neodymium-iron-boron (NdFeB) magnetic blocks, this rare-earth magnetic material has a high remanence, coercive force and maximum energy. The total magnet array includes 432 weak magnet block, 72 strong magnet blocks, and 72 aluminum spacers. Figure 2-5 presents the picture of the finished magnet array.

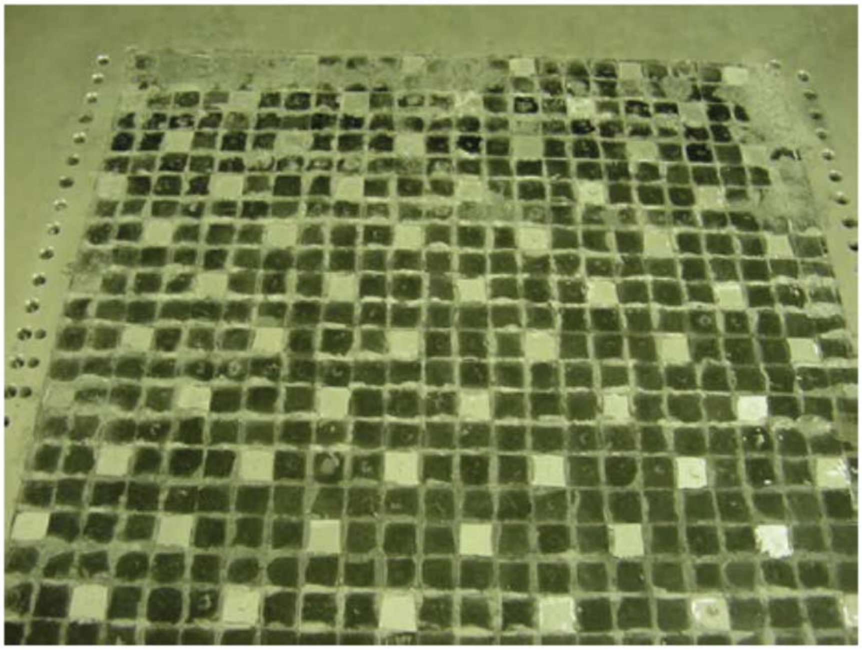


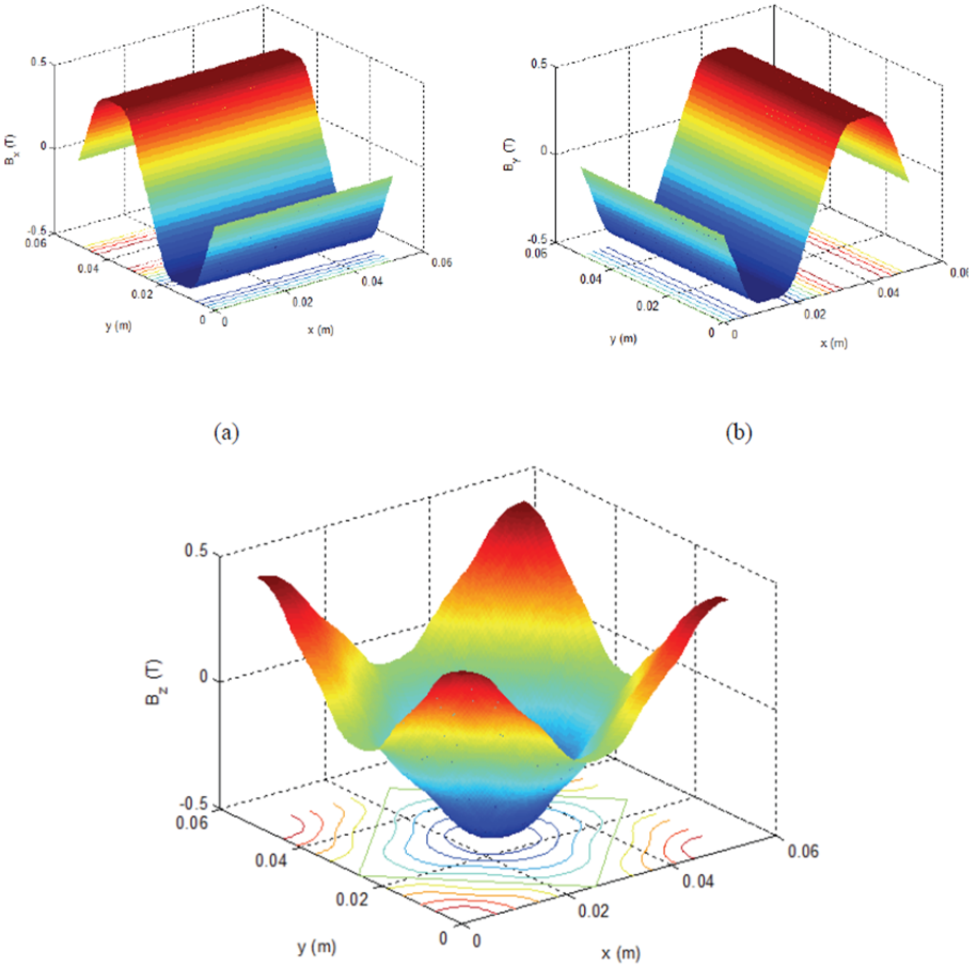
Figure 2-5: Assembled magnetic matrix with aluminum spacers [2]

From the definition of the Fourier series in the previous section, the total magnetic flux densities of the Halbach magnet array on the fundamental strong side are presented as follows,

$$B_z = \sum_{n=0}^{\infty} \tilde{B}_{zn}(e^{-jk_n x} + e^{-jk_n y}), \tag{2.8}$$

$$B_x = 2\sum_{n=0}^{\infty} \tilde{B}_{xn} e^{-jk_n x}, \tag{2.9}$$

Figure 2-6 (a), (b) and (c) present the total magnetic flux density in *x*, *y* and *z* respectively. According to the previous equations, the magnetic flux density decays exponentially with respect to the air gap.



(a)

(b)

Figure 2-6: Flux density representation (a) in *x*, (b) in *y*, and (c) in *z* [7]

2.2 Motor Windings

The positioner carries three planar motors on the bottom face of the body. Each planar motor consists of three phases, with which the positioner totally include nine winding sets. A single winding has three hundred and five turns with AWG#24 heat bondable wire (diameter of 0.0213"). To achieve optimal motion performances with a planar motor, the flatness process of the bottom surface is crucial. Design work of the winding set performed by Hu took care of several parameters such as, the thickness of wire, the number of layers, the number of turns, the peak phase current, and the terminal voltage [2].

2.3 Power Amplifiers

As mentioned in the previous section, the positioner includes total 9 three-phase winding sets. Nine independent channels control the winding actuator for the positioner's motion generation. The source components that control winding set are the currents from the amplifiers. In this system, total nine-channel power amplifiers are required to supply individual phase currents. The programming code and RTAI and Comedi deal with the voltage signals from the analog-to-digital converters. It also generates the voltage commands to the digital-to-analog converters (DACs). In the middle of the winding set and the converters, the power amplifiers supply the exact amount of current flowing to the winding set from the voltage sources. The power amplifier's dynamics is a critical issue for the current signal control.

2.4 Power Supplies

Two linear power supplies of LZS-250-3 manufactured by LAMBDA Electronics[1] are used. It is manufactured as rugged mechanical and superior thermal design. Besides, wide range adjustment of output and protecting transient input voltage are the feature of this power supply. The linear power supplies have 5 mV ripple errors. On the other hand, the switching-type power supplies have the ripple errors of 150 mV. Figure 2-7 shows the picture of the linear power supply in the front, and the back, respectively. The specifications of the LZS-250-3 are presented below:

• Output voltage range:	18.0~29.4 V.
• Nominal Voltage:	24 V.
• Maximum output current at 40°C:	12.5 A
• Maximum output current at 60°C:	10 A
• Ripple and noise:	100 mV (peak to peak)/10 mV (RMS).

[1] LAMBDA Americas, 3055 Del Sol Blvd, San Diego, CA 92154

The power OP Amp ratings and the maximum output current to the planar motors determine the current and voltage ratings of the power supply. The maximum current in winding that we measured was 1.29 A. Since the positioner carries total nine phases in planar motors, the maximum input current should be more than nine times of the maximum current of a single phase winding that is 11.61 A.

Figure 2-7: LZS-250-3 power supply

Nine linear power OP Amp PA12A manufactured by Apex[2] are used. The specifications of the power OP Amp satisfy the current and the voltage swing for the power amplifier circuits. Some of PA12A's specifications [3] are listed below:

• Supply range:	±10 V to ±50 V.
• Output current:	Up to ±15 A peak.
• Settling time to 0.1% in a 2V step at 25°C:	2 μs.
• Slew rate at 25°C:	4 V / μs.
• Maximum power dissipation:	125 W.

This power OP Amp is commonly used in motor, power transducers, temperature controls, programmable power supplies, and audio amplifiers [8].

[2] Apex Microtechnology Corp., 5980 N. Shannon Road, Tucson, Arizona 85741

2.4.1 Power Amplifier Circuit

The power amplifier circuit illustrated in Figure 2-8 was designed by Hu [2]. The differential amplifier, feedback amplifier, and power booster form a circuit. Resistors R_1, R_2, R_3, and R_4 serve on the differential amplifier which blocks common-mode input signals and works as a low pass. A feedback amplifier serves on to stabilize the current control loop. The current-sensing resistor R_{10} measures the current flowing through the winding in real time. The amplifier closed-loop bandwidth of 1.3 kHz was obtained. Parameters of the resistance are presented below.

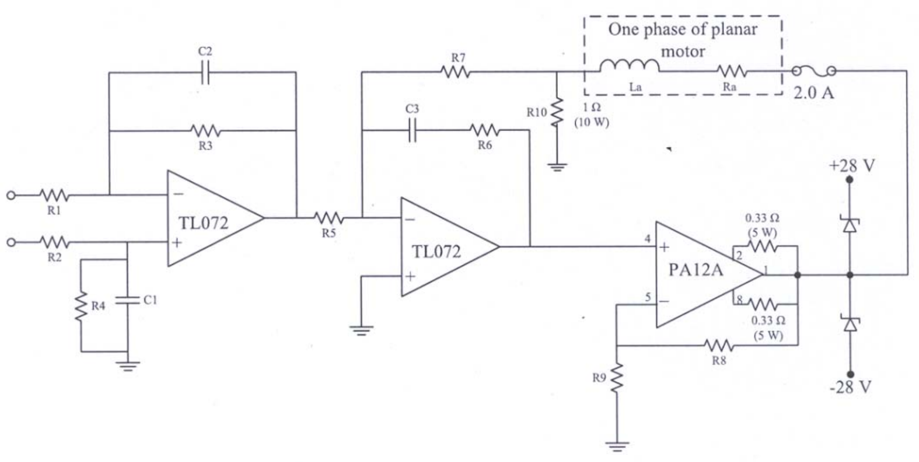


Figure 2-8: Power amplifier circuit [2]

- $R_1 = R_2 = R_3 = R_4 = R_5 = R_8 = R_9$: 10±0.001 kΩ
- R_6: 82 kΩ
- R_7: 1.3 kΩ
- R_{10}: 1 Ω
- $C_1 = C_2 = C_3$: 0.01μF
- R_a: 19.44 Ω
- L_a: 15.26 mH

After the power amplifiers were connected to the planar motors, voltage and current tests were performed. Each single winding was connected to the power amplifiers individually. Then the capabilities of supplying appropriate amount of current were checked by digital multimeter. A single winding has a resistance of 4.98 Ω. The power amplifier receives the voltage signal sources from the DAC converter. Feedback control is performed as voltage signals on RTAI and Comedi. The maximum voltage sensing of the DACs is ±10 V. The maximum output current from the power amplifier is 1.29 A when the DAC output voltage reaches the maximum value of 10 V.

2.5 Data Acquisition

As mentioned in the previous section, feedback signals from sensors are controlled as voltage signals on programs. Digitized signals with sampling frequency of 800 Hz circulate the feedback control loops through the power amplifiers, planar motors, Hall-effect sensors, and laser distance sensors. As an analog-to-digital converter (ADC), NI PCI-6221 board manufactured by National Instrument (NI) is used. The NI PCI-6221 ADC supports 16 16-bit, 250-kS/s analog output channels, 24 digital I/Os, two 16-bit analog outputs and 32-bit counters. All analog voltage signals from three 2-axis Hall-effect sensors and three laser distance sensors come through this board and are converted to digital signals. As a DAC, NI-6703 board manufactured by NI is used. This DAC board supports sixteen 16-bit analog output voltage channels with ±10V and 8 digital I/O lines. These two data acquisition boards are supported by Comedi drivers of “ni_pci.mio” and “ni_670x” for the real-time control. Although commonly used data acquisition board such as NI-6221 has both of analog input and output channels, the dual data acquisition boards were adopted.

For the convenience of signal cable connection, a CB-68LP connector manufactured by NI is used. Each data acquisition board has their own connectors. The numbers represented on the board correspond with data channels on data acquisition board to distinguish channels. In addition, screws placed on the pins give convenience to connect and disconnect data signal cables just using a screw driver. Figure 2-9 presents two connectors for two data acquisition boards. The left side is the DAC connection of NI-6703 and output data signal cables connected to the power amplifiers. The right side is the ADC connection of NI-6221 and input data signal cables from laser distance sensors and 2-axis Hall-effect sensors. Shielded signal cables that have 68 pins are used between the data acquisition boards and the connector blocks. It consists of individually shielded twisted pairs for the analog inputs and outputs as well as the twisted pairs for critical digital I/O.

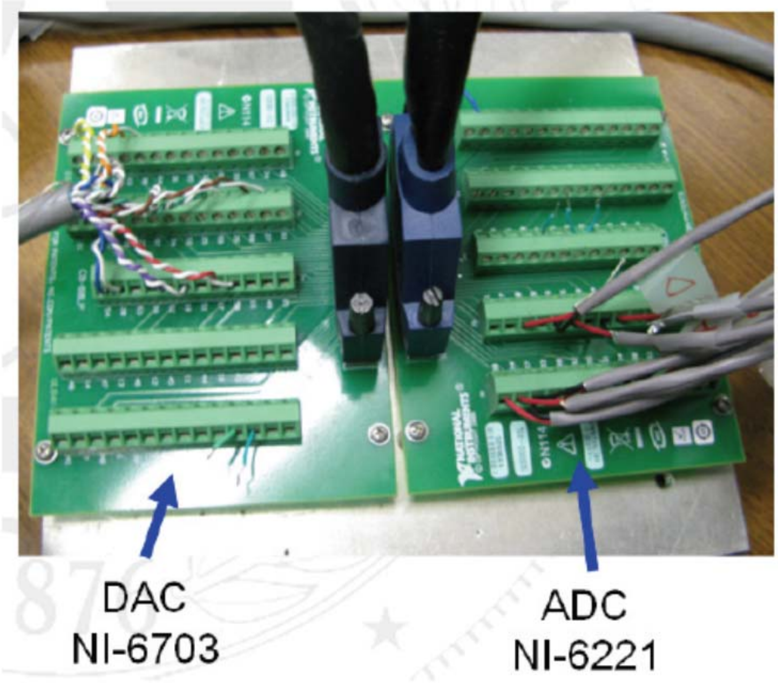


Figure 2-9: Connector for the data acquisition boards [5]

2.6 Instrumentation

2.6.1 Maglev Stage

The maglev stage used in the experimental setup as shown in Figure 2-10 is capable of positioning in 6 DOFs with a travel range of 200 × 220 mm [5]. Beneath the platen, the single moving part, is the double-axis Halbach magnet matrix mentioned before, which is covered with a mirror-finished thin aluminum plate. The platen is currently suspended using three aerostatic bearings, and generates force to move in 6 DOFs using three planar motors, which are attached to the bottom of the platen. Three Hall-effect sensors and three laser distance sensors are used for the 6-DOF position sensing.

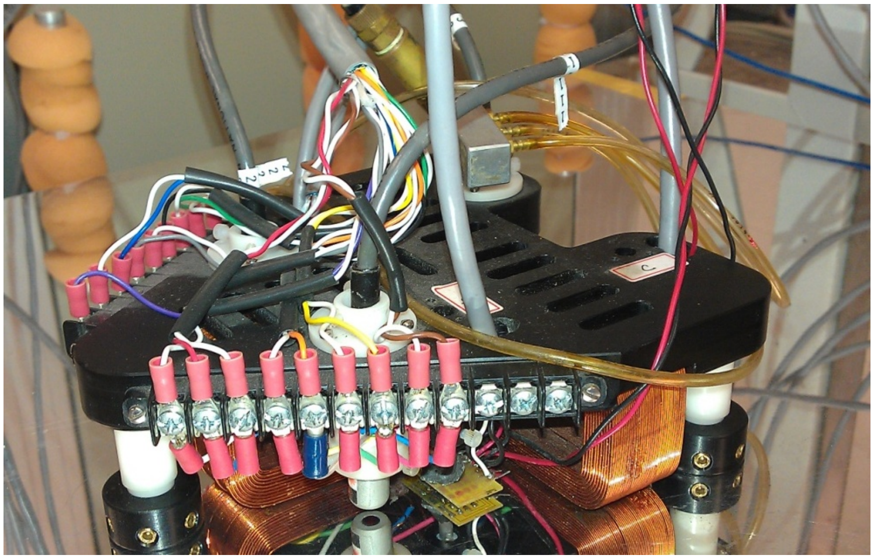


Figure 2-10: Magnetic levitation stage

2.6.2 Hall-effect Sensors

The Hall-effect sensor used in this research is a two-axis Hall-effect sensor 2D-VH-11SO manufactured by Sentron AG[3] [9]. It is an 8-pin, surface mount, small outline integrated circuit. It can measure the magnetic flux density of two orthogonal axes about the chip's surface. A photograph of the 2D-VH-11SO mounted on an IC board is shown in Figure 2-11. The 2D-VH-11SO requires either a constant current source of 2 mA or a constant voltage source of 5 V, these power sources were tested by Kawato [7]. The current power source showed a smaller sensor output fluctuation. Following are the specifications of the sensor for a constant 2 mA current [10]:

- Input resistance: 2.2 kΩ
- Output resistance: 8.5 kΩ

- Sensitivity: 400 mV/T
- Magnetic sensitive volume: 0.25 × 0.25 × 0.20 mm

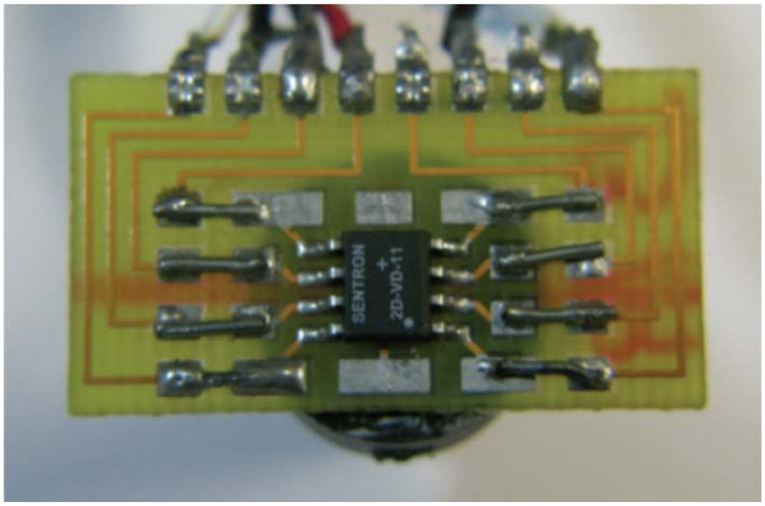


Figure 2-11: 2D-VH-11SO mounted on an IC board

The Hall-effect sensors operate with the power supply circuit, an amplifier, and a low-pass filter to prevent aliasing. The setup includes three sets of power supply circuits for each Hall-effect sensor. To regulate the current source, the current regulator diodes CR200 manufactured by Vishay Siliconix[4] were used. The power supply circuit with the diodes provides the 2-mA constant current and it guarantees ±10% tolerance with good temperature stability.

The position of the moving platen in the base coordinate system fixed to the magnet matrix is determined by the field solution of the magnet matrix and the magnetic flux densities sensed by the Hall-effect sensors. Beyond the sensitive intervals as noted in Figure 2-12, the sensing noise is relatively large and the magnetic flux density may exceed the sensing range of the sensors. A collaboration working approach was used in [5] to take advantages of linearity and low sensing noise in the sensitive intervals.

[3] Sentron AG, Baarerstrasse 73 CH – 6300 Zug, Switzerland

[4] Vishay Intertechnology, Inc. 63 Lincoln Highway, Malvern, PA 19355-2120

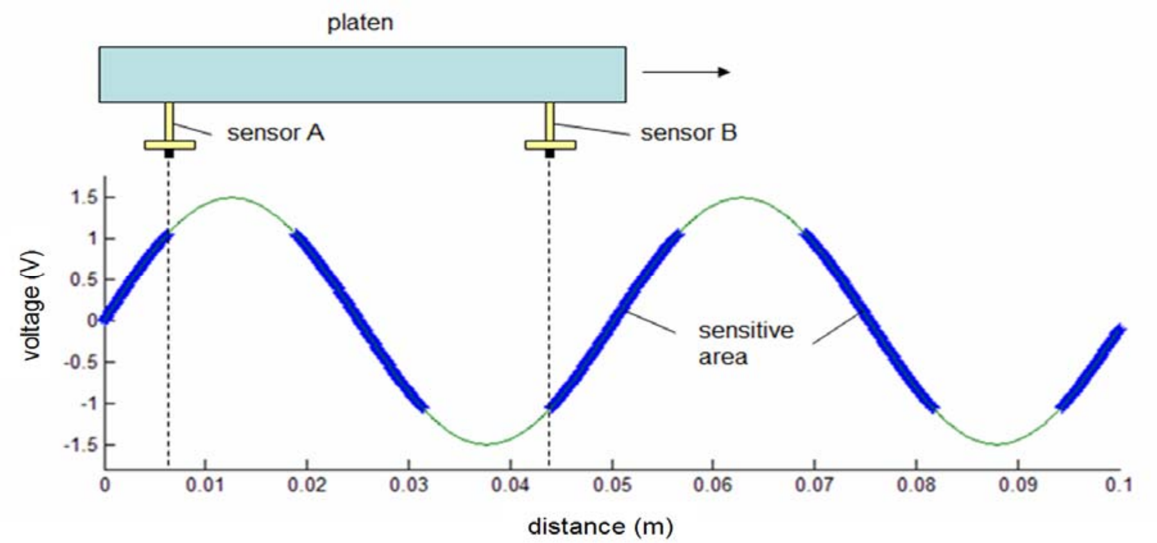


Figure 2-12: Collaboration of two Hall-effect sensors [5]

2.6.3 Amplifier and Analog Filter

The OP Amp TL072ACP manufactured by Texas Instruments[5] is used. To reduce the common-mode noise, the differential amplifier is used. The amplifier requires a ±15-V power source, which is supported by the power supply E3646A manufactured by Agilent. An anti-aliasing filter that is the low-pass filter with the frequency of 200 Hz is placed after an amplifier circuit. This hinders not only aliasing signals, but also the high frequency noise going into the ADC. Sensor noise of 1 mV is measured after the low-pass filter. All Hall-effect sensor outputs after amplifiers and analog filters are connected directly to the NI-6221 ADC board. The specifications of the amplifier and the analog filter circuit are presented in the figure below.

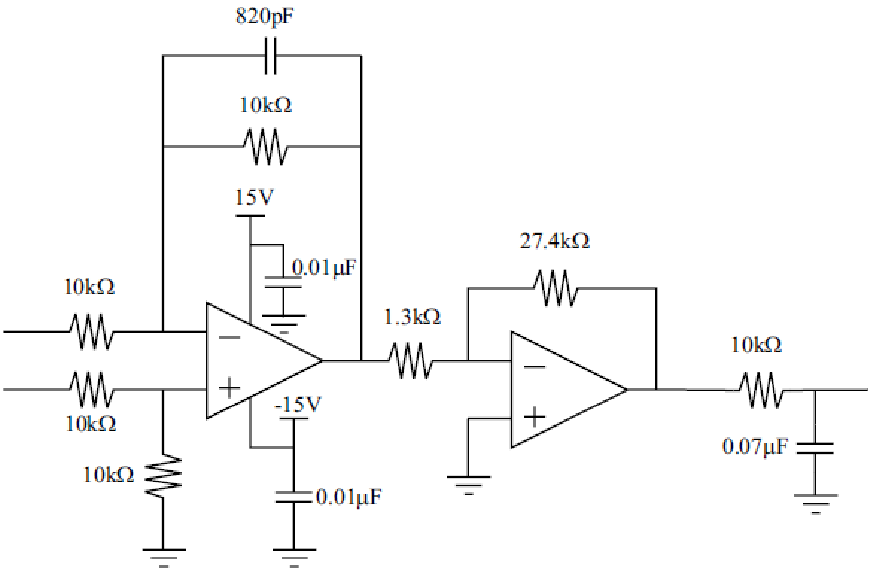


Figure 2-13: Amplifier and filter circuit [7]

2.6.4 Nanogage Laser Distance Sensors

Vertical mode position feedback is performed by three laser distance sensors (Nanogage 100). The vertical mode includes a translation in *z*, pitch, and roll. Nonmagnetic body parts of the sensors do not affect anything on the magnetic field and the planar motors performance [11]. The entire signal-capture range of the laser distance sensor is ±400 μm. However, the area in which the laser sensor can measure linearly is 100 μm in the middle of the signal-capture range. A three-colored LED light indicates the sensor position, red indicates the sensor position loses signals, yellow turns on when the signal is within the capture range and green shows that it's placed at the proper zero-position. By experimental calibrations, the laser distance sensors transmit a voltage signal of ±5 V to the ADC.

A large bandwidth of 100 kHz makes the Nanogage sensor suitable for vibration measurements. The large 3-mm laser head stand-off has benefits in several applications such as handling equipment that requires additional clearance like semiconductor

[5] Texas Instruments Inc. 13532 N. Central Expressway, Dallas, TX 75243-1108

wafers. In addition, the sensors are insensitive to temperature and its small size (2.31-in length and 0.623-in diameter) gives the convenience of installation and equipment development. Figure 2-14 illustrates the working principle of the laser distance sensor.

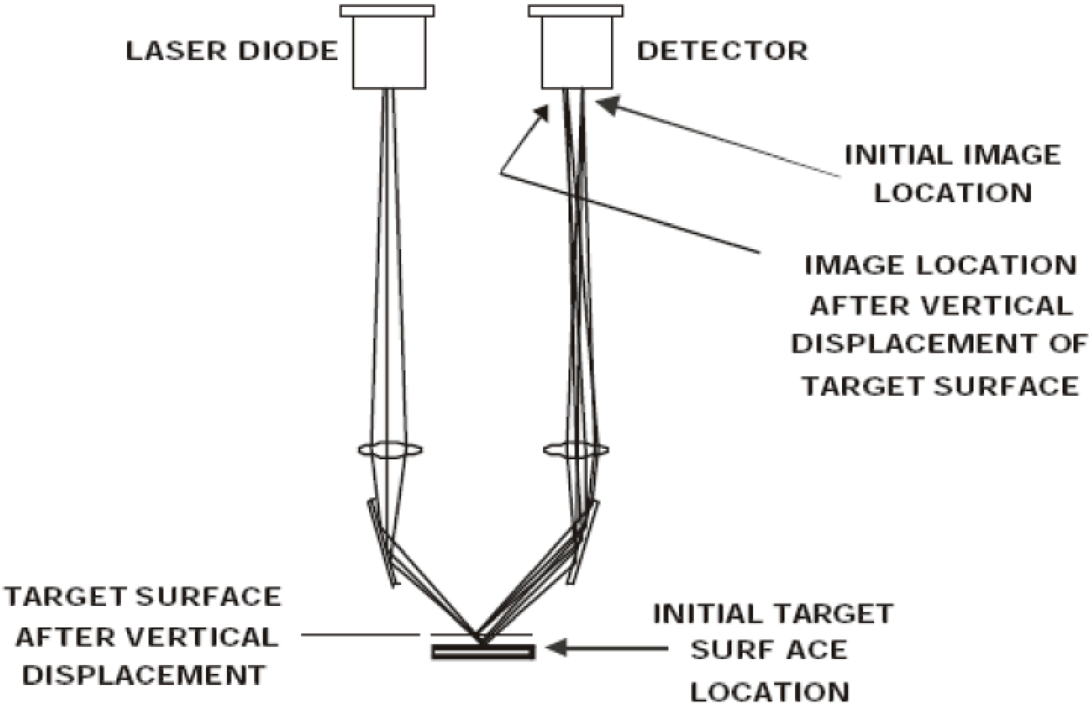

LASER DIODE
DETECTOR
INITIAL IMAGE LOCATION
IMAGE LOCATION AFTER VERTICAL DISPLACEMENT OF TARGET SURFACE
TARGET SURFACE AFTER VERTICAL DISPLACEMENT
INITIAL TARGET SURF ACE LOCATION

Figure 2-14: Basic working principle of the laser distance sensor [12]

The laser distance sensor is a folded optical triangulation sensor composed of a bi-cell detector and has precisely controlled focal spot size. The automatic gain controlled (AGC) laser diode source and the bi-cell detector have a 10-μm gap. The laser beam from the diode is reflected by the target area and then goes into the detector. The output voltage that is proportional to the vertical displacement is generated by the laser signals from the bi-cell detector. The linearity with respect to the voltage output is finally defined by the numerical calibration and corrections [12].

Three laser distance sensors are mounted on the platen. The vertical mode in θ, ψ and z is controlled by feedback from the nanogage laser distance sensors. The planar motion control in the *x-y* plane is achieved with similar methodology because each axis of the magnetic field is independently placed in an orthogonal direction. The Hall-effect sensor and laser distance sensor are shown mounted on the platen in Figure 2-15. Sensor equations are derived from the three nanogage laser distance sensors' geometric parameters. The laser distance sensors produce the displacement information for the position sensing and a relationship matrix between the sensor readings and vertical motions is derived, the equations for the 3 DOFs can be derived by taking the inverse coefficient matrix transformation, as shown in [5].

