DEVELOPING A MAGNETICALLY SUSPENDED DISC SYSTEM FOR

INDUSTRIAL APPLICATIONS, WITH APPLICATION

ON RING SPINNING

Except where reference is made to the work of others, the work described in this dissertation is my own or was done in collaboration with my advisory committee. This dissertation does not include proprietary or classified information.

_____

Sherif M. Abuelenin

Certificate of Approval:

_____	_____
Yehia El Mogahzy Professor Textile Engineering	Faissal Abdel-Hady, Chair Research Assistant Professor Textile Engineering
_____	_____
David G. Beale Professor Mechanical Engineering	Lewis Slaten Associate Professor Consumer Affairs

_____

Stephen L. McFarland
Acting Dean
Graduate School

DEVELOPING A MAGNETICALLY SUSPENDED DISC SYSTEM FOR INDUSTRIAL APPLICATIONS, WITH APPLICATION ON RING SPINNING

Sherif Mohamed Abuelenin

A Dissertation

Submitted to

the Graduate Faculty of

Auburn University

in Partial Fulfillment of the

Requirements for the

Degree of

Doctor of Philosophy

Auburn, Alabama
August 8, 2005

DEVELOPING A MAGNETICALLY SUSPENDED DISC SYSTEM FOR

INDUSTRIAL APPLICATIONS, WITH APPLICATION

ON RING SPINNING

Sherif Abuelenin

_____

Signature of Author

_____

Date of Graduation

DISSERTATION ABSTRACT

DEVELOPING A MAGNETICALLY SUSPENDED DISC SYSTEM FOR INDUSTRIAL APPLICATIONS, WITH APPLICATION ON RING SPINNING

Sherif Abuelenin

Doctor of Philosophy, August 8, 2005
(M.S. Tuskegee University, 2002)
(B.S. Suez Canal University, 1999)

192 Typed Pages

Directed by Faissal Abdel-Hady

A Magnetically suspended and lightweight disc (rotor) system is designed; the rotor can spin freely inside the stator. The stator is equipped with four magnetic actuators that always keep the rotor in its central position. A controller is designed for the operation of the suspended disc. As an application the suspended rotor system is used to introduce a new spinning concept termed "magnetic spinning." The rotor in this configuration replaces the ring and traveler in the traditional spinning system. System concept and control analysis are discussed in this research. We also introduce a numerical simulation of the process; the model used in the simulation was built using Simulink®. Simulink® is a software package for modeling, simulating, and analyzing dynamic systems. The model

takes into account all the mechanical and electrical parts of the system, and the parameters that affect the operation. The simulation is used to analyze the yarn tension and ballooning.

ACKNOWLEDGMENT

The author would like to thank Dr. Faissal Abdel-Hady for his idea of this research and his support, and Dr. Yehia E. Elmogahzy for his assistance and help. Thanks also due to my committee members, and to my family.

Manual style: Textile research journal

Computer software: Microsoft Word 2003

TABLE OF CONTENTS

LIST OF FIGURES ……..……..... ix

1. INTRODUCTION ……. 1

2. REVIEW OF LITERATURE …… 7

3. SYSTEM MODELLING ……..…… 24

4. CONTROL ENGINEERING ASPECTS …… 45

5. SIMULATING THE SYSTEM ……...…… 69

6. EXPERIMENTAL WORK ……. 91

7. RESULTS AND DISCUSSION …… 109

8. CONCLUSIONS ……...… 122

BIBLIOGRAPHY ……..… 126

APPENDICES ……... 128

1. NUMERICAL CALCULATIONS ……..……..……... 128
2. TEST THE LINEARIZED MODEL FOR CONTROLLABILITY …..….. 131
3. SIMUILINK MODEL BASIC BLOCKS ……..…….. 132
4. ASSEMBLY PROGRAM ……....….. 138
5. M-FILE FOR CALCULATING CLAMPED BEAM FREQUENCY …… 179

LIST OF FIGURES

1.1 Two Basic Categories of Production System ……..……. 2

1.2 Forces Applied on Yarn during Ring-Spinning …….…….. 4

2.1 Magnetic circuit for a horseshoe …… 10

2.2 Magnetic system composed of four discrete horseshoes ……..……. 11

2.3 Block Diagram of PID controller ……... 14

2.4 The basic principle of spinning ……..…… 16

2.5 Current offerings in spinning machines ……...…….. 17

2.6 Sketch of ring spinning ……..…… 22

3.1 Magnetic-Ring Spinning …… 24

3.2 Schematic cross-section view of the magnetic system ……...……...… 26

3.3 Solution of Equations (2) and (3) ……..….. 28

3.4 The Two Gaps (g_1 and g_2) …….……. 29

3.5 Magnetic spinning block diagram ……..…… 32

3.6.a Diagram of the Assembled Design of the Magnetic Ring Spinning Machine ...…. 35

3.6.b Cross-Section Diagram of the Assembled Design of the Magnetic Ring Spinning Machine ……...……. 36

3.7 Disassembled design of the magnetic ring spinning machine ……...…… 37

3.8 Detailed components of the design …… 38

3.9 The finite elements model meshed with air cylinder removed ……..…… 39

3.10 Variation of magnetic force with excitation current for different air gaps ……..…. 41

3.11 Axial restoring force …… 41

3.12 Top view of flux intensity pattern due to permanent magnets only …… 42

3.13 Top view of flux intensity pattern due to permanent magnets and control current = 0.5 A ……. 42

3.14 Top view of flux intensity pattern due to permanent magnets and control current = 0.75 A ……..……. 43

3.15 Top view of flux intensity pattern due to permanent magnets and control current = 1.5 A ……..… 43

4.1 Block Diagram of the System ……...……. 46

4.2 Arrangement of Displacement Sensors around the Rotor ……...……. 48

4.3 Diagram of the sensor reading circuit …… 49

4.4 A Discrete PID controller ……..…. 51

4.5 Block Diagram of the Fuzzy Controller for Parameter Adaptation Configuration ……... 52

4.6 Block Diagram of the Fuzzy Controller for PID Output Correction Configuration ……... 53

4.7 Feedback control system ……...……. 54

4.8.a PID firmware implementation ……. 55

4.8.b Main PID Routine (PIDMain) ……...……. 56

4.8.c PID Interrupt Routine (PIDInterrupt) …….. 57

4.9 Fuzzytech® project editor …….. 63

4.10.a Membership functions of the input variable ‘err’ ……. 64

4.10.b Membership functions of the input variable ‘der’ ……..……. 64

4.10.c Membership functions of the output variable ……... 64

4.11 Rules of fuzzy controller ……. 65

4.12 Flowchart of the fuzzy controller firmware ……. 66

5.1 The System Simulink Model ……..…… 70

5.2 The Rotor Model …… 71

5.3.a The Simulink Model of the Yarn ……..…….. 72

5.3.b The Simulated Yarn ……..…….. 73

5.4 Air-Drag Model ……..…… 74

5.5 Discrete PID Controller ……...….. 75

5.6 Simulink Model of Fuzzy Controller for Parameter Adaptation Configuration ……..…. 77

5.7.a Membership functions of fuzzy input variable ‘error’, option 1 ……..…….. 77

5.7.b Membership functions of fuzzy input variable ‘rate’, option 1 ……..…….. 78

5.7.c Membership functions of fuzzy output variable ‘2kd’, option 1 ……..…….. 78

5.8.a Fuzzy controller rules, option 1 ……...…….. 78

5.8.b Fuzzy controller surface, option 1 ……...…….. 79

5.9 Simulink Model of Fuzzy Controller for PID Output Correction Configuration ……..…. 79

5.10.a Membership functions of fuzzy input variable ‘error’, option 2 ……..…….. 80

5.10.b Membership functions of fuzzy input variable ‘rate’, option 2 ……..…….. 80

5.10.c Membership functions of fuzzy output variable ‘2kd’, option 2 ……..…….. 80

5.11.a Fuzzy controller rules, option 2 ……..…….. 81

5.11.a Fuzzy controller surface, option 2 ……..…….. 81

5.12 ISIS Microcontroller circuit ……. 83

5.13 Analog-to-Digital Converter …… 86

5.14 X-NOR gates …… 87

5.15 Transistor switching circuit …….. 89

6.1 Magnetic Ball Levitation ……...…… 92

6.2 Calculating K ……. 96

6.3 Lead-lag controller ……. 97

6.4 Oscilloscope output …… 98

6.5 Matlab step responses for the system and its model …….…….... 98

6.6 PIC16F877 evaluation board ……..……. 101

6.7 Circuit diagram for position sensing …… 103

6.8 Sensor reading circuit …….....……... 104

6.9 Setup to test the sensor reading circuit …… 105

6.10 Oscilloscope reading ……..……. 107

6.11 Prototype …… 107

7.1.a The simulated yarn at the beginning of the simulation, top view ……..... 110

7.1.b The simulated yarn at the beginning of the simulation, side view ……..……. 110

7.1.c The simulated yarn at the beginning of the simulation, 3D view ……..…... 111

7.2.a The simulated yarn after ballooning, top view ……..... 112

7.2.b The simulated yarn after ballooning, side view ……...…. 112

7.2.c The simulated yarn after ballooning, 3D view …….……. 113

7.3 ring speed in rpm ……..…… 114

7.4 Tension of the yarn in cN …….……..…. 115

7.5 Tension as a function of the speed ……..……. 115

7.6.a Change in the ring position with time ……...…… 116

7.6.b Change in the ring position with time, long period …….. 117

7.7 Simulink window displays the used fuzzy rules during operation ……..…… 118

7.8 Inserted noise signal ……. 118

7.9 Fuzzy Controller for parameter adaptation response (top) compared with PID response (bottom) ……... 119

7.10 Fuzzy Controller option 2 (top) compared with PID response (bottom) ……... 120

7.11 response Fuzzy/PID controller vs PID alone, with White Noise Applied ……. 121

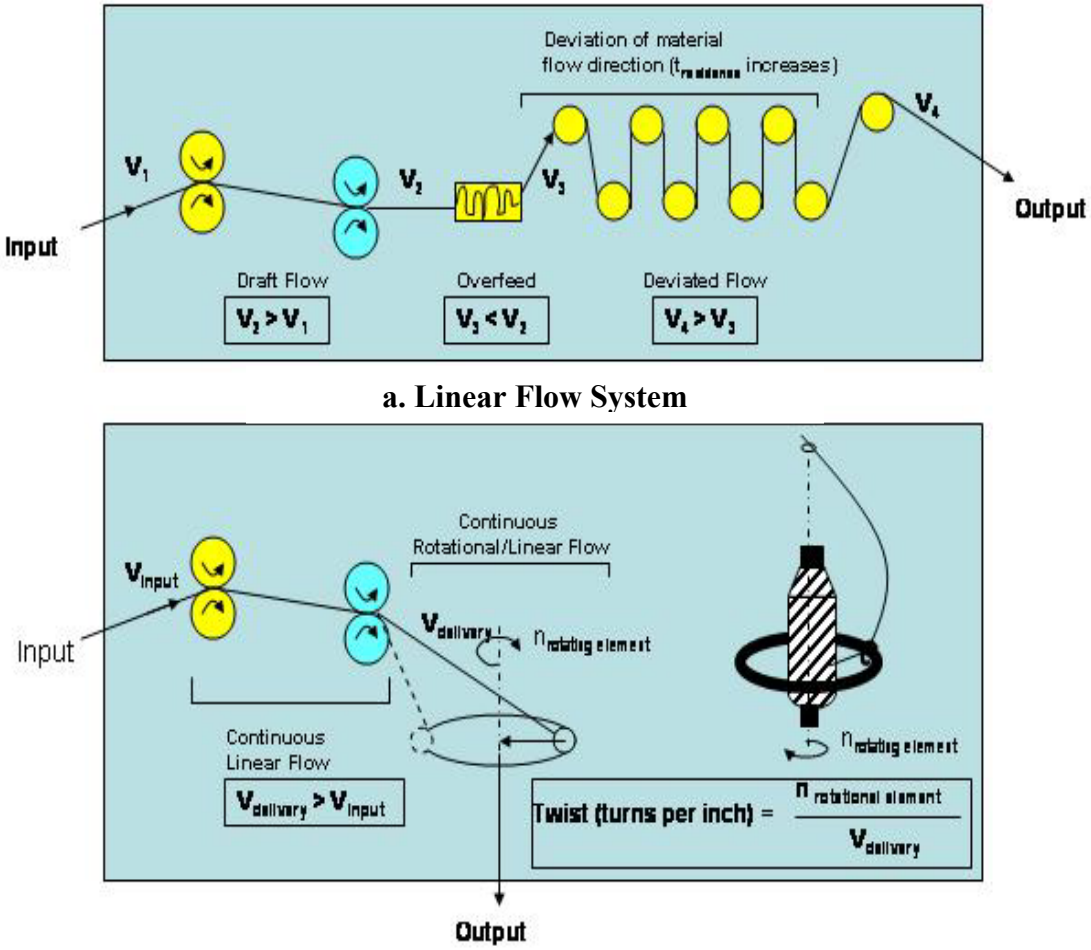
1. INTRODUCTION

In most engineering production systems, the production rate is largely determined by the operating speed of the system, which in most traditional systems, represents the linear speed of material flow through the production system. In a classic linear production system (Figure 1.1.a), the speed of material flow through the system is determined by the speeds of the various consecutive components of the system, possible deliberate overshoot in material flow (e.g. draft), possible deliberate overfeed (e.g. condensation or temporary stuffing to create buckling or texturing effects), and deviation of material flow direction for space requirement or for controlling material residence time during production (e.g. drying or cooling condition) [4].

When the production system consists of nonlinear material flow, the problem of maximizing the production speed becomes a complex engineering challenge. By nonlinear material flow, we mean some form of intentional deformation of material flow in a direction perpendicular (or at some angle) to the linear flow of material. In practice, this type of production systems is best illustrated by the yarn spinning process. In this unique production system, fibers entering the system initially flow in a linear mode to meet specific technological requirements such as drafting to reduce the thickness of the fiber strand down to the desired size. This linear flow is then followed by a twisting mechanism for consolidating the fibers into yarn. As described in the review of literature, the conventional twisting mechanism typically consists of a ring/traveler system. The

traveler is a c-shaped piece of metal that is dragged around a stationary metal ring by the yarn, which rotates by the rotation of a bobbin mounted on a rotating spindle. The amount of twist inserted in the yarn is determined by the simple equation:

$$Twist\,(tuns\ per\ inch) = \frac{n_{traveler}}{V_{delivery}} \qquad (1)$$



a. Linear Flow System

b. Nonlinear Flow System

Figure 1.1 Two Basic Categories of Production System [4]

According to equation 1, increasing the delivery speed or productivity of ring spinning is largely dependent on the traveler speed. In other words, any opportunity to increase productivity in this type of system must be through an increase in traveler speed. Since the principle of ring-spinning dictates a continuous ring-traveler contact, high traveler speed will result in traveler burning out because of the frictional heat initiated during traveler rotation. This point can be illustrated by considering the two main forces acting on the area of traveler/ring contact: the centrifugal force and the frictional force (see Figure 1.2.a). The centrifugal force can be expressed as follows:

$$CF = \frac{2m_t V_t^2}{d_{ring}} \qquad (2),$$

where m_t = traveler mass, V_t = traveler speed, and d_{ring} = ring diameter.

The normal force acting on the traveler during rotation may be approximated by the centrifugal force. The Amonton's Law of friction provides the following relationship between the frictional force, F, and the normal force, N, is [4]:

$$F = \mu_{ring/traveler}.N \cong \mu_{ring/traveler}.CF \cong \mu_{ring/traveler}.\frac{2m_t.V_t^2}{d_{ring}} \qquad (3)$$

Where $\mu_{ring/traveler}$ = the coefficient of friction between the ring and the traveler.

For a given ring/traveler system, the coefficient of friction, the traveler mass, and the ring diameter are normally constants. Accordingly, an increase in the traveler speed will result in an increase in the centrifugal force (or the normal force), and consequently an increase in the frictional force.

The reason that the traveler burns is that the mechanical friction described above results in kinetic energy, which is transformed into heat energy at high traveler speed.

The frictional heat may be described by the so-called thermal load capacity of the ring/traveler system; a traveler burns when the ring/traveler system reaches its limit of thermal load capacity.

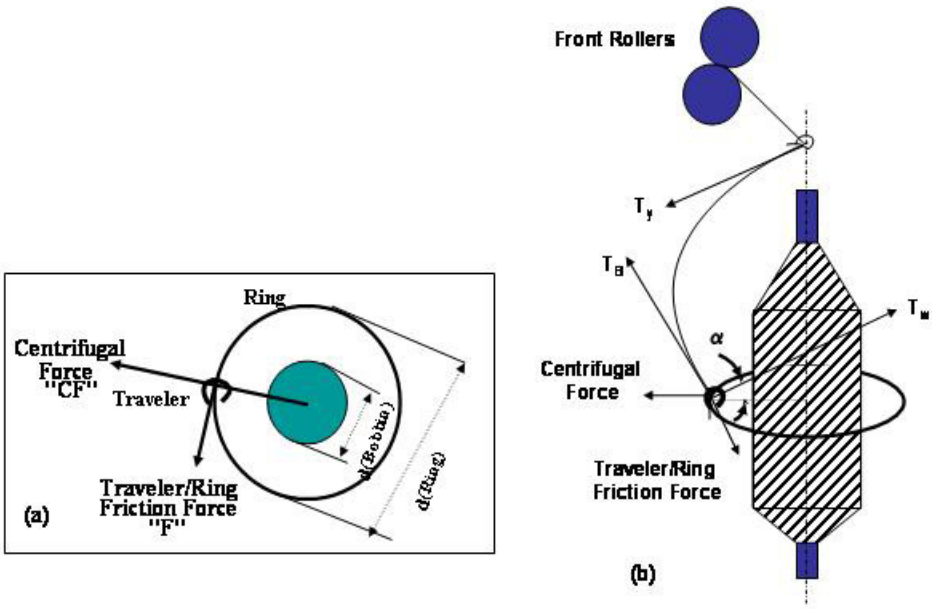


Figure 1.2 Forces Applied on Yarn during Ring-Spinning [4]

Another limitation to the increase of production speed associated with the above system is spinning tension T_y in Figure 1.2.b). This is defined as the tensile force applied on the yarn at the onset of twisting. The critical importance of this parameter lies in the fact that it contributes largely to both the quality of ring-spun yarn, and the spinning performance. Spinning tension results in a closed packing of fibers during twisting, this enhances yarn strength. Variation in spinning tension directly results in variation in yarn strength. Excessive tension or tension peaks may result in end breakage during spinning.

In fact, more than 80% of end breakage during ring spinning is believed to result from variation in spinning tension.

The relationship between traveler speed and spinning tension can be demonstrated by the following equation [4]:

$$T_y = \frac{\mu_{ring/traveler}}{\sin\alpha} \cdot \frac{m_t . V_t^2}{d_r} \qquad (4)$$

Where $\mu_{r/t}$ = the coefficient of friction between ring and traveler, α = the angle between yarn from traveler to bobbin and a straight horizontal line from traveler to spindle axis, m_t = traveler mass, V_t = traveler speed, and d_r = ring diameter. Note that the term ($m_t . V^2_t / d_r$) represents the centrifugal force discussed earlier.

The above equation clearly indicates that the increase in traveler speed will result in an increase in the centrifugal force, and consequently, an increase in the spinning tension. In practice, an increase in spinning tension above a certain critical limit will immediately result in end-breakage.

In order to appreciate the importance of spinning tension, one must consider the spinning geometry of ring spinning shown in Figure 1.2.b. As can be seen in this Figure, during twisting the fiber strand form a balloon shape controlled by different tension components applied on the yarn. The main tension components are: (i) the spinning tension or the tensile force applied on the yarn at the onset of twisting, (ii) the balloon tension or the tensile force applied on the yarn as it enters the ring/traveler zone, and (iii) the winding tension or the tensile force applied on the yarn during winding. These three tension components must act precisely to maintain the stability of the yarn balloon, which is a critical requirement for acceptable spinning performance. Among the three, spinning

tension is the most interactive component with the input fiber strand. This is because of its impact on the fibers being delivered from the nip of the front roller.

Although the present study aimed at dealing with maximizing the production performance of nonlinear production systems in general, it was particularly inspired by the complexity of the conventional ring-spinning system discussed above. In the following sections, we present an overview of a proposed alternative system which makes use of the rotational mechanism in a magnetic media to overcome the problems discussed above.

The objectives of this study are as follows:

- To design a complete magnetic rotational system that is friction free for the purpose of overcoming the limitations associated with traditional systems such as the ring/traveler system discussed above in reaching significantly higher speeds of rotation and consequently higher speeds of production
- To design and develop a software/hardware control system of the magnetic rotational system so that precise flow of material accommodating the new system can be achieved at minimum stresses.

The idea of Magnetic spinning is patented under the name "*A ring spinning system for making yarn having a magnetically elevated ring*", Patent, No. PCT/US03/30317.

2. REVIEW OF LITERATURE

2.1 Active Magnetic Levitation Principles

To understand the main idea of the research and the new designed system, it is necessary to give an introduction on the magnetic levitation principal that is mainly employed in realizing this system. Magnetic bearing systems are systems in which a rotor or a stationary object is suspended in magnetic field. Magnetic levitation of a rotating disk typically incorporates four or more electromagnets to levitate a ferromagnetic disk without contact, where levitation is accomplished through automatic control of the electromagnet coil currents. Position sensors are required to sense the position of the disk. The controller uses position sensor outputs to apply stiffness and damping forces to the rotor (the ring in our case) to achieve a desired dynamic response. In more complicated systems, the controller can automatically compensate for disk imbalances, and selectively damp disk resonant modes. More sophisticated controllers based on adaptive control techniques are also available for active vibration cancellation, automatic rotor balancing, and mechanical impedance synthesis. Besides the obvious benefits of eliminating friction, magnetic systems also allow some perhaps less obvious improvements in performance.

Magnetic systems are generally inherently open-loop unstable, which means that active feedback is required for the systems to have a stable operation. However, the requirement of feedback control brings flexibility into the dynamic response of the

systems. By changing controller gains or strategies, the systems can be made to have virtually any desired closed-loop characteristics.

Magnetic bearing systems are being increasingly used in industrial applications where minimum friction is required, or in harsh environments where traditional bearings and their associated lubrication systems are considered unacceptable [6]. Commercially available technology was pioneered by the French the early 1970's and is licensed by several American and Japanese firms [6]. Typical applications for active bearings include turbo-molecular vacuum pumps and gas pipeline centrifugal compressors, sealed pumps, and electric power-plant equipment [6]. Magnetic bearings are being developed to extend the life and reliability of rotating devices in a vacuum fluid filled environment (commonly an air gap), see [2, 6, 7], and references therein.

Bearings that also employ permanent magnets have been developed. A completely passive and contactless magnetic bearing that is stable in all six degrees of freedom (DOF) cannot be realized under normal conditions [5]. In practice, at least one axis has to be actively controlled by means of electromagnets. Earlier publications on magnetic-bearing wheels aimed at controlling one, two or five DOF actively [1, 3, 5].

The main components of a magnetic bearing system are:

- Rotor; is the suspended body, made from ferric material.
- Actuators to create the magnetic field.
- Sensors to sense the location of the rotor
- Control system that receives the position information from the sensors and controls the actuators signal to modulate their magnetic fields so that any shift of the rotor from its required position can be compensated for.

Magnetic bearings have several advantages over conventional ball bearings. Their main advantage is elimination of contact between bodies -which results in loss of energy on the form of heat, and wearing of the rotor/stator material- hence, a frictionless operation is achieved due to lack of contact between the rotor and the stator, which increases the lifetime of the machine components, reduces the energy losses, and requires less maintenance due to lack of lubrication. Other advantages include potential to provide high-speed operation, and active disturbance elimination, which makes magnetic bearings ideal candidates for several industrial applications such as, vacuum pumps, hard disk drives, high-speed centrifuges, high-speed flywheels, turbo-machinery, and high-precision positioning [2, 7, 9].

The idea of magnetic suspension was introduced as early as 1937. Magnetic bearings have been successfully incorporated into a number of engineering models, some of which were sufficiently engineered to be called flight prototypes for applications in spacecraft.

A motorized electromagnetic bearing was developed by Cambridge Thermionic Corporation in 1969. This unit has substantial load carrying (7 kg) and speed (1,200 Rad/sec) capability and was servoed axially but had a rather large continuous power requirement (40W). Subsequent work at "Goddard Space Flight Center" introduced permanent magnets into the suspension system to supply the steady state magnetic flux without spending power, and in a way that loads no longer had to be carried thru the structure from one end of the assembly to the other. The power situation was further improved by the introduction of the *virtually zero power* "VZP" controller by J. Lyman of Cambion, Inc. which controls the suspension to a position in which all of the steady

state loads are carried by the permanent magnets. With the advent of new permanent magnet materials, the weight efficiency rose substantially.

2.1.1 Flux Density and Force from Circuit Theory

This type of magnetic system is amenable to analysis via magnetic circuit theory. Assuming negligible leakage and fringing and neglecting the small reluctance of the iron parts of the flux path yields the following magnetic circuit:

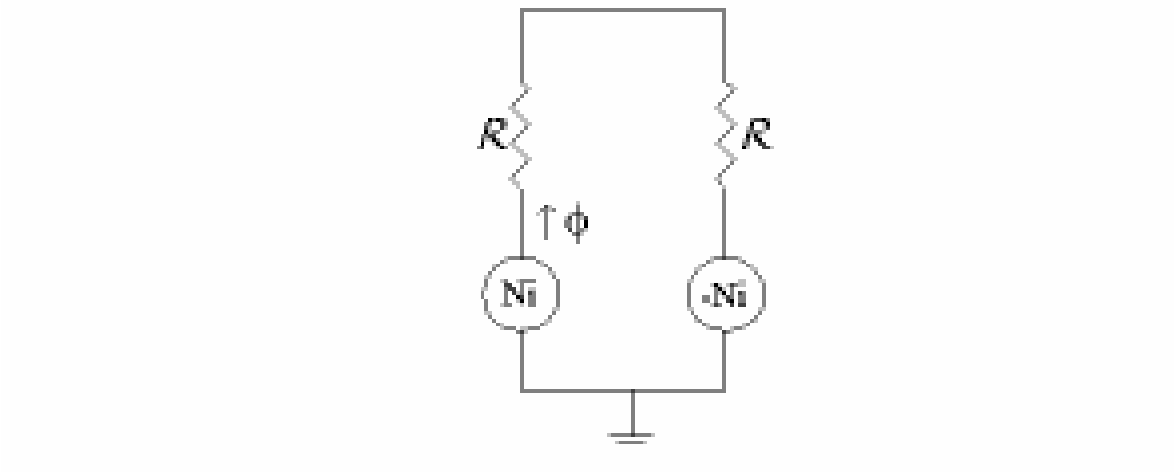


Figure 2.1 Magnetic circuit for a horseshoe

Figure 2.1 looks just like an electric circuit with batteries and resistors. However, instead of current, there is flux Φ, and instead of resistance, there is reluctance R. Rather than voltage sources, and there are magnetomotive force (MMF) sources of strength Ni and -Ni. The flux density, B, in the air- gap is the flux divided by the area of the air-gap.

We can now use *Maxwell's Stress Tensor* to calculate the force. In addition, the flux density obtainable in the gap should be limited by the flux that can get down the legs of the system.

2.1.2 Typical Magnetic System Geometry and Control

Typical magnetic system is composed of four of horseshoe-shaped electromagnets, as shown in Figure 2.2. The four magnets are arranged evenly around a rotor that is to be levitated. Each of the electromagnets can only produce a force that attracts the rotor to it. The four electromagnets must act in concert to produce a force of arbitrary magnitude and direction on the rotor. The coils may be arranged in the same plane of the rotor (as shown in Figure 2.2) or in a plane normal to that plane.

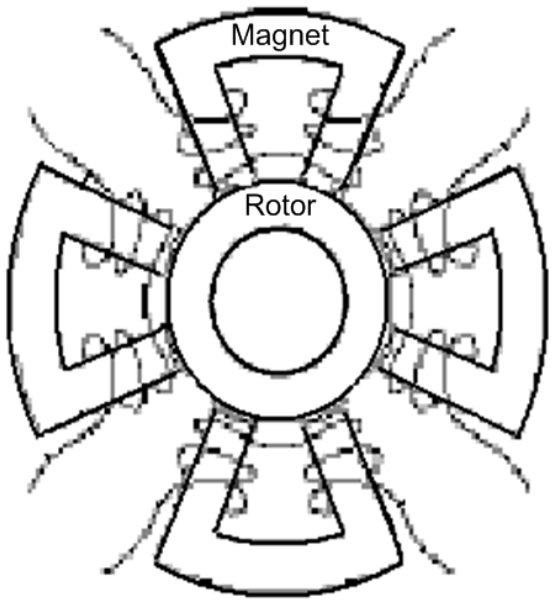


Figure 2.2 Magnetic system composed of four discrete horseshoes

There are *N* turns of wire around each leg of the system. So that the system behaves functionally like a system composed of discrete horseshoes, sets of adjacent coils are connected together in reverse series.

2.2 Control Systems

In a typical magnetic bearing application, the magnetic forces, which are applied by sets of electromagnets, must be adjusted online to ensure that the rotor is accurately positioned. A controller is an essential component in magnetic bearing systems, however, the control problem is complicated due to the inherent nonlinearities associated with the electromechanical dynamics of the system.

The most common method for analyzing and designing the controller for a magnetic bearing system is based on linear approximations of the nonlinear model [6]. Bearing performance near the equilibrium condition can be very good, but control effectiveness deteriorates rapidly when the system deviates far from the nominal operating condition.

The control system consists of controller, power amplifiers, and a power supply. Additional circuitry is present for conditioning of the signals from the position sensors, and conversion of the digital outputs from the controller to analog signals supplied to the analog amplifiers.

Control systems for our system can be either of the analog or digital type. Analog control systems have been used in control systems for over 30 years, but are rapidly being displaced by digital control systems.

Digital control systems take advantage of the tremendous developments in digital signal processing over the past decade, and allow more types of control algorithms to be used, enabling more results to be achieved. Typical sub-systems in magnetically suspended disk control include position signal processing, digital signal processing, digital-to-analog (D/A) conversion, power amplifiers and power supplies.

Many schemes for controlling magnetic bearing systems have been proposed; three approaches involve the feedback of a linear, time-invariant (LTI) controller and differ only in their choice of sensors are:

1. Using a measurement of the rotor position thru displacement sensors for feedback.
2. The self-sensing bearing which uses only a measurement of the coil currents to detect the rotor position.
3. The third approach uses both position and current measurements for control.

There are many types of control strategies or systems; this would include PID control, nonlinear control, sliding Mode control, and fuzzy logic control. Below we discuss some of these control systems.

2.2.1 PID Control

Figure 2.3 shows a block diagram of a general PID controller, it can be represented by the following differential equation:

$$u(t) = K_p e(t) + K_i \int_{t_0}^{t} e(\tau) d\tau + K_d \frac{de(t)}{dt}$$

Where: K_d is the differential gain, K_i is the integral gain, and K_p is the proportional gain. And the input to the controller is *e(t)* which is the measured error of the system, i.e. the difference between the set point and the measured output of the controlled system. And *u(t)* is the output of the controller to act on the final control element (the controlled system). The output here depends on the error, its derivative, and its integral

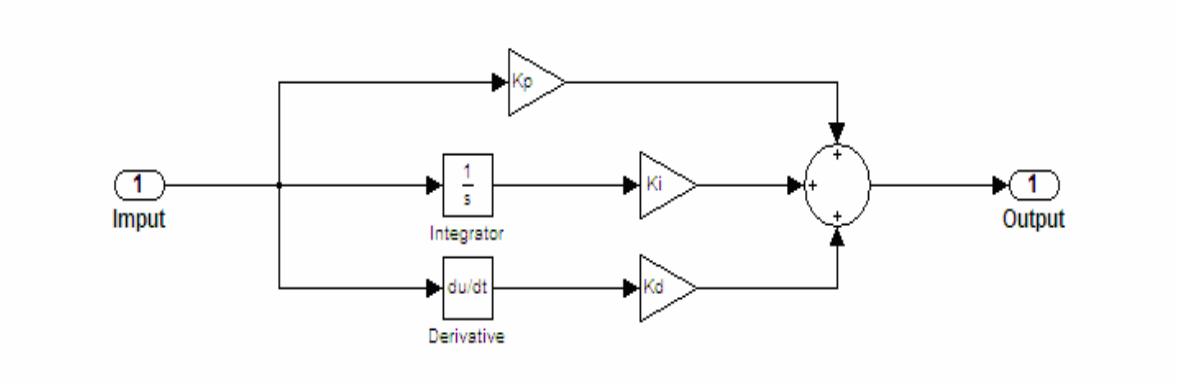


Figure 2.3 Block Diagram of PID controller

This can be represented by the transfer function:

$$P(s) = \frac{U(s)}{E(s)} = K_p + \frac{1}{s}K_i + sK_d$$

PID controllers are the most used controllers in the world; this is because they are easy to design and easy to tune.

2.2.2 Fuzzy Control

Fuzzy control is based on fuzzy logic, which is about mapping inputs to the appropriate outputs [12]. Between the input and the output a block of *fuzzy rules* does the work. The block of fuzzy rules is responsible for making decisions, and the rules themselves are generally written based on observations.

Here is a list of general observations about fuzzy logic [12]:

- The concept of fuzzy logic is easy to understand. The mathematical concepts behind fuzzy reasoning are very simple.
- Fuzzy logic is flexible. For any system, it's easy to add more functionality without starting from scratch. Fuzzy logic is tolerant of imprecise data.

- Nonlinear functions of arbitrary complexity can be modeled using fuzzy logic.
- You can create a fuzzy system to match any set of input-output data. This process is made particularly easy by adaptive techniques like Adaptive Neuro-Fuzzy Systems
- Fuzzy logic can be used along with conventional control techniques, instead of replacing them. In many cases fuzzy systems can be used to improve their operation or to simplify their implementation.
- Fuzzy logic is based on natural language. Some simple sets of if-then rules.

2.3 Yarn Spinning

Spinning is the act of transforming fibers into yarn. In principle, a staple-fiber yarn spinning system should consist of three basic mechanisms as shown in Figure 2.4

- Drafting mechanism
- Consolidation mechanism
- Winding mechanism

The drafting mechanism attenuates the fiber strand down to the wanted yarn size. This may be achieved using drafting rollers, or a combination of an opening roll and air stream. The consolidation mechanism provides inter-fiber cohesive forces to hold the fibers together in the yarn. In other words, it provides yarn strength and structure integrity [4]. The winding mechanism winds the yarn on a package (a bobbin or a cone), and builds the yarn along the length of the package. There are several types of spinning, we

list them here, and since the ring spinning is the focus application of this research we will explain it in details.

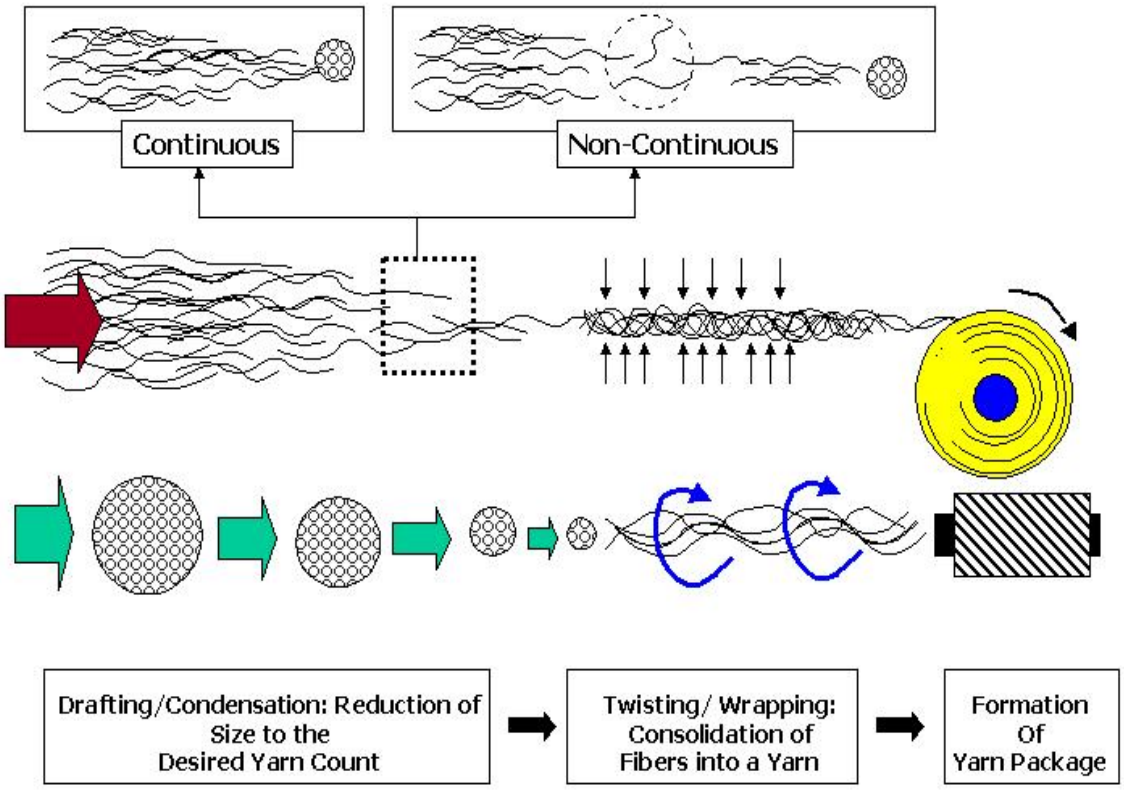


Figure 2.4 The basic principle of spinning

The drafting mechanism attenuates the fiber strand down to the wanted yarn size. This may be achieved using drafting rollers, or a combination of an opening roll and air stream. The consolidation mechanism provides inter-fiber cohesive forces to hold the fibers together in the yarn. In other words, it provides yarn strength and structure integrity [4]. The winding mechanism winds the yarn on a package (a bobbin or a cone), and builds the yarn along the length of the package. There are several types of spinning, we

list them here, and since the ring spinning is the focus application of this research we will explain it in details.

The major four types of spinning systems are the traditional ring spinning, rotor spinning, air-jet spinning, and friction spinning. Figure 2.5 represents the current offerings in spinning machines and their comparative spinning speeds [13].

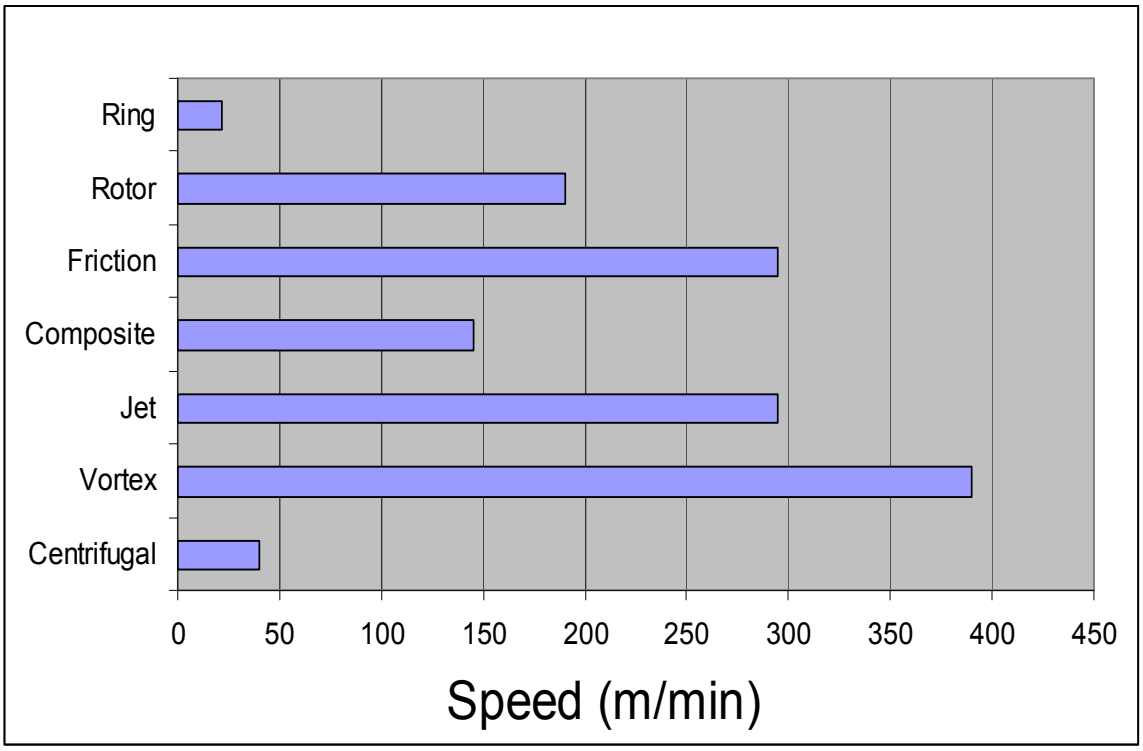


Figure 2.5 Current offerings in spinning machines

Table 2.1, below, summarizes the number of spinning positions for the major technologies, together with their share of the spun yarn market, for the years 1985, 1994, and 2002 [13].

Table 2.1 Major Spinning Systems in the United States

	Ring	Rotor	Jet
# Positions 1985	14,287,000	239,000	26,862
# Positions 1994	6,261,000	1,050,000	106,626
# Positions 2002	2,641,000	796,100	112,000
% of yarn produced	23.1	67.3	9.6

2.3.1 Open End Spinning

Open end spinning is a spinning system in which sliver feedstock is highly drafted and thus creates an open end or break in the fiber flow. The fibers are subsequently assembled on the end of a rotating yarn and twisted in. Techniques for collecting and twisting the fibers into a yarn include rotor spinning and friction spinning [9,13].

2.3.1.1 Rotor Spinning

Rotor spinning is a method of open end spinning which uses a rotor (a high speed centrifuge) to collect and twist individual fibers into a yarn. It offers a significant increase in twisting speeds.

Rotor yarns are different from ring-spun yarns in many aspects; they have lower strength yarns, but with improved hairiness and yarn abrasion. This tends to offer advantages in processing through weaving and knitting. This difference is a result of structural differences introduced during yarn formation, some fibers form a loose or tight spiral around the yarn, they are called ‘wrapper fibers’, and they cause reduction in yarn

quality. It is impossible to eliminate the formation of wrapper fibers. Smaller rotors are required for higher processing speeds, this also negatively impacts wrapper fibers, and thus higher speeds often carry the penalty of a reduction in yarn quality.

2.3.1.2 Friction Spinning

This method consists of fiber separation by a roller, then reassembly of the fibers to form the yarn; the required twist is then inserted by frictional contact between the yarn surface and rotating cylinders (rollers). This method consumes less power than rotor spinning, and it provides lower yarn tension. There is no wrapper fiber.

2.3.1.3 Open-End Yarns

Open-end yarns have the advantage of regularity and cleanliness, the cost of producing them is lower than the cost of ring-spun yarns. They are not suitable applications where low twist, or where fine count is required.

2.3.2 Jet Spinning

Air-flow spirals through a tube in a direction opposite to the yarn, so, the fibers attach to the yarn end and become oriented about the axis of the forming yarn.
Jet-spinning offers high-speed production of finer-count yarns and thus did not directly compete with rotor spinning.

It has the major disadvantage of not being able to produce acceptable 100-percent cotton yarns. It is also restricted to finer counts because yarn tenacity reduces when the yarn becomes coarser.

For optimum processing, there also are higher quality requirements on the feed sliver with extra drawing or combing operations. Despite these limitations, and the necessity to optimize finishing in order to promote an acceptable hand, jet spinning is a viable system in the United States because of high productivity — 250 meters per minute for the MJS 802H — and adequate yarn and fabric quality. also, the core sheath structure of the yarn tends to minimize hairiness, which in turn reduces pilling propensity, often a major problem with polyester-rich blends.

2.3.3 Vortex Spinning

Murata Vortex Spinning (MVS) is a development of jet spinning specifically created to overcome the limitations of fiber type. The major marketing feature of MVS was that it was capable of spinning uncombed cotton slivers into acceptable yarns at speeds that were significantly higher than with any other system. The yarn structure is different from jet-spun yarn with many more wrapper fibers, and in parts the vortex yarn resembles a two-fold yarn.

There were concerns that there is great fiber loss using this spinning machine. But, even though the fiber loss may be about 8%, most of this is short fiber, which would not contribute to yarn quality.

MVS was introduced with a potential processing speed of 350 to 400 m/min. Even though it is claimed that MVS is capable of processing 100-percent cotton, it is believed that the major use of this system is in the processing of cotton-rich blends with polyester.

2.3.4 Ring Spinning

Ring spinning is the most successful form of spinning; it can be used for processing any fiber. As can be seen in Table 1.1, ring spinning is still the most dominant spinning system - in United States there are about three times more spindles than installed rotors. Considering only the quantity of yarn produced, it is obvious that even though there are only one-third as many positions of rotors installed, rotor spinning produces three times more yarn than traditional ring spinning. The technology behind ring spinning has remained largely unchanged for many years; using a traveler to guide the yarn on the package.