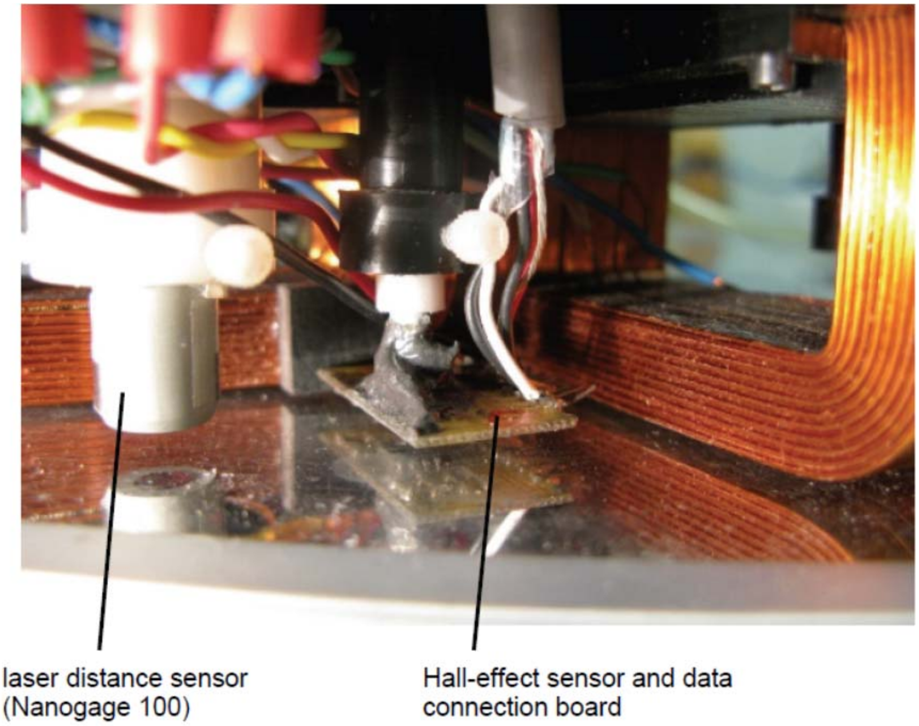


Figure 2-15: Photograph of the laser distance sensor and the Hall-effect sensor

CHAPTER III

DYNAMIC MODELING

With the mass and inertia tensor of the platen and the resistance and inductance of the phase winding, the dynamic model of the platen has been derived by Yu [5] and is described in this chapter. The specifications of the levitation system are also given, and the derivation of the linearized state-space models for vertical and lateral motions.

3.1 Mass and Inertia Tensor of the Platen

The total mass of the platen is M = 1.54 kg, so its weight is 14.91 N, which includes the Delrin triangular frame, three planar-motor coils, three Hall-effect sensors, three air-bearing assemblies, the air distributor and three vertical sensors. The maximum vertical force generated by the 3 levitation motors is 17.15 N, so it can be fully levitated. The moments of inertia are calculated about the platen center of mass

$$I = \begin{bmatrix} I_{xx} & -I_{xy} & -I_{xz} \\ -I_{yx} & I_{yy} & -I_{yz} \\ -I_{zx} & -I_{zy} & I_{zz} \end{bmatrix} = \begin{bmatrix} 0.001898 & -0.000043 & 0.000041 \\ -0.000043 & 0.002153 & 0.000179 \\ 0.000041 & 0.000179 & 0.003375 \end{bmatrix} \tag{3.1}$$

in the unit of kg-m². The products of inertia, $I_{xy}, I_{xz}, I_{yx}, I_{yz}, I_{zx}$, and I_{zy} are neglected in the derivation of the dynamic model because any of them is less than 10 times the value of the principal moments of inertia, I_{xx}, I_{yy} and I_{zz},.

3.2 Specifications of the Positioner

The specifications of the positioner are shown below.

- Number of phases, q = 3
- Phase inductance = 15.264 mH
- Phase resistance = 19.44 Ω
- Nominal phase current = 0.56 A

- Maximum phase current = 1.26 A
- Nominal phase voltage = 11.6 V
- Maximum phase voltage = 26.1 V

The motion capabilities of the maglev positioner are:

- Planar travel range = 120 mm × 120 mm
- Vertical range = 100 μm
- Maximum velocity = 17.5 mm/s

3.3 Decoupled Equations of Motion

The linearized force equations and vertical and lateral linear equations of motions are derived in this section.

3.3.1 *DQ*-decomposition

The *DQ* decomposition is generally used in the rotary machinery to generate torque. In this case both the direct-axis (*D*-axis) and the quadrature-axis (*Q*-axis) are attached to the mover so that these two axes move together with the platen. Therefore there is no dependence in the force equations with respect to the stator which is the magnetic matrix in the *DQ* frame, which means that the nonlinear term can be eliminated. The *D*-axis is parallel to the stator magnetic axis and the *Q*-axis is perpendicular with *D*-axis as shown in Figure 3-1. The vertical motion is affected by the *D*-component current, and the horizontal driving forces are affected by the *Q*-component current. In order to control the two-DOF suspension and the driving force, the planar motor requires the two decoupled orthogonal force components.

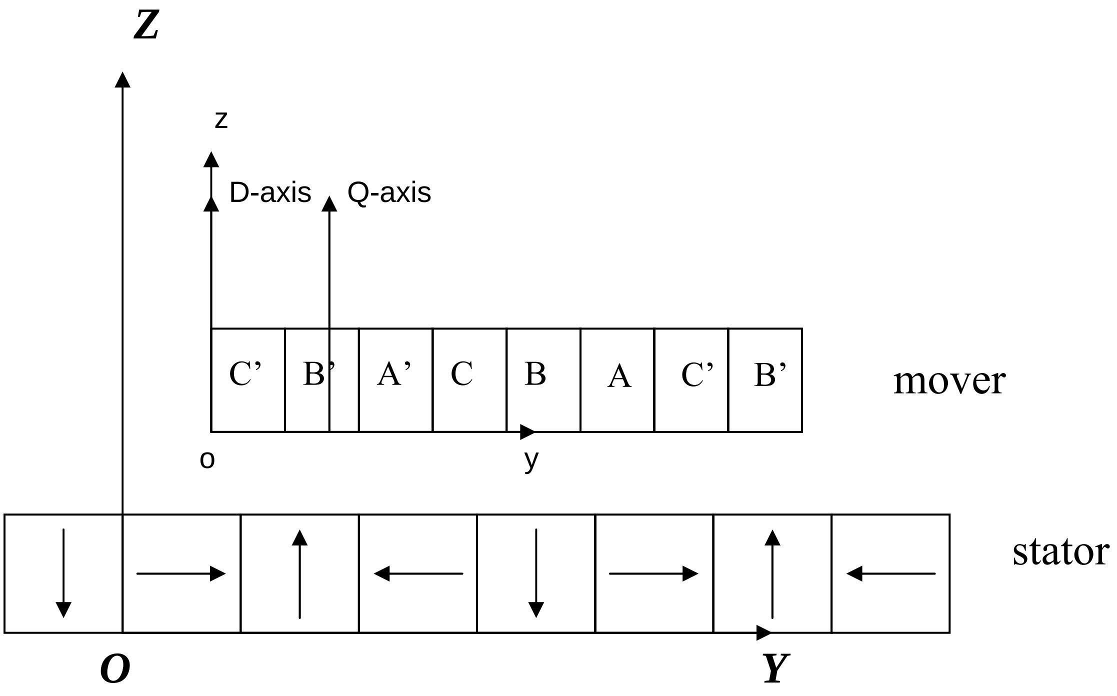


Figure 3-1: *DQ* coordinates attached to the platen [1]

3.3.2 Linearized Force Equations

The following equation shows the relationship between the phase currents and the magnetic force generated by one pitch of the levitation motor [1].

$$\begin{bmatrix} f_y \\ f_z \end{bmatrix} = \frac{1}{2}\mu_0 M_0 \eta_0 N_m G e^{-\gamma_1 z_0} \begin{bmatrix} \cos\gamma_1 y_0 & \sin\gamma_1 y_0 \\ -\sin\gamma_1 y_0 & \cos\gamma_1 y_0 \end{bmatrix} \begin{bmatrix} i_a \\ i_b \end{bmatrix}, \tag{3.2}$$

where the total lateral force is represented as f_y, and f_z is the *z*-directed force, respectively, i_a and i_b are the peak current components and the constant *G* is following.

$$G = \frac{\sqrt{2}wl^2}{\pi^2}(1-e^{-\gamma_1\Gamma})(1-e^{-\gamma_1\Delta}), \tag{3.3}$$

where y_0 is the horizontal relative displacement of the motors A and B from the initial position. If the variable y_0 changes to x_0, the same force equation is applied in motor C. The parameters of the positioner's geometry are shown below in Table 3-1.

Table 3-1 Geometric Parameters [5]

Variable	**Value**
Motor Geometric constant, G	$1.072 \cdot 10^{-5}$ m^3
Magnet remanence, $\mu_0 M_0$	0.71 T
Width of one magnet array, *w*	0.5 in
Magnet thickness, Δ	12.7 mm
Winding thickness, Γ	10.16 mm
Pitch length, *l*	50.98 mm
Absolute value of the fundamental wave number, $\gamma_1 = 2\pi / l$	123.25 m^{-1}
Nominal motor air gap, z_0	2.3 mm
Turn density, η_0	$3.5246 \cdot 10^6$ turns/m²
Number of magnet pitches, N_m	2

3.3.3 Vertical Equations of Motion

The vertical equation of motion is represented as follows,

$$f_z - Mg = 3 \cdot \frac{1}{2} \mu_0 M_0 \eta_0 N_m G (e^{-\gamma_1 z_0} i_D - \gamma_1 e^{-\gamma_1 z_0} i_D z_0 + e^{-\gamma_1 z_0} i_D) \tag{3.4}$$

The factor of three is multiplied because there are 3 planar motors. Equation (3.5) is showed with replacing the weight *Mg* to $3 \cdot \frac{1}{2} \mu_0 M_0 \eta_0 N_m G e^{-\gamma_1 z_0} i_D$, where the platen achieves a dynamic equilibrium. As a result, the force equation is,

$$f_z = 3 \cdot \frac{1}{2} \mu_0 M_0 \eta_0 N_m G e^{-\gamma_1 z_0} i_D - 3 \cdot \frac{1}{2} \mu_0 M_0 \eta_0 N_m G \lambda_1 e^{-\gamma_1 z_0} i_D z_0 \tag{3.5}$$

and the incremental equation of motion is,

$$M \frac{d^2 z_0}{dt^2} + \mu_0 M_0 \eta_0 N_m G e^{-\gamma_1 z_0} i_D z_0 = 3 \cdot \frac{1}{2} \mu_0 M_0 \eta_0 N_m G \lambda_1 e^{-\gamma_1 z_0} i_D . \tag{3.6}$$

3.3.4 Lateral Equations of Motion

The equilibrium condition for the lateral direction comes from (3.2). The force equation in the x-axis is,

$$f_x = \frac{1}{2}\mu_0 M_0 \eta_0 N_m G e^{-\gamma_1 z_0}(\cos(\gamma_1 y_0) i_a + \sin(\gamma_1 y_0) i_b) \tag{3.7}$$

The force equation in the y-axis is,

$$f_y = 2\cdot\frac{1}{2}\mu_0 M_0 \eta_0 N_m G e^{-\gamma_1 z_0}(\cos(\gamma_1 y_0) i_a + \sin(\gamma_1 y_0) i_b)\,. \tag{3.8}$$

Since there are two planar motors generating the y-directional force, the factor of two is multiplied. The incremental equation of motion is showed as followed,

$$M\frac{d^2 y_0}{dt^2} + \mu_0 M_0 \eta_0 N_m G e^{-\gamma_1 z_0} i_D y_0 = 2\cdot\frac{1}{2}\mu_0 M_0 \eta_0 N_m G \gamma_1 e^{-\gamma_1 z_0} i_Q\,. \tag{3.9}$$

3.4 Dynamic Model of the System

As preceding procedures, the linearized equations of motions are derived. We regard the model of the platen as a pure mass without any friction, which means that there is no spring or damper attached to the system.

3.4.1 Linearized equations of Motion in the Horizontal and Vertical Modes

Several commanded forces and the moments of the platen are related to the force allocation of each motor. Figure 3-2 illustrates the free-body diagrams for force allocation. The coordinates for the moving frame and that of the inertial frame coincide at the origin. The magnetic force generated by each motor is considered as the concentrated force applied at the center of each motor. The origin is the center of mass of the platen and the distances are denoted by L.

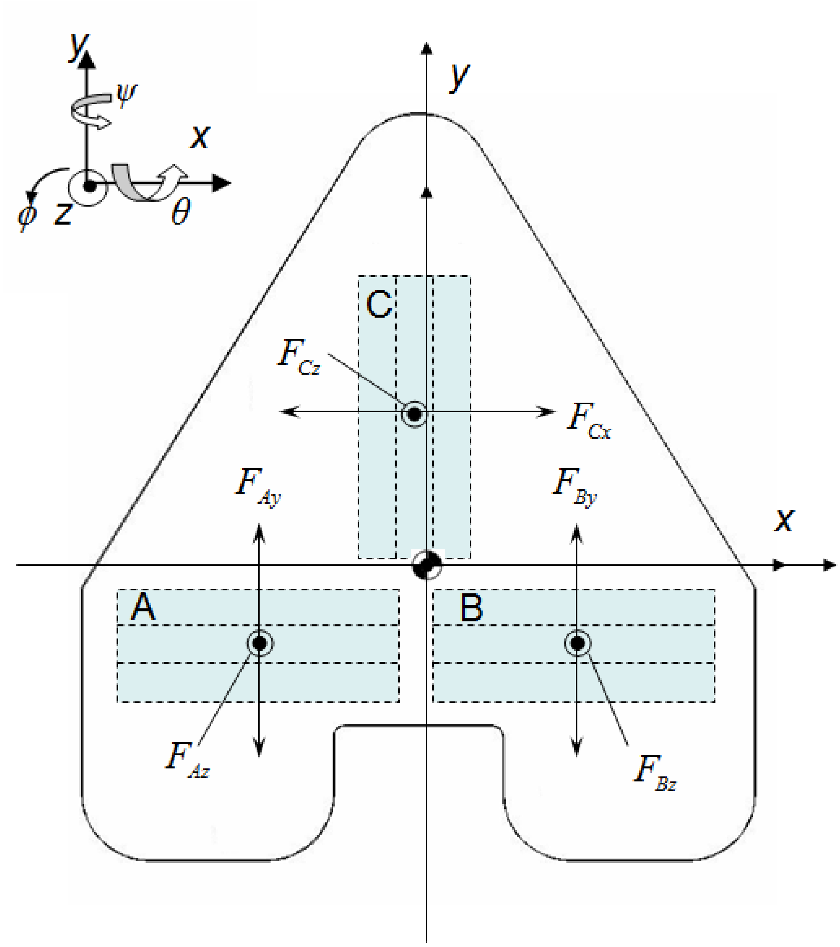


Figure 3-2: Free-body diagram for the force allocation [5]

Since the platen is modeled as a pure mass without friction in full levitation. The following equations represent the dynamics of the pure mass model.

$$M\frac{d^2x}{dt^2} = f_x \tag{3.10}$$

$$f_x = f_{Cx}, \tag{3.11}$$

The mass of platen is M = 1.54 kg, and f_x is the magnetic modal force generated by motor C. The x-directional force is generated by one motor.

$$M\frac{d^2y}{dt^2} = f_y \tag{3.12}$$

$$f_y = f_{Ay} + f_{By} \tag{3.13}$$

where f_y is the y-directional magnetic modal force generated by motors A and B.

In order to control the rotational motion around the z-axis, the following differential equations are used.

$$I_{zz}\frac{d^2\phi}{dt^2} = \tau_z, \tag{3.14}$$

$$\tau = -f_{Ax}l_{Ay} + f_{By}l_{Bx} - f_{Cx}l_{Cy}, \tag{3.15}$$

where the principle moment of inertia for rotation about the z-axis is I_{zz} = 0.003375 kg-m², and τ_z is a torque generated by the interaction between motors A, B, and C from the magnetic origin about the z-axis. Since the three equations of motion in the planar motion have been derived, these are put into the state-space model representation as follows.

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\phi} \\ \dot{h} \\ \dot{u} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ \phi \\ h \\ u \\ v \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \frac{c}{m} \\ \frac{c}{m} & \frac{c}{m} & 0 \\ -\frac{c}{I_{zz}}l_{Az} & \frac{c}{I_{zz}}l_{Bz} & -\frac{c}{I_{zz}}l_{Cy} \end{bmatrix} \begin{bmatrix} i_{AQ} \\ i_{BQ} \\ i_{CQ} \end{bmatrix} \tag{3.16}$$

$$Y_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ \phi \\ h \\ u \\ v \end{bmatrix} \tag{3.17}$$

The equations of motion for the vertical mode are represented as follows, the vertical direction is designed as a spring-mass system in z because the platen is levitated by all three levitation motors.

$$M\frac{d^2z}{dt^2} = f_z - K_z z \tag{3.18}$$

$$f_z = f_{Az} + f_{Bz} + f_{Cz} \tag{3.19}$$

where, K_z is the effective spring constant of the levitation motor derived by experiments based on Hooke's law, its value was derived experimentally, and f_z is the vertical directional force generated by all three motors at the same time in order to lift the platen up.

Because of the three magnetic springs in the three levitation motors, the dynamics in the rotation around the *x*- and *y*-axes are regarded as a spring-mass system as follows,

$$I_{xx}\frac{d^2\theta}{dt^2} = \tau_\theta - K_\theta\theta \tag{3.20}$$

$$\tau_\theta = -f_{Az}l_{Ay} - f_{Bz}l_{By} + f_{Cz}l_{Cy} \tag{3.21}$$

$$I_{yy}\frac{d^2\psi}{dt^2} = \tau_\psi - K_\psi\psi \tag{3.22}$$

$$\tau_\psi = -f_{Az}l_{Ax} - f_{Bz}l_{Bx} + f_{Cz}l_{Cx} \tag{3.23}$$

where θ and ψ are the rotations around the *x*- and *y*-axes, respectively. τ_θ and τ_ψ are the torque around the *x*- and *y*-axes, respectively. K_θ and K_ψ are the effective torsional spring constants about the *x*- and *y*-axes determined by experiments. The principle moments of inertia are I_{xx} = 0.001898 kg-m² and I_{yy} = 0.002153 kg-m² in the *x*- and *y*-axes, respectively. The state-space model of the vertical mode is as follows.

$$\begin{bmatrix} \dot{z} \\ \dot{\theta} \\ \dot{\psi} \\ \dot{p} \\ \dot{q} \\ \dot{w} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ -\frac{K_z}{m} & 0 & 0 & 0 & 0 & 0 \\ 0 & -\frac{K_\theta}{I_{xx}} & 0 & 0 & 0 & 0 \\ 0 & 0 & -\frac{K_\phi}{I_{yy}} & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} z \\ \theta \\ \psi \\ p \\ q \\ w \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ \frac{c}{m} & \frac{c}{m} & \frac{c}{m} \\ -\frac{c}{I_{xx}} l_{Ay} & -\frac{c}{I_{xx}} l_{By} & \frac{c}{I_{xx}} l_{Cy} \\ \frac{c}{I_{yy}} l_{Ax} & -\frac{c}{I_{Bx}} & \frac{c}{I_{yy}} l_{Cx} \end{bmatrix} \begin{bmatrix} i_{AD} \\ i_{BD} \\ i_{CD} \end{bmatrix} \quad (3.24)$$

$$Y_2 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} z \\ \theta \\ \psi \\ p \\ q \\ w \end{bmatrix} \quad (3.25)$$

CHAPTER IV

MULTIVARIABLE CONTROLLER DESIGNS

Essentially this multi-dimensional positioner is a multi-input, multi-output (MIMO) system. Due to the *DQ* decomposition, there are 3 current inputs from the *D*-current of the *DQ* decomposition, which is the current for the planar motion and the outputs which is the planar position and angles, x, y, and φ. Then we have the 3 inputs from the *Q*-current and the 3 outputs which is the levitation position and angles, z, θ and ψ. The MIMO control system design focuses on the dynamic of the whole system instead of the dynamic on each individual axis as with a single-input, single-output (SISO) design [13].

The Linear Quadratic Gaussian (LQG) control methodology is a combined optimal state estimation and optimal state feedback control. The LQG controller is normally comprised of a Kalman filter acting as the state estimator, and the Linear Quadratic Regulator (LQR) controller performs state feedback. Since the Kalman filter and the model-based compensator have a similar structure, the Discrete Kalman Filter (DKF) can be used in the LQG control system design. However it was observed that the outputs from the Hall-effect sensors, are very noisy and fluctuated a lot. A regular Discrete Kalman filter helped in obtaining better estimates and reducing the fluctuation and noise by a certain degree, but there was certainly more that could be done in order to get a cleaner signal from the sensor.

First a digital filter was designed for the Hall-effect sensors, this filter essentially acts as a “pre-filter” and dissipates the static noise with a frequency of 50 Hz or higher. This reduced significantly the biased noise, the next step is to modify the observer of the both the planar and the vertical LQG controllers to further reduce the noise of the sensors and obtain the velocity estimates for the positions. The analysis of the signal and design of the digital filter is discussed in the next chapter.

4.1 Linear Quadratic Regulator for Translational Mode

By plugging the corresponding values into the Eqs. (3.16) and (3.17), the state-space model of the system for the planar motion can be represented as in Eqs. (4.1) and (4.2).

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\phi} \\ \dot{h} \\ \dot{u} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ \phi \\ h \\ u \\ v \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 13.255 \\ 13.255 & 13.255 & 0 \\ -237.91 & 226.65 & -208.87 \end{bmatrix} \begin{bmatrix} i_{AQ} \\ i_{BQ} \\ i_{CQ} \end{bmatrix}, \tag{4.1}$$

$$Y_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ \phi \\ h \\ u \\ v \end{bmatrix} \tag{4.2}$$

The above horizontal dynamic system can be presented as Eq. (4.3), where *x* is the state vector and *u* is input vector.

$$\dot{x}(t) = Ax(t) + Bu(t) \tag{4.3}$$

The LQR problem, where all the states were known, was the deterministic initial value problem: with non-zero initial state *x*(0), find the input signal *u*(*t*) which took the system to the zero state (*x* = 0) in an optimal manner, i.e. by minimizing the deterministic cost [14].

$$J_r = \int_0^\infty (x(t)^T Qx(t) + u(t)^T Ru(t))dt \tag{4.4}$$

The optimal solution (for any initial state) is $u(t) = -K_r x(t)$, where

$$K_r = R^{-1}B^T P \ , \tag{4.5}$$

and $P = P^T \geq 0$ was the unique positive semi-definite solution of the algebraic Riccati equation

$$A^T P + PA - PBR^{-1}B^T P + Q = 0 \ , \tag{4.6}$$

where Q and R are constant weighting matrices needed to be designed to penalize some state variables (or inputs) more than others to meet dynamic requirements. It is necessary that Q be positive semidefinite, and R be positive definite [15]. However LQR is an optimal full-state feedback pole placement approach. It provides an optimal way to place the eigenvalues of the multivariable system by designing appropriate Q and R matrices in Eqs. (4.5) and (4.6).

Since the LQ controller is a regulator, it will regulate the output of the system to be zero. In order to make the positioner track non-zero commands, the error dynamical model of the system is needed for the LQ servo design. The error can be defines as following.

$$\Delta x = x - x^* \tag{4.7}$$

$$\Delta y = y - y^* \tag{4.8}$$

$$\Delta r = r - r^* \tag{4.9}$$

$$\Delta u = u - u^* \ , \tag{4.10}$$

where the variable with an asterisk are the final values of the corresponding state. Since the final values of the state are constant values, the derivatives of those values will be zero. As a result, we can get the following equations.

$$\Delta \dot{x}(t) = A\Delta x(t) + B\Delta u(t) \ , \tag{4.11}$$

$$\Delta y(t) = Cx(t) \ , \tag{4.12}$$

where the *A*, *B*, and *C* matrices are the same as the ones in the original system. As a result, if the LQR controller is designed based on the error dynamics in Eqs. (4.11) and (4.12), the positioning system can track the input reference. So the control law will be

$$\Delta u(t) = -K_r \Delta x(t), \tag{4.13}$$

where K_r is the optimal solution for the Riccati eqn by designing the *Q* and *R* matrices. Since the input to the system is *u* not Δu, from Eq. (4.10) it can be known that the input to the system should be

$$u = \Delta u + u^*. \tag{4.14}$$

Once the system is in steady state, the steady-state input u^* will be zero, which can be seen from Eq. (4.1) by letting the left hand side of the equation be zero. Therefore, Eq. (4.14) can be simplified as

$$u = -K_r \Delta x(t). \tag{4.15}$$

Usually the purpose of a precision positioning system is to achieve high resolution. The precision mainly depends on the position sensors, but the optimized control algorithm can improve the precision positioning performances. Since the original plant and the controller do not include the integrators, steady-state errors in the closed-loop position control may exist. In order to reduce the steady-state errors in position, it is necessary to add three pure integrators in the position outputs. As a result, the augmented system should have nine states altogether. The elements of the *A, B* and *C* matrix are calculated as follows:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\phi} \\ \dot{z}_1 \\ \dot{z}_2 \\ \dot{z}_3 \\ \dot{h} \\ \dot{u} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} 0&0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&0&1&0 \\ 0&0&0&0&0&0&0&0&1 \\ 1&0&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0 \end{bmatrix} \begin{bmatrix} x \\ y \\ \phi \\ z_1 \\ z_2 \\ z_3 \\ h \\ u \\ v \end{bmatrix} + \begin{bmatrix} 0&0&0 \\ 0&0&0 \\ 0&0&0 \\ 0&0&0 \\ 0&0&0 \\ 0&0&0 \\ 0&0&13.255 \\ 13.255&13.255&0 \\ -237.91&226.65&-208.87 \end{bmatrix} \begin{bmatrix} i_{AQ} \\ i_{BQ} \\ i_{CQ} \end{bmatrix} \quad (4.16)$$

$$Y_1 = \begin{bmatrix} 1&0&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0&0 \end{bmatrix} \begin{bmatrix} x \\ y \\ \phi \\ z_1 \\ z_2 \\ z_3 \\ h \\ u \\ v \end{bmatrix} \quad (4.17)$$

4.2 Linear Quadratic Gaussian Control for Translational Mode

LQG control methodology is a combined optimal state estimation and optimal state-feedback control. The LQG controller is comprised of a Kalman filter acting as the state estimator, and the LQR controller performs state feedback. Since the discrete hybrid filter and the Model-Based Compensator (MBC) have a similar structure, the hybrid filter designed in the next section can be used in the LQG control system design, the control structure of the MBC is shown in Figure 4-1. That way in this the LQG control design only one Riccati equation have to be solved, as opposed of two using a model-based observer. Due to its optimization in control effort, LQG control is widely used in the multivariable control industry.

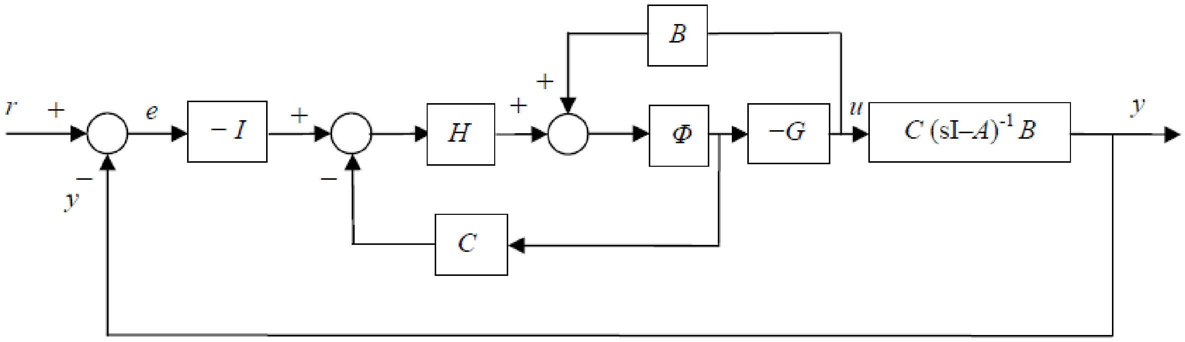


Figure 4-1: Control Structure of LQG controller [18]

There are two assumptions in LQG control design. First is that the plant dynamics is linear and known. Second is that the measurement noise and disturbance signals (process noise) are stochastic with known statistical properties [16]. That is, the plant model can be presented as follows.

$$\dot{x}(t) = Ax(t) + Bu(t) + L\zeta(t) \ , \tag{4.18}$$

$$y(t) = Cx(t) + \theta(t) \ , \tag{4.19}$$

where, the *A, B, L*, and *C* are constant matrices. And $\zeta(t)$ is assumed to be the stationary, white Gaussian disturbance (process noise), which has the following properties

$$E\{\zeta(t)\}=0 \ , \tag{4.20}$$

$$\text{Cov}\{\zeta(t),\zeta(t)\}=\Psi\delta(t-\tau) \ , \tag{4.21}$$

$$\Psi = \Psi^T>0 \ , \tag{4.22}$$

Similarly, θ(t) is also assumed to be stationary, white Gaussian noise coming from the measurement inputs (sensor noise). So its expectation and covariance has the following properties.

$$E\{\theta(t)\}=0 \ , \tag{4.23}$$

$$\mathrm{Cov}\left\{\theta(t),\theta(t)\right\}=\Theta\delta(t-\tau)\ , \tag{4.24}$$

$$\Theta=\Theta^{T}>0\ , \tag{4.25}$$

where $E\left\{\cdot\right\}$ is the expectation operator and $\delta(t-\tau)$ is a delta function [17].

As it was shown in the previous section the optimal control law of the regulator is $u=-K_{r}\Delta x(t)$, which is to minimize the cost function *J*. The matrix K_r is determined by the control algebraic Ricatti equation (CARE). *P* is the unique positive-semidefinite solution of the CARE in (4.6). Those equations further verified that the LQG control methodology is the combination of a Kalman filter and a LQR control law.

LQG is introduced into the control system design in the planar direction for the expectation that the LQG algorithm can optimize the pole location for the controller to get a better dynamic performance in that direction. To eliminate the steady-state error, three integrators are also needed to be implemented in the original plant whose dynamics are shown in Eqs. (4.16) and (4.17).

There are three inputs i_{AQ}, i_{BQ} and i_{CQ} as well as there are three outputs *x*, *y*, and *r*. The *Q* and *R* matrices can be designed by the LQR state feedback method previously mentioned, because the (*A*, *B*) is stabilizable. The matrices *Q* and *R* are

$$Q=\mathrm{diag}\left(\begin{bmatrix}1e4 & 1e4 & 1e4 & 1e8 & 1e8 & 1e8 & 1e1 & 1e1 & 1e1\end{bmatrix}\right) \tag{4.26}$$

$$R=\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{bmatrix} \tag{4.27}$$

The design of *Q* matrix was achieved by much iteration. To emphasize the effect of the integrators, much heavier weight values were given to the integrator states (z_1, z_2 and z_3) than those of the position and velocity states, this was done in order to remove the steady state errors [18-19]. The gain matrix K_r can be calculated by the LQR methodology in Matlab as follows,

$$K_r = \begin{bmatrix} -160.2 & 253.9 & -92 & -3523 & 6372 & -3183 & -3.46 & 4.74 & -1.18 \\ 158.3 & 264.7 & 88.2 & 3464 & 6629 & 3045 & 3.44 & 4.95 & 1.13 \\ 359.3 & -1.88 & -71.7 & 7949 & -59.5 & -2595 & 7.73 & -0.02 & -0.84 \end{bmatrix} \qquad (4.28)$$

The implementation results are presented in Chapter VI and all programming codes are attached.

4.3 Linear Quadratic Gaussian Control for the Vertical Mode

By plugging the corresponding values into Eqs. (3.24) and (3.25), the state-space model of the system for the levitating motion can be represented as in Eqs. (4.29) and (4.30).

$$\begin{bmatrix} \dot{z} \\ \dot{\theta} \\ \dot{\psi} \\ \dot{p} \\ \dot{q} \\ \dot{w} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ -408 & 0 & 0 & 0 & 0 & 0 \\ 0 & -34247 & 0 & 0 & 0 & 0 \\ 0 & 0 & -40409 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} z \\ \theta \\ \psi \\ p \\ q \\ w \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 13.26 & 13.26 & 13.26 \\ -209.9 & -209.9 & 370.4 \\ 371.9 & -354.3 & 12.73 \end{bmatrix} \begin{bmatrix} i_{AD} \\ i_{BD} \\ i_{CD} \end{bmatrix} \quad (4.29)$$

$$Y_2 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} z \\ \theta \\ \psi \\ p \\ q \\ w \end{bmatrix} \qquad (4.30)$$

Once again in order to emphasize the position of the system in the upwards motion, three integrators are added to the state-space model. The original plant and the controller do not include any integrators and steady-state errors in the closed-loop position control may exist. In order to reduce the steady-state errors, three pure integrators are employed in the original system as follows.

$$\begin{bmatrix} \dot{z} \\ \dot{\theta} \\ \dot{\psi} \\ \dot{z}_4 \\ \dot{z}_5 \\ \dot{z}_6 \\ \dot{p} \\ \dot{q} \\ \dot{w} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -408 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -34247 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -40409 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} z \\ \theta \\ \psi \\ z_4 \\ z_5 \\ z_6 \\ p \\ q \\ w \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 13.26 & 13.26 & 13.26 \\ -209.9 & -209.9 & 370.4 \\ 371.9 & -354.3 & 12.73 \end{bmatrix} \begin{bmatrix} i_{AD} \\ i_{BD} \\ i_{CD} \end{bmatrix} \tag{4.31}$$

$$Y_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} z \\ \theta \\ \psi \\ z_4 \\ z_5 \\ z_6 \\ p \\ q \\ w \end{bmatrix} \tag{4.32}$$

Just like in the previous section, the LQG control methodology is applied to the system. As stated before the LQG controller is comprised of a Kalman filter acting as the state estimator, and the LQR controller performs state feedback, however the hybrid filter will act as the model-based compensator as stated before. The LQR control law is designed just like in Section 4.1, and the 6-axis discrete hybrid filter designed in the next section will provide optimal position estimates and velocity estimates. That way, this LQG controller design also has one Riccati equation to solve.

The two assumptions in LQG control design are also satisfied for the vertical mode, since the plant dynamics is linear and known, and the measurement noise and disturbance signals (process noise) are stochastic with known statistical properties. Therefore, the plant model can be presented as follows.

$$\dot{x}(t) = Ax(t) + Bu(t) + L\zeta(t) \tag{4.33}$$

$$y(t) = Cx(t) + \theta(t) \ , \tag{4.34}$$

where the *A, B, L*, and *C* are constant matrices, and $\zeta(t)$ is assumed to be the stationary, white Gaussian disturbance (process noise), which has properties shown in Eqs (4.20-4.22).

Similarly, θ(t) is also assumed to be stationary, white Gaussian noise coming from the measurement inputs (sensor noise). Its expectation and covariance has the properties shown in Eqs (4.23-4.25), where $E\{\cdot\}$ is the expectation operator and $\delta(t-\tau)$ is a delta function [17].

As it was shown in Section 4.1 the optimal control law of the regulator is $u = -K_r \Delta x(t)$, which is considered to minimize the cost function *J*. The matrix K_r is determined by the control algebraic Ricatti equation (CARE). *P* is the unique positive-semidefinite solution of the CARE in (4.6).

There are three inputs i_{AD}, i_{BD} and i_{CD} as well as three outputs *z*, θ, and ψ. The *Q* and *R* matrices can be designed by the LQR state feedback method previously mentioned, because the (*A*, *B*) is stabilizable. The matrices *Q* and *R* are

$$Q = \text{diag}\left([1e4 \quad 1e4 \quad 1e4 \quad 1e9 \quad 1e9 \quad 1e9 \quad 1e1 \quad 1e1 \quad 1e1]\right) \tag{4.35}$$

$$R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.36}$$

The design of Q matrix was achieved by much iteration. Heavier weight values to the displacement states (z, θ, and ψ) and the integrator states (z_4, z_5, and z_6) were given than those of the velocity states. In order to remove the steady state errors, the integrator states weights are much higher than other values [18-19]. The gain matrix K_r can be calculated by the LQR methodology in Matlab as follows,

$$K_r = \begin{bmatrix} 405.5 & -40.9 & 53.2 & 13869 & -2672 & 4109 & 6.1 & -0.9 & 1.4 \\ 442.5 & -40.5 & -49.1 & 15130 & -2633 & -3954 & 6.7 & -0.9 & -1.3 \\ 480 & 69.1 & 2.4 & 16407 & 4837 & 89.8 & 7.3 & 1.63 & 0.03 \end{bmatrix} \tag{4.37}$$

The implementation results are presented in Chapter VIII, and all programming codes are attached in Appendix C.

4.4 State Estimator for Feedback Control

A method of combining discrete-Kalman ($\mathcal{H}_2$) filtering and minimax ($\mathcal{H}_\infty$) has been proposed in [20]. Kalman filtering is an optimal estimation method which minimizes the variance of the estimation error and assumes that the noise inputs have known statistical properties, which makes this filter lack robustness from the errors of the assumed noise statistics [20]. Unfortunately, the assumption that the statistical properties of the noise are known limits the application of the Kalman filter.

The minimax filter aims to minimize the magnitude of the maximum singular value of the transfer function from the noise to the estimation error. In other words, it tries to minimize the worst possible estimated values. If the Kalman filter assumes too much, the minimax filter approach assumes too little, which motivates the interest in combining these two filters into a hybrid $\mathcal{H}_2/\mathcal{H}_\infty$ filter and see if it can provide the best of both worlds. In order to design the hybrid filter that will provide optimal displacement measurements and also estimates of the velocity and angular velocity. The states of the hybrid filter are defined as

$$x_k = \begin{Bmatrix} z_k \\ \dot{z}_k \\ \ddot{z}_k \end{Bmatrix}, \text{ for } z_k = \begin{Bmatrix} X_0 \\ Y_0 \\ \phi_0 \\ Z_0 \\ \Theta_0 \\ \Psi_0 \end{Bmatrix}. \tag{4.38}$$

For continuous time, the dynamic model and measurement (estimated) model are,

$$\begin{gathered} \dot{x}(t) = \begin{Bmatrix} \dot{z}(t) \\ \ddot{z}(t) \\ \dddot{z}(t) \end{Bmatrix} = \begin{bmatrix} 0 & I_{6\times6} & 0 \\ 0 & 0 & I_{6\times6} \\ 0 & 0 & 0 \end{bmatrix} \begin{Bmatrix} z(t) \\ \dot{z}(t) \\ \ddot{z}(t) \end{Bmatrix} + \begin{bmatrix} 0 \\ 0 \\ I_{6\times6} \end{bmatrix} w(t) \\ \tilde{y}(t) = \hat{z}(t) + v(t) = \begin{bmatrix} I_{6\times6} & 0 & 0 \end{bmatrix} x(t) + v(t) \end{gathered} \tag{4.39}$$

The inputs to the hybrid filter are the displacement data calculated from the Hall-effect sensors' outputs, ***v*** is a 6×1 vector and is the noise from the system.

$$\tilde{y}(t) = \begin{Bmatrix} x & y & \phi & z & \theta & \psi \end{Bmatrix}^T \tag{4.40}$$

The relationships between position, velocity, and acceleration are modeled exactly without any modeling error. Note that the process noise ***w*** is a 6×1 vector which has the same unit as the jerk and does not depend on the errors due to system. The constant-acceleration assumption is merely for a simple modeling purpose. Although this may not be a valid assumption, the output shows that the noise is reduced and the accuracy is improved. In discrete time, the model becomes the following,

$$\begin{aligned} x_{k+1} &= \begin{bmatrix} I_{6\times 6} & (t_{k+1}-t_k)I_{6\times 6} & \frac{1}{2}(t_{k+1}-t_k)^2 I_{6\times 6} \\ 0 & I_{6\times 6} & (t_{k+1}-t_k)I_{6\times 6} \\ 0 & 0 & I_{6\times 6} \end{bmatrix} x_k + \begin{bmatrix} \frac{1}{6}(t_{k+1}-t_k)^3 I_{6\times 6} \\ \frac{1}{2}(t_{k+1}-t_k)^2 I_{6\times 6} \\ (t_{k+1}-t_k)I_{6\times 6} \end{bmatrix} w_k \\ &= \Phi x_k + \Upsilon w_k \end{aligned} \tag{4.41}$$

$$\tilde{y}_k = \begin{bmatrix} I_{6\times 6} & 0 & 0 \end{bmatrix} x_k + v_k = Hx_k + v_k, \tag{4.42}$$

where, $(t_{k+1}-t_k)$ is the integration step size, which is constant for simulation and real-time control.

The estimator structure for the hybrid filter in [20] is of the form,

$$\hat{x}^+ = \hat{x}^- + K[\tilde{y} - H\hat{x}^-], \tag{4.43}$$

where K is the hybrid filter gain. This hybrid filter uses a weighted combination of the steady-state Kalman and the steady-state minimax filter gains in the estimator, so the hybrid filter gain is defined as

$$K = d\cdot K_{ss}^{k} + (1-d)\cdot K_{ss}^{\infty}, \tag{4.44}$$

where d is the relative weight given to K_{ss}^{k} which is the steady-state Kalman filter gain, and K_{ss}^{∞} is the steady-state minimax filter gain. This weight ranges from 0 to 1 and must be chosen so as to ensure stability since a convex combination of two stable estimators is not necessarily stable. The DKF formulation is presented in Table 4-1, as given in [21].

Table 4-1 Recursive Discrete-time Kalman Filter

MODEL	$x_{k+1} = \Phi x_k + \Upsilon w_k,\ \ w_k \sim N(0, Q_k)$ $\tilde{y}_k = Hx_k + v_k,\ \ v_k \sim N(0, R_k)$
INITIALIZE	$\hat{x}(t_0) = \hat{x}_0,\ \ P_0 = E\left\{\hat{x}(t_0)\hat{x}(t_0)^T\right\}$
GAIN	$K_k = P_k^- H^T \left[HP_k H^T + R_k\right]^{-1}$
UPDATE	$\hat{x}_k^+ = \hat{x}_k^- + K_k\left[\tilde{y}_k - H\hat{x}_k^-\right]$ $P_k^+ = \left[I - K_k H\right]P_k^-$
PROPAGATION	$\hat{x}_{k+1}^- = \Phi \hat{x}_k^+$ $P_{k+1}^- = \Phi_k P_k^+ \Phi_k^T + \Upsilon_k Q_k \Upsilon_k^T$

The vector ***v***, is the noise from the system, it is assumed to be zero-mean Gaussian noise, $v(t) \sim N(0, R_k)$, where R_k is the error covariance matrix of the system noise ***v***, which is defined in [21] as

$$R_k = (H_k^T W H_k)^{-1} = \text{diag}\left([1e-6 \quad 1e-6 \quad 1e-6 \quad 1e-6 \quad 1e-6 \quad 1e-6]\right) \tag{4.45}$$

Further, ***w*** is assumed to be a zero-mean Gaussian process, $w(t) \sim N(0, Q_k)$, where Q_k is the error covariance matrix of the process noise ***w***, which acts on the acceleration. Q_k is chosen to be a positive constant diagonal matrix,

$$\begin{aligned} Q_k &= \text{diag}\left([q_x \quad q_y \quad q_\theta \quad q_z \quad q_\theta \quad q_\psi]\right) \\ &= \text{diag}\left([1e3 \quad 1e3 \quad 1e5 \quad 1e2 \quad 1e2 \quad 1e2]\right), \end{aligned} \tag{4.46}$$

where an assumption is made that ***v*** and ***w*** are uncorrelated, and the values of R_k and Q_k were chosen by the designer. Selecting the bounds of the process and system noises became the parameters to tune the DKF. The values of the matrices R_k and Q_k were chosen through an ad-hoc approach, where we iterate the design until the best performance is achieved. The initial estimates are set to

$$\hat{x}_0 = \begin{Bmatrix} \hat{z}_0 \\ 0_{12\times 1} \end{Bmatrix}, \ \hat{z}_0 = \{0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0\}^T \tag{4.47}$$

where $\hat{z}_0$ is the same as the initial guess used for the initial conditions. The initial estimation error covariance matrix is specified by

$$P_0 = pI_{18\times 18}. \tag{4.48}$$

The initial value of p is set by the designer to obtain a good convergence behavior of the filter. The DKF updates the P matrix at each time step, and hence the P matrix will converge if good initial values are chosen. For the parameters of Q_k and R_k described previously, the steady-state Kalman filter gain matrix, K_{ss}^k, turned out to be,

$$K_{ss}^k = \begin{bmatrix} 0.0531 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.0531 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.1566 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.0364 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.0364 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.0364 \\ 1.390 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1.390 & 0 & 0 & 0 & 0 \\ 0 & 0 & 9.47 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.752 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.752 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.752 \\ 37.42 & 0 & 0 & 0 & 0 & 0 \\ 0 & 37.42 & 0 & 0 & 0 & 0 \\ 0 & 0 & 351.3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 12.05 & 0 & 0 \\ 0 & 0 & 0 & 0 & 12.05 & 0 \\ 0 & 0 & 0 & 0 & 0 & 12.05 \end{bmatrix}. \tag{4.49}$$

Using the same linear dynamic system as described above, the minimax filtering problem aims to minimize the maximum singular value of the transfer function from the noise to the estimation error [22].

$$J = \frac{avg\left\|x_k - \hat{x}_k\right\|_Q}{avg\left\|w_k\right\|_W + avg\left\|v_k\right\|_V}, \tag{4.50}$$

where *J* is a measure of how good our estimator is. We want to find a state estimate that will minimize the worst possible effect that ***w*** and ***v*** have on our estimation error, where the averages are taken over all time samples *k.* Several minimax filtering formulations have been proposed in $[23-24]$, the one considered here is the following: Find a filter gain K_∞ such that the maximum singular value is less than γ , or $J < {}^{1}\!/_{\gamma}$.

This is a way of minimizing the worst-case estimation error, where γ is the constant which becomes the design parameter to tune this filter. We can find a state estimate so that the maximum value of *J* is always less than $\hat{x}$, regardless of the values of the noise terms ***w*** and ***v***. This problem will have a solution for some values of γ but not for the values which are too small because the *P* matrix will become singular, as seen in the minimax gain equation below. The minimax filtering solution that forces $J < 1/\gamma$ is given in Table 4-2 below.

Table 4-2 Recursive Discrete-time Minimax Filter [20, 22]

MODEL	$x_{k+1} = \Phi x_k + \Upsilon w_k$ $\tilde{y}_k = Hx_k + v_k$
INITIALIZE	$\hat{x}(t_0) = \hat{x}_0, \quad P_0 = E\left\{\hat{x}(t_0)\hat{x}(t_0)^T\right\}$
GAIN	$K_\infty = \left[I + P_\infty^- / \gamma^2\right]^{-1} \; P_\infty^- H^T$
UPDATE	$\hat{x}_k^+ = \hat{x}_k^- + K\left[\tilde{y}_k - H\hat{x}_k^-\right]$ $\left(P_\infty^+\right)^{-1} = M^{-1} - I/\gamma^2 + H^T H$
PROPAGATION	$M = \Phi P_\infty^+ \Phi^T \; + \Upsilon I \Upsilon^T$

Notice that we do not have to worry about defining the matrices for ***w*** and ***v*** noises using the minimax filter, which makes this the main advantage of such filter. For $\gamma = 10$, the steady-state minimax filter gain matrix, K_{ss}^{∞}, turned out to be,

$$K_{ss}^{\infty} = \begin{bmatrix} 0.637 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.637 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.637 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.637 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.637 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.637 \\ 0.0135 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.0135 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.0135 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.0135 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.0135 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.0135 \\ -0.8e-4 & 0 & 0 & 0 & 0 & 0 \\ 0 & -0.8e-4 & 0 & 0 & 0 & 0 \\ 0 & 0 & -0.8e-4 & 0 & 0 & 0 \\ 0 & 0 & 0 & -0.8e-4 & 0 & 0 \\ 0 & 0 & 0 & 0 & -0.8e-4 & 0 \\ 0 & 0 & 0 & 0 & 0 & -0.8e-4 \end{bmatrix}. \tag{4.51}$$

CHAPTER V

ANALYSIS AND OPTIMIZATION OF SIGNALS

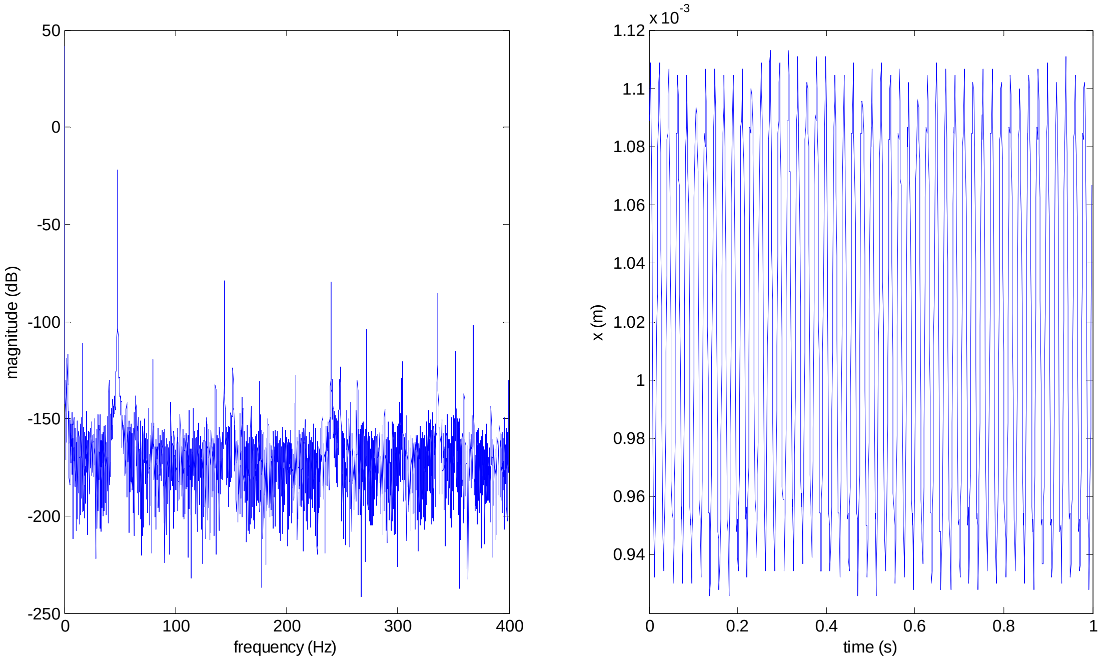
This chapter consists on the analysis of the signals coming from the Hall-effect sensors and the design of digital filters to attenuate the noise and optimize the signal response. It was discovered that much of the noise was static noise, so second-order digital filters were designed in addition to the observer described in the previous chapter, all of this in order to remove the static noise and optimize the signal response.

5.1 Hall-effect Sensor Signal Analysis

In order to analyze the noise of the Hall-effect sensors a Fast Fourier transform (FFT) is conducted for the planar MIMO system under the control law described above. For these experiments, the controller makes the platen move to a specified coordinate in the x-direction with the controller doing its best to stay put. For the first test the LQ control law is applied without the hybrid filter being active. The results are displayed in Figure 5-1. From the FFT one can see that there is a peak at 50 Hz of about -30 dB in magnitude. The response of the signal for one second is also included, and the resolution is about 150 μm peak to peak.

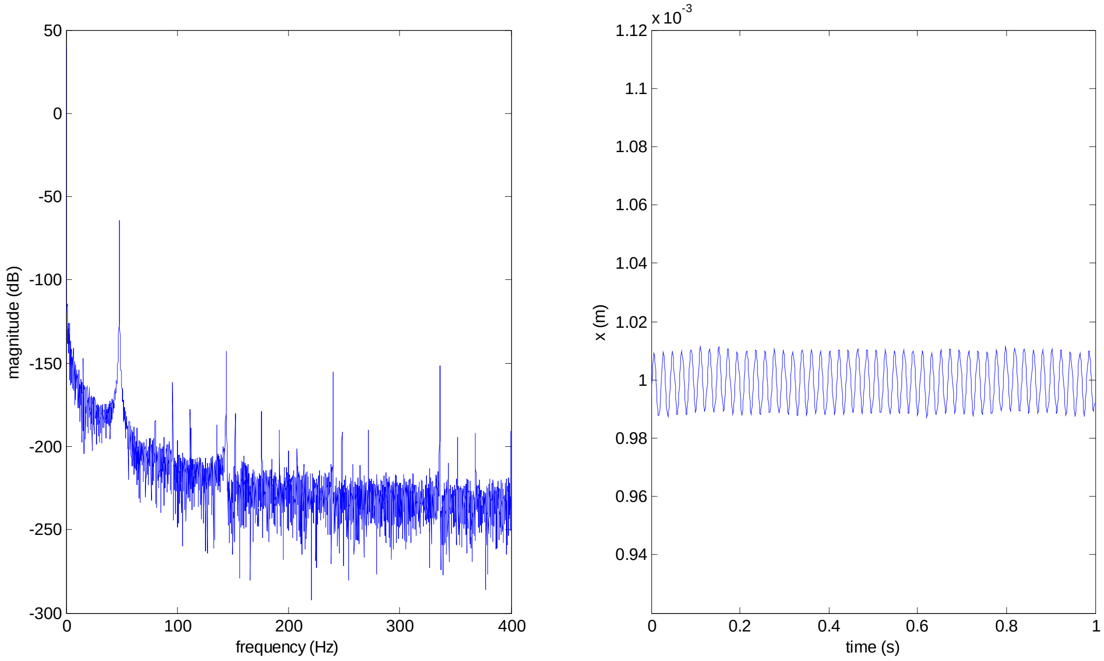
The next test applies the hybrid filter, which would make this the LQG controller. The results are displayed in Figure 5-2. The resolution of the signal has been reduced to 20 μm peak to peak. From the FFT, the 50 Hz peak has been reduced to -60 dB in magnitude. For the experiments involving the hybrid filter being active, the relative weight of the hybrid filter, d, is one so the hybrid filter acts as a DKF.

The nature of the systematic 50-Hz noise was found to be purely electric, coming from the Hall-effect sensors. The Hall-effect sensors are manufactured by a company that’s situated in Europe, where the standard frequency of the electricity coming from the electrical outlet is 50 Hz [10]. Oscilloscope measurements confirmed that the 50-Hz peak noise comes from the Hall-effect sensors.



(a) (b)

Figure 5-1: (a) FFT of the Hall-effect sensor signal and (b) time response



(a) (b)

Figure 5-2: (a) FFT of the signal with Hybrid filtering and (b) time response

From this analysis, it can be concluded that the peak noise at 50 Hz, coming from the Hall-effect sensors, is a biased non-random noise because the FFT of the signal after the implementation of the DKF shows a noise attenuation of barely 30 dB from the signal without any filtering. This is because one of the main conditions for the implementation of the DKF is that the noise has to have a white Gaussian distribution.

Even though the DKF yields acceptable results, there is certainly more than can be done to attenuate the noise even further. A second-order digital low-pass filter will start attenuating all of the noise after the cut-off frequency, regardless of whether the noise is biased or random. This will lead to more precise readings from the Hall-effect sensors. The design of the implemented second-order digital filters is described below.

5.2 Second-order Digital Low-pass Filter Design

The digital filter will essentially act as a 'pre-filter,' because the data being read from the sensor will be filtered first by the digital second-order low-pass filter, and then will be further filtered by the observer of the LQG controller. The results from the analysis done in the previous section shows that the biased noise has a frequency of about 50 Hz, the digital filter is designed with a cut-off frequency of 8 Hz, since the cut-off frequency has to be at least half the frequency to attenuate . The digital second-order low pass filter is of the form as in [25]:

$$H(z)=\frac{1}{k}\left(\frac{z^{-1}}{1-2r\cos(\theta)z^{-1}+r^{2}z^{-2}}\right), \tag{5.1}$$

where θ and r are defined as,

$$\theta=2\pi\frac{f_c}{f_s},\quad r=1-2\zeta\theta, \tag{5.2}$$

where f_c is the cut-off frequency, f_s is the sampling frequency, ζ is the damping coefficient of the filter and k is the constant to normalize the data. The last step in the filter design is to normalize the signal, according to the cut-off frequency. The way we find it is that we set the values of *z* to be one and then we just solve for *k,* such that the equation becomes

$$\frac{1}{1-2r\cos(\theta)+r^2}=k, \tag{5.3}$$

where *k,* is the gain that the input signal needs to be multiplied by in order to normalize the output signal. The final form of the second-order filter is,

$$H(z)=\frac{1}{205.22}\left(\frac{z^{-1}}{1-1.8735z^{-1}+0.8783z^{-2}}\right). \tag{5.4}$$

The Bode plot for this digital filter is displayed in Figure 5-3, as well as the continuous form of the filter. The continuous form of the filter was designed using the ‘d2c’ command in matlab, which transforms a digital transfer function into a continuous transfer function.

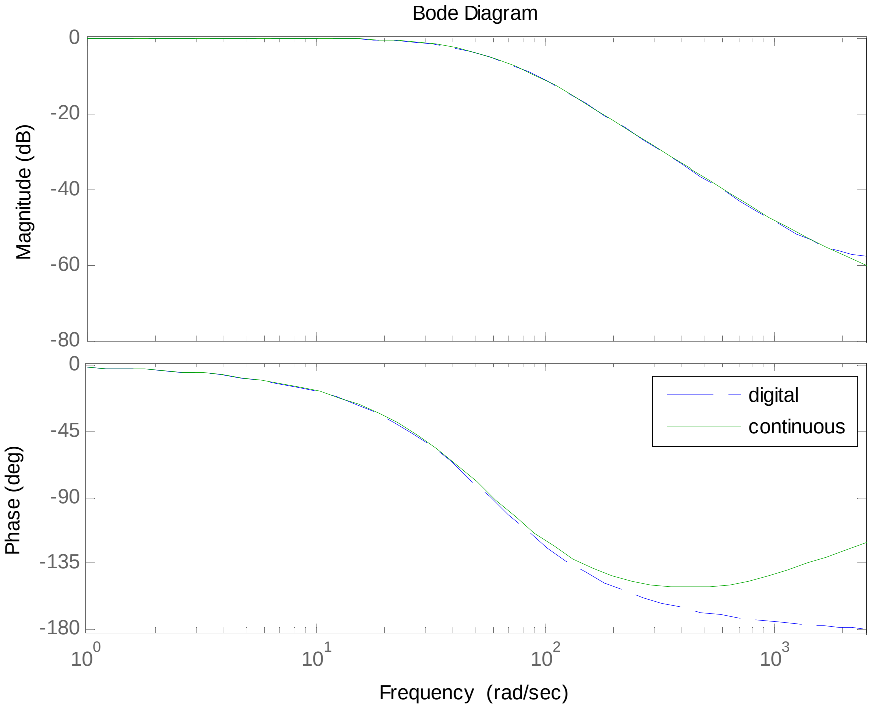
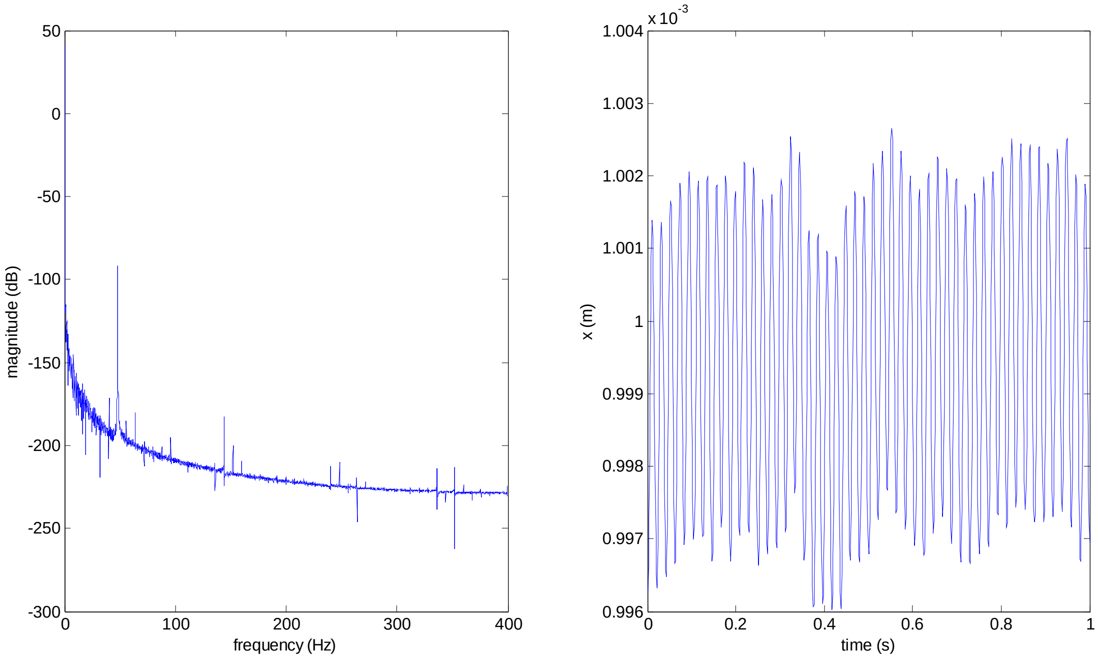


Figure 5-3: Bode plot of the digital and the continuous low-pass filter

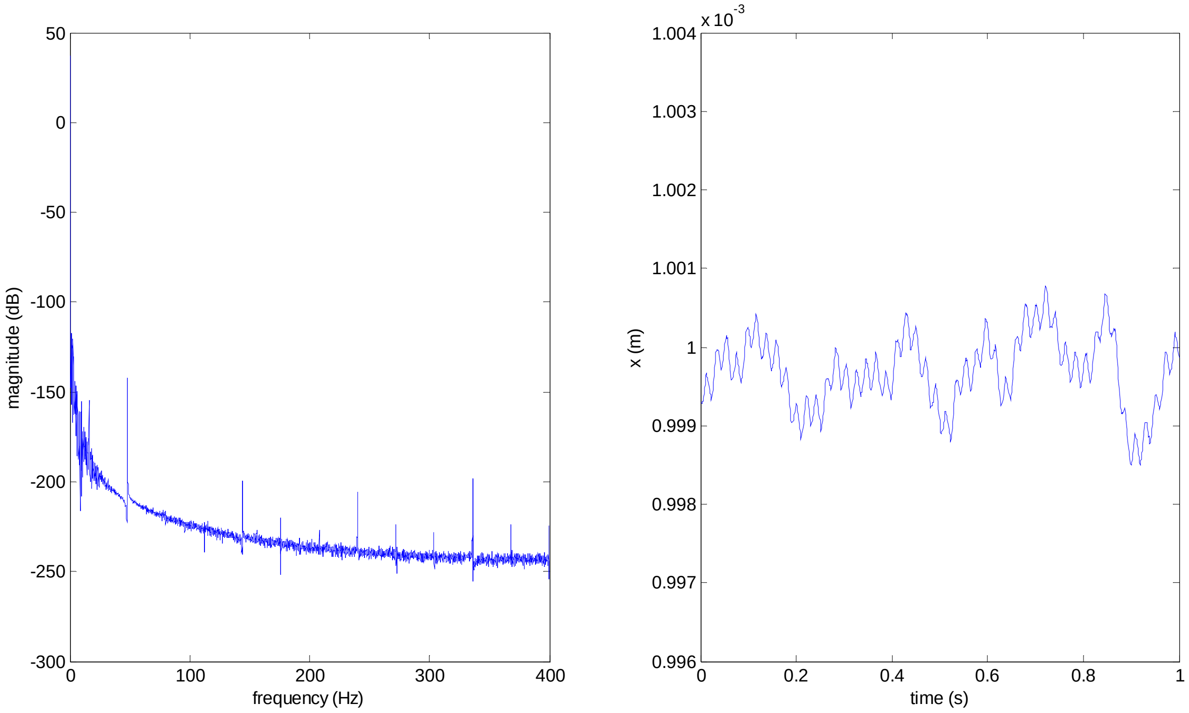
Now the aforementioned digital low-pass filter is implemented, without the hybrid filter making this an LQ controller, because the low-pass filter does not estimate the velocities of the states. An FFT is performed on the signal response and the signal response is plotted for a second, and the results are displayed in Figure 5-4. The resolution of the signal has been enhanced to less than 7 μm peak to peak, and the peak at 50 Hz has been reduced to about -90 dB in magnitude from the FFT.