Ring spinning is characterized by two main features:

1. continuity of fiber flow from roving (the input strand) to yarn
2. Tension-controlled spinning process.

These two features summarize the basic principle of ring spinning as discussed below.

A rough sketch of ring spinning is shown in Figure 1.2. The input fiber strand (roving) is drafted using roller drafting to reduce its size down to the desirable yarn count. The fibers being delivered at the nip of the front roller form a triangle called the "spinning triangle". The bottom end of this triangle is commonly known as the twisting point.

The consolidation mechanism in ring spinning is twisting. Twist is inserted to the fibers by a traveler rotating around a ring flange. The driving mechanism of the traveler is the spindle, which carries the yarn bobbin. The amount of twist inserted in the yarn is controlled by the front roll (delivery) speed and the traveler rotational speed. Specifically,

the number of turns per inch (twist) is expressed by

$$Twist = Turnsperinch(t.p.i) = \frac{n_{traveler}(rpm)}{V_{delivery}(inch/\min)} \qquad [4]$$

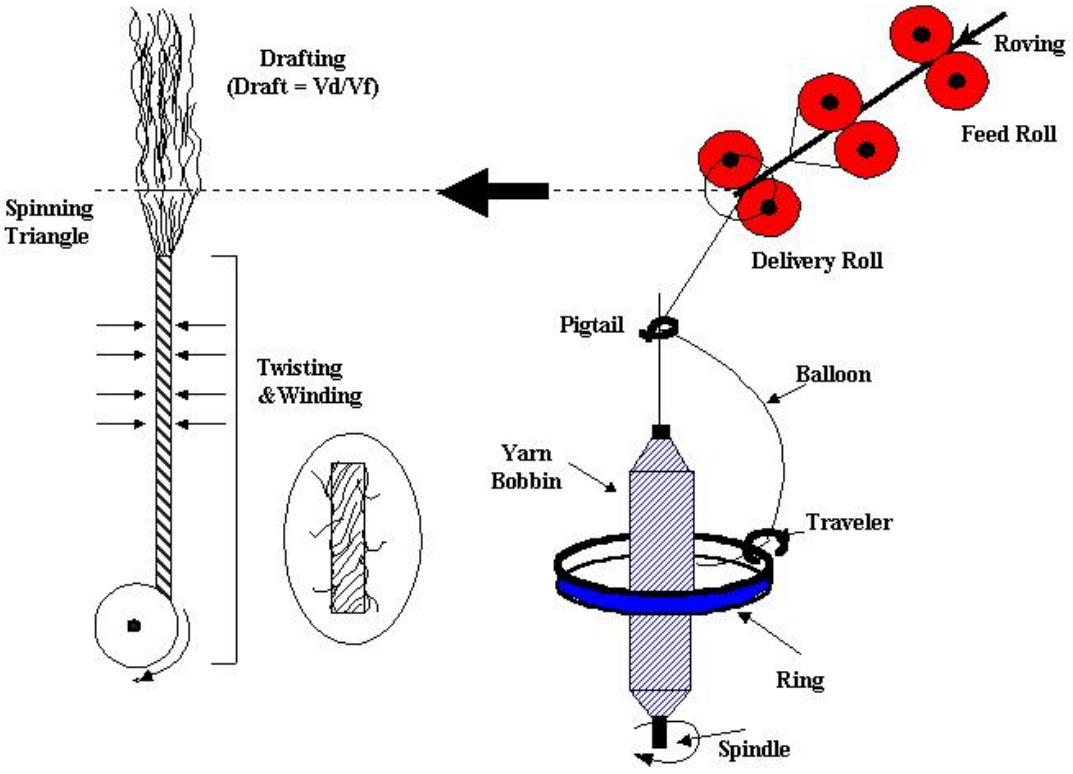


Figure 2.6 Sketch of ring spinning [4]

A rough sketch of ring spinning is shown in Figure 2.6; the fiber strand is reduced in thickness by a drafting system. The fibers receive twist from the rotation of a spindle and traveler) which directs the newly-formed yarn onto the bobbin. The consolidation mechanism in ring spinning is twisting. Twist is inserted to the fibers by a traveler rotating around a ring flange. The driving mechanism of the traveler is the spindle, which

carries the yarn bobbin. The amount of twist inserted in the yarn is controlled by the front roll (or delivery) speed and the traveler rotational speed.

There have been significant refinements. Changes, which on their own offered only slight advantages, provided the following synergies when combined:

• The introduction of longer frames reduced the relative costs of automatic doffing.

• The combination of spinning frame and winding (link winders) further enhanced the adoption of automation.

• The introduction of automatic doffing meant that doffing time was reduced and thus package (and ring) size was less critical.

• The introduction of splicing on the winder meant that yarn joins became less obtrusive — again offering the potential of smaller package.

• Smaller rings meant that for a limiting traveler velocity (40 meters per second [m/s]), higher rotational speeds (and hence twisting rates) could be achieved.

3. SYSTEM MODELLING

3.1 The Concept of Magnetic Control

The concept of magnetic spinning is emphasized in Figure 3.1. A magnetic spinning system mainly consists of a lightweight rotor suspended in a magnetic field created inside a fixed stator; the rotor can spin freely inside the stator without any contact. The stator is equipped with magnetic actuators that always keep the rotor in its central position. The rotor in this configuration replaces the ring and traveler in the traditional spinning system.

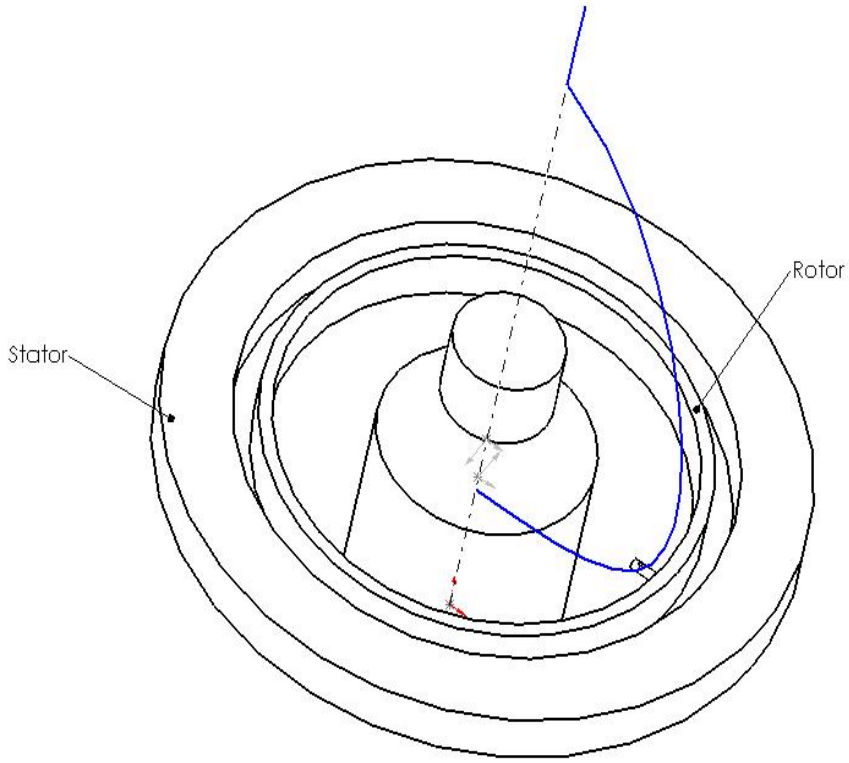


Figure 3.1 Magnetic-Ring Spinning

3.2 Principal of Operation

Figure 3.2 shows a schematic cross section view of the magnetic system (X-Z plan view). Permanent magnets create a bias flux, shown in blue paths, across the air gaps (g_1, and g_2), supporting the weight of the rotating disk in the axial direction. When the ring is exactly in the center, and assuming that all permanent magnets provide equal magnetic field intensities (blue path in Figure 3.2), the ring will remain in the center, but this is a very unstable condition. Any disturbances or noises will cause the ring to be shifted from the center. In case the floating ring is displaced from its central position to any direction, the permanent magnets will create a destabilizing force that attracts the ring even further away from the center towards the same direction, because the attracting force between the magnet and the rotor is a function of the square of the distance between them, the closer the distance the higher the force.

A set of inductive sensors will read out the deviation from the center position as a change in the inductance of the sensors, using two displacement sensors mounted radially to the floating ring, the change in the inductance will be transformed to a voltage signal through control circuit. A control system will read this signal and generate a corrective current signal to power amplifiers. The power amplifiers in turn supply the electromagnetic coils (the actuators) with current to generate the corrective flux (shown in Figure 3.2 by green dotted path). This corrective flux adds to the flux caused by the permanent magnets in the large gap and subtracts from the permanent magnets flux in the small gap, causing the magnetic flux to increase in the large gap and decrease in the small gap, hence, the total magnetic force will tend to bring the floating ring to its central position.

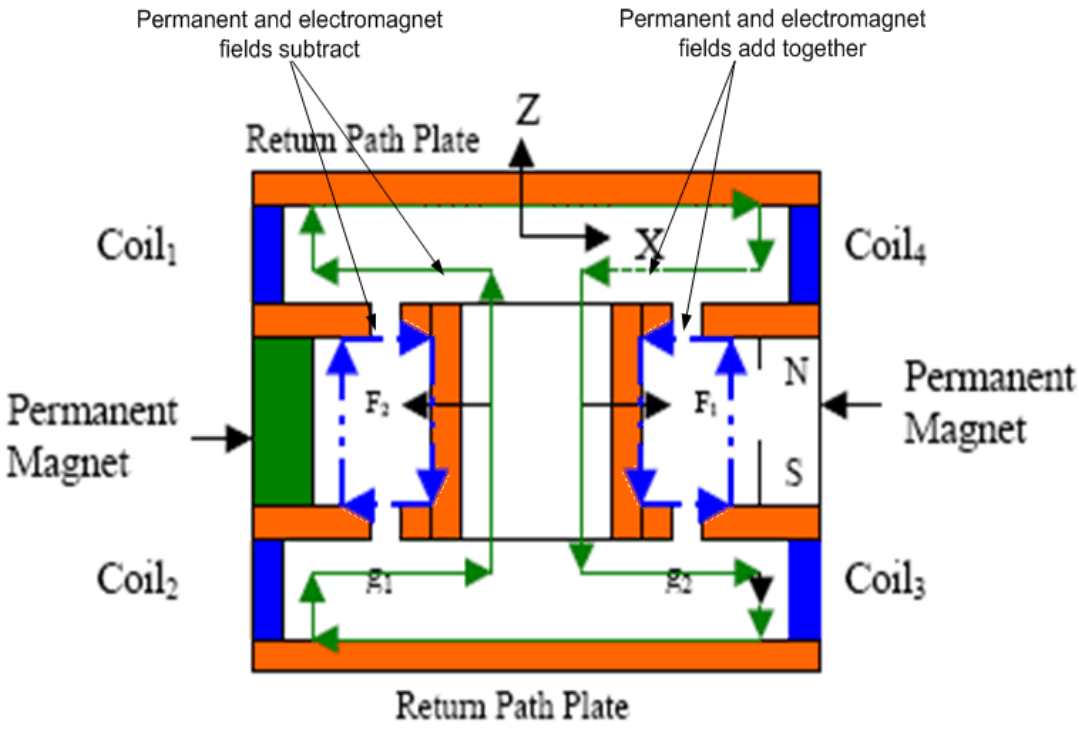


Figure 3.2 Schematic cross-section view of the magnetic system

3.3 System Theoretical Modeling

First a theoretical model of the system had to be derived, that helps in establishing a first trial control system and examining how the system will respond to changes in the operating and control parameters. In this modeling, only one plane of movement is considered (X-Z plane Fig. 3.2). The perpendicular plane (Y-Z) is identical. Also, it is assumed in this study that there is no flux leakage and the magnetic flux densities are below saturation i.e. linear B-H relation exists. As well, we assume that the filed intensity in the cores is negligible due to the high permeability of the cores. In this model, twelve permanent magnets are considered to be mounted around the ring.

3.3.1 Field Intensity

The field intensity of a permanent magnet H_p, in a gap g_i, is equal to:

$$H_{g_i P} = -\frac{H_m l_m}{2g_i} \qquad (1)$$

Where g_i is the width of the gap i (i =1, 2), and (H_m, B_m) is the operating point of the permanent magnet, and it can be found as explained in the next section.

3.3.2 Permanent Magnet Operating Point

The load line (the equation that characterizes the load behavior of the permanent magnet load) is defined by the following equation:

$$B_m = -\mu_0 \frac{H_m l_m}{2g} \qquad (2)$$

Solving this equation with the permanent magnet demagnetization curve (the hysteresis loop)- results in the operating point of the magnet. Since it is hard to solve it with the demagnetization curve, we can solve it with a linearized equation of the demagnetization curve at the suitable quarter of the (B-H) plane.

The approximate demagnetization curve is in the form of:

$$B_m = B_r + \frac{B}{H_c} H_m \qquad (3)$$

Where B_r is the residual flux density and H_c is the coercive force of the permanent magnet.

Figure 3.3 shows the solution of the two equations (2), and (3), (the intersections between the two lines).

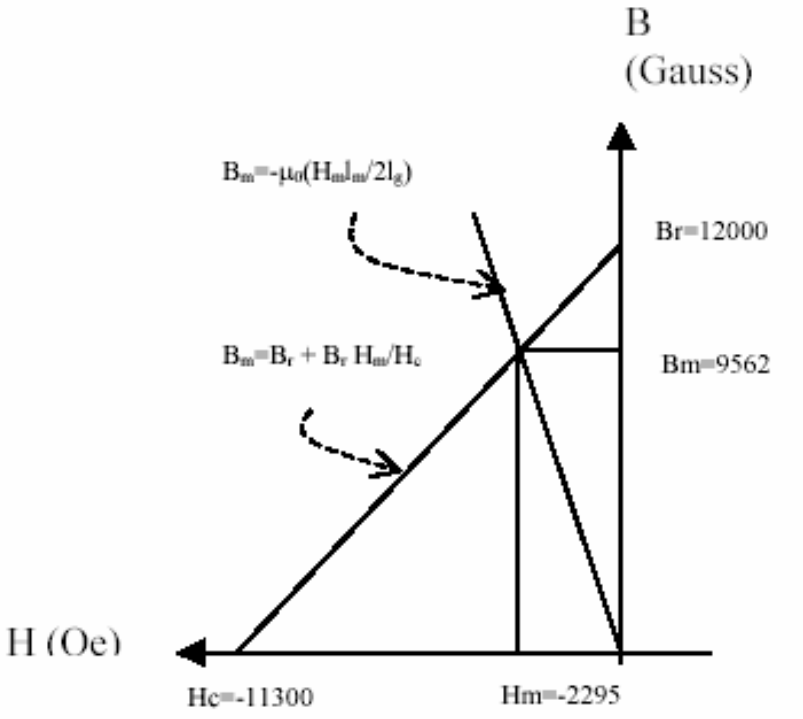


Figure 3.3 Solution of Equations (2) and (3)

3.3.3 The Field Intensity in Each Gap Due to the Coils

The field in the gap due to coils 1 and 2:

$2H_{gc}\, g_i = 2NI$

$$H_{g_{ic}} = \frac{2NI}{g_i} \tag{4}$$

3.3.4 The Total Field intensity and Force

The total field in each gap equals:

$$\mathrm{H}_{g_i} = \frac{NI}{g_i} - \frac{H_m l_m}{2g_i} \tag{5}$$

The attracting force of each side equals:

From (4) in (5);

$$F_i = \frac{\mu_0 S}{g_i^2}(NI - \frac{H_m l_m}{2})^2 \qquad (6)$$

The total force acting on the rotor in x-direction:

$$F_{x-tot} = F_1 - F_2 = \mu_0 S\left[\left(\frac{NI - \frac{H_m l_m}{2}}{g_1}\right)^2 - \left(\frac{NI + \frac{H_m l_m}{2}}{g_2}\right)^2\right] \qquad (7)$$

Let $g_1 = \Delta - x$, hence, $g_2 = \Delta + x$, *see figure 3.4*

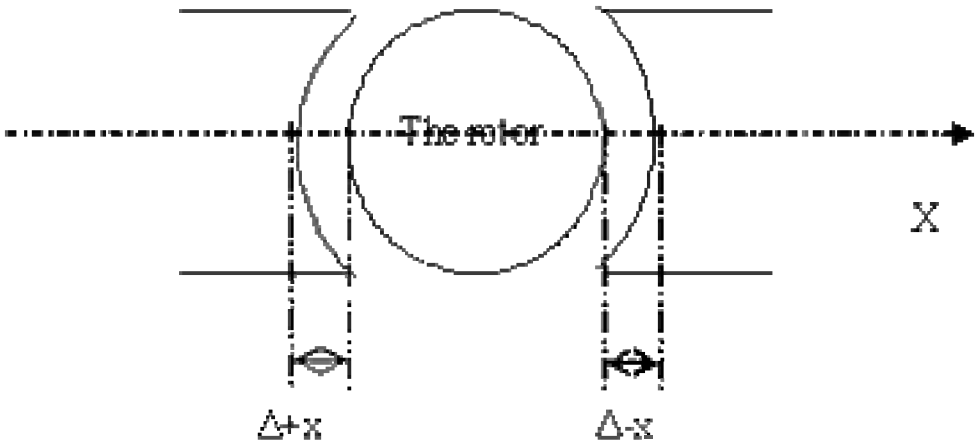


Figure 3.4 The Two Gaps (g_1 and g_2)

And let $H_m l_m = 2NI_m$, (as if the permanent magnet has been replaced with a coil with N_m turns and I_m DC current

Using these two quantities, equation 7 can be rewritten as:

$$F_{x-tot} = \mu_0 SN\left(\frac{I_m}{\Delta}\right)^2\left[\left(\frac{1-I/I_m}{1-x/\Delta}\right)^2-\left(\frac{I+I/I_m}{1+x/\Delta}\right)^2\right] \quad (9)$$

Let $K_1 = \mu_0 SN\left(\frac{I_m}{\Delta}\right)^2$

Equation 9 can be approximated as (see appendix 4):

$$F_{x-tot} = K_1\left(\frac{I}{I_m}+\frac{x}{\Delta}\right) \quad (11)$$

Hence, the differential equation that describes the rotating ring is:

$$mx'' = F_{x-tot} + f(t)$$

$$mx'' = K_1\left(\frac{I}{I_m}+\frac{x}{\Delta}\right) + f(t) \quad (12)$$

Where *f(t)* describes any disturbance force acting on the floating ring (in our case, the yarn tension is considered as a disturbing force), K_1 is a constant depending on the system parameters.

Taking the Laplace transform for both sides of equation (12), and neglecting the disturbance term *f(t)*:

$$mXs^2 = 4K\left(\frac{I}{I_m}+\frac{X}{\Delta}\right)$$

$$X\left(\frac{ms^2}{4K}-\frac{1}{\Delta}\right) = \frac{I}{I_m}$$

$$\frac{X}{I} = \frac{1}{\frac{mI_m}{4K}s^2 - \frac{I_m}{\Delta}} \quad (13)$$

The transfer function (13) has two poles on the right hand side of the complex plane which indicates instability. To overcome this situation a control system has to be added to the system, the function of the control is to stabilize the naturally unstable system, as discussed in the next chapter.

3.4 Magnetic Spinning System Description

Magnetic levitation systems can be realized by using attractive or repulsive forces. A better mass/stiffness ratio can be achieved by using the attractive force mode [10]. Preference was given to the two DOF option where the ring is actively controlled along two orthogonal radial directions where axial movements and all other degrees of rotor freedom are passively controlled by means of permanent magnets, except for the rotor spin. The two radial axes are independently controlled by their control loops. In the system design both permanent magnets and electromagnetic coils are used. Most of the DOF are passively controlled. This has the advantages of high reliability and low power consumption because the amount of electronics is reduced.

The permanent magnets produce the main part of the magnetic flux in the magnetic circuit and the electromagnetic coils modulate this static bias flux, allowing the control of restoring forces on the rotor to keep it centered. This modulation is necessary to provide active control in the radial direction in the presence of imbalance or external forces. Another advantage is the linearized characteristic of force vs. current through the superposition of permanent magnetic and electromagnetic fluxes [12]. Rare-earth permanent magnets were chosen because they offer a high energy density and have advantages in terms of mass and volume; this means that they provide large energy

density for small size and light weight magnets. The magnet used in our design is a 'Neo-35' cylindrical magnet; it has a residual flux density 'B_r' of 12,000 Gauss, and coercive force 'H_c' of 11300 Oersteds. The Height of the magnet is 25mm, and its diameter is 6mm.

3.4.1 The Whole Model

The Block diagram in Figure 3.5 shows the basic components in a magnetic ring spinning system. The mechanical part is basically the rotor/stator setup along with the actuators and the permanent magnets. The displacement sensors convert the position information to voltage, the controller is responsible of generating a correcting signal, and the power amplifier amplifies the output of the controller and feeds it to the electromagnets. In Figure 3.5 'i' is the set point current.

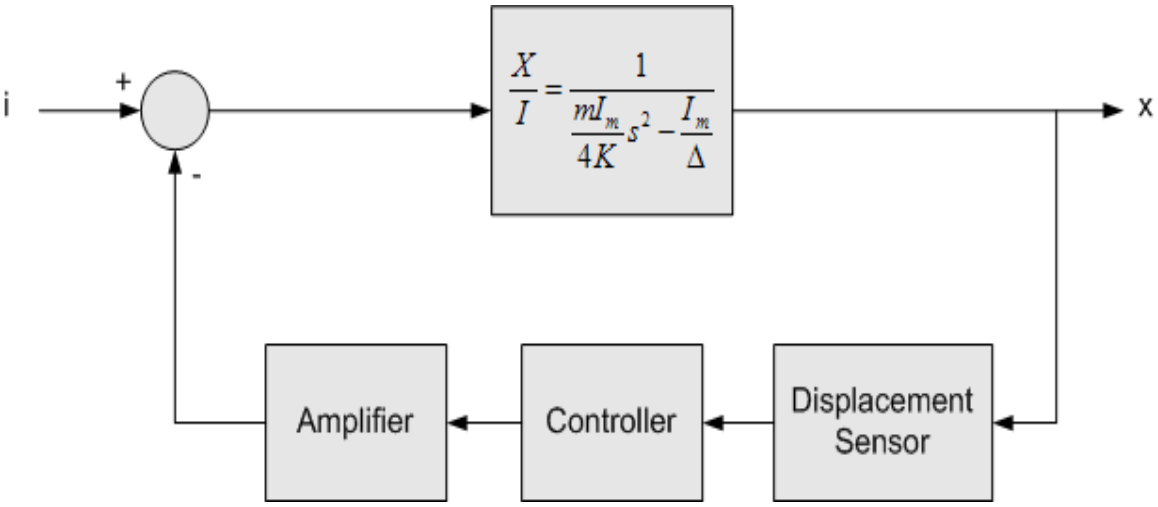


Figure 3.5 Magnetic spinning block diagram

3.4.2 The Magnetic Spinning System

Figure 3.6 shows a diagram of the assembled design of the magnetic ring spinning machine, while Figure 3.7 shows the same design disassembled. The detailed components of the design are shown in Figure 3.8.

The basic components of the system are:

i. The stator body; the main body of the stator, made from non-magnetic material, the holes are to support the permanent magnets.

ii. The permanent magnets; twelve permanent magnets to generate a permanent magnetic field (bias magnetic flux) that suspends the rotor.

iii. The floating ring; made of silicon iron.

iv. Flux plates; eight flux plates; four on the top and four on the bottom of the machine, their function is to carry the magnetic flux from the magnets to the area around the ring.

v. Axial support disc; to prevent the rotor from falling down.

vi. Assembly bolts; to assemble the machine together.

vii. Coils, with ferrite cores; the electromagnets responsible of converting the electrical control signals from the controller to a magnetic flux that modulates the flux from the permanent magnets to control the position of the rotor.

viii. Return flux plates; their function is to provide a return path to the magnetic flux.

The bias magnetic flux is produced by the 12 cylindrical permanent magnets found in the stator body. The rotating ring is made of silicon iron for maximum flux densities without reaching saturation condition. The rotor in this case is working as a

return path of the magnetic field created by the permanent magnets. Four displacement sensors are imbedded into the stator body, each is inserted in one of the four sensor holes in the stator body (see Figure 3.6), the sensors are aligned in two axes Each two sensor are working in differential mode to detect the ring displacement in one axis (X or Y), for example, when the rotor is displaced in one axis towards one of the sensors, this sensor reads a decrease in the displacement, while the opposite sensor reads an increase in the displacement, the two readings are subtracted from each other to provide higher accuracy. The stator body, which is made of non-magnetic material, is assembled with 8 flux segments made of steel (flux plates); these segments are arranged in a symmetric manner around the X and Y axis and on the top and bottom of the stator body. The function of the flux segments is to carry the magnetic flux to the area surrounding the rotor. The stator body has two non-magnetic material rings installed at its inner diameter and facing two other non-magnetic rings installed on the outer diameter of the rotating ring. These rings serve in maintaining a minimum clearance between the flux segments and the rotating ring which is crucial in left off process of the ring (initial starting from rest). Two sets of actuators (electromagnet coils) are arranged in both axis of control (X and Y axis). In order to close the magnetic circuit two separate return flux plates are used for each axis. These plates have an air gap much greater than the gap between the rotating ring and the flux segments, to assure that the flux density is higher in the gap between the ring and the flux segments.

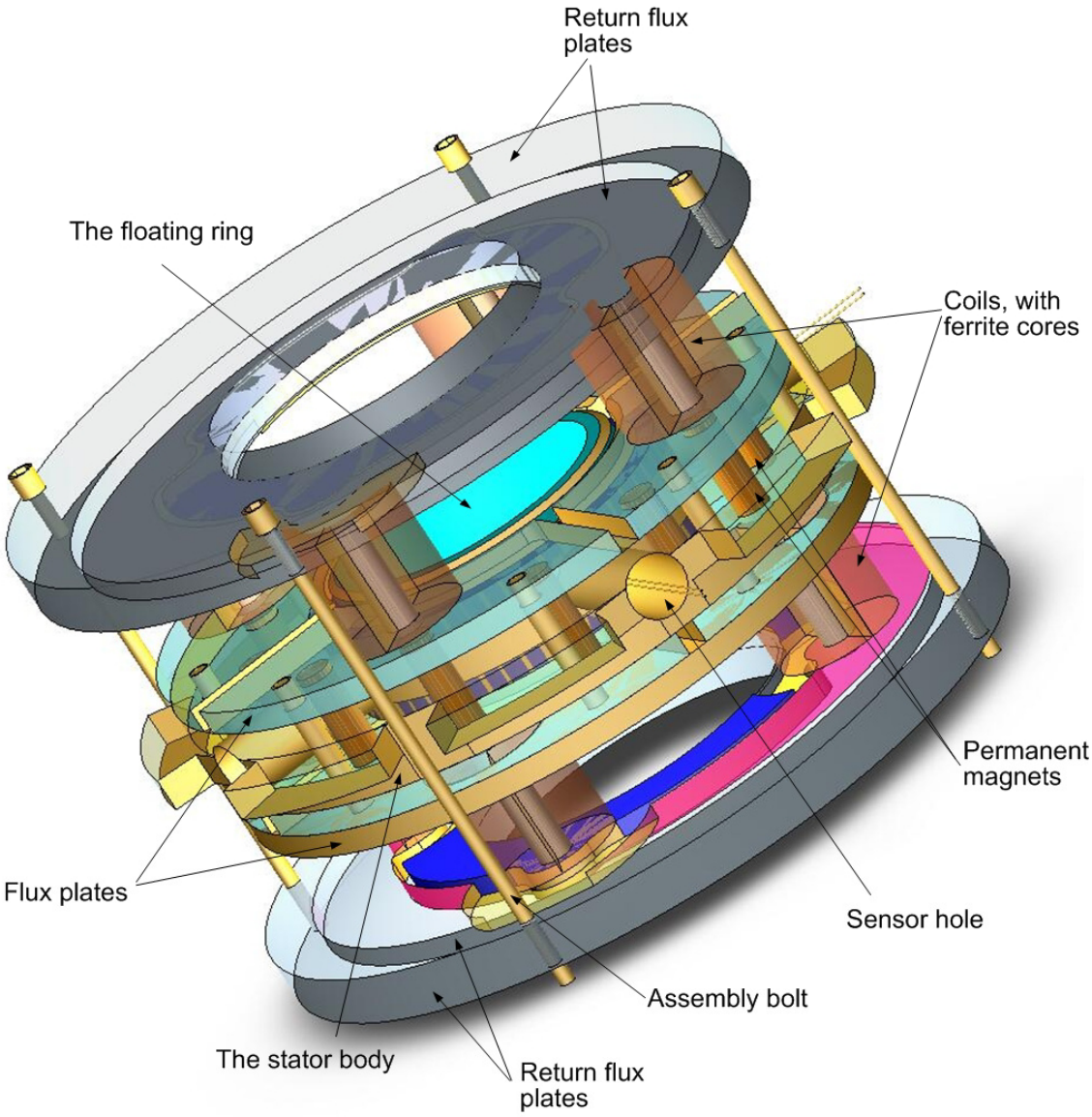


Figure 3.6.a Diagram of the Assembled Design of the Magnetic Ring Spinning Machine

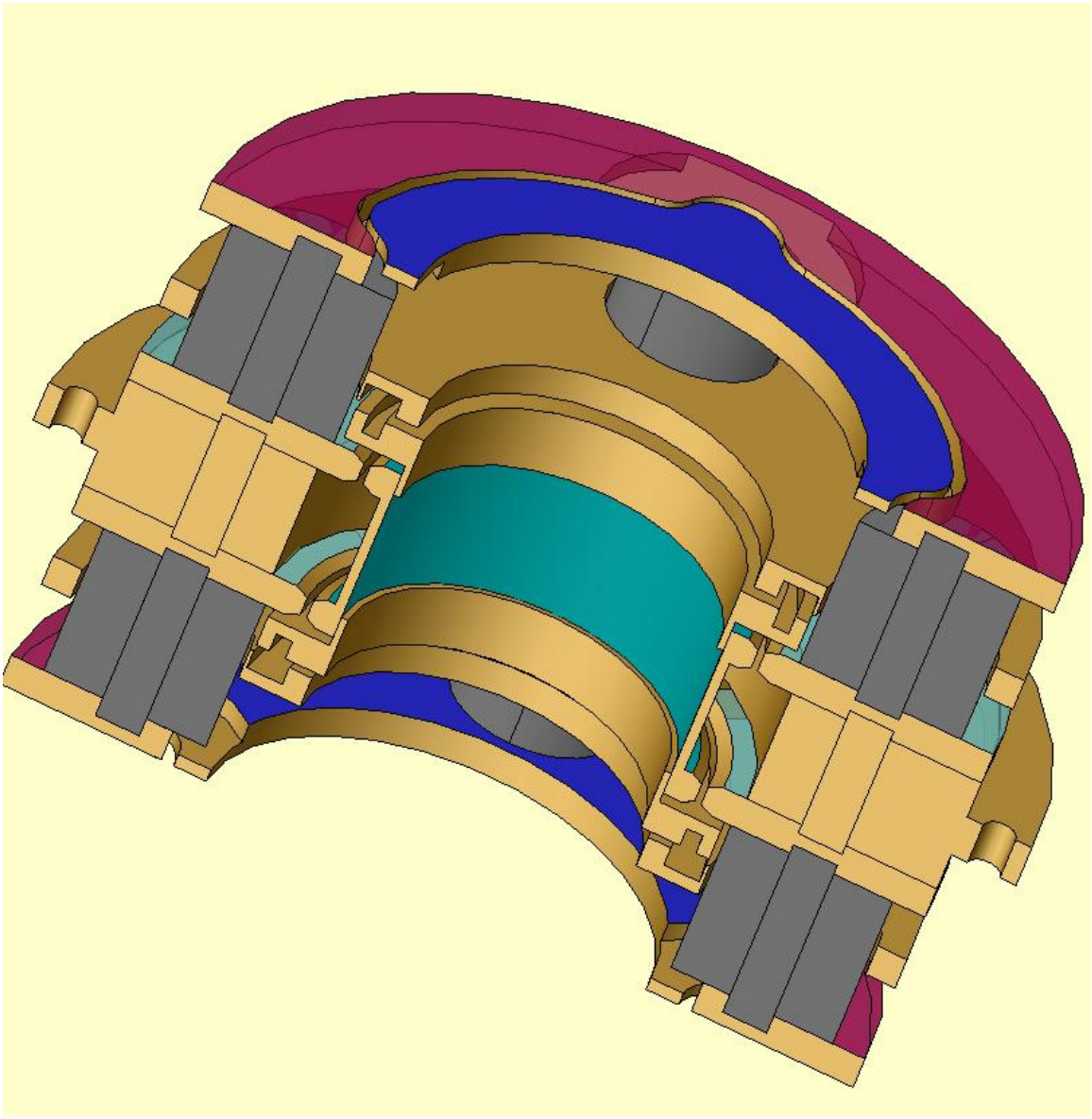


Figure 3.6.b Cross-Section Diagram of the Assembled Design of the Magnetic Ring Spinning Machine

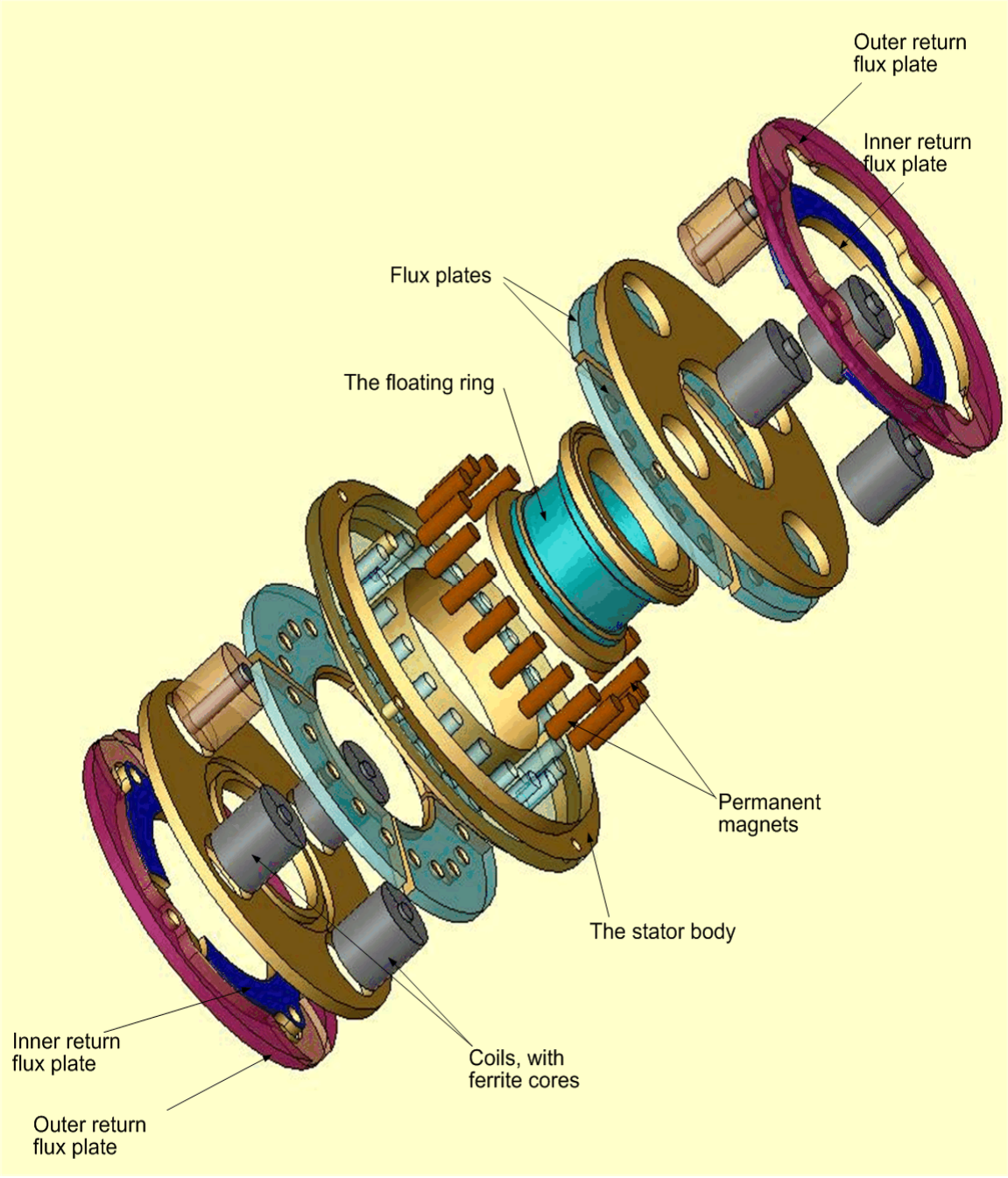


Figure 3.7 Disassembled design of the magnetic ring spinning machine

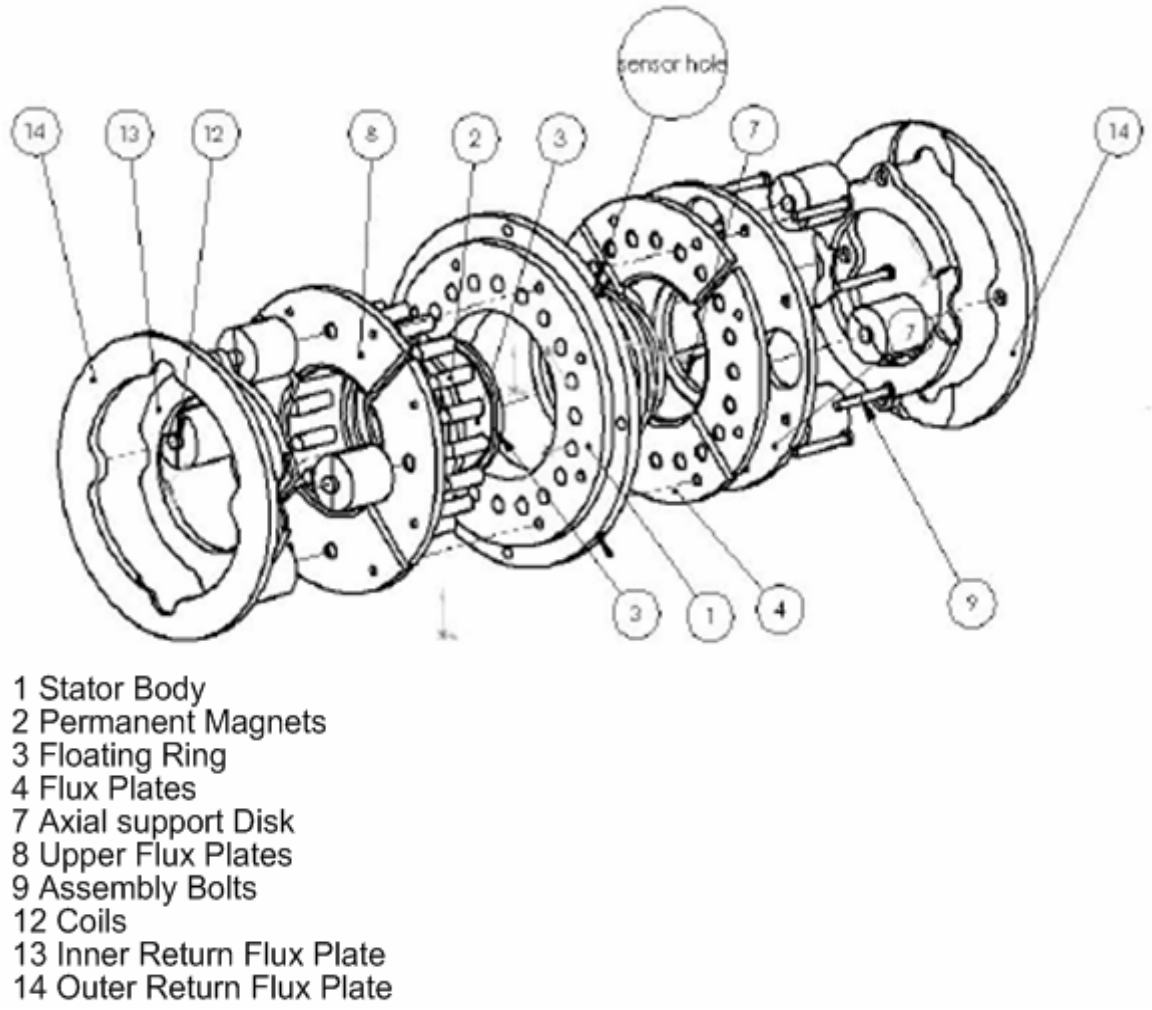


Figure 3.8 Detailed components of the design

3.5 Finite Element Validation:

For full validation of the current system a complete finite element model is carried out. The real model was enclosed with a cylinder of air with a greater diameter than the model by 150% and in height by 150%. This is essential to apply far field boundary conditions at the surfaces of that cylinder. These boundary conditions were imposing a tangential flux field at all the bounding surfaces of that cylinder. Figure 3.9 shows the model meshed with air cylinder removed.

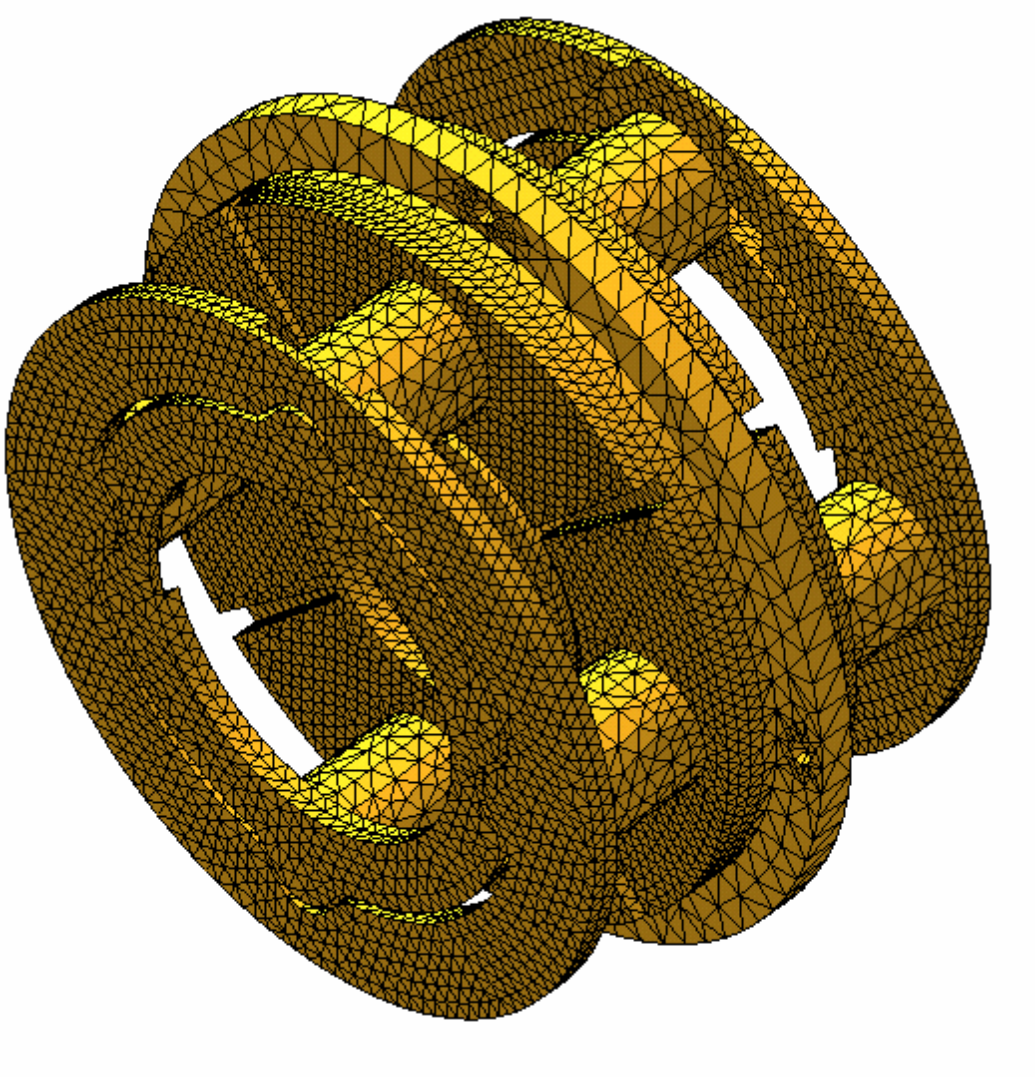


Figure 3.9 The finite elements model meshed with air cylinder removed

The total number of elements used to mesh this model is 370000. This huge number of element is used in order to obtain the most accurate results. On the other hand, the time taken to perform one run is about 10 min. This type of analysis has the advantage over the other methods (i.e. approximate closed form solution) in that there is no simplification of the model geometry and it takes into consideration any flux leakage. The current system was studied with the floating ring displaced from the stator center

incrementally .25 mm step in the X direction. Another study was carried out with the floating ring displaced in the Z direction to get the holding force in the Z direction that will support the ring weight and the axial force resulting from the yarn tension. For the current configuration, the holding Z force calculated (Figure 12) is found to be about 5 [N]. This force is greater than the sum of the floating ring weight and fiber tension in the Z direction by 5 times. Figure 3.10 shows the relation between the restoring force and the excitation current for 0.25, 0.5, and 0.75 mm displacement of the floating ring from its central position. At the point (F0) of intersection with the current axis the sum of the forces acting on the ring is equal to zero. This does not mean that there is no force holding the ring at that point, but the force holding the ring in the X direction is equal to the force in the –X direction. So, by increasing the current with a small amount the ring will start to move in the direction where the air gap is larger. By this action the floating ring will restore its central position.

The supporting disk (support the floating ring in the Z direction) will allow 1 mm for the floating ring to be shifted downwards due to any unexpected disturbance. The permanent magnets will still capable to lift the floating ring up again. This action can be seen from Figure 3.11 at 1 mm displacement where the resultant force in the Z direction is still about 5 N. The total weight of the floating ring and the two spacers are of about 95 gm. Figures 12, 13, 14, and 15 show the flux intensity variation with the increase of the control current. Figure 12 shows a high intensity of the flux at the smaller gap between the stator and the floating ring. To pull the ring back to its central position the control current is increased in steps. This action changes the flux around the floating ring. Figure 15 shows the final step where the flux intensity at the bigger gap is increased over that of

the smaller one. Hence, the total resultant force will attempt enlarge the small gap and reduce the large gap i.e. centering the ring.

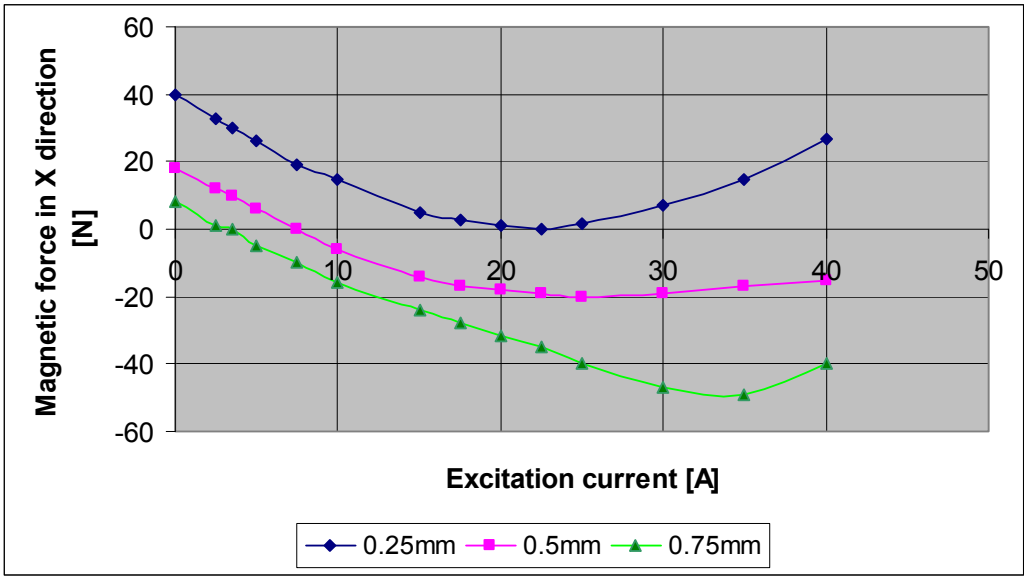


Figure 3.10 Variation of magnetic force with excitation current for different air gaps

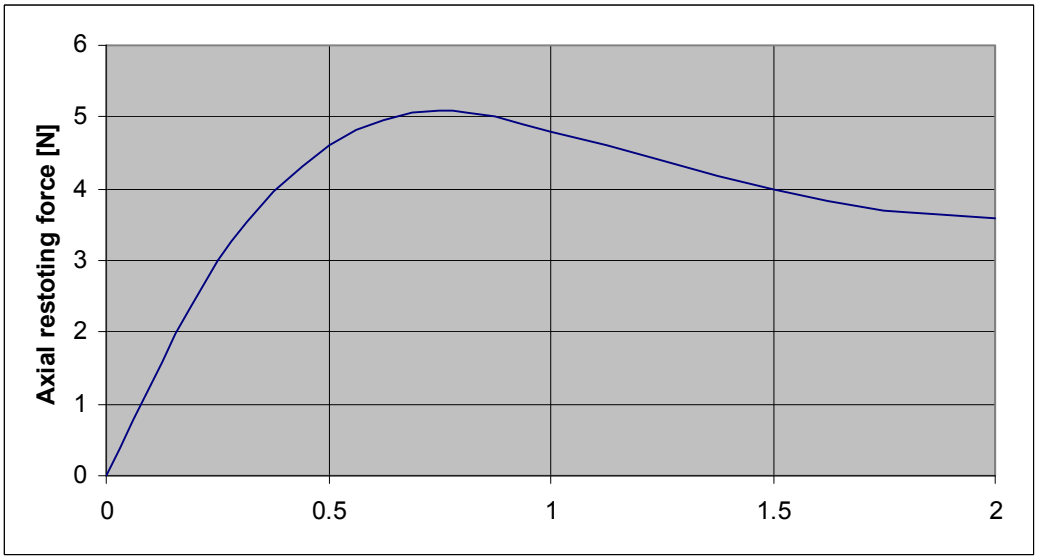


Figure 3.11 Axial restoring force

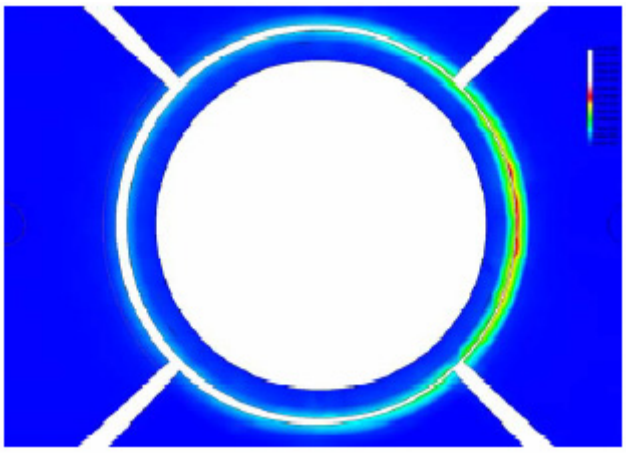


Figure 3.12 Top view of flux intensity pattern due to permanent magnets only (control current = 0)

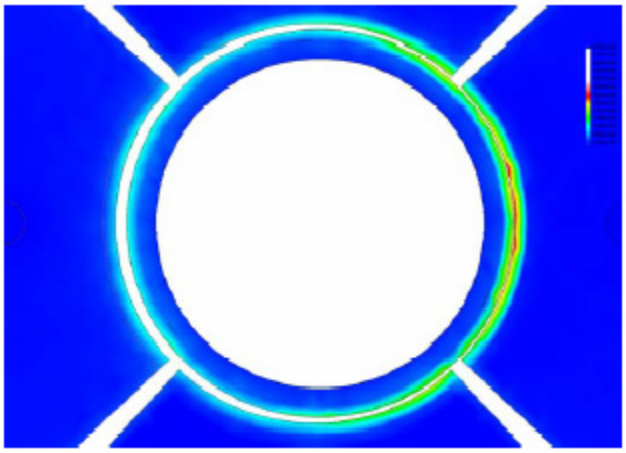


Figure 3.13 Top view of flux intensity pattern due to permanent magnets and control current = .5 A

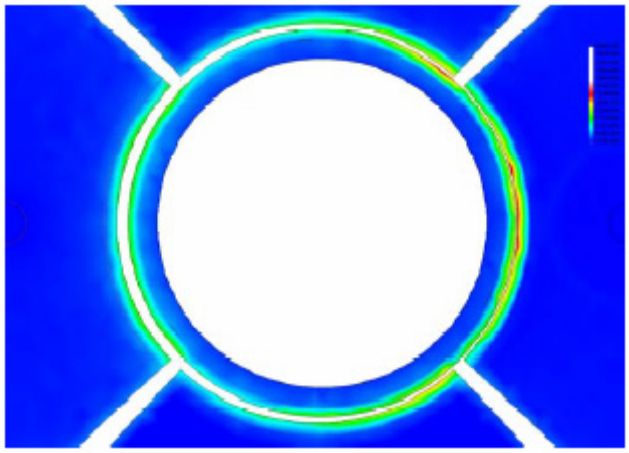


Figure 3.14 Top view of flux intensity pattern due to permanent magnets and control current = .75 A

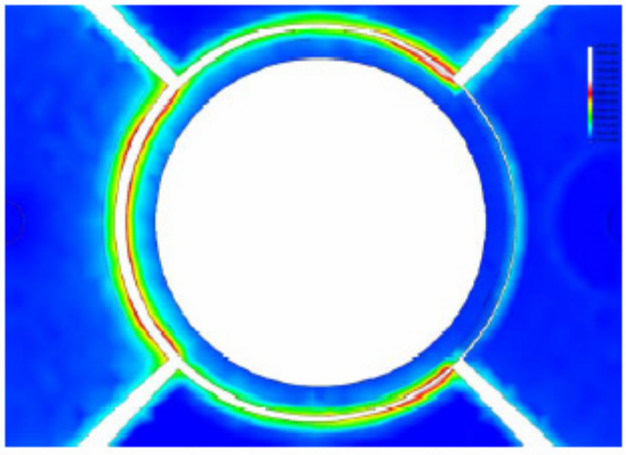


Figure 3.15 Top view of flux intensity pattern due to permanent magnets and control current = 1.5 A

3.5 Conclusions

A new ring-spinning system named “magnetic-spinning” is designed, the system is based on the concept of magnetic-suspension, where a light weight rotor is designed to be magnetically suspended inside a fixed stator, using electromagnets that modulates the field of radially mounted permanent magnets to keep the rotor suspended, displacement sensors to sense the position of the rotor, and control circuit to adjust the electromagnets’ input current based on the position. In the new designed system, the rotor replaces the ring and traveler in the traditional spinning system.

An approximate analytical model and transfer function of the system were derived, based on the design.

The model was then validated using a complete finite element model.

4. CONTROL ENGINEERING ASPECTS

4.1 Introduction

The new developed system requires active control in two radial axes because of the inherent instability in these directions. The role of the control system can be simply explained as it reduces the upper system current when the rotor is above the center and increases the current when the rotor is below the center. Four sensors are inserted into the stator body to measure the distance of the rotating ring in differential mode. Typically, magnetic system control is performed in a single-input, single-output (SISO) manner. The position information from one sensor causes only the control current in the corresponding axis to be varied. In order realize a control system capable of stabilizing the rotating ring during spinning a complete block diagram of the system has to be constructed first. The main element of the system is the dynamic of the rotating ring. The other components of the system include position sensors and accompanying electronics, a controller, and amplifiers. Figure 4.1 shows a block diagram of the system.

4.2 Position Sensing

The displacement sensing system used consists of two main parts; the sensors that read the changes in the displacement of the rotor, and the interface circuit that interprets these changes to a voltage signal.

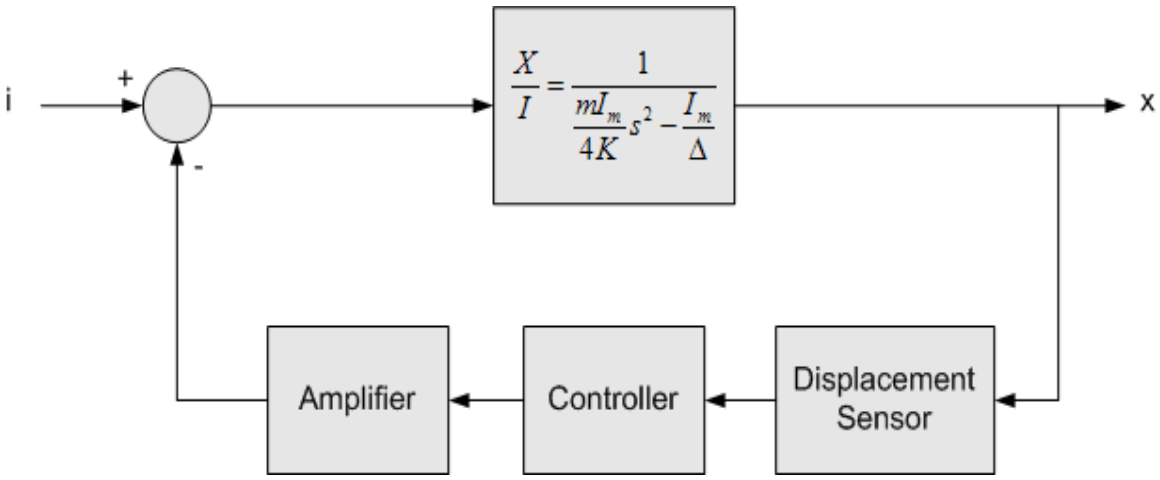


Figure 4.1 Block Diagram of the System

4.2.1 Displacement Sensors

The used sensors are inductive sensors that change their inductance in proportion to the displacement between the top of the sensor and the facing surface of the rotor using a 30 kHz carrier signal, with the change of the target surface (floating ring) distance from the sensor.

Inductance of a coil is affected by the existence of any ferrite material that crosses the magnetic field created by the current going through the coil. This property is used in displacement sensing. An AC current is induced in the coil of the sensor, the value of the inductance changes in proportion to the displacement between the surface of the coil and the ferrite object which displacement from the sensor is to be measured – of course, there is a linear range for the operation-, a circuit is used to read the change in the inductance and interprets it as a displacement (the used circuit is explained below).

The displacement sensors are mounted radially to the floating ring, two in each direction, as clarified in Figure 4.2.

We use differential configuration of the sensors, this means that for each direction the feedback signal that carries the position information is the difference between the signals from the set of two sensors (sensors on opposite sides of the rotor on the same axis), and this increases the sensitivity of the measurement.

4.2.2 Interface Circuit

To read the signal from the sensors, an adaptor circuit was built, Figure 4.3 shows a diagram of the circuit, the function of this circuit is to read the changes in the inductance of a sensor and convert it to a corresponding voltage signal.

The first two NAND-gates (A and B), along with the connected capacitors and resistors work as a pulse generator. The generated square signal is filtered through the high-pass filter made from the inductor and the resistor, the high pass filtered signal is the input for the next stage (NAND-gate), the output of this stage is a PWM (pulse width modulated) signal, which DC (duty cycle) is proportional to the value of the inductance. The next stage is a low pass filter stage that converts the PWM signal to a voltage value proportional to the value of the inductance.

The IC (U2) is a regulator to assure a stable value of voltage supply, and to set a reference to the circuit, the potentiometers RV1, and RV2, are for calibration purpose; they are adjusted with a reference inductance connected (inductance with a known value) to calibrate the output voltage value.

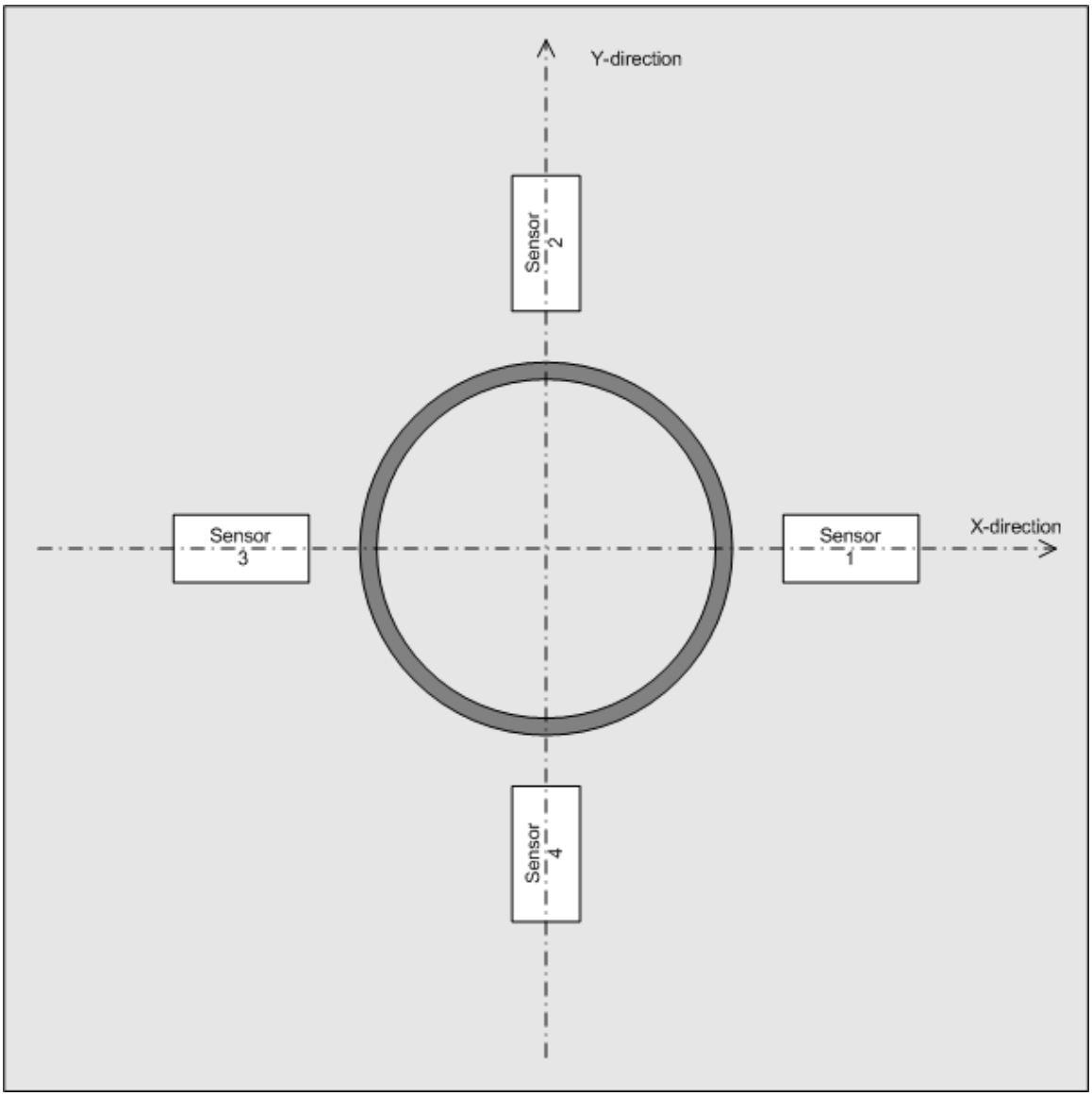


Figure 4.2 Arrangement of Displacement Sensors around the Rotor

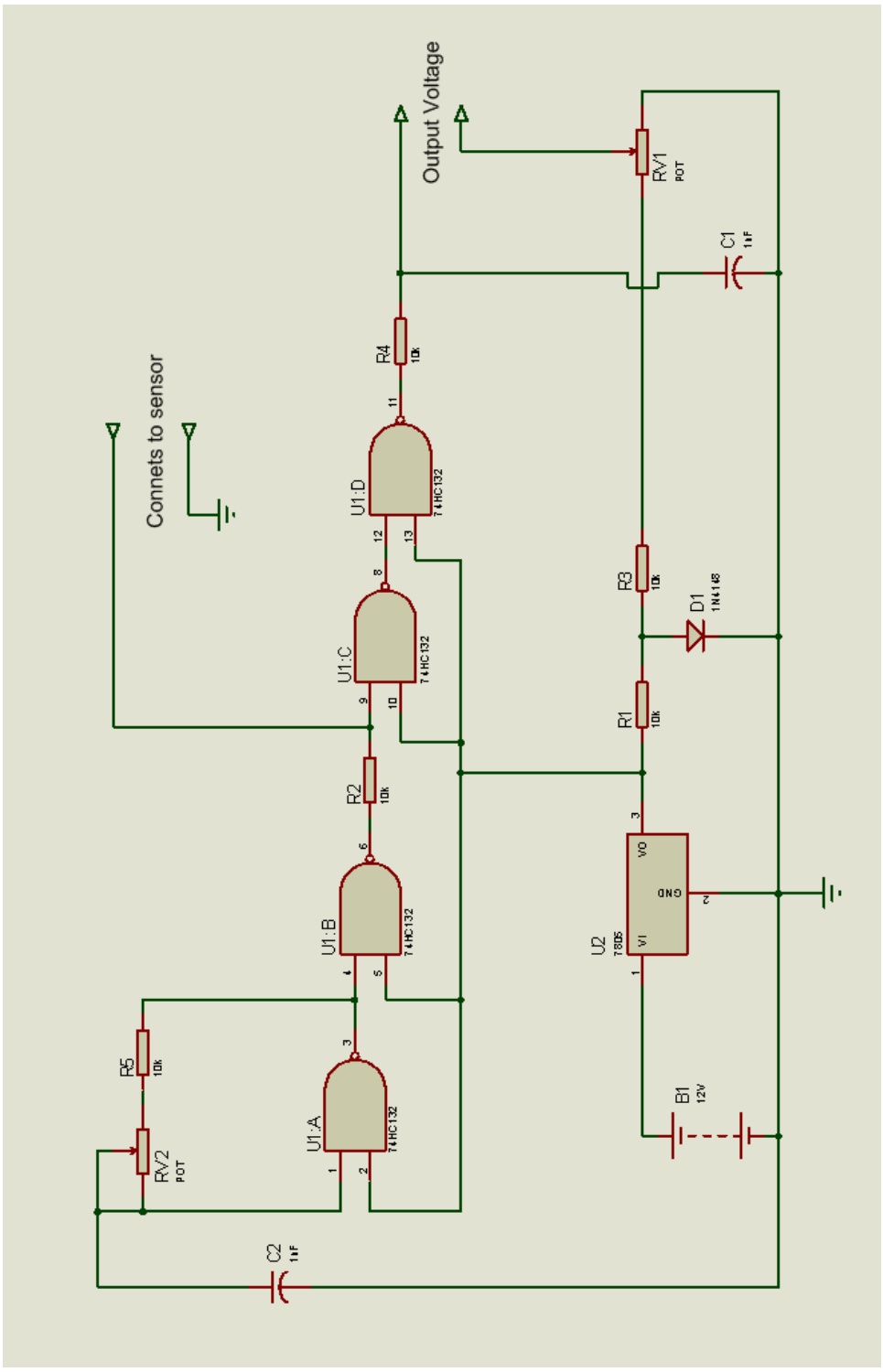


Figure 4.3 Diagram of the sensor reading circuit

4.3 Controller

The control system allows the current in the system to be controlled by feeding back information on the position of the rotor. This is called closed loop feedback control and is necessary for the rotor to be held in a stable position. Feedback control was found to be suitable with this application.

4.3.1 PID Control

A general analog PID controller has the transfer function:

$$P(s) = \frac{U(s)}{E(s)} = K_p + \frac{1}{s}K_i + sK_d$$

But we are using digital methods of implementing controllers, (Computer simulation, and microprocessor implementation); the data is being sampled as discrete instead of continuous signals. This requires using z-transform instead of Laplace-transform. The discrete-time (z-space) PID controller is used. Figure 3.4 shows a discrete-time PID controller.

The main difference between discrete and continuous (analog) PID controllers is that in discrete PID controller a time step has to be defined for the performance calculations, and saturation blocks are added to put maximum and minimum limits for the signals.

For our application, the PID parameters were determined and adjusted using the SIMULINK® model, as explained later.

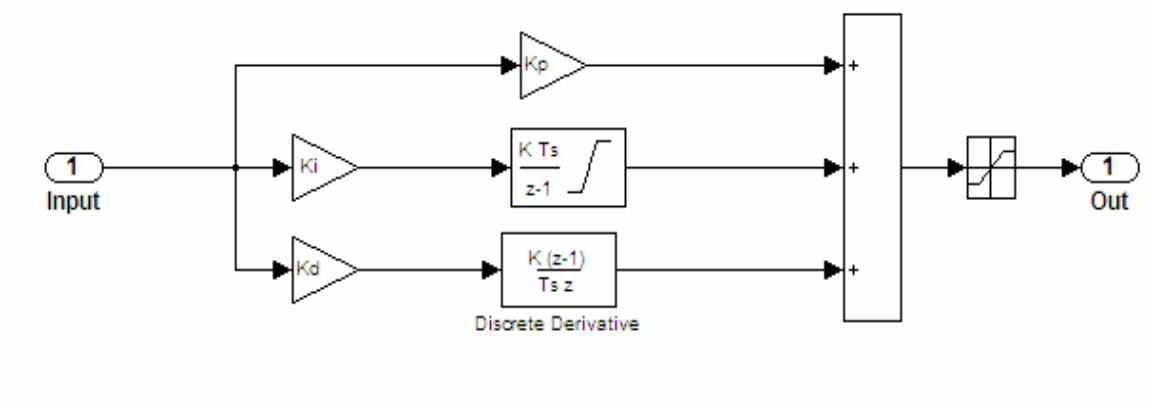


Figure 4.4 Discrete PID controller

4.4 Fuzzy Control

We incorporated fuzzy logic control into the design. Fuzzy logic control offers an easy way to design and tune controller especially for non-linear systems (such as our system). To incorporate fuzzy logic control with the PID controller, two options of control configuration were suggested and tested; both are introduced and explained below.

4.4.1 Fuzzy Controller for Parameter Adaptation

In this configuration a fuzzy controller supervises the operation of the PID controller. The role of the Fuzzy controller is to tune the parameters of the PID controller (the proportional, the derivative, and the integral gains or K_p, K_d, and K_i consecutively), and optimize them according to the operating conditions of the system.

In normal operating conditions, the original parameters of the PID controller are left unchanged, but in abnormal conditions such as overshoots caused by disturbances, these parameters are changed to better control the system with the new disturbances; the

optimized PID parameters are calculated continuously through the fuzzy controller according to the process reaction.

Figure 4.5 shows a block diagram of this configuration. The suggested inputs to the fuzzy controller in this configuration are the set point of the rotor (the desired center of rotation), the rotor position information coming from the sensor of the corresponding axis (from the magnetic spinning process), and the output of the PID controller. All or some of these signals can be used as inputs to the fuzzy controller; each of them has a different weight in affecting the outputs.

The outputs of the fuzzy controller are the corrected values of the PID controller parameters.

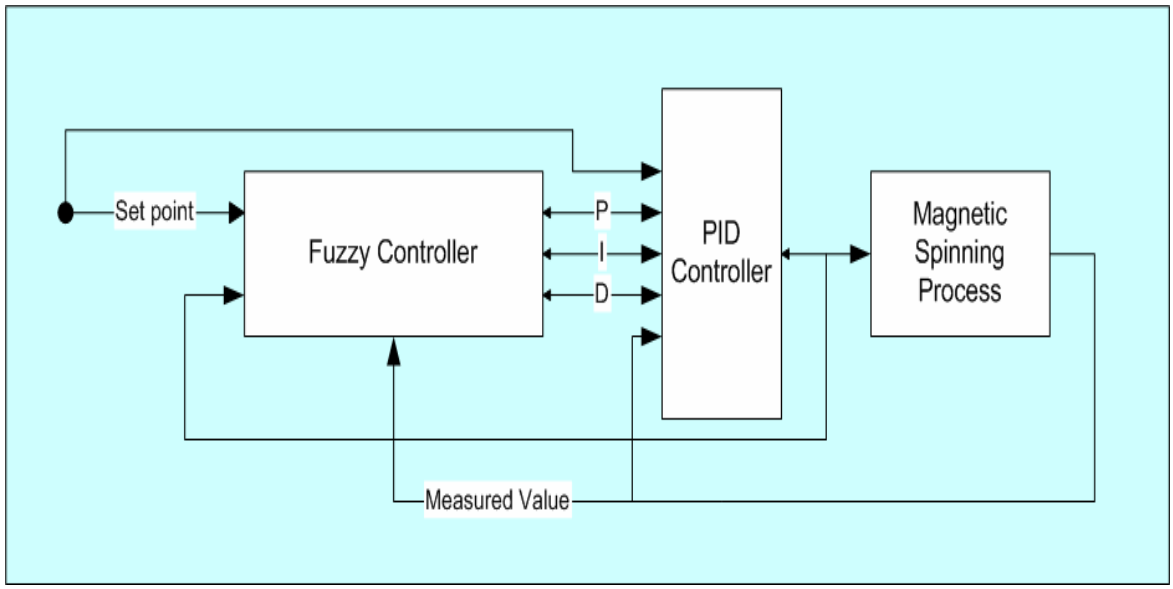


Figure 4.5 Block Diagram of Fuzzy Controller for Parameter Adaptation Configuration

4.4.2 Fuzzy Controller for PID Output Correction

The second option for using Fuzzy control is shown in the block diagram in Figure 4.6. In this configuration, Fuzzy and PID controllers work in parallel as two independent processes each of them takes the set-point and the position information as two inputs, in addition, the fuzzy controller can have a third optional input, which is the output value of the PID controller.

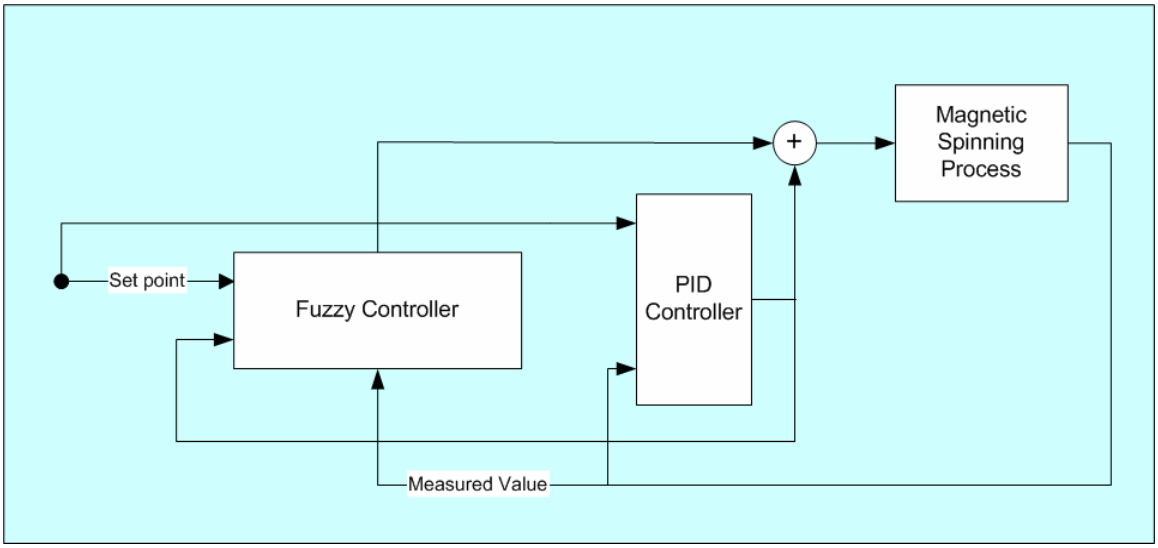


Figure 4.6 Block Diagram of Fuzzy Controller for PID Output Correction Configuration

The PID controller works as designed earlier. The Fuzzy controller is designed so that it has a zero output in normal operating conditions, and in abnormal operating conditions (disturbances, overshoots, long settling time, etc.) it has a nonzero output. The output signals from the two controllers are added together and sent as a control signal to the Magnetic-Ring Spinning machine.

PID controller has the larger effect on the process, i.e. the fuzzy controller output is zero in normal operation conditions, but it intervenes only when abnormalities are detected.

4.5 Implementation of Controller

Microcontrollers offer excellent computational speed needed for real time control with very low cost and low power consumption. We used a PIC18F452 microcontroller which is a 40MHz microprocessor with a 120ns instruction cycle to generate a digital PID controller. It get as input two analog signals from the displacement sensors presenting the displacement of the ring in a two dimensions and generates two control PWM signals, a detailed explanation is introduced later. We developed our program using assembly language and used MPLAP as a simulating environment. MPLAB is a software simulating environment generated by Microchip Technology Inc. Figure 4.7 shows a block diagram of the feedback controller system software [14].

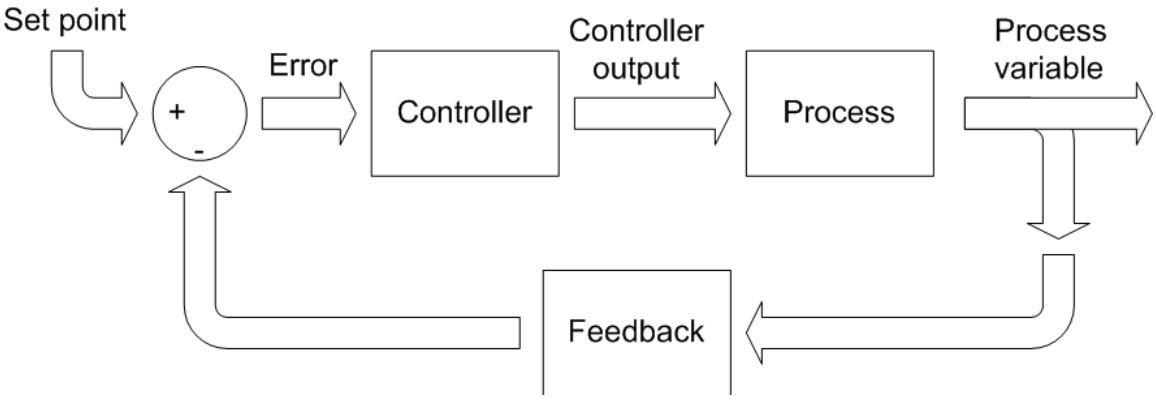


Figure 4.7 Feedback control system

4.5.1 PID Firmware Implementation

In order to implement the Fuzzy-PID controller on the selected microprocessor, we started with a PID control firmware; the fuzzy supervisory control module is added to it afterwards, as discussed in later sections. The PID algorithm itself can be simply implemented. It follows the same structure to the implementation of a PID controller on a PIC18 microcontroller discussed in Microchip Application Note AN937 [14]. It was modified to suit the magnetic ring spinning application. Figure 4.8 a, b, and c show flowchart of the used PID controller, while, names and meanings of variables are summarized in table 4.1 [14]. The main program, reads the error value from an input port of the microprocessor, calls the PID routine, then writes the calculated PID result (controller output) to an output port of the microprocessor.

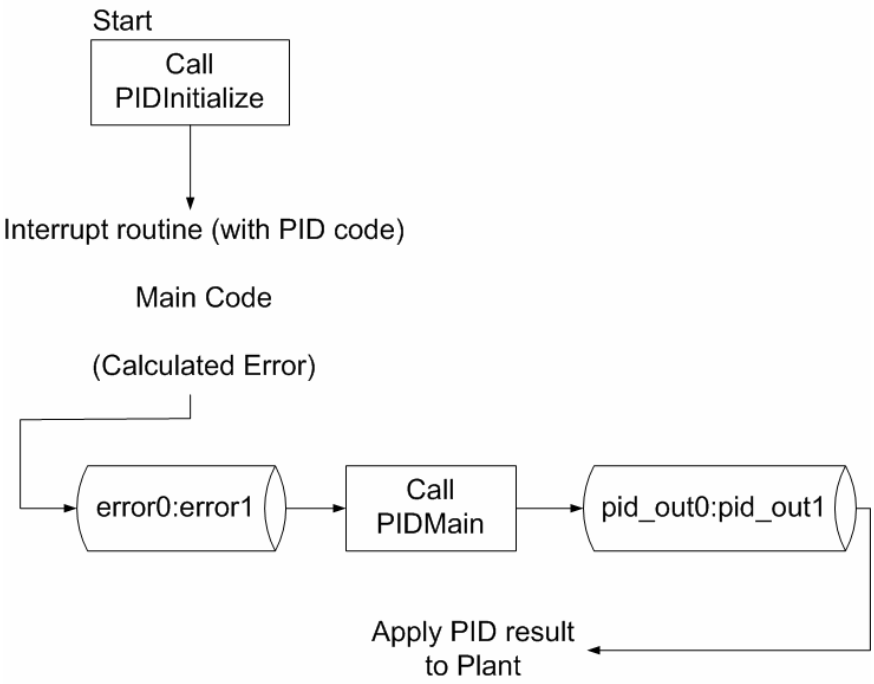


Figure 4.8.a PID firmware implementation

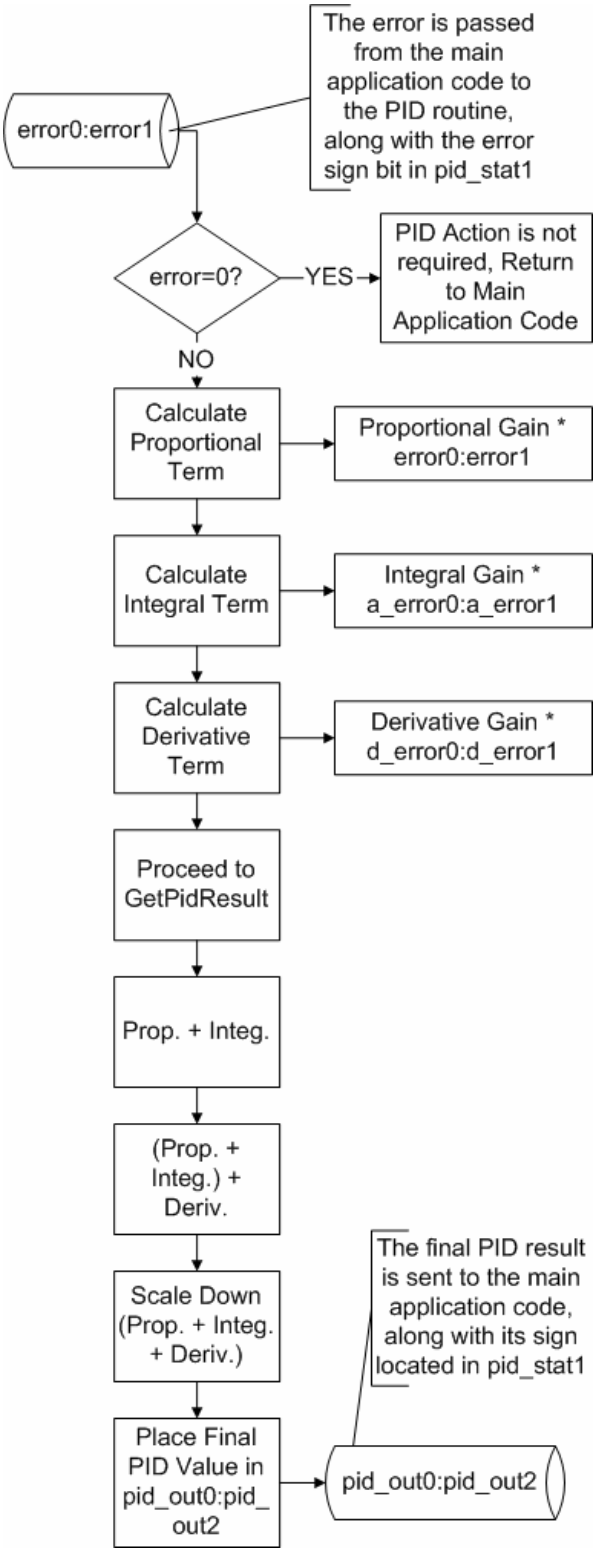


Figure 4.8.b Main PID Routine (PIDMain)

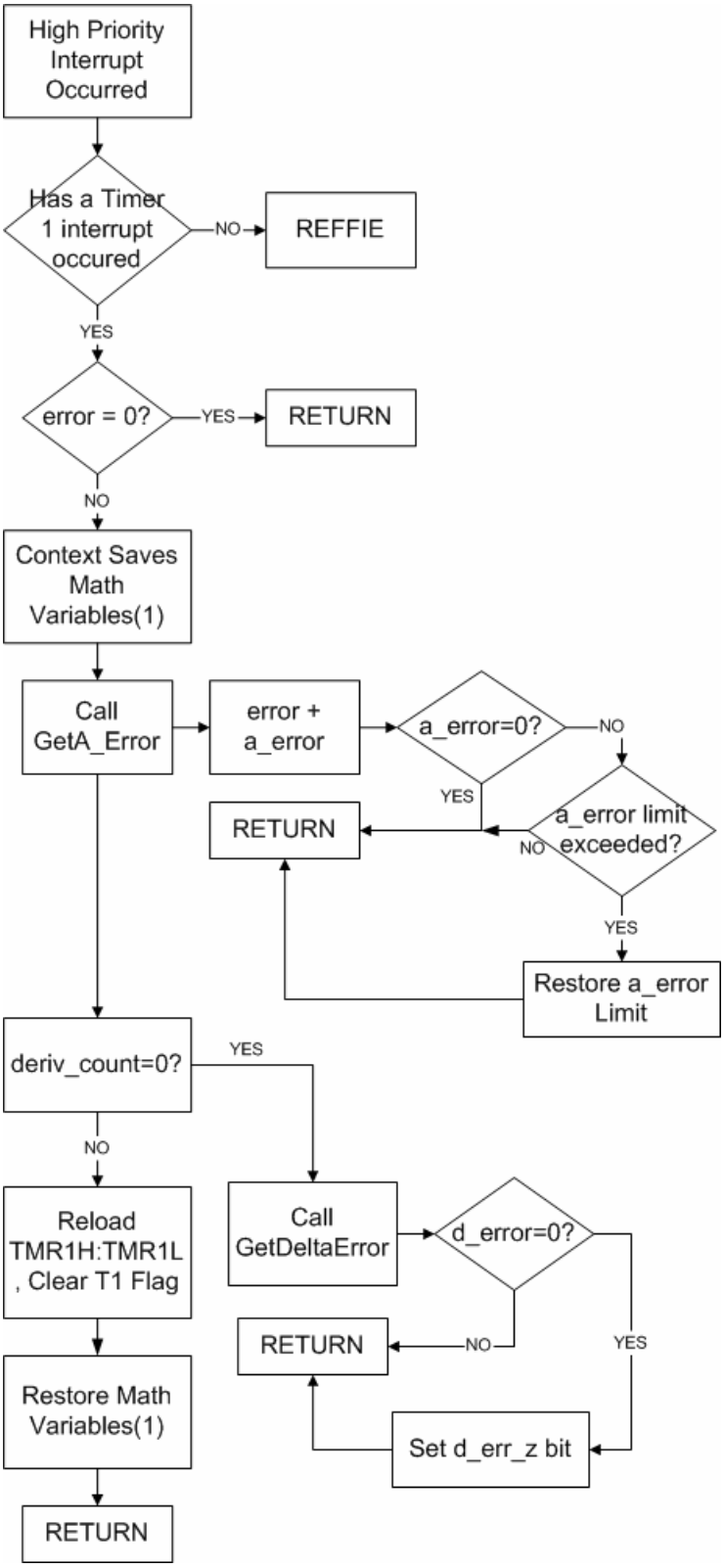


Figure 4.8.c PID Interrupt Routine (PIDInterrupt)

TABLE 4.1: FIRMWARE VARIABLES AND CONSTANTS

Variable/Constant	Type	Definition
error0:error1	Error Variable	Error, difference between the set-point and measured output of the Magnetic system, 16-bit.
a_error0:a_error1	Error Variable	Error Integral, sum of all past errors, 16-bit.
p_error0:p_error1	Error Variable	Value of the last error, 16-bit.
d_error0:d_error1	Error Variable	Error derivative, difference between error0:error1 and p_error0:p_error1, 16-bit.
a_err_1_lim	Error Variable	Constant defining the accumulative error limits, 8-bit.
a_err_2_lim	Error Variable	Constant defining the accumulative error limits, 8-bit.
kd	Gains	Derivative gain, max. = 15, 8-bit.
ki	Gains	Integral gain, max. = 15, 8-bit.
kp	Gains	Proportional gain, max. = 15, 8-bit.
pid_stat1	Status Register	Status bit register 1, 8-bit.
pid_stat2	Status Register	Status bit register 2, 8-bit.
deriv0:deriv2	Terms	Error * Derivative gain; Value of the derivative term, 24-bit.
integ0:integ2	Terms	Error Integral * Integral gain; Value of the integral term, 24-bit.
prop0:prop2	Terms	Error derivative * Derivative gain; Value of the proportional term, 24-bit.
pid_out0:pid_out2	Terms	Final PID results, Sum of the three terms of the PID controller, 24-bit.
timer1_hi	Time Base	8-bit constant loaded into the TMR1H register
timer1_lo	Time Base	8-bit constant loaded into the TMR1L register

Note: In 16-bit variables, the first variable is the Most Significant Byte (MSB), whereas the second variable is the Least Significant Byte (LSB), while in 24-bit variables, the first variable is the MSB, whereas the last variable is the LSB. For example, in the variable *error0:error1*, *error0* = MSB 8-bit and *error1* = LSB 8-bit, and in the variable *deriv0:deriv2*, *deriv0* = MSB 8-bit and *deriv2* = LSB 8-bit [14].

4.5.1.1 Proportional Term

The proportional term is commonly found in control systems. It is calculated by multiplying the proportional gain (*Kp*) times the error value. Therefore, the controller output applied to the plant (magnetic system, in our case) is proportional to the value of the error. Increasing the proportional gain will result in larger changes in the output value of the controller due to changes in error (higher system aggressiveness). Proportional control is used for driving the error to a small value, but the system will not settle to zero steady-state error using proportional control alone. This is the reason integral and derivative control terms are added to the proportional term in various alternatives (PD, PI, or PID controllers).