(a) (b)

Figure 5-4: (a) FFT of the signal with low-pass filtering without hybrid filtering and (b) time response

Finally both filters are applied together, the digital low-pass filter acting as a "pre-filter" while the DKF acts as the observer for the LQG controller. The results are displayed below in Figure 5-5, it can be seen that the resolution has been enhanced to almost 2 μm peak to peak and from the FFT, the peak at 50 Hz has been reduced to almost 0-dB in magnitude, thus proving the effectiveness of having the digital second-order low-pass filter act as a "pre-filter" on the incoming signal and then have the DKF further reduce the noise of the signal.



(a) (b)

Figure 5-5: (a) FFT of the signal with low-pass filtering combined with hybrid filtering and (b) time response

5.3 Second-order Digital Notch Filter Design

As seen in the previous section the second-order digital low-pass filter design is able to attenuate some of the 50 Hz biased noise, but the level of attenuation depends on how low the cut-off frequency is set. However, the cutoff frequency cannot be set too low because instability becomes an issue for the system; in fact the low-pass digital filter design from the previous section is done at the lowest possible cut-off frequency. The limitation of the cut-off frequency in the low-pass digital filter design also limits the level of attenuation for the 50 Hz peak noise.

This limitation led me to design a second-order notch digital filter for which the notch is set at 50 Hz, doing this will attenuate the peak noise more than the previous design and might lead to more precise results than those in the previous section. Again this digital notch filter will essentially act as a pre-filter, since the data being read from the sensor will be filtered first by the digital second-order notch filter and then it will be further filtered by the observer of the LQG control. The second-order notch filter is of the form [25]:

$$H(s) = \frac{ks^2 + \omega_0^2}{s^2 + 2\zeta\omega_0 s + \omega_0^2}, \tag{5.5}$$

where ζ is the damping ratio of the filter, *k* is a constant related to the attenuation at high frequencies and ω_0 is the constant for which we are solving for. Now we plug in the fact that $s = j\omega$, to get,

$$H(j\omega) = \frac{-k\omega^2 + \omega_0^2}{-\omega^2 + 2\zeta\omega_0(j\omega) + \omega_0^2}. \tag{5.6}$$

In order to get a notch, the numerator of the transfer function has to be zero at a fixed frequency, ω_{notch}, which would make the magnitude of the transfer function go to zero at such frequency. The equation for the numerator of the transfer function is,

$$-k\omega_{notch}^2 + \omega_0^2 = 0, \tag{5.7}$$

where we solve for ω_0 since ω_{notch} becomes the frequency in which we want the notch to occur, solving for ω_0 we get the equation,

$$\omega_0 = \omega_{notch}\sqrt{k}. \tag{5.8}$$

As it was shown in the previous sections, the biased peak noise has a frequency of 50 Hz, so the notch frequency will be set at 50 Hz. However we still have not solved for *k,* which relates to the attenuation of the signal at the frequencies after the notch occurs. Due to the harmonics from the 50-Hz biased noise at higher frequencies, some attenuation is necessary at the higher frequencies. A $-20\,\text{dB}$ attenuation is desired at high frequencies, which means that

$$-20\text{dB} = 20\log(k), \tag{5.9}$$

therefore k = 0.1. The attenuation cannot be set too low because the stability of the system becomes an issue, however the signal at high frequencies could even be amplified if desired, but in this case noise attenuation is needed. Substituting k = 0.1 and the notch frequency into Eq. (5.8) gets,

$$\omega_0 = 2\pi \cdot 50\sqrt{0.1} \tag{5.10}$$

Notice that the notch frequency needs to be multiplied by 2π in order to get the frequency in radians per second, not Hertz. Substituting the value of ω_0 into Eq. (5.5) gives us the final form of the second-order analog notch filter,

$$H(s) = \frac{0.1s^2 + (100\pi\sqrt{0.1})^2}{s^2 + (2)(100\pi\sqrt{0.1})s + (100\pi\sqrt{0.1})^2}. \tag{5.11}$$

The Bode plot for this digital filter is displayed in Figure 5-6, as well as the continuous form of the filter. The digital form of the filter was designed using the ‘c2d’ command in Matlab, which transforms a continuous transfer function into a digital transfer function. However one needs to use an advanced transformation such as the ‘pre-warp’ command, which instructs matlab to transform the continuous transfer function using the pre-warping technique around the critical frequency of 50 Hz.

$$H(z) = \frac{0.09318 - 0.1733z^{-1} + 0.09318z^{-2}}{1 - 1.792z^{-1} + 0.8047z^{-2}}. \tag{5.12}$$

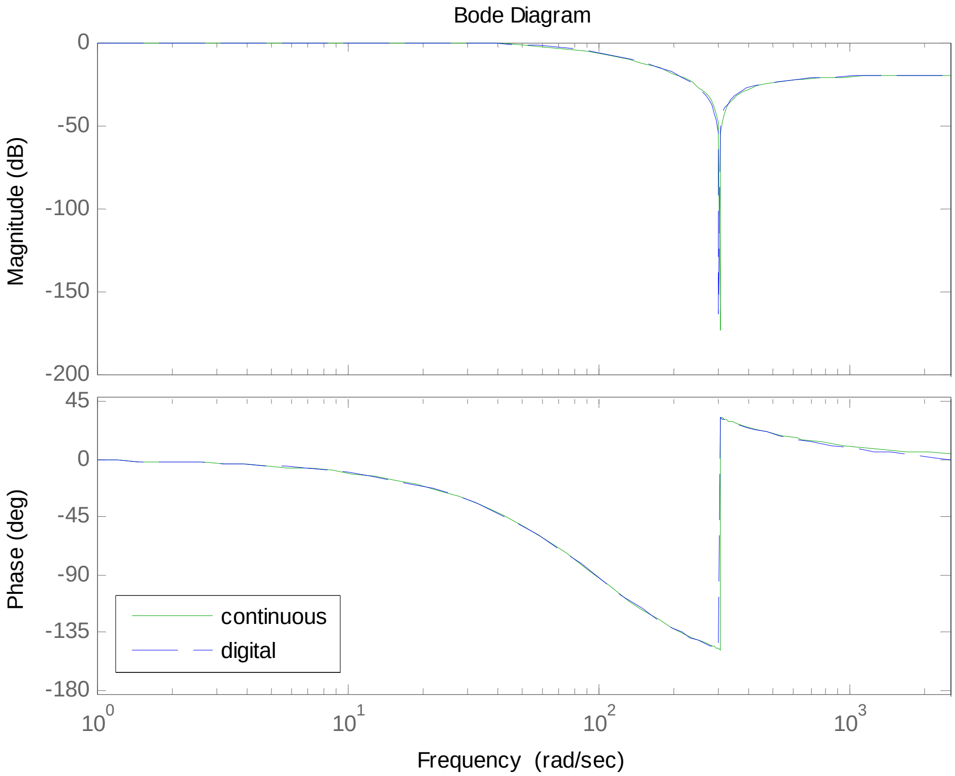


Figure 5-6: Bode plot of the digital and the continuous notch filter

Now the aforementioned digital notch filter is implemented without the DKF making this an LQ controller, since the notch filter does not estimate the velocities of the states. An FFT is performed on the signal response and the signal response is plotted for a second, the results are displayed in Figure 5-7, it can be seen that the resolution of the signal has been enhanced to about 4 µm peak to peak and from the FFT, one can see that the peak at 50 Hz has been completely eliminated.

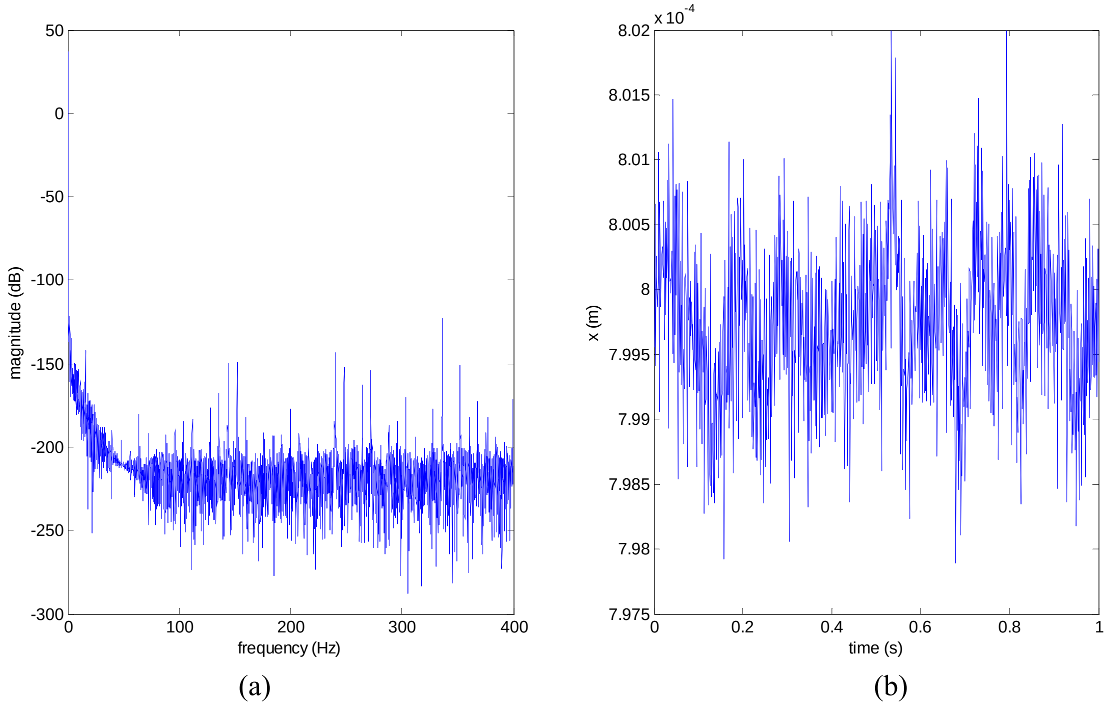


Figure 5-7: (a) FFT of the signal with notch filtering without hybrid filtering and (b) time response

Finally both filters are applied together, the digital notch filter acting as a pre-filter while the hybrid filter acts as the observer for the LQG controller. The results are displayed below in Figure 5-8, it can be seen that the resolution has been enhanced to 1.5 µm peak to peak and from the FFT, the peak at 50 Hz has been eliminated, thus proving the efficiency of having the digital second-order notch filter act as a pre-filter on the incoming signal and then have the hybrid filter further reduce the noise of the signal.

The digital notch filter combined with the hybrid filter not only eliminates the 50 Hz noise but it also provides the best results with optimal position resolution for the planar motion. That is why for Chapter VI, all of the experimental results are done with the digital notch filter being active unless otherwise noted.

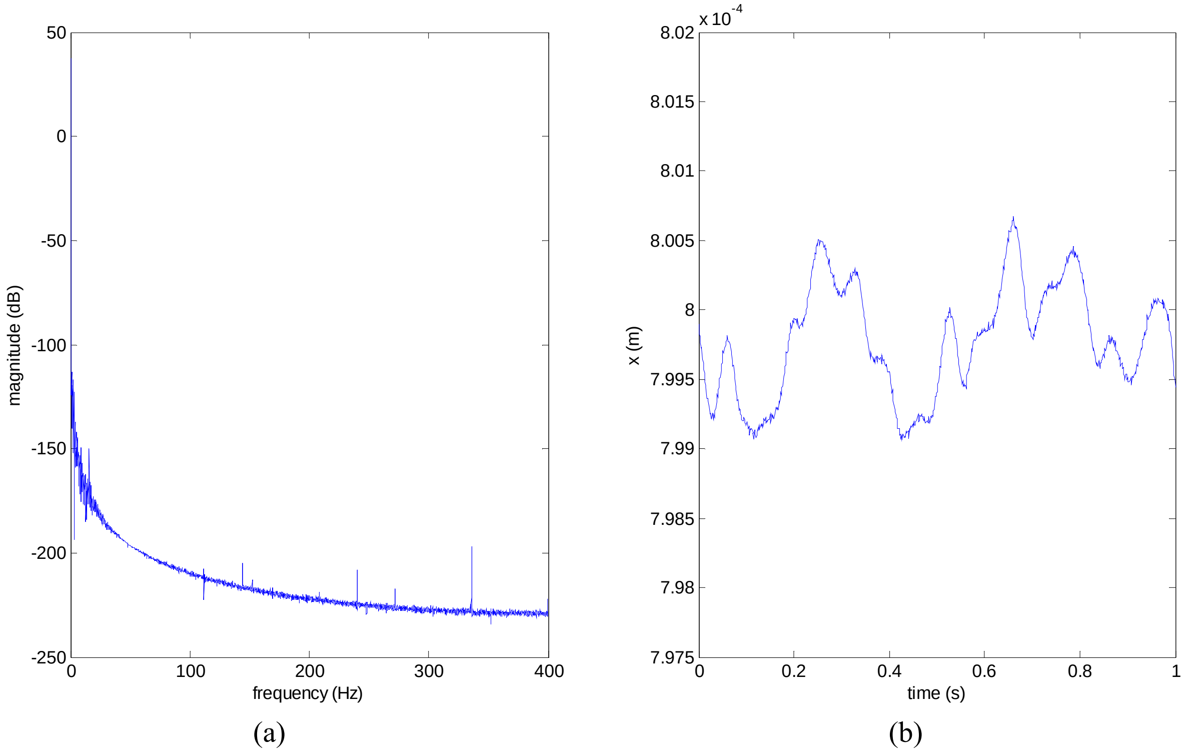


Figure 5-8: (a) FFT of the signal with notch filtering combined with hybrid filtering and (b) time response.

CHAPTER VI

EXPERIMENTAL RESULTS

This chapter illustrates the experimental results for all 6 DOFs of the multidimensional positioner. With the LQG controllers designed in the previous chapters, the dynamics performances of the positioner were tested. Many comparisons for different parameters of the hybrid filter and for the digital filter are shown to demonstrate the effect that each different parameter has on the signal response. For the purpose of demonstrating its precision-positioning capabilities, microscale positioning step responses were performed on all 6 axes. The results in this chapter present that the multidimensional positioner has the potential to generate the appropriate stepping motions required in precision positioning industries.

6.1 Step Responses

The LQG controller for the vertical mode was designed in Chapter IV. The control gain matrix was found in Eq. (4.37), Figures 6-1 through 6-3 present the step responses of the vertical mode with the LQG controller. Several step responses were taken in the *z*-direction and the rotations θ and ψ with step sizes of 5 μm and 30 μrad, respectively. The step response in *z* employs all three planar motors. Since the dynamic coupling was well compensated, there are no steady-state errors in each axis.

Step responses in the experimental results show that the rise time for the *z*-direction is less than 200 ms, the overshoot is about 16%, and the settling time is about 500 ms without steady-state errors. The position noise is 49.3 nm rms in *z*. The position noise is mainly caused by the Nanogage sensors. Besides it might be generated by the umbilical cables, three aerostatic bearing that blow air flow and the modeling error from dynamic analysis.

The step responses for the rotations in ψ and θ based on the LQG controller are presented in Figures 6-2 and 6-3, respectively.

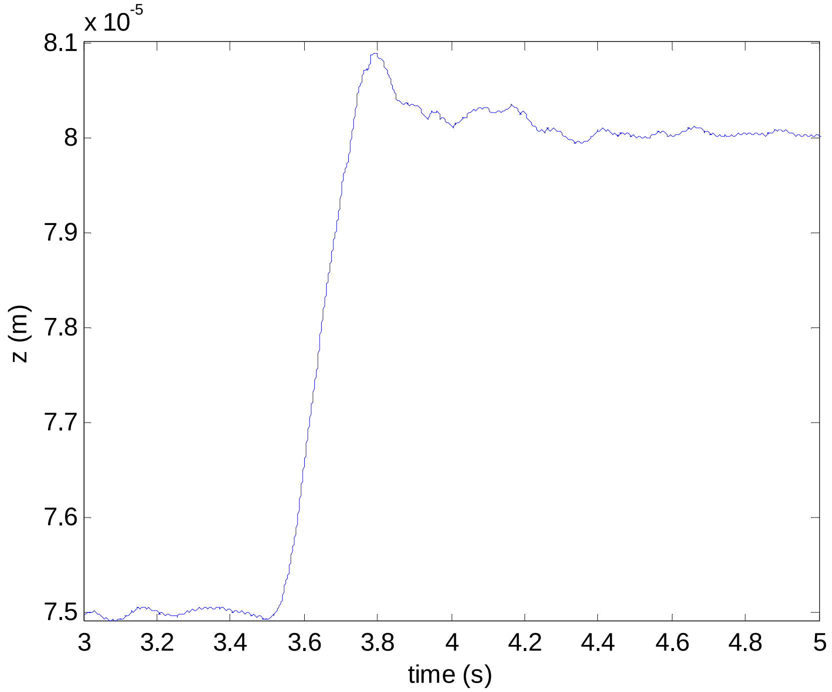


Figure 6-1: 5-µm step response in *z*

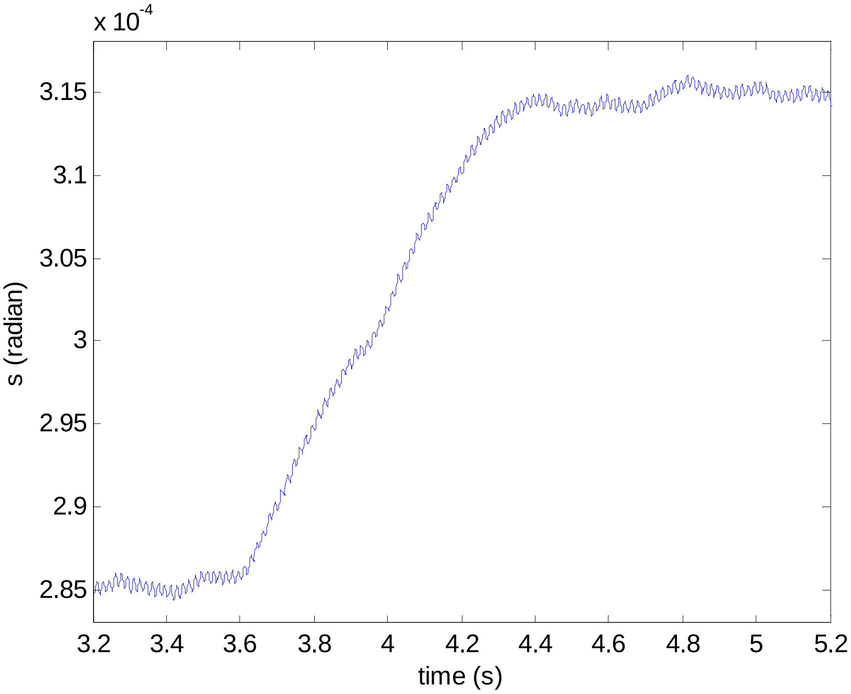


Figure 6-2: 30-µrad step response in ψ

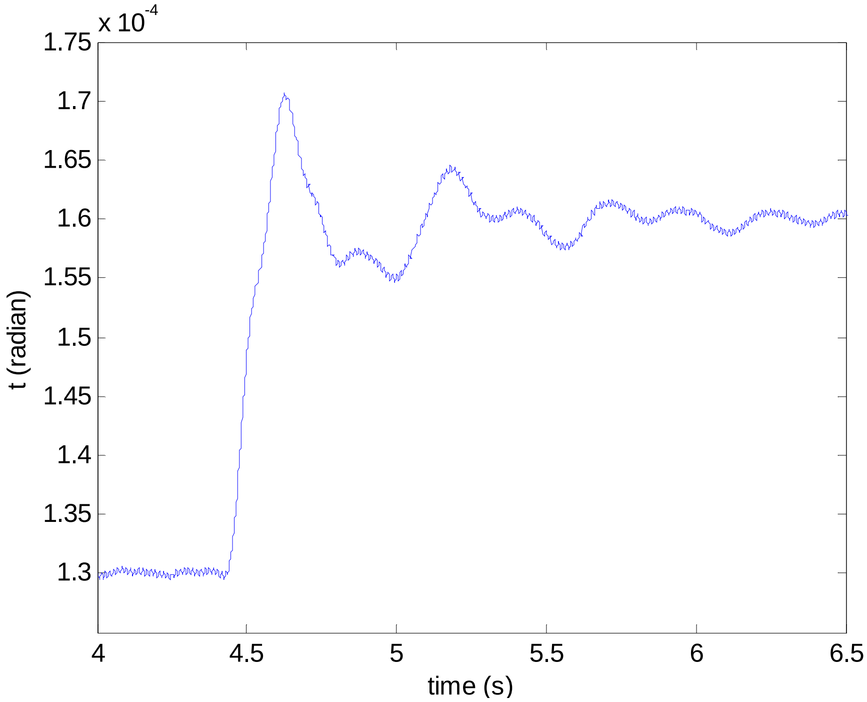


Figure 6-3: 30-µrad step response in θ

The LQG controller for the horizontal mode was designed in Chapter IV. The control gain matrix was found in Eq. (4.37), Figures 6-4 through 6-11 present the step responses of the horizontal mode with the LQG controller. Several step responses were taken in the *x*- and *y*-direction with the step sizes of 50 µm, 20 µm, and 10 µm. The step response in *x* employs the planar motor C. Two motors A and B compensate for the stray torque caused by the geometrical asymmetry. Since the dynamic coupling was well compensated, there are no steady-state errors in each axis.

Step responses in the *x*-direction based on the LQG controller are shown in the following pages. A 10-µm step response in *x* is presented in Figure 6-4, respectively. 20-µm step response in *x* is presented in Figure 6-5. The 50-µm step response in *x* is shown in Figures 6-6.

The 10-µm, 20-µm and 50-µm step responses in *y* are presented in Figures 6-7, 6-8 and 6-9, respectively. Step responses with the LQG compensator in the experimental results show that the rise time is less than 25 ms, the maximum overshoot is around 10%, and the settling time is less than 220 ms without any steady-state errors in the *x* and *y* directions.

These figures illustrate that the positioner can demonstrate the capability of microscale positioning. The position resolution is 1.5 noise is μm with a position error of 0.545 μm rms in *x* and *y*.

Step responses of 0.1-mrad and 6-mrad rotations around *z* based on the LQG controller are presented in Figures 6-10 and 6-11, respectively. The position noise is mainly caused by the Hall-effect sensors. Also the response of the system might be affected by the noise from the sensors, outside disturbances such as the umbilical cables and modeling errors from the dynamic analysis.