In the used program, the error is represented in 16-bits, *error0:error1*. The error is multiplied by the proportional gain, *error0:error1 * kp*, using 16 * 16 multiplication routine. The result *prop0:prop2* is 24-bit variable, *prop0:prop2 = kp * error0:error1*.

This value is used later in the code to calculate the total PID controller output (the manipulated variable, or the amount of correction) to be applied to the electromagnets of the Magnetic system.

4.5.1.2 Integral Term

Integral control considers the accumulative error (integration of the error signals, or summation of past errors, in discrete-time systems). In the used program, the value of the integral term is calculated using the sum of all past errors, but at fixed time intervals [14]. Every time the fixed interval expires, the current error at that moment is added to the accumulated error variable. One common issue to consider in integral control is 'windup'. Windup occurs when the accumulative error is largely increased because the Magnetic system output is saturated. This can be avoided by limiting the time period used to calculate the accumulative error, setting limits to the accumulative error itself, or by disabling the execution of the integral term when the Magnetic system output is saturated. As in proportional control, increasing the integral gain (*Ki*) can cause a problem, this issue is known as 'excessive gain'. Excessive gain can cause the system to oscillate, creating system instability. Generally, the integral gain must be tested for all possible situations. In our design, we use fuzzy control to modulate the value of the gain according to the system response, to help avoid such situations.

In the used program, the accumulated error (*a_error0:a_error2*) is calculated by summing past errors. Table 4.2 shows details on how *a_error* is accumulated [14]. Each time the PID routine receives an error; it may or may not be added to the accumulated error variable. This is dependant upon the *Timer1* overflow rate. If *Timer1* overflowed, then the error at that moment will be added to the accumulated error variable. The *Timer1* overflow rate is interrupt driven and is configured as a high priority interrupt. The values for these constants are calculated based on the response of the Magnetic system.

The accumulated error, represented in 16-bits, a*_error0:a_error1,* is multiplied by the integral gain, *a_error0:a_error2 * ki ,* using 16 * 16 multiplication routine. The result *integ0:integ2* is 24-bit variable; *integ0:integ2 = ki * a_error0:a_error1*. This value is also used later in the code to calculate the overall controller output.

TABLE 4.2 '*a_error*' ACCUMULATION EXAMPLE

Time	Error	Timer1 Overflow	Accumulated Error
t = n	10%	No	x%
t = n + 1	8%	No	x%
t = n + 2	12%	Yes	x + 12%
t = n + 3	9%	No	(x% + 12%)
t = n + 4	6%	No	(x% + 12%)
t = n + 5	4%	Yes	(x% + 12%) + 4%
t = n + ...			

To avoid the problem of integral windup, accumulative error limits were set (*a_err_1_Lim:a_err_2_Lim*). When the calculated value of the accumulative error exceeds these limits, the accumulative error is made equal to the value that is determined defined at the beginning of the code.

4.5.3.3 Derivative Term

The derivative term predicts the future behavior of the system output by considering the present and past error to calculate the slope of the error signal over time, or the rate at which the error changes (the derivative of the error with respect to time), and adjusts the output in proportion to this calculated rate.

When the error is constant, the rate of change is zero; therefore, the effect of this term is minimal. There are systems –such as the Magnetic ring spinning system- where adding the derivative term to the proportional and/or integral is needed for better control. The derivative term slows the rate at which the controller output changes, therefore, it is used to reduce the overshoot caused by the integral term and improve stability.

The derivative term is generally calculated by multiplying the derivative of the error signal times the derivative gain (*Kd*). The derivative term is based on the rate at which the system is changing. In the microprocessor firmware, the derivative routine calculates *d_error*; the difference between the current error and the previous error. Another alternative, and sometimes preferred, method to calculating *d_error* would be to find the difference between the current and past values of the system output. The rate at which this calculation takes place is dependant upon the *Timer1* overflow. In the present code the error is used. To keep the derivative term from being too aggressive, a derivative counter variable is used. This allows *d_error* to be calculated once for an '*x'* number of *Timer1* overflows [14]. To get the derivative term, the difference (*d_errro0:d_error1 = error0:error1 – p_error0:p_error1*) is multiplied by the derivative gain (*kd*) providing the 24-bit integral term; *deriv0:deriv2 = kd * d_error0:d_error1*. This term is then added to the previously calculated proportional and integral terms, to determine the final value of the PID controller output.

4.5.2 Fuzzy Firmware

The idea of using *fuzzy control* along with the designed PID controller was introduced earlier. In this section we show how fuzzy firmware was implemented.

4.5.2.1 Fuzzy Firmware Implementation

In this application, we used Inform® Fuzzytech® software to generate an assembly code for the designed fuzzy controller. Figure 4.9 shows a screen capture of the software's editor window, it shows the input variables, rule block, and output variables of the designed fuzzy controller. The membership functions of the input and output variables are shown in figure 4.10, while figure 4.11 shows the rules of the rule block. The input variable *err* is the error, *der* is the derivative of the error (equals *error*, and *d_error* in the PID code). Table 4.3 lists the variables used in the fuzzy controller firmware.

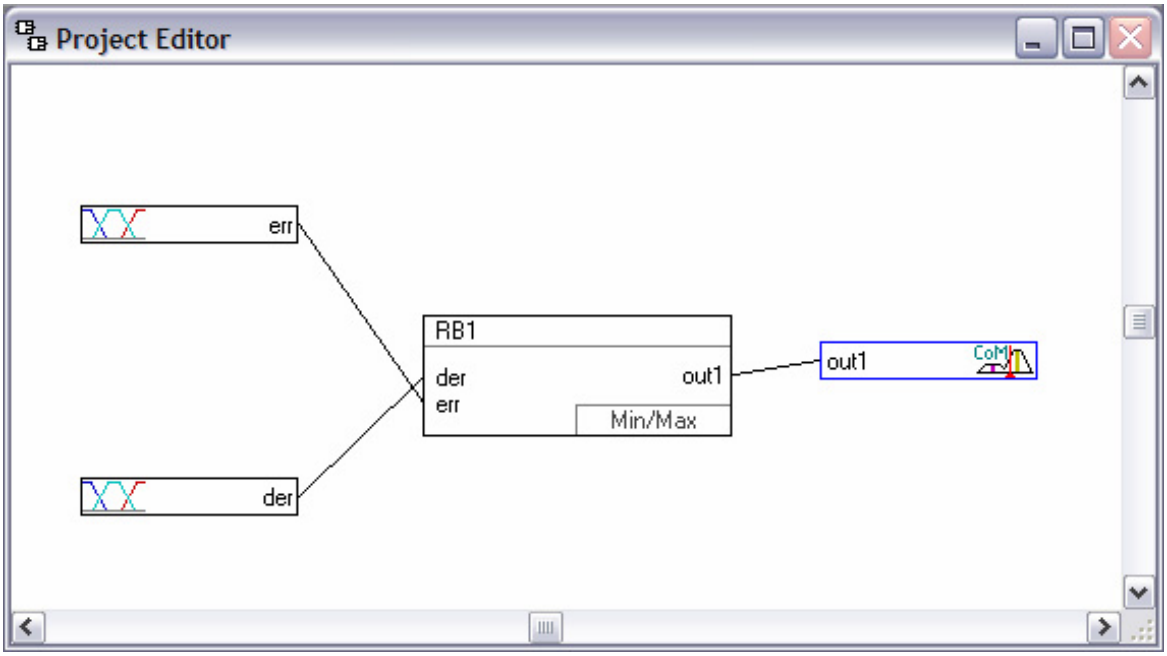


Figure 4.9 Fuzzytech® project editor

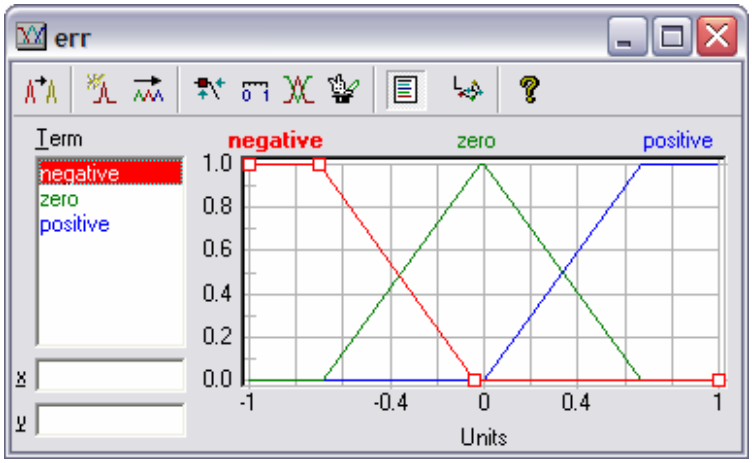


Figure 4.10.a Membership functions of the input variable '*err*'

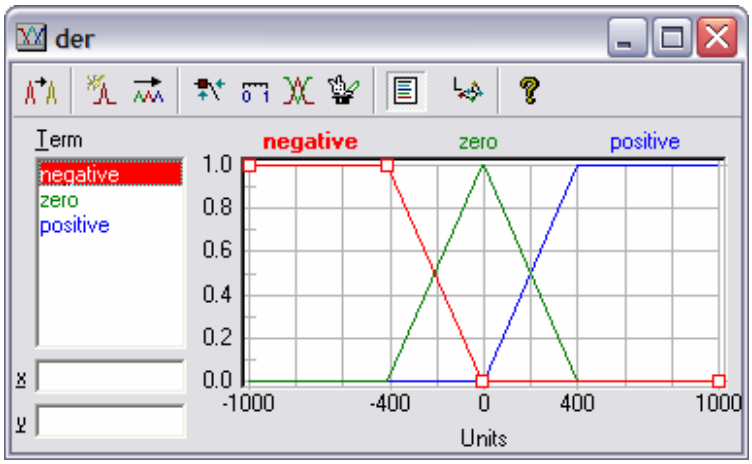


Figure 4.10.b Membership functions of the input variable '*der*'

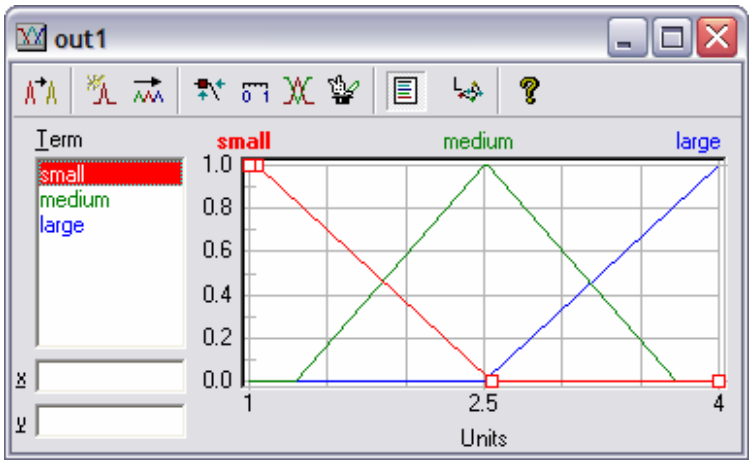
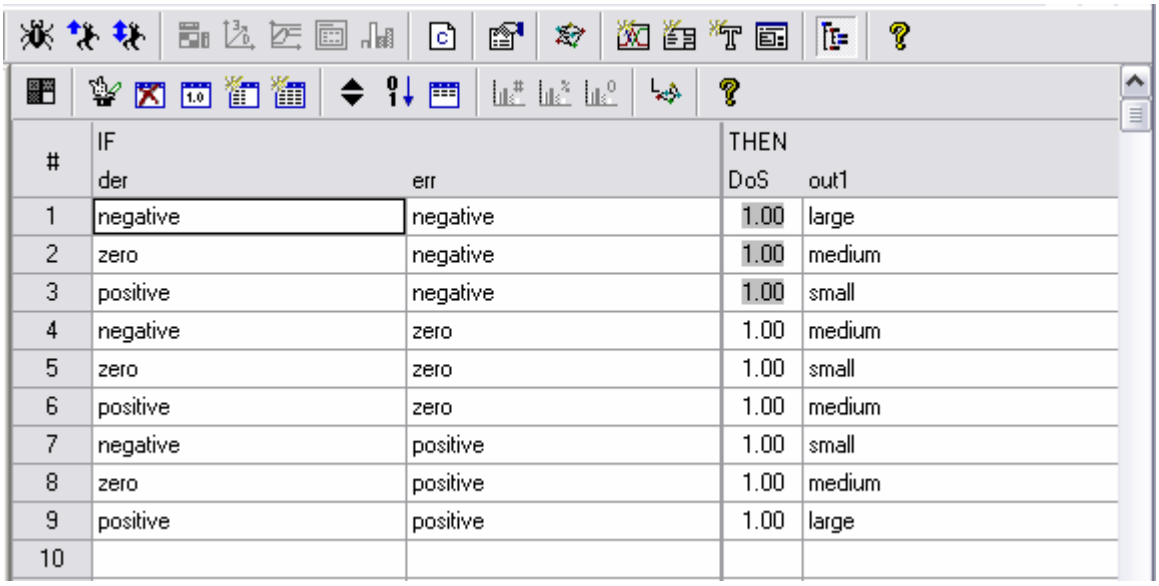


Figure 4.10.c Membership functions of the output variable



#	IF		THEN	
	der	err	DoS	out1
1	negative	negative	1.00	large
2	zero	negative	1.00	medium
3	positive	negative	1.00	small
4	negative	zero	1.00	medium
5	zero	zero	1.00	small
6	positive	zero	1.00	medium
7	negative	positive	1.00	small
8	zero	positive	1.00	medium
9	positive	positive	1.00	large
10				

Figure 4.11 Rules of fuzzy controller

TABLE 4.3 FIRMWARE VARIABLES AND CONSTANTS

Variable/Constant	Type	Definition
*err:err_*	Error Variable	16-bit variable, difference between the setpoint and measured output of the magnetic system (when integrating the fuzzy with the PID firmware, this is made equal to *error0:error1*)
*der:der_*	Error Variable	16-bit variable, the derivative of the error. (Is set equal *d_error0:d_error1*)
*out1:out1_*	Output variable	16-bit variable, the output of the fuzzy controller, is used to change the values of the PID controller parameters

Figure 4.12 shows a flowchart of the fuzzy controller firmware, this flowchart emphasizes option one (Fuzzy Controller for Parameter Adaptation) of the two fuzzy control options explained earlier, since this option –as shown in the next chapters- produces better results.

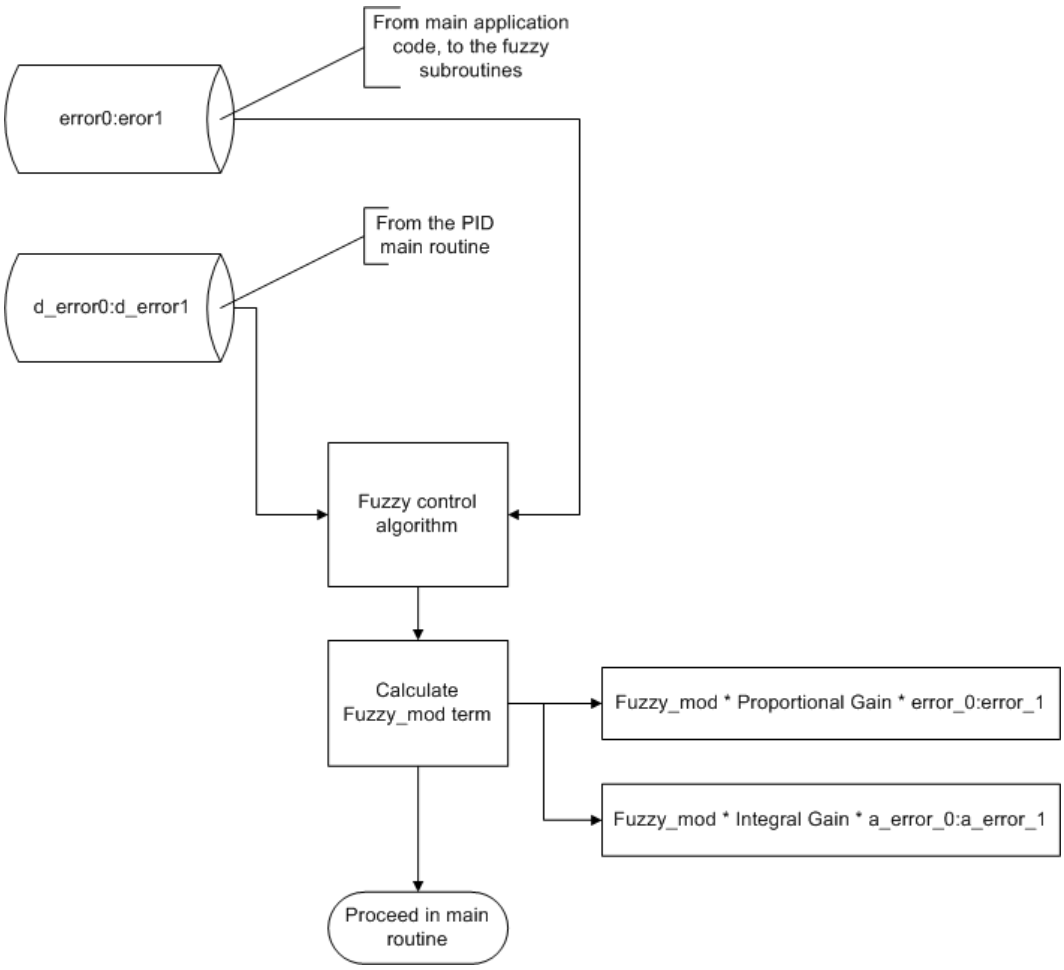


Figure 4.12 Flowchart of the fuzzy controller firmware

4.5.2.2 Fuzzy Control Firmware Generation

Fuzzytech® is used for generating the assembly code for the designed fuzzy controller; however, the software can only generate an assembly code for a limited number of microcontrollers. The microcontroller used in our application is not supported by Fuzzytech®, hence, many changes had to be applied to the generated assembly code; including changes in some instructions, redefining memory organization, and changes in the of the processor settings.

4.6 Conclusion

Control of the developed system is done in single input-single output manner. An Interface circuit was built to read signals from the four radially mounted displacement sensors, and convert them to position information.

PID (Proportional-Integral-Derivative) controller was designed to control the system.

Fuzzy control was used as supervisory controller over the PID controller in two different control schemes. In the first configuration, a fuzzy controller supervises the operation of the PID controller. The role of the Fuzzy controller was to tune the parameters of the PID controller (the proportional, the derivative, and the integral gains or K_p, K_d, and K_i consecutively), and optimize them according to the operating conditions of the system.

In the second configuration, Fuzzy and PID controllers work in parallel as two independent processes each of them takes the set-point and the position information as two inputs, the PID controller works as designed earlier. The Fuzzy controller is designed so that it has a zero output in normal operating conditions, and in abnormal operating conditions (disturbances, overshoots, long settling time, etc.) it has a nonzero output. The output signals from the two controllers are added together and sent as a control signal to the Magnetic-Ring Spinning machine.

To implement the designed controller, PIC18F452 microcontroller was suggested, an assembly firmware for the PID control was used.

The designed fuzzy controller was implemented as an assembly program using FUZZYTECH®; which is developing software for fuzzy control applications, the

generated fuzzy assembly code was modified and implemented into the firmware developed earlier for the PID controller.

5. SIMULATING THE SYSTEM

5.1 The method

A model of the magnetic ring spinning system was built using Simulink®. Simulink® is a software package for modeling, simulating, and analyzing dynamic systems. It supports linear and nonlinear systems, modeled in continuous time, sampled time, or a hybrid of the two. Systems can also be multirate, i.e., have different parts that are sampled or updated at different rates.

The model is shown in Figure 5.1, different blocks in this model represent the components of the designed magnetic spinning system, and this includes the mechanical parts of the system, the electrical circuitry, the sensors, the actuators, and also the effect of air-drag on both the yarn and on the rotor.

5.2 Description of the Model

Below we explain the main items in the Simulink model, the building blocks of the model which are the Simulink blocks are explained in appendix 3.

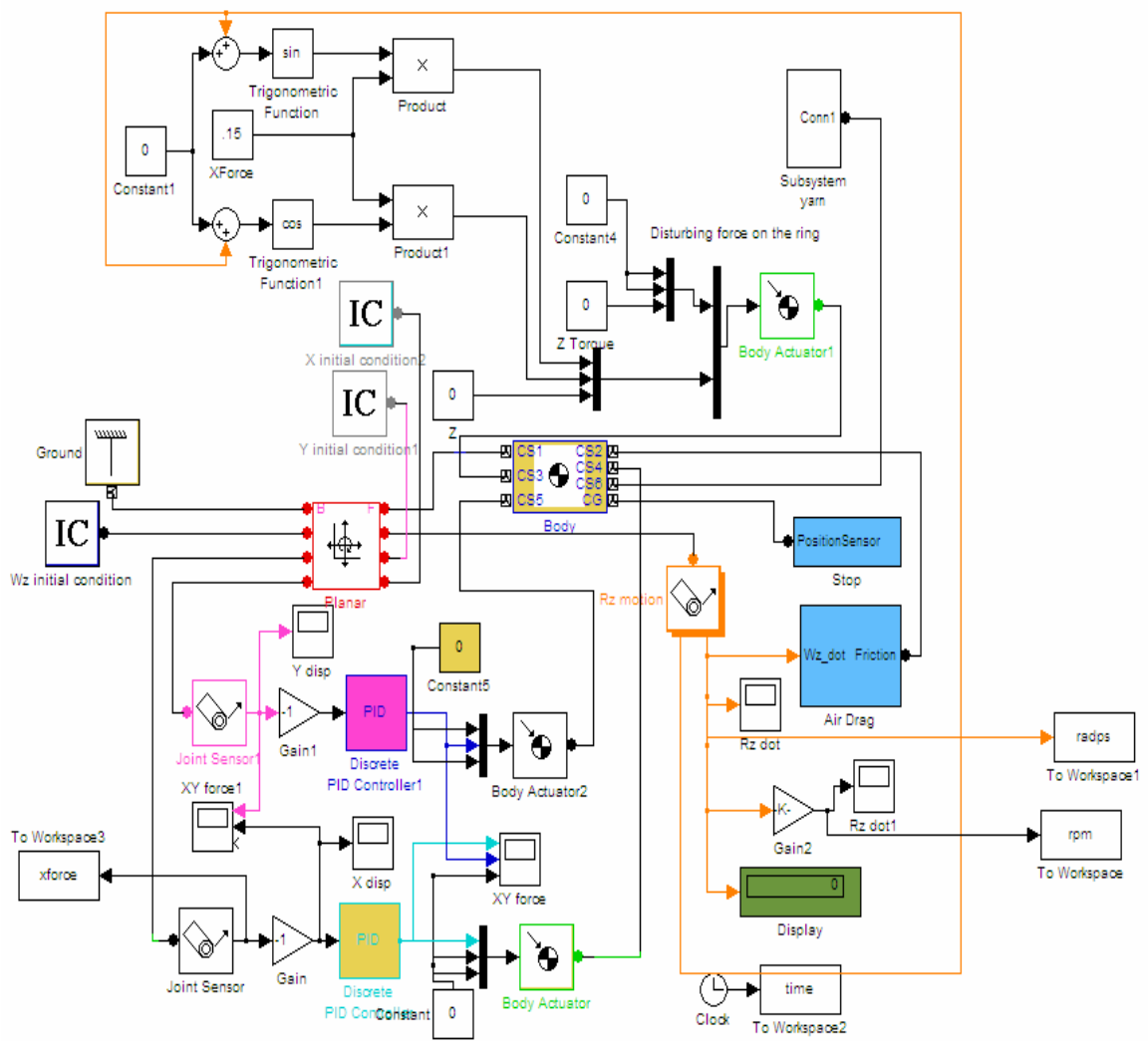


Figure 5.1 The Simulink Model

5.2.1 The Ring

The ring (or the rotor) is a hollow metallic cylinder (model shown in Figure 5.2), during the operation it is suspended in a magnetic field inside the stator, with no contact with any other part of the system, it can rotate freely with no friction, and it replaces the ring and traveler in the traditional spinning system. This is simulated as a cylinder defined with its polar moment of inertia tensor and mass (shown in Figure 5.2). This

cylinder has the ability to spin and to move freely in any direction in the horizontal plane, the degrees of freedom are defined in the model as three degrees of freedom; two axial displacements (X, Y) and one rotational (around Z axis). The initial position and speed can be set to any values.

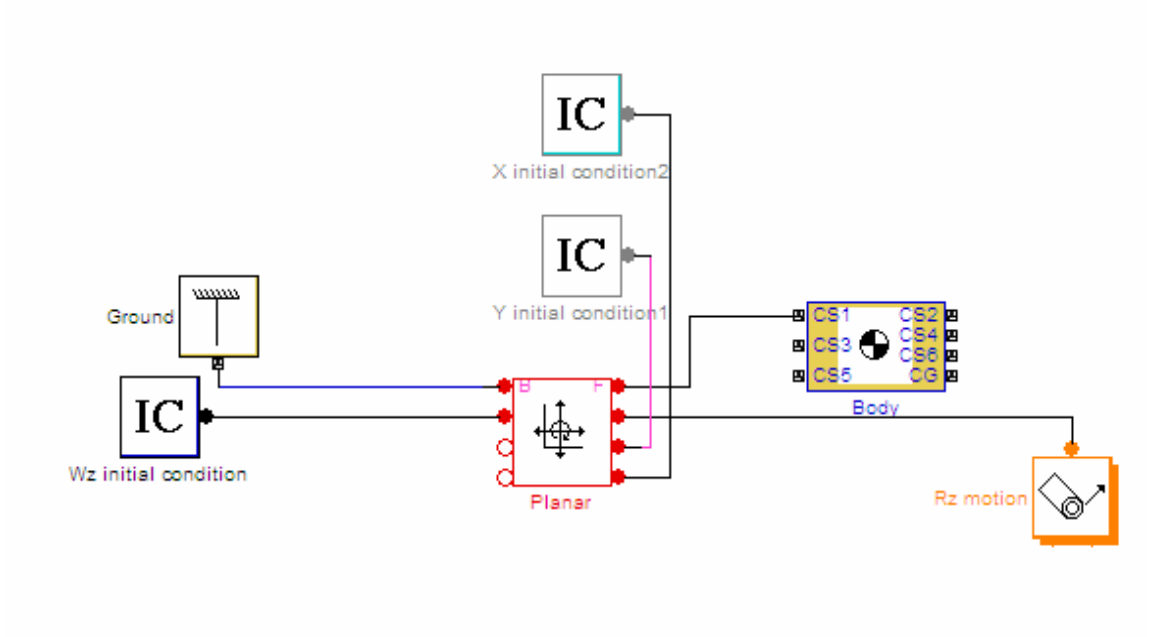


Figure 5.2 The Rotor Model

5.2.2 The Yarn

The yarn is simulated as a finite number of cylinders, shown in Figure 5.3, each two are connected together with spherical frictionless joint to assure the freedom of motion. The cylinders have the density and diameter of an actual yarn. In the simulations presented here, the values used are those of a yarn with a Tex value of seven; diameter equals 0.1 mm and density equals to 1 g/cm3.

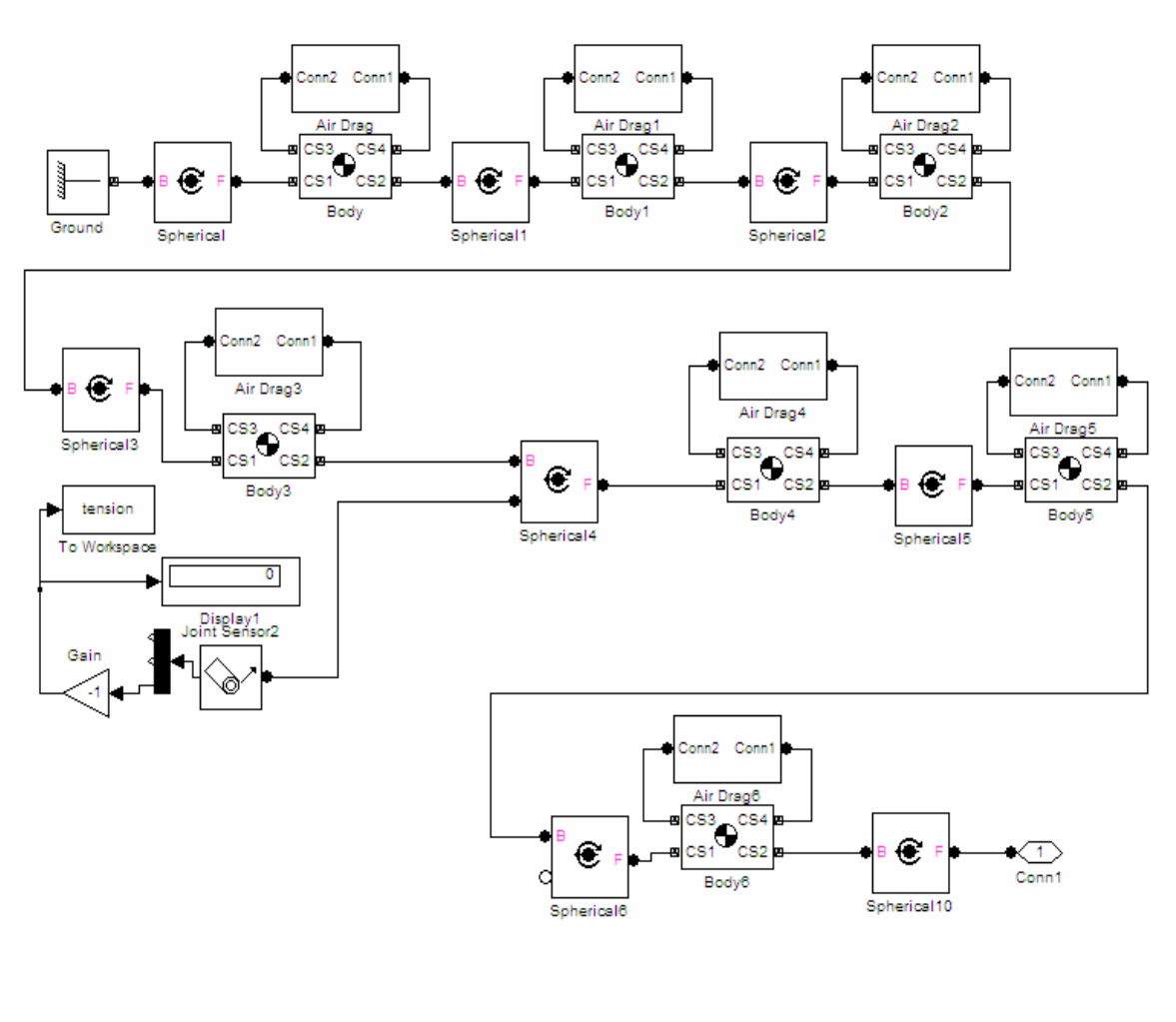


Figure 5.3.a The Simulink Model of the Yarn

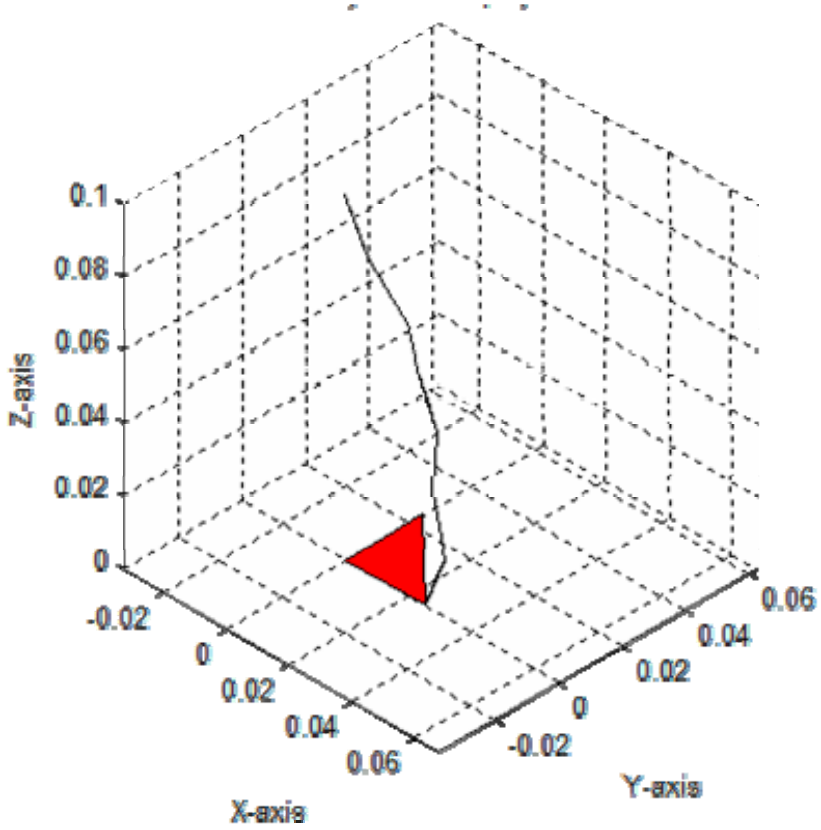


Figure 5.3.b The Simulated Yarn

5.2.3 The Air-Drag

The air-drag plays an important role in the spinning process, the shape of the balloon and the tension of the yarn are both dependant on the air-drag, and so, it had to be considered in the simulation. The air-drag affects both the ring and the yarn. The air drag is simulated as force acting on the yarn and on the ring in the opposite direction of motion.

The Air-drag force is applied to each segment of the yarn and to the rotor. Its magnitude is proportional to the square of the linear velocity of the object. It depends on

the area of the object and on its air-drag coefficient. The air drag coefficient of the yarn was calculated using finite element analysis (FEMLAB software). Figure 5.4 shows the details of an air-drag block, at the input terminal, the velocity of the rotor - measured from sensor- is introduced, and then mathematically processed to produce a force sent through terminal 2 to act on the ring through an actuator.

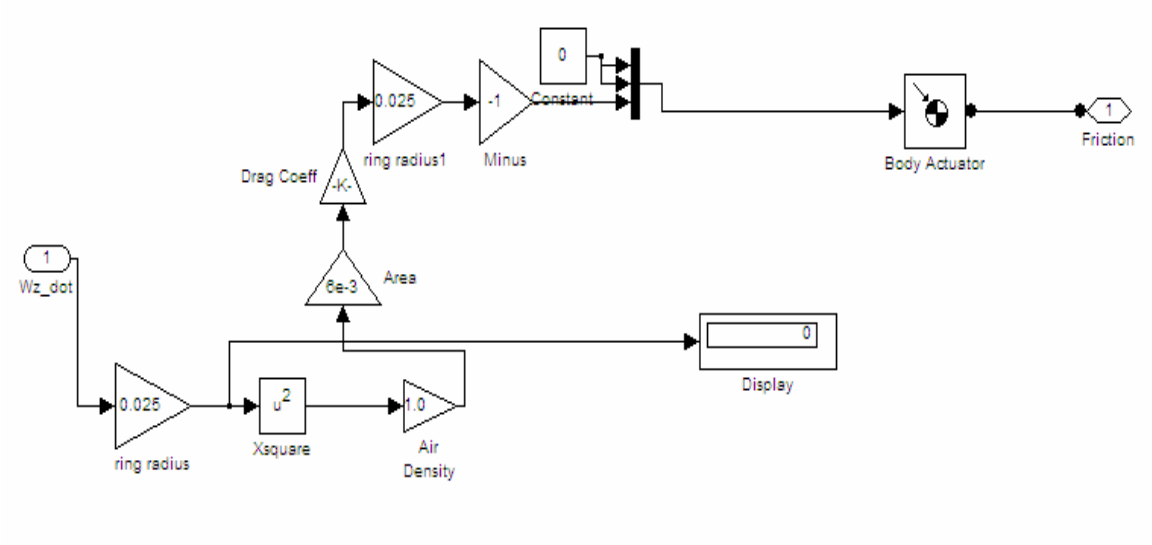


Figure 5.4 Air-Drag Model

5.2.4 Sensors and Actuators

Sensors are used in the model to detect the position and the velocity of the ring, the displacement sensors are parts of the actual system, unlike the velocity sensors that are only used to provide velocity information to the air-drag simulation model. The information about the position is needed so that the controller can accordingly adjust the control signals which are sent to actuators; actuators are the electromagnets responsible on transforming electrical signals to the magnetic field suspending the disc.

5.2.5 The PID Controller

The operation of the system is affected by disturbances; these disturbances along with the yarn tension displace the rotor away from its central position.

A controller is used to keep the rotor in the center by changing the current of the electromagnets according to the measured displacement of the rotor; the attraction forces of the electromagnets change with their currents in a way that brings the rotor to the required position.

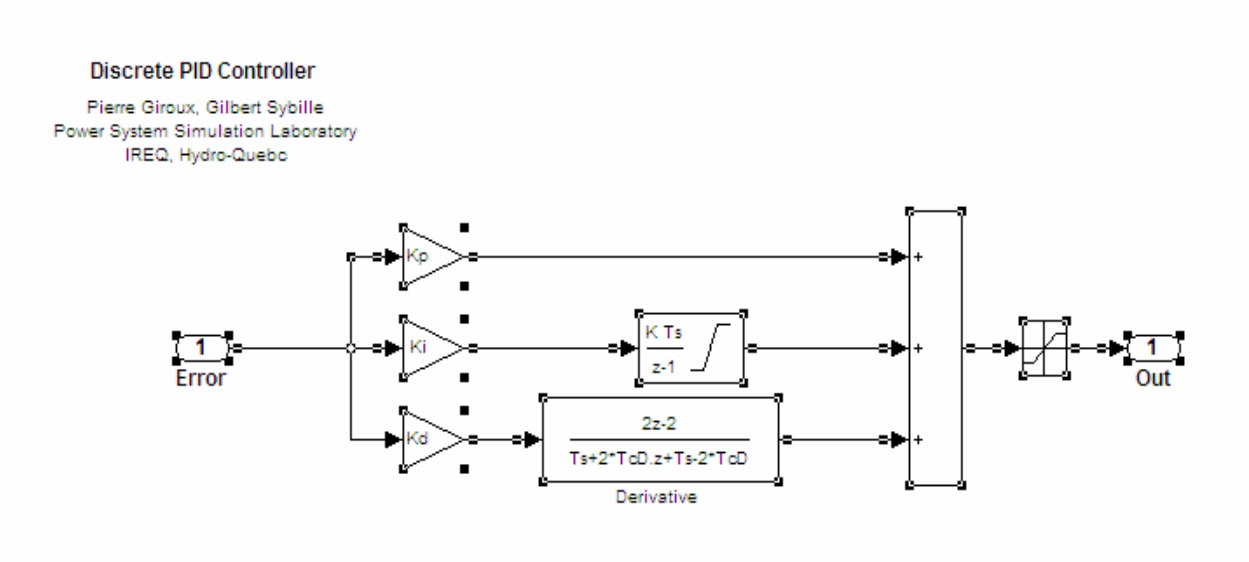


Figure 5.5 Discrete PID Controller

Two identical PID (Proportional, Integral, Derivative) controllers are used in the model; one for each axis of the horizontal plane (X-Y). Although the two controllers are independent, they are connected by the position of the ring; when one controller change the position of the ring in one direction, the other controller will sense any change that in turn happens in the other direction. Block diagram is shown in figure 5, the value of the controller output varies linearly according to the input signal, its derivative, and its

integral, each weighted by a gain, these values are added together to provide the controller output. In our model, the input signal to the controller is "x_i", the distance between an actuator "i", and the surface of the ring. The controller gains "K_p, K_d, and K_i" are initially calculated based on the mathematical model of the system, and later adjusted iteratively according to the results of the simulation. In our work, we found the model of the system to be:

$m\ddot{x}_i = C_1[\frac{(I-i_i)^2}{(L-x_i)^2} - \frac{(I+i_i)^2}{(L+x_i)^2}]$, where C_1, I, and L are constants, and the controller parameters used are: the proportional gain K_p=1.5, the integral gain K_i=4, and the derivative gain K_d=0.1.

5.3 Fuzzy Control

The two options of fuzzy control discussed in the previous chapters were simulated with this model as explained below.

5.3.1 Fuzzy Controller for Parameter Adaptation

Figure 5.6 shows the Simulink model of this control option. The used fuzzy controller has the membership functions shown in figure 5.7. Figure 5.8.a shows the rules of the fuzzy controller, and Figure 5.8.b shows the control surface (the 3D graph of the rules).

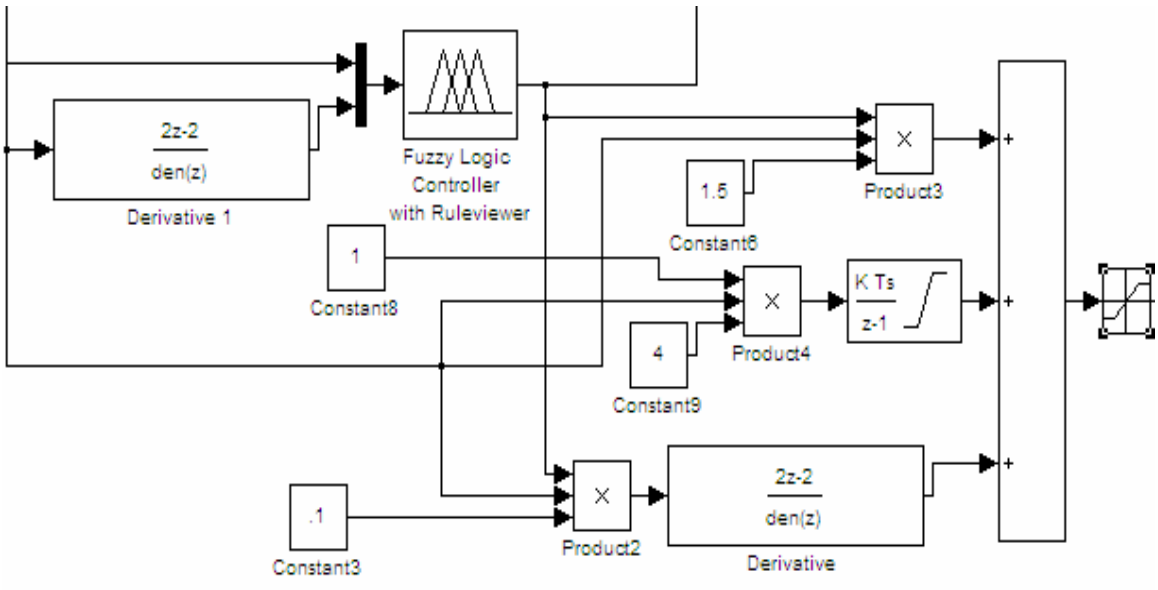


Figure 5.6 Simulink Model of Fuzzy Controller for Parameter Adaptation Configuration

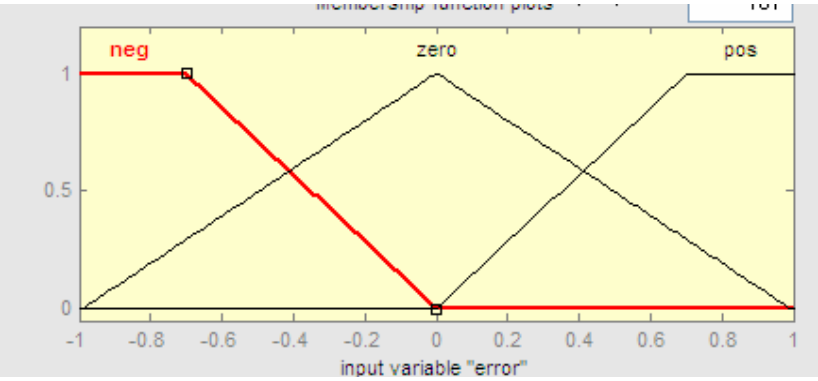


Figure 5.7.a Membership functions of fuzzy input variable '*error*', option 1

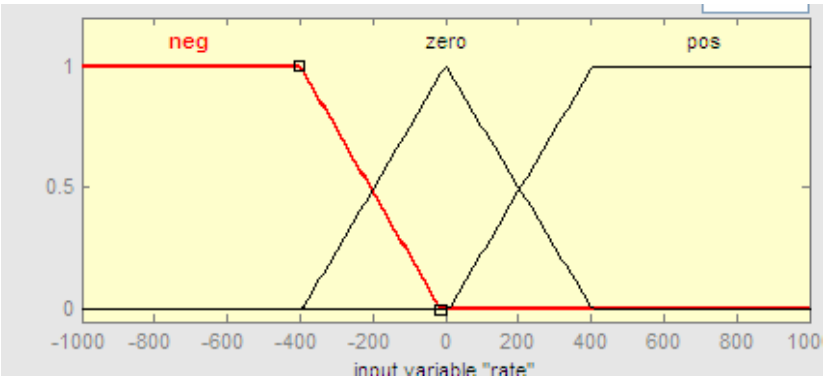


Figure 5.7.b Membership functions of fuzzy input variable '*rate*', option 1

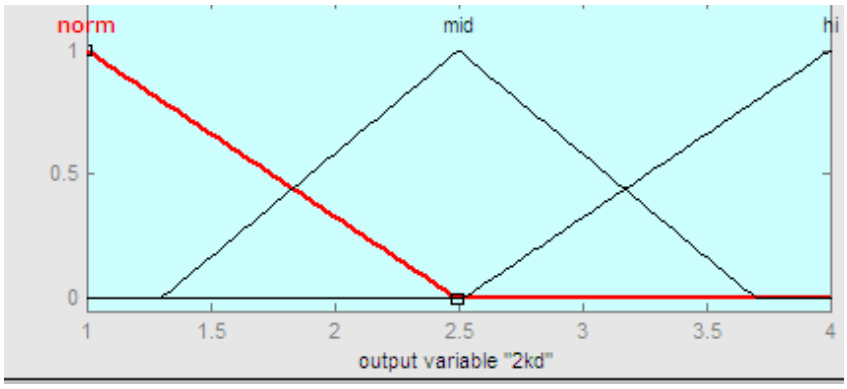


Figure 5.7.c Membership functions of fuzzy output variable '*2kd*', option

1. If (error is neg) and (rate is neg) then (2kd is hi) (1)
2. If (error is neg) and (rate is zero) then (2kd is mid) (1)
3. If (error is neg) and (rate is pos) then (2kd is norm) (1)
4. If (error is zero) and (rate is pos) then (2kd is mid) (1)
5. If (error is zero) and (rate is zero) then (2kd is norm) (1)
6. If (error is zero) and (rate is neg) then (2kd is mid) (1)
7. If (error is pos) and (rate is neg) then (2kd is norm) (1)
8. If (error is pos) and (rate is zero) then (2kd is mid) (1)
9. If (error is pos) and (rate is pos) then (2kd is hi) (1)

1

Figure 5.8.a Fuzzy controller rules, option 1

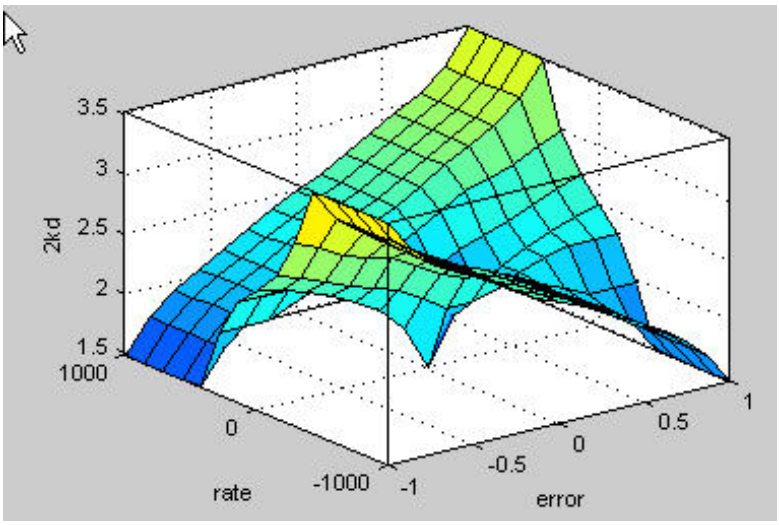


Figure 5.8.b Fuzzy controller surface, option 1

5.3.2 Fuzzy Controller for PID Output Correction

Figure 5.9 shows the Simulink model of this control option.

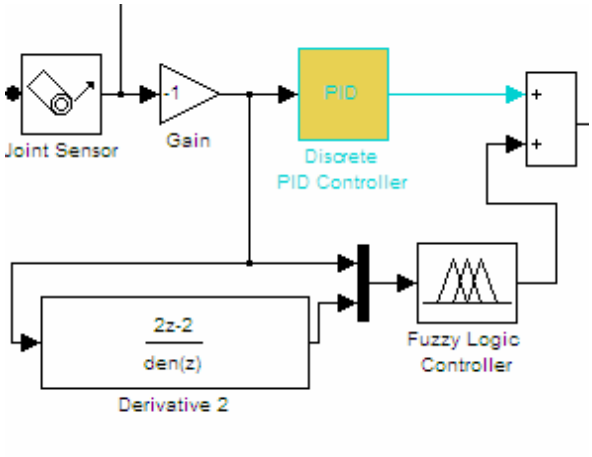


Figure 5.9 Simulink Model of Fuzzy Controller for PID Output Correction Configuration

The used fuzzy controller has the membership functions shown in Figure 5.10, the rules shown in Figure 5.11.a, and the control surface is shown in Figure 5.11.b.

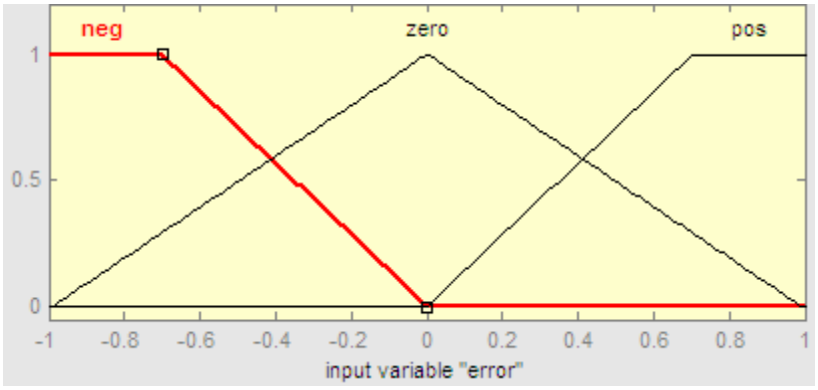


Figure 5.10.a Membership functions of fuzzy input variable '*error*', option 2

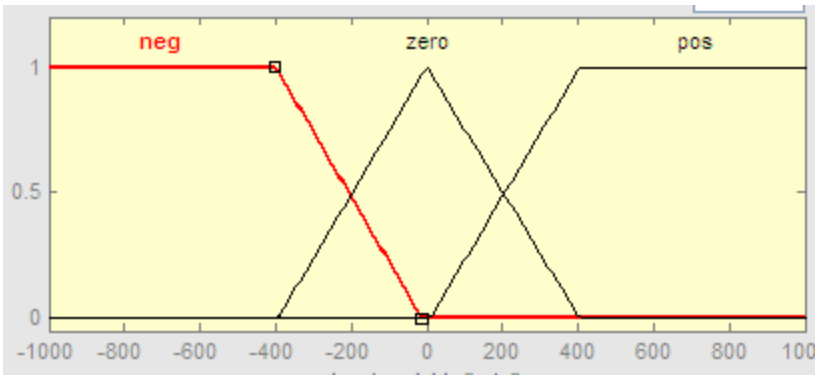


Figure 5.10.b Membership functions of fuzzy input variable '*rate*', option 2

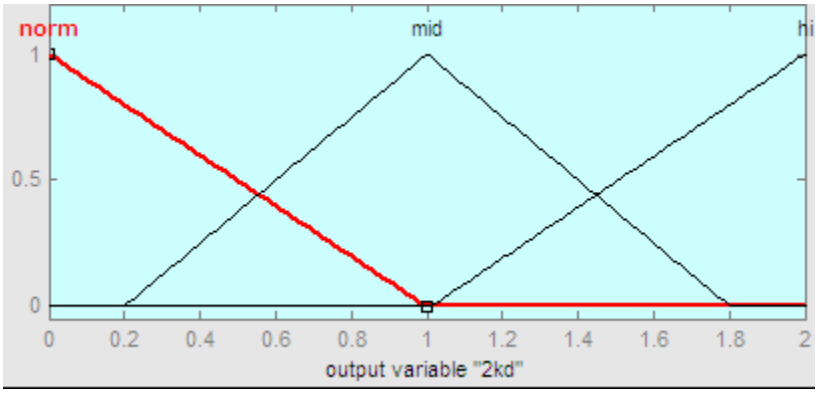


Figure 5.10.c Membership functions of fuzzy output variable '*2kd*', option 2

1. If (error is neg) and (rate is neg) then (2kd is hi) (1)
2. If (error is neg) and (rate is zero) then (2kd is mid) (1)
3. If (error is neg) and (rate is pos) then (2kd is norm) (1)
4. If (error is zero) and (rate is pos) then (2kd is mid) (1)
5. If (error is zero) and (rate is zero) then (2kd is norm) (1)
6. If (error is zero) and (rate is neg) then (2kd is mid) (1)
7. If (error is pos) and (rate is neg) then (2kd is norm) (1)
8. If (error is pos) and (rate is zero) then (2kd is mid) (1)
9. If (error is pos) and (rate is pos) then (2kd is hi) (1)

Figure 5.11.a Fuzzy controller rules, option 2

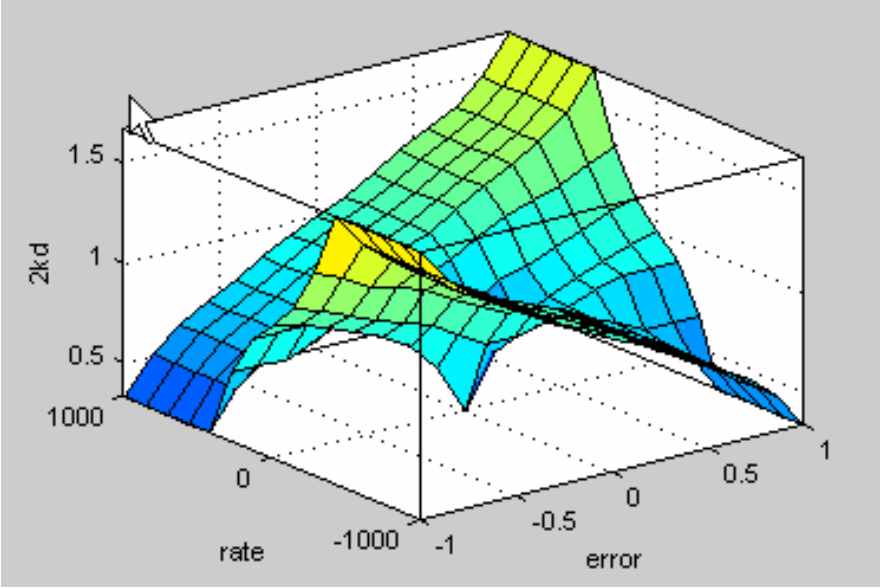


Figure 5.11.b Fuzzy controller surface, option 2

5.4 Microcontroller

An assembly program was written to implement the designed controller on a PIC18F452 microcontroller; the assembly firmware is shown in appendix 4.

To test the program before burning it into a chip, the microcontroller circuit was built on ISIS.

ISIS is a development environment for PROTEUS VSM, a revolutionary interactive system level simulator. This product combines mixed mode circuit simulation, micro-processor models and interactive component models to allow the simulation of complete micro-controller based designs. ISIS provides the means to enter the design in the first place, the architecture for real time interactive simulation and a system for managing the source and object code associated with each project. In addition, a number of graph objects can be placed on the schematic to enable conventional time, frequency and swept variable simulation to be performed.

5.4.1 Microcontroller Circuit

The circuit diagram (from ISIS) is shown in Figure 5.12. The program worked properly with the circuit.

The shown circuit contains the microcontroller, LCD display, two oscilloscopes to examine the signals during the operations of the circuit, and additional circuitry as explained below.

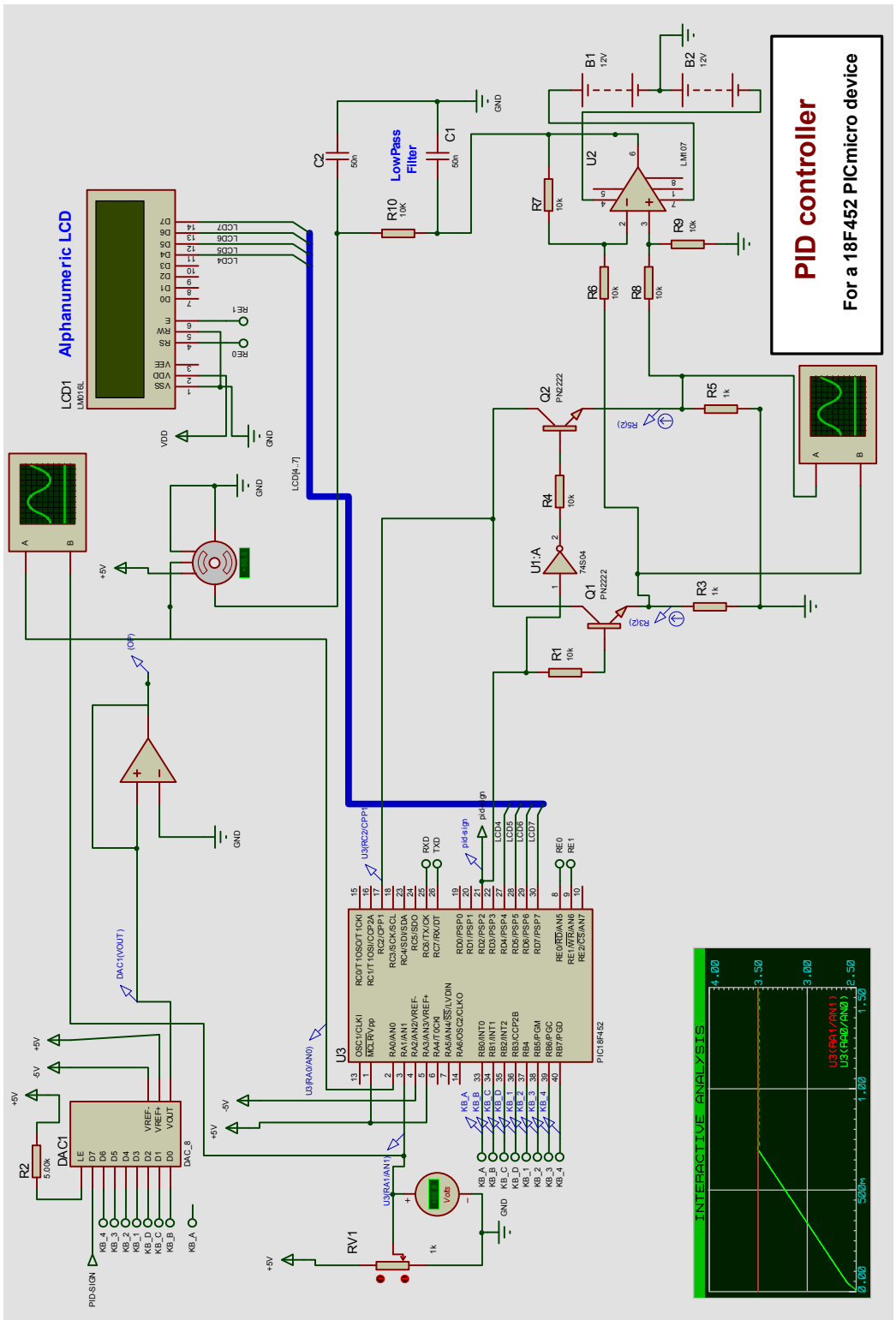


Figure 5.12 ISIS Microcontroller circuit

5.4.2 Circuit Operation

As the microcontroller runs the implemented firmware, the output can be produced in one of two ways; directly as a PWM (Pulse Width Modulated) signal, or its equivalent binary value from the output ports. Both options generate a non-polarized signal, the sign of the signal is calculated and generated as a binary number on a separate output port (High means the signal has a positive sign, and Low means negative sign). A separate electronic circuit was designed for each of these two cases, as explained next.

5.4.2.1 Analog-to-Digital Conversion

When using the option of sending the output of the controller as an 8-bit binary signal, a bipolar ADC (Analog-to-Digital Converter) is used to convert the signal into an analog equivalent value.

To have the control signal as a bipolar signal, the designed circuit works as follows; first, each bit of the binary control byte *'KB_4:KB_1:KB_D:KB_A'*, (see Figure 5.13) is passed through an X-NOR gate (Figure 5.14), with the sign signal *'pid_sign'* as the other input of each gate; if the *'pid_sign'* equals one (positive signal), the X-NOR output byte *'BP_4:BP_1:BP_D:BP_A'* will be the same as the input, and if the *'pid_sign'* equals zero (negative signal), then each bit of the control byte is inverted.

To get the required analog control signal, the least significant bit of the resulted byte is then neglected, leaving 7-bits, these 7-bits are passed too the ADC as its least significant input, with the *pid_sign* signal as the most significant input. Table 5.1 is a truth table that shows six different cases of the operation. The output signal of the ADC in this case may require amplification and low pass filtering.

Table 5.1 Truth table

pid_sign	*pid_sign* (binary value)	*Control byte**	(*pid_sign*) XNOR (*Control byte*)	Output byte: *pid_sign:[7 most significant bits of Control byte]*	Decimal value	Output equivalent analog value
Positive	1	:11111111	:11111111	:1:1111111	255	5.000
Positive	1	…	…	…	…	…
Positive	1	:00001111	:00001111	:1:0000111	135	0.294
Positive	1	…	…	…	…	…
Positive	1	:00000000	:00000000	:1:0000000	128	0.020
Negative	0	:00000000	:11111111	:0:1111111	127	-0.020
Negative	1	…	…	…	…	…
Negative	0	:00001111	:11110000	:0:1111000	120	-0.294
Negative	1	…	…	…	…	…
Negative	0	:11111111	:00000000	:0:0000000	0	-5.000

*Control byte is *(BP_4:BP_1:BP_D:BP_A)*

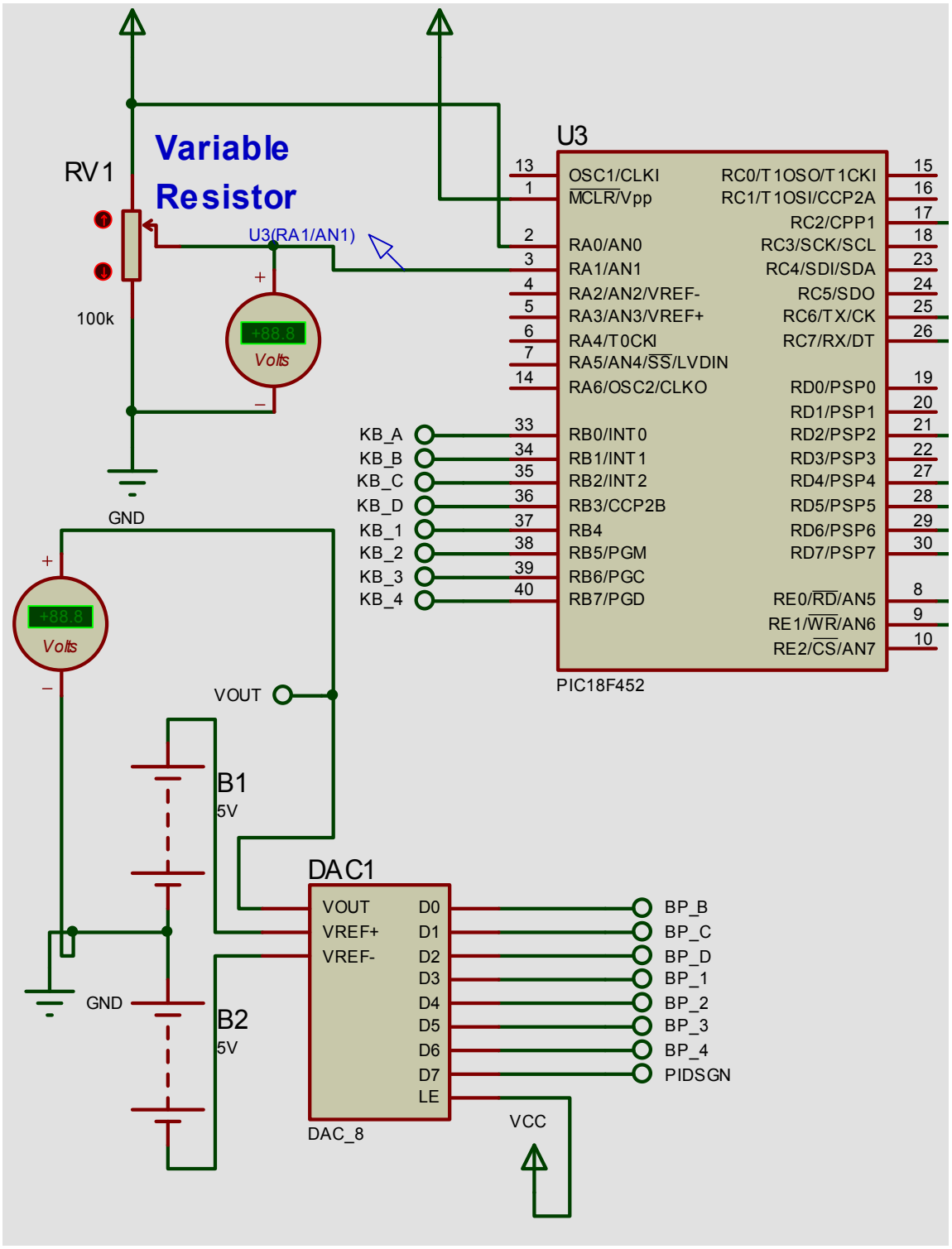


Figure 5.13 Analog-to-Digital Converter

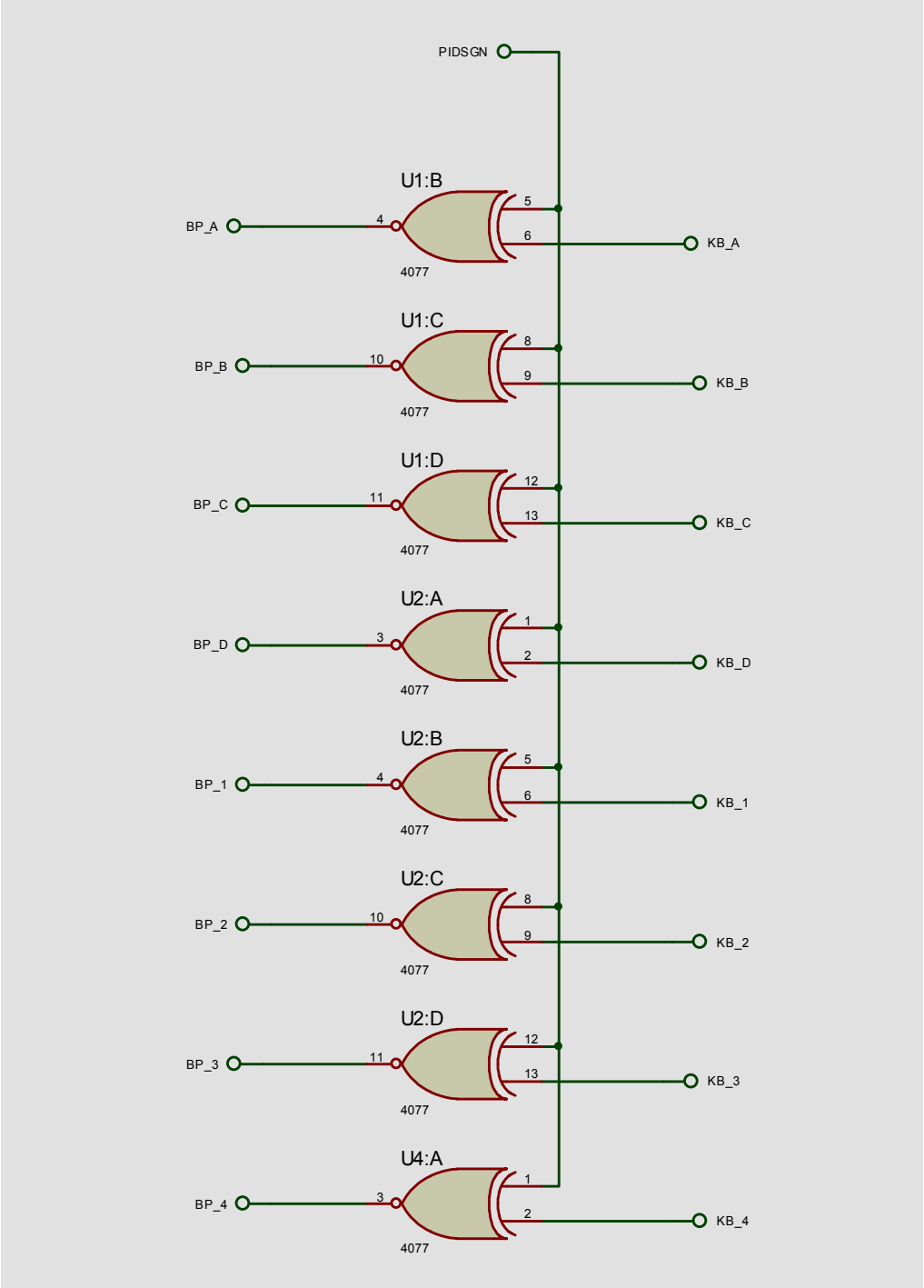


Figure 5.14 X-NOR gates

The disadvantages of this method are:

1. Since the output of the ADC is bipolar and is the analog equivalent of its input, and there is an even number of input combinations, the output can never be absolute zero.
2. There is a loss of information resulted from neglecting one of the bits.

5.4.2.2 Transistor Switching

The second method involves using two BJT's (Bipolar Junction Transistors) as switches controlled by the *'pid_sign'* signal to change the polarity of the PWM signal generated from the microcontroller.

Figure 5.15 shows circuit diagram of the method, the PWM signal is connected to the collectors of the two transistors Q1, and Q2, while the *'pid_sign'* signal is connected directly to the base of Q1, and its inverse is connected to the base of Q2, the output signals of the two transistors are taken from their emitters, and connected to the input pins of a subtracting operational amplifier circuit:

OpAmp_output = Q1_output – Q2_output

Table 5.2 below summarizes the operation of the circuit.

Table 5.2

pid_sign	***Q1 status***	***Q2 status***	***Q1 output***	***Q2 output***	***OpAmp output***
High	On	Off	PWM	0	PWM
Low	Off	On	0	PWM	- PWM

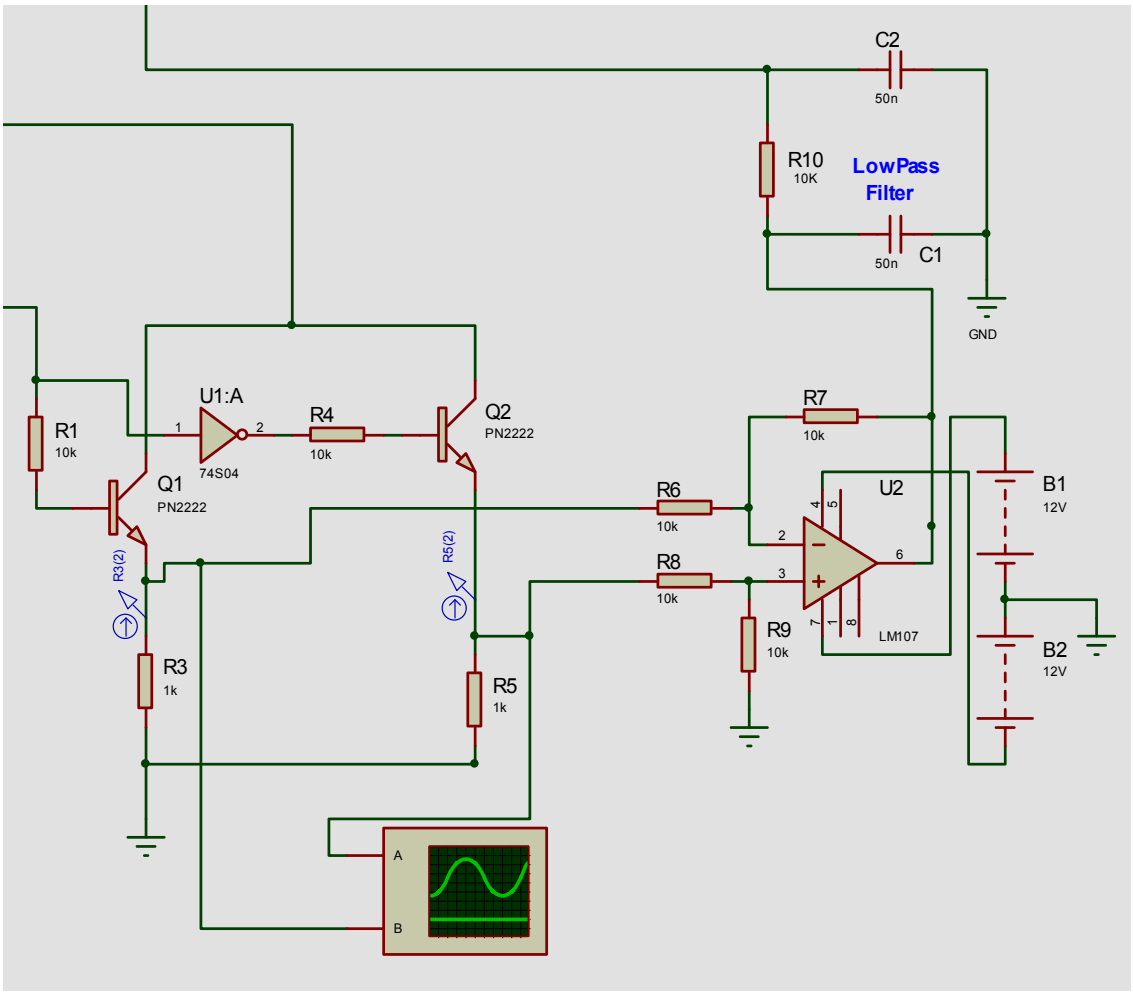


Figure 5.15 Transistor switching circuit

5.4 Conclusions

A model of the magnetic ring spinning system was numerically simulated using Simulink. Different blocks in this model represent every component of the designed magnetic spinning system, this includes the mechanical parts of the system, the electrical circuitry, the sensors, the actuators, and also the effect of air-drag on both the yarn and on the rotor; these parts are explained below.

The designed PID and fuzzy controllers were implemented in this model.

ISIS software was used to simulate the microcontroller with the designed firmware and the control circuit.

Two circuits were designed for the proper operation of the Analog-to-Digital conversion in the ISIS model; the first method involved using logic circuits, and had some disadvantages. The second method used BJT transistors for switching and performed better than the first method.

6. EXPERIMENTAL WORK

6.1 Introduction

In this chapter we explain the experimental work done towards the projects. We started the research by designing a magnetically levitated ball system to study the components used in our future research.

This is a list of the equipments used in the research:

1. Oscilloscope, 100 MHz Mixed-Signal Oscilloscope with two scope (analog) channels and 16 logic timing (digital) channels.
2. Multimeter, 6.5 digit resolution with direct access from PC applications to the instruments to set up and take measurements.
3. DC power supply, triple output, dual tracking, DC 2 0-60V/0-3A Outputs, 1 5V/5A Output Digital DC Power Supply.
4. DC power supply, triple output, dual tracking, two 0-30 VDC, 2A outputs capable of independent series or parallel operation, one 4-6 VDC, 5A Output.
5. Signal Generator, 15 MHz function/arbitrary waveform generator, 12-bit, 40 MSa/s, 16,000-point deep arbitrary waveforms.
6. PC, Pentium II processor, 400 MHz.
7. Hot air soldering system.
8. EPROM eraser holds up to four windowed chips of any size.

9. LAB-X1 pre-assembled microprocessor evaluation board for PICmicro microcontrollers.
10. Microchip development system including: MPLAB software package, PICmicro microcontrollers Programmer, and 68 pin Adapter.

Experimental work also included studying and programming various microcontrollers, building a circuit for reading the sensors position signals, examining the response of the different components of the design, and building a prototype of the system, as explained in details below.

6.2 Magnetically Levitated Ball System

Magnetic ball levitation system is a simple magnetic suspension system, in which, a ball is levitated using one electromagnet, only one sensor is needed to sense the height of the ball, and the control system is required for only one direction (the Z-axis). See

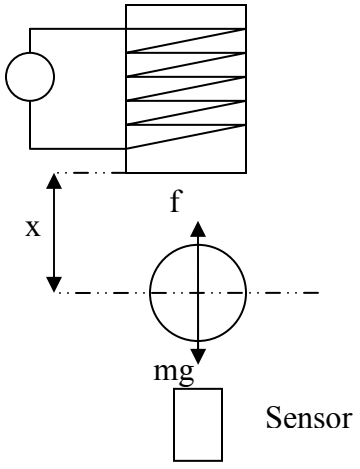


Figure 6.1 Magnetic Ball Levitation

Figure 6.1 for illustration.

We started the research by designing a magnetically levitated ball system, in order to study the sensors, equipments, and the electromagnets used in our future research.

6.2.1 System Response

For the shown configuration for the magnetic ball-suspension system, the electromagnetic force affecting the ball can be approximated by the equation:

$f=ki^2/x$

6.2.2.1 Transfer Function Analysis

The force equation of the system is:

$k\frac{(I_0-i)^2}{X_0-x}=mg+mx''$, where, X_0 and I_0 are the initial ball position and the initial current, respectively.

$$k\frac{I_0{}^2(1+2\frac{i}{I_0}+\frac{i^2}{I_0^2})}{X_0(1-\frac{x}{X_0})}=mg+mx''$$

For small $i/I_{0\ and}\ x/X_0$, the equation can be approximated as:

$$k\frac{I_0{}^2}{X_0}(1+2i/I_0)(1+x/X_0)=mg+mx''\approx k\frac{I_0{}^2}{X_0}(1+2i/I_0+x/X_0)$$

And we know that: at balance, x"=0, therefore: $mg=k\frac{I_0{}^2}{X_0}$

Hence:

$$mx'' = k\frac{{I_0}^2}{X_0}(2i/I_0 + x/X_0)$$

Taking the Laplace transform:

$$mXs^2 = k\frac{{I_0}^2}{X_0}(2I/I_0 + X/X_0)$$

$$\frac{X(s)}{I(s)} = 2k\frac{I_0}{X_0}\frac{1}{s^2 - k\frac{{I_0}^2}{mX_0^2}}$$

And from (1)

$$\frac{X(s)}{I(s)} = \frac{2\sqrt{\frac{gk}{mX_0}}}{s^2 - \frac{g}{X_0}}$$

And we have: v= i*R + L*i'

And $I_0=V_0/R$, hence:

$$\frac{V(s)}{I(s)} = L(s + \frac{R}{L})$$

So, the transfer function of the system becomes:

$$\frac{X(s)}{V(s)} = \frac{\frac{2}{L}\sqrt{\frac{gk}{mX_0}}}{(s^2 - \frac{g}{X_0})(s + \frac{R}{L})}$$

The pole at -R/L, is a very fast staple pole, and can be excluded from the calculations; the transfer function can be approximated as:

$$\frac{X(s)}{V(s)} = \frac{\frac{2}{L}\sqrt{\frac{gk}{mX_0}}}{(s^2 - \frac{g}{X_0})}$$

6.2.2.2 System Parameters

The parameters of the components used in the designed system have the following values:

- Resistance of the electromagnet coil, R = 176 Ohm
- Inductance of the electromagnet coil, L = 0.553 H
- Mass of the suspended ball, m = 5.6 g

Table 6.1 below shows some data taken experimentally from the system, and used to determine the constant *k* in the model.

Table 6.1 Finding k experimentally

r	m (g)	m (kg)	g (kg.m/s2)	y (mm)	y (m)	I (mA)	I^2 (A^2)	$k=mgy/i^2$
0.39	5.6	0.0056	9.8	7.58	0.00758	61	0.003721	0.111795
0.54	5.6	0.0056	9.8	7.43	0.00743	56.7	0.003215	0.126834
0.56	5.6	0.0056	9.8	7.41	0.00741	56	0.003136	0.129675
0.77	5.6	0.0056	9.8	7.2	0.0072	54	0.002916	0.135506
1.12	5.6	0.0056	9.8	6.85	0.00685	47	0.002209	0.17018
1.2	5.6	0.0056	9.8	6.77	0.00677	45	0.002025	0.183475

Using the data from Table 5.1, the graph shown in Figure 6.2 was plotted, and the average value of *k* was found.

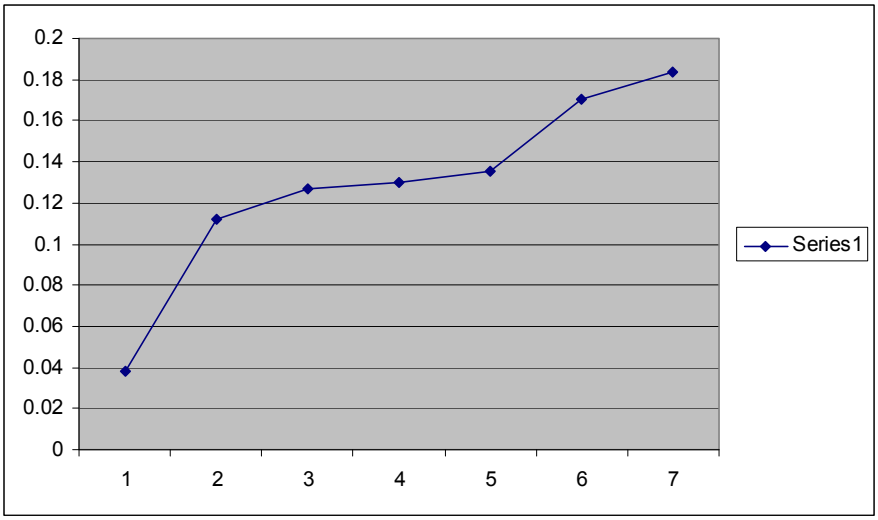


Figure 6.2 Calculating K

After excluding the extreme values, k is approximately 0.13 Nm/A^2

Using these values, the transfer function becomes:

$$\frac{X}{V} = \frac{17.14}{s^2 - 1.66}$$

6.2.2.3 Controller

A simple lead-lag controller with a transfer function in the form: $C(s) = \dfrac{s + \dfrac{1}{\tau_1}}{s + \dfrac{1}{\tau_2}}$ is

used, the controller circuit is shown in Figure 6.3. The values of the parameters τ_1 and τ_2 can be found as:

$\tau_1 = R_{55}C_6$,

$\tau_2 = R_{55}R_5C_6/(R_{55}+R_5)$

The feedback loop gain is approximately 24,

Note: we are using PWM signal instead of a DC signal, so it is hard to precisely determine the gain.

Using this information, the closed loop transfer function is approximately:

$$\frac{17.14s + 1.169e4}{s^3 + 681.8s^2 + 409.7s + 3.003e4}$$

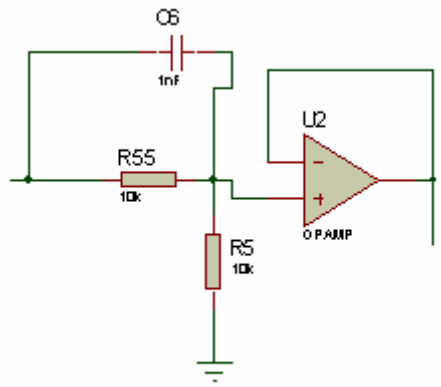


Figure 6.3 Lead-lag controller

6.2.2.4 Measurements

Using the oscilloscope (screenshot is shown in Figure 6.4) to obtain the response of the system, and building a model from the readings, the approximate experimental model is found to be:

$$\frac{440}{s^2 + 1.907s + 987.8}$$

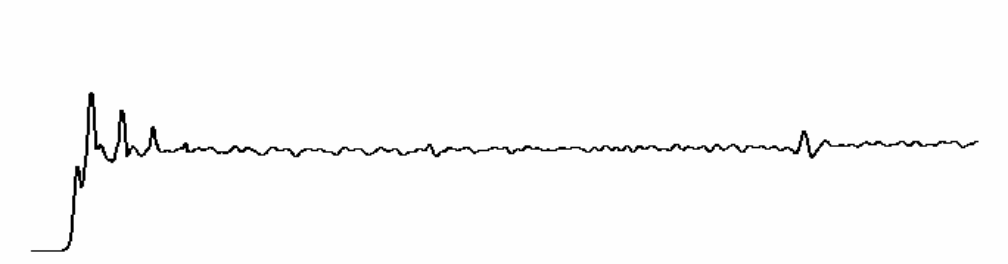


Figure 6.4 Oscilloscope output

Figure 6.5 shows the step response for the two models, the experimental (in green), and the derived (in blue):

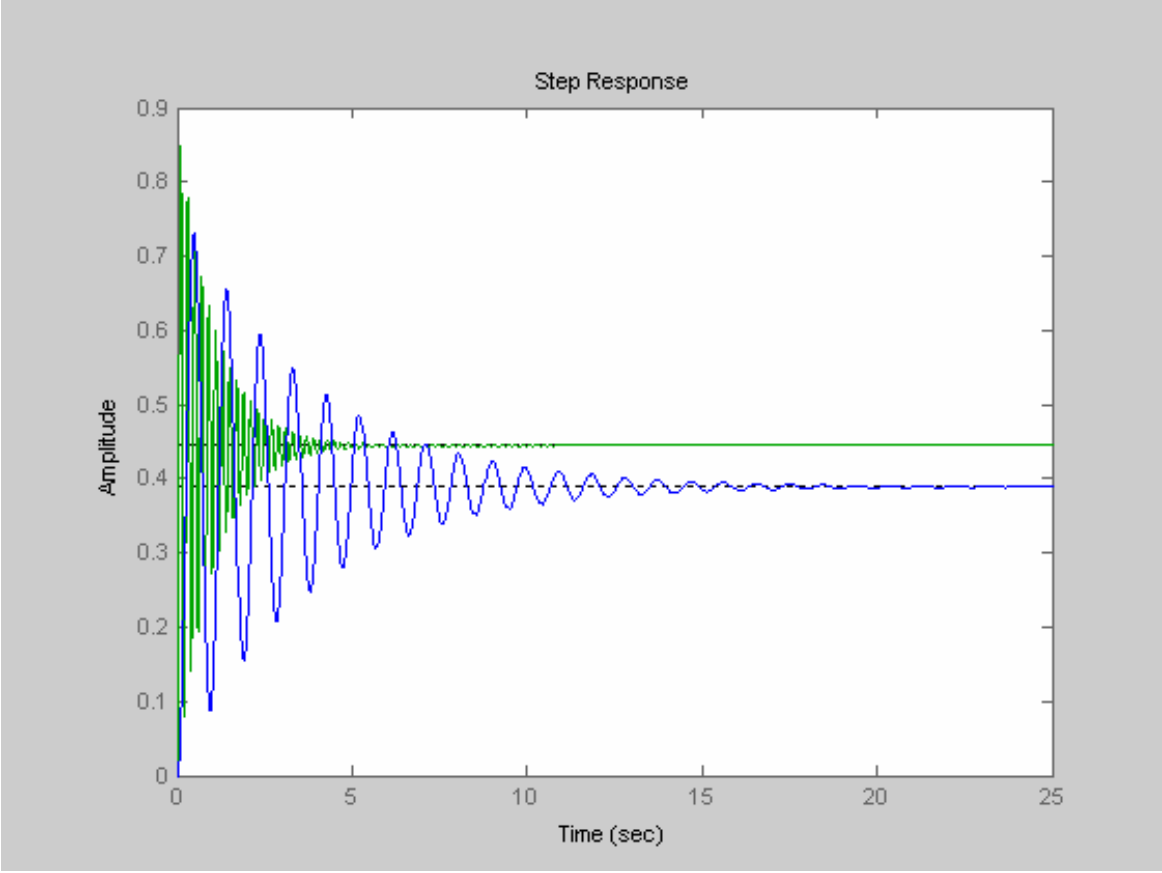


Figure 6.5 Matlab step responses for the system and its model

There is clear difference in the damping frequency, this might be because one or some of the following reasons:

1. The frequency response of the system depends on X_0.
2. When taking the readings from the oscilloscope, the ball vibrates in the horizontal direction, as well as the vertical direction, which will affect the readings (creates other frequency components).
3. The approximations in the derived model (approximations in k, and in the linearization process)
4. The approximations in the measured model (approximated to a second order model).

6.3 Microcontroller

Microcontrollers has economical price, they are chosen for our application since the application is meant for industry, it has to be designed with minimal costs. Microcontrollers can be programmed for a wide variety of applications.

6.3.1 PIC16F877 Microcontroller

Pic16F877 Microcontroller was a first choice in our research, it has a wide variety of applications, but it was found to have limitations as discussed here.

1. The Pic16F877 microcontroller worked on clock input of 20 MHz, the program that was designed to read the input signal from the sensor, apply the PID control law, and sends the signal back as a PWM (pulse-width modulated) signal, consisted of a 334 instruction cycle.

Two design cases were examined:

- The first one was using an LCD display to monitor the function of the system, this needed a total running time of 691.8uS*. If the ring was rotating at a speed of 2500 rad/sec, this means that from the time the sensor senses the location of the ring (the displacement between the ring front and the sensor) the ring would have moved with an angle of 1.73 rad, or 99 degrees. The error due to this enormous angle can't be avoided nor corrected by taking the phase shift into account; it is not only a phase shift problem, but certainly the position information after such a rotation will be changed.
- The second option was to remove the LCD display routine (since it consumed most of the time), the total time in this case was about 66.8us. This means an angle change of 9.6 degrees, although much less than the first design, it still is a big change. It cannot be corrected by simply adding an equal phase shift to the control law, because, this rotation is not only a change of the orientation angle of the rotor, but also a change in the position information, depends on the disturbances on the system.

The program had to have so many instruction cycles because the instruction list didn’t have a multiplication instruction that was needed for the PID control laws, so a separate routine code for the multiplication process had to be written.