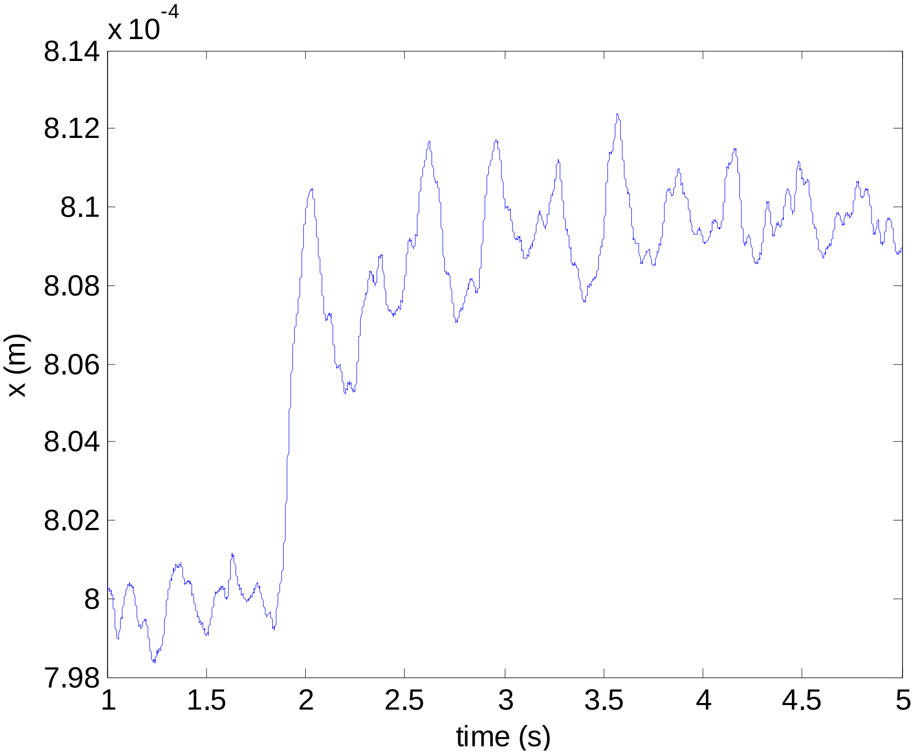


Figure 6-4: 10-μm step response in *x*

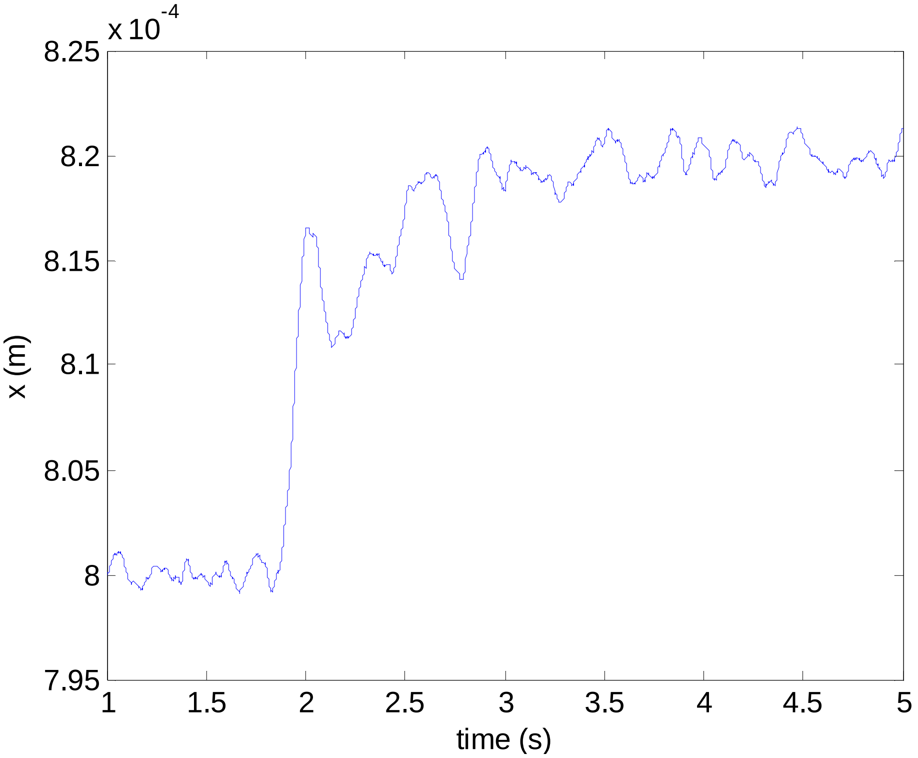


Figure 6-5: 20-µm step response in *x*

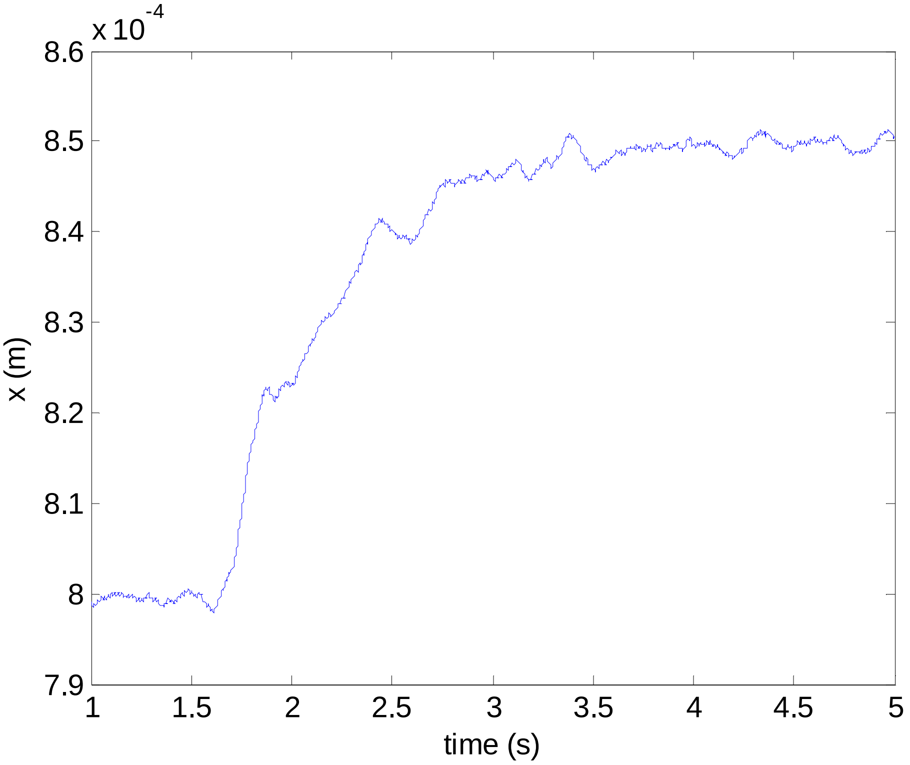


Figure 6-6: 50-µm step response in *x*

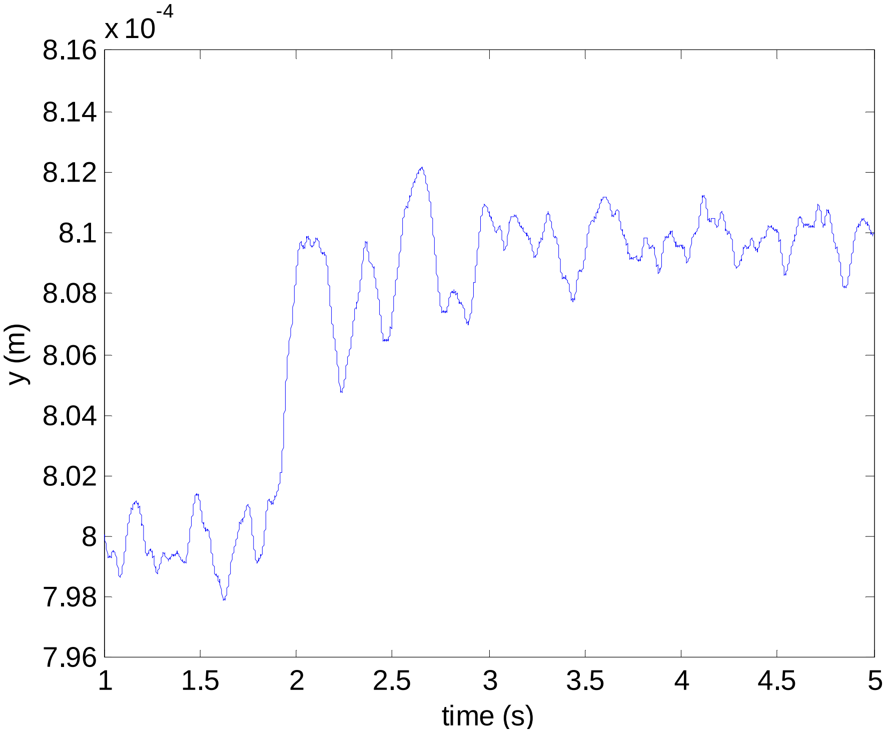


Figure 6-7: 10-µm step response in *y*

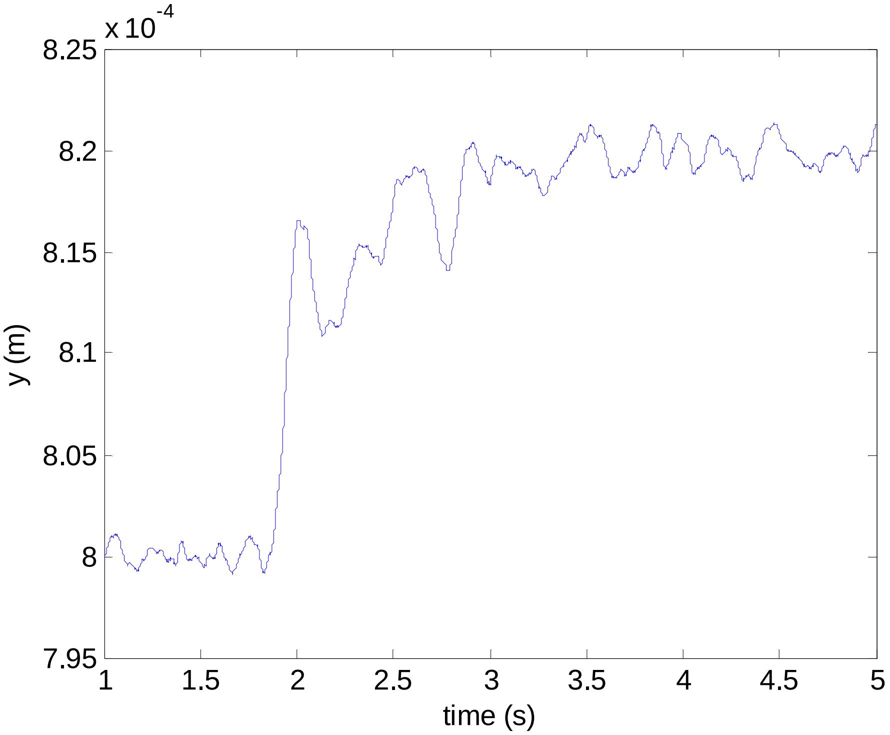


Figure 6-8: 20-µm step response in *y*

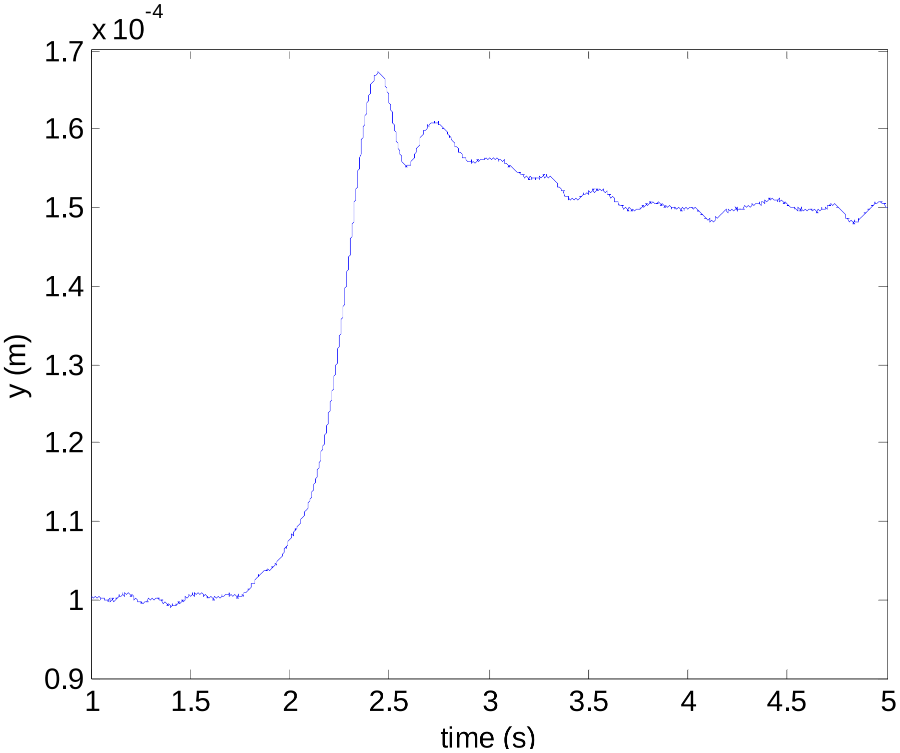


Figure 6-9: 50-µm step response in *y*

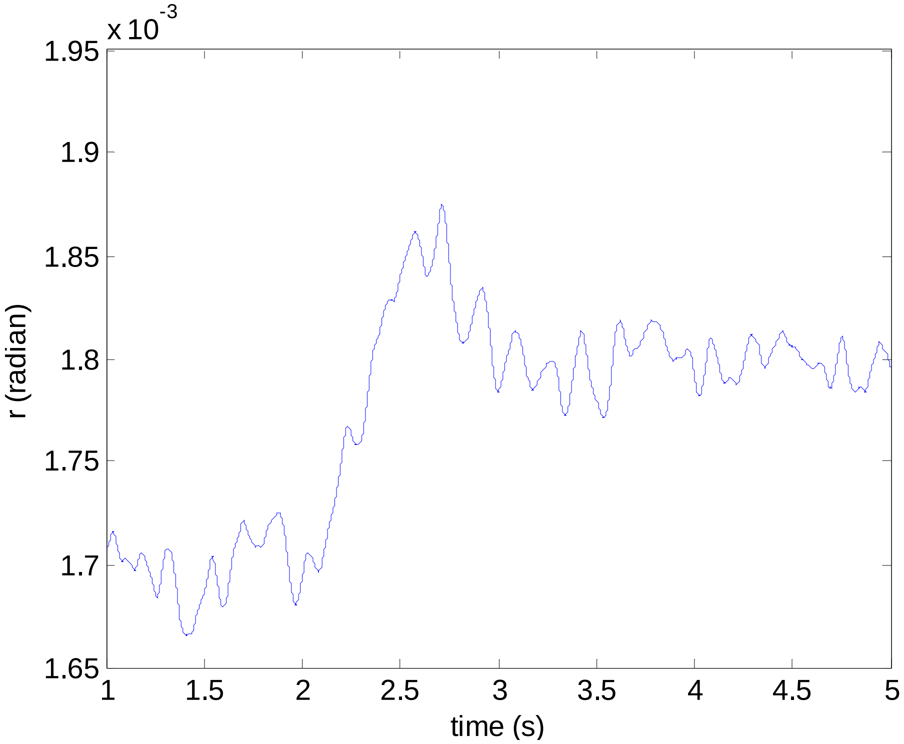


Figure 6-10: 0.1-mrad step response in ϕ

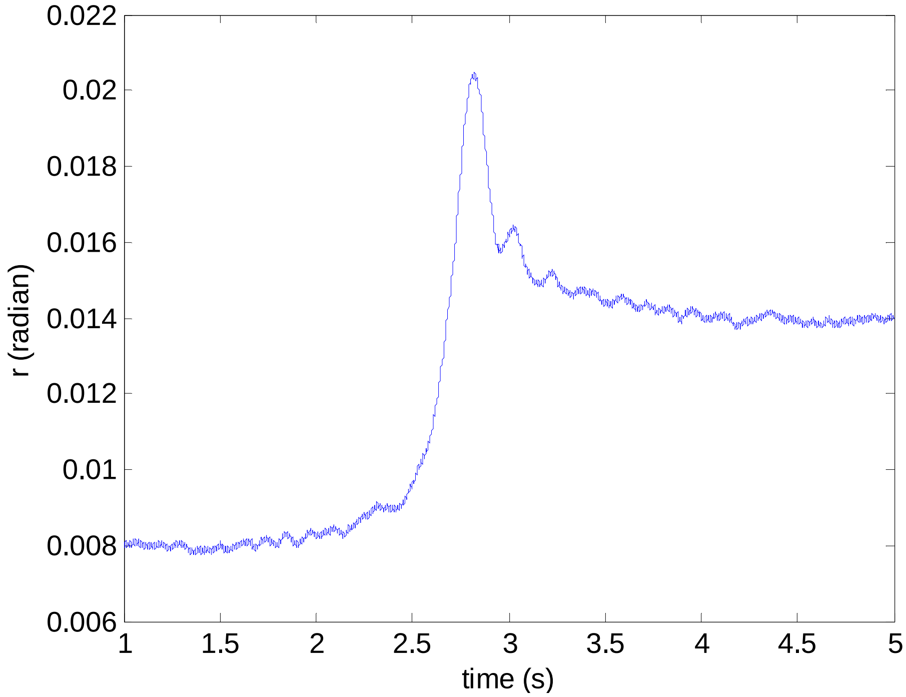


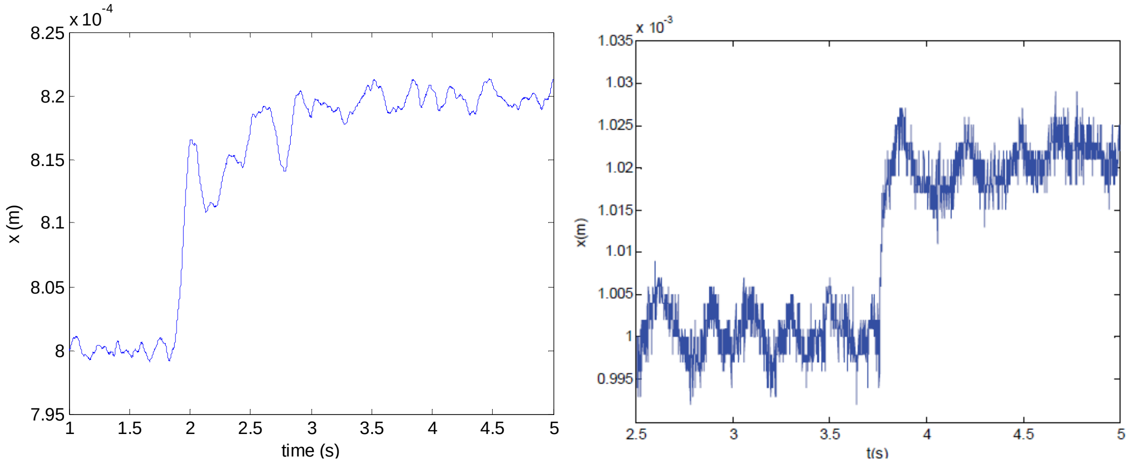
Figure 6-11: 6-mrad step response in ϕ

6.2 Comparisons with Previous Work

This section presents the comparisons between my results and the results from Yu, the designer of the positioner. The step responses used in the comparisons are taken from [5], his Ph.D. dissertation, in which he used lead-lag controllers to control each individual axis and a DKF to obtain better estimates in the planar mode. My results have the combined digital filtering in the planar mode and the DKF for the vertical mode, both of them use the LQG MIMO controllers. Figure 6-12 shows a comparison for a 20-µm step in *x,* while Figures 6-13 and 6-14 show a comparison for a 10-µm step in *y* and a 5-µm step in *z*, respectively.

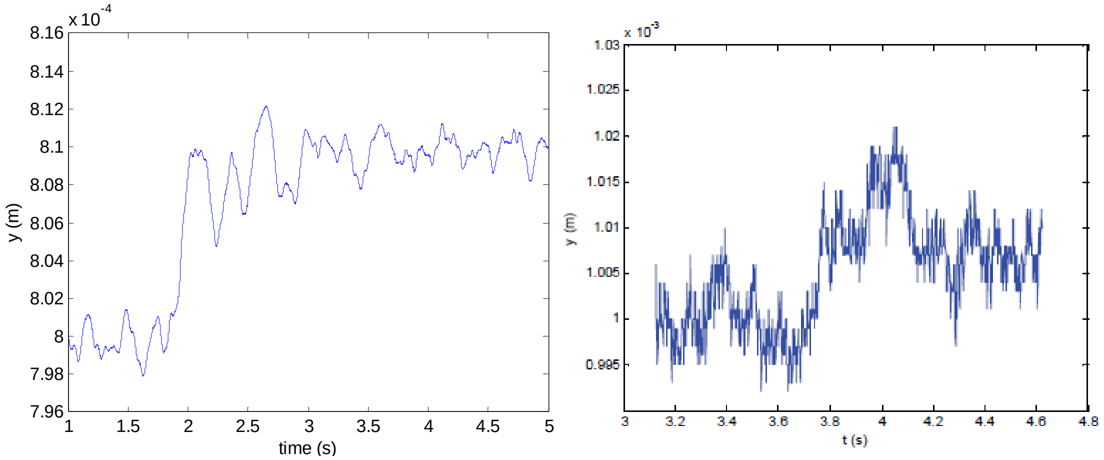
Yu's step responses demonstrated a 14-µm resolution with an 8-µm rms position noise in the planar mode while for the vertical mode, a 0.8-µm resolution with a 0.5-µm rms position noise. My results show a position noise 0.545-µm rms in *x* and *y* with a resolution of 1.5-µm, and for the vertical mode a position noise of 49.3-nm rms with a resolution of 110-nm.

It is clear that a better position resolution is obtained in my results for both the planar and the vertical modes. However it seems that my results have a slower rise time and settling time, but at the same time they have smaller overshoot magnitudes. The slower response time might be an effect from the combined filtering in the planar mode and the hybrid filter in the vertical mode, while the small overshoots is both an effect of the filtering and the LQG controllers.



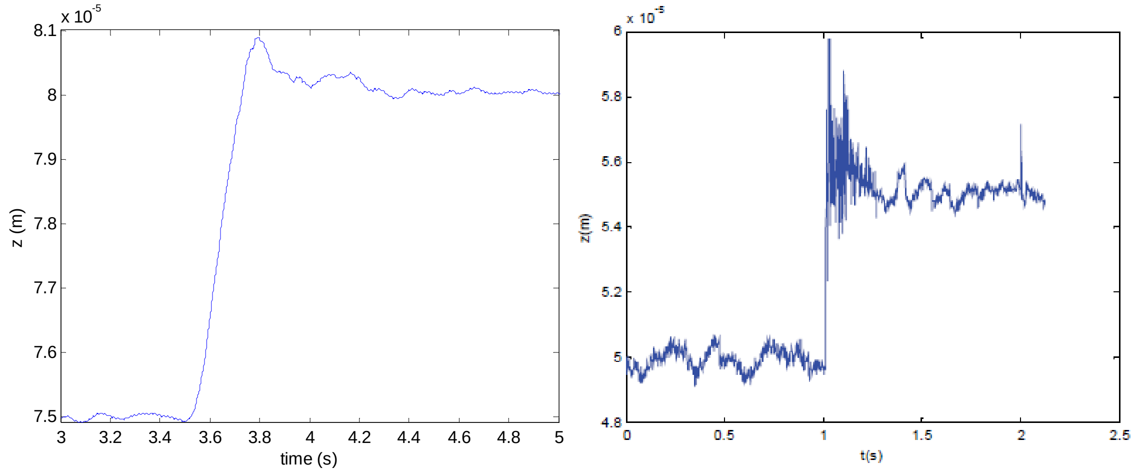
(a) (b)

Figure 6-12: 20-µm step response in *x* comparing (a) my results and (b) Yu's [5]



(a) (b)

Figure 6-13: 10-µm step response in *y* comparing (a) my results and (b) Yu's [5]



(a) (b)

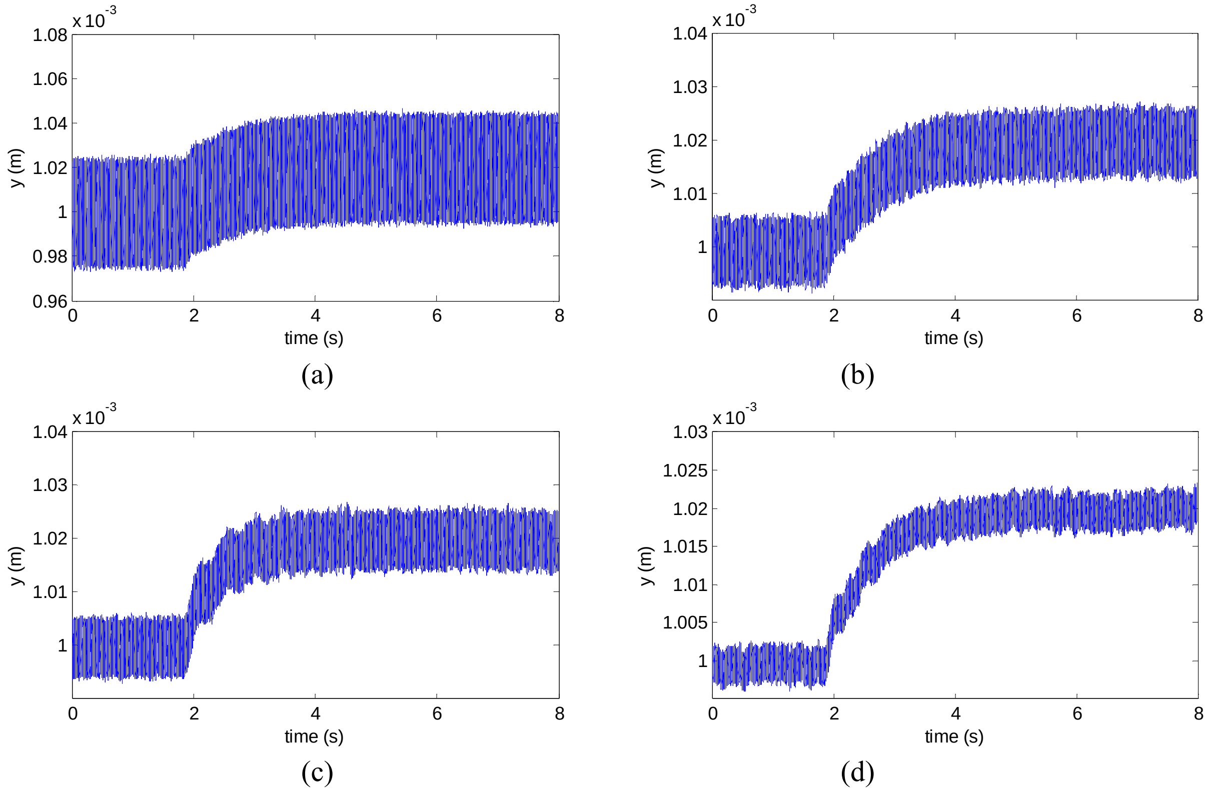
Figure 6-14: 5-μm step response in *z* comparing (a) my results and (b) Yu's [5]

Figure 6-15 uses the second-order low-pass digital filter in order the effect that changing cutoff frequencies of the low-pass filter has on the signal response and compared them, with each other, this is done without the recursive hybrid filter being active. As one would expect as the cutoff frequency decreases, the magnitude of the noise decreases, that's because the cutoff frequency is the point at which the filter starts attenuating all of the subsequent frequencies at a rate of -40 db/dec. The more that one decreases the cutoff frequency, the smaller the magnitude of the 50 Hz noise, which means the better the positioning resolution for the planar motion.

For the recursive Hybrid filter, different hybrid filter weights are used in order to see how it affects the signal response of the Hall-effect sensors and to see whether a certain value of the hybrid filter weight leads to a higher positioning resolution. The comparisons are done with and without the digital notch filter being active. The results are shown in Figures 6-16 and 6-17, all of the experiments are done while giving the system a step response in the *y*-direction.

Figure 6-16 show that without the digital notch filtering a combination of the hybrid filter gives better resolution than the DKF or the minimax filter. The optimal hybrid filter weight is somewhere around *d* = 0.8, for which a 16-μm peak to peak resolution was achieved, however when the notch digital filter is implemented along with the hybrid filter, the DKF provides the best optimal estimates for the position in the planar motion with a 1.5-μm peak to peak resolution.

Figure 6-17 show that the DKF provides better estimates when the digital notch filter is implemented because the notch filter eliminates the main source of the biased noise at 50 Hz and on top of that, it attenuates the harmonics of such noise at high frequencies. By eliminating the main source of the non-random noise, the DKF can provide optimal estimates because the main assumption for the DKF implementation is that the noise has to be random with a Gaussian distribution. Therefore whenever the digital notch filter is not implemented, the 50 Hz biased noise and its harmonics have a significant magnitude, this stops the DKF from being an optimal state estimator, which is the reason why the hybrid filter with a relative weight of 0.8 gives better estimates.



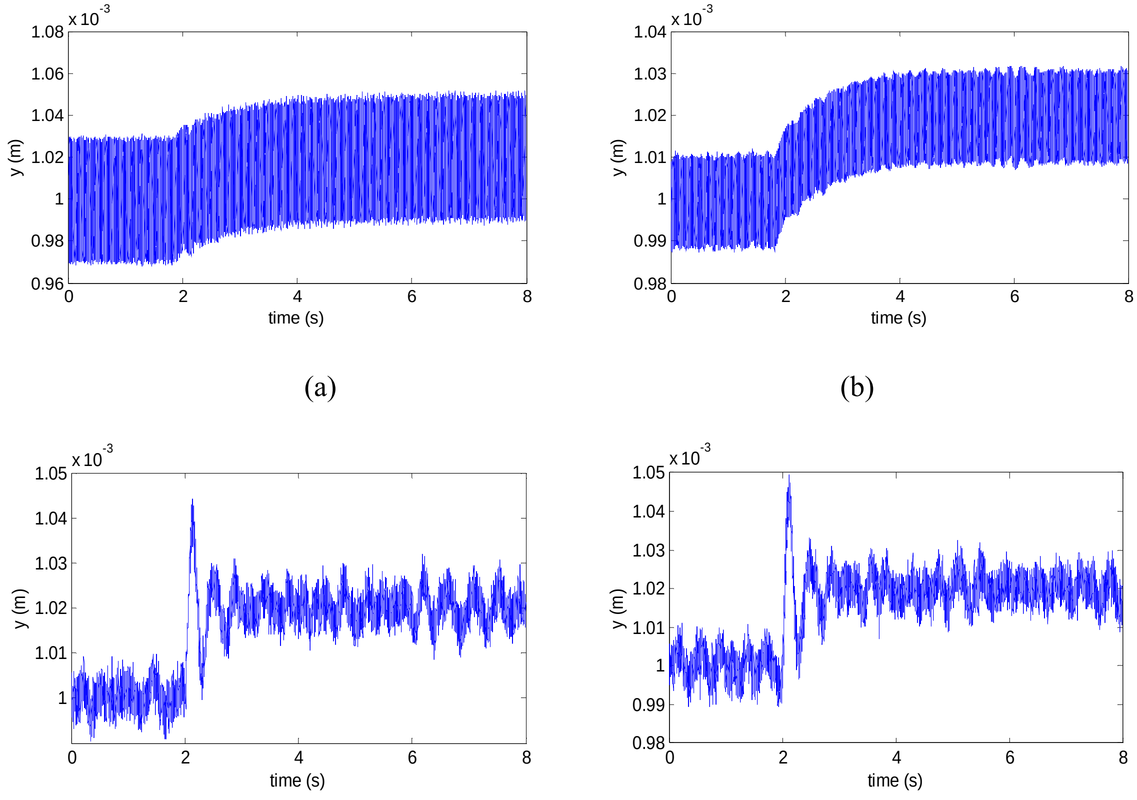
(a)

(b)

(c)

(d)

Figure 6-15: 20-μm step responses for low-pass filter with cut-off frequencies of (a) 20 Hz, (b) 14 Hz, (c) 10 Hz, and (d) 7 Hz without hybrid filtering



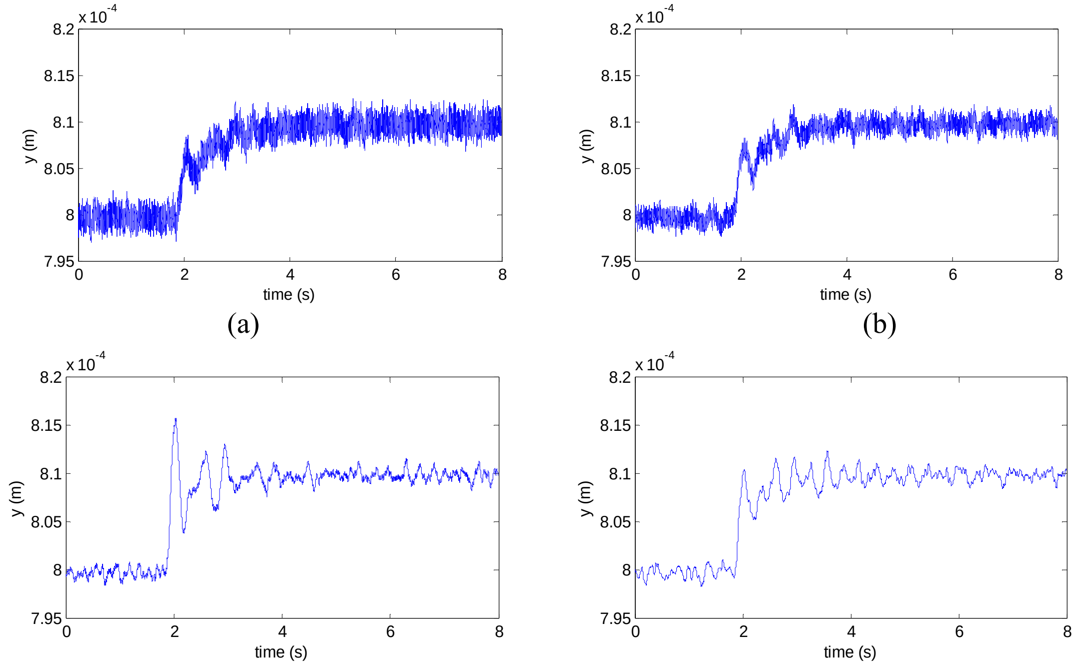
(a)

(b)

(c)

(d)

Figure 6-16: 20-μm step responses for hybrid filter weights of (a) *d* = 0, (b) *d* = 0.3, (c) *d* = 0.8, and (d) *d* = 1 without digital notch filtering



(a)

(b)

(c)

(d)

Figure 6-17: 10-μm step responses for hybrid filter weights of (a) $d = 0$, (b) $d = 0.3$, (c) $d = 0.8$, and (d) $d = 1$ with digital notch filtering

Figures 6-18 through 6-24 show the dynamic behaviors of the positioner in the *z*-direction when different values are used for the *Q* and *R* matrices than those used in Section 4.3 for the design of the LQG controller. For all the figures a step response of 5-μm was executed. Several trends on the step responses were noticed and helped on choosing the final matrix values.

For instance, if a small weight is given to the integrator states of the *Q*-matrix, the response of the system is overdamped with a slow rise time and there might be steady-state errors as seen in Figure 6-24. However if too much weight is given to the integrator states the overshoot will increase and the system will become underdamped with a long settling time as seen in Figures 6-22 and 6-23, and it might saturate the DAC card. On the other hand, if more weight were given to the velocity estimates, the position or integrators states of the system would be blown off. Therefore the optimal value of the *Q*-matrix is achieved by placing more weight on the integrator states but the values should not be too high or too low.

For the *R* matrix, one trend found was that smaller values lead to a response with a fast rise time and settling time, and a reasonable overshoot, however choosing a value close to zero for the *z*-input may lead to instability. Also since the step-responses were taken in *z*, the (1, 1) element of the diagonal matrix was the one that had the biggest effect on the response as seen in Figure 6-20. The other values correspond to the rotation about the *x*- and *y*-axes. It seems that smaller values for the *z*-input of the *R* matrix get the system the best response in *z*, as seen in Figure 6-18; however one has to make sure that it still maintains stability.