2. The other problem was that the design needed two separate PWM signals to control the system in two dimension and PIC16F877 can only produce one PWM signal.

* The program had a total 334 instruction cycle, the execution time of each is 200 nano-seconds, totals 334*200 ns = 66.8 micro sec, and in the LCD routine five 125 micro sec delay are used causing additional delay equals 5*125 = 625 micro-seconds. The total time is 625+66.8=691.8 micro-seconds. Figure 6.6 shows the used Pic16F877 circuit.

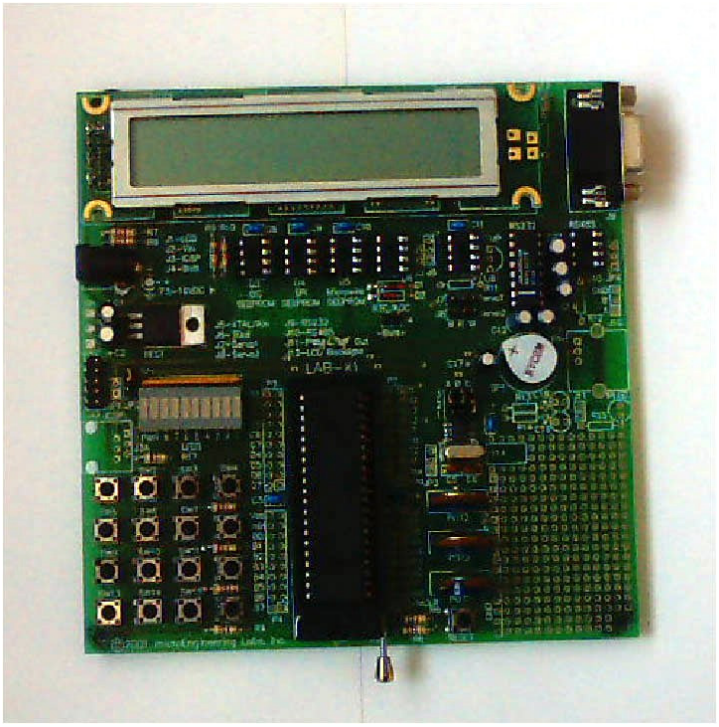


Figure 6.6 PIC16F877 evaluation board

PIC16F877 properties:

- Operating speed: DC - 20 MHz clock input
- Up to 8K x 14 words of FLASH Program Memory
- 10-bit Analog-to-Digital Module with 5 input channels
- Up to 368 x 8 bytes of Data Memory (RAM)
- Up to 256 x 8 bytes of EEPROM Data Memory

- All single cycle instructions except for program branches which are two cycle
- Two Capture, Compare, PWM modules
    - Capture is 16-bit, max. resolution is 12.5 ns
    - Compare is 16-bit, max. resolution is 200 ns
    - PWM max. resolution is 10-bit

6.3.2 PIC 18F4520

Due to the explained limitations of the PIC16F877 microcontroller, another microcontroller had to be selected. The PIC 18F4520 microcontroller offers higher computational performance, with economical price.

6.4 Position Sensing Circuits

A circuit was designed to read the signal from the displacement sensors, the circuit diagram is shown in Figure 6.7, and the actual circuit is shown in Figure 6.8.

6.4.1 Principal of Operation

The first two NAND gates (A and B), along with the connected capacitors and resistors work as a pulse generator. The generated square signal is filtered through the high-pass filter made from the inductor and the resistor, the high pass filtered signal is the input for the next stage (NAND gate), the output of this stage is a PWM (pulse width modulated) signal, which DC (duty cycle) is proportional to the value of the inductance. The next stage is a low pass filter stage that converts the PWM signal to a voltage value proportional to the value of the inductance. The IC (U2) is a regulator to assure a stable

value of voltage supply, and to set a reference to the circuit, the potentiometers RV1, and RV2, are for calibration purpose; they are adjusted with a reference inductance connected (inductance with a known value) to calibrate the output voltage value.

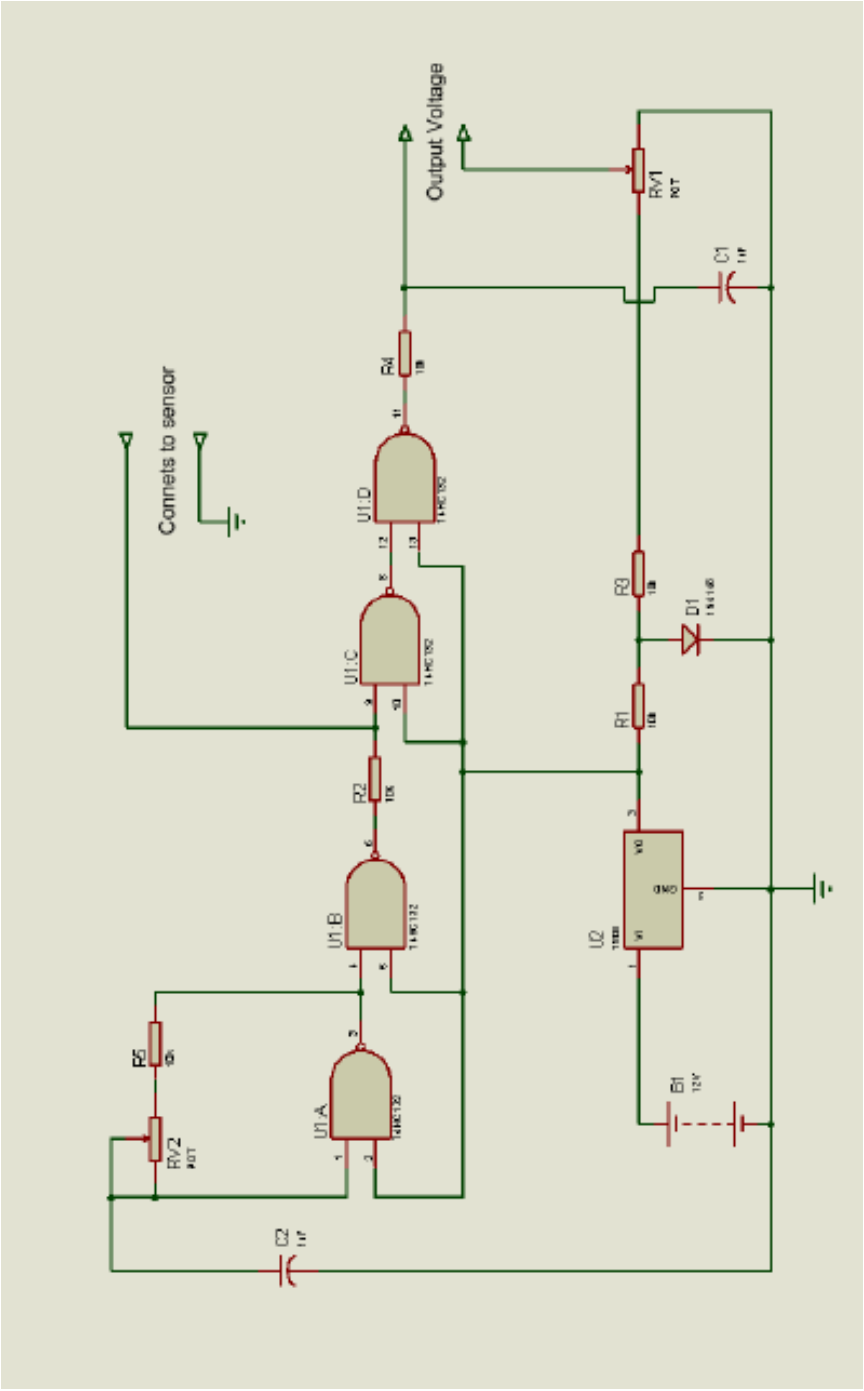


Figure 6.7 Circuit diagram for position sensing

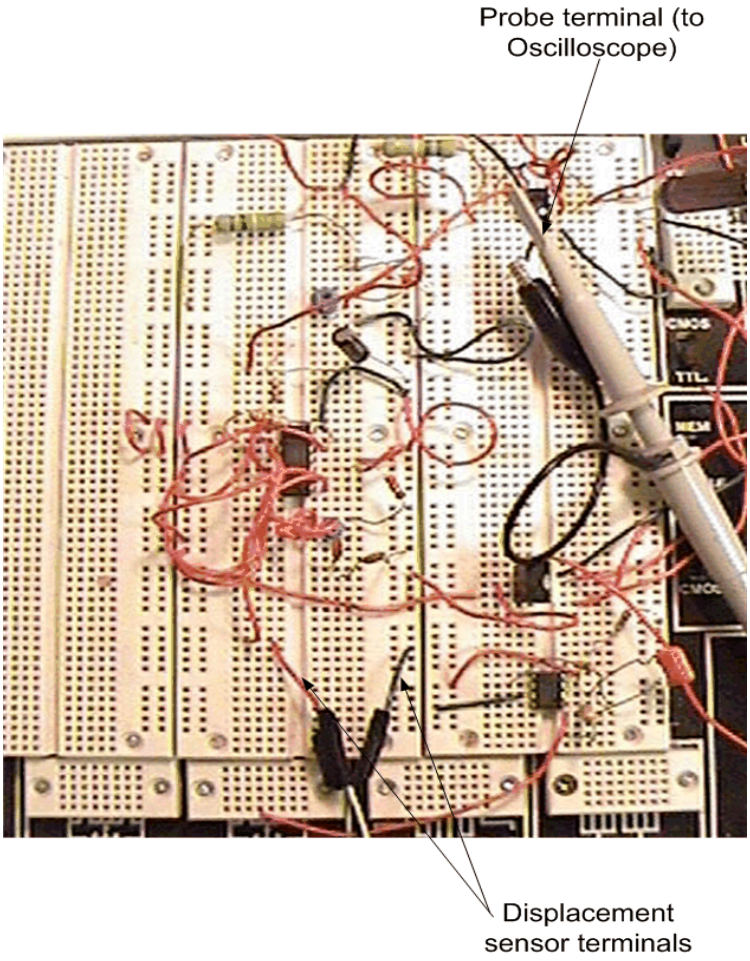


Figure 6.8 Sensor reading circuit

4.2 Displacement Reading Testing

To test if the displacement reading circuit is suitable for our application, the set-up shown in Figure 6.9 was built. It consists of steel beam clamped from one side and free from the other side. The free side faces a displacement sensor that is connected to the tested circuit; the output of the circuit is read using an oscilloscope.

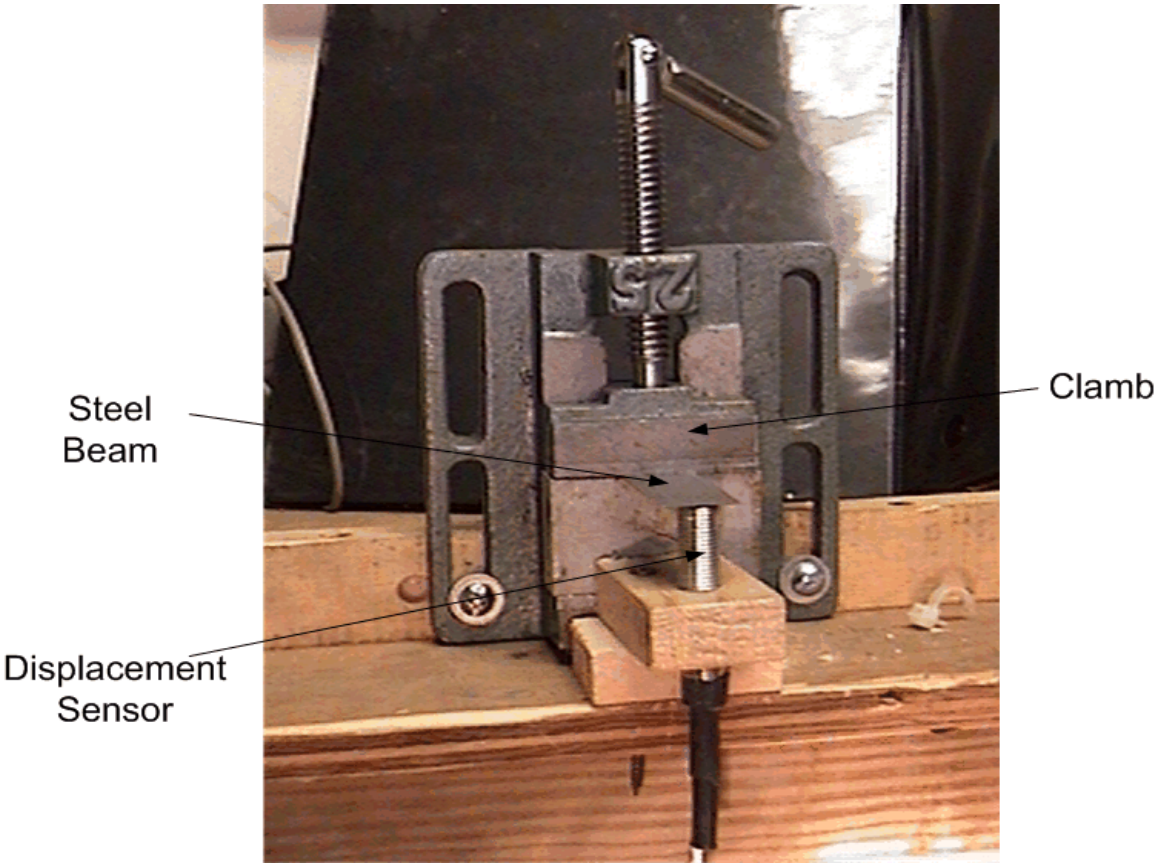


Figure 6.9 Setup to test the sensor reading circuit

6.4.2.1 Operation

The frequency of the clamped beam can be calculated using the following equation:

$$f = 3.52\sqrt{\frac{EI}{mL^4}} rad/\sec$$

Where:

E = steel Young modulus = 210e9 N/m^2

I = section moment of inertia = $\frac{bh^3}{12}$

b = beam width

h = beam thickness

m = mass/length

L = beam length

Then the value of the frequency, resulting from this equation, is compared to the value of the vibrating beam measured using the oscilloscope.

6.4.2.2 Results

The used beam has the following parameters:

L = 52 mm

b = 23 mm

h = 1/64 Inch = 0.3969 mm

Density = Carbon steel density = 7.8 g/cc

Using the Matlab code shown in appendix 5, the frequency was found to be:

Frequency (calculated) = 134.2498

And Figure 6.10 shows the output of the oscilloscope, with measured frequency of:

Frequency (measured) = 134 Hz

6.5 Prototype

A prototype has been built and is being tested, (shown Figure 6.11).

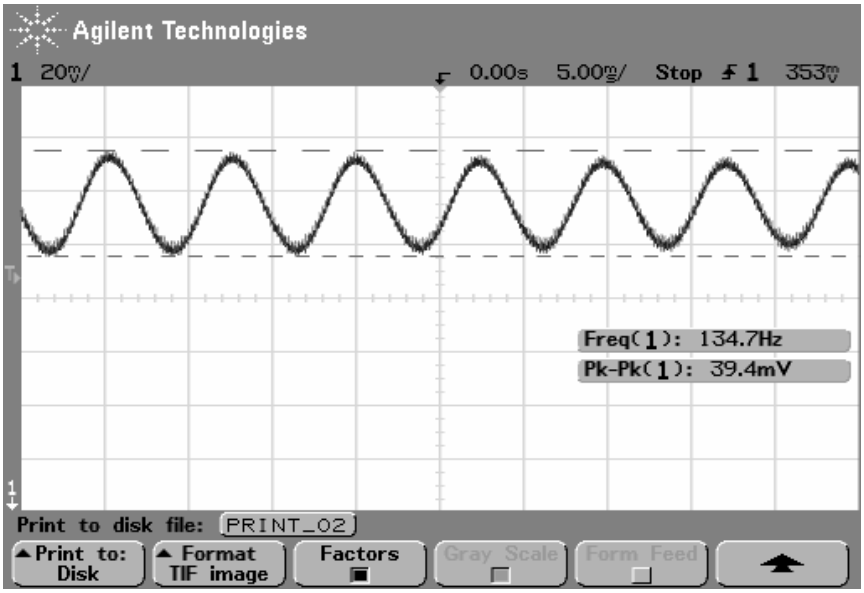


Figure 6.10 Oscilloscope reading

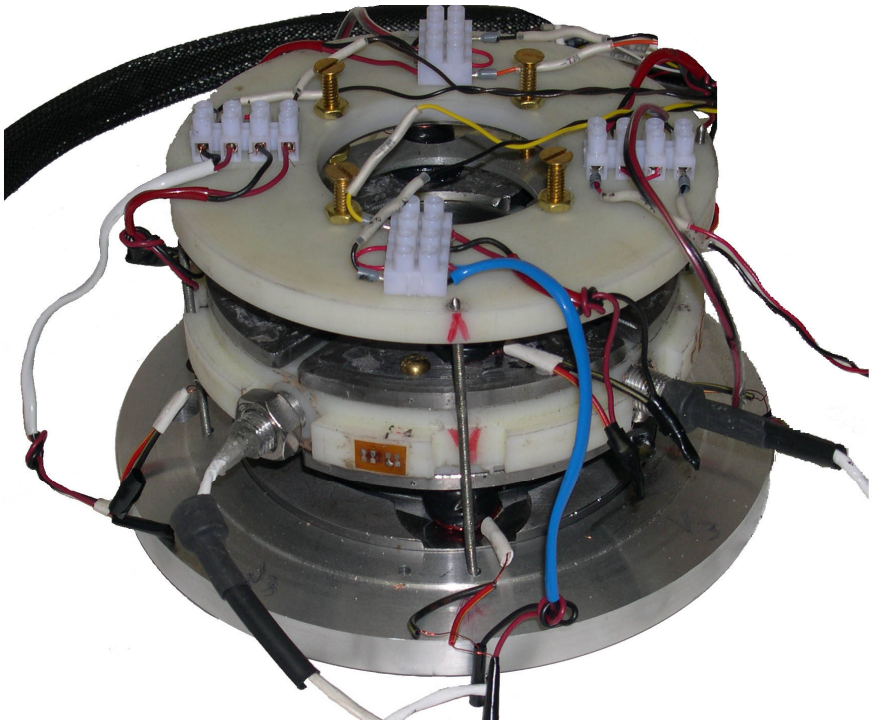


Figure 6.11 Prototype

6.6 Conclusions

Experimental work included building a magnetic suspended ball system, implementing the designed control firmware on microcontrollers and testing their performance, and building the position sensing circuit and testing its functionality. A prototype was built.

7. RESULTS AND DISCUSSION

7.1 Simulink Simulation

The simulation showed the ballooning behavior of the yarn, the response of the controller, the spinning speed behavior, and the response of the system to changes in its parameters. As explained below.

7.1.1 Yarn Ballooning

Figures 7.1 a,b,c shows the simulated yarn from different projection angles (top, side, and three dimensional view) at the beginning of the simulation, (the machine starting operation), the figure shows that the yarn is floating without a defined shape.

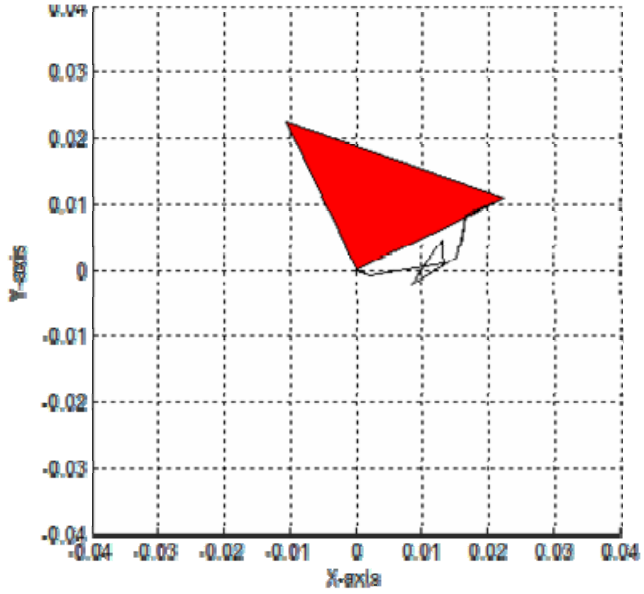


Figure 7.1.a The simulated yarn at the beginning of the simulation, top view

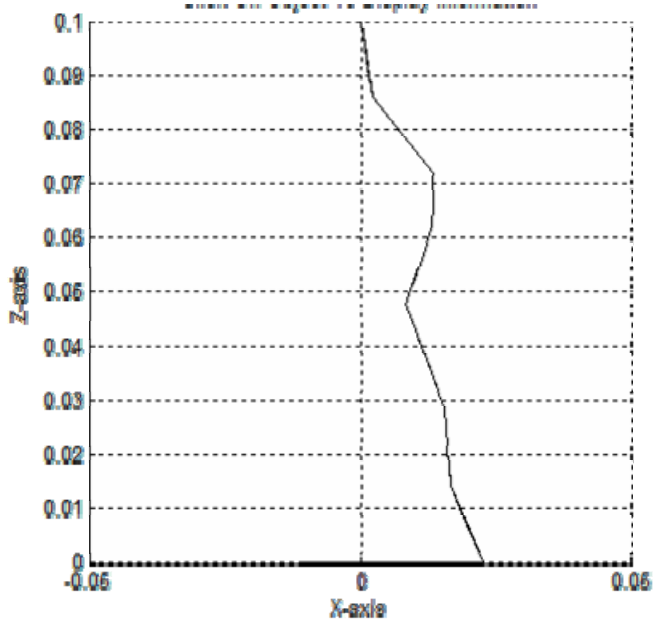


Figure 7.1.b The simulated yarn at the beginning of the simulation, side view

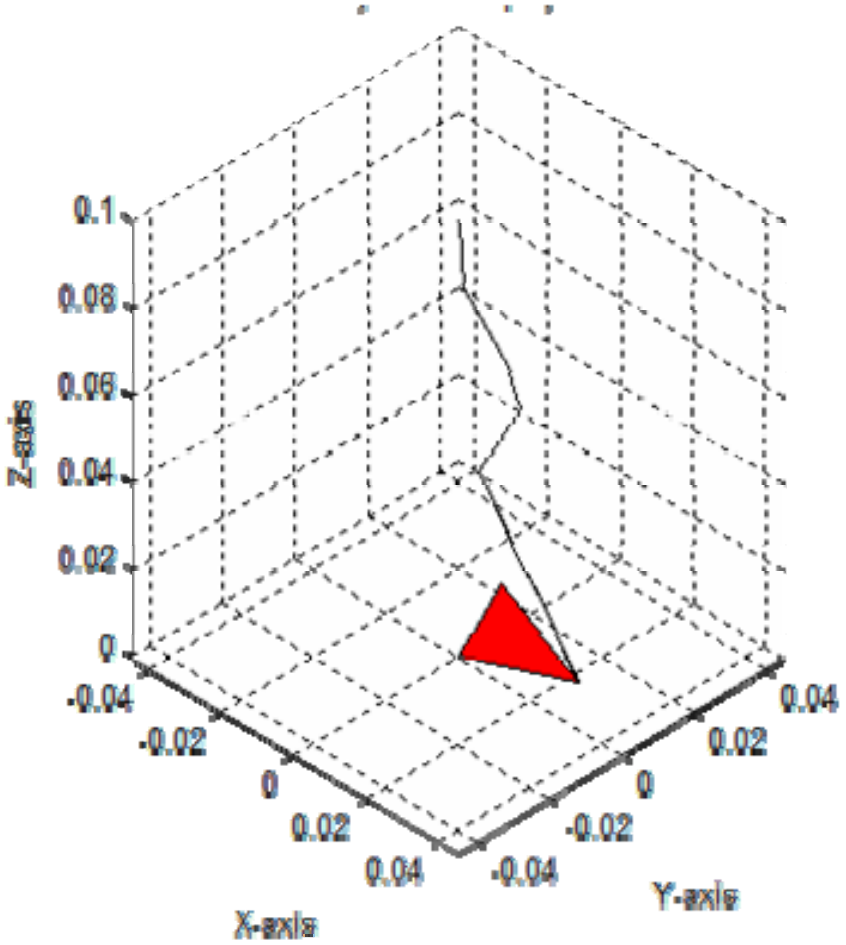


Figure 7.1.c The simulated yarn at the beginning of the simulation, 3D view

Figure 7.2 shows the yarn after the formation of the balloon from the same three angels (after 22 seconds of machine running time), now the machine has reached full speed, and the yarn balloon has properly formed. The shapes of these yarn balloons agree with the actual balloon shapes of the traditional ring-spinning process, shown in earlier works.

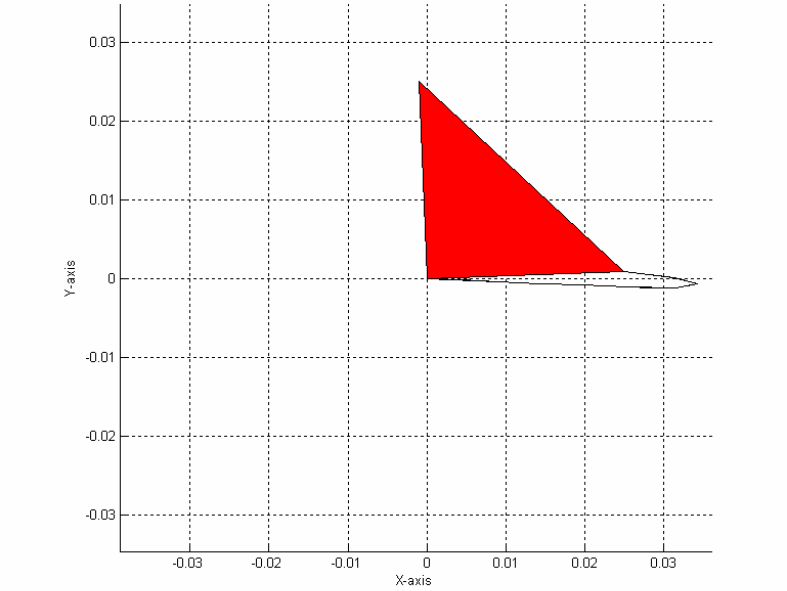


Figure 7.2.a The simulated yarn after ballooning, top view

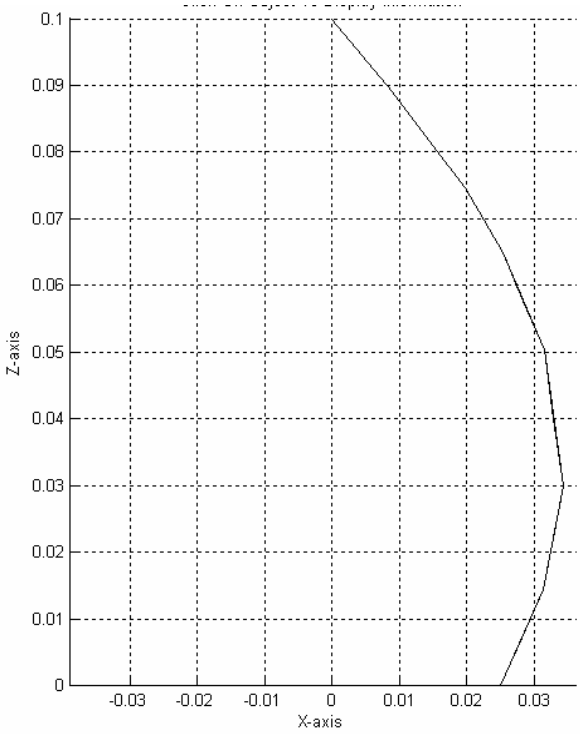


Figure 7.2.b The simulated yarn after ballooning, side view

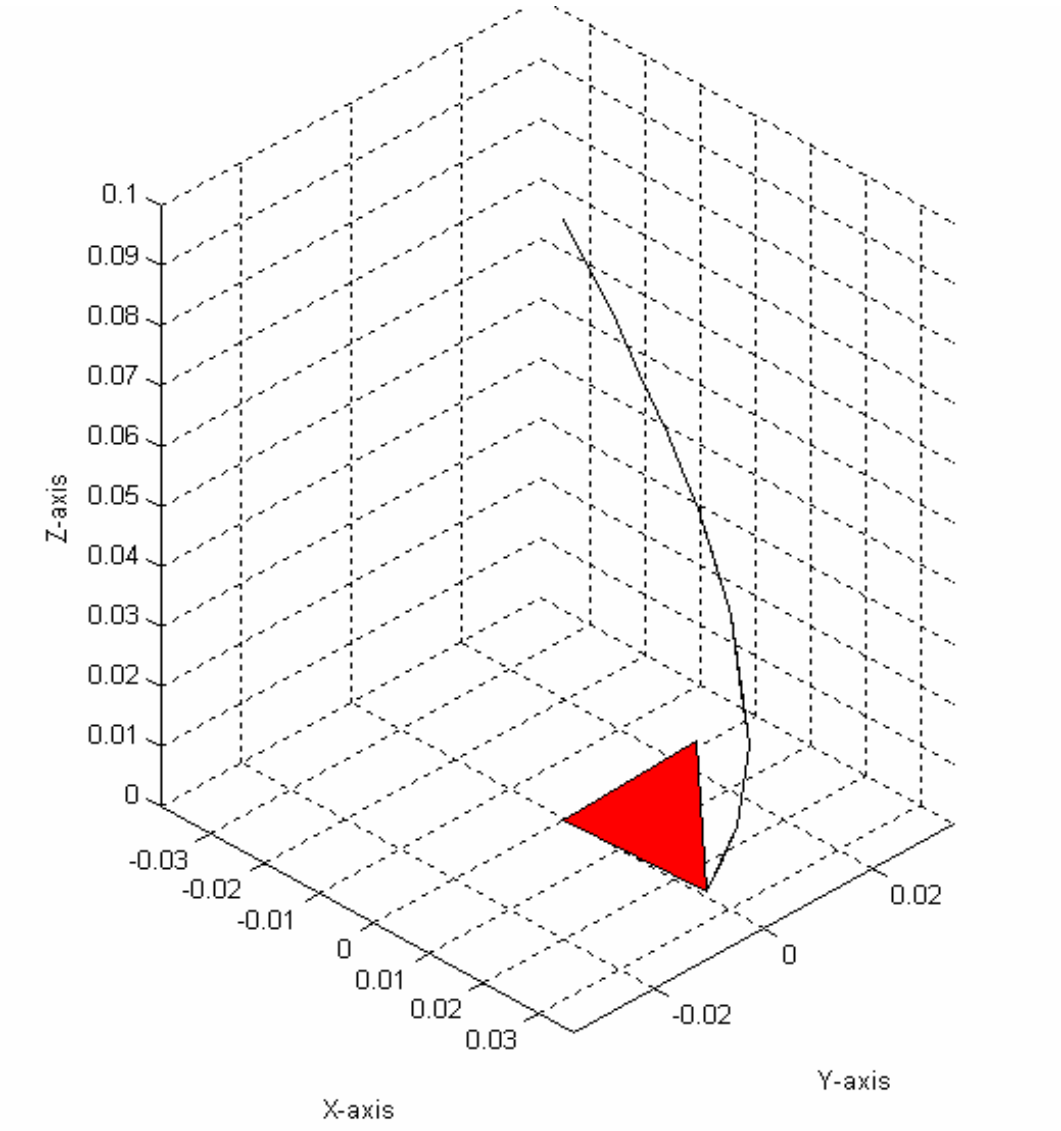


Figure 7.2.c The simulated yarn after ballooning, 3D view

Figure 7.3 shows the ring speed in rpm. The speed reaches its maximum (steady-state operation) of 25,000 rpm in less than 25 seconds from start. The long time (25 seconds) is because of the large mass of the rotor (32 g), by increasing or decreasing the mass of the rotor, longer or shorter start-up times can be achieved. The steady state speed is reached when the air-drag force balances with the driving force of the ring. Therefore,

the steady-state speed is function of the surface area of both the yarn and the rotor; by changing the area or the surface geometry of the rotor we can change the spinning speed. The first part of the curve is linear, due to the fact that with smaller values of speed, the effect of air-drag is negligible. When the speed increases, the effect of air-drag starts taking place, and the curve settles exponentially to its steady-state value.

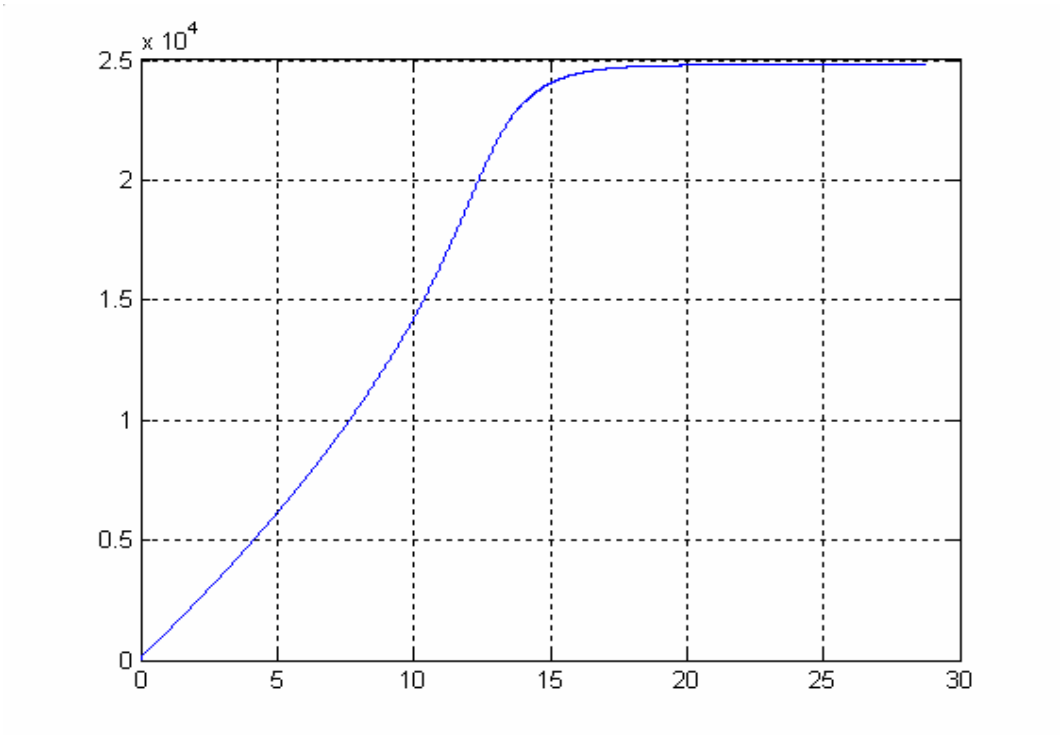


Figure 7.3 ring speed in rpm

The tension of the yarn –in cN- is shown in Figure 7.4, as a function of time. The tension of the yarn reaches its maximum of about 9.8 cN with the speed reaching its maximum, the tension as a function of the speed is shown in figure 7.5. The speed-tension analysis shows that with the magnetic ring spinning system, a high speed of spinning can be achieved along without very high yarn tension. In the simulation the ring

is driven by external force, not by the yarn; the tension in these graphs is only the component of the tension from the air-drag.

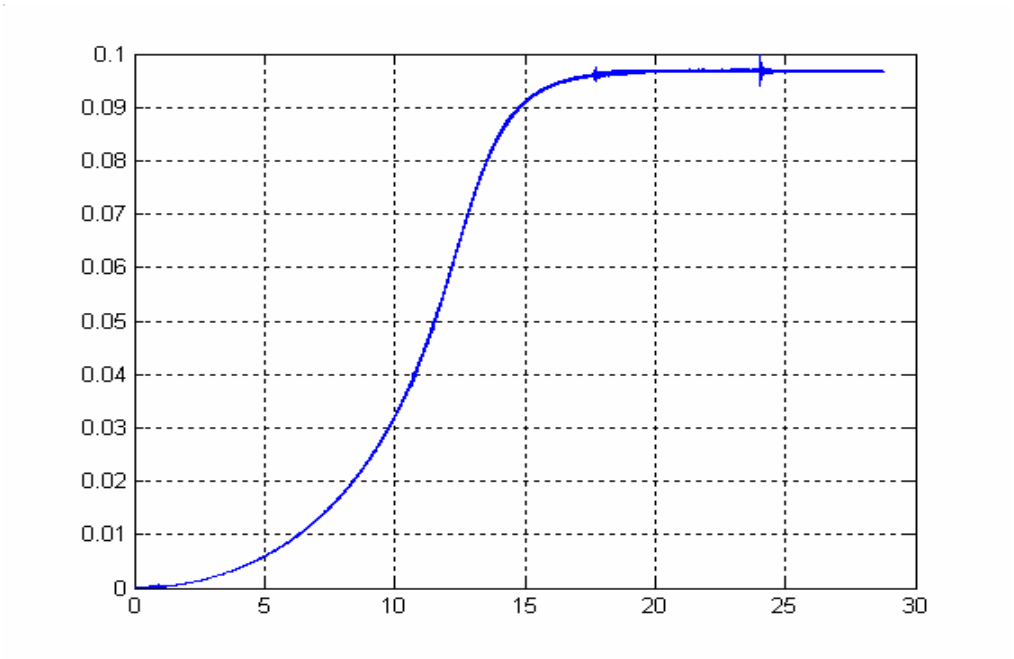


Figure 7.4 Tension of the yarn in cN

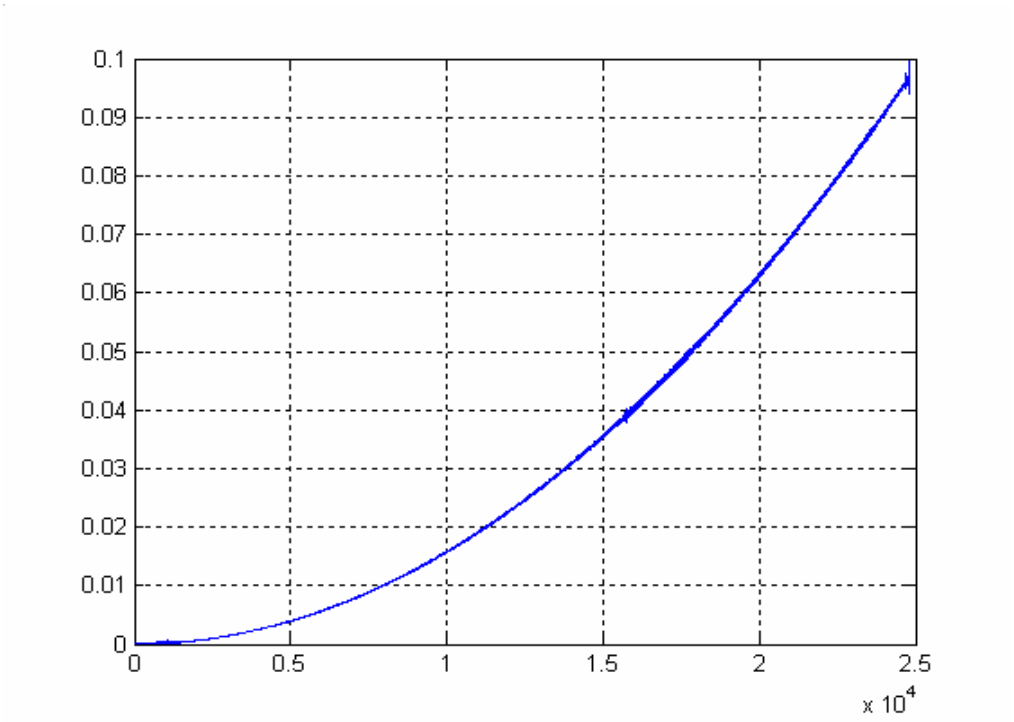


Figure 7.5 Tension as a function of the speed

7.1.2 Discrete PID Controller

Figures 7.6 a and b shows a graph of the change in the ring position with time, this graph is used to analyze the functionality of the controller, it can be seen that the controller was successful in settling the position of he ring around the center of rotation, the ring started with an off-center initial condition of -1mm, and settled in about 4 ms to .1 mm (90%), and to .01 (99%) in 8 ms. and settled in about 4 seconds to 2um fluctuation.

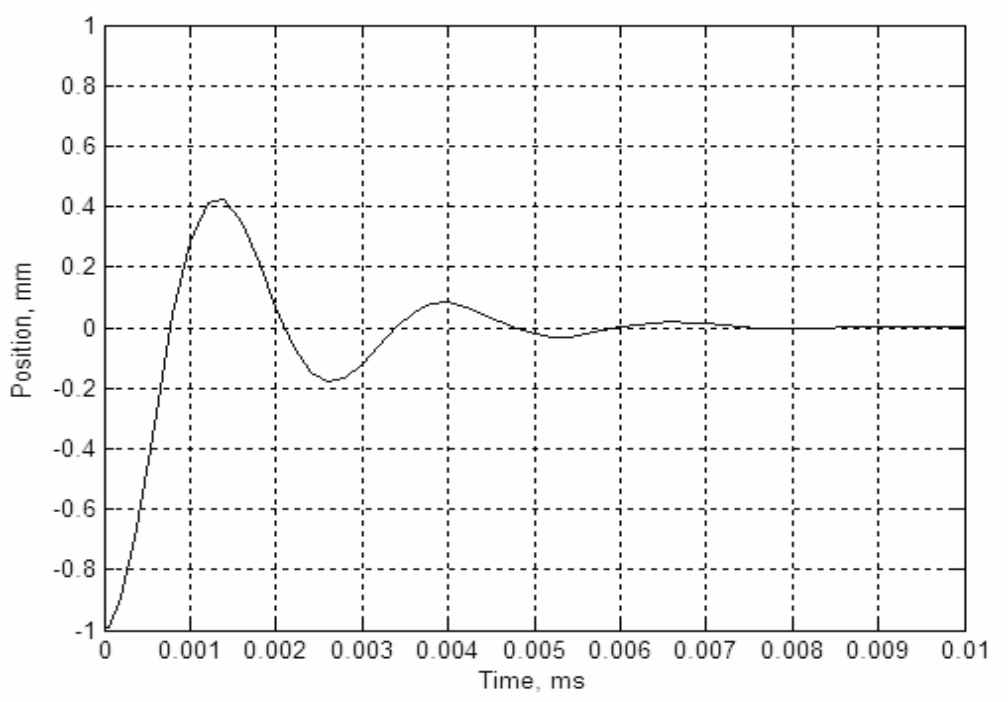


Figure 7.6.a Change in the ring position with time

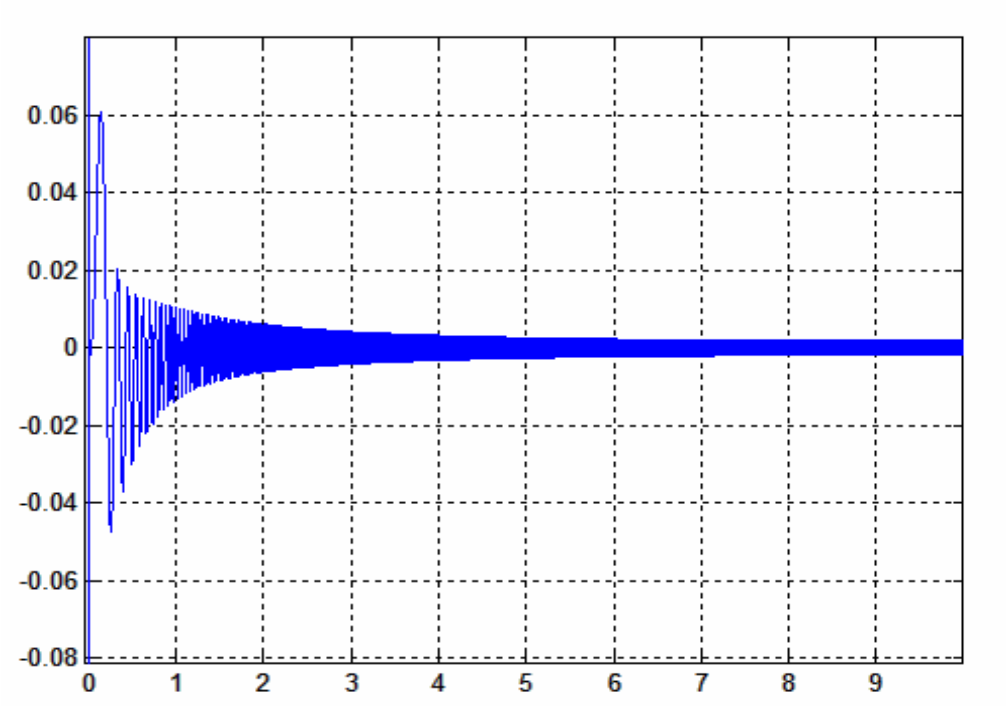


Figure 7.6.b Change in the ring position with time, long period

7.1.3 Fuzzy Controller

The two designed fuzzy controller schemes were tested with the model. Figure 7.7 shows a Simulink window that displays the used fuzzy rules during operation. To study the effect of the fuzzy controllers with the system, the noise signal shown in figure 7.8 was added to the position of the rotor through an actuator. The response of each fuzzy controller with the existence of this noise signal is shown below.

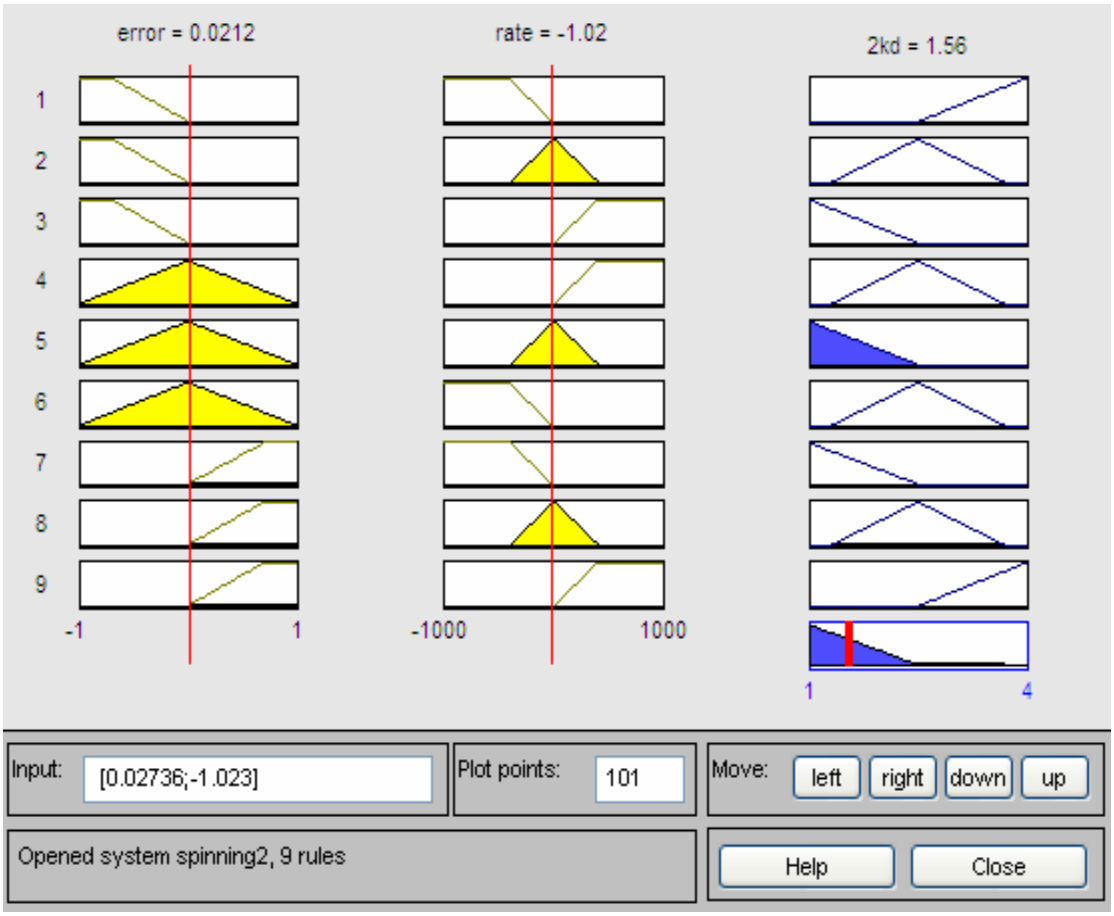


Figure 7.7 Simulink window displays the used fuzzy rules during operation

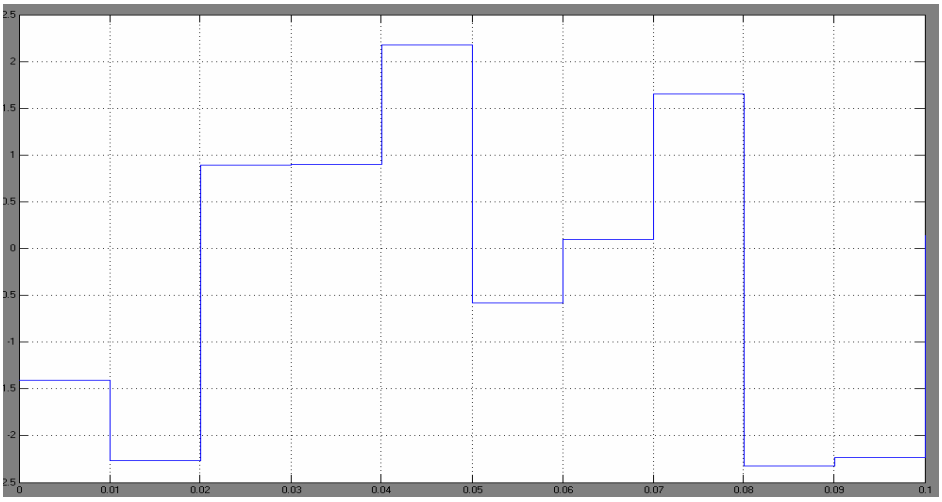


Figure 7.8 Inserted noise signal

7.1.3.1 Fuzzy Controller, Option 1

Figure 7.9 shows the response of the system with PID parameters being adapted using fuzzy controller (top), and only PID controller. From the figure, we find that although the settling time (from start with non-zero initial condition) was more with the fuzzy controller on than without the fuzzy controller, the system responded better to the disturbances with the fuzzy controller.

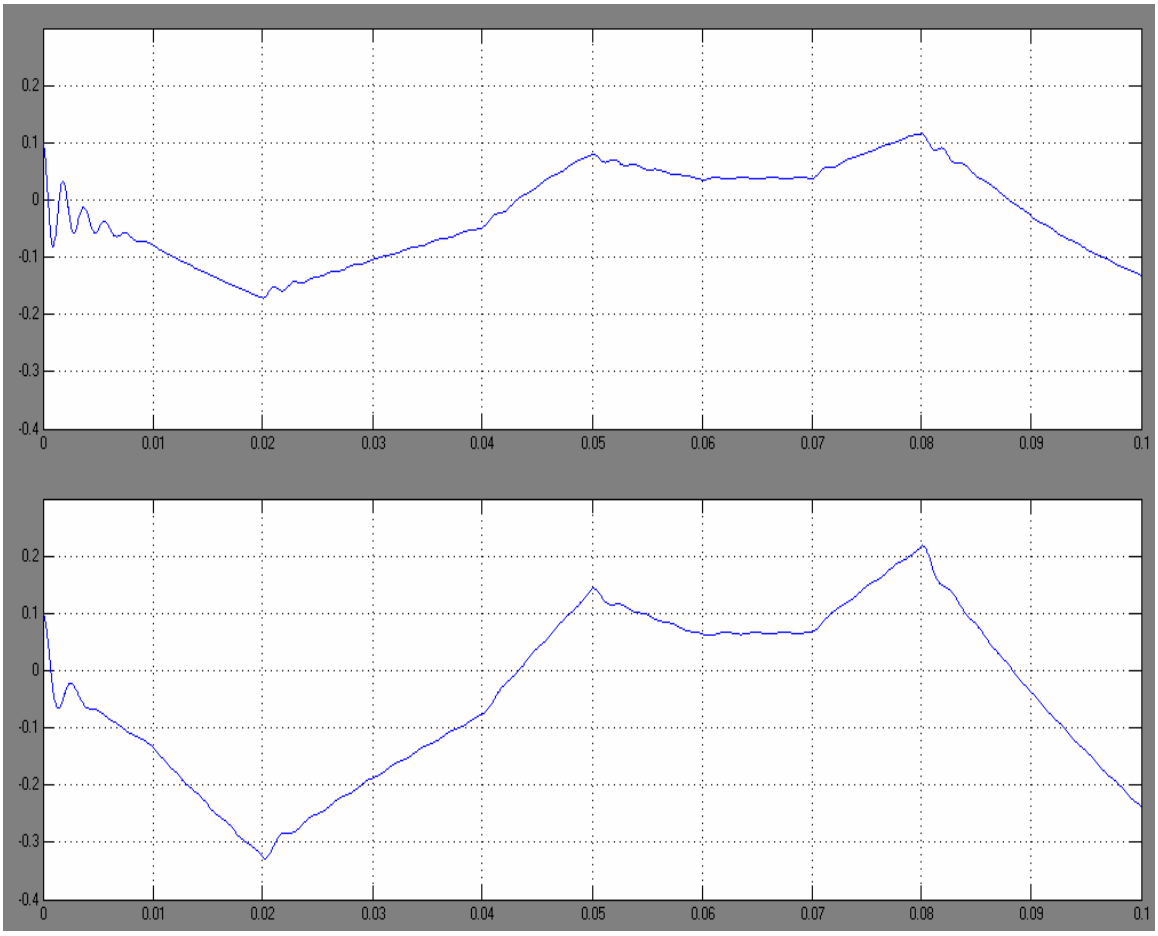


Figure 7.9 Fuzzy Controller for parameter adaptation response (top) compared with PID response (bottom)

7.1.3.2 Fuzzy Controller, Option 2

Using this option resulted in worse response than using the PID controller alone, this option needs further investigations.

Figure 7.10 shows the response of the system with this fuzzy controller compared to the response of the system with PID controller.

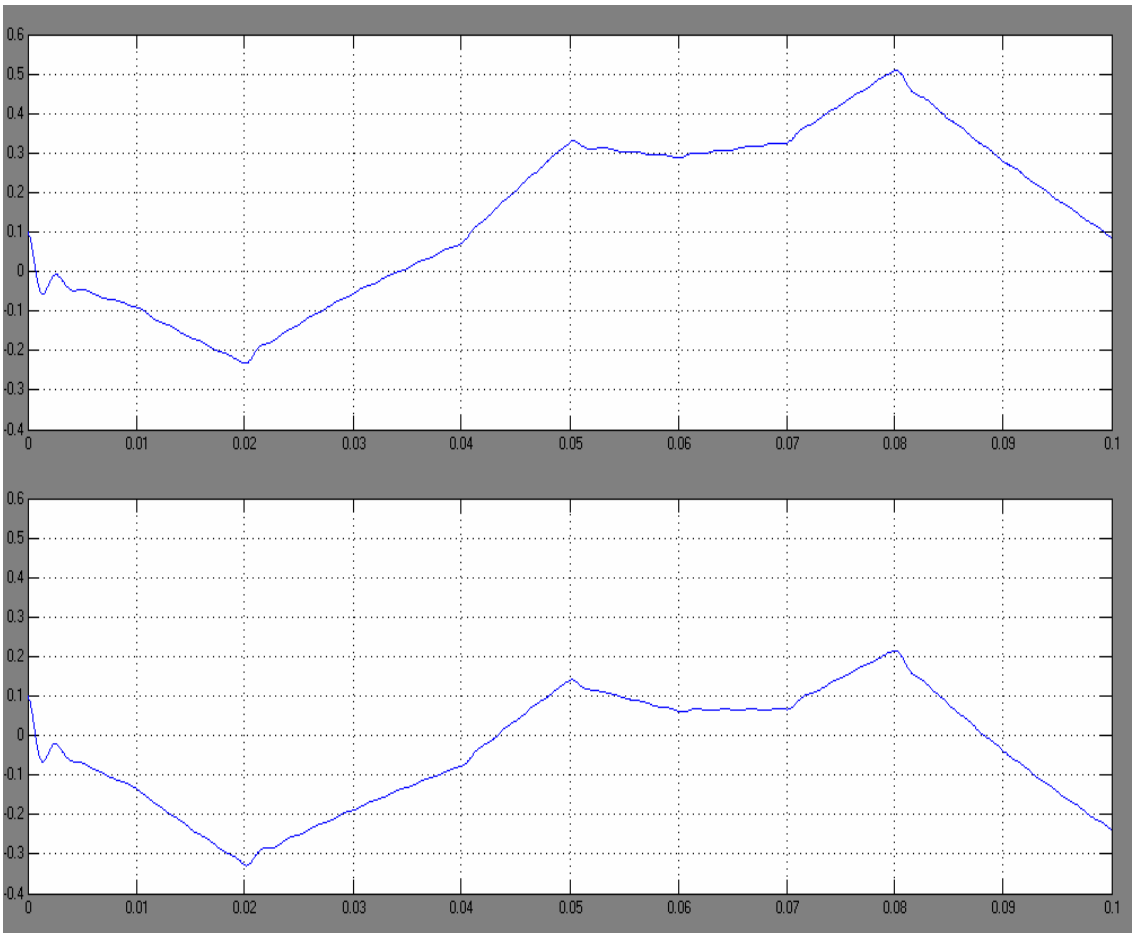


Figure 7.10 Fuzzy Controller option 2 (top) compared with PID response (bottom)

From the graphs it was clear that ‘option 1’ works better than ‘option 2’ for implementing fuzzy control to our application.

7.1.3.3 Fuzzy Controller More Results

The results shown in this section are from examining 'option 1' with other noise signals, Figure 7.11 shows the response of the PID with the fuzzy controller implemented (blue) compared with the response of the PID controller alone (green), when limited bandwidth white noise was applied to the position of the ring.

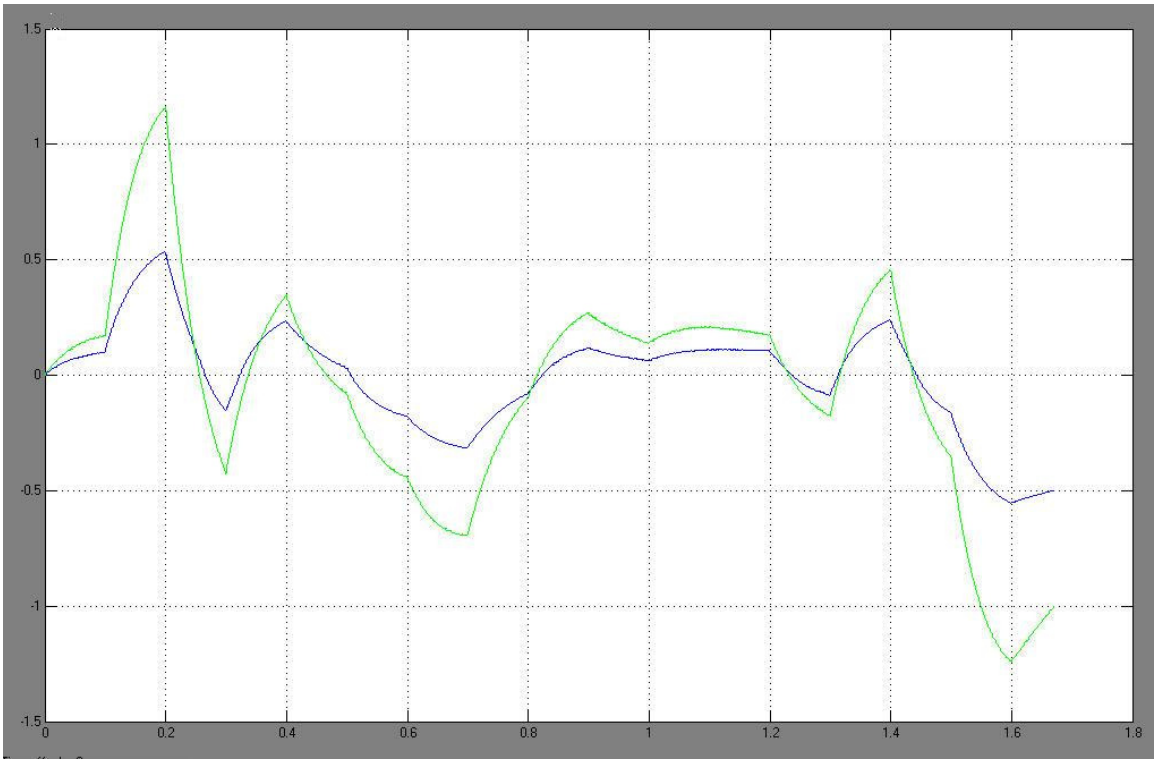


Figure 7.11 Response Fuzzy/PID controller vs PID alone, with White Noise Applied

From the figure, we can see that the response of the system was much better when implementing the fuzzy controller to supervise the operation of the PID controller.

8. CONCLUSIONS

A new ring-spinning system named “magnetic-spinning” is designed, the system is based on the concept of magnetic-suspension, where a light weight rotor is designed to be magnetically suspended inside a fixed stator, using electromagnets that modulates the field of radially mounted permanent magnets to keep the rotor suspended, displacement sensors to sense the position of the rotor, and control circuit to adjust the electromagnets’ input current based on the position. In the new designed system, the rotor replaces the ring and traveler in the traditional spinning system.

An approximate analytical model and transfer function of the system were derived, based on the design.

The design was then validated using a complete finite element model.

Control of the developed system is done in single input-single output manner. An Interface circuit was built to read signals from the four radially mounted displacement sensors, and convert them to position information.

PID (Proportional-Integral-Derivative) controller was designed to control the system.

Fuzzy control was used as supervisory controller over the PID controller in two different control schemes. In the first configuration, a fuzzy controller supervises the operation of the PID controller. The role of the Fuzzy controller was to tune the parameters of the PID controller (the proportional, the derivative, and the integral gains

or K_p, K_d, and K_i consecutively), and optimize them according to the operating conditions of the system.

In the second configuration, Fuzzy and PID controllers work in parallel as two independent processes each of them takes the set-point and the position information as two inputs, the PID controller works as designed earlier. The Fuzzy controller is designed so that it has a zero output in normal operating conditions, and in abnormal operating conditions (disturbances, overshoots, long settling time, etc.) it has a nonzero output. The output signals from the two controllers are added together and sent as a control signal to the Magnetic-Ring Spinning machine.

To implement the designed controller, PIC18F452 microcontroller was suggested, an assembly firmware for the PID control was designed.

The designed fuzzy controller was implemented as an assembly program using FUZZYTECH®; which is developing software for fuzzy control applications, the generated fuzzy assembly code was modified and implemented into the firmware developed earlier for the PID controller.

A complete numerical model of the magnetic ring spinning system was built and simulated using Simulink. Different blocks in this model represent every component of the designed magnetic spinning system, this includes the mechanical parts of the system, the electrical circuitry, the sensors, the actuators, and also the effect of air-drag on both the yarn and on the rotor.

The results of the simulated balloons agreed with the yarn balloon behavior studied in earlier research, this includes the shape of the balloon, and the yarn tension, hence, this numerical simulation done can be used to study the yarn ballooning in other

systems, and it has the advantage over the analytical models used in earlier research, this advantage is simply that no mathematical approximations, or linearizations are done in this simulation, which provides more accuracy in modeling.

The designed PID and fuzzy controllers were also implemented in this numerical model, and showed to work properly.

ISIS software was used to simulate the microcontroller with the designed firmware and the control circuit.

Two circuits were designed for the proper operation of the Analog-to-Digital conversion in the ISIS model; the first method involved using logic circuits, and had some disadvantages. The second method used BJT transistors for switching and performed better than the first method.

The Simulink simulation showed validity of the model, the simulation showed proper ballooning of the yarn, and was also used to calculate the expected tension in the yarn. The designed controller schemes were able to control and stabilize the ring. The effect of each designed control scheme on the operation was studied and compared with the other schemes. The Simulink simulation also showed the relationship between the rotor speed and the yarn tension, the behavior of the yarn tension as a function of time, and the yarn ballooning.

To implement the controller on a microprocessor, an assembly code was written. A variety of microcontrollers were studied, and some of them were found not suitable for our application.

PIC18F452 microprocessor/microcontroller was chosen for the application. An assembly program was written for a PIC18F452 microprocessor, the firmware was simulated with the microcontroller using ISIS developing software.

Experimental work included building a magnetic suspended ball system, implementing the designed control firmware on microcontrollers and testing their performance, and building the position sensing circuit and testing its functionality. A prototype was built. The simulation showed that the designed prototype needs to be improved, lighter weight ring is needed.

BIBLIOGRAPHY

[1] Charara, A. and B. Caron, Lamii, "Magnetic bearing: Comparison between Linear and Nonlinear Functioning", Proceedings of the Third International Symposium on Magnetic Bearings, July 29-31, 1992, Radisson Hotel at mark Center, Alexandria, Virginia.

[2] De Queiroz, Marcio S. and Dawson, Darren M., "Nonlinear Control of Active Magnetic Bearings: A Backstepping Approach" IEEE TRANSACTIONS ON CONTROL SYSTEMS TECHNOLOGY, VOL. 4, NO. 5, SEPTEMBER 1996, p545

[3] Dhar, D. et al, "Optimum Design of Decentralized Magnetic Bearings for Rotor Systems", Proceedings of the Third International Symposium on Magnetic Bearings, July 29-31,1992, Radisson Hotel at mark Center, Alexandria, Virginia.

[4] El Mogahzy, Yehia E., "Cotton Fiber to Yarn Manufacturing Technology", Published by Cotton Incorporated, 2002

[5] Herzog, R. "A Comparison between Passively and Actively Controlled Magnetic Bearings", Proceedings of the Third International Symposium on Magnetic Bearings, July 29-31, 1992, Radisson Hotel at mark Center, Alexandria, Virginia.

[6] Hung, John Y., "Magnetic Bearing Control Using Fuzzy Logic", IEEE TRANSACTIONS ON INDUSTRY APPLICATIONS, VOL, 31, NO. 6, November/December 1995

[7] Li, Lichuan, Shinshi, Tadahiko, and Shimokohbe, Akira "State Feedback Control for Active Magnetic Bearings based on Current Change Rate Alone", IEEE TRANSACTIONS ON MAGNETICS, VOL. 40, NO. 6, NOVEMBER 2004

[8] Morse, Nancy Thibeault, and Smith, Roy S. "Magnetic Bearing Measurement Configurations and Associated Robustness and Performance Limitations", Transactions of the ASME, Journal of Dynamic Systems, Measurement, and Control, p590, Vol. 124, DECEMBER 2002

[9] Oxtoby, Eric. "Spun yarn technology", London; Boston, Butterworths, 1987.

[10] Studer, Philip A. "Magnetic Bearings for Spacecraft", IEEE Transactions ON magnetics, p520, SEPTEMBER 1971

[11] The MathWorks, Inc. “SimMechanics for Use with Simulink® User’s Guide, Version 2”, 2004

[12] The MathWorks, Inc., “Fuzzy Logic Toolbox User’s Guide For Use with MATLAB®, Version 2”, 2002

[13] www.textileworld.com

[14] Valenti, Chris “Implementing a PID Controller Using a PIC18 MCU”, Microchip Application Note AN937, 2004;
http://ww1.microchip.com/downloads/en/AppNotes/00937a.pdf

APPENDIX 1. NUMERICAL CALCULATIONS

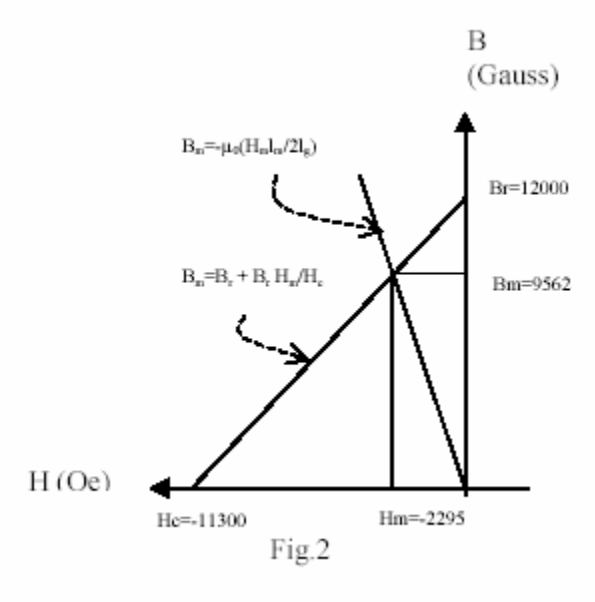


Fig.2

The field intensity in the gap1 due to the permanent magnet H_{gp}:

$$2H_{gp}g_1 = -H_m l_m$$

$$H_{gp} - \frac{H_m l_m}{2g_1} \qquad \text{.....(1)}$$

Where g_1 is the gap thickness and H_m & B_m is the operating point of the permanent magnet found by the intersection between the load line and the demagnetization curve of the magnet Fig.2

The load line is defined by the following equation:

$$B_m = -\mu_0 \frac{H_m l_m}{2l_g} \cdots (2)$$

The approximate demagnetization curve has the form:

$$B_m = B_r + \frac{B_r}{H_c} H_m$$

For Neo 35 magnets:

The residual flux density *Br* = 12000 Gauss

The coercive Force *Hc* = 11300 Oersteds

Solving the two equations, for lm=25 mm , lg= 2 mm, Ag/Am=2/6

Gives:

Bm = 9562 Gauss = 0.9562 Wb.m-2

Hm = -2295 Oersteds = -182636.1 A.m-1

The field in the gap due to coils1 and coils2:

$2H_{gc}g_1 = 2NI$ ... (3)

Where number of turns in each coil is N, and the field due to the current is opposing the field due to the permanent magnet.

$H_{gc} = NI/g_1$

$H_{g.tot.} = NI/g_1 - H_m l_m/2g_1$ ……(4)

$B_g = \mu_0 H_g$,

$F_1 = (1/2\mu_0)^2 B_g^2 S$, where S is the gap area

$F_1 = (\mu_0/g_1^2)S(NI - H_m l_m/2)^2$ ……..(5)

Similarly, the Force in side 2:

$F2 = (\mu_0/g_2^2)S(NI + H_m l_m/2)^2$

$F_{tot.} = \mu_0 S[((NI- H_m l_m/2)/ g_1)^2- ((NI+ H_m l_m/2)/ g_2)^2]$ …….(6)

g_1=g-x, and g_2= g+x

Where g is nominal gap thickness (the gap between the floating ring centered and the stator in our present system this gap is 1 mm) for small displacement .x and small current .i, and using polynomial expansion and approximation

$\Delta Ft=4M(-N\Delta i/b + \Delta x/g)$

Where: $M = \mu_0 S(H_m l_{m/}2)^2/g_2$, and $b= H_m l_m/2$

Let us define two stiffness parameters:

Position stiffness $K_x = \frac{\Delta F_{tot}}{\Delta x}$ $K_x=\Delta Ft/\Delta x$,(at $\Delta i = 0$) = 4M/g

And current stiffness: $K_i = \frac{\Delta F_{tot}}{\Delta i}$,(at $\Delta x = 0$) = -4M/b

Now, the dynamics of the floating ring can be expressed using the following:

$$\Delta F_t = m\ddot{x} + K_i \Delta_i + K_x \Delta_x$$

Using Laplace transform:

$$ms^2 X + K_x X + K_i I = F$$

From which the transfer function between the force and the displacement at zero current is:

$$\frac{X(s)}{F(s)} = \frac{1}{ms^2 + K_x}$$

APPENDIX 2. TEST THE LINEARIZED MODEL FOR CONTROLLABILITY

The state-space model of the magnetic ring system is:

$$\dot{x}_1 = \dot{x} = x_2$$

$$\dot{x}_2 = \ddot{x} = \frac{k}{m}\left[\frac{(I - x_3)^2}{(G - x_1)^2} - \frac{(I + x_3)^2}{(G + x_1)^2}\right]$$

$$\dot{x}_3 = \frac{d}{dt}x_3 = \frac{1}{L}v - \frac{R}{L}x_3$$

Linearizing around the equilibrium point ($\dot{X} = 0$)

$$A = \left[\frac{\partial F}{\partial X}\right]\Bigg|_{X = X_{eq}}$$

$$B = \left[\frac{\partial F}{\partial U}\right]\Bigg|_{X = X_{eq}}$$

$\therefore$

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 4\frac{I^2}{G^3} & 0 & -4\frac{I}{G^2} \\ 0 & 0 & -\frac{R}{L} \end{bmatrix}$$

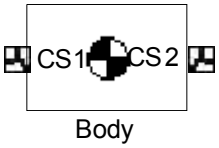
$$B = \begin{bmatrix} 0 \\ 0 \\ \frac{1}{L} \end{bmatrix}$$

The Controllability matrix has rank of 3, hence, the linearized model is controllable.

APPENDIX 3. SIMUILINK MODEL BASIC BLOCKS

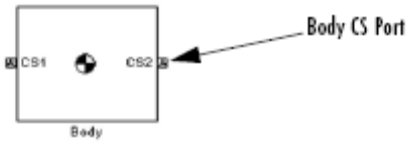
This appendix explains the basic blocks used in Simulink® SimMechanics to simulate mechanical systems; this is required to understand the operation of the simulations. The source of the material in this appendix is "SimMechanics for Use with Simulink® User's Guide" [11].

Body Block



Represents a rigid body. The representation has customizable properties that includes:

- Mass of body and its moment of inertia tensor
- The coordinates its center of gravity (CG)
- Body coordinate systems (CSs)



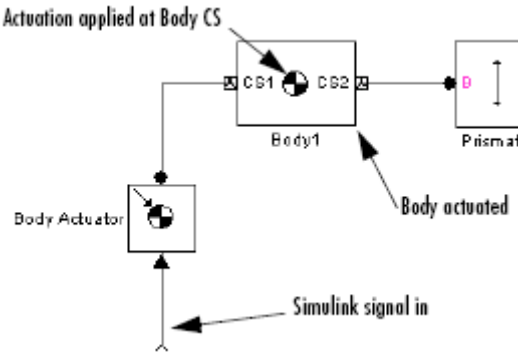
A rigid body is defined in space by the position of its CG (or center of mass) and its orientation in selected CS.

Body Actuator Block

Used to actuate a Body block with a force or a torque signal, representing a force/torque applied to the body:

- Force for translational motion
- Torque for rotational motion

The generalized force is a function of time specified by a Simulink input signal. This signal can be any Simulink signal, including a signal feedback from a Sensor block. The body Actuator applies the actuation signal in the reference coordinate system specified in the block dialog box.



The input is the Simulink input signal. The output is the connector port you connect to the Body block you want to actuate. You should carefully distinguish the Body Actuator from the Driver blocks:

- The Body Actuator block applies generalized forces to one body in a specified reference CS.

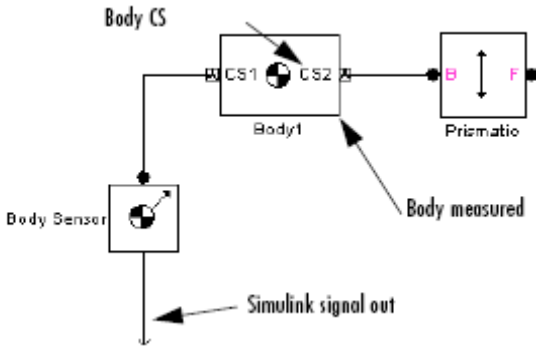
- The Driver blocks drive relative *degrees of freedom* between pairs of bodies.

Body Sensor Block

Used to measure physical quantities associated with the motion of a body. It senses the position, velocity, and/or acceleration of a body in the specified reference coordinate system.

It can measure one or more of these motion types:

- Translational motion of a body, in terms of linear position, velocity, and/or acceleration vectors
- Rotational motion of a body, in terms of angular velocity and/or acceleration vectors
- Rotational motion of a body, in terms of a 3-by-3 rotation matrix *R*



The input is the connector port connected to the Body being sensed. The output is a set of Simulink signals or one bundled Simulink signal of the position, velocity, and/or acceleration vector(s) and/or the rotation matrix of the body.

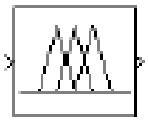
Planar Block

It represents a composite joint with two translational degrees of freedom (DoFs) and one rotational DoF. The rotation axis must be orthogonal to the plane defined by the two translation axes. Each side of the Joint block must be connected to a Body block at a Body coordinate system point. The origins of the Body coordinate systems must lie along the primitive axes, and the Body coordinate system origins on either side of the Joint must be spatially collocated points, to within assembly tolerances. You must connect any Joint block to two and only two Body blocks, and Joints have a default of two connector ports for connecting to base and follower Bodies.

Joint Sensor Block

Similar to the Body Sensor Block, it is used to measure the position, velocity, and/or acceleration of a joint primitive in a Joint block.

Fuzzy Logic Controller Block

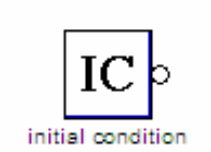


Used to implement a fuzzy logic controller (FLC), or generally, a fuzzy inference system (FIS), or a in Simulink models.

Multiplexer

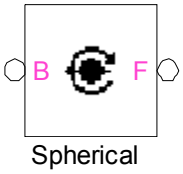
It combines multiple input signals into one output signal or a vector.

Joint Initial Condition Actuator Block



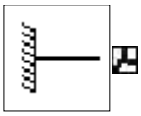
Used to define the initial value data of the prismatic and revolute joint primitives of a Joint block. The initial values are the positions and velocities of the joint primitives. and fully specify the initial state of motion (initial kinematic state) of those primitives.

Spherical Joint Block



Represents three rotational degrees of freedom (DoFs) at a single pivot point, a "ball-in-socket" joint. Two rotational DoFs specify a directional axis, and a third rotational DoF specifies rotation about that directional axis. The sense of each rotational DoF is defined by the right-hand rule, and the three rotations together form a right-handed system.

Ground Block



Used to represent a fixed ground point at rest in the absolute inertial World reference frame. Connecting it to a Joint prevents one side of that Joint from moving. You can also connect a Ground block to a Machine Environment block.