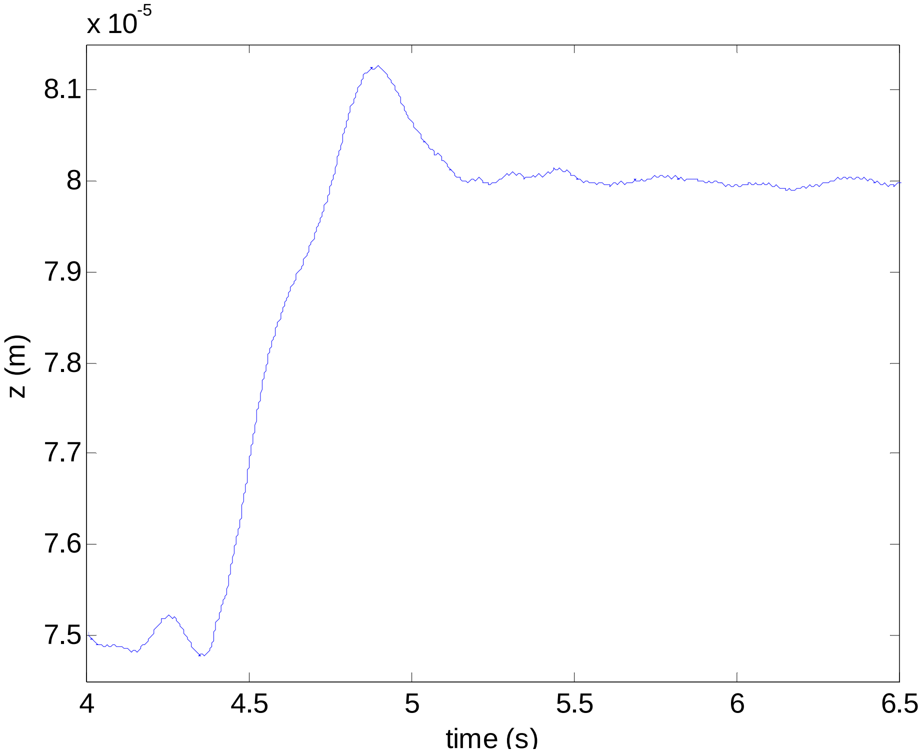


Figure 6-18: 5-µm step response in *z* with an *R* matrix of [2, 5, 5]

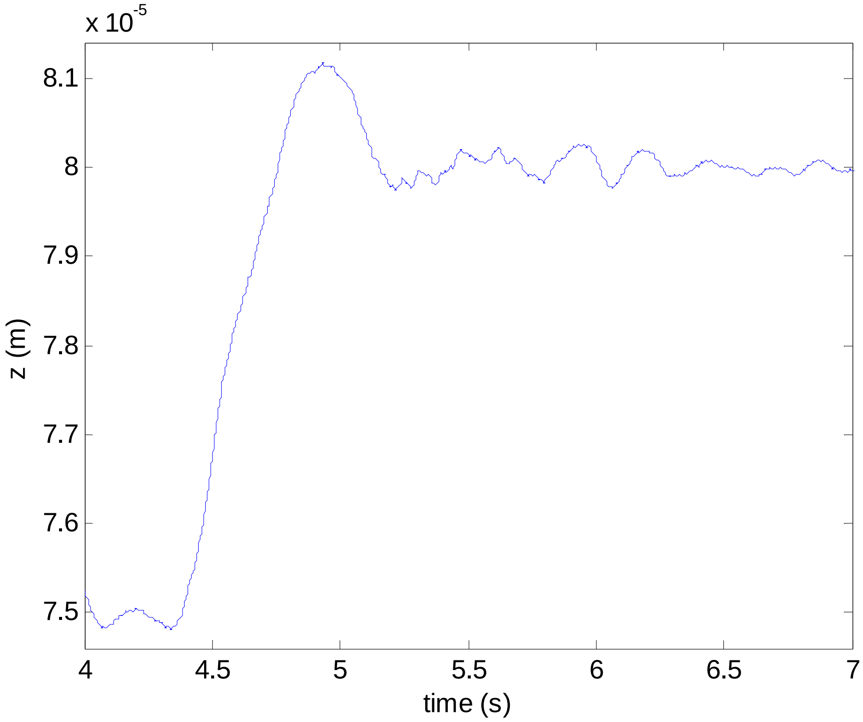


Figure 6-19: 5-μm step response in *z* with an *R* matrix of [5, 10, 10]

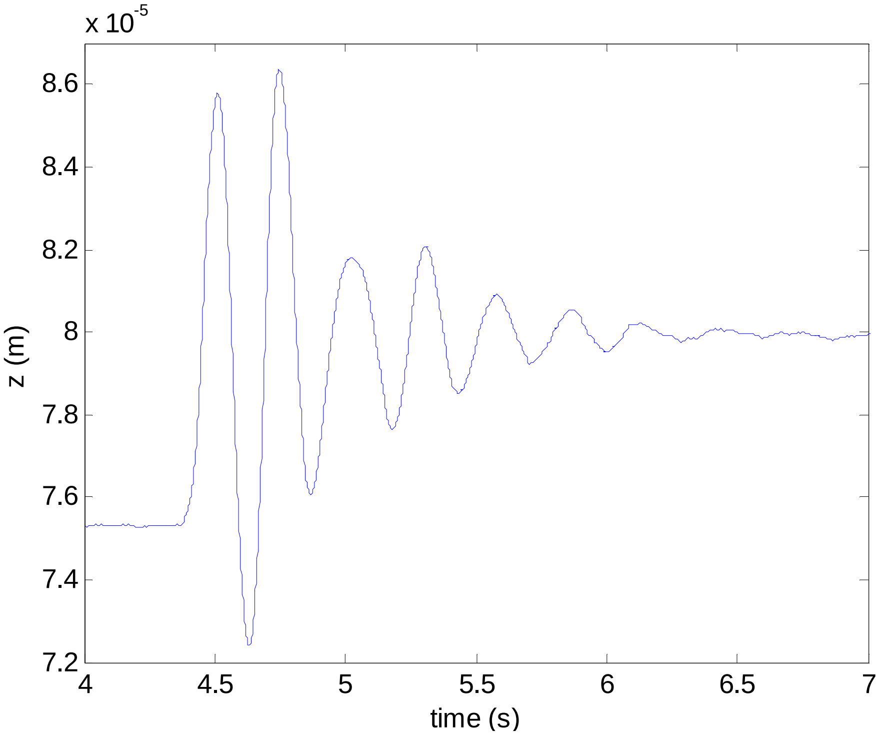


Figure 6-20: 5-μm step response in *z* with an *R* matrix of [30, 5, 5]

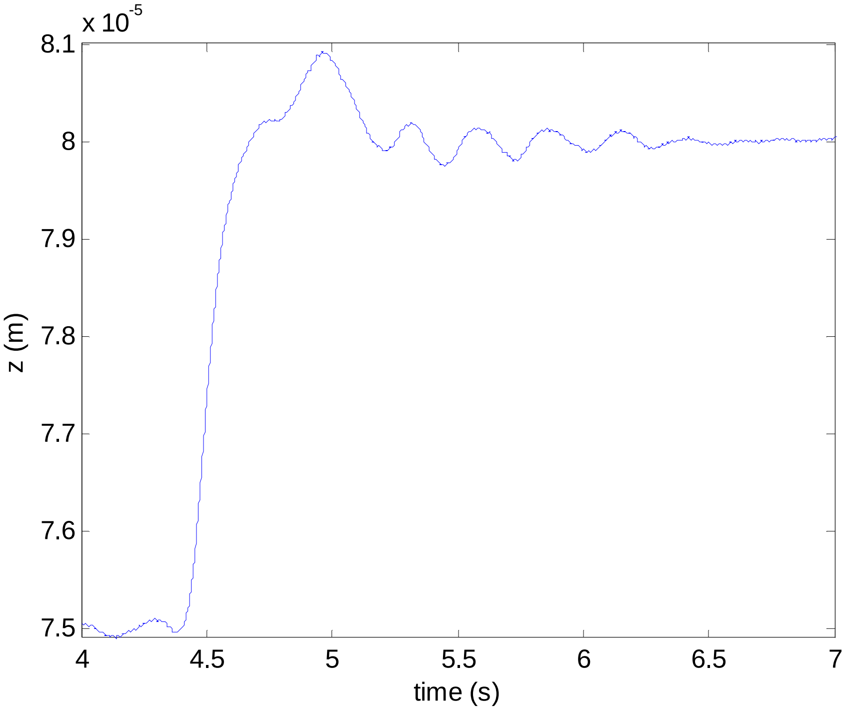


Figure 6-21: 5-µm step response in z with a *Q* matrix of [1e4, 1e4, 1e4, 1e9, 1e9, 1e9, 1e2, 1e2, 1e2]

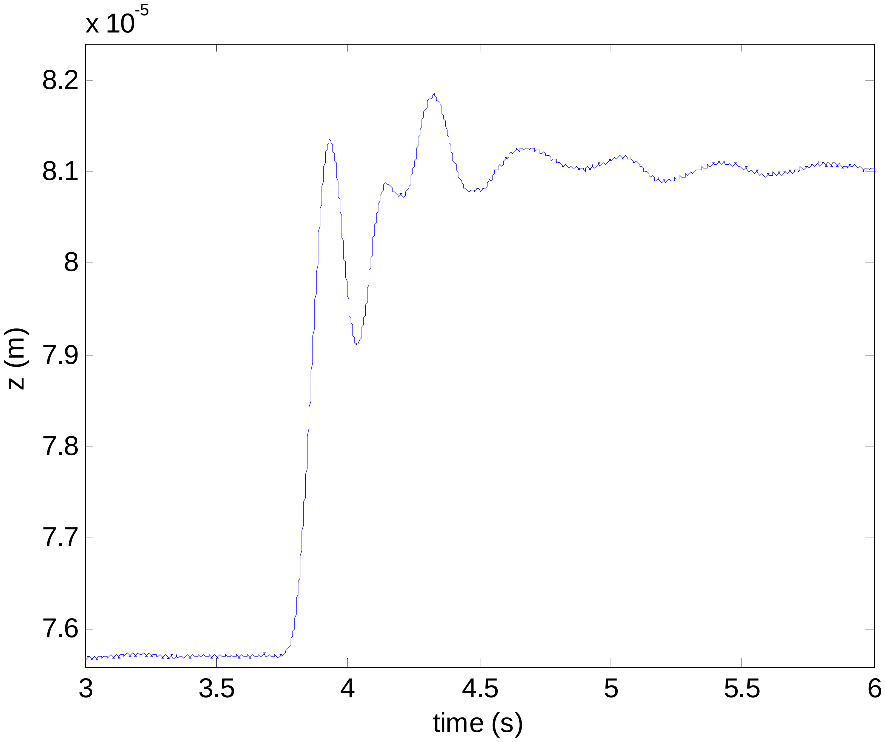


Figure 6-22: 5-µm step response in z with a *Q* matrix of [1e3, 1e3, 1e3, 1e9, 1e9, 1e9, 1e1, 1e1, 1e1]

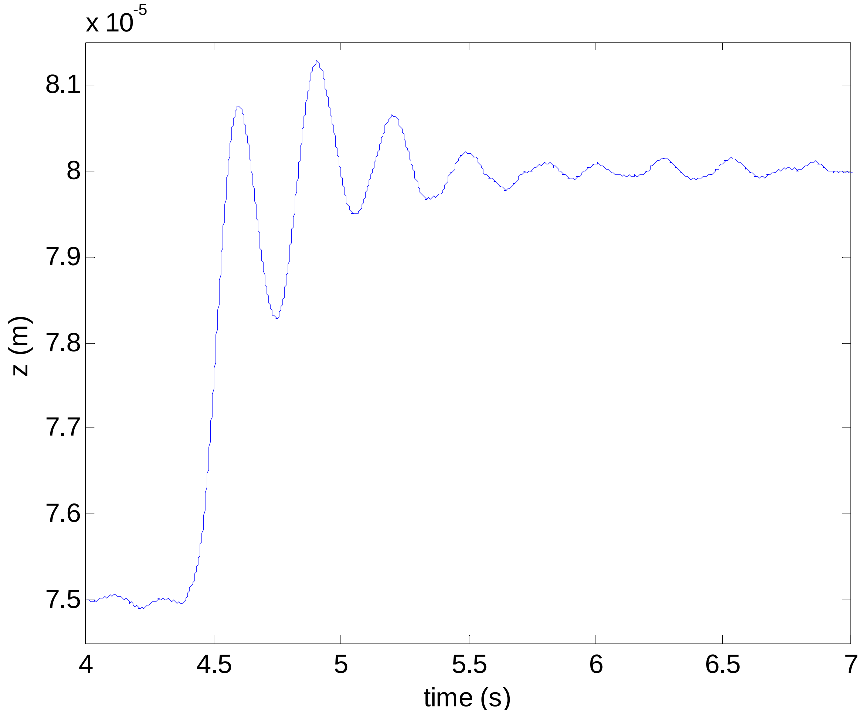


Figure 6-23: 5-μm step response in *z* with a *Q* matrix of [1e4, 1e4, 1e4, 1e10, 1e10, 1e10, 1e1, 1e1, 1e1]

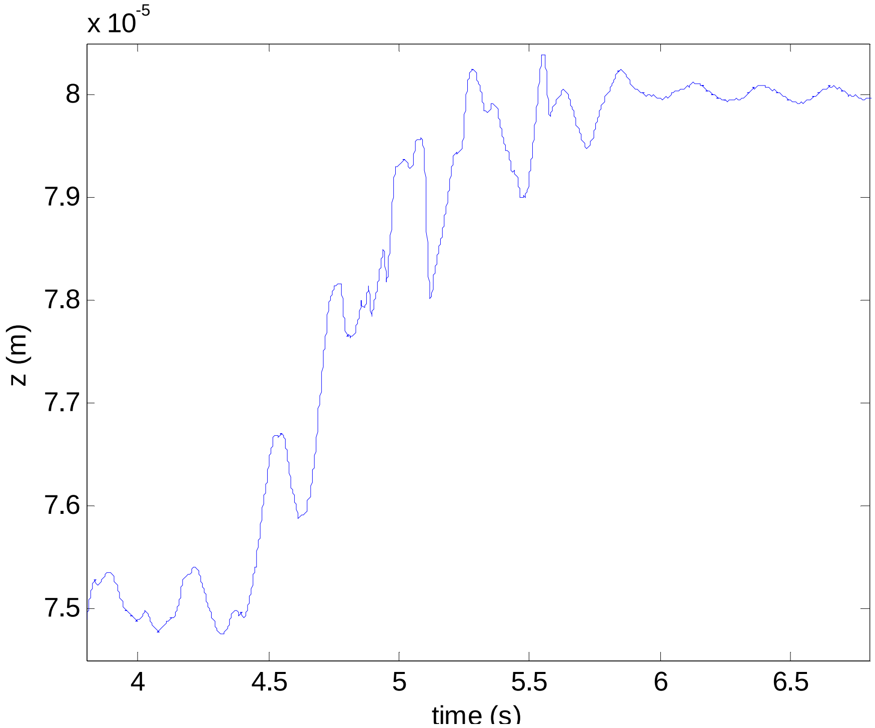


Figure 6-24: 5-μm step response in *z* with a *Q* matrix of [1e4, 1e4, 1e4, 1e7, 1e7, 1e7, 1e2, 1e2, 1e2]

6.3 Long-range Planar Motions

Planar motions in a long-range are commonly used in precision-positioning application, such as scanning, microlithography, and microscopy. Long-range planar motion generation is performed and several experimental results are presented in this section. These experimental results demonstrate the long-range scanning motion capabilities of the positioner.

First, one of the key advantages of this positioning stage is that the angular measurement range with the Hall-effect sensors is much larger than that of the laser interferometers. The rotation motion range of the laser interferometer is just a few milliradians because it uses the reflected laser beam, where laser beams stray off in large angles of motion. However, rotation motion control with Hall-effect sensors does not have this limitation. Hence, it can generate larger rotating motions than the system with laser interferometers. Figure 6-25 presents the large angle of the rotation motion of 12° with the Hall-effect sensors. The positioner started rotating to almost 6° and then the rotating motion switched to generate the 12° motion in the opposite direction.

The maximum travel ranges that can be generated by this positioner are of 220 mm in *x* and 200 mm in *y* are achieved in translation. The travel range of the positioner is only limited by the number of pitches in the magnet matrix, currently consisting of 6 by 6 pitches. The platen's geometric asymmetry accounts for the difference in the maximum travel ranges in *x* and *y*. Figure 6-26 shows a 160 mm travel range in *x*, and Figure 6-27 shows a 100 mm travel range in *y*. The scanning velocities are 6 mm/s in *x* and 4 mm/s in *y*. The two planar motors are employed for the *y*-directional motion, while a single planar motor is employed for the *x* direction motion. The maximum travel range in translation can be simply enlarged by increasing the number of pitches in the magnet matrix. Since the positioner with Hall-effect sensors does not need to carry the stick mirrors that reflect the laser beam in the laser interferometry, it can generate larger motions than the positioners using laser interferometers.

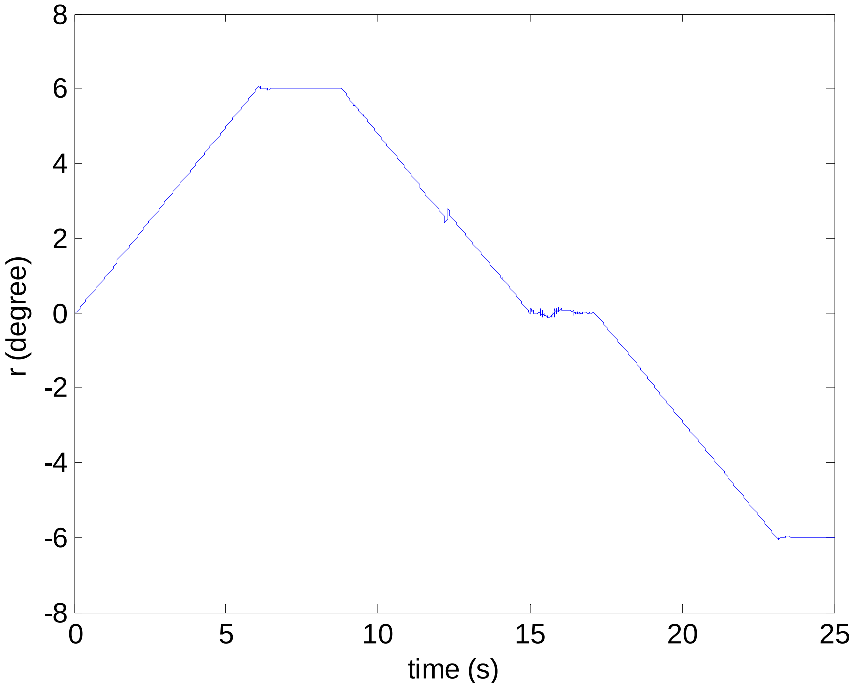


Figure 6-25: 12° long-range motion in ϕ

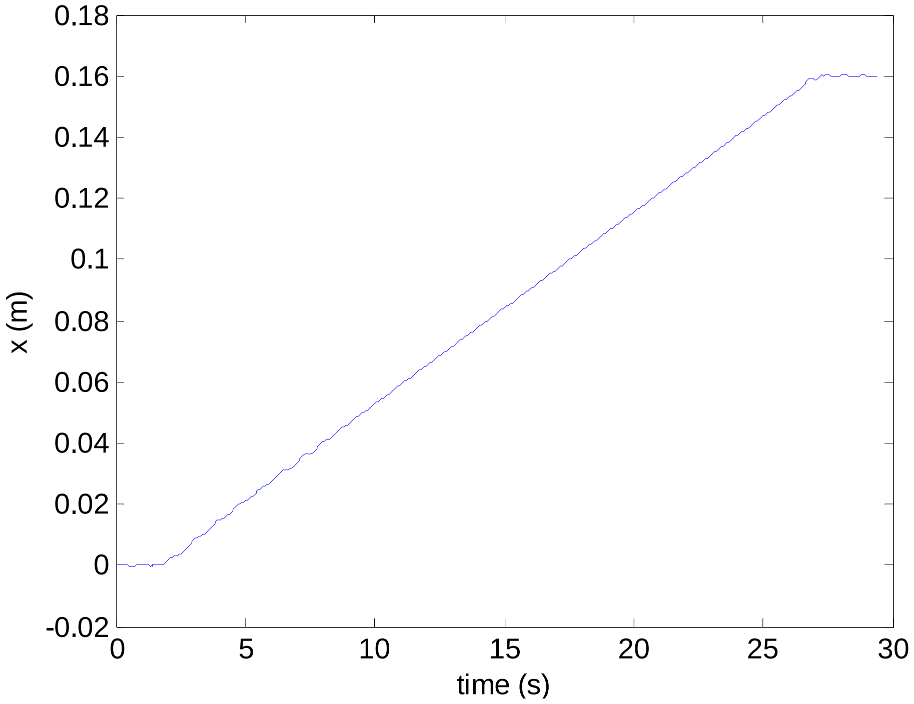


Figure 6-26: 160-mm long-range motion in x

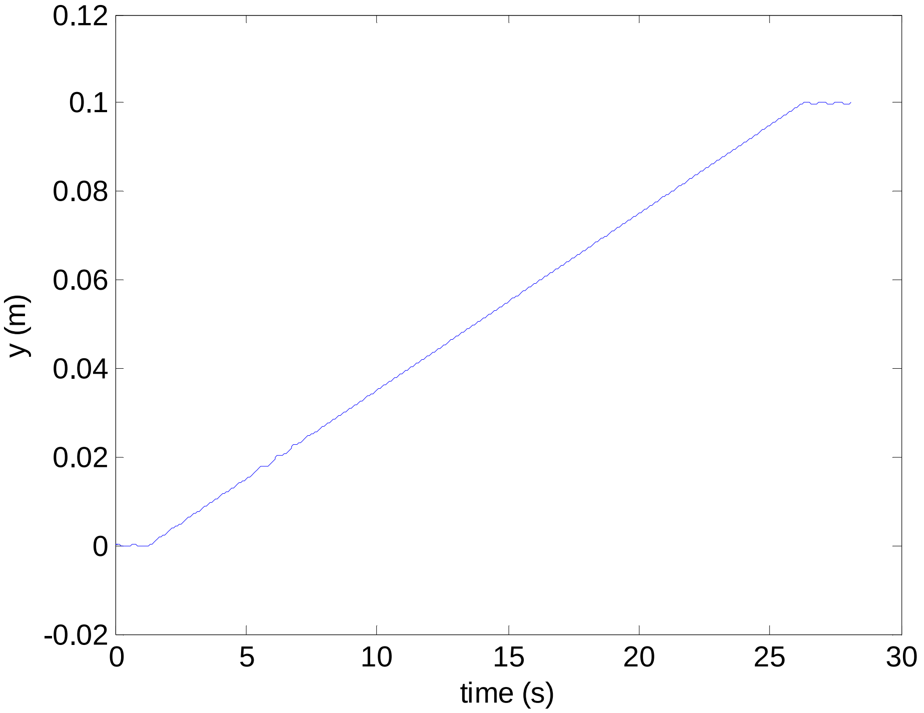


Figure 6-27: 100-mm long-range motion in *y*

Figure 6-28 shows the planar back-and-forth motions of 10 × 10 mm that demonstrate the capabilities in the wafer stepping/scanning process. A motion of an equilateral triangle with each side measuring 50-mm is shown in Figure 6-29 (a), the motion was achieved with the position errors in *x* and *y* shown in Figure 6-29 (b) and (c). The position errors of 0.34 mm in *x* and 0.70 mm in *y* are seen in the figure. Step-and-repeat motions were generated in the *x* and *y* direction, and each is shown in Figures 6-30 and 6-31. It is a useful demonstration for semiconductor-manufacturing applications. The positioner started the motion from the origin with a slow step motion of 4 mm/s in the positive *x*-direction with a step size of 5 mm each. After 9 s, the scanning motions in the *x*-direction turned negative. The fast scanning motion of 16 mm/s in the positive *y*-direction followed with a step size of 1 mm each. After 10 s, the scanning motions in the *y*-direction turned negative.

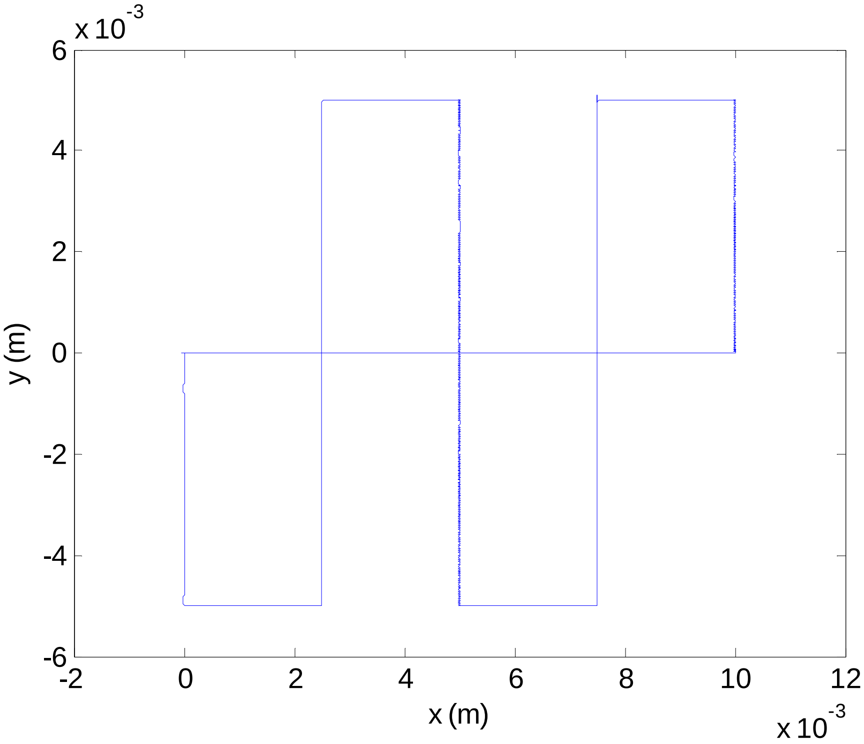
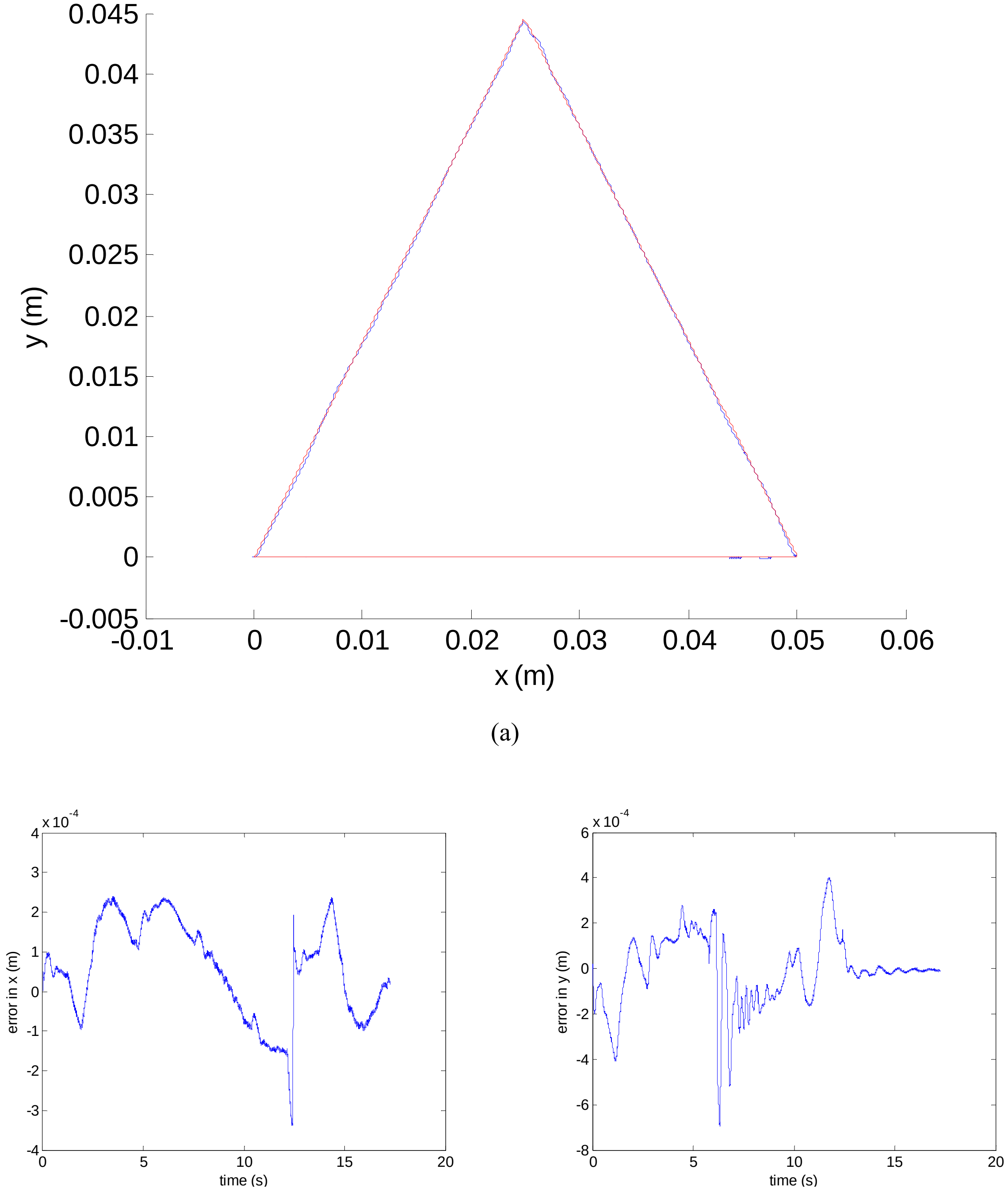


Figure 6-28: Back-and-forth motion of 10 × 10 mm in *x* and *y*



(a)

(b)

(c)

Figure 6-29: (a) 5-cm equilateral triangle motion, and position error (b) in *x* and (c) in *y*

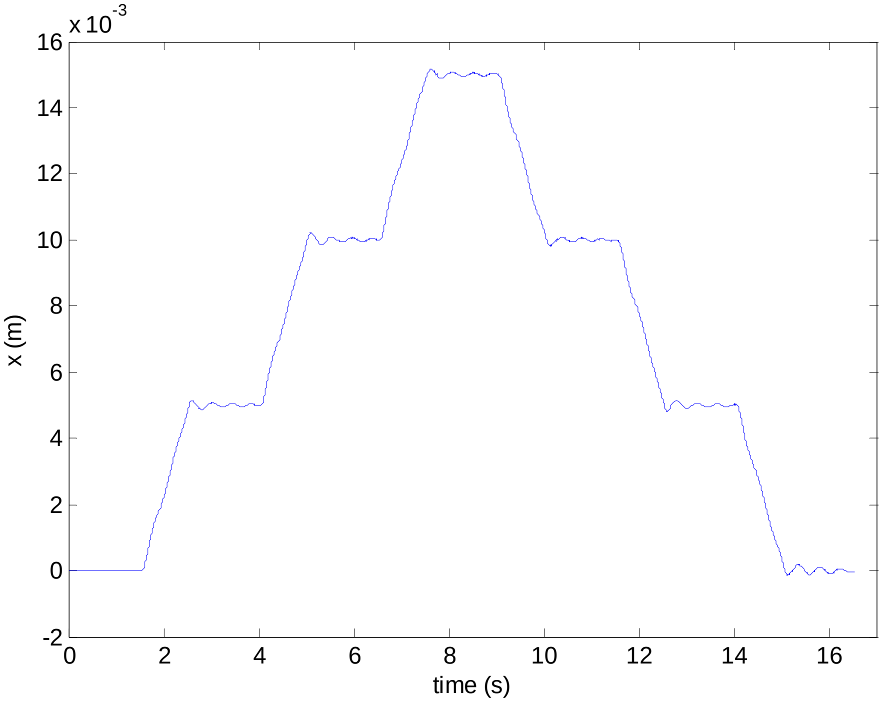


Figure 6-30: Slow 5-mm step-and-repeat motion in *x*

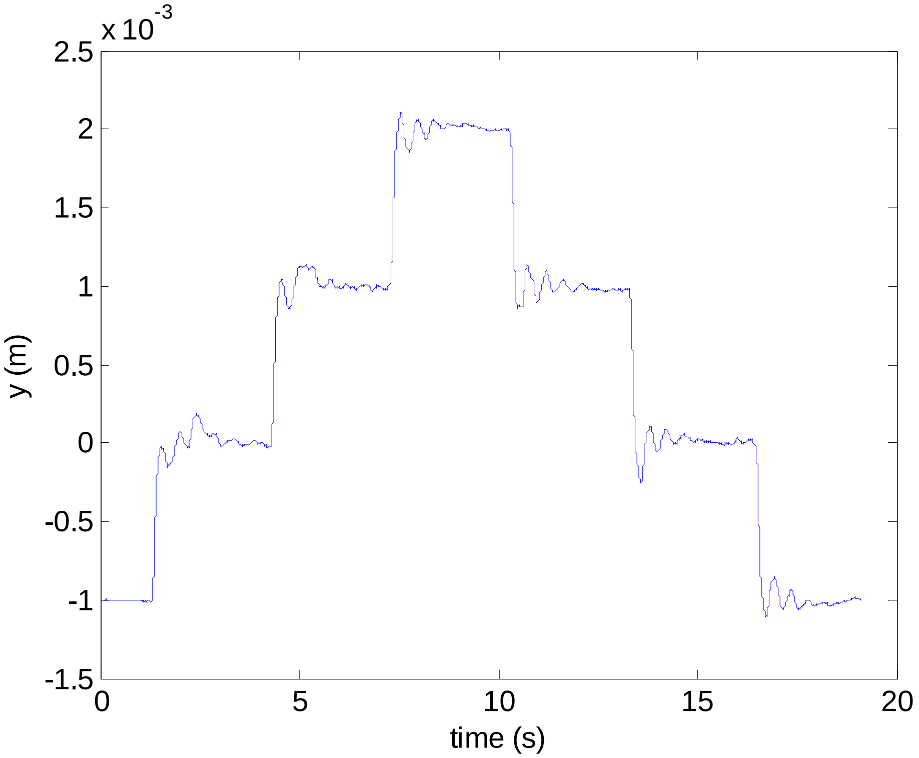


Figure 6-31: Fast 1-mm step-and-repeat motion in *y*

CHAPTER VII

CONCLUSIONS AND FUTURE WORK

This final chapter includes the conclusions of this research regarding the design and performance of the LQG multivariable controllers for the positioner, as well as the analysis of the Hall-effect sensor signals and the experimental achievements. Several suggestions for future work to improve the performance of the maglev platen are also given in this chapter.

7.1 Conclusions

This thesis presents two LQG controllers successfully implemented for a 6-DOF positioner with Linux-based real-time control. In addition an analysis was done for the Hall-effect sensors signals and based on the results several digital second-order filters were tested and implemented in order to achieve better resolution in the planar motion. Its potential applications in the semiconductor industry were demonstrated.

Hall-effect sensors are absolute position sensors and can be placed at any position atop a magnet array. Unrestricted translational and rotational moving ranges with respect to the size of magnet array are the advantage of the Hall-effect sensor. Since the Hall-effect sensors and the nanogage laser distance sensors only provide the position displacement data, a recursive hybrid filter was used to estimate the remaining states. In order to remove the steady-state error, the modified dynamic plant model was constructed with additional integrator terms. Good experimental results were achieved without steady-state errors.

The experimental setup, which is more cost-effective than that of laser interferometer, requires the Hall-effect sensors, ADC and DAC boards, power supplies, and power amplifiers. The positioner was controlled by a Linux-based real-time system. The real-time feedback control includes Ubuntu as a Linux OS, RTAI, and Comedi. The sampling frequency of the control system is 800 Hz.

The microscale accuracy and position errors are based on many different factors. The manufacturing error of the magnet array and the inconsistent magnet flux density may exist. The misalignment of Hall-effect sensors and an external magnetic field can interfere the accurate position sensing. The modeling error may affect the position output but the main reason is the sensitivity of the Hall-effect sensor and the noise coming from the wall outlet. The recursive discrete hybrid filter was used to achieve better position estimations from the noisy sensor signals. However, the position errors and noise from the Hall-effect sensors were significant even after the implementation of the recursive discrete hybrid filter. This is why an analysis was done on the Hall-effect sensor signals, based on the results from the analysis, a 50-Hz biased noise was found to have a significant effect in the sensor signals. A second-order digital low-pass and a second-order notch filters were tested and implemented in order to attenuate or eliminate such noise. The notch filter ended up being the filter that gave the best positioning resolution in the *x*-*y* plane.

After the planar multivariable LQG controller and filters were designed and implemented, the positioner was able to achieve a position resolution of 1.5 μm with position noise of 0.545 μm rms in the *x*-*y* plane. Experimental results were presented to display the performance of the controller and filter designs. Step responses of 10-μm, 20-μm, and 50-μm were implemented in the *x*-*y* plane, as well as step responses for the rotation around *z*. Several long range and step-and-repeat motions are also presented for the planar motion, for which the unrestricted moving range is a result of the periodic magnetic flux density in the *x*-*y* plane.

For the vertical mode multivariable LQG controller, levitation, the rotation around *x*, and the rotation around *y* were presented. Experimental results for step responses of 5 μm in *z* and 30 mrad rotation motion around *x*, and *y* were implemented. The positioner achieved a position resolution of less than 110 nm with position noise of 49.3 nm rms in *z*.

7.2 Suggestions for Future Work

This section suggests several possible ways to improve the precision positioner for further research. A positioner that has better dynamic performance and positioning resolutions can be designed with improving mechanical, electromagnetic, control and filter design.

Currently, the real-time control is performed by a PC with Linux operating system. Although Linux can provide stable real-time feedback control in a low capacity computer, the sampling speed to obtain high quality dynamic performance is limited. A Digital Signal Processor that can accommodate a high-speed sampling frequency, which may yield better dynamic performance.

A Hall-effect sensor exhibits the unlimited travel range with a microscale position resolution. Otherwise, a laser interferometer has nanoscale resolution in spite of its limited sensing range. Hence, a hybrid sensing methodology that has advantages both the two sensors can be proposed for a new positioning stage. The scanning motion that requires large moving range can be monitored by Hall-effect sensors, and the step-and repeat motions that need to be controlled in nanoscale can be measured by laser interferometers.

Finer tuning of the hybrid filter may yield better results, as well as looking into more elaborate techniques for the minimax filtering part of the hybrid filter. One can also look into further optimization of the Hall-Effect sensor signal, one way would be by modeling the sensors dynamics as a nonlinear system, not a linear system as I did, and doing this might lead to a better positioning resolution. Finally better positioning resolution in the *x*-*y* plane may be achieved by implementing a higher-order low-pass filter.

REFERENCES

[1] W.-J. Kim., June 1997, “High-precision planar magnetic levitation,” Ph.D. Dissertation, Dept. of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA.

[2] T.-J. Hu., May 2005, “Design and control of a 6-degree-of-freedom levitated positioner with high precision,” Ph.D. Dissertation, Texas A&M University, College Station, TX.

[3] Ir. J. C. Compter., April 2004, “Electro-dynamic planar motor,” Precision Engineering, **28**(2), pp. 171−180.

[4] V. Nguyen., May 2011, “A multi-axis compact positioner with a 6-coil platen moving over a superimposed Halbach magnet matrix,” M.S. Thesis, Texas A&M University, College Station, TX.

[5] H. Yu., August 2009, “Design and control of a compact 6-Degree-of-freedom precision positioner with linux-based real-time control,” Ph.D. Dissertation, Texas A&M University, College Station, TX.

[6] D. L. Trumper., W.-J. Kim., and M. E. Williams., March/April 1996, “Design and analysis framework for permanent-magnet machines,” IEEE Transactions on Industry Applications, **32**(2), pp. 371–379.

[7] Y. Kawato., May 2005, “Multi-DOF precision positioning methodology using two-axis Hall-effect sensors,” M.S. Thesis, Department of Mechanical Engineering, Texas A&M University, College Station, TX.

[8] Apex. [Online]., 1984, Available, 14 November 2011: http://apex.cirrus.com/en/.

[9] Hall-effect sensor. [Online]., 2003, Available, 14 November 2011: http://content.honeywell.com/sensing/prodinfo/solidstate/technical/chapter2.pdf.

[10] Sentron AG website. [Online]., 2010, Available, 14 November 2011: http://www.sentron.nl.

[11] G. A. Fries., A. A. Rizzi., and R. L. Hollis., May 1999, "Fluorescent dye based optical position sensing for planar linear motors," in *Proceedings of IEEE International Conference on Robotics & Automation*, Detroit, MI, **2**, pp. 1614−1619.

[12] Optra, Inc., Topsfield, MA. [Online]., 1985, Available, 14 November 2011: http://www.optra.com/nanogage.html.

[13] X. Shan., S.-K. Kuo., J. Zhang., and C.-H. Menq., March 2002, "Ultra precision motion control of a multiple degrees of freedom magnetic suspension stage." IEEE/ASME Transactions on Mechatronics, **7**(1), pp. 67-78.

[14] S. Skogestad., and I. Postlethwaite., 1996, *Multivariable Feedback Control Analysis and Design*, John Wiley & Sons, New York, NY.

[15] J. S. Bay., 1999, *Fundamentals of Linear State Space Systems*, WCB/McGraw-Hill, Boston, MA.

[16] L.Dai., October 1989, "Filtering and LQG problems for discrete-time stochastic singular system," IEEE Transactions on Automatic Control, **34**(10), pp. 1105-1108.

[17] P. S. Maybeck., 1979, *Stochastic Models, Estimation, and Control*, Academic Press, New York, NY.

[18] A. Parlos., Spring 2010, “MEEN 652 - Multivariable control system design,” *Lecture Notes.* Texas A&M University, College Station, TX.

[19] T. Glad., and L. Ljung., 2000, *Control Theory Multivariable and Nonlinear Methods,* Taylor & Francis, New York, NY.

[20] D. Simon., and H. El-Sheried., May 1996, “Hybrid Kalman/minimax filtering in phase-locked loops,” Control Engineering Practice, **4**(5), pp. 615-623.

[21] J. L. Crassidis., and J. L. Junkins., 2004, *Optimal Estimation of Dynamic Systems*, Chapman & Hall / CRC Press, 2nd ed. Boca Raton, FL.

[22] D. Simon., October 2001, “From here to infinity,” *Embedded Systems Programming,* **14**(11), pp. 20–32.

[23] U. Shaked., and Y. Theodor., December 1992, “$\mathcal{H}_\infty$ Optimal Estimation: A tutorial,” *in Proceedings of the 31st IEEE Conference on Decision and Control*, Tucson, AZ, **2**, pp 2278 – 2286.

[24] Zhou, K., and P. Khargonekar., 1988, “An algebraic Ricatti Equation approach to $\mathcal{H}_\infty$ Optimization” Systems and Control Letters, **11**, pp. 167 – 172.

[25] J. G. Proakis., and D. G. Manolakis., 1996, *Digital Signal Processing*, Prentice Hall, 3rd ed. Upper Saddle River, NJ.

APPENDIX A

MATLAB CODE FOR LQR CONTROL LAWS

```
clear all
%
% %LQR for lateral motion only
% A=[zeros(3,6),eye(3);eye(3),zeros(3,6);zeros(3,9)];
% B=[zeros(6,3); 0,0,13.2547; 13.2547,13.2547,0; -235.91, 224.65, -206.78];
% C=[eye(3),zeros(3,6)]; D=[zeros(3,3)];
%
%
% Q=eye(9);
% Q(1,1)=1e24; Q(2,2)=1e24; Q(3,3)=1e24;
% Q(4,4)=1e24; Q(5,5)=1e24; Q(6,6)=1e24;
% Q(7,7)=1e24; Q(8,8)=1e24; Q(9,9)=1e24;
% R=[1e10,0,0;0,1e10,0;0,0,1e10];
%
% [G,S,E] = lqr(A,B,Q,R);
%
% Gs=ss(A,B,C,D);
% Ks=ss(A-B*G,B,G,D);
% GKs=series(Ks,Gs);
% figure(2)
% sigma(GKs,{1e-1,1e1});
%
% %Hfeed=feedback(GKs,eye(3),-1);
% %sigma(Hfeed,{10^-2 10^3});
%
% [Gd,Sd,Ed] = lqrd(A,B,Q,R,1/800);
% Gd

%LQR for upwards motion only
A=[zeros(3,6),eye(3);eye(3),zeros(3,6);-407.9,zeros(1,8);0,-34246.6,
zeros(1,7);0,0,-40408.7,zeros(1,6)];
B=[zeros(6,3); 13.255,13.255,13.255; -209.86,-209.86,370.35; 371.87,-354.28,
12.73];
C=[eye(3),zeros(3,6)]; D=[zeros(3,3)];
Q=eye(9);
Q(1,1)=1e5; Q(2,2)=1e4; Q(3,3)=1e4;
Q(4,4)=1e9; Q(5,5)=1e9; Q(6,6)=1e9;
Q(7,7)=1e2; Q(8,8)=1e2; Q(9,9)=1e2;
R=1*eye(3);

[G,S,E] = lqr(A,B,Q,R);

Gs=ss(A,B,C,D);
Ks=ss(A-B*G,B,G,D);
GKs=series(Ks,Gs);
figure(2)
sigma(GKs);

%Hfeed=feedback(GKs,eye(3),-1);
%sigma(Hfeed,{10^-2 10^3});

[Gd,Sd,Ed] = lqrd(A,B,Q,R,1/800);
Gd
```

APPENDIX B

MATLAB CODE FOR HYBRID FILTER

```
 function [ErrKarray, ErrHinfarray] = jojo(g, T, tf)

% INPUTS
%   g = gamma for minimax filter
%   T = time step in seconds
%   tf = final time in seconds
% OUTPUTS
%   ErrKarray = time varying array of error of Kalman state estimate
%   ErrHinfarray = time varying array of error of Minimax state estimate

if ~exist('g', 'var')
    g = 10;end
if ~exist('T', 'var')
    T = 1/800;end
if ~exist('tf', 'var')
    tf = 200/800;end

Q=zeros(6); Q(1,1)=1e3;Q(2,2)=1e3;Q(3,3)=1e5;
Q(4,4)=1e2;Q(5,5)=1e2;Q(6,6)=1e2; % Process noise covariance
R = 1e-6*[eye(6)]; % Measurement noise covariance (m, m, m)

% Define the initial state x, initial Kalman filter estimate xhat
x = [7;7;1;5;1;1; 0;0;0;0;0;0; 0;0;0;0;0;0;];
P = 1e-9*eye(18); % Initial estimation error covariance

% System matrices
dt=0.00125; %800Hz
A = [eye(6) dt*eye(6) 1/2*dt^2*eye(6);  zeros(6) eye(6) dt*eye(6);  zeros(6)
zeros(6) eye(6)];
% Input matrix.
B = [dt^3/6*eye(6); dt^2/2*eye(6); dt*eye(6)];
% measurement matrix.
C =[eye(6) zeros(6,12)];
% Initialize arrays for saving data for plotting.
xhat = x;
xarray = [];
xhatarray = [];
randn('state', sum(100*clock));

% Minimax initialization. & mixed
Qbar = P;
xhatinf = x;
xhatinfarray = [];
xhatmix = x;
xhatmixarray = [];

for t = T : T : tf
    % Get the noise-corrupted measurement z.
    z = C * x;
    MeasErr = .0000005*randn(size(z));
    zi = z + MeasErr;
    %step
    if t==0.1
```

```
        x = [7.000005;7;1;5.0005;1;1; 0;0;0;0;0;0; 0;0;0;0;0;0;];end

    % Run the unconstrained minimax filter.
    Pinf = inv(eye(9) - (Qbar /g^2) + Qbar * C' * C) * Qbar;
    Qbar = A * Pinf * A' + eye(9);
    Kh = A * Pinf * C';

    xhatinf = A * xhatinf + Kh * (zi - C * xhatinf);
    xhatinfarray = [xhatinfarray xhatinf];

    % Run the unconstrained Kalman filter.
    K = P * C' * inv(C * P * C' + R);
    % Update the state estimation error covariance.
    Pup=(eye(18) - K*C)*P;
    P=Pup;
    % Estimate the heading on the basis of the state estimate.
    xhat = xhat + K * (zi - C * xhat);
    xhat = A*xhat;
    P = (A * P * A') + B*Q*B';
   xhatarray = [xhatarray xhat];

   %Mixed/Hybrid filtering
    dd=.40;     %hybrid filter weight
    MK=dd*K+(1-dd)*Kh;
    xhatmix = A * xhatmix + MK * (zi - C * xhatmix);
    xhatmixarray = [xhatmixarray xhatmix];
    % Simulate the system dynamics.
    x = A * x + .0000001*randn(size(x));

    % Uncomment the following line to add unmodeled process noise.
    %x = x + [3e-7; 0; 0; 0; 0; 0; 0; 0; 0];
    xarray = [xarray x];
    end

% % Compute average position estimation errors.
EstError = xarray - xhatarray;
EstError = sqrt(EstError(1,:).^2 + EstError(2,:).^2); EstError =
mean(EstError);
disp(['Average Kalman Unconstrained Position Estimation Error = ',
num2str(EstError)]);

EstError = xarray - xhatinfarray;
EstError = sqrt(EstError(1,:).^2 + EstError(2,:).^2); EstError =
mean(EstError);
disp(['Average Minimax Unconstrained Position Estimation Error = ',
num2str(EstError)]);

EstError = xarray - xhatmixarray;
EstError = sqrt(EstError(1,:).^2 + EstError(2,:).^2); EstError =
mean(EstError);
disp(['Average Hybrid Unconstrained Position Estimation Error = ',
num2str(EstError)]);

% % Compute average velocity estimation errors.
% EstError = xarray - xhatarray;
% EstError = sqrt(EstError(5,:).^2 + EstError(4,:).^2); EstError =
mean(EstError);
% disp(['Average Kalman Unconstrained Velocity Estimation Error = ',
num2str(EstError)]);
```

```
%
% EstError = xarray - xhatinfarray;
% EstError = sqrt(EstError(5,:).^2 + EstError(4,:).^2); EstError =
mean(EstError);
% disp(['Average Minimax Unconstrained Velocity Estimation Error = ',
num2str(EstError)]);

%
% EstError = xarray - xhatmixarray;
% EstError = sqrt(EstError(5,:).^2 + EstError(4,:).^2); EstError =
mean(EstError);
% disp(['Average Hybrid Unconstrained Velocity Estimation Error = ',
num2str(EstError)]);

 t = 0 : T : tf-T;

% Plot the position step responses.
figure;
plot(t, xarray(1, :), '-', t, xhatarray(1, :), 'm-',t, xhatinfarray(1,
:),'r:',t, xhatmixarray(1, :), 'x');
set(gca,'FontSize',12); set(gcf,'Color','White');
legend('true state', 'Kalman estimate', 'H_{\infty} estimate', 'Hybrid');
title('True Position') ;xlabel('seconds'); ylabel('meters');

% % Plot the position errors.
% figure;
% plot(t, xarray(1, :) - xhatarray(1, :), ':', ...
%     t, xarray(1, :) - xhatinfarray(1, :), '-',t, xarray(1, :) -
xhatmixarray(1, :));
% set(gca,'FontSize',12); set(gcf,'Color','White');
% legend('Kalman estimate', 'H_{\infty} estimate', 'Hybrid');
% title('Position Estimation Errors'); xlabel('seconds'); ylabel('meters');

% % Plot the velocity errors.
% figure;
% plot(t, xarray(4, :) - xhatarray(4, :), ':', ...
%     t, xarray(4, :) - xhatinfarray(4, :), '-',t, xarray(4, :) -
xhatmixarray(4, :));
% legend('Kalman estimate', 'H_{\infty} estimate', 'Hybrid');
% set(gca,'FontSize',12); set(gcf,'Color','White');
% title('Velocity Estimation Errors'); xlabel('seconds'); ylabel('m/s');
```