APPENDIX 4. ASSEMBLY PROGRAM

```
;*****************************************************************
;*  Name    : UNTITLED.BAS                                       *
;*  Author  : Faissal Abdel-Hady                                 *
;*  Notice  : Copyright (c) 2005                                 *
;*          : All Rights Reserved                                *
;*  Date    : 1/3/2005                                           *
;*  Version : 1.0                                                *
;*  Notes   :                                                    *
;*          :                                                    *
;*****************************************************************
     Device = 18F452
     Include "proton18_4.inc"



;*****************************************************************************
********
;   This file is the assembly code for a PIC18F452.
                *
;*****************************************************************************
********
;
*
;    Filename:  PID for Magnetic Ring Spinning system
*
;    Date: 12/24/2004
*
;    File Version: 1
*
;
*
;    Author:   Faissal Abdel Hady
*
;    Company:  Auburn University
*
;
*
;*****************************************************************************
********
;Configuration bits
; The __CONFIG directive defines configuration data within the .ASM
file.
; The labels following the directive are defined in the P18F452.INC
file.
; The PIC18FXX2 Data Sheet explains the functions of the configuration
bits.
```

```
@CONFIG_REQ
 @__CONFIG config1h, OSCS_OFF_1 & HS_OSC_1
 @__CONFIG config2l, BOR_ON_2 & BORV_20_2 & PWRT_ON_2
 @__CONFIG config2h, WDT_OFF_2 & WDTPS_128_2
 @__CONFIG config3h, CCP2MX_ON_3
 @__CONFIG config4l, STVR_ON_4 & LVP_OFF_4 & DEBUG_OFF_4


;*********************************************************************
********
;*********************************************************************
********
;* This section contains PID functions with interrupt.

;*********************************************************************
********
;
;PID Notes:

;     PROPORTIONAL = (system error * Pgain )

;     System error = error0:error1

;

;     INTEGRAL = (ACUMULATED ERROR  * Igain)

;     Accumulated error (a_error) = error0:error1 + a_Error0:a_Error2

;

;     DERIVATIVE = ((CURRENT ERROR  - PREVIOUS ERROR) * Dgain)

;     delta error(d_error) = errro0:error1 - p_error0:p_error1

;

;     Integral & Derivative control will be based off sample periods of
"x" time.
;     The above sample period should be based off the PLANT response

;     to control inputs.

;           SLOW Plant response = LONGER sample periods

;           FAST Plant response = SHORTER sample periods

;

;     If the error is equal to zero then no PID calculations are
completed.
;

;     The PID routine is passed the 16- bit errror data by the main
application
```

```
;     code through the error0:error1 variables.

;     The sign of this error is passed through the error sign bit:

;                pidStat1,err_sign

;     The PID outputs a 24-bit vaule in pidOut0:pidOut2 and the sign of
this
;     result is the pid_sign bit in the pidStat1 register.

;-------------------------------------------------------------------------
-

;*********************  Variable definitions used for math routines
************************
Symbol      _Z    =     STATUS.2
Symbol      _C    =     STATUS.0
Symbol LSB = 0
Symbol MSB = 7
;*********************  Variable definitions used for PID routines
************************
Symbol      aErr1Lim =        $0F   ;accumulative error limits (4000d)
Symbol      aErr2Lim =        $A0

Symbol      timer1Hi =        $fF   ;Timer1 timeout defined by timer1Lo
& timer1Hi
Symbol      timer1Lo =        $0A   ;this timout is based on Fosc/4

Symbol derivCountVal =  20          ;determies how often the derivative
term will be executed.
Dim pidStat1      As Byte                ;PID bit-status register
Dim pidStat2      As Byte                ;PID bit-status register2

;                                     pidStat1 register
;
      ____________________________________________________________________
_________________________
;     |  bit 7   |   bit 6    |  bit 5 |    bit 4   |   bit 3    |  bit
2   |   bit 1    |  bit 0   |
;     | pid_sign | d_err_sign |   mag  | p_err_sign | a_err_sign |
err_sign |  a_err_z   |  err_z   |
;
      |__________|____________|________|____________|____________|_____
_____|____________|__________|
;
Symbol err_z        = pidStat1.0               ;error zero flag, Zero
= set
Symbol a_err_z        = pidStat1.1             ;a_error zero flag,
Zero = set
Symbol err_sign     = pidStat1.2               ;error sign flag, Pos =
set/ Neg = clear
Symbol a_err_sign   = pidStat1.3               ;a_error sign flag, Pos
= set/ Neg = clear
Symbol p_err_sign   = pidStat1.4               ;a_error sign flag, Pos
= set/ Neg = clear
```

```
Symbol mag          = pidStat1.5                ;set = AARGB magnitude,
clear = BARGB magnitude
Symbol d_err_sign   = pidStat1.6                ;d_error sign flag, Pos
= set/ Neg = clear
Symbol pid_sign     = pidStat1.7                ;PID result sign flag,
Pos = set/ Neg = clear

;     _____________________________________ pidStat2
register_____________________________________
;     | bit 7 |  bit 6  |  bit 5   |    bit 4   |   bit 3    | bit 2 |
bit 1    |  bit 0   |
;     |       |         |          |            |            |       |
| d_err_z  |
;
      |_______|_________|__________|____________|____________|_______|_
___________|__________|

Symbol d_err_z  = pidStat2.0                    ;d_error zero flag,
Zero = set


    Dim AARGB0 As Byte
    Dim AARGB1 As Byte
    Dim AARGB2 As Byte
    Dim AARGB3 As Byte
    Dim BARGB0 As Byte
    Dim BARGB1 As Byte
    Dim BARGB2 As Byte
    Dim BARGB3 As Byte
    Dim REMB0  As Byte
    Dim REMB1  As Byte
    Dim REMB2  As Byte
    Dim REMB3  As Byte
    Dim TEMP   As Byte
    Dim TEMPB0 As Byte
    Dim TEMPB1 As Byte
    Dim TEMPB2 As Byte
    Dim TEMPB3 As Byte
    Dim ZARGB0 As Byte
    Dim ZARGB1 As Byte
    Dim ZARGB2 As Byte
    Dim CARGB2 As Byte
    Dim AEXP   As Byte
    Dim LOOPCOUNT As Byte

    ;pid_data1    UDATA
    Dim Raw As Word
    Dim read_sens0  As Raw.BYTE1                ;sensor reading low
byte
    Dim read_sens1      As Raw.BYTE0                ; sensor reading
high byte
    Dim temp_word As Word
    Dim count_time      As Byte
    Dim percent_out  As Byte
    Dim percent_err As Byte
```

```
#DEFINE     pid_100                 ;comment out if not using a 0 -
100% scale


;***** VARIABLE DEFINITIONS

;pid_data
    Dim sign_out As String *1
    Dim derivCount      As Byte                 ;This value determins
how many times the Derivative term is
                                            ;calculated based on each
Integral term.
    Dim pidout      As DWord
    Dim pidOut0         As pidout.BYTE2               ;24-bit Final
Result of PID for the "Plant" extracted from 32-bit
    Dim pidOut1         As pidout.BYTE1
    Dim pidOut2         As pidout.BYTE0

    Dim error0          As Byte                 ;16-bit error, passed
to the PID
    Dim error1          As Byte

    Dim a_Error0  As Byte                 ;24-bit accumulated error,
used for Integral term
    Dim a_Error1  As Byte
    Dim a_Error2  As Byte

    Dim p_Error0  As Byte                 ;16-bit previous error, used
for Derivative term
    Dim p_Error1  As Byte

    Dim d_Error0  As Byte                 ;16-bit delta error (error -
previous error)
    Dim d_Error1  As Byte

    Dim prop0           As Byte                 ;24-bit proportional
value
    Dim prop1           As Byte
    Dim prop2           As Byte

    Dim integ0          As Byte                 ;24-bit Integral value
    Dim integ1          As Byte
    Dim integ2          As Byte

    Dim deriv0          As Byte                 ;24-bit Derivative
value
    Dim deriv1          As Byte
    Dim deriv2          As Byte

    Dim kp              As Byte                 ;8-bit proportional
Gain
    Dim ki              As Byte                 ;8-bit integral Gain
    Dim kd              As Byte                 ;8-bit derivative Gain
```

```
    Dim tempReg         As Byte                 ;temporary register
    Dim Value As Word               ; Quantasize the result
      Dim Volts As Byte
      Dim Millivolts As Word
    Dim puls_width As Float
    Dim channel As Byte
    Symbol channel_number = channel.0

;*****************************************************************
********
;High priority interrupt vector
; This code will start executing when a high priority interrupt occurs
or
; when any interrupt occurs if interrupt priorities are not enabled.

   ON_INTERRUPT GoTo PidInterrupt
   movlw    0x00
   movwf    INTCON
   movlw    0x00
   movwf    T1CON
   movlw    0x01
   movwf    IPR1
;'    INTCON      = %00000000
;    T1CON = $00
 ;   IPR1  = $01
    movlw   timer1Hi                    ;reload T1 registers with
constant time count (user defined)
      movwf TMR1H
      movlw timer1Lo
      movwf TMR1L
    movlw    0x01
    movwf    T1CON
    movlw    0xC0
    movwf    INTCON
;     T1CON = $01                   ; Turn on Timer1, prescaler = 1
;     INTCON      = %11000000       ; 7-Global Interrupt(GIE),
                            ; 6-Peripheral Int(PEIE),
                            ; 5- TMR0 Int. Enable(TMR0IE),
                            ; 4- INT0 External Int Enable (INT0IE),
                            ; 3- RB Port Change Int Enable(RBIE),
                            ; 2- TMR0 Overflow Int Flage(TMR0IF),
                            ; 1- INT0 External Int Flag(INT0IF),
                            ; 0- RB Port Change Int Flag(RBIF)

    movlw    0x01
    movwf    PIE1
;     PIE1 = %00000001        ; 0- TMR1IE TMR1 OVERFLOW INTERRUPT
ENABLE BIT
;        DelayMS 500
        GoTo      Main
;*****************************************************************

;*****************************************************************
*;
; Function: PidInterrupt                                        ;
```

```
;
;
; PreCondition: This Routine will be called by the application's main
      ;
;           code.                                          ;
;
;
; Overview: When Timer 1 overlfows, an updated value for the Integral
      ;
;         term will be calculated. An updated value for the derivative;
;     term will be calculated if derivCount = 0. This routine     ;
;     will check for error = 0, if this is true,then the routine  ;
;     will return back to the main line code.                     ;
;                                                    ;
; Input: pidStat1, a_error                                    ;
;
;
; Output: Integral & Derivative terms, Timer1 registers reloaded
;
;
;
; Side Effects: W register is changed                                ;
;
;
; Stack requirement: 4 levels deep max.                              ;
;
;
;*********************************************************************
*;
 ASM
PidInterrupt:
      bcf    PIR1,TMR1IF     ; clear int flag
      btfsc err_z       ;Is error = 00 ?
      Return                                    ;YES, done.

      Call  GetA_Error              ;get a_error, is a_error = 00?
reached limits?

derivative_ready:
      decfsz      derivCount,f            ;is it time to calculate
d_error ?
      bra         skip_deriv              ;NO, finish ISR
      Call  GetDeltaError           ;error - p_error

      movlw derivCountVal                       ;prepare for next delta
error
      movwf derivCount                  ;delta error = TMR1H:TMR1L *
derivCount

skip_deriv:
      movlw timer1Hi                    ;reload T1 registers with
constant time count (user defined)
      movwf TMR1H
      movlw timer1Lo
      movwf TMR1L
```

```
      retfie      fast                                    ;return back to
the application's ISR
;*********************************************************************
**********************
ENDASM
;*************
GetSensorReading:

; ******   loop to wait 2us before starting the A/D conversion at 40MHz
******
      bcf         PIE1,TMR1IE             ;disable T1 interrupt
            Raw = ADIn channel_number

      bsf         PIE1,TMR1IE             ;enable T1 interrupt
      Return
;*********************************************************************
********
;*********************************************************************
*;
; Function: PidInit
      ;
;
;
; PreCondition: Called by the application code for PID initalization
      ;
;
;
; Overview: PID variables are cleared, PID gains are given values,
      ;
;                 flags are initialized.
                        ;
;
                                      ;
; Input:
                               ;
;
;
; Output: none
;
;
;
; Side Effects: W register is changed
              ;
;
;
; Stack requirement: 1 levels deep
;
;
;
;*********************************************************************
*;
ASM
PidInitalize:
      clrf  error0
      clrf  error1
```

```
      clrf  a_Error0
      clrf  a_Error1
      clrf  a_Error2
      clrf  p_Error0
      clrf  p_Error1
      clrf  d_Error0
      clrf  d_Error1

      clrf  prop0
      clrf  prop1
      clrf  prop2
      clrf  integ0
      clrf  integ1
      clrf  integ2
      clrf  deriv0
      clrf  deriv1
      clrf  deriv2

      clrf  kp
      clrf  ki
      clrf  kd

      clrf  pidOut0
      clrf  pidOut1
      clrf  pidOut2

      clrf  AARGB0
      clrf  AARGB1
      clrf  AARGB2
      clrf    AARGB3
      clrf  BARGB0
      clrf  BARGB1
      clrf  BARGB2
      clrf    BARGB3
;-----------------------------------------------------------------------
-------------------------------------
; replace this section with key pad input for the three variables
Kp,Ki,Kd and derivCount -------------------
      movlw 0xFF                          ;10 x 16, Kp, Ki & Kd are 8-
bit vlaues that cannot exceed 255
      movwf kp                                  ;Enter the PID gains
scaled by a factor of 16, max = 255

      movlw 0x14                          ;10 x 16
      movwf ki

      movlw 0xA0                          ;10 x 16
      movwf kd

      movlw 0x0A
      movwf derivCount                    ;derivative action =
TMR1H:TMR1L * derivCount
;-----------------------------------------------------------------------
-------------------------------------
      bcf         err_z             ;start w/error not equal to zero
```

```
      bsf         a_err_z           ;start w/a_error equal to zero
      bsf         d_err_z           ;start w/d_error equal to zero
      bsf         p_err_sign        ;start w/ previous error = positive
      bsf         a_err_sign        ;start w/ accumulated error =
positive

      bcf         PIR1,TMR1IF                  ;clear T1 flag
      bsf         PIE1,TMR1IE                  ;enable T1 interrupt

      movlw 0x01             ;%00000001         configure T1 for Timer
operation from Fosc/4
      movwf T1CON
      movlw timer1Hi                      ;load T1 registers with 5ms
count
      movwf TMR1H
      movlw timer1Lo
      movwf TMR1L

      Return                                          ;return back to
the main application code
;     endasm
;*********************************************************************
*****************

;*********************************************************************
*;
; Function: PidMain
      ;
;
;
; PreCondition: error0:erro1 are loaded with the latest system error
;
;
;
; Overview: This is the routine that the application code will call
      ;
;                 to get a PID correction value. First, the error is
checked     ;
;                 to determine if it is zero, if this is true, then the
PID   ;
;                 code is complete.
                        ;
;
                                    ;
; Input: error0:error1, sign of the error: pidStat1,err_sign
;
;
;
; Output: prop0:prop2
;
;
;
; Side Effects: W register is changed
              ;
```

```
;
;
; Stack requirement: 5 levels deep
;
;
;
;*****************************************************************
*;

PidMain:

      bcf         PIE1,TMR1IE             ;disable T1 interrupt
#IFDEF            pid_100                       ;if using % scale then
scale up PLANT error
      movlw .40                           ; 0 - 100% == 0 - 4000d

      mulwf percent_err,1           ;40 * percent_err --> PRODH:PRODL
      movff PRODH,error0
      movff PRODL,error1            ;percentage has been scaled and
available in error0:error1
#ENDIF
      movlw 0
      cpfseq      error0                  ;Is error0 = 00 ?
      bra         call_pid_terms          ;NO, done checking

      cpfseq      error1                  ;YES, Is error1 = 00 ?
      bra         call_pid_terms          ;NO, start proportional term
      bsf         err_z       ;YES, set error zero flag
      bsf         PIE1,TMR1IE             ;enable T1 interrupt
      Return                              ;return back to the
main application code

call_pid_terms:
      Call  Proportional            ;NO, start with proportional term
      Call  Integral                ;get Integral term
      Call  Derivative              ;get Derivative term

      Call  GetPidResult            ;get the final PID result that will
go to the system
      bsf         PIE1,TMR1IE             ;enable T1 interrupt
      Return                              ;return back to the
main application code

;*****************************************************************
*************************

;*****************************************************************
*;
; Function: Proportional
;
;
;
; PreCondition: error0:erro1 are loaded with the latest system error
;
```

```
;
;
; Overview: This routine will multiply the system's 16-bit error by the
;
;           proportional gain(Kp) --> error0:error1 * Kp
      ;
;
                                    ;
; Input: error0:error1, sign of the error: pidStat1,err_sign
;
;
;
; Output: prop0:prop2
;
;
;
; Side Effects: W register is changed
              ;
;
;
; Stack requirement: 2 levels deep
;
;
;
;*********************************************************************
*;
Proportional:
      clrf  BARGB0
      movff kp,BARGB1
      movff error0,AARGB0
      movff error1,AARGB1
      Call  FXM1616U                ;proportional gain * error

      movff AARGB1,prop0            ;AARGB2 --> prop0
      movff AARGB2,prop1            ;AARGB3 --> prop1
      movff AARGB3,prop2            ;AARGB4 --> prop2
      Return                                ;return to mainline
code


;*********************************************************************
*;
; Function: Integral
      ;
;
;
; PreCondition:  error0:erro1 are loaded with the latest system error
      ;
;
;
; Overview: This routine will multiply the system's 16-bit accumulated
      ;
;           error by the integral gain(Ki)--> a_Error0:a_Error1 * Ki
      ;
```

```
;
                                    ;
; Input: a_Error0:a_Error1, sign of a_Error: pidStat1,a_err_sign
;
;
;
; Output: integ0:integ2
;
;
;
; Side Effects: W register is changed
              ;
;
;
; Stack requirement: 2 levels deep
;
;
;
;*********************************************************************
*;
Integral:
      btfsc a_err_z                  ;Is a_error = 0
      bra         integral_zero           ;Yes

      clrf  BARGB0                        ;No
      movff ki,BARGB1               ;move the integral gain into BARGB1
      movff a_Error1,AARGB0
      movff a_Error2,AARGB1
      Call  FXM1616U                ;Integral gain * accumulated error

      movff AARGB1,integ0           ;AARGB1 --> integ0
      movff AARGB2,integ1           ;AARGB2 --> integ1
      movff AARGB3,integ2           ;AARGB3 --> integ2
      Return                                ;return
integral_zero:
      clrf  integ0                        ;a_error = 0, clear Integral
term
      clrf  integ1
      clrf  integ2
      Return


;*********************************************************************
*;
; Function: Derivative
      ;
;
;
; PreCondition:  error0:erro1 are loaded with the latest system error
;
;
;
; Overview: This routine will multiply the system's 16-bit delta
      ;
```

```
;           error by the derivative gain(Kd) --> d_Error0:d_Error1 * Kd
      ;
;                 d_Error0:d_Error1 = error0:error1 - p_Error0:p_Error1
      ;
;
                                        ;
; Input: d_Error0:d_Error1, pidStat2,d_err_z
         ;
;
;
; Output: deriv0:deriv2
;
;
;
; Side Effects: W register is changed
              ;
;
;
; Stack requirement: 2 levels deep
;
;
;
;*********************************************************************
*;
Derivative:
      btfsc d_err_z                   ;Is d_error = 0?
      bra         derivative_zero          ;YES

      movff d_Error1,BARGB1         ;result ---> BARGB1
      movff d_Error0,BARGB0         ;result ---> BARGB0
      movff kd,AARGB1
      clrf  AARGB0
      Call  FXM1616U                ;Derivative gain * (error_l -
prv_error1)

      movff AARGB1,deriv0           ;AARGB1 --> deriv0
      movff AARGB2,deriv1           ;AARGB2 --> deriv1
      movff AARGB3,deriv2           ;AARGB3 --> deriv2
      Return                                ;return
derivative_zero:
      clrf  deriv0                        ;d_error = 0, clear
Derivative term
      clrf  deriv1
      clrf  deriv2
      Return


; ENDASM
;*********************************************************************
*;
; Function: GetPidResult
      ;
;
;
```

```
; PreCondition: Proportional, Integral & Derivative terms have been
      ;
;                       calculated. The Timer1 interrupt is disabled
within            ;
;                       this routine to avoid corruption of the PID
result.           ;
;
;
; Overview: This routine will add the PID terms and then scale down
      ;
;                 the result by 16. This will be the final result that
is    ;
;                 calcualted by the PID code.
                        ;
;
                                      ;
; Input: prop0:prop2, integ0:integ2, deriv0:deriv2
          ;
;
;
; Output: pidOut0:pidOut2
;
;
;
; Side Effects: W register is changed
              ;
;
;
;
;
;*********************************************************************
*;
GetPidResult:
      movff prop0,AARGB0                  ;load Prop term & Integral
term
      movff prop1,AARGB1
      movff prop2,AARGB2
      movff integ0,BARGB0
      movff integ1,BARGB1
      movff integ2,BARGB2

      Call  SpecSign                      ;YES, call routine for
add/sub sign numbers
      btfss mag               ;which is greater in magnitude ?
      bra         integ_mag                     ;BARGB is greater in
magnitude
      bra         prop_mag                      ;AARGB is greater in
magnitude

integ_mag:                                      ;integ > prop
      bcf         pid_sign          ;PID result is negative
      btfsc a_err_sign
      bsf         pid_sign          ;PID result is positive
      bra   add_derivative                ;(Prop + Integ) + derivative
```

```
prop_mag:                                          ;integ < prop
      bcf         pid_sign          ;PID result is negative
      btfsc err_sign
      bsf         pid_sign          ;PID result is positive

add_derivative:
      movff deriv0,BARGB0                ;YES, AARGB0:AARGB2 has
result of Prop + Integ
      movff deriv1,BARGB1                ;load derivative term
      movff deriv2,BARGB2

      movff pidStat1,tempReg        ;pidStat1 ---> tempReg
      movlw 0xc0                    ;prepare for sign check of bits 7 &
6
      andwf tempReg,f

      movf  tempReg,w                    ;check error sign & a_error
sign bits
      sublw 0x00
      btfsc STATUS,Z
      bra         add_neg_d                    ;bits 7 & 6 (00) are
NEGATIVE, add them
      bra         other_combo_d                ;bits 7 & 6 not equal
to 00
add_neg_d:
      Call  _24_BitAdd                   ;add negative sign values
      bra         scale_down                   ;scale result

other_combo_d:
      movf  tempReg,w
      sublw 0xC0
      btfsc STATUS,Z
      bra         add_pos_d                    ;bits 7 & 6 (11) are
POSITIVE, add them
      bra         find_mag_sub_d               ;bits 7 & 6 (xx) are
different signs , subtract them
add_pos_d:
      Call  _24_BitAdd                   ;add positive sign values
      bra         scale_down                   ;scale result
find_mag_sub_d:
      Call  MagAndSub                    ;subtract unlike sign numbers
      btfss mag               ;which is greater in magnitude ?
      bra         deriv_mag                    ;BARGB is greater in
magnitude
      bra         scale_down                   ;derivative term < part
pid term, leave pid_sign as is

deriv_mag:                                         ;derivative term > part
pid term
      bcf         pid_sign          ;PID result is negative
      btfsc d_err_sign
      bsf         pid_sign          ;PID result is positive

scale_down:
```

```
      clrf  BARGB0                              ;(Prop + Integ + Deriv)
/ 16 = FINAL PID RESULT to plant
      movlw 0x10
      movwf BARGB1
      Call  FXD2416U
      movff AARGB2,pidOut2                ;final result ---> pidOut2
      movff AARGB1,pidOut1                ;final result ---> pidOut1
      movff AARGB0,pidOut0                ;final result ---> pidOut0

#IFDEF            pid_100                           ;Final result
needs to be scaled down to  0 - 100%
      movlw 0x06                          ;% ratio for propotional &
integral & derivative
      movwf BARGB0
      movlw 0x40
      movwf BARGB1

      Call  FXD2416U                      ;pidOut0:pidOut2 / % ratio =
0 - 100% value
      movf  AARGB2,W                      ;AARGB2 --> percent_out
      movwf percent_out                   ;error has been scaled down
and is now available in a 0 -100% range
#ENDIF

      Return                                          ;return to
mainline code


;*********************************************************************
*;
; Function: GetA_Error
      ;
;
;
; PreCondition: Proportional term has been calculated
      ;
;
;
; Overview: This routine will add the current error with all of the
      ;
;                 previous errors. The sign of the accumulated error
will  ;
;                 also be determined.     After the accumulated error
is                ;
;                 calculated then it is checked if it = 00 or as
exceeded          ;
;                 the defined limits.
                            ;
;
                                      ;
; Input: a_Error0:a_Error1, error0:error1
          ;
;
;
```

```
; Output: a_Error0:a_Error1 (updated value)
;
;
;
; Side Effects: W register is changed
              ;
;
;
;
;*****************************************************************
*;
GetA_Error:
      movff a_Error0,BARGB0               ;load error & a_error
      movff a_Error1,BARGB1
      movff a_Error2,BARGB2
      clrf  AARGB0
      movff error0,AARGB1
      movff error1,AARGB2

      Call  SpecSign                      ;call routine for add/sub
sign numbers
      btfss mag               ;which is greater in magnitude ?
      bra         a_err_zero                    ;bargb, keep sign as is
or both are same sign

      bcf         a_err_sign        ;aargb, make sign same as error,
a_error is negative
      btfsc err_sign
      bsf         a_err_sign        ;a_error is positive

a_err_zero:
      bcf         a_err_z           ;clear a_error zero flag
      movlw 0
      cpfseq      AARGB0                              ;is byte 0 = 00
      bra         chk_a_err_limit               ;NO, done checking

      cpfseq      AARGB1                              ;is byte 1 = 00
      bra         chk_a_err_limit               ;NO, done checking

      cpfseq      AARGB2                              ;is byte 2 = 00
      bra         chk_a_err_limit               ;NO, done checking
      bsf         a_err_z           ;YES, set zero flag

      movff AARGB0,a_Error0               ;store the a_error
      movff AARGB1,a_Error1
      movff AARGB2,a_Error2
      Return                                          ;a_error = 00,
return

chk_a_err_limit:
      movff AARGB0,a_Error0               ;store the a_error
      movff AARGB1,a_Error1
      movff AARGB2,a_Error2
```

```
      movlw 0                                    ;a_error reached
limits?
      cpfseq      a_Error0                       ;Is a_Error0 > 0 ??, if
yes limit has been exceeded
      bra         restore_limit                  ;YES, restore limit
value

      cpfseq      a_Error1                       ;Is a_Error1 = 0 ??, if
yes, limit not exceeded
      bra         chk_a_Error1                   ;NO
      Return                                           ;YES
chk_a_Error1:
      movlw aErr1Lim
      cpfsgt      a_Error1                       ;Is a_Error1 >
aErr1Lim??
      bra         equal_value                    ;NO, check for a_Error1
= aErr1Lim ?
      bra         restore_limit                  ;YES, restore limit
value
equal_value:
      cpfseq      a_Error1                       ;a_Error1 = aErr1Lim?
      Return                                           ;no, done
checking a_error
chk_a_Error2:
      movlw aErr2Lim                       ;Yes, a_Error1 = aErr1Lim
      cpfsgt      a_Error2                       ;Is a_Error2 > aErr2Lim
??
      Return                                           ;NO, return to
mainline code

restore_limit:
      clrf  a_Error0                       ;YES, a_error limit has been
exceeded
      movlw aErr1Lim
      movwf a_Error1
      movlw aErr2Lim
      movwf a_Error2
      Return                                           ;return to
mainline code


;*********************************************************************
*;
; Function: GetDeltaError
;
;
;
; PreCondition: The derivative routine has been called to calculate the
      ;
;                       derivative term.
                        ;
;
;
; Overview: This routine subtracts the previous error from the current
      ;
```

```
;                 error.
                                    ;
;
                                    ;
; Input: P_Error0:p_Error1, error0:error1
          ;
;
;
; Output: d_Error0:d_Error1, d_Error sign
      ;
;
;
; Side Effects: W register is changed
              ;
;
;
;
;*********************************************************************
*;
GetDeltaError:
      clrf  AARGB0                              ;load error and p_error
      movff error0,AARGB1
      movff error1,AARGB2
      clrf  BARGB0
      movff p_Error0,BARGB1
      movff p_Error1,BARGB2

      movf  pidStat1,w                    ;pidStat1 ---> tempReg
      movwf tempReg                             ;prepare for sign check
of bits 4 & 2
      movlw 0x14
      andwf tempReg,f

      movf  tempReg,w                     ;check error sign & a_error
sign bits
      sublw 0x00
      btfsc STATUS,Z
      bra         p_err_neg                     ;bits 4 & 2 (00) are
NEGATIVE,
      bra         other_combo2                  ;bits 4 & 2 not equal
to 00
p_err_neg:
      Call  MagAndSub
      bcf         d_err_sign        ;d_error is negative
      btfsc p_err_sign        ;make d_error sign same as p_error sign
      bsf         d_err_sign        ;d_error is positive
      bra         d_error_zero_chk        ;check if d_error = 0

other_combo2:
      movf  tempReg,w
      sublw 0x14
      btfsc STATUS,Z
      bra         p_err_pos                     ;bits 4 & 2 (11) are
POSITIVE
```

```
      bra   p_err_add                     ;bits 4 & 2 (xx) are
different signs

p_err_pos:
      Call  MagAndSub
      bcf         d_err_sign        ;d_error is negative
      btfsc p_err_sign        ;make d_error sign same as p_error sign
      bsf         d_err_sign        ;d_error is positive
      bra         d_error_zero_chk        ;check if d_error = 0
p_err_add:
      Call  _24_BitAdd                    ;errors are different sign
      bcf         d_err_sign        ;d_error is negative
      btfsc err_sign          ;make d_error sign same as error sign
      bsf         d_err_sign        ;d_error is positive

d_error_zero_chk:
      movff AARGB1,d_Error0
      movff AARGB2,d_Error1

      movff error0,p_Error0               ;load current error into
previous for next deriavtive term
      movff error1,p_Error1               ;load current error into
previous for next deriavtive term
      bcf         p_err_sign        ;make p_error negative
      btfsc err_sign          ;make p_error the same sign as error
      bsf         p_err_sign        ;make p_error positive

      bcf         d_err_z                      ;clear delta error
zero bit
      movlw 0
      cpfseq      d_Error0                    ;is d_error0 = 00
      Return                                        ;NO, done
checking

      cpfseq      d_Error1                    ;YES, is d_error1 = 00
      Return                                        ;NO, done
checking
      bsf         d_err_z           ;set delta error zero bit
      Return                                        ;YES, return to
ISR

;*********************************************************************
*;
; Function: SpecSign
      ;
;
;
; PreCondition: The sign bits in pidStat1 have been set or cleared
      ;
;                     depending on the variables they represent.
                  ;
;
;
; Overview: This routine takes the numbers loaded into the math
      ;
```

```
;                       variables (AARGB, BARGB) and determines whether
they  ;
;                       need to be added or subtracted based on their
sign        ;
;                       which is located in the pidStat1 register.
                ;
;
                                        ;
; Input: pidStat1
                    ;
;
;
; Output: add/sub results in math variables (AARGB, BARGB)
      ;
;
;
; Side Effects: W register is changed
              ;
;
;
;
;*********************************************************************
*;
SpecSign:
      movff pidStat1,tempReg        ;pidStat1 ---> tempReg
      movlw 0x0c          ;%00001100       prepare for sign check of
bits 3 & 2
      andwf tempReg,f

      movf  tempReg,w                     ;check error sign & a_error
sign bits
      sublw 0x00
      btfsc STATUS,Z
      bra         add_neg                             ;bits 3 & 2 are
NEGATIVE (00), add them
      bra         other_combo                   ;bits 3 & 2 not equal
to 00
add_neg:
      Call  _24_BitAdd                    ;add negative sign values
      Return

other_combo:
      movf  tempReg,w
      sublw 0x0C
      btfsc STATUS,Z
      bra         add_pos                             ;bits 3 & 2 are
POSITIVE (11), add them
      bra         find_mag_sub                ;bits 3 & 2 are
different signs (xx), subtract them
add_pos:
      Call  _24_BitAdd                    ;add positive sign values
      Return
find_mag_sub:
      Call  MagAndSub                     ;subtract unlike sign numbers
```

```
      Return

;*********************************************************************
*;
; Function: MagAndSub
      ;
;
;
; PreCondition: This routine has been called by SpecSign because the
      ;
;                        numbers being worked on are different in sign.
            ;
;
;
; Overview: This routine will detemine which math variable
      ;
;                 (AARGB or BARGB) is greater in number magnitude and
then  ;
;                 subtract them, the sign of the result will be
determined by     ;
;                 the values in the math variables and their signs.
            ;
;
                                        ;
; Input: pidStat1
                        ;
;
;
; Output: add/sub results in math variables (AARGB, BARGB)
      ;
;
;
; Side Effects: W register is changed
               ;
;
;
;
;*********************************************************************
*;
MagAndSub:
      movf  BARGB0,w
      subwf AARGB0,w                      ;AARGB0 - BARGB0 --> W
      btfsc STATUS,Z                      ;= zero ?
      bra         check_1                           ;YES
      btfsc STATUS,C                      ;borrow ?
      bra         aargb_big                   ;AARGB0 > BARGB0, no
borrow
      bra         bargb_big                   ;BARGB0 > AARGB0,
borrow
check_1:
      movf  BARGB1,w
      subwf AARGB1,w                      ;AARGB1 - BARGB1 --> W
      btfsc STATUS,Z                      ;= zero ?
      bra         check_2                           ;YES
```

```
      btfsc STATUS,C                        ;borrow ?
      bra         aargb_big                       ;AARGB1 > BARGB1, no
borrow
      bra         bargb_big                       ;BARGB1 > AARGB1,
borrow

check_2:
      movf  BARGB2,w                        ;AARGB2 - BARGB2 --> W
      subwf AARGB2,w
      btfsc STATUS,C                        ;borrow ?
      bra         aargb_big                       ;AARGB2 > BARGB2, no
borrow
      bra         bargb_big                       ;BARGB2 > AARGB2,
borrow

aargb_big:
      Call  _24_bit_sub
      bsf         mag               ;AARGB is greater in magnitude
      Return

bargb_big:
      movff BARGB0,tempReg                  ;swap AARGB0 with BARGB0
      movff AARGB0,BARGB0
      movff tempReg,AARGB0

      movff BARGB1,tempReg                  ;swap AARGB1 with BARGB1
      movff AARGB1,BARGB1
      movff tempReg,AARGB1

      movff BARGB2,tempReg                  ;swap AARGB2 with BARGB2
      movff AARGB2,BARGB2
      movff tempReg,AARGB2

      Call  _24_bit_sub                     ;BARGB > AARGB
      bcf         mag               ;BARGB is greater in magnitude
      Return


;*****************************************************************
*******

; Math routines for PID control
;-----------------------------------------------------------------
;           24-BIT ADDITION
_24_BitAdd:
;     GLOBAL      _24_BitAdd
      movf  BARGB2,w
      addwf AARGB2,f

      movf  BARGB1,w
      btfsc _C
      incfsz      BARGB1,w
      addwf AARGB1,f

      movf  BARGB0,w
```

```
      btfsc _C
      incfsz      BARGB0,w
      addwf AARGB0,f
      Return

;-----------------------------------------------------------------------
;           24-BIT SUBTRACTION
_24_bit_sub:
;     GLOBAL      _24_bit_sub
      movf  BARGB2,w
      subwf AARGB2,f

      movf  BARGB1,w
      btfss STATUS,C
      incfsz      BARGB1,w
      subwf AARGB1,f

      movf  BARGB0,w
      btfss STATUS,C
      incfsz      BARGB0,w
      subwf AARGB0,f
      Return

;-----------------------------------------------------------------------
---
;       16x16 Bit Unsigned Fixed Point Multiply 16 x 16 -> 32

FXM1616U:
;           GLOBAL      FXM1616U

            movff AARGB1,TEMPB1

            movf  AARGB1,W
            mulwf BARGB1
            movff PRODH,AARGB2
            movff PRODL,AARGB3

            movf  AARGB0,W
            mulwf BARGB0
            movff PRODH,AARGB0
            movff PRODL,AARGB1

            mulwf BARGB1
            movf  PRODL,W
            addwf AARGB2,F
            movf  PRODH,W
            addwfc      AARGB1,F
            clrf  WREG
            addwfc      AARGB0,F

            movf  TEMPB1,W
            mulwf BARGB0
            movf  PRODL,W
            addwf AARGB2,F
            movf  PRODH,W
```

```
            addwfc      AARGB1,F
            clrf  WREG
            addwfc      AARGB0,F

            retlw 0x00

;-----------------------------------------------------------------------
FXD2416U:
;           GLOBAL            FXD2416U
            clrf        REMB0
            clrf        REMB1
            clrf        WREG
            tstfsz            BARGB0
            GoTo        D2416BGT1
            movff       BARGB1,BARGB0
            Call        FXD2408U
            movff       REMB0,REMB1
            clrf        REMB0
            retlw       0x00

D2416BGT1:
            cpfseq            AARGB0
            GoTo        D2416AGTB
            movff       AARGB1,AARGB0
            movff       AARGB2,AARGB1
            Call        FXD1616U

            movff       AARGB1,AARGB2
            movff       AARGB0,AARGB1
            clrf        AARGB0
            retlw       0x00
D2416AGTB:
            movff       AARGB2,AARGB3
            movff       AARGB1,AARGB2
            movff       AARGB0,AARGB1
            clrf        AARGB0

            movff       AARGB0,TEMPB0
            movff       AARGB1,TEMPB1
            movff       AARGB2,TEMPB2
            movff       AARGB3,TEMPB3

            movlw       0x02              ; set loop count
            movwf       AEXP

            movlw       0x01
            movwf       ZARGB0

            btfsc       BARGB0,MSB
            GoTo        D2416UNRMOK

            Call        DGETNRMD          ; get normalization factor
            movwf       ZARGB0

            mulwf       BARGB1
```

```
            movf        BARGB0,W
            movff       PRODL,BARGB1
            movff       PRODH,BARGB0
            mulwf       ZARGB0
            movf        PRODL,W
            addwf       BARGB0,F

            movf        ZARGB0,W
            mulwf       AARGB3
            movff       PRODL,TEMPB3
            movff       PRODH,TEMPB2
            mulwf       AARGB1
            movff       PRODL,TEMPB1
            movff       PRODH,TEMPB0
            mulwf       AARGB2
            movf        PRODL,W
            addwf       TEMPB2,F
            movf        PRODH,W
            addwf       TEMPB1,F

D2416UNRMOK:
            bcf         _C
            clrf        TBLPTRH
            rlcf        BARGB0,W
            rlcf        TBLPTRH,F
            addlw       Low (IBXTBL256+1) ; access reciprocal table
            movwf       TBLPTRL
            movlw       High (IBXTBL256)
            addwfc            TBLPTRH,F
            TBLRD       *-

D2416ULOOP:
            movff       TEMPB0,AARGB0
            movff       TEMPB1,AARGB1

            Call        FXD1608U2         ; estimate quotient digit

            btfss       AARGB0,LSB
            GoTo        D2416UQTEST

            setf        AARGB1
            movff       TEMPB1,REMB0
            movf        BARGB0,W
            addwf       REMB0,F

            btfsc       _C
            GoTo        D2416UQOK

D2416UQTEST:
            movf        AARGB1,W          ; test
            mulwf       BARGB1

            movf        PRODL,W
            subwf       TEMPB2,W
            movf        PRODH,W
```

```
            subwfb          REMB0,W

            btfsc       _C
            GoTo        D2416UQOK

            decf        AARGB1,F

            movf        BARGB0,W
            addwf       REMB0,F

            btfsc       _C
            GoTo        D2416UQOK

            movf        AARGB1,W
            mulwf       BARGB1

            movf        PRODL,W
            subwf       TEMPB2,W
            movf        PRODH,W
            subwfb          REMB0,W

            btfss       _C
            decf        AARGB1,F

D2416UQOK:
            movff       AARGB1,ZARGB1

            movf        AARGB1,W
            mulwf       BARGB1
            movf        PRODL,W
            subwf       TEMPB2,F
            movf        PRODH,W
            subwfb          TEMPB1,F

            movf        AARGB1,W
            mulwf       BARGB0
            movf        PRODL,W
            subwf       TEMPB1,F
            movf        PRODH,W
            subwfb          TEMPB0,F

            btfss       TEMPB0,MSB      ; test
            GoTo        D2416QOK
            decf        ZARGB1,F

            movf        BARGB1,W
            addwf       TEMPB2,F
            movf        BARGB0,W
            addwfc          TEMPB1,F

D2416QOK:
            dcfsnz          AEXP,F                  ; is loop done?
            GoTo        D2416FIXREM

            movff       ZARGB1,ZARGB2
```

```
            movff       TEMPB1,TEMPB0
            movff       TEMPB2,TEMPB1
            movff       TEMPB3,TEMPB2

            GoTo        D2416ULOOP

D2416FIXREM:
            movff       TEMPB1,REMB0
            movff       TEMPB2,REMB1

            movlw       0x01
            cpfsgt            ZARGB0
            GoTo        D2416REMOK
            rrncf       ZARGB0,W
            movwf       BARGB0
            Call        DGETNRMD

            mulwf       TEMPB2
            movff       PRODH,REMB1
            mulwf       TEMPB1
            movf        PRODL,W
            addwf       REMB1,F
            movff       PRODH,REMB0

D2416REMOK:
            clrf        AARGB0
            movff       ZARGB1,AARGB2
            movff       ZARGB2,AARGB1

            retlw       0x00

;-----------------------------------------------------
FXD2408U:
            movff       AARGB0,TEMPB0
            movff       AARGB1,TEMPB1
            movff       AARGB2,TEMPB2

            Call        FXD1608U

            movff       AARGB0,TEMPB0
            movff       AARGB1,TEMPB1

            movff       TEMPB2,AARGB1
            movff       REMB0,AARGB0

            Call        FXD1608U

            movff       AARGB1,AARGB2
            movff       TEMPB1,AARGB1
            movff       TEMPB0,AARGB0

            retlw       0x00

;-----------------------------------------------------
```

```
FXD1608U:

            movlw       0x01
            cpfsgt            BARGB0
            GoTo        DREMZERO8

FXD1608U1:

            bcf         _C
            clrf        TBLPTRH
            rlcf        BARGB0,W
            rlcf        TBLPTRH,F
            addlw       Low (IBXTBL256+1) ; access reciprocal table
            movwf       TBLPTRL
            movlw       High (IBXTBL256)
            addwfc            TBLPTRH,F
            TBLRD       *-

FXD1608U2:

            movff       AARGB0,REMB1
            movff       AARGB1,REMB0

            movf        TABLAT,W          ; estimate quotient
            mulwf       REMB1
            movff       PRODH,AARGB0
            movff       PRODL,AARGB1

            TBLRD       *+
            movf        TABLAT,W
            mulwf       REMB0
            movff       PRODH,AARGB2

            mulwf       REMB1
            movf        PRODL,W
            addwf       AARGB2,F
            movf        PRODH,W
            addwfc            AARGB1,F
            clrf        WREG
            addwfc            AARGB0,F

            TBLRD       *-
            movf        TABLAT,W
            mulwf       REMB0
            movf        PRODL,W
            addwf       AARGB2,F
            movf        PRODH,W
            addwfc            AARGB1,F
            clrf        WREG
            addwfc            AARGB0,F

            movf        BARGB0,W
```

```
            mulwf       AARGB1
            movff       PRODL,AARGB3
            movff       PRODH,AARGB2
            mulwf       AARGB0
            movf        PRODL,W
            addwf       AARGB2,F

            movf        AARGB3,W          ; estimate remainder
            subwf       REMB0,F
            movf        AARGB2,W
            subwfb            REMB1,F

            btfss       REMB1,MSB         ; test remainder
            retlw       0x00

            decf        AARGB1,F
            clrf        WREG
            subwfb            AARGB0,F

            movf        BARGB0,W
            addwf       REMB0,F

        retlw       0x00


;-----------------------------------------------------------
FXD1616U:
            tstfsz            BARGB0
            GoTo        D1616B0GT0
            movff       BARGB1,BARGB0
            Call        FXD1608U
            movff       REMB0,REMB1
            clrf        REMB0

            retlw       0x00

D1616B0GT0:
            movf        BARGB0,W
            subwf       AARGB0,W
            btfss       _C
            GoTo        D1616QZERO
            btfss       _Z
            GoTo        D1616AGEB

            movf        BARGB1,W
            subwf       AARGB1,W
            btfss       _C
            GoTo        D1616QZERO

D1616AGEB:
            movff       AARGB0,TEMPB0
            movff       AARGB1,TEMPB1

            movff       AARGB1,CARGB2
            movff       AARGB0,AARGB1
```

```
            clrf        AARGB0

            movff       BARGB0,BARGB2
            movff       BARGB1,BARGB3

            btfsc       BARGB0,MSB
            GoTo        D1616UNRMOK

            movf        BARGB0,W
            rlncf       WREG,F
            addlw       Low (IBXTBL256+3) ; access reciprocal table
            movwf       TBLPTRL
            movlw       High (IBXTBL256)
            clrf        TBLPTRH
            addwfc            TBLPTRH,F
            TBLRD       *

            movf        TABLAT,W          ; normalize
            mulwf       BARGB3
            movff       PRODL,BARGB1
            movff       PRODH,BARGB0
            mulwf       BARGB2
            movf        PRODL,W
            addwf       BARGB0,F

            movf        TABLAT,W
            mulwf       TEMPB1
            movff       PRODL,CARGB2
            movff       PRODH,AARGB1
            mulwf       TEMPB0
            movf        PRODL,W
            addwf       AARGB1,F
            clrf        AARGB0
            movf        PRODH,W
            addwfc            AARGB0,F

D1616UNRMOK:
            Call        FXD1608U1         ; estimate quotient digit

            movf        AARGB1,W
            mulwf       BARGB1

            movf        PRODL,W
            subwf       CARGB2,W
            movf        PRODH,W
            subwfb            REMB0,W

            btfss       _C                ; test
            decf        AARGB1,F

D1616UQOK:
            movf        AARGB1,W          ; calculate remainder
            mulwf       BARGB3
            movf        PRODL,W
            subwf       TEMPB1,F
```

```
            movf        PRODH,W
            subwfb            TEMPB0,F

            movf        AARGB1,W
            mulwf       BARGB2
            movf        PRODL,W
            subwf       TEMPB0,F

;     This test does not appear to be necessary in the 16 bit case, but
;     is included here in the event that a case appears after testing.

;           BTFSS       TEMPB0,MSB        ; test
;           GOTO        D1616QOK
;           DECF        AARGB1

;           MOVF        BARGB3,W
;           ADDWF       TEMPB1
;           MOVF        BARGB2,W
;           ADDWFC            TEMPB0

D1616QOK:
            movff       TEMPB0,REMB0
            movff       TEMPB1,REMB1

            retlw       0x00
;------------------------------------------------------------
DGETNRMD:
            movlw       0x10
            cpfslt            BARGB0
            GoTo        DGETNRMDH
DGETNRMDL:
            btfsc       BARGB0,3
            retlw       0x10
            btfsc       BARGB0,2
            retlw       0x20
            btfsc       BARGB0,1
            retlw       0x40
            btfsc       BARGB0,0
            retlw       0x80
DGETNRMDH:
            btfsc       BARGB0,6
            retlw       0x02
            btfsc       BARGB0,5
            retlw       0x04
            btfsc       BARGB0,4
            retlw       0x08

;------------------------------------------------------------------------
------------------------
;     Routines for the trivial cases when the quotient is zero.
;     Timing:     9,7,5 clks
;   PM: 9,7,5               DM: 8,6,4

;D3232QZERO
;           MOVFF       AARGB3,REMB3
```

```
;           CLRF        AARGB3

;D2424QZERO
;           MOVFF       AARGB2,REMB2
;           CLRF        AARGB2

D1616QZERO:
            movff       AARGB1,REMB1
            clrf        AARGB1
            movff       AARGB0,REMB0
            clrf        AARGB0
            retlw       0x00

DREMZERO8:
            clrf        REMB0
            retlw       0x00

               ;-----------------------------------------------------
----------------------------------------
;     The table IBXTBL256 is used by all routines and consists of 16-
bit
;     upper bound approximations to the reciprocal of BARGB0.
  IBXTBL256:


            Data  0x0000    ;00000000 00000000     0
            Data  0x0001    ;00000000 00000001     1
            Data  0x8001    ;10000000 00000001     32769
            Data  0x5556    ;01010101 01010110     21846
            Data  0x4001    ;01000000 00000001     16385
            Data  0x3334    ;00110011 00110100     13108
            Data  0x2AAB    ;00101010 10101011     10923
            Data  0x2493    ;00100100 10010011
            Data  0x2001    ;00000000 00000000
            Data  0x1C72    ;00011100 01110010
            Data  0x199A    ;00011001 10011010
            Data  0x1746    ;
            Data  0x1556    ;
            Data  0x13B2    ;
            Data  0x124A    ;
            Data  0x1112    ;
            Data  0x1001    ;
            Data  0x0F10    ;
            Data  0x0E39    ;
            Data  0x0D7A    ;
            Data  0x0CCD    ;
            Data  0x0C31    ;
            Data  0x0BA3    ;
            Data  0x0B22    ;
            Data  0x0AAB    ;
            Data  0x0A3E    ;
            Data  0x09D9    ;
            Data  0x097C    ;
            Data  0x0925
            Data  0x08D4
```

```
Data  0x0889
Data  0x0843
Data  0x0801
Data  0x07C2
Data  0x0788
Data  0x0751
Data  0x071D
Data  0x06EC
Data  0x06BD
Data  0x0691
Data  0x0667
Data  0x063F
Data  0x0619
Data  0x05F5
Data  0x05D2
Data  0x05B1
Data  0x0591
Data  0x0573
Data  0x0556
Data  0x053A
Data  0x051F
Data  0x0506
Data  0x04ED
Data  0x04D5
Data  0x04BE
Data  0x04A8
Data  0x0493
Data  0x047E
Data  0x046A
Data  0x0457
Data  0x0445
Data  0x0433
Data  0x0422
Data  0x0411
Data  0x0401
Data  0x03F1
Data  0x03E1
Data  0x03D3
Data  0x03C4
Data  0x03B6
Data  0x03A9
Data  0x039C
Data  0x038F
Data  0x0382
Data  0x0376
Data  0x036A
Data  0x035F
Data  0x0354
Data  0x0349
Data  0x033E
Data  0x0334
Data  0x032A
Data  0x0320
Data  0x0316
Data  0x030D
```

```
Data  0x0304
Data  0x02FB
Data  0x02F2
Data  0x02E9
Data  0x02E1
Data  0x02D9
Data  0x02D1
Data  0x02C9
Data  0x02C1
Data  0x02BA
Data  0x02B2
Data  0x02AB
Data  0x02A4
Data  0x029D
Data  0x0296
Data  0x0290
Data  0x0289
Data  0x0283
Data  0x027D
Data  0x0277
Data  0x0271
Data  0x026B
Data  0x0265
Data  0x025F
Data  0x025A
Data  0x0254
Data  0x024F
Data  0x024A
Data  0x0244
Data  0x023F
Data  0x023A
Data  0x0235
Data  0x0231
Data  0x022C
Data  0x0227
Data  0x0223
Data  0x021E
Data  0x021A
Data  0x0215
Data  0x0211
Data  0x020D
Data  0x0209
Data  0x0205
Data  0x0201
Data  0x01FD
Data  0x01F9
Data  0x01F5
Data  0x01F1
Data  0x01ED
Data  0x01EA
Data  0x01E6
Data  0x01E2
Data  0x01DF
Data  0x01DB
Data  0x01D8
```

```
Data  0x01D5
Data  0x01D1
Data  0x01CE
Data  0x01CB
Data  0x01C8
Data  0x01C4
Data  0x01C1
Data  0x01BE
Data  0x01BB
Data  0x01B8
Data  0x01B5
Data  0x01B3
Data  0x01B0
Data  0x01AD
Data  0x01AA
Data  0x01A7
Data  0x01A5
Data  0x01A2
Data  0x019F
Data  0x019D
Data  0x019A
Data  0x0198
Data  0x0195
Data  0x0193
Data  0x0190
Data  0x018E
Data  0x018B
Data  0x0189
Data  0x0187
Data  0x0184
Data  0x0182
Data  0x0180
Data  0x017E
Data  0x017B
Data  0x0179
Data  0x0177
Data  0x0175
Data  0x0173
Data  0x0171
Data  0x016F
Data  0x016D
Data  0x016B
Data  0x0169
Data  0x0167
Data  0x0165
Data  0x0163
Data  0x0161
Data  0x015F
Data  0x015D
Data  0x015B
Data  0x0159
Data  0x0158
Data  0x0156
Data  0x0154
Data  0x0152
```

```
Data  0x0151
Data  0x014F
Data  0x014D
Data  0x014B
Data  0x014A
Data  0x0148
Data  0x0147
Data  0x0145
Data  0x0143
Data  0x0142
Data  0x0140
Data  0x013F
Data  0x013D
Data  0x013C
Data  0x013A
Data  0x0139
Data  0x0137
Data  0x0136
Data  0x0134
Data  0x0133
Data  0x0131
Data  0x0130
Data  0x012F
Data  0x012D
Data  0x012C
Data  0x012A
Data  0x0129
Data  0x0128
Data  0x0126
Data  0x0125
Data  0x0124
Data  0x0122
Data  0x0121
Data  0x0120
Data  0x011F
Data  0x011D
Data  0x011C
Data  0x011B
Data  0x011A
Data  0x0119
Data  0x0117
Data  0x0116
Data  0x0115
Data  0x0114
Data  0x0113
Data  0x0112
Data  0x0110
Data  0x010F
Data  0x010E
Data  0x010D
Data  0x010C
Data  0x010B
Data  0x010A
Data  0x0109
Data  0x0108
```

```
            Data  0x0107
            Data  0x0106
            Data  0x0105
            Data  0x0104
            Data  0x0103
            Data  0x0102
            Data  0x0101
 ENDASM
;*********************************************************************
********
;Start of main program
; The main program code is placed here.

Main:
;*********************************************************************
*
; prepare for A/D conversion on PortA
        ADIN_RES  10                            ; Set the resolution to
10
        ADIN_TAD  FRC                           ; Choose the RC osc for
ADC samples
            ADIN_STIME  50                  ; Allow 50us for charge
time
      movlw %00000011                   ; set port direction (input->1,
output->0)
      movwf TRISA
    ADCON1 = %10000010                      ; Set PORTA analog and
right justify result
    TRISD.1 = 0
    TRISD.2 = 0
;*********************************************************************

      Call  PidInitalize
      Raw = 0
    clrf    channel
      bsf         pid_sign
      movlw %00000000                   ; set port direction (input->1,
output->0)
      movwf TRISB
loop:

    pidout.BYTE3 = %00000000        ; zero byte 4 of pidout it is only
3 bytes

    bcf channel_number
    Call GetSensorReading

    Value = 489 *(Raw / 10)               ; Quantasize the result
      Volts = Value / 10000
      Millivolts = (Value // 10000) / 100
    ; Print At 1,1,DEC1 Volts,".",DEC2 Millivolts,"V" ; Display the
result


    temp_word = Raw
```

```
    bsf channel_number
    Call GetSensorReading
    Value = 489 *(Raw / 10)              ; Quantasize the result
      Volts = Value / 10000
      Millivolts = (Value // 10000) / 100
    'Print At 1,1,DEC1 Volts,".",DEC2 Millivolts,"V" ; Display the
result
    BARGB0 = temp_word.BYTE1          ; set point value
    BARGB1 = temp_word.BYTE0

    AARGB0 = Raw.BYTE1     ; process output variable
    AARGB1 = Raw.BYTE0

    Call MagAndSub                ; find error by subtraction (aargb -
bargb ) and get the sign
    'Print At 1,1, BIN mag
    If mag=1 Then
    err_sign = 1
    High PORTD.1
    Else
    err_sign = 0
    Low PORTD.1
    EndIf

    ;Raw = Raw - temp_word
    error0 = AARGB0                 ; load error variable
    error1 = AARGB1
    'Print At 2,1,"               "

    'Print At 2,1,DEC3 AARGB0,".", DEC3 AARGB1
      Call  PidMain
    If pid_sign =1 Then
    sign_out ="+"
    High PORTD.2
    Else
    sign_out = "-"
    Low PORTD.2
    EndIf

    'Print At 1,1,@pidout
    AARGB0 = pidOut0
    AARGB1 = pidOut1
    AARGB2 = pidOut2

    'Call FXD2408U
    pidOut0 = AARGB0
    pidOut1 = AARGB1
    pidOut2 = AARGB2
    If pidOut0 <> 0 Then
    pidOut1 = 255
    ElseIf pidOut1 <> 0 Then
    AARGB2 = 255
    EndIf
    Print At 1,1, @pidOut2
```

```
    Print At 2,1,BIN8 pidOut2
    HPWM 1,AARGB2,2500

    PORTB = AARGB2
   ; delayus 1000
     GoTo   loop
;*****************************************************************
********
;End of program


      End
```

APPENDIX 5. M-FILE FOR CALCULATING CLAMPED BEAM FREQUENCY

```
E=210e9;

b=input('beam width= ');

h=input('beam thickness= ');

I=(b*h^3)/12;

ro=7715;

m=ro*b*h;

L=input('beam length= ');

freq=3.528*(sqrt(E*I/(m*L^4)))/2/pi
```