APPENDIX C

C-CODE FOR LQG CONTROLLERS

This appendix presents a sample of the C-code used to run the magnetic positioner. This program includes the LQG controller for the horizontal motion and the hybrid filter.

```
//* signal.c
// 3-DOF Kalman FIlter - LQG all 3-axis (w integrators)

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <signal.h>
#include <string.h>
#include <asm/errno.h>
#include <sys/types.h>
#include <sys/user.h>
#include <sys/mman.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <math.h>
#include "/home/comedilib-0.8.0/comedilib/include/comedilib.h"
#include "hall1.h"

#include <rtai_sem.h>
#include <rtai_usi.h>
#include <rtai_lxrt.h>

#define MAX_BANDWIDTH 333000
#define DFLT_FREQUENCY 1000

//Functions
void GLSDC_2(void);
void GLSDC_3(void);
void sensor_test(void);
void EKF_2(void);
void x_mov(void);
void y_mov(void);

comedi_t *it;
comedi_t *it2;
int in_subdev = 0; //0 is AIO
int out_subdev = 0;
int sigtest = 0;

int in_chan_1 = 0; //Vertical senfor (NANOGAGE100)
int in_chan_2 = 1;
int in_chan_3 = 2;
int in_chan_4 = 3; //A1 Hall-effect sensor
int in_chan_5 = 5; //A2
int in_chan_6 = 6; //B1
int in_chan_7 = 7; //B2
int in_chan_8 = 8; //C1_x
int in_chan_9 = 9; //C2_y

int out_chan_1 = 9; //A
int out_chan_2 = 10;
int out_chan_3 = 11;
int out_chan_4 = 6; //B
int out_chan_5 = 7;
int out_chan_6 = 8;
int out_chan_7 = 3; //C
int out_chan_8 = 4;
int out_chan_9 = 5;

// DIO test
int out_chan_10 = 0;

int in_range = 0;
int out_range = 0;
int aref = AREF_GROUND;

int i=0;
int j=0,p=0,k=0;
float m1=0,m2=0,m3=0,m4=0,m5=0,m6=0,m7=0,m8=0,m9=0;
char h=0,y=0;
char q;

//rtai declarations
unsigned long mtsk_name;
RT_TASK *mtsk;
struct sched_param mysched;

//comedi declarations
lsampl_t in_data_1;
lsampl_t in_data_2;
lsampl_t in_data_3;
lsampl_t in_data_4; //Hall-effect sensor inputs
lsampl_t in_data_5;
lsampl_t in_data_6;
lsampl_t in_data_7;
lsampl_t in_data_8;
lsampl_t in_data_9;

lsampl_t out_data_1;
lsampl_t out_data_2;
lsampl_t out_data_3;
lsampl_t out_data_4;
lsampl_t out_data_5;
lsampl_t out_data_6;
lsampl_t out_data_7;
lsampl_t out_data_8;
lsampl_t out_data_9;
lsampl_t out_data_10;

int t=0,zc_lev=0,sc_ang=0,tc_ang=0;
float z_pos1=0, z_pos2=0, z_pos3=0,z1,z2,z3,z_pos10=0,z_pos20=0,z_pos30=0;
float i_Aa,i_Ab,i_Ba,i_Bb,i_Ca,i_Cb,i_Aq,i_Bq,i_Cq,i_Ad,i_Bd,i_Cd;
float xr=0,xr1=0,yr=0,xc=0,yc=0,rr=0,rc=0,f21h=0.04963;
float x0_est=0,y0_est=0,r0_est=0;
float fA=0,fB=0,fC=0,fZA=0,fZB=0,fZC=0;
float sr=0,tr=0,zc=0,sc=0,tc=0,zr=0,zr10=0,zr_vol=0, izc;
float Ct,St,ho=0,yi,xi,XR,YR,cos_x,cos_y,sin_x,sin_y;
float xA_pos,xB_pos,xB_pos_cal,yA_pos,yB_pos, xrpre=0, yrpre=0,
xA_after,xB_after, delta_x,xC_pos,yC_pos;
float delta_xA=0, delta_xB=0,delta_xC=0, xA_pre=0, xB_pre=0, xC_pre=0,
xc1=0,yc1=0,xc2=0,yc2=0;
float va1e=0,va2e=0, xr_pre, xd=0, xdpre=0, xa1_pre=0, xc1_pre=0, xe=0,
error_x=0, error_y=0;
float mj=0, mn=0, xa1_pre2=0, xc1_pre2=0, delta2_xA=0, delta2_xC=0,
delta_xA2=0, delta_xC2=0;
float delta_yA=0, delta_yB=0, ya1_pre=0, yb1_pre=0, delta_y=0, yd=0, ye=0;
float u0rr=0,u0b=0,u1b=0,u2b=0,er0b=0,er1b=0,er2b=0,u0br=0,bc=0,br=0;
float sea=0,sec=0,delta_sea=0,delta_sec=0,ya1_old=0,
yc1_old=0,delta_sya=0,delta_syc=0;
float rrr=0,brr=0,xa1_old=0,xb1_old=0,sxa=0,sxb=0,delta_sxa=0,
delta_sxb=0,jul=0;
float dm=0, spd=0,spdop=0;

// CALIBRATION FACTORS
float ya1,xa1,yb1,xb1;
float kya= 1.0213;
float yaoffset = -0.035;
float kyb = 0.891;
float yboffset = 0.13;
float kxa = 1.19625;
float xaoffset = 0.02;
float kxb = 0.805;
float xboffset = 0.095;
float kyc=0.8824;
float ycoffset=0.1;
float kxc=0.6994;
float xcoffset=-0.06;

float z0er=0, z0cer=0, z1cer=0, z0r=0, z1r=0, z0cr=0, z1cr=0;
float er0z=0,er0s=0,er0t=0,u0z=0,
u0s=0,u1s=0,u2s=0,u0t=0,u1t=0,u2t=0,u1z=0,u2z=0,er1z=0,er2z=0;
float er0x=0,er1x=0,er2x=0,er0y=0,er1y=0,er2y=0,er0r=0,
er1r=0,er2r=0,er1t=0,er2t=0,er1s=0,er2s=0;
float u0x=0,u1x=0,u2x=0,u0y=0,u1y=0,u2y=0,u0r=0,u1r=0,u2r=0;
float vo00=0,vo01=0,vo02=0,vi00=0,vi01=0,vi02=0,vo10=0,vo11=0,
vo12=0,vi10=0,vi11=0,vi12=0;
float vo20=0,vo21=0,vo22=0,vi20=0,vi21=0,vi22=0,vi30=0,
vi31=0,vi32=0,vo30=0,vo31=0,vo32=0;

float ier0x=0,ier0y=0,ier0r=0,ier1x=0,ier1y=0,ier1r=0;
float stepsize=0,step=0, ux,uy,ur;

float current_A_1 = 0.0, current_A_2 = 0.0, current_A_3 = 0.0;
float current_B_1 = 0.0, current_B_2 = 0.0, current_B_3 = 0.0;
float current_C_1 = 0.0, current_C_2 = 0.0, current_C_3 = 0.0;

int in_maxdata = 0, out_maxdata = 0;
comedi_range *in_range_ptr, *out_range_ptr;

void solve(void);       //* For calculation of control input
void result(void);      //* For Book keeping
void terminate_normally(int signo);

int main(int argc, char* argv[])
```

```
{
       FILE *fp1 = NULL; //DATA SAVE
       FILE *fp2 = NULL;
       FILE *fp3 = NULL;
       FILE *fp4 = NULL;
       FILE *fp5 = NULL;
       FILE *fp6 = NULL;
RTIME actual_period = 0;

       //signal handling
       struct sigaction sa;
       mtsk_name = nam2num("MTSK");

       //Initialize the signal handling structure
       sa.sa_handler = terminate_normally;
       sa.sa_flags = 0;
       sigemptyset(&sa.sa_mask);

       if(sigaction(SIGINT, &sa, NULL))
       {      perror("sigaction");}
       if(sigaction(SIGTERM, &sa, NULL))
       {perror("sigaction");}

       fp1 = fopen("result1.txt", "w"); //*xls -> txt
       fp2 = fopen("result2.txt", "w");
       fp3 = fopen("result3.txt", "w");
       fp4 = fopen("result4.txt", "w");
       fp5 = fopen("result5.txt", "w");
       fp6 = fopen("result6.txt", "w");
       if(fp1 == NULL)
       {
              printf("Error opening result1.txt file\n");
              exit(1);
       }

       fp1 = fopen("result1.txt", "w");
       if(fp1 == NULL)
       {
              printf("Error opening result1.txt file\n");
              exit(1);
       }
        if(fp2 == NULL)
        {
                printf("Error opening result2.txt file\n");
                exit(1);
        }

        fp2 = fopen("result2.txt", "w");
        if(fp2 == NULL)
        {
                printf("Error opening result2.txt file\n");
                exit(1);
        }
        if(fp3 == NULL)
        {
                printf("Error opening result3.txt file\n");
                exit(1);
        }

        fp3 = fopen("result3.txt", "w");
```

```
 if(fp3 == NULL)
 {
         printf("Error opening result3.txt file\n");
         exit(1);
 }
if(fp4 == NULL)
{
       printf("Error opening result4.txt file\n");
       exit(1);
}
fp4 = fopen("result4.txt", "w");
if(fp4 == NULL)
{
       printf("Error opening result4.txt file\n");
       exit(1);
}
 if(fp5 == NULL)
 {
         printf("Error opening result5.txt file\n");
         exit(1);
 }
 fp5 = fopen("result5.txt", "w");
 if(fp5 == NULL)
 {
         printf("Error opening result5.txt file\n");
         exit(1);
 }
 if(fp6 == NULL)
 {
         printf("Error opening result6.txt file\n");
         exit(1);
 }
 fp6 = fopen("result6.txt", "w");
 if(fp6 == NULL)
 {
         printf("Error opening result6.txt file\n");
         exit(1);}

it = comedi_open("/dev/comedi0");
it2 = comedi_open("/dev/comedi1");
if(it == NULL)
{      printf("Could not open comedi\n");
       exit(1);}
if(it2 == NULL)
{      printf("Could not open comedi\n");
       exit(1);}

in_maxdata = comedi_get_maxdata(it, in_subdev, in_chan_1);
in_maxdata = comedi_get_maxdata(it, in_subdev, in_chan_2);
in_maxdata = comedi_get_maxdata(it, in_subdev, in_chan_3);
in_maxdata = comedi_get_maxdata(it, in_subdev, in_chan_4);
in_maxdata = comedi_get_maxdata(it, in_subdev, in_chan_5);
in_maxdata = comedi_get_maxdata(it, in_subdev, in_chan_6);
in_maxdata = comedi_get_maxdata(it, in_subdev, in_chan_7);
in_maxdata = comedi_get_maxdata(it, in_subdev, in_chan_8);
in_maxdata = comedi_get_maxdata(it, in_subdev, in_chan_9);

out_maxdata = comedi_get_maxdata(it2, out_subdev, out_chan_1);
out_maxdata = comedi_get_maxdata(it2, out_subdev, out_chan_2);
out_maxdata = comedi_get_maxdata(it2, out_subdev, out_chan_3);
```

```
       out_maxdata = comedi_get_maxdata(it2, out_subdev, out_chan_4);
       out_maxdata = comedi_get_maxdata(it2, out_subdev, out_chan_5);
       out_maxdata = comedi_get_maxdata(it2, out_subdev, out_chan_6);
       out_maxdata = comedi_get_maxdata(it2, out_subdev, out_chan_7);
       out_maxdata = comedi_get_maxdata(it2, out_subdev, out_chan_8);
       out_maxdata = comedi_get_maxdata(it2, out_subdev, out_chan_9);

       in_range_ptr = comedi_get_range(it, in_subdev, in_chan_1, in_range);
       in_range_ptr = comedi_get_range(it, in_subdev, in_chan_2, in_range);
       in_range_ptr = comedi_get_range(it, in_subdev, in_chan_3, in_range);
       in_range_ptr = comedi_get_range(it, in_subdev, in_chan_4, in_range);
       in_range_ptr = comedi_get_range(it, in_subdev, in_chan_5, in_range);
       in_range_ptr = comedi_get_range(it, in_subdev, in_chan_6, in_range);
       in_range_ptr = comedi_get_range(it, in_subdev, in_chan_7, in_range);
       in_range_ptr = comedi_get_range(it, in_subdev, in_chan_8, in_range);
       in_range_ptr = comedi_get_range(it, in_subdev, in_chan_9, in_range);

out_range_ptr = comedi_get_range(it2, out_subdev, out_chan_1, out_range);
out_range_ptr = comedi_get_range(it2, out_subdev, out_chan_2, out_range);
out_range_ptr = comedi_get_range(it2, out_subdev, out_chan_3, out_range);
out_range_ptr = comedi_get_range(it2, out_subdev, out_chan_4, out_range);
out_range_ptr = comedi_get_range(it2, out_subdev, out_chan_5, out_range);
out_range_ptr = comedi_get_range(it2, out_subdev, out_chan_6, out_range);
out_range_ptr = comedi_get_range(it2, out_subdev, out_chan_7, out_range);
out_range_ptr = comedi_get_range(it2, out_subdev, out_chan_8, out_range);
out_range_ptr = comedi_get_range(it2, out_subdev, out_chan_9, out_range);
begin:
        printf("Choose a(controller),s(NanoGage),n(Exit))\n");
        scanf("%c",&h);

       t=0.0; XR=0; YR=0;
       yc=0; yr=0;yi=0; xi=0; ux=0; uy=0; ur=0;
       u0z=0; u1z=0; u2z=0; er0z=0; er1z=0; er2z=0;
       u0x=0; u1x=0; u2x=0; er0x=0; er1x=0; er2x=0;
       u0y=0; u1y=0; u2y=0; er0y=0; er1y=0; er2y=0;
       u0r=0; u1r=0; u2r=0; er0r=0; er1r=0; er2r=0;
       u0s=0; u1s=0; u2s=0; er0s=0; er1s=0; er2s=0;
       u0t=0; u1t=0; u2t=0; er0t=0; er1t=0; er2t=0;
       cos_x=0; cos_y=0; sin_x=0; sin_y=0;

       // GLSDC & EKF
       deltay[0]=0;deltay[1]=0;deltay[2]=0;deltay[3]=0;
       X0=0; Y0=0; theta=0; det=0;
       X01=0; Y01=0; theta1=0;
       X02=0; Y02=0; theta2=0;
       X03=0; Y03=0; theta3=0;
       X04=0; Y04=0; theta4=0; Ct=0; St=0;
       matA[9][9]=0; matB[9][9]=0; matC[9][9]=0;
       xk[9]=0; xknew[9]=0; Pknew[9][9]=0; ytildek[3]=0;
       Kk[9][9]=0; invA[9][9]=0; deltax[3]=0; Jbefore=0; Jafter=0;
       UpskQkUpskT[9][9]=0; Pk[9][9]=0;

       printf("%f\n", deltay[3]);

       mysched.sched_priority = 99;

       if( sched_setscheduler(0, SCHED_FIFO, &mysched ) == -1){
       puts( "ERROR IN SETTING THE SCHEDULER UP");
       perror( "error" );
       exit( 0 );   }
       mlockall(MCL_CURRENT | MCL_FUTURE);

       if (!(mtsk = rt_task_init(mtsk_name, 0, 0, 0))){
              printf("CANNOTINIT MASTER TASK\n");
              exit(1);      }

       printf("MASTER TASK STARTS THE ONESHOT TIMER\n");
       actual_period = start_rt_timer(nano2count(10000));
printf("actual_period = %lld\n", actual_period);
rt_make_soft_real_time();
       printf("MASTER TASK MAKES ITSELF PERIODIC \n");
       rt_task_make_periodic(mtsk, rt_get_time()+ nano2count(3000),
nano2count(1000000));

        if(h=='a')
        {
//======================================================
//==   DEMO(LQG)
        while(1)
        {

m1=comedi_data_read(it,in_subdev,in_chan_1,in_range,aref,&in_data_1);
m2=comedi_data_read(it,in_subdev,in_chan_2,in_range,aref,&in_data_2);
m3=comedi_data_read(it,in_subdev,in_chan_3,in_range,aref,&in_data_3);
        //Hall-effect sensors
m4=comedi_data_read(it,in_subdev,in_chan_4,in_range,aref,&in_data_4);
m5=comedi_data_read(it,in_subdev,in_chan_5,in_range,aref,&in_data_5);
m6=comedi_data_read(it,in_subdev,in_chan_6,in_range,aref,&in_data_6);
m7=comedi_data_read(it,in_subdev,in_chan_7,in_range,aref,&in_data_7);
m8=comedi_data_read(it,in_subdev,in_chan_8,in_range,aref,&in_data_8);
m9=comedi_data_read(it,in_subdev,in_chan_9,in_range,aref,&in_data_9);

        //Hall-effect sensors data
 va1 = comedi_to_phys(in_data_4, in_range_ptr, in_maxdata); // y-sensor a
 vb1 = comedi_to_phys(in_data_5, in_range_ptr, in_maxdata); // x-sensor a
 vc1 = comedi_to_phys(in_data_6, in_range_ptr, in_maxdata); // x-sensor b
 vd2 = comedi_to_phys(in_data_7, in_range_ptr, in_maxdata); // y-sensor b
 va2 = comedi_to_phys(in_data_9, in_range_ptr, in_maxdata); // y-sensor c
 vb2 = comedi_to_phys(in_data_8, in_range_ptr, in_maxdata); // x-sensor c

//vo00=vb2; vo10=vd2; vo20=vb1; vo30=va1;
//digital filters
vi00=vb2; vi10=vd2; vi20=vb1; vi30=va1;
//vo00=2*0.9185*0.997*vo01-(0.9185*0.9185)*vo02+(vi01/82.2825);
vo00=1.792*vo01-0.8047*vo02+(0.09318*vi00-0.1733*vi01+0.09318*vi02);  //notch
//vo00=2*0.9995*0.9372*vo01-(0.9372*0.9372)*vo02+(vi11/205.22);

//vo10=2*0.9185*0.997*vo11-(0.9185*0.9185)*vo12+(vi11/82.2825);
vo10=1.792*vo11-0.8047*vo12+(0.09318*vi10-0.1733*vi11+0.09318*vi12);  //notch
//vo10=2*0.90575*0.9989*vo11-(0.90575*0.90575)*vo12+(vi11/91.8);
//vo10=2*0.9995*0.9372*vo11-(0.9372*0.9372)*vo12+(vi11/205.22);

//vo30=2*0.9185*0.997*vo31-(0.9185*0.9185)*vo32+(vi31/82.2825);
//vo30=2*0.9995*0.9372*vo31-(0.9372*0.9372)*vo32+(vi31/205.22);
vo30=1.792*vo31-0.8047*vo32+(0.09318*vi30-0.1733*vi31+0.09318*vi32);  //notch
                ya1 = kya*vo30 + yaoffset;
                xa1 = kxa*vo20 + xaoffset;
                xb1 = kxb*vc1 + xboffset; //changed
                yb1 = kyb*vo10 + yboffset; //changed
                yc1 = kyc*va2 + ycoffset;
                xc1 = kxc*vo00 + xcoffset;

                z1 = comedi_to_phys(in_data_1, in_range_ptr, in_maxdata);
                z_pos1 = 50e-3+z1*1.01833e-2;
                z_pos10 = z_pos1;
                z_pos1 = z_pos1-z_pos10;
                z2 = comedi_to_phys(in_data_2, in_range_ptr, in_maxdata);
                z_pos2 = 50e-3+z2*1.01833e-2;
                z_pos20 = z_pos2;
                z_pos2 = z_pos2-z_pos20;
                z3 = comedi_to_phys(in_data_3, in_range_ptr, in_maxdata);
                z_pos3 = 50e-3+z3*1.01833e-2;
                z_pos30 = z_pos3;
                z_pos3 = z_pos3-z_pos30;
/*POSITIONS*/
              //xA_pos=(1/123.25)*(xa1);
              //  xB_pos=(1/123.25)*asin((1/1)*xb1);
               yA_pos=(1/123.25)*(ya1);
                yB_pos=(1/123.25)*(yb1);
                xC_pos=(1/123.25)*(xc1);
              //  yC_pos=(1/123.25)*(yc1);

             xr = xC_pos;
             yr = yB_pos;  //YC good too
             rr = atan((yB_pos - yA_pos)/0.06438);
       //     rr = ((xC_pos - xA_pos)/0.0635);
       //     rr = (yA_pos - yC_pos)/10;
         t++;

//GLSDC_3();
  EKF_2();
             er0x = xc-xr;/*errors calc*/
                er0y = yc-yr;
                er0r = rc-rr;
                er0z = zc-zr10;
                er0s = sc-sr;
                er0t = tc-tr;

             ier1x=ier0x+0.00125*er0x;  //integrators
             ier1y=ier0y+0.00125*er0y;
             ier1r=ier0r+0.00125*er0r;

/*Lead-lag controls*/
       u0x = 200;  //      u0y=0;
//     u0x = 1.2423*u1x-0.2423*u2x+250000*(er0x-1.9773*er1x+0.977428*er2x);
//     u0y = 1.2423*u1y-0.2423*u2y+500000*(er0y-1.9773*er1y+0.977428*er2y);

//     u0y = 1.3141*u1y-0.3141*u2y+100000*(er0y-1.9773*er1y+0.977428*er2y);
//450<gain<1100
//     u0r = 1.2741*u1r-0.2741*u2r+1000*(er0r-1.8966*er1r+0.977428*er2r);

       //Moment of inertia
             fA = 0.5*u0y;
             fB = 0.5*u0y; fC = u0x;
 //        fA =-0.45*u0x+0.5*u0y+12.887*u0r; //-0.488*u0y;
  //       fB = 0.45*u0x+0.5*u0y-12.887*u0r; fC = u0x;//0.512*u0y;

//save data
//     if(t>8200 && t<16200){       fprintf(fp1, "%f \n",rr);}
```

```
        if(t>8200 && t<16200){       fprintf(fp1, "%f \n",yr*1000);}
     if(t>8200 && t<16200){fprintf(fp2, "%f  %f  %f\n",yr*1000,yc*1000, rr);}

//steps
       if(t>10 && t<10000){yc=0.00080;  xr=xc; rr=rc;}
       else if(t>=10000 && t<10060){ yc=0.00083;  xr=xc; rr=rc;}
       else if(t>=10060 && t<16200){ yc=0.00081;  xr=xc; rr=rc;}

  //         if(t>10 && t<10000){xc=-0.00100;  yr=yc; rr=rc;}
   //  else if(t>=10000 && t<16200){ xc=-0.001100;  yr=yc; rr=rc;}
   //        if(t>10 && t<10000){rc=0.017; xr=xc; yr=yc;}
   //  else if(t>=10000 && t<16200){ rc=0.018; xr=xc; yr=yc;}

//vertical (DQ Decomposition)...
       i_Ad = 0; i_Bd = 0;  i_Cd = 0;

//horizontal
 /*LQG*/
i_Aq =-160.2*er0x+254*er0y- 92*er0r-0*ier0x+6372*ier0y-0*ier0r-
3.46*x0_est+4.74*y0_est-1.18*r0_est;
i_Bq
=158.3*er0x+264.7*er0y+88.2*er0r+0*ier0x+6629*ier0y+0*ier0r+3.44*x0_est+5*y0_e
st+1.13*r0_est;
i_Cq =359.3*er0x -1.9*er0y-71.7*er0r+7949*ier0x-59.5*ier0y-
2595*ier0r+7.7*x0_est-0.84*r0_est;

//           i_Aq =259.3*er0y+6501.8*ier0y+4.85*y0_est;
//           i_Bq =259.3*er0y+6501.8*ier0y+4.85*y0_est;   i_Cq =8;
//i_Aq =2;   i_Bq=2;   i_Cq =404*er0x+9312*ier0x+8.3*x0_est;
//i_Aq =fA*f21h;   i_Bq=fB*f21h;   i_Cq =fC*f21h;
                cos_x=cos(123.25*xr);
                cos_y=cos(123.25*yr);
                sin_x=sin(123.25*xr);
                sin_y=sin(123.25*yr);

                i_Aa = cos_y*i_Aq -sin_y*i_Ad;  //Current a & b
                i_Ba = cos_y*i_Bq -sin_y*i_Bd;
                i_Ca = cos_x*i_Cq -sin_x*i_Cd;
                i_Ab = sin_y*i_Aq +cos_y*i_Ad;
                i_Bb = sin_y*i_Bq +cos_y*i_Bd;
                i_Cb = sin_x*i_Cq +cos_x*i_Cd;

       if(t>16210)
        {i_Aa=0; i_Ab=0; i_Ba=0; i_Bb=0; i_Ca=0; i_Cb=0;}

                current_A_1 = i_Aa;                     //Final coil currents
                current_A_2 = 0.5*i_Aa+0.8660254037844*i_Ab;
                current_A_3 = -0.5*i_Aa+0.8660254037844*i_Ab;
                current_B_1 = i_Ba;
                current_B_2 = 0.5*i_Ba+0.8660254037844*i_Bb;
                current_B_3 = -0.5*i_Ba+0.8660254037844*i_Bb;
                current_C_1 = -i_Ca;
                current_C_2 = -0.5*i_Ca-0.8660254037844*i_Cb;
                current_C_3 = 0.5*i_Ca-0.8660254037844*i_Cb;

     out_data_1 = comedi_from_phys(current_A_1, out_range_ptr, out_maxdata);
     out_data_2 = comedi_from_phys(current_A_2, out_range_ptr, out_maxdata);
     out_data_3 = comedi_from_phys(current_A_3, out_range_ptr, out_maxdata);
     out_data_4 = comedi_from_phys(current_B_1, out_range_ptr, out_maxdata);
     out_data_5 = comedi_from_phys(current_B_2, out_range_ptr, out_maxdata);
```

```
       out_data_6 = comedi_from_phys(current_B_3, out_range_ptr, out_maxdata);
       out_data_7 = comedi_from_phys(current_C_1, out_range_ptr, out_maxdata);
       out_data_8 = comedi_from_phys(current_C_2, out_range_ptr, out_maxdata);
       out_data_9 = comedi_from_phys(current_C_3, out_range_ptr, out_maxdata);
       out_data_10= comedi_from_phys(sigtest, out_range_ptr, out_maxdata);

  comedi_data_write(it2, out_subdev, out_chan_1, out_range, aref, out_data_1);
  comedi_data_write(it2, out_subdev, out_chan_2, out_range, aref, out_data_2);
  comedi_data_write(it2, out_subdev, out_chan_3, out_range, aref, out_data_3);
  comedi_data_write(it2, out_subdev, out_chan_4, out_range, aref, out_data_4);
  comedi_data_write(it2, out_subdev, out_chan_5, out_range, aref, out_data_5);
  comedi_data_write(it2, out_subdev, out_chan_6, out_range, aref, out_data_6);
  comedi_data_write(it2, out_subdev, out_chan_7, out_range, aref, out_data_7);
  comedi_data_write(it2, out_subdev, out_chan_8, out_range, aref, out_data_8);
  comedi_data_write(it2, out_subdev, out_chan_9, out_range, aref, out_data_9);
  comedi_dio_write(it, out_subdev, out_chan_10, sigtest);
        rt_task_wait_period();

                if(t>16210)
                       { break;}
               xrpre = xr;
             u2x=u1x; u1x=u0x; er2x=er1x; er1x=er0x;
             u2y=u1y; u1y=u0y; er2y=er1y; er1y=er0y;
                u2r=u1r; u1r=u0r; er2r=er1r; er1r=er0r;
                vo02 = vo01; vo01 = vo00; vi02 = vi01; vi01 = vi00;
                vo12 = vo11; vo11 = vo10; vi12 = vi11; vi11 = vi10;
                vo22 = vo21; vo21 = vo20; vi22 = vi21; vi21 = vi20;
                vo32 = vo31; vo31 = vo30; vi32 = vi31; vi31 = vi30;
             ier0x=ier1x; ier0y=ier1y; ier0r=ier1r;
        } //while loop close
        } // if close

       else if(h=='n')
       {     goto answer_no;}
       goto begin;

answer_no:
       out_data_1 = (lsampl_t)2048;
       out_data_2 = (lsampl_t)2048;
       out_data_3 = (lsampl_t)2048;
       out_data_4 = (lsampl_t)2048;
       out_data_5 = (lsampl_t)2048;
       out_data_6 = (lsampl_t)2048;
       out_data_7 = (lsampl_t)2048;
       out_data_8 = (lsampl_t)2048;
       out_data_9 = (lsampl_t)2048;
       out_data_10= (lsampl_t)2048;

       rt_make_soft_real_time();
//     printf("MASTER TASK YIELDS ITSELF\n");
       rt_task_yield();
//     printf("MASTER TASK YIELD THE PERIODIC TIMER\n");
       stop_rt_timer();
//     printf("MASTER TASK DELETES ITSELF\n");
       rt_task_delete(mtsk);

       comedi_close(it);
       comedi_close(it2);
       fclose(fp1); fclose(fp2); fclose(fp3);
       printf("END MASTER TASK\n");
       printf("%d\n", out_data_1);
       return 0;
}//end of main

void GLSDC_2(void)
{
// Steady-state 6-axis Hybrid filter
Kh[0][0]=1.001;   Kh[1][1]=1.001;    Kh[2][2]=1.001;
Kh[3][3]=1.001;   Kh[4][4]=1.001;    Kh[5][5]=1.001;
Kh[6][0]=-2.6e-12; Kh[7][1]=-2.6e-12;  Kh[8][2]=-2.6e-9;
Kh[9][3]=-2.6e-13; Kh[10][4]=-2.6e-13; Kh[11][5]=-2.6e-13;
Kh[12][0]=-4.11e-9;  Kh[13][1]=-4.11e-9;  Kh[14][2]=-4.11e-6;
Kh[15][3]=-4.11e-10; Kh[16][4]=-4.11e-10; Kh[17][5]=-4.11e-10;

Kk[0][0]=0.0158;  Kk[1][1]=0.0158;  Kk[2][2]=0.0225;
Kk[3][3]=0.0113;  Kk[4][4]=0.0113;  Kk[5][5]=0.0113;
Kk[6][0]=0.124162; Kk[7][1]=0.124162; Kk[8][2]=0.32431;
Kk[9][3]=0.1062; Kk[10][4]=0.1062; Kk[11][5]=0.1062;
Kk[12][0]=0.9842; Kk[13][1]=0.9842; Kk[14][2]=1.132;
Kk[15][3]=0.8421; Kk[16][4]=0.8421; Kk[17][5]=0.8421;

       ytildek[0]=xr;      ytildek[1]=yr;      ytildek[2]=rr;
       ytildek[3]=xr;      ytildek[4]=yr;      ytildek[5]=rr;

       //Kalman d=1, Hinf d=0.0
       for(i=0;i<17;i++){
             xknew[i]=0;
             for(j=0;j<5;j++){
                    MK[i][j]=0.9*Kk[i][j]+(1-0.9)*Kh[i][j];
                    xknew[i]+=MK[i][j]*(ytildek[j]-xk[j]);
                           }
             xknew[i]=xknew[i]+xk[i];
             }
             for(i=0;i<17;i++)
                    {xk[i]=xknew[i];}

                    xr=xk[0]; yr=xk[1]; rr=xk[2];
                  zr10=xk[3]; sr=xk[4]; tr=xk[5];
                    x0_est=xk[6]; y0_est=xk[7]; r0_est=xk[8];
                    z0_est=xk[9]; s0_est=xk[10]; t0_est=xk[11];
}

void GLSDC_3(void)
{ //Recursive planar hybrid filter
Kh[0][0]=1.0101;   Kh[1][1]=1.0101;    Kh[2][2]=1.0101;
Kh[3][0]=-2.6e-12; Kh[4][1]=-2.6e-12;  Kh[5][2]=-2.6e-11;
Kh[6][0]=-4.11e-9;Kh[7][1]=-4.11e-9; Kh[8][2]=-4.11e-8;

             ytildek[0]=xr;
             ytildek[1]=yr;
             ytildek[2]=rr;
       //gain matriX
             // Hk*Pk*Hk'+Rk=matA matA is symmetric
             for(i=0;i<3;i++){
                    for(j=i;j<3;j++){
                           matA[i][j]=Pk[i][j]+Rkk[i][j];
                           matA[j][i]=matA[i][j];
                    }
             }
```

```
              // solve for inverse
       det=matA[0][0]*matA[1][1]*matA[2][2]+matA[1][0]*matA[2][1]*matA[0][2]+m
atA[2][0]*matA[0][1]*matA[1][2]-matA[0][0]*matA[2][1]*matA[1][2]-
matA[2][0]*matA[1][1]*matA[0][2]-matA[1][0]*matA[0][1]*matA[2][2];

              invA[0][0]=(matA[1][1]*matA[2][2]-matA[1][2]*matA[2][1])/det;
              invA[0][1]=(matA[0][2]*matA[2][1]-matA[0][1]*matA[2][2])/det;
              invA[0][2]=(matA[0][1]*matA[1][2]-matA[0][2]*matA[1][1])/det;
              invA[1][1]=(matA[0][0]*matA[2][2]-matA[0][2]*matA[2][0])/det;
              invA[1][2]=(matA[0][2]*matA[1][0]-matA[0][0]*matA[1][2])/det;
              invA[2][2]=(matA[0][0]*matA[1][1]-matA[0][1]*matA[1][0])/det;
                     invA[1][0]=invA[0][1];
                     invA[2][0]=invA[0][2];
                     invA[2][1]=invA[1][2];
                     // Pk*Hk' is the left 3 rows of Pk
                     // m_times_m(Pk, 9,3,invA,3,3,Kk);
                     for(i=0;i<9;i++){
                            for(j=0;j<3;j++){
                                   Kk[i][j]=0;
                                   for(k=0;k<3;k++){
                                          Kk[i][j]+=Pk[i][k]*invA[k][j];
                                   }
                            }
                     }
              // update x
              //Kalman d=1, Hinf d=0.0
                     for(i=0;i<9;i++){
                            xknew[i]=0;
                            for(j=0;j<3;j++){
                                   MK[i][j]=1.0*Kk[i][j]+(1-1.0)*Kh[i][j];
                                   xknew[i]+=MK[i][j]*(ytildek[j]-xk[j]);
                            }
                                   xknew[i]=xknew[i]+xk[i];
                     }
                            for(i=0;i<9;i++)
                                   xk[i]=xknew[i];

                     // update Pk, Ck is symmetric matrix
                            for(i=0;i<9;i++){
                                   for(j=i;j<9;j++){
                                          matC[i][j]=0;
                                          for(k=0;k<3;k++){
                                                 matC[i][j]+=Kk[i][k]*Pk[k][j];
                                          }
                                          matC[j][i]=matC[i][j];
                                   }
                            }
                            for(i=0;i<9;i++){
                                   for(j=i;j<9;j++){
                                          Pk[i][j]=Pk[i][j]-matC[i][j];
                                          Pk[j][i]=Pk[i][j];
                                   }
                            }
                     // propagation xk
                            // m_times_v(Phik,9,9,xk,9,xknew);
                            for(i=0;i<9;i++)
                                   xknew[i]=0;

                            xknew[0]=xk[0]+dt*xk[3]+dt22*xk[6];
                            xknew[1]=xk[1]+dt*xk[4]+dt22*xk[7];
```

```
                xknew[2]=xk[2]+dt*xk[5]+dt22*xk[8];
                xknew[3]=xk[3]+dt*xk[6];
                xknew[4]=xk[4]+dt*xk[7];
                xknew[5]=xk[5]+dt*xk[8];
                xknew[6]=xk[6];
                xknew[7]=xk[7];
                xknew[8]=xk[8];

                for(i=0;i<9;i++)
                    xk[i]=xknew[i];

            //propagation PK
                // Phik*PK*Phik'=Pknew
        for(i=0;i<3;i++){
            for(j=i;j<3;j++){
        Pknew[i][j]=Pk[i][j]+2*dt*Pk[i][j+3]+dt2*Pk[i][j+6]+dt3*Pk[i+3][j
        +6]+dt4/4*Pk[i+6][j+6];
                    }
                }
        for(i=0;i<3;i++){
            for(j=i+3;j<6;j++){
    Pknew[i][j]=Pk[i][j]+dt*Pk[i][j+3]+1.5*dt2*Pk[i+3][j+3]+dt*Pk[i][j+3]+d
t3/2*Pk[i+6][j+3];
                    }
                }
                Pknew[1][3]=Pknew[0][4];
                Pknew[2][3]=Pknew[0][5];
                Pknew[2][4]=Pknew[1][5];
            for(i=0;i<3;i++){
                for(j=i+6;j<9;j++){

    Pknew[i][j]=Pk[i][j]+dt*Pk[i+3][j]+dt2*Pk[i+6][j];
                    }
                }
                Pknew[1][6]=Pknew[0][7];
                Pknew[2][6]=Pknew[0][8];
                Pknew[2][7]=Pknew[1][8];
            for(i=3;i<6;i++){
                for(j=i;j<6;j++){

    Pknew[i][j]=Pk[i][j]+dt*Pk[i][j+3]+dt2*Pk[i+6][j];
                    }
                }
                for(i=3;i<6;i++){
                    for(j=i+3;j<9;j++){
                        Pknew[i][j]=Pk[i][j]+dt*Pk[i+3][j];
                    }
                }
                Pknew[4][6]=Pknew[3][7];
                Pknew[5][6]=Pknew[3][8];
                Pknew[5][7]=Pknew[4][8];
                for(i=6;i<9;i++){
                    for(j=i;j<9;j++){
                        Pknew[i][j]=Pk[i][j];
                    }
                }
                for(i=0;i<9;i++){
                    for(j=i;j<9;j++){
                        Pknew[j][i]=Pknew[i][j];
                    }
```

```
                           }

                           for(i=0;i<9;i++){
                                 for(j=i;j<9;j++){

       Pk[i][j]=Pknew[i][j]+UpskQkUpskT[i][j];
                                        Pk[i][j]=Pk[i][j];
                                 }
                           }
                           xr=xk[0]; yr=xk[1]; rr=xk[2];
                           x0_est=xk[3]; y0_est=xk[4]; r0_est=xk[5];
}

void sensor_test(void)
{      }

void terminate_normally(int signo)
{      fflush(stdin);

       if(signo==SIGINT ||signo==SIGTERM)
       {     printf("Terminating the program normally\n");
             out_data_1 = (lsampl_t)2048;
             out_data_2 = (lsampl_t)2048;
             out_data_3 = (lsampl_t)2048;
             out_data_4 = (lsampl_t)2048;
             out_data_5 = (lsampl_t)2048;
             out_data_6 = (lsampl_t)2048;
             out_data_7 = (lsampl_t)2048;
             out_data_8 = (lsampl_t)2048;
             out_data_9 = (lsampl_t)2048;
             out_data_10= (lsampl_t)2048;

             for(i=0;i<20;i++){
comedi_data_write(it2, out_subdev, out_chan_1, out_range, aref, out_data_1);
comedi_data_write(it2, out_subdev, out_chan_2, out_range, aref, out_data_2);
comedi_data_write(it2, out_subdev, out_chan_3, out_range, aref, out_data_3);
comedi_data_write(it2, out_subdev, out_chan_4, out_range, aref, out_data_4);
comedi_data_write(it2, out_subdev, out_chan_5, out_range, aref, out_data_5);
comedi_data_write(it2, out_subdev, out_chan_6, out_range, aref, out_data_6);
comedi_data_write(it2, out_subdev, out_chan_7, out_range, aref, out_data_7);
comedi_data_write(it2, out_subdev, out_chan_8, out_range, aref, out_data_8);
comedi_data_write(it2, out_subdev, out_chan_9, out_range, aref, out_data_9);
comedi_dio_write(it2, out_subdev, out_chan_10,16);
       rt_task_wait_period();
             }
             rt_make_soft_real_time();

             printf("MASTER TASK YIELDS ITSELF\n");
             rt_task_yield();
             printf("MASTER TASK STOPS THE PERIODIC TIMER\n");
             stop_rt_timer();
             printf("MASTER TASK DELETES ITSELF\n");
             rt_task_delete(mtsk);
             comedi_close(it);
             comedi_close(it2);
             printf("END MASTER TASK\n");
       }
       exit(0);}
```

VITA

The author, Jose Christian Silva-Rivas, was born in the city of Puebla, Mexico. He moved to College Station, Texas, when he was 12 years old. He graduated from The University of Texas at Austin with the Bachelor of Science degree in mechanical engineering and a minor in Spanish in May 2009. He began his graduate research at Texas A&M University in August 2009 where he graduated with a Master of Science degree in mechanical engineering in December 2011.

His permanent address is:

Department of Mechanical Engineering
3123 TAMU
College Station, Texas 77843

His e-mail address is:

j_c_s_r@hotmail.com